Grass-roots Class Alignment

Baoshi Yan
Information Sciences Institute, University of Southern California
4676 Admiralty Way, Marina del Rey, California 90292
baoshi@isi.edu

Abstract

Current ontology alignment practices adopt a centralized approach: the alignment task is carried out by a domain expert, possibly with the help of some ontology alignment tools. In a Peer-to-Peer environment, each end-user (peer) may explicitly or implicitly generate ontology alignments for their own purposes during its own use of the semantic data. This kind of end-user-generated ontology alignments, which we call grass-roots ontology alignments, could be mined and generalized to help other users with their alignment tasks, thus enabling a p2p-style ontology alignment. Grass-roots ontology alignment, often generated as a side effect of other data manipulations, could be peer-specific, task-specific, approximate, or even contradictory. Its semantics is not necessarily the equivalency between classes or properties. This paper reports our work on reusing grass-roots class alignment for aligning class hierarchies. A grass-roots class alignment, though approximate, still reveals some facts about relationships between different classes. We formalize facts about class relationships that can be inferred from an alignment under different cases. We then apply forward-chaining inference to the facts knowledge base to infer more facts. The facts KB is then leveraged for ontology alignment purposes. To deal with uncertainty and inconsistency, each fact is associated with an evidence that tells how the fact is obtained. The evidences are used to select better-supported facts in case of inconsistency.

1 Introduction

In a Peer-to-Peer knowledge management environment, each end user (peer) might have access to a different set of ontologies, and might explicitly or implicitly align ontologies for their own purposes. This kind of end-user-generated ontology alignment, which we call grass-roots ontology alignment, is a useful source of information that should help other peers with future alignment tasks. For example, when one user (implicitly) aligns "O1:PhDStudent"[1] with "O2:DoctoralStudent", it should help other users align these classes when they see ontologies O1 and O2. Furthermore, it should help us with our future alignment tasks when these terms appear in other ontologies again. That is, it should help us align "O3:PhDStudent" with "O4:DoctoralStudent".

However, not all grass-roots alignments are that universal. Grass-roots ontology alignment, often generated as a side effect of other data manipulations, could be user-specific, task-specific, approximate, or even contradictory. A university secretary, when counting the number of university personnel, may put together all "Professors", "Staffs" and "Students" from different sources, thus implying that "Professor", "Staff" and "Student" are aligned. These classes, however, cannot be aligned in many other cases. The semantics of grass-roots alignment is not necessarily that of class equivalency or property equivalency.

Since alignment is often generated implicitly, it could be erroneous when users perform mistaken data manipulations.

Approximateness of alignment often comes from the lack of corresponding concepts. For example, in one ontology there might be a "GraduateStudent" class, while in another one there might be only a "MasterStudent" class. Thus a person building a report of all graduate students will simply put all "GraduateStudent" and "MasterStudent" instances together, which will implicitly produce alignment that "GraduateStudent" is aligned with "MasterStudent".

Alignment is not transitive. If we regard alignment as transitive, in extreme cases everything would be aligned with everything else. As depicted in Figure 1 (Different color blocks represent different ontologies; arrows represent subclass relationships), every alignment (represented in dashed double line) is the best possible alignment between the two ontologies. All classes in Figure 1 will be aligned with each other if we regard alignment as transitive.

Therefore, to reuse grass-roots alignment for ontology alignment purposes, we must deal with the approximateness

[1] The notation of O:C defines a class with term C in ontology O, C is a meaningful string representing the class, such as a class label.

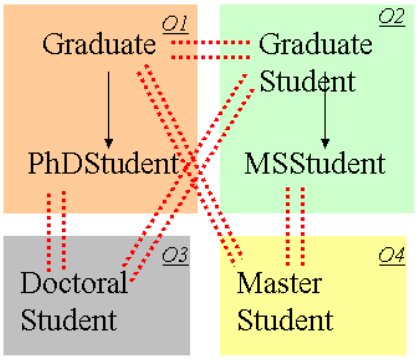

Figure 1. Alignment is not transitive

or even erroneousness of those alignments.

2 Objective: Aligning Classes with Grass-roots Alignment

In a Peer-to-Peer environment each peer might generate some ontology alignment, which, although approximate, is a useful source of information that should help other peers with their alignment tasks. In this paper we focus on the problem of aligning classes by reusing grass-roots class alignment. To be more precise, the information we have is a set of grass-roots alignments between the classes of a set of ontologies. Now given two ontologies in the same domain that might not be among the aforementioned set of ontologies, our objective is to align the classes of the two ontologies.

From now on we sometimes use alignment as shorthand for grass-roots alignment when it is not ambiguous.

3 Our Grass-roots Class Alignment Approach

As discussed above, reusing grass-roots class alignment for other alignment tasks is not straightforward. "GraduateStudent" and "MasterStudent" are aligned in one case does not mean they can be aligned in other cases. Furthermore, the non-transitivity of alignment limits our ability to reuse previous alignments. We cannot align A and C just because A and C are both aligned to B.

In the ideal world, there is a complete and correct class hierarchy. Class alignment then becomes easy. For every class in one ontology, its best alignment candidate in the second ontology can be determined based on their respective positions in the class hierarchy.

Unfortunately such a class hierarchy does not exist in most cases. It is even less probable to exist on the Semantic Web. Our goal, thus, is to construct an approximate class hierarchy based on grass-roots class alignment that may be approximate or inconsistent.

To make the task easier, note that we actually do not need to explicitly construct a single class hierarchy. All the uncertainties and inconsistencies may lead to a collection of candidate class hierarchies with different confidence measures. All we need is the relationships (e.g., superclass, subclass, sibling) between different classes. Given these relationships, it is still possible to determine how well two classes can be aligned.

Therefore, our goal is to obtain as many class relationships as possible from grass-roots alignment, and then use them for future class alignment tasks. To do that, we must look at the semantics of grass-roots alignments, that is, what they imply, which we know are not necessarily equivalencies.

3.1 Observations and Assumptions

Notations:

Before we proceed, let's first define some notations. Given two terms A and B, there are four kinds of relationships between them:

1. A is more general than B, i.e., B is the subclass of A, which can be represented as $A > B$. We also call this A is an ancestor of B and B is a descendant of A.

2. B is more general than A, represented as $B > A$.

3. A and B are parallel ($A \parallel B$), that is, neither of A and B is more general than the other one but they are related via class subsumptions. For example, A and B are siblings.

4. A and B are not related via class subsumptions, e.g., a "Movie" and a "Person".

For the sake of brevity, we use $A \bullet B$ to represent the situation when there is ancestor/descendant relationship between A and B but we are not sure which one is more general.

We use $A * B$ to represent that A and B are related (either via $>$, $\bullet$ or $\parallel$).

Although grass-roots alignment seems rather arbitrary because users might align two classes for their own purposes, we can still make some observations regarding grass-roots alignments.

Observation 1:

Our first observation is that: when users align two classes, the aligned classes tend to describe same "kind" of things. For example, users might align "MasterStudent" with "Student", or even "MasterStudent" with "Person". But they rarely align "MasterStudent" with "Project". In other words, aligned classes are highly likely related via a series of class subsumption relationships.

Observation 2:

Our second observation is that when facing several approximate alignment candidates, users tend to select the one

that is semantically closer.

Take Figure 2(a) for example, we all know that $Student > GraduateStudent > MasterStudent$. If $Student > GraduateStudent$ appears in one ontology O1 while $MasterStudent$ appears in another ontology O2, out of $Student$ and $GraduateStudent$ users tend to pick $GraduateStudent$ to align $MasterStudent$ with. Similar observation can be made on the case depicted in Figure 2(b).

Figure 2(c) suggests another situation. For example, we know that $MasterStudent$ and $PhDStudent$ are both subclasses of $GraduateStudent$. When $GraduateStudent > MasterStudent$ appear in ontology O1 and $PhDStudent$ appears in ontology O2, users tend to align $O2 : PhDStudent$ with $O1 : GraduateStudent$. This is reasonable because B and C are not directly related. From the perspective of set theory, $A \cap C$ contains $B \cap C$. $B \cap C$ is $\emptyset$ when B and C are mutually exclusive.

Figure 2(d) is actually a rare case compared with case (a) (b) and (c). It basically says that given $O1 : Student > O1 : FemaleStudent$ and $O2 : Female$ users tend to align $O2 : Female$ with $O1 : FemaleStudent$.

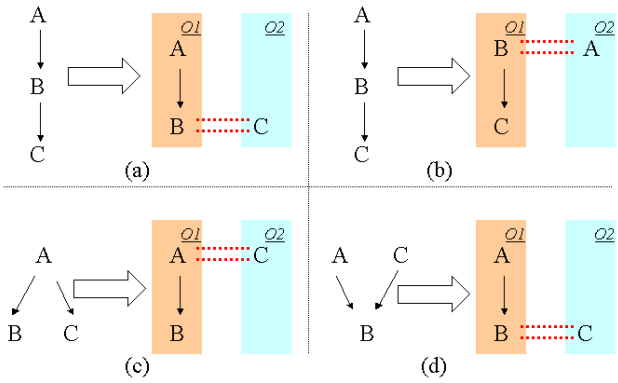

Figure 2. Observations on Grass-roots Alignment

The assumption our approach based on is the validity of above observations. We assume that most users tend to align according to our observations. Note that in Figure 2 class hierarchies are on the left side and alignments are on the right side. Given the validity of the observations, the right side (alignments) gives hints on how the left side should look like. Such hints are then combined to infer the class hierarchies on the left side.

3.2 The Class Alignment Algorithm

The first step of our class alignment algorithm is to get an initial set of facts about relationships between different classes. Such facts come from two sources: the subclass relationships explicitly specified in the original ontologies, and other facts implied by alignments.

Step 1: Subclass Relationships Specified in the Ontology:

Given an ontology O1, if it is specified in O1 that O1:B is a subclass of O1:A, we represent such a fact in a form: $(A > B, e = O1)$, which means that term A is more general than term B, the evidence for this claim is ontology O1. Similarly, if in ontology O2 it is also specified that O2:B is subclass of O2:A, then we get $(A > B, e = O1 + O2)$.

Step 2: Subclass Relationships Implied by Grass-roots Alignments: the Semantics of Grass-roots Alignments:

For simplicity reason, let's assume here that there is no multiple inheritance in the class hierarchies we are working on. Our experiences with many class hierarchies suggest that multiple inheritance occurs relatively infrequently. Thus we first present our algorithm for the case of single-inheritance class hierarchy. We deal with the multiple-inheritance situation later on.

Case 1: Suppose an alignment exists as on the left side of Figure 3 (the double dash line stands for alignment), that is, A is a superclass of B and B is aligned with C. Such an alignment implies that C cannot be a superclass of A, otherwise C should be aligned with A, not B. Thus the possible relationship between A and C is $A > C$ or $A \parallel C$. Similarly the alignment implies that B and C are not parallel, otherwise C is better aligned with A. Thus the possible relationship between B and C is $C > B$ or $B > C$. Take all A, B and C into account, there are four kinds of combination:

1. $A \parallel C$ and $B > C$: this combination is invalid because $B > C$ combined with $A > B$ will lead to $A > C$, in contradiction with $A \parallel C$.
2. $A \parallel C$ and $C > B$: in this case B inherits from both A and C, thus it is pruned because of our single-inheritance assumption.
3. $A > C$ and $C > B$, or
4. $A > C$ and $B > C$.

We can combine 3 and 4 and $A > B$ to get $(A > B$ AND $A > C$ AND $B \bullet C)$. For brevity we can also rewrite it in a shorter form $A > B \bullet C$.

Add in the evidence we will get $(A > B \bullet C, e = align1)$ where align1 is the alignment that links the O1:B class in the first ontology to O2:C class in the second ontology.

Case 2: Suppose an alignment exists as on the left side of Figure 4, that is, A is a superclass of B and A is aligned with C. Such an alignment implies that B cannot be a superclass of C, otherwise C should be aligned with B, not A. Further analysis will lead to (NOT $(B > C)$ AND $A * C$). The relationship between A and C can be arbitrary.

Similar to case 1, add in the alignment as evidence we get (NOT $(B > C)$ AND $A * C$, $e = align2$) where align2 is

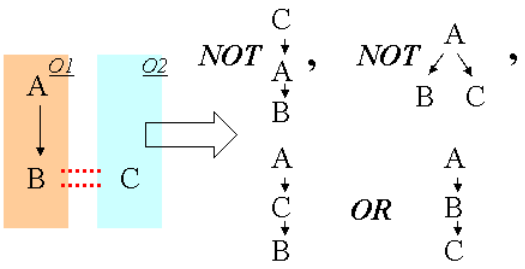

Figure 3. Case 1

the alignment that links the O1:A class in the first ontology to O2:C class in the second ontology.

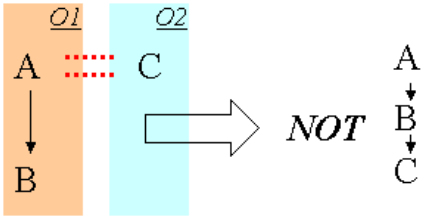

Figure 4. Case 2

Case 3: This is actually not a separate case (Figure 5). It is an aggregation of two case 1's. Therefore, we have:

$(A > B$ AND $B \sim C) => A > B \bullet C$

$(D > C$ AND $B \sim C) => D > B \bullet C$

where $\sim$ stands for alignment.

Since we are dealing with single-inheritance case, the last two can be combined into $A \bullet D > B \bullet C$.

Add in the algnment evidence align3 we will have $(A \bullet D > B \bullet C, e = align3)$. Similarly, for the case $(A > B$ AND $D > C$ AND $A \sim D)$ we will get (NOT $(B > D)$ AND NOT $(C > A)$ AND $A * D$).

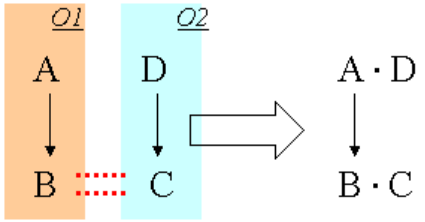

Figure 5. Case 3

Step 3: Forward-chaining Inference:

After we get all the facts from the subclass relationships in the ontologies and from the alignments, we apply forward-chaining inference to the facts knowledge base to obtain more facts. The inference rules used here are propositional rules. Some sample rules include:

$(A > B$ OR $A < B)$ and NOT$(A > B)$ will lead to $B > A$ (Unit Resolution).

$A > B$ and $B > C$ will lead to $A > C$ (Transitivity of Class Subsumption).

NOT$(A > B)$ and NOT$(A \parallel B)$ will lead to $B > A$.

The computation of evidence is as following:

When a new fact f is obtained from several other facts, $(f1, e1)AND(f2, e2)...AND(fi, ei) => (f, e)$, its evidence $e = e1 * e2 * .. * ei$. When a fact can be obtained several times with different evidences $e1, e2, ..ei$, its evidence is updated as $e = e1 + e2 + ... + ei$. Also note that same evidence doesn't count twice, that is, $e1 + e1 = e1, e1 * e1 = e1$.

We do not use backward-chaining inference. The facts knowledge base is very likely to be inconsistent because users may assert contradictory alignments. Thus everything you want the KB to prove will be proven.

With the forward-chaining inference, we try to infer as many facts as possible along with evidences for each fact. The evidences will be used to pick out better-supported facts in case of fact contradictions.

Quantifying Evidences: We want to quantify evidences for comparison reason. The value of an evidence is a numerical value between (0, 1). Let V(e) be the numerical value of evidence e. The computation of evidence value is according to the following:

V(e1+e2) = 1-(1-V(e1))*(1-V(e2))

V(e1*e2) = V(e1)*V(e2)

The value of primitive evidences can be determined separately, for example, they can be based on user authority or ontology quality. In our experiment, for simplicity we assign each ontology evidence a value of 0.6 and each alignment evidence a value of 0.3.

Note that evidence values are not probabilities. A fact with evidence value 0.8 does not mean it has a probability of 0.8 to be true. It's rather a measure of confidence which is only meaningful for comparison purposes.

Step 4: Class Alignment Using Facts KB:

The facts obtained from the inference step above will be used for the next class alignment task. Given a class A from ontology O1, we try to find a class B from the second ontology O2 such that B is the best alignment candidate for A.

Note that one desirable side effect of our algorithm is that it takes only a one-step query to get all superclasses or subclasses of a class because of the application of subsumption transitivity rule in Step 3.

Let's define three class sets as following:

Sup(A) is the set of superclasses of A as in facts KB.

Sub(A) is the set of subclasses of A as in facts KB.

Ind(A) is the set of all classes A' such that $(A > A'$ OR $A' > A)$ but there is no fact in KB specifying either

$A > A'$ or $A' > A$. That is, A' and A are indistinguishable according to facts KB.

To deal with possible inconsistencies, for each A' from Sup(A), if there is a better-supported fact $A > A'$, NOT$(A' > A)$ or $A' \parallel A$, remove A' from Sup(A). Do the same to Sub(A).

We then determine the alignment candidate for class A in the following order:

If there are one or more classes from O2 that belong to Ind(A), choose the best-supported one as the alignment candidate for A.

If there are one or more classes from O2 that belong to Sup(A), choose the one closest to A, that is, out of B and C choose B if $C > B$. If the order between B and C cannot be determined, pick the better-supported one.

If there is one or more classes from O2 that belong to Sub(A), choose the one closest to A, that is, out of B and C choose B if $B > C$. If the order between B and C cannot be determined, pick the better-supported one.

Otherwise there is no alignment candidate for A in O2.

Example: Let's use the ontologies and alignments in Figure 6 as an example to illustrate how to obtain facts about class relationships. Using the obtained facts for aligning classes is straightforward and thus not elaborated here. All the obtained facts are listed in Table 1. In order to (implicitly) integrate the three small class hierarchies into a bigger one, it is useful to determine the relationship between "Graduate" class and "UnivStudent" class. Note that since alignment does not mean equivalence, $(Student > Graduate, O2)$ and the alignment between "UnivStudent" and "Student" do not immediately imply $UnivStudent > Graduate$. However, we will show that by combining facts obtained from different alignments and ontologies , we will still be able to infer $UnivStudent > Graduate$. As listed in Table 1, Facts 0 to 5 are directly obtained from respective ontologies, Facts 6 to 13 are obtained from the two alignments, the rest of facts are obtained with forward-chaining inference. From align2 we get Fact 9: NOT$(Graduate > UnivStudent)$, which is Case 2 as depicted in Figure 4. From align1 we get Fact 8: $(Graduate > UnivStudent$ OR $UnivStudent > Graduate)$, which is case 3 as depicted in Figure 5. Applying unit resolution on Facts 8 and 9 results in $UnivStudent > Graduate$.

Dealing With Multiple Inheritance:

We only need to make small adjustments to our algorithm when dealing with multiple-inheritance class hierarchies. For the case 1 scenario as depicted in Figure 3, the following facts are implied from the alignment instead: NOT$(C > A)$, NOT$(B \parallel C)$, $((C > B$ AND $A \parallel C)$ OR $(A > C$ AND $(B > C$ OR $C > B)))$. The case 2 scenario remains the same while the case 3 is not used. In the multiple-inheritance case we introduce more uncertainties than in the single-inheritance case. Nevertheless, we

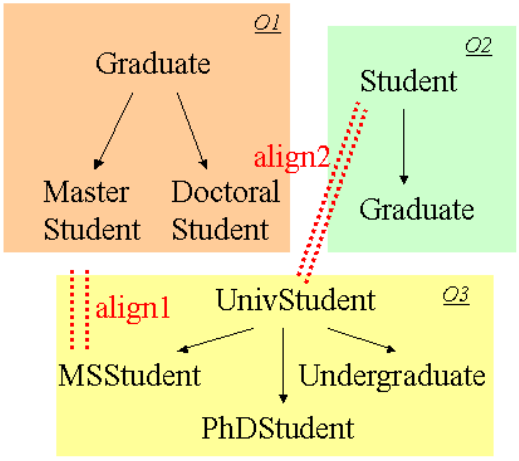

Figure 6. Example 1

Fact#	Fact	Evidence
0	Graduate > MasterStudent	O1
1	Graduate > DoctoralStudent	O1
2	Student > Graduate	O2
3	UnivStudent > Undergraduate	O2
4	UnivStudent > MSStudent	O3
5	UnivStudent > PhDStudent	O3
6	UnivStudent > MasterStudent	align1
7	Graduate > MSStudent	align1
8	Graduate > UnivStudent OR UnivStudent > Graduate	align1
9	NOT (Graduate>UnivStudent)	align2
10	Student * UnivStudent	align2
11	NOT (MSStudent>Student)	align2
12	NOT (PhDStudent>Student)	align2
13	NOT (Undergraduate>Student)	align2
14	UnivStudent > Graduate	align1*align2 /*from 8 and 9*/
15	Student > UnivStudent OR UnivStudent > Student	O2*align1*align2, /*from 2 and 14, single-inheritance*/
...	...	...

Table 1. Facts Knowledge Base

are still able to infer useful facts. Take Figure 7 for example, from align1 we get NOT$(MSStudent > Graduate)$, from align2 we get NOT$(MSStudent \parallel Graduate)$. Together they result in $Graduate > MSStudent$.

4 Evaluation

To evaluate the feasibility of our approach, we performed an experiment in the university student domain. We retrieved and downloaded 26 ontologies about university stu-

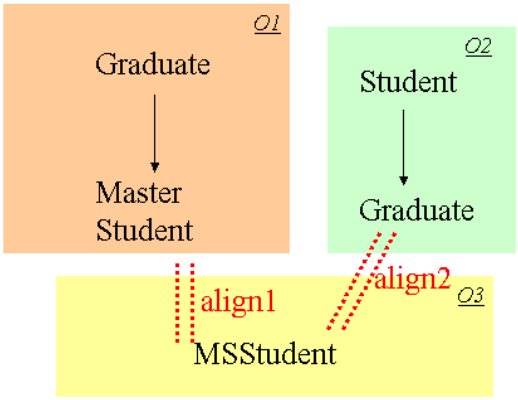

Figure 7. Example 2

dents with the help of swoogle[1] ontology search engine. The part of class hierarchy related to university student in each ontology consists of 5 classes on average. We found a high redundancy in class names, with about 3 different names for each class. As a reference set, we constructed a complete class hierarchy manually covering all related classes in the 26 ontologies. We then measure the precision and recall of inferred facts against the reference set. As shown in Figure 8, grass-roots alignments increase the recall with a good precision. The single-inheritance case has a higher recall than the multiple-inheritance case with roughly the same precision. This is understandable because the class hierarchies in the experiment are all single-inheritance hierarchies.

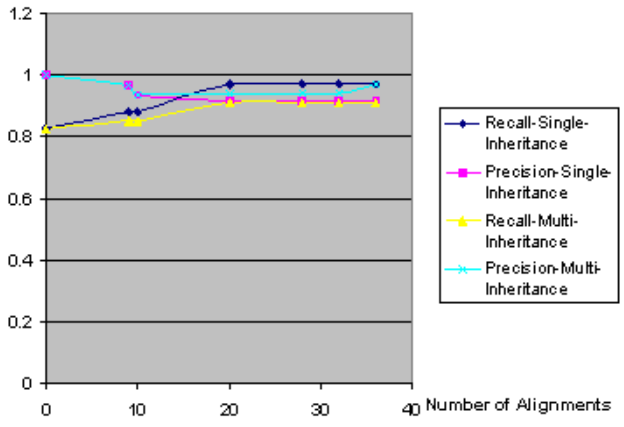

Figure 8. Precision and Recall of Obtained Facts

5 Survey of Current Alignment Tools and Techniques

Ontology alignment has been studied by many researchers in ontology and semantic web community. It has also been studied extensively in database community under the name of schema mapping. It is often studied under different names such as schema mediation, schema reconciliation, schema matching, semantic coordination, semantic mapping, and ontology mapping. An excellent survey on schema matching was given in [18].

In most research work, ontology alignment is defined as this problem: given two separately conceived ontologies, find likely matches between terms of the two ontologies.

ONION [16] and PROMPT [17] [15] use name similarity and structure similarity between ontologies as hints to guess matching between terms. Later on, people realized that data instances are an import source of information as well, and proposed alignment techniques (LSD [5] GLUE[6] Automatch [3] SemInt[10]) that take data instances into account. Different techniques are good at dealing with different kinds of information. To further improve matching performance, systems that integrate different matching techniques were also proposed (CUPID [13],LSD [5], COMA [4]).

The idea of reusing previous alignments was stated in [18] and further developed in COMA[4]. In order to match $S1$ and $S2$, COMA[4] requires the existence of S that has been already matched with $S1$ and $S2$, which makes it unusable for schemas unseen before. Alon Halevy [7] [11] proposed to use a corpus of schemas and schema mappings to help schema matching. [12] defines a representation for schema mapping and its associated semantics. But they didn't deal with the approximateness or inconsistency of the reused alignments.

Unlike other ontology alignment techniques that take two ontologies as input, [8] [9] takes a set of ontologies as input and tries to align them with holistic approaches. Their approaches are based on the mutual exclusiveness of equivalent terms to appear in the same ontology.

Ontology alignment is unlikely to be fully automatic. User interaction is inevitable. Aslan et.al.[2] proposed an iterative process for resolving semantic heterogeneity in federated databases by leverage user input for stepwise semantic evolution. Yan et.al.[19] proposed an interactive tool that uses instance data to guide users in data navigation and schema mapping.

The MOBS Project[14] tries to minimize the schema mapping effort of the data integration system builder by explicitly asking users of the system to answer schema mapping questions. The questions may not be of interest to users; they are rather users' "payment" for using other services provided by the system. This is very different from

our approach where end users generate alignments on ontologies of their own interests for their own purposes.

Most ontology alignment techniques take a semi-automated approach to ontology interoperation: the system guesses likely matches between terms of two separately conceived ontologies, a human expert knowledgeable about the semantics of both ontologies then verifies the inferences, possibly using a graphical user interface.

One big difference between our alignment algorithm and others is the information used by the algorithm. The grass-roots alignments, which are generated explicitly or implicitly by end users and could be approximate or erroneous, are not taken into account by other alignment algorithms.

6 Summary

End-user-generated ontology alignment, which we call grass-roots ontology alignment, might be approximate or erroneous. We discussed our work on dealing with the approximateness and inconsistencies of grass-roots class alignment in order to reuse them for class alignment purposes. Our preliminary results show a high precision and promising recall of our algorithm. Our work can be directly applied to Peer-to-Peer settings where each peer aligns classes for its own purposes and their work as a whole helps each other achieve high-accuracy class alignments.

7 Acknowledgments

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

We also gratefully acknowledge DARPA DAML program funding for WebScripter under contract number F30602-00-2-0576.

References

[1] Swoogle. http://swoogle.umbc.edu/.

[2] G. Aslan and D. McLeod. Semantic Heterogeneity Resolution in Federated Databases by Metadata Implantation and Stepwise Evolution. *VLDB Journal*, 8(2):120–132, 1999.

[3] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pages 452–466. Springer-Verlag, 2002.

[4] H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[5] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[6] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *The Eleventh International World Wide Web Conference*, 2002.

[7] A. Halevy, O. E. A. Doan, Z. Ives, and J. Madhavan. Crossing the structure chasm. In *the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.

[8] B. He and K. C.-C. Chang. A Holistic Paradigm for Schema Matching. *SIGMOD Record*, 33(3):120–132, September), year = 2004.

[9] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 217–228. ACM Press, 2003.

[10] W.-S. Li and C. Clifton. Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowl. Eng.*, 33(1):49–84, 2000.

[11] J. Madhavan, P. A. Bernstein, A. Doan, and A. Y. Halevy. Corpus-based schema matching. In *ICDE*, 2005.

[12] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. In *Eighteenth national conference on Artificial intelligence*, pages 80–86. American Association for Artificial Intelligence, 2002.

[13] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[14] R. McCann, A. Kramnik, W. Shen, V. Varadarajan, O. Sobulo, and A. Doan. Integrating data from disparate sources: A mass collaboration approach. In *ICDE*, pages 487–488, 2005.

[15] S. Melnik, H. Molina-Garcia, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *the International Conference on Data Engineering (ICDE)*, 2002.

[16] P. Mitra and G. Wiederhold. An algebra for semantic interoperability of information sources. In *2nd Annual IEEE International Symposium on Bioinformatics and Bioengineering*, pages 174–82, Bethesda, MD, USA, November 4-6 2001.

[17] N. F. Noy and M. A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on AI*, 2000.

[18] E. Rahm and P. Bernstein. On matching schemas automatically. Technical report, Microsoft Research, Redmon, WA, 2001. MSR-TR-2001-17.

[19] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):485–496, 2001.