OMEN: A Probabilistic Ontology Mapping Tool

Prasenjit Mitra[1], Natalya F. Noy[2], Anuj R. Jaiswal[1]

[1] The Pennsylvania State University, University Park, PA 16802, U.S.A.,
{pmitra,ajaiswal}@ist.psu.edu
[2] Stanford University, Stanford, CA 94305, U.S.A.,
noy@smi.stanford.edu

Abstract. Most existing ontology mapping tools do not provide exact mappings. Rather, there is usually some degree of uncertainty. We describe a framework to improve existing ontology mappings using a Bayesian Network. Omen, an Ontology Mapping ENhancer uses a set of meta-rules that capture the influence of the ontology structure and the semantics of ontology relations and matches nodes that are neighbors of already matched nodes in the two ontologies. We have implemented a protype ontology matcher that can enhance existing matches between ontology concepts. Preliminary experiments demonstrate that Omen successfully identifies and enhances ontology mappings.

1 Introduction

Information sources, even those from the same domain, are heterogeneous in nature. The semantics of the information in one source differs from that in another. In order to enable interoperation among heterogeneous information sources or to compose information from multiple sources, we often need to establish mappings between database schemas or between ontologies. These mappings capture the semantic correspondence between concepts in schemas or ontologies.

In recent years, researchers have developed a number of tools for finding these mappings in a semi-automated fashion (see Section 7 for a brief overview). In addition, there are interactive tools that enable experts to specify the mappings themselves. However, in most cases, the mappings produced are imprecise. For instance, automatic ontology-mapping tools can rank possible matches, with the ones that are more likely to be correct getting higher rankings. Most automatic ontology-mapping tools use heuristics or machine-learning techniques, which are imprecise by their very nature. Even experts sometimes could be unsure about the exact match between concepts and typically assign some certainty rating to a match. Once a particular set of mappings is established (by an expert or a tool), we can analyze the structure of ontologies in the neighborhood of these mappings to produce additional mappings.

Our main premise in this work is the following: if we know a mapping between two concepts from the source ontologies (i.e., they *match*), we can use the mapping to infer mappings between related concepts. For example, if two properties and their domains match, then we can infer (with some certainty) that their ranges may be related as well. We build a Bayesian Net with the concept mappings. The Bayesian Net uses a set of *meta-rules* based on the semantics of the ontology relations that expresses how each mapping affects other related mappings. We can use existing automatic and semi-automatic tools to come up with initial probability distributions for mappings. Next, we use this probability distribution to infer probability distributions for other mappings.

We have implemented a tool called OMEN (Ontology Mapping ENhancer). OMEN uses a Bayesian Net and enhances existing ontology mappings by deriving missed matches and invalidating existing false matches. Our preliminary results show that by using OMEN we can enhance the quality of existing mappings between concepts across ontologies.

The primary contributions of this paper are as follows:

1. We introduce a probabilistic method of enhancing existing ontology mappings by using a Bayesian Net to represent the influences between potential concept mappings across ontologies.
2. In OMEN, we provide an implemented framework where domain knowledge of mapping influences can be input easily using simple meta-rules.
3. We demonstrate the effectiveness of OMEN in our preliminary experiments.

To the best of our knowledge, no existing work has extensively used a probabilistic representation of ontology mapping rules and probabilistic inference to improve the quality of existing ontology mappings.

2 Knowledge Model

We assume a simple ontology model (similar to RDF Schema). We use the following components to express ontologies:

Classes Classes are concepts in a domain, organized in a subclass–superclass hierarchy with multiple inheritance.

Properties Properties describe attributes of classes and relationships between classes. Properties have one or more **domains**, which are classes to which the property can be applied; and one or more **ranges**, which restrict the classes for the values of property.

We use the following notation conventions through the rest of this paper:

- all concepts from O have no prime ('); all concepts from O' have a prime (');
- upper-case C with or without a subscript is a class;
- lower-case q with or without a subscript is a property;
- $P(C_1\ \theta\ C_2, x)$ indicates that the probability of the match $(C_1\ \theta\ C_2)$ is x.

3 Construction of the Bayesian Net

We now discuss how the Bayesian Net is constructed.

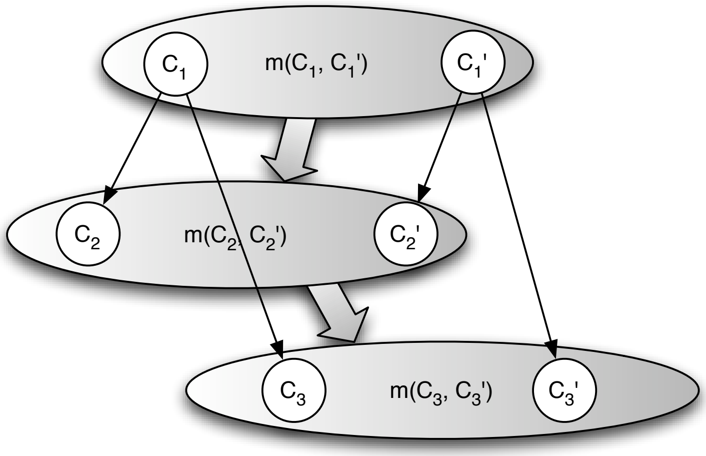

Fig. 1. Subgraphs representing some concepts in ontologies O and O' (small circles) and relations between them (thin arrows). The large gray ovals and solid arrows represent a snippet of the BN graph with nodes corresponding to matches and arrows corresponding to influences in the BN graph.

3.1 The BN-Graph

Nodes in the BN-graph represent individual pairs of matches. Consider Figure 1. This figure represents some classes in ontology O in the left-hand tree and some classes in ontology O' in the right-hand tree. The thin arrows in the figure are relationships between classes in the ontology, such as subclass–superclass relationships. The gray nodes and arrows represent the BN graph superimposed on the graphs representing ontologies. Nodes in the BN graph are matches between pairs of classes or properties from different ontologies. Arrows in the BN graph (the solid gray arrows in Figure 1) represent the influences between the nodes in the BN graph. The Conditional Probability Tables (CPTs) represent how a probability distribution in one node in the BN graphs affects the probability distribution in another node downstream from it. For example, in Figure 1, the mapping between concepts C_1 and C_1' affects the mapping between concepts C_2 and C_2', which in turn affects the mapping between C_3 and C_3'

A Scalable Selection of Nodes If we create a node for all possible pairings of concepts in two ontologies, the number of nodes in the BN-graph grows quadratically with respect to the number of nodes in the source ontologies. For example, if the sizes of the ontologies are 100 nodes each, the BN-graph will have 10,000 nodes. However, most of these nodes will express matches that are extremely unlikely to hold, because an evidence node will not influence a node that is distant from an evidence node significantly. Therefore, for performance, reasons, it makes sense to prune the BN-graph. We generate all possible nodes in the BN-graph that are at a maximum distance of k from an evidence node. The value of k is tuneable by the expert running the system, but empirically, we found a small value like $k = 1$ or $k = 2$ suffices. Larger values of k make very little difference to the result but increase the size of the Bayesian Net significantly.

Anther factor that effects the size of the BN-graph is the number of parents (i.e., nodes that influence the match) that each node has. For example, if a concept C has 5 parents, and C' has 8 parents, the node representing a match between C and C' would have 40 parent nodes in the BN-graph. As we discuss in the next section, the size of a CPT is exponential with respect to the number of parents of a node. Therefore, generating the CPT would cost $2^4 0$ units of computation. Even if the computation is very small, this number is exceedingly large and very soon makes the Bayesian Net unweildy. Thus, we restrict the maximum number of parent nodes for a single node to 10. We choose these 10 parents by selecting the top 5 parents with the maximum *a priori* probability and the top 5 parents with the minimum *a priori* probability.

If the Bayesian Net is constructed by adding edges such that matching ancestor nodes in an ontology influence the children nodes, we refer to this method as the "down-flow" method. A method where the Bayesian Net edges are constructe such that matching descendant nodes influence their ancestors, we call the method a "top-down" method. In case the ontologies contain cycles and this introduces cycles in the BN-graph, the algorithm breaks cycles in the BN-graph by rejecting the edges from the parents whose matching information is minimum (confidence score near 0.5).

3.2 Evidence and CPTs

In order to run a Bayesian Net we need to provide it with two types of information: (1) evidence (obtained from the initial probabilities) describing what we already know with high confidence, and (2) Conditional Probability Tables, describing how the parent nodes influence the children in the BN-graph.

The input to the OMEN algorithm consists not only of the two source ontologies to be matched, but also, of the initial probability distributions on the "root" nodes (nodes with no parents) in the BN graph. Note that our definition of mapping allows for inputs that are themselves imprecise and contain some probability values. For instance, if there is an ontology matcher that produces a set of pairs of matches ordered according to the algorithm's certainty about the match we can translate that into specific values for each $m(C_n, C'_k)$ where the probability value for "=" is less than 1 and diminishes as we go further down in the ranked list of the external matcher's result.

The final missing piece are the CPTs. The CPTs describe how a match between two classes affects other matches (these are the solid gray arrows in Figure 1). For example, a match between two classes from the source ontologies affects the match between their superclasses. Or a match between properties affects the match between their domains. These rules depend on the knowledge model and semantics of the relationships (such as subclass or domain) defined in the knowledge model. Therefore, we have developed a set of generic *meta-rules* that enable us to generate CPTs for each particular pair of ontologies automatically. In fact, our implementation is parameterized with respect to the meta-rules, and we can add or remove meta-rules to evaluate which ones work best for a particular knowledge model.

We present some of the meta-rules that we used in the next Section.

The following summarizes the OMEN algorithm:

- Input: source ontologies O and O', initial probability distribution for matches
- Steps:
  1. If initial probability of a match is above a given threshold, create a node representing the match and mark it as evidence node.
  2. Create nodes in the BN graph representing each pair of concepts (C, C') , such that $C \in O$ and $C' \in O'$ as a node in the graph and the nodes are within a distance k of an evidence node
  3. Create edges between the added nodes
  4. Use the meta-rules to generate CPTs for the BN
  5. Run the BN
- Output: a new set of matches

In the next section we discuss how Step 1 above can be modified to prune out unnecessary nodes.

4 Meta-rules for Generating New Probability Distributions

In this section, we show examples of meta-rules that are used to match the ontologies and discuss how the algorithm generates new probability distributions depending upon the existing ones.

4.1 Examples of Meta-rules

The following is one of the basic meta-rules we used in our implementation: if two concepts C_1 and C'_1 match, and there is a relationship q between C_1 and C_2 in O and a matching relationship q' between C'_1 and C'_2 in O', then we can increase the probability of match between C_2 and C'_2. Informally, if two nodes in an ontology graph match and so do two arrows coming out of these nodes, then the probability that nodes at the other end of the arrows match as well is increased. In the formal rule below we generalize this meta-rule to any relationship θ between C_1 and C'_1, not just match.

$$P(C_1\ \theta\ C'_1, x) \wedge P(q = q', 1) \wedge q(C_1, C_2) \wedge q'(C'_1, C'_2) \Rightarrow P(C_2\ \theta\ C'_2, min(1, x + \delta) \tag{1}$$

where δ is an expert-provided constant less than 1. We use a similar meta-rule for the case where relationships q and q' do not match (i.e., for arbitrary pair of outgoing edges), but subtract *delta* from x in the consequent.

While not in our initial implementation, other meta-rules rely more heavily on the semantics of the components in the ontology language. Below are some informal examples of such rules.

Mappings between properties and ranges of properties: If two properties match, and each of them has a single range, we can increase the probability of match between the classes representing the range. Similarly if two properties q and q' match and the range of q is a union of classes C_1 and C_2, and the range of q' is a class C', then the tool can increase the probability that C_1 is a specialization of C' and C_2 is a specialization of C'.

Mappings between superclasses and all but one sibling: In this case, we say that the existing matches between the superclasses and the matched siblings result in the remaining siblings matching with high probability.

We experimented with three different ways of generating the CPTs for the nodes in a BN graph:

1. Fixed Influence Method (FI): The meta-rules state that the probability of the children matching depends upon whether the parents match and is given by a set of constants. An example of such a rule is:
$$P[C_p = C'_p, x] \wedge x > t_{max} \wedge q(C_p, C_c) \wedge q(C'_p, C'_c) \Rightarrow P[C_c = C'_c, 0.9] \quad (2)$$
where t_{max} is an expert-defined threshold value. There are similar rules for the other cases.
2. Initial Probability Method (AP): The meta-rules state that the probability distribution of a child node is affected depending upon the probability distribution of the parent node by a set of constants. An example of this class of meta-rules is:
$$P(C_1\ \theta\ C'_1, x) \wedge P(q = q', 1) \wedge q(C_2, C_1) \wedge q'(C'_2, C'_1) \wedge P[C_2\ \theta C'_2, y] \wedge (y > t_{max})$$
$$\Rightarrow P(C_1\ \theta\ C'_1, min(1, (x + \delta))$$
where t_{max} and δ are expert-provided constants less than 1.
3. Parent Probability Method (PP): The meta-rules state that the probability distribution of the child node is derived from the probability distribution of the parent node using a set of constants. Rule 1 is an example of a meta-rule used in this method.

The algorithm must combine probabilitic influences of different rules and determine the probability distribution of a mappings. For example, consider a pair of classes, C and C' (Figure 2). In the example in the figure, the following mappings can affect the probability that they match (depending on a specific set of meta-rules used):

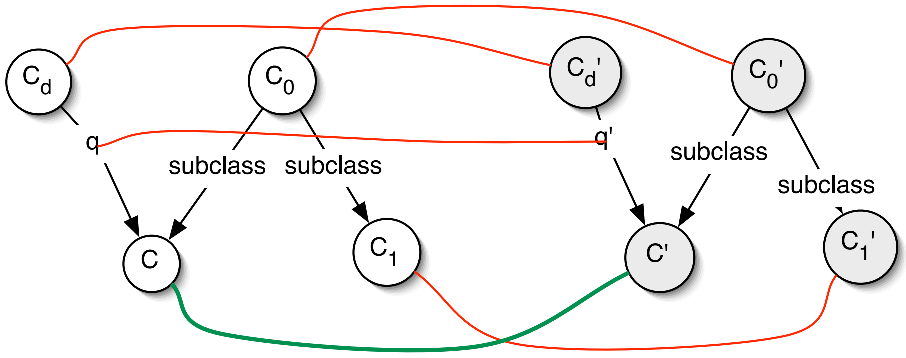

Fig. 2. The probability distribution for the mapping between C and C' is affected by the mappings between their superclasses, siblings, and domains of the properties q and q' for which C and C' are ranges.

– A mapping between superclasses of C and C'
– Mappings between the siblings of C and C'
– A mapping between properties q and q' ($P(q = q', 1)$) for which C and C' are ranges respectively, and mappings between domains of q and q' ($P(C_d = C'_d, z)$).

In this work, we combine probabilitistic influences as follows. If a child in the Bayesian Net (not the ontologies) has two parents, we combine the conditional probability distributions of the child on each parent using the assumption that the two parents are independent. That is,

$$P[C|A, B] = P[C|A] \times P[C|B] \tag{3}$$

In cases, where the match of two parents influence each other, this assumption is not true. However, empirically, even with this simplifying assumption we have obtained encouraging results. A more sophisticated method of combining influences is left for future work.

5 Experimentation and Results

Omen uses BNJ, Version 3.0 pre-Alpha 3 [1] as its probabilistic inference engine.

We used two ontologies obtained from the Knowledge Representation and Reasoning group at the Vrije University.[3] The ontologies are expressed in RDF using RDF-Schema. They contain concepts related to university departments and students, staff and faculty of the departments.

[3] `http://wbkr.cs.vu.nl/`

For our experiments, we extracted portions of the ontologies manually to make sure that they have at least some overlap. Because matching predicates is beyond the scope of this tool, we matched predicates across the ontologies by manual examination. When we decided that two predicates represented the same relationship, the names of one predicate was replaced by the names of the matching predicate in the other.

If the generated Bayesian Net contained a cycle, we manually weeded out one edge choosing one at random from those edges that are between nodes that are farthest from the root.

For this preliminiary experimentation, instead of using several values as cited in Section 2 that a node in the Bayesian Net can have, we just assigned two values "true" and "false" to the nodes. A value of "true" represents that the concepts represented by the node match and vice-versa.

We generated initial probability distribution for matches using a simple script using string-edit distance. Recall that in practice these probability distributions can come from any other tool. To generate probability distributions for the experiment, we manually identified i) the nodes across the ontolgies that the matched and ii) the nodes that surely did not match. The matching nodes were assigned a random probability between 0.7 to 0.9 and the nodes that did not match assigned a random number between 0.1 to 0.4 with the rest of the nodes lying in between 0.4 to 0.7.

We fixed the threshold values to 0.85 and 0.15. That is, if the probability of a match is determined by our methods or by the previous method to be greater than 0.85, then we determine that the concepts match and if the probability is less than 0.15, then we declare that the concepts do not match. Such matches and mismatches are used as evidence to the Bayesian NetḞurthermore, the same threshold value is used to determine a match from the posterier probability generated by OMEN. In some cases, the threshold was taken to be too stringent and resulted in lower recall. As future work, we intend to look into dynamically selecting proper thresholds by clustering.

We experimented with two sets of ontology graphs. In the first set, both graphs had 11 nodes each and in the second case both had 19 nodes. The preliminary results that we obtained are given in the two tables below:

Table 1. Summary of results for the smaller ontologies

Case No.	CPT-Method	Precision	Recall	F-measure
1	FI	0.75	0.375	0.5
	AP	1.0	0.5	0.67
2	FI	1.0	0.5	0.67
	AP	1.0	0.875	0.933
3	AP	1.0	0.75	0.85
4	AP	1.0	0.125	0.22

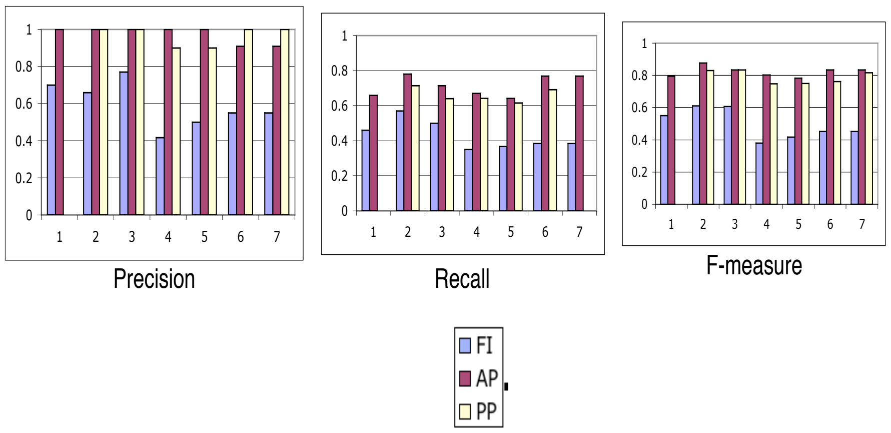

Fig. 3. Results for the ontologies of size 19 nodes

Table 1 lists the results for the case where the source ontologies were of size 11 nodes each. In this case, we specified three matching nodes as positive evidence and four pairs of nodes that do not match as negative evidence. The precision, recall and f-measure are calculated in the usual Information Retrieval (IR) sense using both the positive match and the negative match results together. In case 1, the evidence was introduced at random points. In case 2, the evidence was introduced at or very near the leaf nodes. The results show that introducing the evidence at or near the leaf nodes increased the performance of the algorithm. Case 3 is very similar to Case 2, but with false evidence introduced. Case 4 shows the effect of introducing drastic errors in the initial probabilities. Since the CPTs in the AP method depend directly on the quality of the initial probabilities, when we assigned the initial probabilities at random intentionally, the quality of the results deteroriate.

Overall, we see that the AP method outperforms the FI method in both cases. We also see that by giving only 3 matches out of 11, we could generate upto 7 of the missing matches. This implies that the method can be very useful even when the results of the previous matcher is not very good as not as it is totally random.

We show the results for the case where the source ontologies contained 19 nodes each in Figure 3. The figure shows 7 different test cases for the three different CPT-generating methods. The FI method is shown first, followed by the AP method, followed by the PP method for each test case.

The evidence provided in the cases above were as follows: For case 1, we provided positive evidence of 4 matches at or near the leaf nodes. For cases 2, 3, and 4, we provided positive evidence of 5 matches and negative evidence of 4 matches. For cases 5, 6, and 7 we provided positive evidence of 6 matches and negative evidence of 4 matches. In cases 2, and 3 the evidence was also provided at or near the leaf nodes. In cases 4, and 5, the evidence was provided at or near the root nodes. In cases 6 and 7, the evidence was provided at randomly selected nodes. In cases 3, 5, and 7, wrong evidence was introduced.

Not surprisingly, we see that both the AP and the PP methods outperforms the FI method of constructing CPTs and provide good precision and recall values. The AP method slightly outperforms the PP method in general. However, the PP method is more stable, that is, it recovers from a few wrong evidences better than the AP method. In this case, the place where the evidence was introduced did not matter much for the AP and PP methods.

6 Future Work

In the future, we intend to perform experiments to determine whether a system based on up-flow rules outperform one based on down-flow rules and to identify the scenarios when one outperforms the other. Furthermore, we will extend our algorithm to perform multiple iterations on the data. For example, we can employ alternate iterations using down-flow and up-flow rules for a fixed number of iterations or until the results converge. Designing better CPTs using more of the semantics of the ontology relationships, and emperically evaluating them and experimenting with large ontologies and coupling our matcher with various external matchers are charted for as future work.

7 Related Work

Two research directions are related to our work: automatic or semi-automatic discovery of ontology mappings and the use of uncertainty in knowledge-based systems.

7.1 Automatic ontology mapping

Over the past decade, researchers have actively worked on developing methods for discovering mappings between ontologies or database schemas. These method employ a slew of different techniques. For example, Similarity Flooding [8] and AnchorPrompt [10] algorithms *compare graphs* representing the ontologies or schemas, looking for similarities in the graph structure. GLUE [3] is an example of a system that employs *machine-learning techniques* to find mappings. GLUE uses multiple learners exploiting information in concept instances and taxonomic structure of ontologies. GLUE uses a probabilistic model to combine results of different learners. Hovy [5] describes a set of heuristics that researchersat ISI/USC used for semi-automatic alignment of domain ontologiesto a large central ontology. Their techniques are based mainly on*linguistic analysis* of concept names and natural-languagedefinitions of concepts. A number of researchers propose *similarity metrics* between concepts in different ontologies based on their relations to other concepts. For example, a similarity metric between concepts in

OWL ontologies developed by Euzenat and Volchev [4] is a weighted combination of similarities of various features in OWL concept definitions: their labels, domains and ranges of properties, restrictions on properties (such as cardinality restrictions), types of concepts, subclasses and superclasses, and so on. Finally, approaches such as ONION [9] and Prompt [11] use a combination of *interactive specifications* of mappings and *heuristics* to propose potential mappings.

The approach that we describe in this paper is complementary to the techniques for automatic or semi-automatic ontology mapping. Many of the methods above produced pairs of matching terms with some degree of certainty. We can use these results as input to our network and run our algorithm to improve the matches produced by others or to suggest additional matches. In other words, our work complements and extends the work by other researchers in this area.

7.2 Probabilistic knowledge-base systems

Several researchers have explored the benefits of bringing together Nayes Nets an knowledge-based systems and ontologies. For instance, Koller and Pfeffer [7] developed a "probabilistic frame-based system," which allows annotation of frames in a knowledge base with a probability model. This probability model is a Bayesian Net representing a distribution over the possible values of slots in a frame. In another example, Koller and colleagues [6] have proposed probabilistic extensions to description logics based on Bayesean Networks.

In the context of the Semantic Web, Ding and Peng [2] have proposed probabilistic extensions for OWL. In this model, the OWL language is extended to allow probabilistic specification of class descriptions. The authors then build a Bayesean Network based on this specification, which models whether or not an individual matches a class description and hence belongs to a particular class in the ontology.

Researchers in machine learning have employed probabilistic techniques to find ontology mappings. For example, the GLUE system mentioned earlier [3], uses a Bayes classifier as part of its integrated approach. Similarly, Prasad and colleagues [12] use a Bayesean approach to find mappings between classes based on text documents classified as exemplars of these classes. These approaches, however, consider instances of classes in their analysis and not relations between classes, as we do. As with other approaches to ontology mapping, our work can be viewed as complementary to the work done by others.

8 Conclusion

We have outlined the design and implementation of Omen, an ontology match enhancer tool, that improves existing ontology matches based on a probabilistic inference. This tool is dependent upon a set of meta-rules which express the influences of matching nodes on the existence of other matches across concepts in source ontologies that are located in the proximity of the matching nodes. We described how we implemented a simple first version of the matching tool and discussed our preliminary results. We have also outlined several improvements that can be made to the tool and identified several open questions that if resolved can make the performance of the tool even better.

References

1. Bayesian network tools in java(bnj), version 2.0, July 2004.
2. Z. Ding and Y. Peng. A probabilistic extension to ontology language owl. In *37th Hawaii International Conference On System Sciences (HICSS-37)*, Big Island, Hawai, 2004.
3. 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.
4. J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-Lite. In *The 16th European Conference on Artificial Intelligence (ECAI-04)*, Valencia, Spain, 2004.
5. E. Hovy. Combining and standardizing largescale, practical ontologies for machine translation and other uses. In *The First International Conference on Language Resources and Evaluation (LREC)*, pages 535–542, Granada, Spain, 1998.
6. D. Koller, A. Levy, and A. Pfeffer. P-Classic: a tractable probabilistic description logic. In *14th National Conference on Artificial Intelligence (AAAI-97)*, 1997.
7. D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisconsin, 1998. AAAI Press.
8. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *18th International Conference on Data Engineering (ICDE-2002)*, San Jose, California, 2002. IEEE Computing Society.
9. P. Mitra, G. Wiederhold, and S. Decker. A scalable framework for interoperation of information sources. In *The 1st International Semantic Web Working Symposium (SWWS'01)*, Stanford University, Stanford, CA, 2001.
10. 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.
11. N. F. Noy and M. A. Musen. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
12. S. Prasad, Y. Peng, and T. Finin. A tool for mapping between two ontologies using explicit information. In *AAMAS 2002 Workshop on Ontologies and Agent Systems*, Bologna, Italy, 2002.