OMEN: A Probabilistic Ontology Mapping Tool

Prasenjit Mitra[1], Natasha F. Noy[2], and Anuj Rattan Jaiswal[1]

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

Abstract. Most existing ontology mapping tools are inexact. Inexact ontology mapping rules, if not rectified, result in imprecision in the applications that use them. We describe a framework to probabilistically improve existing ontology mappings using a Bayesian Network. OMEN, an Ontology Mapping ENhancer, is based on a set of meta-rules that captures the influence of the ontology structure and the existing matches to match nodes that are neighbours to matched nodes in the two ontologies. We have implemented a protype ontology matcher that can either map concepts across two input ontologies or enhance existing matches between ontology concepts. Preliminary experiments demonstrate that OMEN enhances existing ontology mappings in our test cases.

1 Introduction

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

Several problems arise when an expert entrusted with matching two ontologies constructs the semantic mappings between them. First, the expert needs automatic or semi-automatic means for establishing the mappings. Often, the schemas or ontologies are quite large. Manually establishing the mappings among large ontologies is prohibitively expensive or downright impossible due to the sheer number of entities that the expert needs to match. Thus, an expert needs an automated ontology matching tool. However, automatic tools often use heuristics and may be imprecise. The expert would like to specify, along with each mapping, the amount of uncertainty associated with it. Second, precise mappings may not even exist. Even a human expert may be able to come up only with approximate mappings. In particular, the expert may not be able to come up with precise mappings if the expert has incomplete information about class definitions. Therefore, we introduce **probabilistic mappings** that map each pair

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 537–547, 2005.

of concepts from two sources and the mapping between them has an associated probability.

Our main premise in this work is the following: if we know a mapping between two concepts from the sources, we can use the mapping to infer mappings between related concepts. For example, if two properties are equivalent, and so are their domains, we can infer (with some certainty) that their ranges are equivalent as well. Therefore, we can build a Bayes Net with the concept mappings. The Bayes Net uses a set of *meta-rules* 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, OMEN, an Ontology Mapping Enhancer. OMEN uses a Bayes Net to provide probabilistic inference and aids in enhancing existing ontology mappings by deriving missed matches and invalidating existing false matches. Preliminary results show that using OMEN an expert 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 Bayes 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. 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.

The rest of the paper is organized as follows. Section 2 contains a description of the knowledge model used to represent the ontologies and the mapping expressions, In Section 3, we discuss the use of meta-rules to generate new probability distributions based on existing ones. Section 4 contains a description of how the Bayes Net that OMEN uses is constructed. In Section 5, we briefly outline our prototype implementation and provide the results of our experiments. Section 6 describes some open issues and scope for future work. In Section 7, we discuss the related work and conclude the paper in Section 8.

2 Knowledge Model and Mapping Expressions

We assume a simple ontology model (that is more or less similar to RDF Schema or a subset of OKBC with more commonly used facets). 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 (which are primitive values, such as numbers, strings, etc., and we omit these from mapping consideration) and relationships between classes. Properties have one or more **domains**, which are classes to which the property can be attached; and one or more **ranges**, which restrict the classes for the values of property.

Given two ontologies, O and O', a **mapping** between two concepts, $C \in O$ and $C' \in O'$, $m(C, C')$ (C and C' can be either classes or properties), is one of the following values:

- $=$ if the concepts are equivalent;
- $\subset$ if C is a specialization of C';
- $\supset$ if C' is a generalization of C;
- $\cap$ if there is an overlap between C and C';
- $\times$ if C and C' are not related.

Note, that we may not know the value of the mapping at all. In other words, if $(m(C, C')) = \times$, then we *know* that they are not related, which is different from not knowing what their relationship is.

Therefore, for each mapping $m(C, C')$, we can talk about a **probability distribution** over the mapping values $\{=, \subset, \supset, \cap, \times\}$. For each mapping $m(C, C')$, the sum of the values in the distribution must be less than or equal to 1. It is less than one if we do not have complete information about the distribution.

3 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. The algorithm starts the process by initializing probability distributions for some of the mappings, using the output of various automatic mapping tools. It then uses a set of meta-rules to derive new mappings based on existing ones.

3.1 Examples of Meta-rules

Before giving an example of a meta-rule, we describe the notations and shorthands that we use for brevity. In all the rules for mapping between two ontologies, O and O', we use the following:

- 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.

The following is an example of a meta-rule to generate a mapping between classes in the range of a property based on the mapping of the properties themselves and their domains. Not included explicitly in the rule (for brevity) is an assumption that both properties, q and q', have a single domain and range.

$$P(C_d = C'_d, x) \wedge P(q = q', 1) \wedge$$
$$domainOf(q, C_d) \wedge domainOf(q', C'_d) \wedge$$
$$rangeOf(q, C_r) \wedge rangeOf(q', C'_r)$$
$$\Rightarrow P(C_r = C'_r, x)$$

Below are some (informal) examples of other metarules:

Mappings between superclasses and all but several siblings: Here we can probably still say something about the probabilities that the rest of the siblings match in a pairwise fashion. (If these are later combined with rules producing other evidence, such an observation will be helpful).

Mappings between a property and domain of a property: **Natasha, can you please fill in what you meant by this**

Mapping between properties: If two classes are ranges for matching properties, then they may match (but probabilities are reduced somewhat compared to the case when the domains match as well).

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.

3.2 Combining the Results from Several Rules for the Same Mapping

Consider a pair of classes, C and C' (Figure 1). Let us assume that initially we do not have any probability distribution for $m(C, C')$. Then a number of meta-rules affect the probability distribution for $m(C, C')$. In the example in the figure, the following rules and mappings affect the result:

– A mapping between superclasses of C and C'; mappings between the siblings of C and C'. Let's say we know that $P(C_0 = C'_0, x)$, for some $x < 1$, and $P(C_0 \subset C'_0, y)$, for some $y < 1$, $(x + y \leq 1)$, and $P(C_1 = C'_1, 1)$. We then use the following rule (we omit the relationships that are obvious from the figure, such as *subclassOf* (C, C_0) from the rule):

$$p(C_0 =/\subset C'_0, x) \wedge p(C_1 = C'_1, 1) \Rightarrow$$
$$p(C =/\subset C', x)$$

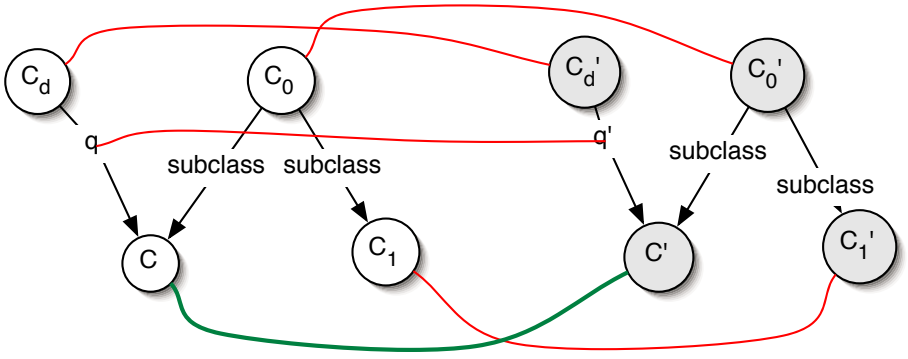

Fig. 1. The probability distribution for the mapping between C and C' is affected by the mappings between their superclasses, siblings, and domains of the properties q and q' for which C and C' are ranges

– A mapping between properties q and q' ($P(q = q', 1)$) for which C and C' are ranges respectively, and mappings between domains of q and q' ($P(C_d = C'_d, z)$). We are assuming that both q and q' have a single domain and range each.

$$P(C_d = C'_d, z) \wedge P(q = q', 1) \Rightarrow$$
$$P(C = C', z)$$

For the moment, let us assume that there are no more influences on the mapping $m(C, C')$.

We now need to combine the following three results of the rules above:

1. $P(C = C, x)$
2. $P(C \subset C, y)$
3. $P(C = C', z)$

We also know that $x + y \leq 1$. The algorithm must combine probabilitic influences of different rules and determine the probability distribution of the matching nodes.

In theory, we could introduce some certainty factor into the rules. However, an easier solution would be to use probabilities to reflect that. For example, if the rule is fairly certain, then we simply propagate the initial probability x (as in the rules above). If the rule is more speculative, we can reduce the probability in the result of the rule by some factor.

3.3 Eliminating Circular Dependencies

We define a **generation** of classes in the following way:

1. All subclasses of the same superclass (siblings) belong to the same generation.

2. If a class C is a range of property p, then all other classes in the range of p are in the same generation as C.

In order to make the problem tractable, we use only the rules that link neighboring generations.

We can now categorize the rules in the following three categories:

Down-flow rules: rules where mapping values for one generation affects the mappings of the *next* generation

Up-flow rules: rules where mapping values for one generation affects the mappings for the *previous* generation

Same-generation rules: rules where mapping values for one generation affect mapping values for the same generation.

Combination rules: any combination of the above.

In order to avoid circular dependencies, we use the following restrictions:

1. In same-generation rules, only the values that were available prior to the current iteration, can be used in the antecedent of rules. Same-generation rules that introduce cycles are broken arbitrarily by dropping some edges.
2. Each iteration is either a down-flow iteration or an up-flow iteration. The only rules that are allowed in a down-flow (up-flow) iteration are down-flow (up-flow) rules, same-generation rules, and combination rules with only down-flow (up-flow) rules.

For the rest of this note, we only consider a down-flow iteration.

During a down-flow iteration, a mapping between two classes, $m(C, C')$ is affected by the mappings between:

- their superclasses;
- their siblings;
- the properties for which C and C' are ranges, the domains of these properties, and other ranges of these properties.

Note that all the examples of the rules in section 3are down-flow, same-generation, or combination rules (where combination is same-generation and down-flow). In other words, all these rules can be used in a down-flow iteration.

4 Construction of the Bayesian Network

We construct a Bayes Net to represent the influences and inter-relationships between ontology matches as described below. We describe the construction of a down-flow graph, that is a graph constructed using only the down-flow rules, or same-generation rules. An up-flow graph can be constructed correspondingly.

4.1 The BN-Graph

Initially, we construct all possible nodes representing all possible matches across the two ontologies. That is, if ontology $O1$ contains n nodes, and ontology $O2$ contains m nodes, we construct $m \times n$ nodes in the Bayes Net graph representing all possible matches. Each node in the Bayes Net graph represents a unique match between two concepts in the source ontologies. That is, each node represents a match from a pair of concepts such that no two concepts in a pair come from the same ontology and no two nodes represent the same set of concepts. As is evident, we create a large number of matches that are very unliquely to exist in practice. We discuss later how we can reduce the number of nodes in the Bayes Net for large ontologies.

Down-flow edges: The down-flow edges in the graph are constructed as follows. Let there be parent concepts P_1 and P_1' such that concepts C_1 and C_2 are the children of the parent concepts respectively in the source ontologies. For all possible quadruple (P_1, P_2, C_1, C_2), a directed edge between nodes $n1$ and $n2$ is added to the graph, where node $n1$ represents the match between P_1 and P_2 and node $n2$ represents the match between C_1 and C_2.

In the current implementation, we do not add any same-generation edges to avoid creating cycles. Recall, the graph in a Bayes Net must not contain cycles and is a DAG. How same-generation information can be introduced in a Bayes Net is left for future work.

4.2 Apriori Probabilities and Evidence

The "root" nodes in the BN-Graph, that is nodes that have no parents, must be provided with a set of apriori probabilities. This set of apriori probabilities is obtained from the previous matching – the output of a matcher that is input to OMEN. [1] We assume that the previous matching matched the source ontologies and generated probabilities that model the certainty of all possible candidate matches (or at least the matches represented by the root nodes in the BN-Graph). OMEN also obtains the evidence to the Bayes Net from the final result of the previous matcher.

4.3 Meta-rules and Conditional Probability Tables

The meta-rules form the basis for constructing the conditional-probability tables(CPTs) for each node. The construction of appropriate conditional probability tables is the hardest and most subjective task in the design of our Bayes Net . Yet, the accuracy of the CPTs deeply influence the veracity of the results obtained from any Bayes Net .

The interface to OMEN allows external functions to generate and provide the tool with appropriate CPTs. These functions are called on a per node basis and

[1] Recall, that OMEN enhances existing ontology matches.

passed a neighbourhoods (in the source ontologies) of the two concepts that the node represents as arguments.

OMEN also provides internal functions that generate the CPTs based on meta-rules and the apriori probabilities. Ideally, the CPTS should not depend upon the apriori probabilities of the nodes and an expert will indicate how one match in the Bayes Net influences other matches and also provide the probability tables expressing such an influence. However, when such an expert is absent, we have empirically observed that using the apriori probabilities in conjunction with the meta-rules to generate the CPTs provides better matches than those obtained by just using the meta-rules.

4.4 Reducing the Number of Nodes

If the source ontologies are very large, the number of nodes in the Bayes Net being quadratic in the number of nodes in the source ontologies, are extremely large too. We reduce this number by removing nodes in the Bayes Net that do not have any parents or children. We observed empirically that there are a significant number of such nodes.

The Bayes Net that is constructed is a collection of DAGs. Consider a node present in a DAG none of whose nodes are supplied as evidence. This situation happens because all the nodes in the DAG represent matches that are deemed unrealistic by the previous matcher. In absence of a single evidence in the entire DAG, the Bayes Net cannot perform any inference in that DAG. Therefore, we can reduce the Bayes Net by removing that DAG from the collection of DAGs altogether.

If we want to reduce the number of nodes even further, we can adopt the following policy. Before the Bayes Net is constructed, identify the evidence nodes — those that are classfied as matches by the previous matcher. Now create a node in the Bayes Net only for those pairs of nodes such that their distance from the evidence node is at most k, where k is a small integer $(1, 2, 3, ...)$. This restriction implies that any node in the Bayes Net represents nodes in the ontology whose distance from a node (in the ontology) that has been matched by the previous matcher is at most k.

5 Implementation and Experimentation

OMEN uses BNJ, Version 2.0 [1] as its probabilistic inference engine. In the current implementation, only down-flow meta-rules were used. The experiments were run on ...

6 Future Work

Up-flow rules: The current implementation takes into account only down-flow rules. In the future, we intend to perform experiments to determine whether the system based on up-flow rules outperform one based on down-flow rules and to identify the scenarios when one outperforms the other.

Multiple iterations: We start with some initial set of probability distributions and use a down-flow iteration to create new probability distributions for mappings. A case could be made to combine successive passes of down-flow and up-flow iterations using the down-flow and up-flow meta-rules alternately. Ideally, the system will perform multiple iterations until it meet some convergence criteria (or have done some "sufficient" number of iterations). The usefulness of performing multiple iterations needs to be empirically verified in the future. We also need to characterize whether the system automatically converges or whether there needs to be an arbitrary cut-off on the number of passes done. Another issue is whether the same meta-rules and CPTs be used during mutliple iterations or should different CPTs be generated for different passes?

Using inferred mappings in rules: We start with a set of mappings that we get "from the outside" as an initial sample of values. These mappings are set and we do not alter them. We generate new mappings and posterior probabilities in the first pass. Should the second pass, the first up-flow pass be performed using the output of the first pass? Or should it be performed on the original results and the results of the two independent passes combined to get better results. Should the evidence provided by the previous matcher be constantly treated as evidence, or should it be changed depending upon the posterior probabilities generated by the several iterations of OMEN?

Other mappings to consider: For now, all the rules are "symmetric" in the way we treat generations. For example, the mappings between superclasses and some of their siblings affect the mapping between the remaining siblings. What if we have a mapping between two classes, C and C', some of their subclasses, but then a subclass of C is mapped to a class in O' that is not closely related to C' (definitely not its subclass)?

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-Based 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 Bayes 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.