Inexact Matching of Ontology Graphs Using Expectation-Maximization

Prashant Doshi
LSDIS Lab, Dept. of Computer Science
University of Georgia
Athens, GA 30602
pdoshi@cs.uga.edu

Christopher Thomas
LSDIS Lab, Dept. of Computer Science
University of Georgia
Athens, GA 30602
chaos@uga.edu

Abstract

We present a new method for mapping ontology schemas that address similar domains. The problem of ontology mapping is crucial since we are witnessing a decentralized development and publication of ontological data. We formulate the problem of inferring a match between two ontologies as a maximum likelihood problem, and solve it using the technique of expectation-maximization (EM). Specifically, we adopt directed graphs as our model for ontologies and use a generalized version of EM to arrive at a mapping between the nodes of the graphs. We exploit the structural and lexical similarity between the graphs, and improve on previous approaches by generating a many-one correspondence between the concept nodes. We provide preliminary experimental results in support of our method and outline its limitations.

Introduction

The growing popularity of the semantic Web is fueled in part by the development and publication of an increasing number of ontologies. Because the development of these ontologies is occurring in a decentralized manner, the problems of matching similar ontologies (alignment) and merging them into a single comprehensive ontology gain importance. Previous approaches for matching ontologies have utilized either the instance space associated with the ontologies, the ontology schema, or both. For example, FCA-Merge (Stumme & Maedche 2001) and IF-MAP (Kalfoglou & Schorlemmer 2003) rely on the instances of the concepts and documents annotated by the ontologies to generate the mappings. While FCA-Merge applies linguistic techniques to the instances, IF-MAP utilizes information flow concepts for identifying the mappings. Other approaches that rely heavily on the instance space include BayesOWL (Ding *et al.* 2005) which uses the instances to learn the parameters of the Bayesian network. The joint probability of a pair of concepts is used as a similarity measure. The FALCON-AO system (Hu *et al.* 2005) on the other hand proposes metrics for evaluating the structural similarity between the ontology schemas to arrive at a mapping between the ontologies. In the same vein, OMEN (Mitra, Noy, & Jaiswal 2004) probabilistically infers a match between classes given a match between parents and children, using Bayesian networks. The GLUE system (Doan *et al.* 2002) uses the instances to compute a similarity measure (Jaccard coefficient) between the concepts, and feeds it to a relaxation labeler that exploits general and domain-specific heuristics to match the ontology schemas. The performance of systems utilizing instance spaces is closely linked to the volume of the training data – instances – available.

Contemporary languages for describing ontologies such as RDF(S) and OWL allow ontologies to be modeled as *directed labeled graphs*. Therefore, analogous to graph matching techniques, the ontology matching approaches also differ in the cardinality of the correspondence that is generated between the ontological concepts. For example, several of the existing approaches (Ding *et al.* 2005; Hu *et al.* 2005; Doan *et al.* 2002; Mitra, Noy, & Jaiswal 2004) focus on identifying a one-one (exact) mapping between the concepts. More general are the many-one and many-many (inexact) correspondences between the concepts. These are of particular interest, since they allow mappings between concepts of differing semantic granularity. In other words, a cluster of concepts may be mapped to a single target concept. The methods that generate such correspondences have wider applicability as independently developed ontologies seldom have the same number of concepts.

We present a graph-theoretic method that generates many-one mappings between the participating ontologies to be matched. We focus our analysis on the ontology schemas and use directed graphs as the underlying models for the ontologies. We formulate the problem as one of finding the most likely map between two ontologies, and compute the likelihood using the expectation-maximization (EM) technique (Dempster, Laird, & Rubin 1977). The EM technique is typically used to find the maximum likelihood estimate of the underlying model from observed data containing missing values. In our formulation, we treat the set of correspondences between the pair of ontologies to be matched as hidden variables, and define a mixture model over these correspondences. The EM algorithm revises the mixture models iteratively by maximizing a weighted sum of the log likelihood of the models. The weights, similar to the general EM algorithm, are the posterior probabilities of the hidden correspondences. Within the EM approach, we exploit the structural as well as the lexical similarity between the schemas to compute the likelihood of a map. While analogous ap-

proaches for graph matching appear in computer vision (Luo & Hancock 2001), these are restricted to unlabeled graphs.

The particular form of the mixture models in our formulation precludes a closed form expression for the log likelihood. Subsequently, standard maximization techniques such as (partial) differentiation are intractable. Instead, we adopt the generalized version of the EM (Dempster, Laird, & Rubin 1977) which relaxes the maximization requirement and simply requires the selection of a mixture model that improves on the previous one. Since the complete space of candidate mixture models tends to be large and to avoid local maximas, we randomly sample a representative set of mixture models and select the candidate from among them. To speed up convergence, we supplement the sampled set with locally improved estimates of the mixture models that exploit simple mapping heuristics. We evaluate our approach on example benchmark ontology pairs that were obtained from the I^{3}CON repository (Hughes & Ashpole 2004).

Background: Expectation-Maximization

The expectation-maximization (EM) technique was originally developed by Dempster et al. (1977) to find the maximum likelihood estimate of the underlying model from observed data instances in the presence of missing values. The main idea behind the EM technique is to compute the expected values of the hidden or missing variable(s) using the observed instances and a previous estimate of the model, and then recompute the parameters of the model using the observed and the expected values as if they were observations.

Let X be the set of observed instances, M the underlying model, and Y be the set of missing or hidden values. The expectation step is a weighted summation of the log likelihood, where the weights are the conditional probabilities of the missing variables:

$$\textbf{E Step: } Q(M^{n+1}|M^n) = \sum_{y \in Y} Pr(y|X, M^n) L(M^{n+1}|X, y)$$

where $L(M^{n+1}|X, y)$ is the log likelihood of the model, computed as if the value of the hidden variable is known. The logarithm is used for simplifying the likelihood computation.

The maximization step consists of selecting the model that maximizes the expectation:

$$\textbf{M Step: } M_*^{n+1} = \underset{M^{n+1} \in \mathcal{M}}{argmax} \quad Q(M^{n+1}|M^n)$$

The above two steps are repeated until the model parameters converge. Each iteration of the algorithm is guaranteed to increase the log likelihood of the model estimate, and therefore the algorithm is guaranteed to converge to either the local or global maxima (depending on the vicinity of the start point to the corresponding maxima).

Often, in practice, it is difficult to obtain a closed form expression in the E step, and consequently a maximizing M^{n+1} in the M step. In this case, we may replace the original M step with the following: Select M^{n+1} such that $Q(M^{n+1}|M^n) \geq Q(M^n|M^n)$. The resulting generalized EM (GEM) method (Dempster, Laird, & Rubin 1977) retains the convergence property of the original algorithm, while improving its applicability.

Ontology Model

Contemporary languages for describing ontologies – categorized as description logics – include RDF and OWL. Both these languages allow the ontologies to be modeled as directed labeled graphs (Hayes & Gutierrez 2004) where the nodes of the graphs are the concepts (classes in RDF) and the labeled edges are the relationships (properties) between the classes. Following graph matching terminology, we assume the graph with the larger number of nodes to be the *data* while the other as the *model* graph. Formally, let the data graph be $\mathcal{O}_d = \langle V_d, E_d, L_d \rangle$, where V_d is the set of labeled vertices representing the concepts, E_d is the set of edges representing the relations which is a set of ordered 2-subsets of V_d, and $L_d : E_d \rightarrow \Delta$ where Δ is a set of labels, gives the edge labels. Analogously, $\mathcal{O}_m = \langle V_m, E_m, L_m \rangle$ is the model graph against which the data graph is matched.

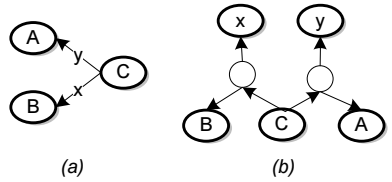


Figure 1: The process of reification. (a) An edge labeled graph. (b) The reified bipartite graph in which each distinct edge label is a node, and additional dummy nodes are introduced to preserve the relations. These nodes may have identical labels.

To facilitate graph matching, we transform the edge-labeled graphs into unlabeled ones by elevating the edge labels to first class citizens. This process, called reification, involves treating the relationships as resources, thereby adding them as nodes to the graph. We observe that reification becomes unnecessary, from the perspective of graph matching, when all edges have the same labels. We illustrate reification using a simple example in Fig. 1 and point out that the reified graph is a bipartite graph (Hayes & Gutierrez 2004). Consequently, the functions L_d in $\mathcal{O}_d$ and L_m in $\mathcal{O}_m$ become redundant. However, reification comes at a price: The reified graph contains as many additional nodes as the number of edges and distinct edge labels.

Graph Matching Using GEM

As we mentioned previously, we model the ontologies as graphs, $\mathcal{O}_d$ and $\mathcal{O}_m$, and consequently focus on the graph matching problem. Let M be a $|V_d| \times |V_m|$ matrix that represents the match between the two graphs. In other words,

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1|V_m|} \\ m_{21} & m_{22} & \dots & m_{2|V_m|} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ m_{|V_d|1} & m_{|V_d|2} & \dots & m_{|V_d||V_m|} \end{bmatrix}$$

Here each assignment variable,

$$m_{a\alpha} = \begin{cases} 1 & \text{if } f(x_a) = y_\alpha : x_a \in V_d,\ y_\alpha \in V_m, \\ 0 & \text{otherwise} \end{cases}$$

If the correspondence, $f : V_d \rightarrow V_m$, is a one-one mapping and $\{x_a, x_b\} \in E_d \Leftrightarrow \{f(x_a), f(x_b)\} \in E_m$, then f is an isomorphism. We call the property of preserving edges across transformations as *edge consistency*. The correspondence f is a homomorphism if it is a many-one or many-many mapping, and is edge consistent. In this paper, we focus on tractably generating homomorphisms with many-one mappings.

We formulate the graph matching as a maximum likelihood (ML) problem. Specifically, we are interested in the match matrix, M, that gives us the maximum conditional probability of the data graph, $\mathcal{O}_d$, given the model graph, $\mathcal{O}_m$ and the match assignments. Formally,

$$M_* = \underset{M \in \mathcal{M}}{argmax}\ Pr(\mathcal{O}_d|\mathcal{O}_m, M) \tag{1}$$

where $\mathcal{M}$ is the set of all match assignments. In general, there may be $2^{|V_d||V_m|}$ different matrices, but by restricting our analysis to many-one correspondences – these may be partial – we reduce the search space to $(|V_m|+1)^{|V_d|}$. As is common, we may assume the data graph nodes to be conditionally independent, and sum over the model graph nodes using the law of total probability.

$$\begin{aligned} Pr(\mathcal{O}_d|\mathcal{O}_m, M) &= \prod_{x_a \in V_d} \sum_{y_\alpha \in V_m} Pr(x_a|y_\alpha, M) Pr(y_\alpha|M) \\ &= \prod_{x_a \in V_d} \sum_{y_\alpha \in V_m} Pr(x_a|y_\alpha, M)\pi_\alpha \end{aligned}$$

where $\pi_\alpha = Pr(y_\alpha|M)$ is the prior probability of the model graph vertex, y_α, given the mixture model, M.

In order to solve the ML problem, we note that the correspondence, f, is hidden from us. Additionally, if we view each assignment variable, $m_{a\alpha}$, as a model, then the matrix M may be treated as a mixture model. Consequently, the mixture model, M, is parameterized by the set of the constituent assignment variables. Both these observations motivate the formulation of an EM technique to compute the model with the maximum likelihood.

E Step

We start our analysis by formulating a conditional expectation of the *log likelihood* with respect to the hidden variables given the data graph and a guess of the mixture model, M^n.

$$Q(M^{n+1}|M^n) = E\Big[log\ Pr(x_a|y_\alpha, M^{n+1})\pi_\alpha^{n+1}|x_a, M^n\Big] \tag{2}$$

The expectation may be rewritten as a weighted summation of the log likelihood with the weights being the posterior probabilities of the hidden correspondences under the matrix of assignment variables at iteration n. Equation 2 becomes:

$$\begin{aligned} &Q(M^{n+1}|M^n) = \textstyle\sum_{\alpha=1}^{|V_m|}\sum_{a=1}^{|V_d|} Pr(y_\alpha|x_a, M^n) log Pr(\\ &x_a|y_\alpha, M^{n+1}) + \textstyle\sum_{\alpha=1}^{|V_m|}\sum_{a=1}^{|V_d|} Pr(y_\alpha|x_a, M^n) log\ \pi_\alpha^{n+1} \end{aligned} \tag{3}$$

Next, we address the computation of each of the terms in Eq. 3. We first focus on the posterior, $Pr(y_\alpha|x_a, M^n)$. Once we establish a method of computation for this term, the generation of $log\ Pr(x_a|y_\alpha, M^{n+1})$ follows analogously.

Using Bayes theorem $Pr(y_\alpha|x_a, M^n)$ may be rewritten,

$$Pr(y_\alpha|x_a, M^n) = \frac{Pr(x_a|y_\alpha, M^n)\pi_\alpha^n}{\sum_{\alpha=1}^{|V_m|} Pr(x_a|y_\alpha, M^n)\pi_\alpha^n} \tag{4}$$

We turn our attention to the term $Pr(x_a|y_\alpha, M^n)$ in Eq. 4. This term represents the probability that the data graph node, x_a, is in correspondence with the model graph node, y_α, under the match matrix of iteration n, M^n. Using Bayes theorem again,

$$Pr(x_a|y_\alpha, M^n) = \frac{Pr(M^n|y_\alpha, x_a)Pr(y_\alpha, x_a)}{Pr(y_\alpha, M^n)}$$

As we mentioned before, M^n is a mixture of the models, $m_{a\alpha}$. We treat the models to be independent of each other. This allows us to write the above equation as,

$$Pr(x_a|y_\alpha, M^n) = \frac{Pr(y_\alpha, x_a)\Pi_{b=1}^{|V_d|}\Pi_{\beta=1}^{|V_m|} Pr(m_{b\beta}^n|y_\alpha, x_a)}{Pr(y_\alpha)\Pi_{b=1}^{|V_d|}\Pi_{\beta=1}^{|V_m|} Pr(m_{b\beta}^n|y_\alpha)}$$

We note that,

$$Pr(m_{b\beta}^n|y_\alpha, x_a) = \frac{Pr(x_a|y_\alpha, m_{b\beta}^n)Pr(m_{b\beta}^n|y_\alpha)}{Pr(x_a|y_\alpha)}$$

Substituting this into the numerator of the previous equation results in,

$$\begin{aligned} Pr(x_a|y_\alpha, M^n) = \ & [\tfrac{1}{Pr(x_a|y_\alpha)}]^{|V_d||V_m|-1} \\ & \times \Pi_{b=1}^{|V_d|}\Pi_{\beta=1}^{|V_m|} Pr(x_a|y_\alpha, m_{b\beta}^n) \end{aligned} \tag{5}$$

We first focus on the term $Pr(x_a|y_\alpha, m_{b\beta}^n)$, which represents the probability that x_a is in correspondence with y_α given the assignment model, $m_{b\beta}$. As we mentioned previously, $m_{b\beta}$ is 1 if x_b is matched with y_β under the correspondence f, otherwise it is 0. Let us call the set of nodes that are adjacent to x_a as its *neighborhood*, $\mathcal{N}(x_a)$. Since we seek f to be a homomorphism that must be edge consistent, for each vertex, $x \in \mathcal{N}(x_a)$, the corresponding vertex, $f(x) \in \mathcal{N}(y_\alpha)$. Therefore, the probability of x_a being in correspondence with y_α is dependent, in part, on whether the neighborhood of x_a is mapped to the neighborhood of y_α under f. Several approaches for schema matching and graph matching (Luo & Hancock 2001) are based on this observation. To formalize this, we introduce EC:

$$EC = \begin{cases} 1 & \langle x_a, x_b\rangle \in E_d \wedge \langle y_\alpha, y_\beta\rangle \in E_m \wedge m_{b\beta} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

In addition to the structural similarity, $Pr(x_a|y_\alpha, m_{b\beta}^n)$ is also influenced by the lexical similarity between the concept labels of the nodes x_a and y_α.

$$Pr(x_a|y_\alpha, m_{b\beta}^n) = (1 - P_\epsilon(x_a, y_\alpha))^{EC} P_\epsilon(x_a, y_\alpha)^{1-EC} \tag{7}$$

Here $P_\epsilon : V_d \times V_m \rightarrow [0, 1]$ is the correspondence error based on the lexical similarity of the node labels. We address the computation of P_ϵ later in this paper.

In the term $Pr(x_a|y_\alpha)$ in Eq. 5, x_a is independent of y_α in the absence of the mixture model. Therefore, $Pr(x_a|y_\alpha) =$

$Pr(x_a)$ whose value depends only on the identity of the node, x_a. In this paper, we assume this distribution to be uniform. Substituting Eqs. 5 and 7 into Eq. 4, we get,

$$Pr(y_\alpha|x_a, M^n) = C_a[\frac{1}{Pr(x_a|y_\alpha)}]^{|V_d||V_m|-1} \times \Pi_{b=1}^{|V_d|}\Pi_{\beta=1}^{|V_m|}(1 - P_\epsilon(x_a, y_\alpha))^{EC} P_\epsilon(x_a, y_\alpha)^{1-EC} \quad (8)$$

where C_a is the normalizing constant and EC is as in Eq. 6.

We now look at the log likelihood term, $log\ Pr(x_a|y_\alpha, M^{n+1})$, in Eq. 3. The computation of this term follows a similar path as before, with the difference being that we use the new mixture model, M^{n+1}. Analogous to Eq. 5, we get,

$$log\ Pr(x_a|y_\alpha, M^{n+1}) = log\ \left[\frac{1}{Pr(x_a|y_\alpha)}]^{|V_d||V_m|-1} \times \Pi_{b=1}^{|V_d|}\Pi_{\beta=1}^{|V_m|} Pr(x_a|y_\alpha, m_{b\beta}^{n+1})\right]$$

The presence of the log considerably simplifies the above.

$$log\ Pr(x_a|y_\alpha, M^{n+1}) = (|V_d||V_m| - 1)log\ \frac{1}{Pr(x_a|y_\alpha)} + \sum_{b=1}^{|V_d|}\sum_{\beta=1}^{|V_m|} log\ Pr(x_a|y_\alpha, m_{b\beta}^{n+1})$$

$Pr(x_a|y_\alpha, m_{b\beta}^{n+1})$ may be computed analogously to Eq. 7.

M Step

The maximization step involves choosing the mixture model, M_*^{n+1}, that maximizes $Q(M^{n+1}|M^n)$, shown in Eq. 3. This mixture model then becomes the input for the next iteration of the E-step. However, the particular formulation of the E step and the structure of the mixture model make it difficult to carry out the maximization. Therefore, we relax the maximization requirement and settle for a mixture model, M_*^{n+1}, that simply improves the Q value. As we mentioned before, this variant of the EM technique is called the *generalized EM*.

$$M_*^{n+1} = M^{n+1} \in \mathcal{M} :\ \ Q(M^{n+1}|M^n) \geq Q(M^n|M^n) \quad (9)$$

The priors, π_α^{n+1}, for each α are those that maximize Eq. 3. We focus on maximizing the second term, $\sum_{\alpha=1}^{|V_m|}\sum_{a=1}^{|V_d|} Pr(y_\alpha|x_a, M^n) log\ \pi_\alpha^{n+1}$, of the equation. Differentiating it partially with respect to π_α^{n+1}, and setting the resulting expression to zero results in,

$$\pi_\alpha^{n+1} = \frac{1}{|V_d|}\sum_{a=1}^{|V_d|} Pr(y_\alpha|x_a, M^n)$$

The term $Pr(y_\alpha|x_a, M^n)$ was computed previously in Eq. 4. We use π_α^{n+1} in the next iteration of the E step.

Lexical Similarity

We compute the correspondence error, P_ϵ, between a pair of data graph and model graph nodes (Eq. 7) as one minus the normalized lexical similarity between their respective labels. Under the umbrella of edit distance, several metrics for computing the similarity between strings, such as n-grams, Jaccard, and sequence alignment exist. We use the Smith-Waterman (SW) sequence alignment algorithm (Smith & Waterman 1981) for calculating the lexical similarity between the node labels. The SW algorithm may be implemented as a fast dynamic program and requires a score for the similarity between two characters as the input. We assign a 1 if the two characters in consideration are identical and 0 otherwise. The algorithm generates the optimal local alignment by storing the maximum similarity between each pair of segments of the labels, and using it to compute the similarity between longer segments. We normalize the output of the SW algorithm by dividing it with the length of the longer of the two labels.

Random Sampling with Local Improvements

In this section, we address the computation of the mixture model, M_*^{n+1}, that satisfies the inequality in Eq. 9. We observe that an exhaustive search of the complete model space is infeasible due to its large size – there are $(|V_m| + 1)^{|V_d|}$ many distinct mixture models. On the other hand, both the EM and its generalization, GEM, are known to often converge to the local maxima (Dempster, Laird, & Rubin 1977) (instead of the global) when the search space for selecting M_*^{n+1} is parsimonious. This suggests that any technique for generating M_*^{n+1} should attempt to cover as much of the model space as possible, while maintaining tractability.

A straightforward approach for generating M_*^{n+1} is to randomly sample K mixture models, $\hat{\mathcal{M}} = \{M^{(1)}, M^{(2)}, \cdots, M^{(K)}\}$, and select the one as M_*^{n+1} that satisfies the constraint in Eq. 9. We sample the models by assuming a flat distribution over the model space. The set of samples, $\hat{\mathcal{M}}$, may be seen as a representative of the complete model space. However, since there is no guarantee that a sample within the sample set will satisfy the constraint in Eq. 9, we may have to sample several $\hat{\mathcal{M}}$, before a suitable mixture model is found. This problem becomes especially severe when the model space is large and a relatively small number of samples, K, is used.

In order to reduce the number of $\hat{\mathcal{M}}$s that are discarded, we exploit intuitive heuristics that guide the generation of M^{n+1}. For example, if M^n exhibits mappings between some subclasses in the two graphs, then match their respective parents, to generate a candidate M^{n+1}. For the case where a subclass has more than one parent, lexical similarity is used to resolve the conflict. This and other general and domain-specific heuristics have been used previously in (Doan *et al.* 2002; Mitra, Noy, & Jaiswal 2004) where they were shown to be effective. However, a simple example, Fig. 2, demonstrates that solely utilizing such a heuristic is insufficient to generate all the mappings. To minimize the convergence to local maximas, we augment the set of heuristically generated mixture models with those that are randomly sampled. In this manner, not only do we select candidate mixture models that have a better chance of satisfying Eq. 9, but also cover the model space.

Computational Complexity

We first analyze the complexity of computing $Q(M^{n+1}|M^n)$ which forms the E step. From Eq. 8, the

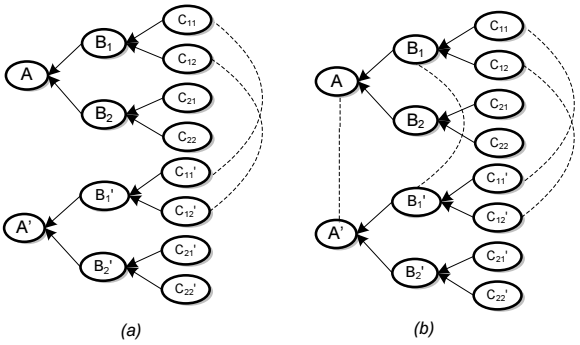


Figure 2: An illustration of the mappings generated by the heuristic – if at least one pair of subclasses is matched, then match the respective parents. (a) M^n (b) Mappings after multiple applications of the heuristic. Unless we combine the heuristic with others, no more mappings can be generated and a local maxima is reached.

complexity of the posterior, $Pr(y_\alpha|x_a, M^n)$, is a combination of the complexity of computing EC, the correspondence error (P_ϵ), and the term $[\frac{1}{Pr(x_a|y_\alpha)}]^{|V_d||V_m|-1}$. We observe that EC may be computed through a series of look-up operations, and is therefore of constant time complexity. The complexity of calculating P_ϵ, dependent on the algorithm for arriving at the lexical similarity, is $O(l^2)$ for the SW technique, where l is the length of the largest concept label. The complexity of the exponential term is $O(\log_2|V_d||V_m|-1)$. Hence the computational complexity of the posterior is $O(\log_2|V_d||V_m|-1)+O(|V_d||V_m|)+O(l^2) = O(|V_d||V_m|)$. The computational complexity of the log likelihood term is also $O(|V_d||V_m|)$, because its computation proceeds analogously. Since the product of these terms is summed over $|V_d||V_m|$, the final complexity of the E step is $O([|V_d||V_m|]^2)$. In the M step, if we generate K samples within a sample set, the worst case complexity is $O(K[|V_d||V_m|]^2)$. [1]

Experiments

We analyze the performance of our methods on example ontology pairs obtained from the I^{3}CON Repository (Hughes & Ashpole 2004). These ontologies are expressed in the N3 language – an experimental non-XML variant of RDF. While, a good match accuracy is of utmost importance, we also focus on the computational resources consumed in arriving at the match. We utilize a partial and modified subset of the Weapons ontologies for a detailed analysis of our methods. In Fig. 3(a) we show the match produced by the GEM algorithm. We utilized the random sampling combined with heuristic improvement to generate M^{n+1} at each step of the iteration. To illustrate the need for considering graph structure while matching, we also show the match obtained by using just the lexical similarity between concept labels (Fig. 3(b)). We point out that the many-one homomorphism in Fig. 3(a) mapped both *PT Boat* and *Missile Boat* nodes of the data graph to the *Fast Attack Craft* node

[1]For edge-labeled graphs, the complexity of the reification step is $O(|E|log_2|E|)$ where $|E|$ is the number of edges in the graph.

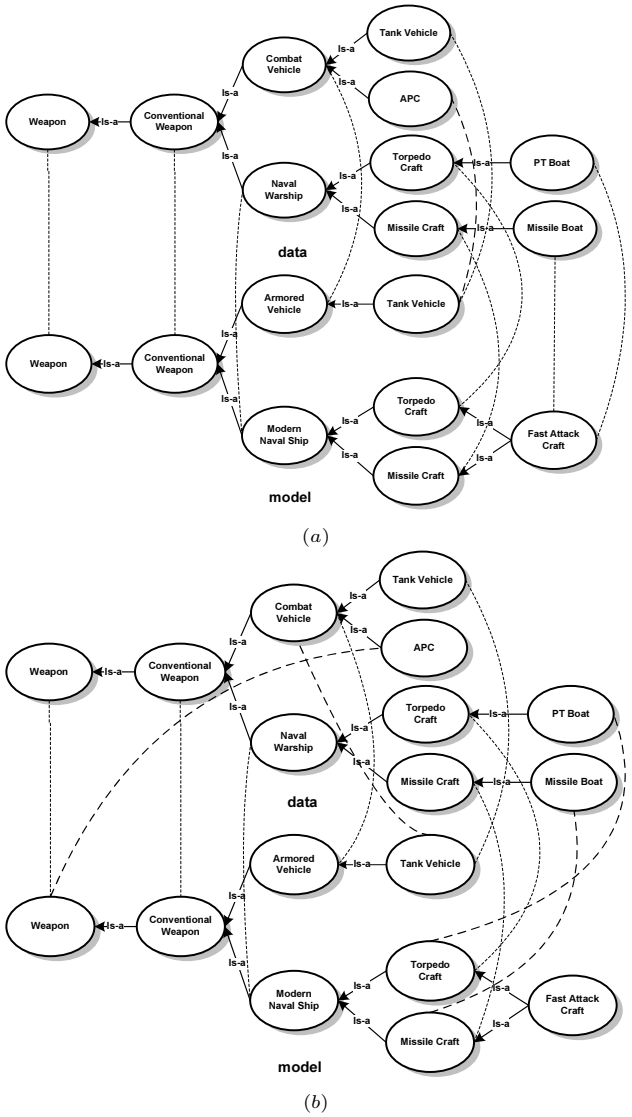
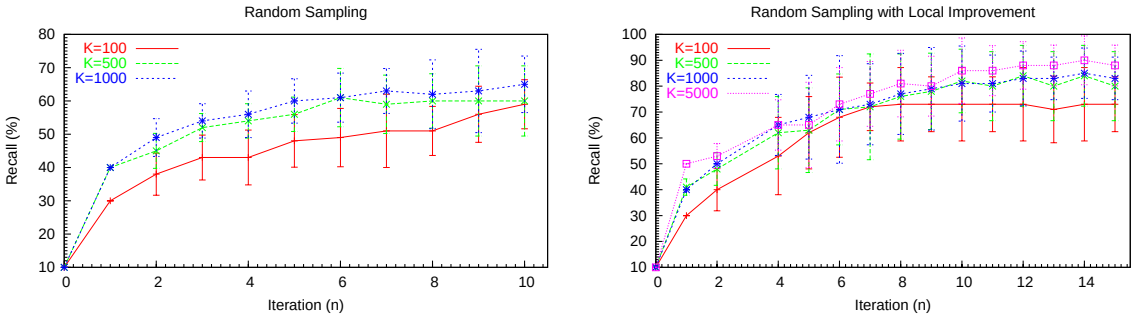


Figure 3: Utility of considering structure while matching. (a) The mappings generated by our GEM method (recall: 100%; precision: 90% – 1 false-positive). (b) Mappings generated using only the lexical similarity between concept labels (recall = 77.8%; precision = 63.6%). The dashed lines in bold are the incorrect matches.

of the model graph. This illustrates a subsumption match because the former concepts are encompassed by the latter.

In Fig. 4(a) we show the performance (recall) profile of the GEM with random sampling. Each data point in the plots is an average of 10 independent runs, and our seed mixture model (M^0) contained a single match between the *Tank Vehicle* nodes of both ontologies. As we increase the size of the sample sets (from 100 to 1,000 samples), we obtain a greater recall at each iteration. This is because a larger percentage of the complete search space ($\approx 10^9$ models) is covered. However, from Fig. 4(c), we observe that the running time over all the iterations also increases as the sample size increases. To measure the effectiveness of random sampling, we also provide the total number of sample sets, $\hat{\mathcal{M}}$, that were generated over all the iterations. For 100 samples, an average



(a) (b)

Method	Metric	K=100	K=500	K=1000	K=5000
Random sampling	Time	3m 39.29s ± 4m 19.09s	19m 26.46s ± 19m 24.5s	23m 25s ± 16m 42.63s	*
	Sample sets	570 ± 666	373 ± 368	376 ± 264	*
Sampling with local improvements	Time	10.3s ± 1.7s	48.12 ± 11.2s	1m 50.6s ± 24.11s	8m 41.3s ± 1m 22.46s
	Sample sets	26 ± 5	25 ± 6	27 ± 6	27 ± 4

(c)

Figure 4: Performance profiles of the GEM method on Weapons ontologies. (a) Random sampling was used for generating the next M^{n+1}. (b) A combination of heuristic and random sampling is used for generating M^{n+1}. (c) The total running times (JDK 1.5 program on a dual processor Xeon 2.1GHz, 4GB RAM, and Linux) and sample sets generated over all the iterations.

of 57 sample sets per iteration were generated before a satisficing M^{n+1} was found, while an average of 38 sample sets were used per iteration for 1,000 samples. On including the heuristic (Fig. 2) in addition to random sampling during the optimization step, we obtain the performance profiles shown in Fig. 4(b). The heuristic not only improves the recall but also significantly reduces the number of sample sets generated and therefore the time consumed in performing the iterations (Fig. 4(c)). While smaller size sample sets lead to local maximas, sets of 5,000 samples produced 90%-100% recall for all the runs. We observe that the heuristic by itself is not sufficient: starting from our seed model, employing just the heuristic for the optimization step leads to only a 40% recall.

In order to judge the performance of our method on more complex ontologies, we tested it on larger subsets of the Weapons ontologies, rooted at *Conventional Weapons*, and subsets of the Network ontologies. The data and model graphs for Weapons contained 22 and 19 nodes, respectively. The GEM method combined with heuristically and randomly generated samples converged to a match with a recall of 100% and a precision of 86.4% in 17m 47.2s with a seed model of 10% accurate matches. The Network ontologies, in addition to containing more nodes, exhibit labeled relationships. After performing reification using the procedure illustrated in Fig. 1, both the data and model graphs contained 19 nodes and 24 edges each. The GEM method converged to a match with a recall and precision of 100% in 10m 53.1s with a seed model of 10% accuracy.

Conclusion and Limitations

We presented a new method for identifying mappings between ontologies that model similar domains. We formulated the problem as one of finding the most likely map and solved it iteratively using the GEM technique. We improved on previous approaches by generating inexact matches between the ontologies; such methods have a wider applicability. Our results illustrate the good performance of our methods, but also highlight some limitations. In particular, for large ontologies, more efficient methods are required for performing the optimization step.

Acknowledgments

This research is supported by a grant from UGARF.

References

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* 39:1–38.

Ding, Z.; Peng, Y.; Pan, R.; and Yu, Y. 2005. A Bayesian Methodology towards Automatic Ontology Mapping. C & O, AAAI.

Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to map between ontologies on the semantic web. In *WWW*, 662–673. ACM Press.

Hayes, J., and Gutierrez, C. 2004. Bipartite graphs as intermediate models for rdf graphs. In *ISWC*.

Hu, W.; Jian, N.; Qu, Y.; and Wang, Y. 2005. Gmo: A graph matching for ontologies. In *Integrating Ontologies Wksp., K-CAP*.

Hughes, T., and Ashpole, B. 2004. Information interpretation and integration conference. http://www.atl.lmco.com/projects/ontology/i3con.html.

Kalfoglou, Y., and Schorlemmer, M. 2003. If-map: an ontology mapping method based on information flow theory. *Journal on Data Semantics* 1(1):98–127.

Luo, B., and Hancock, E. 2001. Structural graph matching using the em algorithm and singular value decomposition. *Graph Algorithms and Computer Vision* 23(10):1120–1136.

Mitra, P.; Noy, N.; and Jaiswal, A. 2004. OMEN: A probabilistic ontology mapping tool. In *Meaning, Coord. and Neg., ISWC*.

Smith, T. F., and Waterman, M. S. 1981. Identification of common molecular subsequences. *J. of Mol. Biology* 147:195–197.

Stumme, G., and Maedche, A. 2001. FCA-MERGE: Bottom-up merging of ontologies. In *IJCAI*, 225–234.