Markov Network based Ontology Matching *

Sivan Albagli
Dept. of Computer Science
Ben-Gurion University
albagli@cs.bgu.ac.il

Rachel Ben-Eliyahu-Zohary
JCE and Ben-Gurion University
Israel
rachel@bgu.ac.il

Solomon E. Shimony
Dept. of Computer Science
Ben-Gurion University
shimony@cs.bgu.ac.il

Abstract

iMatch is a probabilistic scheme for ontology matching based on Markov networks, which has several advantages over other probabilistic schemes. First, it uses undirected networks, which better supports the non-causal nature of the dependencies. Second, it handles the high computational complexity by doing approximate reasoning, rather then by ad-hoc pruning. Third, the probabilities that it uses are learned from matched data. Finally, iMatch naturally supports interactive semi-automatic matches. Experiments using the standard benchmark tests that compare our approach with the most promising existing systems show that iMatch is one of the top performers.

1 Introduction

Ontology matching has emerged as a crucial step when information sources are being integrated, such as when companies are being merged and their corresponding knowledge bases are to be united. As information sources grow rapidly, manual ontology matching becomes tedious, time-consuming and leads to errors and frustration. Therefore, automated ontology matching systems have been developed. In this paper, we present a probabilistic scheme for interactive ontology matching based on Markov networks, called iMatch.

The first approach for using inference in structured probability models to improve the quality of existing ontology mappings was introduced as OMEN in [Mitra *et al.*, 2005]. OMEN uses a Bayesian network to represent the influences between potential concept mappings across ontologies. Some of our ideas are based on [Mitra *et al.*, 2005] - the method we present herein also contains inference over networks, albeit with several improvements. First, iMatch uses Markov Networks rather than Bayesian Networks. This representation is more natural since there is no inherent causality in ontology matching. Second, iMatch uses approximate reasoning to confront the formidable computation involved rather than arbitrary pruning as done by OMEN.

*Supported by the IMG4 consortium under the MAGNET program of the Israel ministry of trade and industry; and the Lynn and William Frankel center for computer science.

Third, the clique potentials used in the Markov networks are learned from data, instead of being provided in advance by the system designer. Finally, iMatch performs better than OMEN, as well as numerous other methods, on standard benchmarks.

In the empirical evaluation, we use benchmarks from the Information Interpretation and Integration Conference (I3CON).[1] These standard Benchmarks are provided by the initiative to forge a consensus for matching systems evaluation, called the Ontology Alignment Evaluation Initiative (OAEI), on which we compare iMatch with the state of the art ontology matching systems such as Lily [Wang and Xu, 2008], ASMOV [Jean-Mary and Kabuka, 2008], RiMOM [Zhang *et al.*, 2008]. Our experiments show that iMatch is competitive with, and in some cases better than, the best ontology matching systems known so far.

2 Background

2.1 Ontology Matching

We view ontology matching as the problem of matching labeled graphs. That is, given ontologies O and O', (see Figure 1) plus some additional information I, find a mapping m between nodes in O and nodes in O' that is *optimal* in some respect. A difficulty in the field of ontology matching is that a formal definition of optimality is lacking in general. There are criteria for what makes a "good" match [Do *et al.*, 2002], and in some cases even formal definitions of match qualities [Jerome Euzenat, 2007], but there is no agreed upon global standard. Therefore, matching schemes are usually scored based on what a human expert would do, given the same type of input - assumed to be "ground truth", an evaluation scheme adopted in this paper.

The labels in the ontology convey information, as they are usually sequences of (parts of) words in a natural language. The additional information I is frequently some knowledge base containing words and their meanings, how words are typically abbreviated, and so on. In addition, in an interactive setting, user inputs (such as partial matching indications: a human user may be sure that a node labeled "address" in O should match the node labeled "location" in O') can also be seen as part of I. In this paper, we focus on performing what

[1] http://www.atl.external.lmco.com /projects/ontology/i3con.html

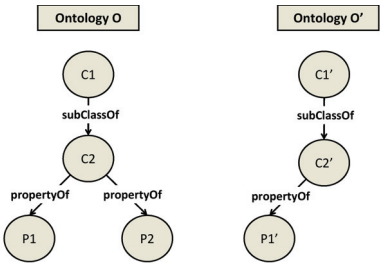

Figure 1: Two ontologies, O and O', to be matched.

is sometimes called a "second line matching", that is, we assume that a similarity measure over pairs of nodes (o, o') is provided. Then, we use the structure of the ontologies, in conjunction with type information and the evidence provided by the similarity measure (and possibly a user) to compute the matching. The initial similarity is usually provided by a first-line matcher.

In general, an ontology mapping m can be a general mapping between subsets of nodes in O and subsets of nodes in O'. However, in most schemes m is constrained in various ways. One commonly used **constraint** is requiring m to be **one to one**. This paper for the most part assumes the **one to one constraint**, but we also show how the proposed model can be generalized to mappings that are one to many or many to one.

For simplicity, we use a rather simple standard scheme of ontology representation, where the nodes of the ontology graph (see figure 1) denote entities such as classes, properties, and ranges. Arcs denote relationships $R(o_1, o_2)$ between entities, such as subClassOf (used to construct the class hierarchy), instanceOf, etc. For example, in Figure 1 we have subClassOf(C_2, C_1) in ontology O. Properties are objects in the ontology that describe attributes of classes. Range is an instance of Property that is used to state that the values of a property are instances of one or more classes. Domain is an instance of Property that is used to state that any resource that has a given property is an instance of one or more classes. Figure 1 depicts an ontology O on the left-hand side, and an ontology O' on the right-hand side, that we might wish to match. Given the structure of the ontologies, it seems likely that $m(C_1, C_1')$. We use the following notation conventions throughout this paper: all concepts from O are unprimed, and those from O' are primed. C denotes a class, and P is a property. In order to denote confidence in a match, $Pr(m(C, C'))$ denotes the probabiliy that C matches C'.

2.2 Markov Networks

Markov networks are structured probability models [Pearl, 1988], used to compactly represent distibutions over random variables. Like their directed counterpart, Bayesian networks, Markov network structure is a graph $G = (V, E)$, where each node $v \in V$ stands for a random variable, and each edge $e \in E$ represents a direct statistical dependency between the variables represented by its incident nodes (Figure 2).

However, unlike Bayesian networks, edges in Markov networks are *undirected*, and additionally do not imply a causal influence, which is a commonly used interpretation of the semantics of edges in Bayes networks. The distribution in a Markov network is defined by *potential functions* over the cliques of G. Let C be a clique of G. The potenial over C is a function p_C from the cross-product of the domains of variables represented by the nodes in C to $[0, 1]$. The value of p_C for a specified value assignment A of the clique variables is a probability of occurence of the event where the variables get a value as specified by A. One example of a potential function over the 2-clique of binary variables $\{m(C_1, C_1'), m(C_2, C_2')\}$ appears in Table 1.

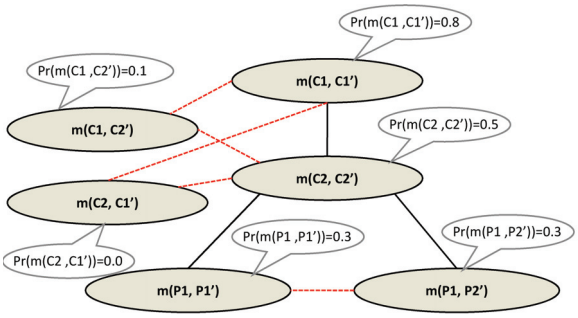

Figure 2: Markov network for matching O and O'

In order to allow for an unconstrained definition of the potential functions, the potentials are assumed to be unnormalized probabilities, and require a normalization using a global *partition function* Z, an issue beyond the scope of this paper. The distribution defined by a Markov network is:

$$P(V) = \frac{1}{Z} \prod_{C \in cliques(G)} p_C(C) \quad (1)$$

In this paper, probabilities in potential functions stand for *compatibility* of the values of the random variables, as detailed below. We will be using the probabilities to compute the probability that a given pair of ontology nodes (o, o') match.

3 Probabilistic Matching Scheme

When creating a probabilistic scheme, one should first define the semantics of the distribution based on real-world distributions. Ideally, one should have a generative model for the distribution in question, as has been done in various domains, such as natural language understanding and plan recognition [Charniak and Goldman, 1994]. Although we have not been able to come up with a generative model we would want the probability that a match (o, o') occur as defined in the model, to be the same as what we would expect would occur if the same evidence were presented to a human expert.

In order to achieve this, we use standard common sense rules, such as: "when (o, o') match, then the (respective) parents of these nodes frequently match as well", to define the dependencies and potential functions. The distribution model is also used to encode constraints on the mapping, such as the **one-to-one constraint**.

Our matching scheme as a whole works as follows.

1. Given the ontologies O and O', construct a problem instance specific Markov network $N(O, O')$.

2. Using a first-line matcher, find initial match distributions for all possible pairs (o, o'), and use them to initialize evidence potentials.
3. Perform probabilistic reasoning in N in order to compute a second-line match.

3.1 Constructing the probabilistic model

Since a-priori any element of O can match any element of O', we need to model this as a possible event in the probabilistic model. Thus, we have one binary variable (node) in the network for each possible (o, o') pair. The topology of the network is defined based on the following common-sense rules and constraints:

1. The matching is one to one.
2. If concepts $c_1 \in O$, $c'_1 \in O'$ match, and there is a relationship $R(c_1, c_2)$ in O, and $R(c'_1, c'_2)$ in O', then it is likely that c_2 matches c'_2.

The first rule is encoded by making cliques of the following node-sets (required as in this case the indicated matching events below are all directly statistically dependent). Consider node $v = (o, o')$. This node participates in a clique consisting of all nodes $\{(o, o'')|o'' \in O'\}$, i.e. the nodes standing for all possible matches of o. This setup will allow the model to control the number of matches for o. Likewise, v also participates in a clique consisting of the nodes: $\{(o'', o')|o'' \in O\}$, i.e. the nodes standing in for all possible matches of o', allowing control over the number of matches for o'. For example, part of the Markov network constructed for the ontologies O and O' from Figure 1 is depicted in Figure 2. The dotted arcs represent cliques of nodes where each clique contains the same source node, such as the 2-clique consisting of $m(C_1, C'_1)$ and $m(C_1, C'_2)$. The actual control over number of matches is achieved by specifying an appropriate potential function over these cliques, as discussed below.

The second rule (which actually results from packing 2 rules adapted from OMEN [Mitra *et al.*, 2005]), has several sub-cases, depending on the nature of the relation R. Two special cases considered here are the subClassOf relation, and the propertyOf relation. In the former case, if two classes match, the probability of match between each pair of their sub-classes might change (usually increase), according to the strength of the appropriate dependency. A similar argument holds for the case where the relationship is one between a class and one of its properties. In both cases, the dependency is represented by an arc between the node $m(C_1, C'_1)$ and the node $m(C_2, C'_2)$. For example, in Figure 2, this type of relationship between the Markov network nodes is shown by solid arcs. The nature of the dependency (such as correlation strength between these matches) is encoded in the potential function over this 2-node clique, as in Table 1.

One can consider more complicated rules, such as introducing context dependency (e.g. the fact that superclasses match increase the probability that the respective subclasses match may depend on the type of the superclass, or on the number of subclasses of each superclass, etc.). The iMatch scheme can easily support such rules by defining potential functions over the respective sets of nodes, but in this version such rules have not been implemented - leaving this issue for future work.

		$m(C_1, C'_1)$	
		T	F
$m(C_2, C'_2)$	T	0.19	0.25
	F	0.063	0.55

Table 1: Potential function for rule-derived arcs.

Defining the clique potentials

For each of the cliques types discussed above, we need to define the respective potential functions (also called factors). We begin by describing what these potentials denote, and how they should be determined. Later on, we discuss how some of the potential function values can be learned from data.

The simplest to define are clique potentials that enforce the one-to-one constraint. Each such clique is defined over k binary nodes, at most one of which must be true. Barring specific evidence to the contrary, the clique potential is thus defined as: 1 for each entry where all nodes are false, a constant strictly positive value **a** for each entry where exactly one node is true, and 0 elsewhere (denoting a probability of zero for any one to many match). The value of the constant **a** is determined by the prior probability that a randomly picked node o matches *some* node in o', divided by the number of possible candidates. In our implementation, the potentials for these cliques were defined by implicit functions, as all but a number linear in k of the values are zero; in order to avoid the 2^k space and time complexity for these cliques. As an indication of the flexibility of our scheme, observe that in order to allow multiple matches all that need be done is to change the potential in this clique, to reflect the probability that a multiple match occur.

The second type of clique potential is due to rules-based arcs (solid arcs in Figure 2). This is a function over 2 binary variables, requiring a table with 4 entries. As indicated above, a match between concepts increases the probability that related concepts also match, and thus intuitively should have main diagonal terms higher than off-diagonal terms. We could cook up reasonable numbers, and that is what we did in a preliminary version. We show empirically that results are not too sensitive to the exact numbers. However, it is a better idea to estimate these numbers from training data, and even a crude learning scheme performs well. We have tried several learning schemes, but elected to use a simple maximum likelihood parameter estimation. For the rule-derived clique potentials, we simply calculated the likelihood of the data, by counting the number of instances for each one of the observations, resulting in potentials reported in Table 1. In retrospect, these numbers do make sense, even though they did not conform with our original intuitions. This is due to the fact that the "no match" entries are far more frequent in the data, as most candidate matches are false.

The third type of clique potentials is used to indicate evidence. Two types of evidence are used: indication of prior beliefs about matching (provided by a first-line matcher), and evidence supplied by user input during interaction. These can

be added as single-node potentials, as indicated by the probabilities in Figure 2. However, in the actual implementation this was done by adding a dummy node and potentials over the resulting 2-clique, which is a standard scheme for introducing evidence [Pearl, 1988]. We support evidence of the form: $Pr(m(o, o')) = x$, where x is the probability of the match (o, o') according to the first-line matcher or the user. Observe that adding several sources of evidence (whether mutually independent or dependent) for the same node can also be supported in this manner.

3.2 Reasoning in the probabilistic model

Given a probabilistic model, there are several types of probabilistic reasoning that are typically performed, the most common being computation of posterior marginals (also called belief updating), and finding the most probable assignment (also called most probable explanation (MPE), or belief revision) [Pearl, 1988]. Both of these tasks are intractable, being NP-hard in the general case. And yet there are a host of algorithms that handle large networks, that have a reasonable runtime and deliver good approximations in practice. Of particular interest are sampling-based schemes [Pearl, 1988] and loopy belief propagation [Kschischang *et al.*, 2001], on which our experimental results are based. A match is reported for any (Markov) network node that has a posterior probability of being true greater than a given "decision threshold", which could either be user-defined (as in Section 4) or based on a decision-theoretic scheme, e.g. based on penalty for making a false match vs. bonus for a correct match. Note, however, that our scheme is not committed to these specific algorithms.

iMatch supports user interaction quite naturally, as follows. User input is simply added as evidence in the Markov network, after which belief updating is perfromed. There is no need to change any of the algorithms in the system in order to do so. The evidence provided by the user can be encoded easily in the clique potentials. Currently, we use the (2×2 identity) potential matrix, thus assuming that the user is always correct, but changing it to one that has non-zero off-diagonal entries allows for handling user errors.

4 Empirical Evaluation

In order to evaluate our approach, we used the benchmark tests from the OAEI ontology matching campaign 2008[2]. We followed the standard evaluation criteria from the OAEI campaign, calculating the precision, recall and f-measure over each test. The version computed here is the harmonic mean of precision and recall.

4.1 Comparison to existing systems

For each test, we computed the prior probability of each matching event using edit distance based similarity as the "first-line matcher" [Wagner and Fischer, 1974]. we then performed belief updating in the network using loopy belief propagation. In experiments 1 and 2, the results were evaluated against a reference alignment of each test. Finally we compared the f-measure with other systems on each test. Experiment (1): The Conference test suite from the 2008 OAEI ontology matching campaign was used. The OAEI Conference track contains realistic ontologies, which have been developed within the OntoFarm project. Reference alignments have been made over five ontologies (cmt, confOf, ekaw, iasted, sigkdd). Some statistics on each of these is shortly described in Table 2. Figure 3 is a comparison of the matching quality of our algorithm and the other 3 systems, computed with two thresholds over the posterior match probability: 0.5 and 0.7 (thresholds picked to fit theresholds reported for the competing systems). As is evident from the figure, results of iMatch depend on the decision threshhold, but not too sharply. iMatch out-performed the competition here on a range of thresholds.

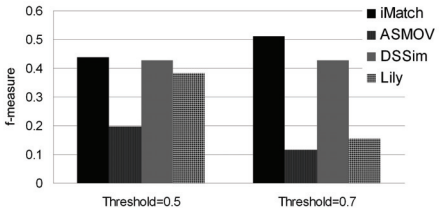

Figure 3: Results for the Conference test ontologies.

Name	Number of Classes	Number of Properties
Ekaw	77	33
Sigkdd	49	28
Iasted	140	41
Cmt	36	59
ConfOf	38	36

Table 2: The conference ontologies.

Experiment (2): Here we used the benchmark test samples suite from the 2008 OAEI ontology matching campaign. The benchmarks test case includes 51 ontologies in OWL, The different gross categories are summarized in Table 3. Figure 4 compares the outcome quality of our algorithm to 14 othe systems. One of the schemes, EDNA, is the edit distance scheme alone, also the first-line matcher input to iMatch. As is evident from the figure, iMatch is one of the top performers in most of the categories, and the best in some of them. A notable exception is the set of ontologies requiring linguistic techniques, in which iMatch, as well as many other systems, did badly, as iMatch knows nothing about language techniques (and systems that did well there do). One way to improve iMatch here would be to use evidence from linguistic matchers - the probabilistic scheme is sufficiently general that it should be possible to use this type of evidence as well, in future research.

Tests	Description
# 101-104	$O,\ O'$ have equal or totally different names
# 201-210	$O,\ O'$ have same structure, different linguistic level
# 221-247	$O,\ O'$ have same linguistic level, different structure
# 248-266	$O,\ O'$ differ in structure and linguistic level
# 301-304	O' are real world cases

Table 3: The benchmark test samples suite.

[2]http://oaei.ontologymatching.org/2008/

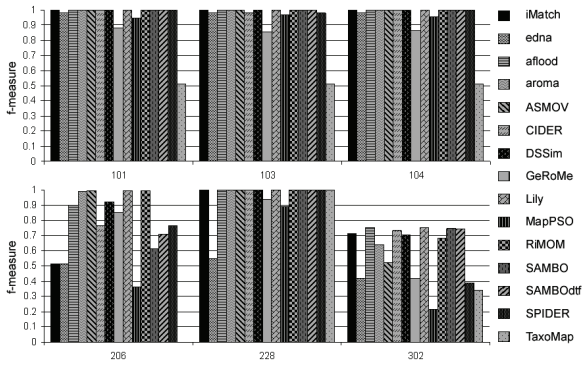

Figure 4: Results for the benchmarks test ontologies.

4.2 Auxiliary experiments

Better understanding of tradeoffs and parameters in iMatch would greatly facilitate future enhancements. Several auxiliary experimental results are of special interest, described very briefly below due to lack of space. First, we wish to evaluate the sensitivity of the results to the clique potentials learned from data, as one could not expect to get the exact right numbers for an unknown ontology. (In our case, we learned the numbers using matching results from 2 pairs of ontologies, and applied them throughout all the other experiments.)

In one experiment we tweaked each of the parameters of Table 1 in iMatch (the (T,T), (T,F), (F,T), (F,F) entries) by 0.1 to 0.3 in various ways. For comparison, the parameters resulting from hindsight - i.e. the frequency of occurences of the correct matching, are also tested. Indeed the best F-measure occurs for the unachivable correct potentials, as shown in figure 5, but for the actual learned parameters the result was a change in the F-measure by approximately 0.06 in most cases for the "Hotel" data set (I3CON). For most other data sets, the effect was even smaller.

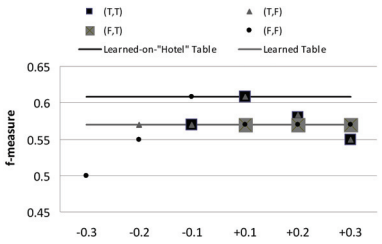

Figure 5: Sensitivity to clique potentials.

Finally, although some of the benchmark ontologies are from real applications, many realistic ontologies are orders of magnitude larger, resulting in unacceptable runtimes for iMatch. This is an issue left for future research. Nevertheless, we did a preliminary evaluation in this direction on the quality of results versus the computation time, which is easily traded off in loopy belief propagation if the run is stopped before convergence. It turns out that even though convergence was achieved in most cases after 7-10 cycles of propagation, most of the benefit by far occured on the *first* propagation for most of the datasets. This may be sufficient for many applications, especially when we have a human in the loop.

5 Related work

Several researchers have explored using Bayesean Networks for ontology matching [Mitra *et al.*, 2005; Doan *et al.*, 2002]. The Bayesian Net of OMEN [Mitra *et al.*, 2005] is constructed on-the-fly using a set of meta-rules that represent how much each ontology mapping affects other related mappings, based on the ontology structure and the semantics of ontology relations. iMatch follows the same general scheme, except for the differences discussed above. Moreover, iMatch preforms better than OMEN on most of the I3Con benchmarks (not shown due to lack of space).

Another scheme sets up matching equations [Udrea and Getoor, 2007] taking into account various features, including neighboring matches, followed by finding a fixed point of the system, used to determine the match. In some sense, this scheme is similar to iMatch, since using loopy belief propagation also finds a fixed point to a set of equations. However, in iMatch we have an explicit probabilistic semantics, the fixed point being an artefact of the approximation algorithm we happen to use, rather than a defining point of the scheme.

The GLUE system [Doan *et al.*, 2002] employs machine learning algorithms that use a Bayes classifier for ontology matching. This approach, however, does not consider relations between concepts, as iMatch does.

Many ontology mappers, as presented in Hovy [Hovy, 1998], PROMPT [Noy and Musen, 2000] and ONION [Mitra *et al.*, 2001], use rule-based methods. Examples of methods that look for similarities in the graph structure include Anchor-Prompt [Noy and Musen, 2002] and Similarity Flooding [Melnik *et al.*, 2002]. Other systems for ontology matching are referred to in Section 4. The three top-ranked systems from the OAEI campaign 2008, i.e. RiMOM [Zhang *et al.*, 2008], LILY [Wang and Xu, 2008] and ASMOV [Jean-Mary and Kabuka, 2008], differ from iMatch as they use graphical probabilistic models combined with other tools. RiMOM is a general ontology mapping system based on Bayesian decision theory. It utilizes normalization and NLP techniques and uses risk minimization to search for optimal mappings from the results of multiple strategies. LILY is a generic ontology mapping system based on the extraction of semantic subgraphs. It exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. Then a similarity propagation strategy is applied to produce more alignments if necessary. ASMOV is an automated ontology mapping tool that iteratively calculates the similarity between concepts in ontologies by analyzing four features, i.e., textual description, external structure, internal structure, and individual similarity. It then combines the measures of these four features using a weighted sum. Note that in the conference tests (see Figure 3), we preform better than the above three top-ranked systems.

Many of the methods above produce alignments with some degree of certainty and thus can be integrated into iMatch by providing prior probabilities.

6 Conclusion

iMatch is a novel probabilistic scheme for ontology matching, where a Markov network is constructed on the fly according to the two input ontologies; evidence from first-line matchers is introduced, and probabilistic reasoning is used to produce matchings. Our current implementation uses loopy belief propagation [Kschischang *et al.*, 2001] to meet the inherent computational complexity, although any other belief updating algorithm could be used. Evidence from other sources like human experts, or other matchers, can be easily integrated into the loop, and hence our system is inherently interactive.

Experiments show that our system is competitive with (and frequently better than) many of the top matchers in the field. We are currently working on improving the scalability of iMatch, which is facilitated by the anytime behavior of the reasoning algorithms, and on integrating other sources of evidence in order to enhance the alignments produced by iMatch.

References

[Charniak and Goldman, 1994] Eugene Charniak and Robert P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 1994.

[Do *et al.*, 2002] Hong Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In *Web, Web-Services, and Database Systems*, pages 221–237, 2002.

[Doan *et al.*, 2002] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Learning to map between ontologies on the semantic web. In *WWW*, pages 662–673, 2002.

[Hovy, 1998] E. Hovy. Combining and standardizing largescale, practical ontologies for machine translation and other uses. page 535542, 1998.

[Jean-Mary and Kabuka, 2008] Yves R. Jean-Mary and Mansur R. Kabuka. Asmov results for oaei 2008. In *Ontology Alignment Evaluation Initiative*, 2008.

[Jerome Euzenat, 2007] Pavel Shvaiko Jerome Euzenat. *Ontology Matching*. Springer Verlag, Berlin Heidelberg, 2007.

[Kschischang *et al.*, 2001] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[Melnik *et al.*, 2002] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[Mitra *et al.*, 2001] Prasenjit Mitra, Gio Wiederhold, and Stefan Decker. A scalable framework for the interoperation of information sources. In *SWWS*, pages 317–329, 2001.

[Mitra *et al.*, 2005] Prasenjit Mitra, Natasha F. Noy, and Anuj R. Jaiswal. Omen: A probabilistic ontology mapping tool. In *International Semantic Web Conference*, pages 537–547, 2005.

[Noy and Musen, 2000] Natalya Fridman Noy and Mark A. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *AAAI/IAAI*, pages 450–455, 2000.

[Noy and Musen, 2002] Natalya F. Noy and Mark A. Musen. Anchor-prompt:using non-local context for semantic matching. In *IJCAI*, 2002.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Udrea and Getoor, 2007] Octavian Udrea and Lise Getoor. Combining statistical and logical inference for ontology alignment. In *IJCAI*, 2007.

[Wagner and Fischer, 1974] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.

[Wang and Xu, 2008] Peng Wang and Baowen Xu. Lily: Ontology alignment results for oaei 2008. In *Ontology Alignment Evaluation Initiative*, 2008.

[Zhang *et al.*, 2008] Xiao Zhang, Qian Zhong, Juanzi Li, Jie Tang, Guotong Xie, and Hanyu Li. Rimom results for oaei 2008. In *Ontology Alignment Evaluation Initiative*, 2008.