Ontology matching based on Probabilistic Description Logic[*]

ZhiMing Li
College of Computer Science
Zhejiang University
Hangzhou, P.R.China 310027

Shanping Li
College of Computer Science
Zhejiang University
Hangzhou, P.R.China 310027

Zhiyu Peng
College of Computer Science
Zhejiang University
Hangzhou, P.R.China 310027

Abstract: - A lot of attention has been devoted to probabilistic methods for discovering semantic mappings between ontologies. Despite impressive theory foundation, these methods usually require massive data instances to learn the mappings, which are not always available in practice. In this paper we present the Probabilistic Description Logic based Ontology Matcher (PDLOM) for discovering such mappings using the inference service provided by Probabilistic Description Logic (P-CLASSIC), which allows for computing the probability of a concept description in an ontology. Unlike other probability based approaches, PDLOM only needs a probability distribution over primitive concepts instead of massive data instances of all concepts in the ontologies. We propose to exploit a search engine to get the probability distribution required by Probabilistic Description Logic. We evaluate our algorithm and compare against the schema matching tool COMA++. PDLOM shows an average improvement of 6% in quality over COMA++.

Key-Words: - Ontology matching, Probabilistic Description Logic, Bayesian networks, Search engine, Page count

1 Introduction

Ontologies play a crucial role in the emerging Semantic Web. However, there always exist multiple ontologies for overlapped domains and even for the same domain due to the decentralized nature of the Web. Therefore, ontology matching, or ontology alignment, is necessary to establish interoperation between Web applications using different ontologies. Informally, the ontology matching problem can be stated as: given two ontologies O_1 and O_2, determine correspondences between entities (classes, properties, and individuals) in the two ontologies.

Many diverse solutions to the matching problem have been proposed so far [13]. Among them methods based on probability become prevalent in concept mapping between two ontologies. Semantic similarities between concepts are difficult to be represented logically, but can easily be represented probabilistically. This has motivated development of ontology mapping taking probabilistic approaches (HICAL [5], GLUE [3], oPLMap [11]). However, these approaches need massive data instances of the ontologies to learn the mappings. But unfortunately, sometimes the ontologies do not come along with that many data instances. This makes all these approaches not always applicable.

This paper proposes a new approach called Probabilistic Description Logic based Ontology Matcher (PDLOM), for automatically discovering the mappings among ontologies. Our approach is based on Probabilistic Description Logic (P-CLASSIC) [7]. Using Probabilistic Description Logic bears some nice features: First, in many cases mappings are not absolutely correct, but hold only with a certain probability. Like other probability based approach, defining mappings by means of probabilistic rules is a natural solution to this problem. Second, Probabilistic Description Logic provide inference services to compute the probability P(C) of a concept description C, which defines the probability of instances in the domain belong to the concept C, from the probability distribution over primitive concepts. Unlike other probability based approaches, it does not require massive data instances of all the concepts in the ontologies but only a probability distribution over the primitive concepts.

The contributions of this paper are as follows:

1. We first propose to employ the inference service of Probabilistic Description Logic to compute the semantic similarity between concepts from different ontologies.

2. We propose to exploit a search engine to extend the ontologies with p-classes defined in Probabilistic Description Logic.

[*] This work is supported by National Natural Science Foundation of China (No. 60473052) and Zhejiang Provincial Natural Science Foundation of China (No.Y106427)

The paper is structured as follows: The next section briefly introduces the basic definitions of Probabilistic Description Logics and explanations of the reasoning mechanisms. Section 3 presents a theoretically founded approach for finding these mappings, where the inference service of Probabilistic Description Logics is employed. In section 4, we report on some preliminary experimental evaluation of the approach proposed in this paper and summarize the results.

2 Background

Description Logics (DLs) are a family of class (concept) based knowledge representation formalisms. Modern ontology languages, such as OWL [12], are based on description logics and, to a certain extent, are syntactic variants thereof. In particular, OWL DL corresponds to $\mathcal{SHOIN}$ [13]. In this paper, we assume an ontology O based on a description logic L to be a set of axioms in L.

For exposition, suppose we have got the two ontologies in Table 1. We divide the concepts occurring in ontologies into two sets, the name symbols that occur on the left-hand side of some axiom and the base symbols that occur only on the right-hand side of axioms. Name symbols are often called defined concepts and base symbols primitive concepts. For example, Man, Rich and Old are primitive concepts in O_1.

Table 1: Two example ontologies

O_1	RichOldMan ≡ Man∩Rich∩Old
O_2	HappyFather ≡ Father∩Rich
	Father ≡ Man∩≥1hasChildren

In classical Description Logics, one has very restricted means of expressing (and testing for) relationships between concepts. Given two concepts C and D, subsumption tells us whether C is contained in D, and the satisfiability test (applied to C∩D) tells us whether C and D are disjoint. Relationships that are in-between (e.g., 90% of all Cs are Ds) can neither be expressed nor be derived. As for the two ontologies in Table 1, RichOldMan in O_1 and HappyFather in O_2 are not equivalent, but there are chances that an instance of RichOldMan also belongs to HappyFather.

This deficiency is overcome in Probabilistic Description Logic [7] by presenting a framework for the specification of a unique probability distribution on the set of all concept descriptions. The probabilistic component of Probabilistic Description Logic is called p-class (probabilistic class), which specifies the probability distribution of a certain set of individuals over primitive concepts and roles. A p-class is represented using a Bayesian network. In general the Bayesian network contains a node for each primitive concept and each role R.

To compute the probability P(C) of a concept description C, which defines the probability of individuals in the domain belongs to the concept C, the basic idea is to use the canonical form of a description. A canonical form of a description is $\alpha \cap \beta R_1 \cap \beta R_2 \cdots \cap \beta R_m$, where α is a conjunction of primitive concepts and their negations and of filler specifications (with no concept or attribute appearing more than once), andβR is of the form $(\geq mR) \cap (\leq nR) \cap (\forall R.C)$, where C is also in canonical form. For example, the canonical form of the concept HappyFather in O_2 is Man∩Rich∩≥1hasChildren.

The depth of a description is defined as follows. The depth of α is 0. The depth of a concept $\alpha \cap \beta R_1 \cap \beta R_2 \cdots \cap \beta R_m$, is 1 + Max (depth (α), depth (βR_1),⋯, depth (R_m)). The way of providing a unique probability distribution on the set of all concept descriptions is to specify a distribution on concepts of role-depth 0, and then to specify how to extend a distribution on concepts of role-depth n to one on concepts of role-depth n + 1.

So with the specification of the probability distribution over primitive concepts and roles, the probability P(C) of a concept description C can then be computed by using inference algorithms developed for Bayesian networks. The complexity of this computation is linear in the length of C. Under certain restrictions on the Bayesian networks used in the specification, it is polynomial in the size of that specification.

3 The PDLOM approach

The matching problem is to find correspondences between entities in the two ontologies. We attack the ontology matching problem by computing similarity coefficients between concepts of the two ontologies and then deducing a mapping from those coefficients. The coefficients, in the [0, 1] range, are calculated in two phases. The first phase, called linguistic matching, matches individual concepts based on their names. The result is a linguistic similarity coefficient, *lsim*, between each pair of concepts (see detailed definition of *lsim* in section 3.1).

The second phase is the matching of ontologies based on reasoning about the probability of a concept description in Probabilistic Description Logics introduced in last section. The result is a Jaccard similarity coefficient, *jsim*, for each pair of concepts (see detailed definition of *jsim* in section 3.2).

The weighted similarity (*wsim*) is a mean of *lsim* and *jsim*: *wsim* = *w* × *jsim* + (1 - *w*) × *lsim*, where the constant w is in the range 0 to1. A mapping is created by choosing pairs of ontology concepts with maximal weighted similarity. The two concepts are considered matched when the *wsim* valued exceeds a given threshold λt (see section 4.1).

In the next sub-sections, we describe the linguistic and probabilistic matching phases in more detail.

3.1 Linguistic matching

The first phase of ontology matching is based primarily on concept names. In the absence of data instances, such names are probably the most useful source of information for matching. Linguistic matching proceeds in two steps: normalization and comparison.

Normalization considers names as words in some natural language. It is based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words. Normalization usually consists of tokenization, lemmatization and elimination.

• Tokenization. Names of entities are parsed into sequences of tokens by a tokenizer which recognizes punctuation, cases, blank characters, digits, etc. E.g. hasChildren→ {Has, Children}

• Lemmatization. The strings underlying tokens are morphologically analyzed in order to find all their possible basic forms, e.g. {Has, Children} → {Have, Child}

• Elimination. The tokens that are articles, prepositions, conjunctions, and so on, are marked to be discarded.

The similarity of two name tokens t1 and t2, *sim* (t1, t2), is computed by matching sub-string of the words t1 and t2.

$$sim(\mathrm{t1},\mathrm{t2}) = \frac{\mathrm{length(max\_substring(t1, t2))}}{\mathrm{max(length(t1), length(t2))}}$$

We use WordNet [10] to help match names by identifying acronyms and synonyms. For example, the concept Man in O1 and the concept Male in O2 do not match in names, but they are synonyms.

Given two concepts C∈O1 and D∈O2, let T(C) denote the set of name tokens for concept C, then linguistic similarity coefficients (*lsim*) are then computed between the concepts by comparing the tokens extracted from their names.

lsim (C, D) =

$$\frac{\sum_{t1\in T(C)} [\max_{t2\in T(D)} sim(t1,t2)] + \sum_{t2\in T(D)} [\max_{t1\in T(C)} sim(t1,t2)]}{|T(C)| + |T(D)|}$$

3.2 Probabilistic matching

Given two concepts C ∈ O_1 and D ∈ O_2, the probabilistic matching is based on a well-defined similarity measure:

Jaccard-sim(C, D) = $P(C \cap D) / P(C \cup D)$

$$= \frac{P(C \cap D)}{P(C) + P(D) - P(C \cap D)}$$

This similarity measure is known as the Jaccard coefficient. It takes the lowest value 0 when C and D are disjoint and the highest value 1 when C and D are the same concept. To compute the Jaccard coefficient of two concepts, we need to compute three probabilities: the probability P(C) of the concept description C, the probability P(D) of the concept description D, and the probability P(C∩D) of the concept description C∩D.

In summary we employ Probabilistic Description Logic to compute the Jaccard coefficient as follows:

• 1. Pick out all the primitive concepts and roles from O_1 and O_2 and form the set S of primitive concepts and roles. As for ontologies in Table 1, S = {Man (O_1), Rich, Old, Man (O_2), hasChildren, HavingMoney}.

• 2. With the result of linguistic matching, we have the *lsim* value between each pair of entities from the set S. A threshold lt is defined. If *lsim* exceeds *lt*, we consider the pair of primitive entities matches. The matched entities are treated as the same element in the set S and we remove the duplicate element from the set. As for ontologies in Table 1, S = {Man, Rich, Old, hasChildren}, and the canonical form of every concept can be expressed with the primitive concepts and roles in set S.

• 3. Construct the Bayesian networks over set S and every role-filler sets as required by Probabilistic Description Logic. We will discuss it in detail in the next sub-section.

• 4. Use the Bayesian network to compute the probabilities for the three concept descriptions and then compute the Jaccard coefficient *jsim*.

From above we see that in order to exploit the inference service provided by Probabilistic Description Logic, the key step is to build the Bayesian network over set S whose elements are primitive concepts and roles.

3.3 Construct BNs using a search engine

A Bayesian network is a Directed Acyclic Graph (DAG) in which the nodes represent random variables. Each variable takes on a value in some predefined range. Each node in the network is associated with a conditional probability table (CPT), which defines the probability of each possible value of the node, given each combination of values for the node's parents in the DAG. We propose to exploit a search engine to construct the Bayesian network over set S.

A search engine is a searchable online database of internet resources. Search engines attempt to index and locate desired information by searching for keywords in which a user specifies. Page counts and snippets are two useful information sources provided by most Web search engines. Especially, we will exploit the page counts information to construct the Bayesian network.

Page count of a query is the number of pages that contain the query words. For example, the page count of the query "Man" in Google is 1,130,000,000, whereas the same for "have Child" is only 605,000,000. The page count of the query is considered as a measure of individual occurrences of the primitive concept [2] . We define f(C) as the page count of the query whose search term is the normalized name of the concept C, so we have

f(Man) = 1,130,000,000.
f(⩾1hasChildrenren) = 605,000,000
f(Old) = 740,000,000

The page count for the query "man AND have Child" is 254,000,000, which is considered as a measure of individual occurrences of both concepts. Similarly we define f(C1, C2, …, Cn) as the page count of the query whose search term is the AND composition of the names of the concepts C1, C2, …, Cn. So we get

f(Man, ⩾1hasChildren) = 299,000,000
f(Man, Old) = 385,000,000
f(⩾1hasChildren, Old) = 413,000,000
f(⩾1hasChildren, Man, Old) = 206,000,000

In order to compute the probability that an individual in the domain of the ontology belongs to a concept, we need to know the number of all individual occurrences related to set S. For our example, we need the page count of the query "have Child OR Old OR Rich OR Man", which is 2,190,000,000. This page number is considered as the cardinality of the domain of the ontology and we denote it by M. Then the probability that an individual belongs to the concepts can be computed, for example:

P(Man) = f(Man) / M = 0.51
P(Man, Old) = f(Man, Old) / M = 0.18
P(Man, ¬Old)= (f(Man) - f(Man, Old)) / M= 0.34
P(¬Man, Old)=(f(Old) - f(Man, Old)) / M= 0.16
P(¬Man, ¬Old)=1 - 0.26 - 0.31 + 0.21= 0.32

This can be easily extended for more primitive concepts and roles. Given primitive concepts A, B and C, the following probabilities can be computed:

P(A, B, C) = f(A, B, C) / M
P(A, B, ¬C) = (f(A, B) - f(A, B, C))/ M
P(A, ¬B, C) = (f(A, C) - f(A, B, C))/ M
P(A, ¬B, ¬C)
= (f(A) - f(A, B) – f(A, C) + f(A, B, C))/ M

With the formulas above we have the following probabilities for our examples:

P(⩾1hasChildren, Man, Old)= 0.094
P(⩾1hasChildren, Man, ¬Old) = 0.042
P(⩾1hasChildren, ¬Man, Old)= 0.095
P(⩾1hasChildren, ¬Man, ¬Old)= 0.045

In order to construct the Bayesian network, we need first to determine the topology of it, which means to determine the edges of the graph whose nodes are primitive concepts and roles in set S. When there is a strong relativity holding between two primitive concepts, we add an edge between the nodes of the two concepts. We use the page count overlap coefficient to measure the relativity between two primitive concepts. Given two primitive concepts A and B, the page count overlap coefficient between A and B is defined as:

PC_Overlap (A, B) = f(A, B) / min(f(A), f(B))

With a predefined a threshold pt, for example pt = 0.25, so we get PC_Overlap (Man, Old) = 0.44 > pt, then we consider the relativity holds between concept Man and concept Rich, and we add an edge between the nodes of the two concepts. Since f(Man) > f(Old), we think concept Man is more common so the direction of the edge is from node Man to node Old. With the topology of the Bayesian network defined, we could further continue to construct the CPTs for the Bayesian network. In our example, for node ⩾1hasChildren, whose parent nodes are node Man and node Old, the CPT items for node ⩾1hasChildren can be computed as follows:

P(⩾1hasChildren | Man, Old)
= P(⩾1hasChildren, Man, Old) / P(Man, Old)
=0.094 / 0.18= 0.52

P(⩾1hasChildren | Man, ¬Old)
= P(⩾1hasChildren, Man, ¬Old) / P(Man, ¬Old)

= 0.042 / 0.34=0.12

P(⩾1hasChildren | ¬Man, Old)
= P(⩾1hasChildren, ¬Man, Old) / P(¬Man, Old)
= 0.95 / 0.16=0.59

P(⩾1hasChildren | ¬Man, ¬Old)
= P(⩾1hasChildren, ¬Man, ¬Old) / P(¬Man, ¬Old)
= 0.045 / 0.32= 0.14

After the CPTs are constructed, we can employ the inference algorithm developed for Bayesian networks to compute the probabilities of the concept descriptions whose canonical form refers to only the primitive concepts and roles, which are nodes in the Bayesian network. For example, with the Bayesian network in Figure 1, the probability of the concept description RichOldMan ∩ HappyFather can thus be computed:

P(RichOldMan ∩ HappyFather)
=P(Man ∩ Rich ∩ Old ∩ ⩾1hasChildren)
=P(Man)•P(Rich | Man)•P(Rich | Old)•
P(⩾1hasChildrenren | Man, Old)
=0.51•0.13•0.34•0.52=0.011145

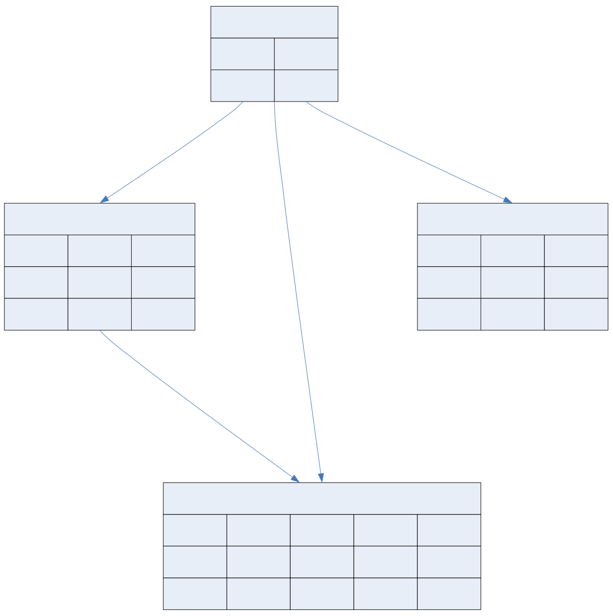
Figure 1: Bayesian network for our example

4 Experiments

We evaluated PDLOM experimentally on a collection of ontologies dealing with conference organization. They have been developed within OntoFarm [14] project. Four ontologies are included. Each of them is shortly described in Table 2. Two human reviewers were asked to manually match the 6 pairs of ontologies. The matching provided by the reviewers was used as ground truth when measuring precision and recall.

Table 2: Test Ontologies

Name	Number of Classes	Number of Properties	DL expressivity
SigKdd	49	28	$\mathcal{ALCI(D)}$
ConfTool	38	36	$\mathcal{SIF(D)}$
Cmt	36	59	$\mathcal{ALCIF(D)}$
Cocus	55	35	$\mathcal{ALCIF(D)}$

The PDLOM is implemented on top of the Probabilistic Description Logic reasoner [6]. We use Google as the search engine to construct our Bayesian networks. All experiments were performed on a Celeron D 3Ghz desktop machine with 1GB of RAM under Windows XP SP2.

The objectives of our experimental evaluation are two-fold. First, we investigate the optimal choice of the weight parameter and the threshold that produce the best F1 quality[1] w.r.t. the ground truth. Second, we compare the precision, recall and F1 quality of PDLOM with those of COMA++ [1].

4.1 Optimal PDLOM Parameters

The set of parameters for PDLOM is comprised of:

- The weight parameter for similarity computations.
- The similarity threshold *lt* is the minimum similarity value against which two primitive concepts are considered matched.
- The similarity threshold λt is the minimum similarity value against which two concepts are considered matched.
- The threshold *pt* is used to determine whether an edge exists between two nodes in the Bayesian network.

In our first set of experiments, we determine the parameter setting that maximizes the average F1 quality over the entire dataset. The optimal values of pt turned out to be independent of the specific pair of ontologies.

Since most of the running time is spent on inference and the inference complexity is highly connected with the number of edges in the Bayesian network, we expected a sharp increase with the decrease of *pt*. The experiments confirmed that the running time increases almost exponentially for *pt* < 0.25, while F1 quality stabilizes to a value of

[1] For a pair of ontologies, we denote by A the ground truth and by B the set of matched pairs determined by the algorithm. We compute precision as P = |A ∩ B| / |A| and recall as R = |A ∩ B| / |B|. The F1 quality measure is defined as usual F1 = 2•P•R / (P+R).

approximately 0.66 for the same values of *pt* (Figure 2). From these two observations, we determined that pt = 0.25 is a good compromise between quality and running time.

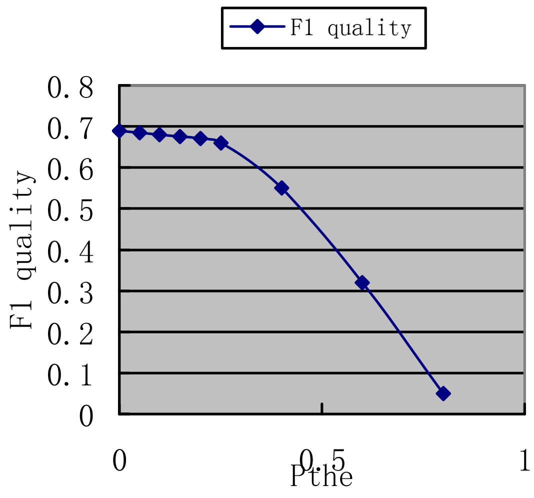

Figure 2: F1 quality decreases with the increase of the threshold

We varied the values of the weight parameter and the similarity threshold λt in increments of .05 in the range [0, 1]. We then identified the configuration of parameters that maximizes the average F1 value over the entire dataset. For each pair of ontologies, we also determined the configuration of the parameters that maximizes the F1 quality value for that pair. This resulted in a set of 6 values for each configuration of parameters. Line 7 of Table 3 contains the parameter configuration that maximizes the average F1 value for all the pairs.

Table 3: PDLOM optimal parameter values

	w	*lt*	λt
SigKdd-ConfTool	0.4	0.7	0.65
SigKdd-Cmt	0.5	0.75	0.7
SigKdd-Cocus	0.35	0.7	0.7
ConfTool-Cmt	0.55	0.7	0.6
ConfTool-Cocus	0.5	0.65	0.55
Cmt-Cocus	0.4	0.65	0.6
Average	0.47	0.69	0.63

4.2 Comparison with COMA++

We compared precision, recall and F1 quality against one of the leading systems COMA++. COMA++ is a tool that implements multiple match strategies to align relational schemas, XML and OWL. PDLOM is configured with the optimal parameter values for average.

Figure 3 shows the F1 quality for every pair in Table 3. We have found that PDLOM performs significantly better than average when the two ontologies share highly similar primitive concepts and relations such as ConfTool, Cmt and Cocus. Since under such conditions, canonical forms of concepts to be compared share same primitive concepts and this makes the inference service of Probabilistic Description Logic to play a very important part to capture the similarity between concepts.

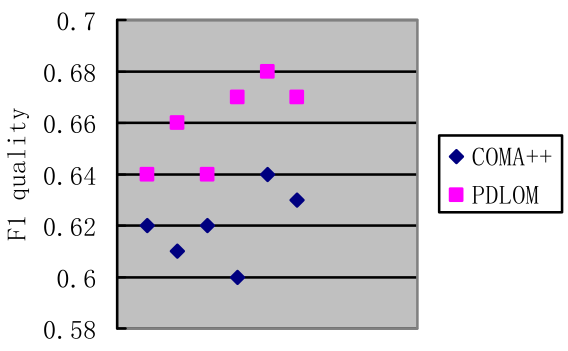

Figure 3: Comparison of F1 quality for the 6 selected pair of concepts

Figure 4 shows the average precision, recall and F1 quality for both methods. From the experimental data, we see that, on average, the improvement in F1 quality of PDLOM is significant, 6% over COMA++.

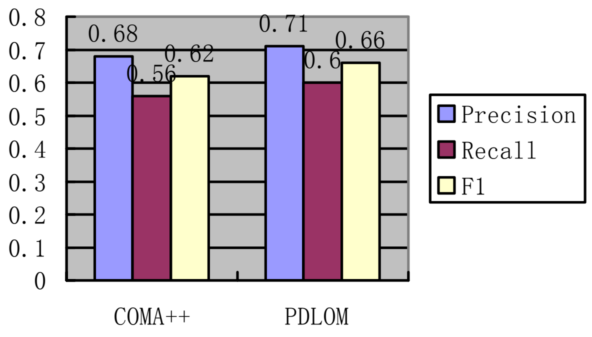

Figure 4: Comparison of average Results

5 Related work

Ontology matching methods can generally be classified as linguistic, structural, and instance-based. Many of the linguistic approaches use an external lexicon to match terms which are semantically related. PDLOM uses WordNet [10] as well, which is an electronic lexical database, where various senses of words are put together into sets of synonyms.

Structural approaches compute the correspondences by analyzing how entities appear together in a structure and define the similarity measure based on node neighborhoods. Similarity Flooding [9] presents schemas as directed labeled graphs and computes the correspondences of nodes in the graphs as the fix-point of an operator. Cupid [8] computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which elementary schema elements occur.

In the area of instance-based methods, HICAL [5] uses k-statistics from the data instances to infer rule mappings among concepts. Glue [3] learns classifiers for classes based on instance data, and finally computes the joint probability distribution of instances. The system uses a relaxation labeling process to propagate similarity. oPLMap [11] is a formal framework based on Horn predicate logics and probability theory, which allow for automatically learning mapping rules between ontologies.

The most successful ontology matching systems, like COMA++, do not fall neatly into any of these categories, but rather use a mixture of techniques. PDLOM seems to fall into the category of instance-based methods. However, Probabilistic Description Logic uses the canonical form of a concept description to compute the probability. The canonical form of a concept description is just the structure information of the concept. So PDLOM employs both structure information and instances to compute the probability for concepts.

6 Conclusion

We have presented PDLOM, an ontology matching approach, which employs the inference service provided by Probabilistic Description Logic to compute the probability of a concept description and further the Jaccard coefficient between concepts. We have described how to exploit a search engine to get the probability distribution over primitive concepts and roles required by Probabilistic Description Logic. Our approach has the advantage of a well defined semantic similarity, but avoids requiring massive data instances of all the concepts in the ontologies to learn the Jaccard coefficient as other probability based approach.

References:

[1] D. Aumueller, H. Do, S. Massmann, E. Rahm. Schema and ontology matching with COMA++. In Proceedings of SIGMOD (Demonstration), 2005.

[2] Cilibrasi, R., Vitanyi, P.: The google similarity distance. IEEE Transactions on knowledge and data engineering 19(3), pages 370–383, 2007.

[3] A. Doan, J. Madhavan, P. Domingos, A. Halevy. Learning to map ontologies on the semantic web. In Proceedings of WWW, 2002.

[4] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. J. of Web Semantics, 1(1):7–26, 2003.

[5] R. Ichise, H. Takeda and S. Honiden. Rule induction for concept hierarchy alignment. In Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), 2001.

[6] A. Kaplunova and R. Möller. Probabilistic LCS in a P-Classic Implementation. Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany, 2007.

[7] D. Koller, A. Levy and A. Pfeffer. P-Classic: a tractable probabilistic description logic. In AAAI, 1997.

[8] J. Madhavan, P. Bernstein, E. Rahm. Generic Schema Matching with Cupid. In Proceedings of VLDB, 2001.

[9] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In Proceedings of the International Conference on Data Engineering (ICDE), pages 117–128, 2002.

[10] A. G. Miller. WordNet: A lexical database for English. Communications of the ACM, (38(11)):39–41, 1995.

[11] H. Nottelmann, U. Straccia: A Probabilistic, Logic-based Framework for Automated Web Directory Alignment Soft Computing in Ontologies and the Semantic Web, 2006.

[12] P. Patel-Schneider, P. Hayes, and I. Horrocks. Web ontology language OWL Abstract Syntax and Semantics. W3C Recommendation, 2004.

[13] P. Shvaiko, J. Euzenat. A Survey of Schema-based Matching Approaches. In Journal on Data Semantics, 2005.

[14] O. Svab, V. Svatek, P. Berka, D. Rak, and P. Tomasek. Ontofarm: Towards an experimental collection of parallel ontologies. In Poster Proceedings of the International Semantic Web Conference 2005, 2005.