Ontology Matching with Knowledge Rules

Shangpu Jiang, Daniel Lowd, and Dejing Dou

Department of Computer and Information Science
University of Oregon, USA
{shangpu,lowd,dou}@cs.uoregon.edu

Abstract. Ontology matching is the process of automatically determining the semantic equivalences between the concepts of two ontologies. Most ontology matching algorithms are based on two types of strategies: terminology-based strategies, which align concepts based on their names or descriptions, and structure-based strategies, which exploit concept hierarchies to find the alignment. In many domains, there is additional information about the relationships of concepts represented in various ways, such as Bayesian networks, decision trees, and association rules. We propose to use the similarities between these relationships to find more accurate alignments. We accomplish this by defining soft constraints that prefer alignments where corresponding concepts have the same local relationships encoded as *knowledge rules*. We use a probabilistic framework to integrate this new *knowledge-based* strategy with standard terminology-based and structure-based strategies. Furthermore, our method is particularly effective in identifying correspondences between complex concepts. Our method achieves better F-score than the state-of-the-art on three ontology matching domains.

1 Introduction

Ontology matching is the process of aligning two semantically related ontologies. Traditionally, this task is performed by human experts from the domain of the ontologies. Since the task is tedious and error prone, especially in large ontologies, there has been substantial work on developing automated or semi-automated ontology matching systems [18]. While some automated matching systems make use of data instances, in this paper we focus on the *schema-level* ontology matching task, in which no data instance is used.

Previous automatic ontology matching systems mainly use two classes of strategies. *Terminology-based* strategies discover corresponding concepts with similar names or descriptions. *Structure-based* strategies discover corresponding groups of concepts with similar hierarchies. In many cases, additional information about the relationships among the concepts is available through domain models, such as Bayesian networks, decision trees, and association rules. A domain model can be represented as a collection of *knowledge rules*, each of which denotes a semantic relationship among several concepts. These relationships may be complex, uncertain, and rely on imprecise numeric values. In this paper, we introduce a new *knowledge-based strategy* which uses the structure of these knowledge rules as (soft) constraints on the alignment.

As a motivating example, consider two ontologies in the basketball game domain. One ontology has datatype properties `height`, `weight`, `center`, `forward` and `guard` for players, while the other ontology has the corresponding datatype properties `h`, `w`, and `position`. Terminology-based strategies may not identify these correspondences. However, if we know that a large value of `height` implies `center` is true in the first ontology, and the same relationship holds for `h` and `position` = `Center` in the second ontology, then we tend to believe that `height` maps to `h` and `center` maps to `position` = `Center`.

We use Markov logic networks (MLNs) [4] as a probabilistic language to combine the knowledge-based strategy with other strategies, in a formalism similar to that of [12]. In particular, we encode the knowledge-based strategy with weighted formulas that increase the probability of alignments where corresponding concepts have isomorphic relationships. We use an MLN inference engine to find the most likely alignment. We name our method Knowledge-Aware Ontology Matching (KAOM).

Our approach is also capable of identifying *complex correspondences*, an extremely difficult task in ontology matching. A complex correspondence is a correspondence between a simple concept and a complex concept (e.g., `grad_student` maps to the union of `PhD` and `Masters`). This can be achieved by constructing a set of *complex concepts* (e.g., unions of concepts) in each ontology, subsequantly generating candidate complex correspondences, and using multiple strategies – including the knowledge-based strategy – to find the correct ones.

The contributions of this work are as follows:

- We show how to represent common types of domain models as knowledge rules, and how to use these knowledge rules to obtain more accurate alignments. Our approach is especially effective in identifying the correspondences of numerical or nominal datatype properties. By incorporating complex concepts, our approach is also capable of discovering complex correspondences, which is a very difficult scenario in the ontology matching task.
- We evaluate the effectiveness of KAOM in three domains with different types of knowledge rules, and show that our approach not only outperforms the state-of-the-art approaches for ontology matching in one-to-one matching, but also discovers complex correspondences successfully.

The paper is organized as follows. In Section 2, we define ontology matching and review previous work. In Section 3, we introduce the concept of “knowledge rules” with a definition and examples. In Section 4, we present the knowledge-based strategy. In Section 5, we show how to incorporate complex concepts in our method. In Section 6, we formalize our method with Markov logic networks. We present experimental results in Section 7 and conclude in Section 8.

2 Ontology Matching

We begin by formally defining ontology matching.

Definition 1 (Ontology Matching [5]).
Given two ontologies O_1 and O_2, a correspondence *is a 3-tuple $\langle e_1, e_2, r\rangle$ where e_1 and e_2 are entities of the first and second ontologies respectively, and r is a semantic relation such as equivalence ($\equiv$) and subsumptions ($\sqsubseteq$ or $\sqsupseteq$). An* alignment *is a set of correspondences.* Ontology matching *is the task or process of identifying the correct semantic alignment between the two ontologies. In most cases, ontology matching focuses on equivalence relationships only.*

Most existing schema-level ontology matching systems use two types of strategies: terminology-based and structure-based. Terminology-based strategies are based on terminological similarity, such as string-based or linguistic similarity measures. Structure-based strategies are based on the assumption that two matching ontologies should have similar local or global structures, where the structure is represented by subsumption relationships of classes and properties, and domains and ranges of properties. Advanced ontology matching systems often combine the two types of strategies [1, 10, 11, 14]. See [18] for a survey of ontology matching systems and algorithms.

Recently, a probabilistic framework based on Markov logic was proposed to combine multiple strategies [12]. In particular, it encodes multiple strategies and heuristics into hard and soft constraints, and finds the best matching by minimizing the weighted number of violated constraints. The constraints include string similarity, the cardinality constraints which enforce that each concept matches at most one concept, the coherence constraints which prevent inconsistency induced by the matching, and the stability constraints which penalize dissimilar local subsumption relationships.

Definition 2 (Complex Correspondences).
A complex concept *is a composition (e.g., unions, complements) of one or more simple concepts. In OWL*[1]*, there are several constructors for creating complex classes and properties (see the top part of Table 1 for an incomplete list of constructors). A* complex correspondence *is an equivalence relation between a simple class or property and a complex class or property in two ontologies [17].*

Previous work has taken several different approaches to find complex correspondences (i.e., complex matching). [2] constructs candidates for complex correspondences using operators for primitive classes, such as string concatenation or arithmetic operations on numbers. [17] summarizes four patterns for building up complex correspondences based on linguistic and structural features given a candidate one-to-one alignment: Class by Attribute Type (CAT), Class by Inverse Attribute Type (CIAT), Class by Attribute Value (CAV), and Property Chain pattern (PC). Finally, when aligned or overlapping data is available, inductive logic programming (ILP) techniques can be used as well [6, 15].

Many ontology matching systems make use of data instances to some extent (e.g., [2, 3, 6, 15]). However, in this paper, we focus on the case where data are not available or data sharing is not preferred because of communication cost or privacy concerns.

[1] `http://www.w3.org/TR/owl2-primer/`

3 Representation of Domain Knowledge

In the AI community, knowledge is typically represented in *formal languages*, among which ontology-based languages are the most widely used forms. The Web Ontology Language (OWL) is the W3C standard ontology language that describes the classes and properties of objects in a specific domain. OWL and many other ontology languages are based on variations of description logics.

In ontology languages such as OWL, knowledge is represented as logic *axioms*. These axioms describe properties of classes or relations (e.g., a relation is functional, symmetric, or antisymmetric, etc.), or a relationship of several entities (e.g., the relation 'grandfather' is the composition of the two relations 'father' and 'parent').

The choice of using description logic as the foundation of the Semantic Web ontology languages is largely due to the trade-off between expressivity and reasoning efficiency. In tasks such as ontology matching, reasoning does not need to be instant, so we can afford to consider other forms of knowledge outside of a specific ontology language or description logic.

Definition 3 (Knowledge Rule).
A knowledge rule is a sentence $R(a, b, \ldots; \theta)$ in a formal language which consists of a relation R, a set of entities (i.e., classes, attributes or relations) $\{a, b, \ldots\}$, and (optionally) a set of parameters θ. A knowledge rule carries logical or probabilistic semantics representing the relationship among these entities. The specific semantics depend on R.

Many domain models and other types of knowledge can be represented as sets of knowledge rules, each rule describing the relationship of a small number of entities. The semantics of each relationship R can typically be expressed with a formal language. Table 1 shows some examples of the symbols used in formal languages such as description logic, along with their associated semantics.

We illustrate a few forms of knowledge rules with the following examples. For each rule, we provide a description in English, a logical representation, and an encoding as a knowledge rule with a particular semantic relationship, R_i. We define a new relationship in each example, but, in a large domain model, most relationships would be appear many times in different rules.

Example 1. The submission deadline precedes the camera ready deadline:

$$\texttt{paperDueOn} \prec \texttt{manuscriptDueOn}$$

This is represented as $R_1(\texttt{paperDueOn}, \texttt{manuscriptDueOn})$ with $R_1(a, b) : a \prec b$.

Example 2. A basketball player taller than 81 inches and heavier than 245 pounds is likely to be a center:

$$\texttt{h} > 81 \wedge \texttt{w} > 245 \Rightarrow \texttt{pos} = \texttt{Center}$$

This rule can be viewed as a branch of a *decision tree* or an *association rule.* It can be represented as $R_2(\texttt{h}, \texttt{w}, \texttt{pos=Center}, [81, 245])$, with $R_2(a, b, c, \theta) : a > \theta_1 \wedge b > \theta_2 \Rightarrow c$.

Table 1. Syntax and semantics of DL symbols (top), DL axioms (middle), and other knowledge rules used in the examples of the paper (bottom)

Syntax	Semantics
$\top$	$\mathcal{D}$
$\bot$	$\emptyset$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\mathcal{D} \backslash C^{\mathcal{I}}$
$\forall R.C$	$\{x \in \mathcal{D} \mid \forall y((x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
$\exists R.C$	$\{x \in \mathcal{D} \mid \exists y((x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
$R \circ S$	$\{(x,y) \mid \exists z((x,z) \in R^{\mathcal{I}} \wedge (z,y) \in S^{\mathcal{I}})\}$
R^{-}	$\{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$
$R \upharpoonright C$	$\{(x,y) \in R^{\mathcal{I}} \mid x \in C^{\mathcal{I}}\}$
$R \downharpoonleft C$	$\{(x,y) \in R^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\}$
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$C \sqsubseteq \neg D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
$R \prec S$	$y < y'$ for $\forall (x,y) \in R^{\mathcal{I}} \wedge (x,y') \in S^{\mathcal{I}}$
$C \Rightarrow D$	$\Pr(D^{\mathcal{I}} \mid C^{\mathcal{I}})$ is close to 1

Example 3. A smoker's friend is likely to be a smoker as well:

$$\texttt{Smokes}(x) \wedge \texttt{Friend}(x,y) \Rightarrow \texttt{Smokes}(y)$$

Relational rules such as this one describe relationships of attributes across multiple tables, as opposed to propositional data mining rules that are restricted to a single table. This rule can be represented as $R_3(\texttt{Smoke}, \texttt{Friend})$ with $R_3(a,b) : a(x) \wedge b(x,y) \Rightarrow a(y)$.

For the remainder of this paper, we will assume that the knowledge in both domains is represented as knowledge rules, as described in this section.

4 Our New Knowledge-Based Strategy

We propose a new strategy for ontology matching that uses the similarity of knowledge rules in the two ontologies. It is inspired by the structure-based strategy in many ontology matching algorithms (e.g., [11] and [12]). It naturally extends the subsumption relationship of entities in structure-based strategies to other types of relationships.

We use Markov logic to combine the knowledge-based strategy with other strategies. In particular, each strategy is represented as a set of *soft constraints*, each of which assigns a score to the alignments satisfying it, and the alignment with the highest total score is chosen as the best alignment. We now describe the soft constraints encoding the knowledge-based strategy. Our complete Markov logic-based approach, including the soft constraints required for the other strategies, will be described in Section 6.

For each relation R_k that appears in both domains, we introduce a set of soft constraints so that the alignments that preserve these relationships are preferred to those that do not:

$$\begin{aligned} +w_k \qquad & R_k(a,b) \wedge \neg R_k(a',b') \Rightarrow a \not\equiv a' \vee b \not\equiv b' \\ +w'_k \qquad & R_k(a,b) \wedge R_k(a',b') \Rightarrow a \equiv a' \wedge b \equiv b' \\ & \forall a, b \in O_1, a', b' \in O_2 \end{aligned}$$

These formulas assume R_k is a binary relation, but they trivially generalize to any arity, e.g., $R_k(a,b,c,d,e,\ldots)$. Note that separate constraints are created for each possible tuple of constants from the respective domains. The numbers preceding the constraints (w_k and w'_k) are the *weights*. A larger weight represents a stronger constraint, since alignments are ranked based on the total weights of the constraints they satisfy. A missing weight means the constraint is a hard constraint which must be satisfied.

Example 4. A reviewer of a paper cannot be the paper's author. In the `cmt` [2] ontology we have $R_4(\texttt{writePaper}, \texttt{readPaper})$ and in the `confOf` ontology we have $R_4(\texttt{write}, \texttt{reviews})$ where $R_4(a,b) : a \sqsubseteq \neg b$ is the disjoint relationship of properties. Applying the constraint formulas defined above, we increase the score of all alignments containing the two correct correspondences: `writePaper` $\equiv$ `writes` and `readPaper` $\equiv$ `reviews`.

Rules involving continuous numerical attributes often include parameters (e.g., thresholds in Example 2) that do not match between different ontologies. In order to apply the knowledge-based strategy to numerical attributes, we make the assumption that corresponding numerical attributes roughly have a *positive linear* transformation. This assumption is often true in real applications, for instance, when an imperial measure of height matches to a metric measure of height. We propose two methods to handle numerical attributes.

The first method is to compute a *distance measure* (e.g., Kullback-Leibler divergence) between the distributions of the corresponding attributes in a candidate alignment. Although the two distributions describe different attributes, the distance can be computed by assuming a linear transformation between the two attributes. The coefficients of the mapping relation can be roughly estimated using the ranges of attribute values appearing in the knowledge rules (see Example 5 below).

Specifically, if the distance between rules $R(\texttt{a}, \texttt{b}, \ldots, \theta)$ and $R(\texttt{a'}, \texttt{b'}, \ldots, \theta')$ is d, then we add the constraint:

$$a \equiv a' \wedge b \equiv b' \wedge c \equiv c'$$

with a weight of $\max(d_0 - d, 0)$ for a given threshold d_0.

[2] Throughout the paper, we will use ontologies in the conference domain (`cmt`, `confOf`, `conference`, `edas`, `ekaw`) and the NBA domain (`nba-os`, `yahoo`) in our examples. The characteristics of these ontologies will be further described in Section 7.

Example 5. In the `nba-os` ontology, we have conditional rules converted from a decision tree, such as

$$\texttt{h} > 81 \wedge \texttt{w} > 245 \Rightarrow \texttt{Center}$$

Similarly, in the `nbayahoo` ontology, we have

$$\texttt{h'} > 2.06 \wedge \texttt{w'} > 112.5 \Rightarrow \texttt{Center'}$$

Here the knowledge rules represent the conditional distributions of multiple entities. We define the distance between the two conditional distributions as

$$d(\texttt{h}, \texttt{w}, \texttt{Center}; \texttt{h'}, \texttt{w'}, \texttt{Center'}) = \mathbb{E}_{p(\texttt{h},\texttt{w})} d(p(\texttt{Center}|\texttt{h}, \texttt{w})||p(\texttt{Center'}|\texttt{h'}, \texttt{w'}))$$

where $\mathbb{E}(\cdot)$ is expectation and $d(p||p')$ is a distance measure. Because `Center` and $\texttt{Center}'$ are binary attributes, we simply use $|p-p'|$ as the distance measure. For numerical attributes, we can use the difference of two distribution histograms as the distance measure. We assume the attribute correspondences (`h` and `h'`, `w` and `w'`) are linear mappings, and the linear relation can be roughly estimated (e.g., by simply matching the minimum and maximum numbers in these rules). When computing the expectation over `h` and `w`, we apply the linear mapping to generate corresponding values of `h'` and `w'`, e.g., $\texttt{h'} = 0.025\ \texttt{h}$, $\texttt{w'} = 0.45\ \texttt{w}$. The distribution of the conditional attributes $p(\texttt{h}, \texttt{w})$ can be roughly estimated as independent and uniform over the ranges of the attributes.

The second method for handling continuous attributes is to discretize them, reducing the continuous attribute problem to the discrete problem described earlier. For example, suppose each continuous attribute x is replaced with a discrete attribute x^d, indicating the quartile of x rather than its original value. Then we have $R_5(\texttt{h}^d, \texttt{w}^d, \texttt{Center})$ and $R_5(\texttt{h'}^d, \texttt{w'}^d, \texttt{Center'})$ with relation $R_5(a, b, c) : a = 4 \wedge b = 4 \Rightarrow c$, and the discrete value of 4 indicates that both a and b are in the top quartile. Other discretization methods are also possible, as long as the discretization is done the same way in both domains.

Our method does not rely on the forms of knowledge rules, nor does it rely on the algorithms used to learn these rules. As long as similar techniques or tools are used on both sides of ontologies, we would always be able to find interesting knowledge-based similarities between the two ontologies.

5 Finding Complex Correspondences

Our approach can also find complex correspondences, which contain complex concepts in either or both of the ontologies. We add the complex concepts into consideration and treat them the same way as simple concepts, and then we jointly solve all the simple and complex correspondences by considering terminology, structure, and knowledge-based strategies in a single probabilistic formulation.

First, because complex concepts are recursively defined and potentially infinite, we need to select a finite subset of complex concepts and use them to generate the candidate correspondences. We will only include the complex concepts occurring in the ontology axioms or in the knowledge rules.

Second, we need to define a string similarity measure for each type of complex correspondence. For example, [17] requires two conditions for a Class by Attribute Type (CAT) matching pattern $O_1 : a \equiv O_2 : \exists p.b$ (e.g., $a =$ `Accepted_Paper`, $p =$ `hasDecision`, $b =$`Acceptance`): a and b are terminologically similar, and the domain of p (`Paper` in the example) is a superclass of a. We can therefore define the string similarity of a and $\exists p.b$ to be the string similarity of a and b which coincides with the first condition, and the second condition is encoded in the structure stability constraints. The string similarity measure of many other types of correspondences can be defined similarly based on the heuristic method in [17]. If there does not exist a straight-forward way to define the string similarity for a certain type of complex correspondences, we can simply set it to 0 and rely on other strategies to identify such correspondences.

Lastly, we need constraints for the correspondence of two complex concepts. The corresponding component concepts and same constructor always implies the corresponding complex concepts, while in the other direction, it is a soft constraint.

$$\begin{array}{ll} & \mathrm{cons}_k(a, b) \equiv \mathrm{cons}_k(a', b') \Leftarrow a \equiv a' \wedge b \equiv b' \\ +w^c_k & \mathrm{cons}_k(a, b) \equiv \mathrm{cons}_k(a', b') \Rightarrow a \equiv a' \wedge b \equiv b' \end{array}$$

where cons_k are different constructors for complex concepts, e.g., union, $\exists p.b$.

Some complex correspondences are almost impossible to be identified with traditional strategies. With the knowledge-based strategy, it becomes possible.

Example 6. A reviewer of a paper cannot be the paper's author. In the `cmt` ontology we have

$$\texttt{writePaper} \sqsubseteq \neg\texttt{readPaper}$$

and in the `conference` ontology we have

$$\texttt{contributes} \downharpoonleft \texttt{Reviewed\_contribution} \sqsubseteq \neg(\texttt{contributes} \circ \texttt{reviews})$$

We first build two complex concepts $\texttt{contributes} \downharpoonleft \texttt{Reviewed\_contribution}$ and $\texttt{contributes} \circ \texttt{reviews}$. With $R_4(a, b) = a \sqsubseteq \neg b$ (disjoint properties), the score function would favor the correspondences

$$\begin{aligned} \texttt{writePaper} &\equiv \texttt{contributes} \downharpoonleft \texttt{Reviewed\_contribution} \\ \texttt{readPaper} &\equiv \texttt{contributes} \circ \texttt{reviews} \end{aligned}$$

6 Knowledge Aware Ontology Matching

In this section, we present our approach, Knowledge Aware Ontology Matching (KAOM). KAOM uses Markov logic networks (MLNs) to solve the ontology matching task. The MLN formulation is similar to [12] but incorporates the knowledge-based matching strategy and treatment of complex correspondences.

An MLN [4] is a set of weighted formulas in first-order logic. Given a set of constants for individuals in a domain, an MLN induces a probability distribution over Herbrand interpretations or "possible worlds". In the ontology matching problem, we represent a correspondence in first-order logic using a binary relation, `match(a1,a2)`, which is true if concept `a1` from the first ontology is semantically equivalent to concept `a2` from the second ontology (e.g., `match(writePaper, writes)` means `writePaper` $\equiv$ `writes`). Each possible world therefore corresponds to an alignment of the two ontologies. We want to find the most probable possible world, which is the configuration that maximizes the sum of weights of satisfied formulas.

We define three components of the MLN of the ontology matching problem: *constants*, *evidence* and *formulas.* The logical constants are the entities in both ontologies, including the simple named ones and the complex ones. The evidence includes the complete set of OWL-supported relationships (e.g., subsumptions and disjointness) among all concepts in each ontology, and rules represented as first-order atomic predicates as described in the Section 3. We use an OWL reasoner to create the complete set of OWL axioms.

For the formulas, we begin with a set of formulas adapted from [12]:

1. *A-priori similarity* is the string similarity between all pairs of concepts:

$$s_{a,a'} \quad \texttt{match}(a, a')$$

where $s_{a,a'}$ is the string similarity between a and a', which also serves as the weight of the formula. We use the Levenshtein measure [9] for simple correspondences. This atomic formula increases the probability of matching pairs of concepts with similar strings, all other things being equal.

2. *Cardinality constraints* enforce one-to-one simple (or complex) correspondences:

$$\texttt{match}(a, a') \wedge \texttt{match}(a, a'') \Rightarrow a' = a''$$

3. *Coherence constraints* enforce consistency of subclass relationships:

$$\texttt{match}(a, a') \wedge \texttt{match}(b, b') \wedge a \sqsubseteq b \Rightarrow a' \sqsubseteq \neg b'$$

4. *Stability constraints* enforce consistency of the subclass relationships between the two ontologies. They can be viewed as a special case of the knowledge-based constraints we introduce below.

Knowledge-based Constraints We now describe how we incorporate knowledge-based constraints into the MLN formulation through new formulas relating knowledge rules to matchings. The *stability* constraints in [12] consider three subclass relationships, including a is a subclass of b (`subclass`), and a is a subclass or superclass of the domain or range of a property b (`domainsub`, `rangesub`). We extend the relationships (knowledge rule patterns) to sub-property, disjoint properties, and user-defined relations such as ordering of dates, and non-deterministic relationships such as correlation and anti-correlation:

$$-w_k \qquad R_k(a,b,...) \wedge \neg R_k(a',b',...) \Rightarrow \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k=1,...,m \tag{1}$$

where m is the number of knowledge rule patterns. User-defined relations include those derived from decision trees, association rules, expert systems, and other knowledge sources outside the ontology.

Besides the stability constraints, we introduce a new group of *similarity* constraints that encourage knowledge rules with the same pattern to have corresponding concepts.

$$+w'_k \qquad R_k(a,b,...) \wedge R_k(a',b',...) \Rightarrow \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k=1,...,m \tag{2}$$

For numerical rules, we instead use MLN formulas:

$$d_0 - d \quad \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k=1,...,m \tag{3}$$

where d is a distance measure of the two rules $R_k(a,b,...)$ and $R'_k(a',b',...)$ and d_0 is a threshold determining whether the rules are similar or not.

To handle complex correspondences, we add complex concepts that occur in knowledge rules as constants of the MLN, and add knowledge rules that contain these new complex concepts. We define the string similarity and enforce type constraints between simple and complex concepts, as described in Section 5. For complex to complex correspondences, the string similarity measure is zero, but we have constraints

$$\begin{array}{ll} & \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ... \Rightarrow \texttt{match}(c,c') \\ w_k^c & \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ... \Leftarrow \texttt{match}(c,c') \end{array}$$

where $c = \text{cons}_k(a,b,...), c' = \text{cons}_k(a',b',...)$ for each constructor cons_k.

7 Experiments

We test our KAOM approach on three domains: NBA, census, and conference. The sizes of the ontologies of these domains are listed in Table 2. These domains contain very different forms of ontologies and knowledge rules, so we can examine the generality and robustness of our approach.

We use Pellet [19] for logical inference of the ontological axioms and TheBeast[3] [16] and Rockit[4] [13] for Markov logic inference. We ran all experiments on a machine with 24 Intel Xeon E5-2640 cores @2500 MHz and 8GB memory. We compare our system (KAOM) with three others: KAOM without the

[3] `http://code.google.com/p/thebeast/`

[4] `https://code.google.com/p/rockit/`. We use RockIt for the census domain because TheBeast is not able to handle the large number of rules in that domain.

Table 2. Number of classes, object properties, data properties and nominal values of each ontology used in the experiments.

domain	ontology	# classes	# object props	# data props	# values
NBA	nba-os	3	3	20	3
	yahoo	4	4	21	7
census	adult	1	0	15	101
	income	1	0	12	97
OntoFarm	cmt	36	50	10	0
	confOf	38	13	25	0
	conference	60	46	18	6
	edas	103	30	20	0
	ekaw	78	33	0	0

knowledge-based strategy (MLOM), CODI [7] (a new version of [12], which is essentially a different implementation of MLOM), and logmap2 [8], a top performing system in OAEI 2014 [5].

We manually specify the weights of the Markov logic formlas in KAOM and MLOM. The weights of stability constraints for subclass relationships are set with values same as the ones used in [12], i.e., the weight for subclass is -0.5, and those for sub-domain and range are -0.25. In KAOM, we also set the weights for different types of similarity rules based on our assessment of their relative importance and kept these weights fixed during the experiments.

7.1 NBA

The NBA domain is a simple setting that we use to demonstrate the effectiveness of our approach. We collected data from the NBA official website and the Yahoo NBA website. For each ontology, we used the WinMine toolkit [6] to learn a decision tree for each attribute using the other attributes as inputs.

For each pair of conditional distributions based on decision tree with up to three attributes, we calculate their similarity based on the distance measure described in Example 5. We use the Markov logic formula (3) with the threshold $d_0 = 0.2$. To make the task more challenging, we did not use any name similarity measures. Our method successfully identified the correspondence of all the numerical and nominal attributes, including height, weight and positions (center, forward and guard) of players. In contrast, without a name similarity measure, no other method can solve the matching problem at all.

7.2 Census

We consider two census datasets and their ontologies from UC Irvine data repository[7]. Both datasets represent census data but are sampled and post-processed

[5] `http://oaei.ontologymatching.org/2014/`

[6] `http://research.microsoft.com/en-us/um/people/dmax/WinMine/Tooldoc.htm`

[7] `https://archive.ics.uci.edu/ml/datasets.html`

differently. These two census ontologies are flat with a single concept but many datatype properties and nominal values. For this domain, we use association rules as the knowledge. We first discretize each numerical attribute into five intervals, and then generate association rules for each ontology using the Apriori algorithm with a minimum confidence of 0.9 and minimum support of 0.001. For example, one generated rule is:

```
age='(-inf-25.5]' education='11th' hours-per-week='(-inf-35.5]'
  ==> adjusted-gross-income='<=50K' conf:(1)
```

This is represented as

$$R_6(\texttt{age}^d, \texttt{11th}, \texttt{hours-per-week}^d, \texttt{adjusted-gross-income}^d)$$

where x^d refers to the discretized value of x, split into one fifth percentile intervals, and $R_6(a, b, c, d) : a = 1 \land b \land c = 1 \Rightarrow d = 1$. For scalability reasons, we consider up to three concepts in a knowledge rule, i.e., association rules with up to three attributes. We set the weight of knowledge similarity constraints for the association rules to 0.25.

In the Markov logic formulation in [12], only the correspondences with apriori similarity measure larger than a threshold τ are added as evidence. In the experiments, we set τ with different values from 0.50 to 0.90. When τ is large, we deliberately discard the string similarity information for some correspondences. MLOM for this task is an extension of [12] by adding correspondences of *nominal values* and their dependencies with the related attributes. The results are shown in Figure 1. We can see that KAOM always gets better recall and F1, with only a slight degradation in precision. This means our approach fully leverages the knowledge rule information and thus does not rely too much on the names of the concepts to determine the matching. For example, when τ is 0.70, KAOM finds 6 out of 8 correspondences of values of `adult:workclass` and `income:class_of_worker`, while MLOM finds none. The other two systems were not designed for nominal value correspondences. CODI only finds 7 and logmap2 only finds 3 attribute correspondences, while KAOM and MLOM find all the 12 attribute correspondences.

7.3 OntoFarm

In order to show how our system can use manually created expert knowledge bases, we use OntoFarm, a standard ontology matching benchmark for an academic conference domain as the third domain in our experiments. As part of OAEI, it has been widely used in the evaluation of ontology matching systems. The process of manually knowledge rule creation is time consuming, so we only used 5 of the OntoFarm ontologies (`cmt`, `conference`, `confOf`, `edas`, `ekaw`). Using their knowledge of computer science conferences and the structure of just one ontology, two individuals listed a number of rules (e.g., Example 1). We then translated these rules into each of the five ontologies. Thus, the same knowledge was added to each of the ontologies, but its representation depended on the specific ontology. For some ontologies, some of the rules were not representable with

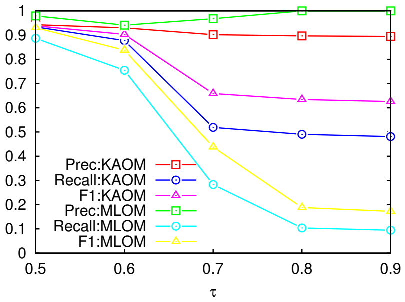

Fig. 1. Precision, recall and F1 on the census domain as a function of the string similarity threshold τ.

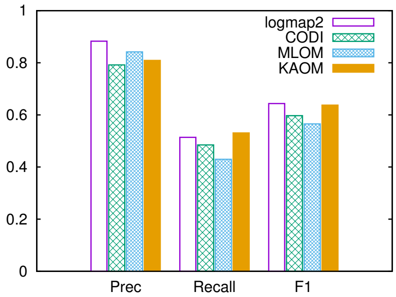

Fig. 2. Precision, recall and F1 on the OntoFarm domain with only the one-to-one correspondences.

the concepts in them and thus had to be omitted. This manually constructed knowledge base was developed before running any experiments and kept fixed throughout our experiments. Among the 5 ontologies, we have 10 pairs of matching tasks in total. We set τ to 0.70, and the weight for the knowledge similarity constraints to 1.0.

We first compare the four methods to the reference one-to-one alignment from the benchmark (Figure 2). KAOM achieves similar precision and F1, and better recall than other systems. It was able to identify correspondences in which the concept names are very different, for instance, `cmt:readPaper` $\equiv$ `confOf:reviews`. Note that the similarity constraints work in concert with other constraints. For instance, in Example 4, since disjointness is a symmetric knowledge rule, domain and range constraints could be helpful to identify whether `cmt:writePaper` should match to `confOf:writes` or `confOf:reviews`.

To evaluate our approach on complex correspondences, we extended the reference alignment with hand-labeled complex correspondences (Figure 3). MLOM does not perform well in this task because the complex correspondences require a good similarity measure to become candidates (such as the linguistic features in [17]). KAOM, however, uses the structure of the rules to find many complex correspondences without relying on complex similarity measures. For this task we also tried learning the weights of the formulas [8] (KAOM-learn). For each of the 10 pairs of ontologies, we used the rest 9 pairs as training data. KAOM-learn performs slightly better than KAOM.

With the hand-picked or automatically learned weights, KAOM produces a single most-likely alignment. However, we can further tune KAOM to produce alignments with higher recall or higher precision. We accomplish this by adding the MLN formula $\texttt{match}(a, a')$ with weight w. When w is positive, alignments with more matches are more likely, and when w is negative, alignments with fewer matches are more likely (all other things being equal). We adjusted this weight to produce the precision-recall curve shown in Figure 4. KAOM dominates

[8] We used MIRA implemented in TheBeast for weight learning.

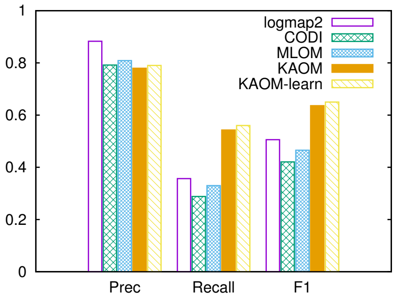

Fig. 3. Precision, recall and F1 on the OntoFarm domain with the complex correspondences.

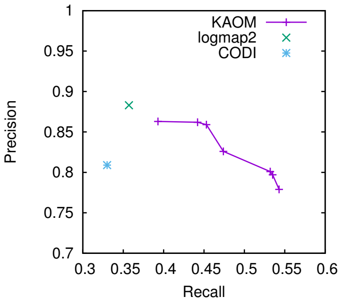

Fig. 4. Precision-recall curve on the OntoFarm domain with the complex correspondences.

CODI and provides much higher recall values than logmap2, although logmap2's best precision remains slightly above KAOM's.

8 Conclusion

We proposed a new ontology matching algorithm KAOM. The key component of KAOM is the knowledge-based strategy, which is based on the intuition that ontologies about the same domain should contain similar knowledge rules, in spite of the different terminologies they use. KAOM is also capable of discovering complex correspondences, by treating complex concepts the same way as simple ones. We encode the knowledge-based strategy and other strategies in Markov logic and find the best alignment with its inference tools. Experiments on the datasets and ontologies from three different domains show that our method effectively uses knowledge rules of different forms to outperform several state-of-the-art ontology matching methods.

Acknowledgement This research is funded by NSF grant IIS-1118050.

References

1. M. E. Cotterell and T. Medina. A Markov model for ontology alignment. *CoRR*, 2013.
2. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 383–394, 2004.
3. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662–673, 2002.

4. P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2009.
5. J. Euzenat and P. Shvaiko. *Ontology Matching.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
6. W. Hu, J. Chen, H. Zhang, and Y. Qu. Learning complex mappings between ontologies. In *Proceedings of Joint International Semantic Technology Conference*, pages 350–357, 2011.
7. J. Huber, T. Sztyler, J. Noessner, and C. Meilicke. CODI: Combinatorial optimization for data integration–results for OAEI 2011. *Ontology Matching*, page 134, 2011.
8. E. Jiménez-Ruiz, B. C. Grau, and Y. Zhou. LogMap 2.0: Towards logic-based, scalable and interactive ontology matching. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, pages 45–46, 2012.
9. V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.
10. M. Mao, Y. Peng, and M. Spring. An adaptive ontology mapping approach with neural network based constraint satisfaction. *Web Semantics*, 8(1):14–25, 2010.
11. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *Proceedings of Eighteenth International Conference on Data Engineering*, 2002.
12. M. Niepert, C. Meilicke, and H. Stuckenschmidt. A probabilistic-logical framework for ontology matching. In M. Fox and D. Poole, editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1413–1418, July 2010.
13. J. Noessner, M. Niepert, and H. Stuckenschmidt. RockIt: Exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
14. N. F. Noy and M. A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455, 2000.
15. H. Qin, D. Dou, and P. LePendu. Discovering executable semantic mappings between ontologies. In *Proceedings of International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2007)*, pages 832–849, 2007.
16. S. Riedel. Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceedings of the Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 468–475, 2008.
17. D. Ritze, C. Meilicke, O. Svb-Zamazal, and H. Stuckenschmidt. A pattern-based ontology matching approach for detecting complex correspondences. In *Ontology Matching (OM-2009)*, volume 551, 2008.
18. P. Shvaiko and J. Euzenat. Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, PP(99), 2011.
19. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53, 2007.