Learning Complex Mappings between Ontologies

Wei Hu, Jianfeng Chen, Hang Zhang, and Yuzhong Qu

State Key Laboratory for Novel Software Technology,
Nanjing University, China
{whu,yzqu}@nju.edu.cn, jf_chen@ymail.com,
hangzhang@smail.nju.edu.cn

Abstract. In this paper, we introduce a new approach for constructing complex mappings between ontologies by transforming it to a rule learning process. Derived from the classical Inductive Logic Programming, our approach uses instance mappings as training data and employs tailoring heuristics to improve the learning efficiency. Empirical evaluation shows that our generated Horn-rule mappings are meaningful.

1 Introduction

As the amount of ontologies grows with the development of the Semantic Web, it is now common to have different ontologies in a single domain. Ontologies can be *heterogeneous* in various forms including terminological and conceptual heterogeneities. Such heterogeneities are dealt with the *ontology matching* process, which plays a vital role in semantic interoperability between applications.

There exist plenty of works that tackle the ontology matching problem [2], however they mainly focus on finding simple $1:1$ equivalence mappings between named classes and/or named properties in ontologies. In the real world, *complex* $m:n$ *mappings* are also pervasive and needed by many applications for query rewriting, distributed reasoning, instance migration, etc. Neither the discovery methods to complex mappings nor their formal semantics has been well studied in literature yet.

In this paper, we propose a new approach which transforms the problem of constructing complex mappings to rule learning across ontologies. Our approach extends a pioneering *Inductive Logic Programming* (ILP) method named FOIL [8], and generates complex mappings in the form of first-order *Horn-rules*. Furthermore, our approach exploits instance mappings as the training data for variable bindings, and heuristically reduces the search and storage space by tailoring irrelevant data. Empirical evaluation shows the feasibility of our approach.

The rest of this paper is organized as follows. Preliminaries are introduced in Section 2. Section 3 presents our learning approach while Section 4 reports experimental results. Finally, we conclude this paper in Section 5.

2 Problem Statement

The Web Ontology Language (OWL), as a W3C recommendation, is proposed for authoring ontologies on the Web [5]. OWL DL is a species of OWL, whose

J.Z. Pan et al. (Eds.): JIST 2011, LNCS 7185, pp. 350–357, 2012.

logic foundation is Description Logic (DL), a subset of First-Order Logic (FOL). An *OWL DL ontology* declares axioms and facts for its classes, properties and instances, where a *class* corresponds to a unary predicate in FOL with one free variable, a *property* to a binary predicate with two free variables, and an *instance* to a constant.

Regarding Horn-rules, a *clause* is a disjunction of literals, where a *literal* is a (class or property) predicate that is applied on constants or variables. A *positive* literal is a literal that can be satisfied; otherwise it is *negative*. A clause is called a *Horn-clause* if it has at most one positive literal. A *Horn-rule* is a category of Horn-clause that has one positive literal and at least one negative literal, which can be written as: $H \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n$, where H is called the *head* of the rule, and $L_1 \wedge L_2 \wedge \ldots \wedge L_n$ is called the *body*. $\leftarrow$ gives an *implication* from the body to the head. A *variable binding* is a substitution which maps a variable to a constant in a Horn-rule.

We define the construction of complex mappings as a rule learning process across two ontologies. The mappings between classes or properties in different ontologies are expressed in the form of Horn-rules to reflect their (directional) semantics.

Definition 1. *Let $\mathcal{O}_1, \mathcal{O}_2$ be the source and target ontologies, respectively. Finding complex mappings from $\mathcal{O}_1$ to $\mathcal{O}_2$ is to learn a set of Horn-rules: $\mathcal{M}_{\mathcal{O}_1 \rightsquigarrow \mathcal{O}_2} = \{M_1, M_2, \ldots, M_k\}$. A complex mapping M_i $(i = 1, 2, \ldots, k)$ is as follows:*

$$\mathcal{O}_2 : H \leftarrow \mathcal{O}_1 : L_1 \wedge \mathcal{O}_1 : L_2 \wedge \ldots \wedge \mathcal{O}_1 : L_n,$$

where H, L_j $(j = 1, 2, \ldots, n)$ are literals in $\mathcal{O}_2, \mathcal{O}_1$, respectively. A complex mapping satisfies that its rule body has at least two literals ($n \geq 2$).

In practice, we often match $\mathcal{O}_1$ to $\mathcal{O}_2$, and switch them to match $\mathcal{O}_2$ to $\mathcal{O}_1$. It is worth noting that the above Horn-rule mappings can only cover a kind of "sequential" complex mappings, rather than some "combinational" mappings [3] or formula-like ones [1]. The semantics of Horn-rule mappings may be interpreted using the binding semantics [11], where each ontology has a "subjective semantics" based on local interpretation and a "foreign semantics" based on semantic binding to matched ontologies. In general, reasoning with OWL and Horn-rules is an undecidable problem [6].

A Motivating Example. Fig. 1 shows the source and target ontologies $\mathcal{O}_1, \mathcal{O}_2$. $\mathcal{O}_1$ has two instances `Jianfeng_Chen` and `Yuzhong_Qu` of the class `foaf:Person`, and an instance `Semantic_Web` of `Course`. There are two properties `takeCourse` and `teachBy` that link these instances. $\mathcal{O}_2$ defines in a similar way. Let us assume that $\mathcal{O}_1$: `Jianfeng_Chen` and $\mathcal{O}_2$: `jfchen` denote the same person (which can be identified by some instance matching algorithms), and so do $\mathcal{O}_1$: `Yuzhong_Qu` and $\mathcal{O}_2$: `yzqu`, we can learn complex mappings as follows:

$$\mathcal{O}_2 : \texttt{hasTeacher}(x, y) \leftarrow \mathcal{O}_1 : \texttt{takeCourse}(x, z) \wedge \mathcal{O}_1 : \texttt{teachBy}(z, y),$$
$$\mathcal{O}_2 : \texttt{Student}(x) \leftarrow \mathcal{O}_1 : \texttt{Person}(x) \wedge \mathcal{O}_1 : \texttt{takeCourse}(x, z). \qquad \square$$

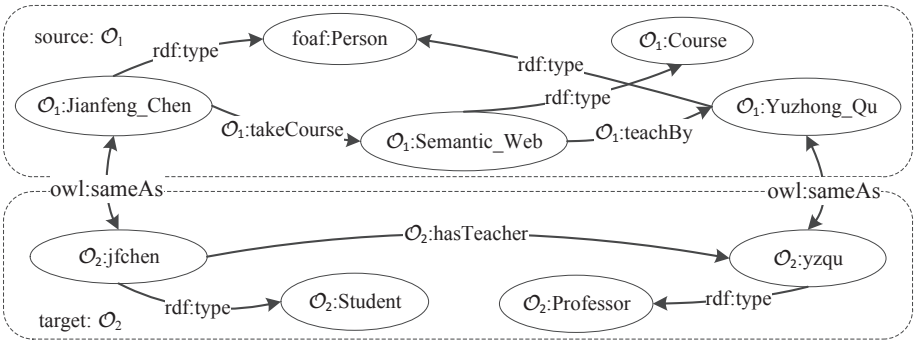

Fig. 1. A motivating example

Related Work. The state of the art ontology matching techniques are limited to detect simple mappings between atomic classes and properties, while finding complex mappings is relatively new. Ritze et al. [9] defined various hand-tailored patterns and used heuristic search methods, and Dhamankar et al. [1] developed a similar system called iMAP. He et al. [3] proposed to apply correlation mining to find co-occurrence entities. Qin et al. [7] gave a systematic approach to create executable mapping rules in the Web-PDDL language. Stuckenschmidt et al. [10] recently summarized the challenges of applying ILP to learn complex mappings, which inspires us to present a concrete approach in this paper.

3 Approach

The outline of our algorithm for learning complex mappings is shown in Algorithm 1, which starts with two ontologies with their instances as input, and its goal is to create as output a set of complex mappings in the form of Horn-rules. In general, the algorithm contains three phases:

- *Instance matching phase* matches instances, which are then used as correspondences between ontologies for finding and validating complex mappings.
- *Data tailoring phase* conducts a class-based extraction mechanism to eliminate irrelevant data from the source ontology.
- *Mapping learning phase* performs a general-to-specific search to construct Horn-rule mappings for classes and properties.

We will describe the details of these phases in the rest of this section.

3.1 Instance Matching Phase

We use a set of OWL terms, namely `owl:sameAs`, `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, `owl:cardinality` and `owl:maxCardinality`, to infer instance mappings (see Line 2 in Algorithm 1). For example, `foaf:mbox` is

Algorithm 1: ComplexMatch

Input: Source and target ontologies $\mathcal{O}_s, \mathcal{O}_t$, including their instances $\mathcal{I}_s, \mathcal{I}_t$.
Output: A set of complex mappings $\mathcal{M}$.

```
1  begin
2      A = MatchInstances(I_s, I_t);                    // Instance matching phase
3      C_t = SelectClassesFromTargetOntology(O_t, A);
4      foreach C ∈ C_t do                                // Data tailoring phase
5          (P_t, I_t^C) = SelectPredicatesAndInstances(C, O_t, I_t, δ); // |I_t^C| ≤ δ
6          (P_s, F_s) = SelectPredicatesAndFactsFromSourceOntology(O_s, I_s, I_t^C, A);
7          foreach P ∈ P_t do                            // Mapping learning phase
8              M = ConstructMapping(P, P_s, F_s, A);
9              Add M to M;
10     return M;
11 end
```

an inverse functional property. If $\mathcal{O}_1$: `Jianfeng_Chen` and $\mathcal{O}_2$: `jfchen` have the same email address, they constitute an instance mapping. These five built-in terms in OWL are frequently used in data fusion systems [4], and we integrate them as a transitive closure to form a set of semantically equivalent mappings.

We employed seven students to evaluate 500 instance mappings in [4], and observed that the average precision of our approach is about 0.95. Additionally, this matching phase can be completed in a few microseconds. Please note that our approach would lost some mappings, which leads to a low recall. However, we argue that the ILP-based learning relies more on the quality of training data than its quantity.

3.2 Data Tailoring Phase

We extract relevant predicates and facts from the source ontology by leveraging the instance mappings (see Line 6 in Algorithm 1). As a result, we reduce both the searching space for candidate predicates and the storage space for negative variable bindings. We propose different strategies for classes and properties to extract the corresponding training data:

- Classes in an ontology represent the conceptual classification. We guide the data tailoring by typing information, e.g., `rdf:type` and `rdfs:subClassOf`. Specifically, we start from each instance in the source ontology and prefer to collect the facts about those instances having the same type.
- Properties express the relationships between instances. We collect the facts for the instances that can constitute property chains, i.e., a set of properties that are chained with compatible `rdfs:domain`(s) and `rdfs:range`(s).
- We also set a boundary for the relevant data search, which is terminated once five facts have been crossed at a search direction. Additionally, for each class, we randomly select at most δ instances (denoted by $\mathcal{I}_t^C$) and associated predicates ($\mathcal{P}_t$) from the target ontology to avoid the combinatorial explosion (see Line 5 in Algorithm 1).

Example. Supposing that the data tailoring phase utilizes $\mathcal{O}_2$: `jfchen` as the start and obtains the matched instance $\mathcal{O}_1$: `Jianfeng_Chen` from $\mathcal{O}_1$. Then, we collect the facts local to $\mathcal{O}_1$: `Jianfeng_Chen`. For the class `Student`, the phase prefers to the facts about the instances that have the same class `foaf:Person` as $\mathcal{O}_1$: `Jianfeng_Chen` does, e.g., $\mathcal{O}_1$: `Yuzhong_Qu`, whose facts can provide the evidence for negative bindings. For the property $\mathcal{O}_2$: `hasTeacher`, it prefers to the fact `takeCourse`(`Jianfeng_Chen`, `Semantic_Web`). □

3.3 Mapping Learning Phase

In Algorithm 2, we learn a complex mapping in the form of Horn-rules (called at Line 8 in Algorithm 1). The algorithm conducts a greedy search to select the best literals, where the goodness of a candidate literal is measured by Gain() in Line 7. This learning process terminates when the number of positive variable bindings versus negative ones is insignificant or the length of rule body exceeds a threshold (e.g., $\epsilon = 10, \theta = 7$ in our case).

Algorithm 2: ConstructMapping

Input: A target predicate P, a set of source predicates $\mathcal{P}_s$, a set of source facts $\mathcal{F}_s$ and a set of instance mappings $\mathcal{A}$.
Output: A complex mapping M.

```
1  begin
2      Initialize a Horn-rule M: P ←;
3      PVB(M) = |PositiveVariableBindings(M, P_s, F_s, A)|;
4      NVB(M) = |NegativeVariableBindings(M, P_s, F_s, A)|;
5      while PVB(M)/NVB(M) > ε && BodyLength(M) < θ do
6          L = GenerateAllCandidateLiterals(M, P_s);
7          L_max = arg max_{L∈L} Gain(L, M);
8          Append L_max to the body of M;
9          Update PVB and NVB w.r.t. the new M_{L_max};
10     return M;
11 end
```

Whether appending a new literal to the current mapping is evaluated based upon the numbers of positive and negative variable bindings, with an objective to cover more positives and fewer negatives. The two sets of bindings keep updating during the learning. More specifically, our algorithm constructs positive variable bindings in terms of the instance mappings, whereas inferring negative variable bindings in terms of the constants extracted from ontology facts. For each matched instance within the source ontology, we create the negative variable bindings that only refer to the constants from its close facts. For example, in Fig. 1 $\{x/$`Jianfeng_Chen`$, y/$`Yuzhong_Qu`$\}$ is a positive variable binding for $\mathcal{O}_2$: `hasTeacher`(x, y), while $\{x/$`Jianfeng_Chen`, $y/$`Semantic_Web`$\}$ and $\{x/$`Yuzhong_Qu`$, y/$`Jianfeng_Chen`$\}$ are negative ones, due to no corresponding fact can be found in the source ontology. It worth noting that all constants in

the negative variable bindings are local to `Jianfeng_Chen`, departing from those of other instances, such as the constants from the facts close to `Hang_Zhang`.

We use the FOIL gain [8] to measure whether a candidate literal should be appended, which is computed as follows:

$$\mathsf{Gain}(L, M) = t \times (\log \frac{PVB(M_L)}{PVB(M_L) + NVB(M_L)} - \log \frac{PVB(M)}{PVB(M) + NVB(M)}),$$

where $PVB(M), NVB(M)$ are the numbers of positive and negative variable bindings of the complex mapping M, respectively. t is the number of positives of M that are still covered after appending a new literal L to M (denoted by M_L).

Example. Let $\mathcal{O}_2$: `hasTeacher`$(x, y) \leftarrow$ be the initial complex mapping with an empty body. Given the three possible constants in $\mathcal{O}_1$, there are nine possible variable bindings for the initial mapping, where only one is positive. PVB is 1 and NVB is 8. Consider the candidate literals $\mathcal{O}_1$: `foaf:Person`(x) and $\mathcal{O}_1$: `takeCourse`(x, z). To add `foaf:Person`, PVB remains 1 but NVB reduces to 5, so its Gain equals 0.85. Similarly, the Gain of `takeCourse` is 1.58. The new complex mapping (in progress) is: $\mathcal{O}_2$: `hasTeacher`$(x, y) \leftarrow \mathcal{O}_1$: `takeCourse` (x, z). □

4 Evaluation

We developed the proposed approach in Java and conducted empirical evaluation on its performance. Our three test cases are constituted by ontologies in various domains, namely Restaurants from IM@OAEI2010, SwetoDBLP vs. ESWC, and Mondial vs. Factbook, each of which contains tens of classes or properties and thousands of instances.

Table 1. Statistics of three datasets

$\mathcal{O}_1$	#Classes	#Props.	#Insts.	$\mathcal{O}_2$	#Classes	#Props.	#Insts.
Restaurant1	3	7	339	Restaurant2	3	7	2,256
SwetoDBLP	17	46	83	ESWC	35	31	1,261
Mondial	17	40	15,398	Factbook	36	171	24,990

We found six complex mappings from the three cases excluding simple 1 : 1 ones, where the average body length is 2.17. According to human observation, all the mappings are correct, so the precision equals to 1.0. It also shows that the number of complex mappings are relatively few, because the schemas of the ontologies in the experiment are not complicated. However, we believe that, with the wider acceptance of OWL 2 [5], especially the property chain feature, there will be more complex mappings that can be learnt from our ILP-based method. So, an interesting future work is to see what kinds of complex mappings can be obtained in more realistic cases.

We also counted the learning time w.r.t. the number of instance mappings, and found that our approach accelerated twice more than the original approach that does not tailor irrelevant data. The optimized approach spent nearly two minutes to complete the learning on 100 instance mappings.

We illustrate two complex property mappings for Restaurants and Mondial vs. Factbook respectively, which give some clear meanings and are difficult to be logically entailed by the union of simple mappings and the input ontologies:

$$\mathcal{O}_{\text{Rest1}} : \texttt{city}(x, y) \leftarrow \mathcal{O}_{\text{Rest2}} : \texttt{is\_in\_city}(x, z) \wedge \mathcal{O}_{\text{Rest2}} : \texttt{name}(z, y),$$
$$\mathcal{O}_{\text{Mondial}} : \texttt{neighbor}(x, y) \leftarrow \mathcal{O}_{\text{Factbook}} : \texttt{border}(x, z) \wedge \mathcal{O}_{\text{Factbook}} : \texttt{country}(z, y).$$

Additionally, a complex class mapping for SwetoDBLP vs. ESWC is as follows:

$$\mathcal{O}_{\text{ESWC}} : \texttt{InProceedings}(x) \leftarrow \mathcal{O}_{\text{DBLP}} : \texttt{Publication}(x) \wedge \mathcal{O}_{\text{DBLP}} : \texttt{isIncludeIn}(x, y) \wedge \mathcal{O}_{\text{DBLP}} : \texttt{Proceedings}(y).$$

These complex mappings can be easily translated to SPARQL queries, such as using the Named Graph to query the two ontologies and fuse the query results together in applications.

5 Conclusion

In this paper, we proposed a new approach for learning complex mappings between ontologies. We utilized instance mappings as training data and applied data tailoring to improve efficiency. We implemented the approach and experimentally evaluated it on three pairs of real-world ontologies, which showed the high precision and clear meanings of the constructed Horn-rule mappings. The work reported here is a first step, and many issues still need to be addressed in future. For example, we look forward to exploring more possible semantics and rules for learning and testing our approach steadily on new datasets in Linked Data. We also want to formally analyze the learning complexity of our approach, especially for the negative example construction.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61003018 and 60903010, in part by the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20100091120041, and in part by the Natural Science Foundation of Jiangsu Province under Grant Nos. BK2011189 and BK2009268.

References

1. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: iMAP: Discovering Complex Semantic Matches Between Database Schemas. In: SIGMOD 2004, pp. 383–394 (2004)
2. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)

3. He, B., Chang, K.C.-C.: Automatic Complex Schema Matching Across Web Query Interfaces: A Correlation Mining Approach. ACM Transactions on Database Systems 31(1), 346–395 (2006)
4. Hu, W., Chen, J., Qu, Y.: A Self-training Approach for Resolving Object Coreference on the Semantic Web. In: WWW 2011, pp. 87–96 (2011)
5. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (2009)
6. Motik, B., Horrocks, I., Rosati, R., Sattler, U.: Can OWL and Logic Programming Live Together Happily Ever After? In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 501–514. Springer, Heidelberg (2006)
7. Qin, H., Dou, D., LePendu, P.: Discovering Executable Semantic Mappings Between Ontologies. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 832–849. Springer, Heidelberg (2007)
8. Quinlan, J.R.: Learning Logical Definitions from Relations. Machine Learning 5(3), 239–266 (1990)
9. Ritze, D., Meilicke, C., Šváb-Zamazal, O., Stuckenschmidt, H.: A Pattern-Based Ontology Matching Approach for Detecting Complex Correspondences. In: ISWC Workshop on Ontology Matching (2009)
10. Stuckenschmidt, H., Predoiu, L., Meilicke, C.: Learning Complex Ontology Alignments – A Challenge for ILP Research. In: ILP 2008 (2008)
11. Zhao, Y., Wang, K., Topor, R., Pan, J.Z., Giunchiglia, F.: Semantic Cooperation and Knowledge Reuse by Using Autonomous Ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 666–679. Springer, Heidelberg (2007)