An Instance-based Learning Approach for Ontology Matching

Pedro H. R. de Assis[1], Alberto H. F. Laender[2]

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
[2] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
passis@inf.puc-rio.br, laender@dcc.ufmg.br

Abstract. This paper proposes an instance-based learning approach for the ontology matching problem. This approach is applicable to scenarios where instances of the ontologies to be matched are exchanged between sources. An initial population of instances is used as a training set of a non-supervised algorithm that constructs mappings between properties of classes from the ontologies. To demonstrate the effectiveness of our approach, we carried out a set of experiments in which we varied the size of the training set for two different ontologies. Our results showed that our approach achieves low false positive rates for sufficient large datasets, although it is totally dependent on the heterogeneity of the domains of properties.

Categories and Subject Descriptors: H.Information Systems [**H.m. Miscellaneous**]: Databases

General Terms: Semantic Web, Languages

Keywords: ontology matching, learning methods, semantic web

1. INTRODUCTION

The paradigm for publishing data on the Web has been changed from publishing isolated data to publishing information that is linked to other resources and data [Bizer et al. 2009]. By doing this, we share knowledge by publishing and accessing documents as part of a global information space. This vision is called the Semantic Web, in which all data published has a structure and semantics are described by ontologies [Berners-Lee et al. 2001; Breitman et al. 2006].

One of the challenges of the Semantic Web is to find semantic mappings among ontologies. By the de-centralized nature of the Semantic Web, there is a considerable number of ontologies that describe similar domains using different terminologies. The problem of finding a semantic correspondence between elements characterizes the problem of ontology matching [Euzenat and Shvaiko 2010].

In this paper, we present a learning approach that analyzes instances of two ontologies in order to perform the matching task. Our algorithm evaluates the creation of new instances based on an example dataset composed of instances of two given ontologies. This is done by establishing mappings that can be either temporary or definitive, based on similarity of classes properties and thresholds. As long as new instances are created, the algorithm converges to definitive mappings. In our experiments, we demonstrate the effectiveness of our approach by assessing the impact of varying the size of the learning set in two different ontology datasets. We show that for sufficiently large datasets the false positive rates are low and that our method is more effective for matching properties with distinct domains.

The rest of the paper is organized as follows. Section 2 provides a brief characterization of the ontology matching problem. Section 3 describes our proposed approach and Section 4 our experimental results. Section 5 addresses related work. Finally, Section 6 presents our conclusions and gives some directions for future work.

2. THE ONTOLOGY MATCHING PROBLEM

Ontologies have been used in several areas of computer science [Breitman et al. 2006]. Considering the divergences among all areas that employ the concept of ontology, a wide range of definitions have been proposed. Even though all of those definitions have divergences, they share certain common concepts. In this paper, we used the definition provided in [Mcguinness and van Harmelen 2004]. It states that ontologies should provide descriptions for the following elements: classes ("things") in the various domains of interest, relationships among classes and properties that classes contain.

Two different ontologies may contain classes that share the same concept. Applications that require an alignment between those classes need strategies to perform the ontology matching. This is a well-known problem in the literature [Bernstein et al. 2000; Lenzerini 2002; Kalfoglou and Schorlemmer 2003; Bouquet et al. 2004; Euzenat and Shvaiko 2010]. An ontology matching solution finds an alignment A' for a pair of ontologies O_1 and O_2 that makes references between classes and properties of O_1 and O_2. Most of the time, an external agent must specify parameters such as metrics, thresholds and external resources that are required by the matching solution.

Among all the types of ontology matching solution, an instance-based matching technique evaluates instances of both ontologies using a given metric to find an alignment. In general, this kind of approach is a sub-category of string-based techniques which classifies our algorithm.

3. PROPOSED APPROACH

In this section, we present the algorithm that implements our instance-based learning approach for ontology matching. Without loss of generalization, we focus on two given classes from each ontology. As we shall see, our algorithm performs comparisons on instances for each pair of properties of these classes.

Formally, let C_1 and C_2 be two classes from ontologies O_1 and O_2, respectively. Each class will be represented by its set of properties $P = \{p_1, \cdots, p_n\}$. We assume that every instance of a given class has at least one value assigned to a property.

We, then, define an operator ϕ that takes a class C_i and a property p_j and returns a list of strings S_{ij} associated to the pair (i, j) for every instance:

$$\phi : (C_i, p_j) \mapsto S_{ij}.$$

Furthermore, in order to determine how similar two lists of strings are, we need a function m to compare them. Therefore m has to be defined as:

$$m : (S_{ij}, S_{kl}) \mapsto r_{ijkl}$$

$$r_{ijkl} \in \mathbb{R}_+.$$

Although m is key to the effectiveness and performance of our algorithm, we take it as an input defined by the user.

Next, we define a weighted complete bipartite graph $G = (P_1 + P_2, E)$, where nodes represent properties of unmatched classes. The weight between two properties $p_i \in P_1$ and $p_j \in P_2$ is defined by:

$$w(p_i, p_j) = m(\phi(C_i, p_i),\ \phi(C_j, p_j))$$

in other words, the weight between two properties p_i and p_j is the distance given by the metric m, between the set of strings associated to values of p_i and p_j in instances of classes C_i and C_j respectively.

Algorithm 1: Machine learning algorithm for ontology matching

```
Input: Source and target ontologies $O_1$ and $O_2$ with sets of instances $S_1$ and $S_2$ associated to
       properties $P_1 = \{p_{11}, \cdots, p_{1n}\}$ and $P_2 = \{p_{21}, \cdots, p_{2m}\}$, respectively.
A metric $m$ and a threshold $\theta$.
Output: A set of mappings between properties of classes of ontologies $O_1$ and $O_2$.
1  $M \leftarrow \emptyset$, the set of definitive matching
2  $T \leftarrow \emptyset$, the set of temporary matching
3  foreach $s \in S_2$ do
4      build a complete bipartite graph $G = (P_1 + P_2, E)$
5      compute weights $w(p_i, p_j) \leftarrow m(\phi(C_i, p_i), \phi(\{s\}, p_j))$, for all $p_i, p_j \in G$
6      while $G$ is not empty do
7          take the edge $(p_i, p_j)$ of highest weight $w'$
8          if $w' \geq \theta$ then
9              $M[p_i] \leftarrow p_j$
10             $P_1 \leftarrow P_1 - \{p_i\}$
11             $P_2 \leftarrow P_2 - \{p_j\}$
12             remove $T[p_i]$ if it exists
13         else
14             $T[p_i] \leftarrow p_j$
15         end
16         remove the nodes $p_i$ and $p_j$ from $G$
17     end
18     insert $s$ in $S_1$ according to the maps $M$ and $T$.
19 end
20 return $M \cup T$
```

The cost of building the graph G for every (p_i, p_j) as stated above is high, since the ϕ function takes all the instances available for both ontologies. For this reason, we adopt a learning approach to reduce costs and provide a good match, since we apply a condition for which very high matching thresholds can be chosen.

The idea behind our approach is to sequentially "insert" an instance from the target ontology into the source ontology. By doing that, the computation of w is reduced and, if this value is higher than a given threshold, the matching between two properties can be established and no longer computed again.

The algorithm then performs, for each instance of the target ontology, the computation of $w(p_i, p_j)$, $\forall p_i \in P_1, p_j \in P_2$. The list of all weights is sorted and then the largest value is taken. An edge (p_i, p_j) represents a mapping (not necessarily definitive), and p_i and p_j can be removed from the graph. This procedure continues until the last two nodes of the graph remain. If one of those chosen weights are higher than a given threshold, the respective properties will not be part of future comparison graphs. As long as new instances of O_2 are inserted into O_1, respecting temporary mappings, there is a contribution to the increase of the value of $w(p_i, p_j)$ for the ideal mapping (p_i, p_j). This continuously tends to increase this weight to a value higher than a given threshold and then converges to a definitive mapping. The complete technique is formalized in Algorithm 1.

As an example, let O_1 be an ontology with a class C_1 conceptualizing person. Let the properties of this class be $\{p_{11} = \text{name}, p_{12} = \text{age}, p_{13} = \text{city}\}$. Now, let O_2 be a target ontology with another class C_2 also conceptualizing person but with the following properties: $\{p_{21} = \text{full\_name}, p_{22} = \text{native\_language}, p_{23} = \text{registered\_age}, p_{24} = \text{origin\_city}\}$.

Let S_1 and S_2, be sets of instances for the ontologies O_1 and O_2 as follows:

Table I. Similarity according to m for instances of S_1 and the first instance of S_2

$w(p_i, p_j)$	**full_name**	**native_language**	**registered_age**	**origin_city**
name	0.72	0.67	0	0.60
age	0	0	0.91	0
city	0.64	0.55	0	0.81

S_1(name, age, city) = {(Peter Lane, 23, Rio de Janeiro), (Brian Jones, 30, New York), (Mark Mills, 18, San Francisco)};

S_2(full_name, native_language, registered_age, origin_city) = {(Barbara Webber, english, 19, London), (Laurent Patel, french, 33, Paris)}.

For those sets, we have to apply the operator ϕ for each class in O_1:

$\phi(C_1, \text{name})$ = {Peter Lane, Brian Jones, Mark Mills}

$\phi(C_1, \text{age})$ = {23, 30, 18}

$\phi(C_1, \text{city})$ = {Rio de Janeiro, New York, San Francisco}

The complete bipartite graph for the matching of the classes C_1 and C_2 as stated in line 4 of Algorithm 1 is depicted in Fig 1.

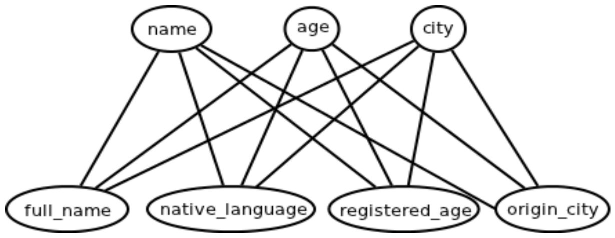

Fig. 1. The complete bipartite graph $G = (E_1 + E_2, V)$ of properties of classes C_1 and C_2

Considering O_2 as the target ontology, the first iteration of the algorithm takes the first instance *(Barbara Webber, english, 19, London)* and calculates all values of $w(p_i, p_j)$ in line 5 of the Algorithm 1. For example, the calculation of $w(name, full\_name)$ for the first instance of S_2 is:

$$w(name, full\_name) = m(\{Peter\,Lane, Brian\,Jones, Mark\,Mills\}, \{Barbara\,Webber\})$$

Suppose that, for a given metric $m \in [0, 1]$, we have the weight values in Table I. The edge with the highest weight in the graph G is *(age, registered_ age)* with value 0.91. If we set a threshold $\theta = 0.90$, we can establish a definitive mapping between the properties *age* and *registered_ age*.

The edge with the second highest weight in the graph is *(city, origin_ city)* with a value of 0.81. This value is not higher than the θ, so the algorithm does not establish a definitive mapping, although it will map *city* to *origin_ city* and will insert this instance of S_2 into S_1 according to this mapping. The learning algorithm lies on the idea that as long as new instances of S_2 with property *origin_ city* are mapped to *city* in S_1, the value of $w(\text{city}, \text{origin\_city})$ tends to increase to a value higher than θ, establishing a definitive mapping between these two properties.

If a mapping is set between two properties (p_i, p_j), neither p_i nor p_j is evaluated again in the current algorithm iteration, regardless of the kind of the mapping. The iterations of the algorithm continues to process until at least all the properties are fully mapped. Then the instance of S_2 is inserted into S_1 according to the mappings.

The algorithm ends when either definitive mappings are set to all the classes of at least one ontology or all the instances of the target class is processed.

4. EXPERIMENTS

In our experiments, we evaluated the effectiveness of our method. We built four ontologies: two of them conceptualizing books and the other ones conceptualizing movies. A total of 785 instances were extracted from Google Books[1] and over $10,000$ instances about movies were extracted from IMDB database[2].

For the book ontology, we defined only one class with the following properties: {*Title*, *Year*, *Country*, *Language*, *Director*}. Analogously, the movie ontology received just one class with the following properties: {*Title*, *Authors*, *Pages*, *Year*}. Some of the instances may not have defined values for some properties, but at least one property has an associated value. Although we could have used existing ontologies for these experiments (e.g., The Bibliographic Ontology[3] and The Movie Ontology[4]), we defined our own ontologies to fit our datasets in order to perform faster experiments to fairly evaluate the effectiveness of our algorithm.

We also created a general purpose ontology with only one class and general properties $\{p_1, p_2, p_3, p_4\}$ to play the role of the target ontology in the matching process. Part of the instances were randomly inserted in this general purpose ontology so that we could run our algorithm and analyze its results. We also explored the impact of the number of initial instances in the source ontology.

For all the experiments, we used the cosine distance as our similarity metric. The metric was defined with n dimensions where n is the number of different characters (case insensitive) presented in the operands.

The diversity of the values in the instances can be verified in Table II. The first column in both tables indicate the number of instances in the source ontology. The values for the remaining columns show the similarity of a group of randomly chosen instances and the source ontology instances. The first line of Table II (left), for example, indicates the distance mean value of 100 instances to a set of randomly chosen instances. As we can notice, in general, the similarity is proportional to the number of instances in the source ontology.

#	Title	Year	Pages	Authors
100	0.9247	0.9007	0.6672	0.9701
200	0.9288	0.8862	0.6549	0.9685
300	0.9313	0.8790	0.6673	0.9679
400	0.9320	0.8923	0.6967	0.9662
500	0.9330	0.8852	0.7336	0.9706
600	0.9345	0.8688	0.7249	0.9718
700	0.9469	0.8503	0.7253	0.9723

#	Title	Year	Country	Language	Director
1,000	0.9075	0.9426	0.9967	0.9998	0.8824
2,000	0.9064	0.9423	0.9969	0.9998	0.8835
3,000	0.9072	0.9424	0.9971	0.9998	0.8838
4,000	0.9080	0.9427	0.9967	0.9998	0.8830
5,000	0.9081	0.9426	0.9968	0.9998	0.8833
6,000	0.9879	0.9428	0.9969	0.9998	0.8836
7,000	0.9083	0.9429	0.9973	0.9998	0.8838
8,000	0.9085	0.9427	0.9970	0.9998	0.8837
9,000	0.9084	0.9428	0.9971	0.9998	0.8840
10,000	0.9086	0.9430	0.9974	0.9998	0.8841

Table II. Threshold impact on the number of instances in the book ontology (left) and movies ontology (right)

[1]http://books.google.com
[2]http://www.imdb.com
[3]http://bibliontology.com
[4]http://www.movieontology.org

4.1 The book ontology

In this set of experiments we randomly chose groups of 100 instances to insert on the general target ontology with a class with properties $\{p_1, p_2, p_3, p_4\}$. The correct mapping is *Title* $\rightarrow p_1$, *Year* $\rightarrow p_2$, *Pages* $\rightarrow p_3$ e *Authors* $\rightarrow p_4$.

Table III shows the average rate of correct matchings achieved by the algorithm for four sets of 100 random instances presented in the target ontology. The number of instances in the source ontology varies according to the first column of Table III. The values for *Title* and *Authors* are symmetric, since their domains contain a high frequency of letters. Analogously, *Year* and *Pages* have the same domain (natural numbers). By this symmetry, if a mapping is incorrect, its corresponding property of same domain misses a mapping as well.

The results expose a limitation in our method. The domain of class properties of ontologies is crucial to the effectiveness of the method. To be more effective, the domains must be as disparate as possible. Otherwise, a false definitive mapping may occur early in the learning process implying a high miss rate.

#	Title	Year	Pages	Authors
100	92.4%	82.4%	82.4%	92.4%
200	93%	98.6%	98.6%	93%
300	90.2%	98%	98%	90.2%
400	90%	97.8%	97.8%	90%

Table III. Correct mapping rate for different numbers of instances in the source book ontology

Table IV shows definitive mappings occurred for each of the five groups of 100 instances for each property. A line on the table, for example, in the property *Title* containing "(p_1, 15)", implies that after processing the 15th instance of the target ontology, a definitive mapping between *Title* and p_1 was established. After this definition, neither *Title* nor p_1 was inserted into the graph G by the Algorithm 1. The results presented in Table IV are influenced by the randomly chosen instances and the number of instances in the source ontology.

#	Title	Year	Pages	Author
1	(p_1, 15) / (p_1, 15)	(x, x) / (x, x)	(p_3, 98) / (p_3, 98)	(p_4, 3) / (p_4, 4)
2	(x, x) / (x, x)	(x, x) / (x, x)	(x, x) / (x, x)	(x, x) / (x, x)
3	(p_1, 55) / (p_1, 55)	(p_2, 8) / (p_2, 8)	(p_3, 44) / (x, x)	(p_4, 55) / (x, x)
4	(p_1, 99) / (p_1, 99)	(x, x) / (x, x)	(x, x) / (x, x)	(x, x) / (x, x)
5	(x, x) / (x, x)	(p_2, 9) / (p_2, 9)	(p_3, 45) / (x,x)	(x, x) / (x,x)

Table IV. Definitive mappings (p_i, n_i) / (p_j, n_j) for 300 and 400 instances, respectively. (p, n) indicates that the property p was permanently aligned after analyzing the n-th instance. (x, x) indicates that no definitive alignment occurred.

4.2 The movie ontology

The following set of experiments are the same applied for the book ontology. In this case, we used the movie ontology described at the beginning of this section. Here, the general target ontology has five properties, $\{p_1, p_2, p_3, p_4, p_5\}$, that must be mapped as: *Title* $\rightarrow p_1$, *Year* $\rightarrow p_2$, *Country* $\rightarrow p_3$, *Language* $\rightarrow p_4$ e *Director* $\rightarrow p_5$.

Table V shows the average rate of correct mappings achieved by the algorithm for all five sets of $1,000$ random instances presented in the target ontology. Due to the small dimension of the domains and the small number of different instances, the properties *Year*, *Country* and *Language* were easily matched. On the other hand, the properties *Title* and *Director*, that are more diverse, shared misses symmetrically. These results reinforce the importance of heterogeneity of domains in the effectiveness of our method.

#	Title	Year	Country	Language	Director
1,000	54.27%	100%	100%	100%	54.27%
2,000	46.77%	100%	100%	100%	46.77%
3,000	75.92%	100%	99.5%	100%	75.92%
4,000	73.94%	100%	99.6%	100%	73.94%
5,000	74.02%	100%	100%	100%	74.06%

Table V. Correct mapping rate for different numbers of instances in the source movie ontology

Table VI shows definitive mappings for each of the five groups of $1,000$ instances for every property. The results, again, are influenced by the randomly chosen instances and the number of instances in the source ontology. For movie instances, we can notice the impact on the domains of *Title* and *Director* in a small number of instances in the source ontology. Incorrect definitive mappings were calculated affecting negatively more than half of the matchings. Although these domains are similar, a big number of instances in the source ontology provided better results.

#	Title	Year	Country	Language	Director
1	(x, x) / (x, x)	(p_2, 36) / (p_2, 36)	(p_3, 389) / (p_3, 29)	(p_4, 3) / (p_4, 4)	(x, x) / (x, x)
2	(p_5, 848) / (p_1, 848)	(p_2, 79) / (p_2, 79)	(p_3, 4) / (p_3, 4)	(p_4, 4) / (p_4, 4)	(p_1, 848) / (p_5, 848)
3	(x, x) / (x, x)	(p_2, 60) / (p_2, 334)	(p_3, 3) / (p_3, 3)	(p_4, 4) / (p_4, 3)	(x, x) / (x, x)
4	(p_1, 140) / (p_1, 140)	(x, x) / (x, x)	(p_3, 56) / (p_3, 63)	(p_4, 4) / (p_4, 4)	(x, x) / (x, x)
5	(p_1, 803) / (p_1, 803)	(p_2, 2) / (p_2, 2)	(p_3, 5) / (p_3, 7)	(p_4, 4) / (p_4, 4)	(p_5, 803) / (p_5, 803)

Table VI. Definitive mappings (p_i, n_i) / (p_j, n_j) for $4,000$ and $5,000$ instances, respectively. (p, n) indicates that the property p was permanently aligned after analyzing the n-th instance. (x, x) indicates that no definitive alignment occurred.

5. RELATED WORK

In this section, we briefly address related work that proposes strategies for ontology matching. For instance, [Euzenat and Shvaiko 2010] provide a good survey on the ontology matching problem.

[Doan et al. 2004] applied a machine learning approach for several groups of information. They used a multi-strategy learning approach that might present high costs but achieved good results. [Parundekar et al. 2010] explored the space of hypothesis supported by the existing equivalence statements to create equivalence and subsumption relationships between classes of two different ontologies. Although interesting results were found, if there is a lack of logical statements defined in the ontologies, this method may not be efficient.

The approach presented by [Jain et al. 2011] was based on a combination of the use of external resources and a proposed metric that exploit contextual information of the ontologies. Their method presented good matching results. However, the cost of using external resources in comparisons with the ontology sub-classes structures may invalidate its use in certain applications. The use of external resources to improve semantic analysis is also proposed by [Bouquet et al. 2003]. The matching of concept hierarchies is done in two steps: the building of semantic hierarchy models of concepts and the comparison problem which is encoded in a satisfatibility problem. Although the complexity and usage of resources are high, the main advantage of this method is that it performs fairly well when less data is available.

In general, the main contribution of our approach in comparison to state-of-the-art methods is that we can easily adapt the alignment if a new instance of the target ontology is produced.

6. CONCLUSION

In this paper we proposed an instance-based learning approach for the problem of ontology matching. Our algorithm takes as input two classes of ontologies, a metric and a threshold and outputs an alignment for the classes. For assessing the effectiveness of our approach, we conduct a set of experiments to evaluate the influence of the number of instances in the source ontology and the influence of heterogeneity of domains. Our results showed that the algorithm proposed is intrinsically dependent on these aspects.

As we can see from our results, the combination of a small number of instances in the source ontology and properties with similar domains can lead to erroneous definitive mappings compromising the complete matching process. On the other hand, if a combination of a sufficient number of instances is provided, the algorithm presents high rates of correct mappings.

As future work, we want to explore language-based methods to improve our results but maintaining its feasibility for general purpose applications. Using extrinsic methods such as the use of dictionaries, lexicons and terminologies we can vary the results of $w(p_i, p_j)$ enough to provide an extra semantic analysis over the simple use of a single string-based metric. We also want to integrates metadata analysis facilities such as data type property definitions, labels and domain constraints in order to improve the matching process. Also, we want to study how combining metrics can improve our results, especially, how a set of mappings for each metric can be managed to show better results.

REFERENCES

Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American* 284 (5): 34–43, May, 2001.

Bernstein, P. A., Halevy, A. Y., and Pottinger, R. A. A vision for management of complex models. *SIGMOD Rec.* 29 (4): 55–63, Dec., 2000.

Bizer, C., Heath, T., and Berners-Lee, T. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems* 5 (3): 1–22, 2009.

Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krötzsch, M., Serafini, L., Stamou, G., S ure, Y., and Tessaris, S. Specification of a common framework for characterizing alignment. Knowledge Web Deliverable 2.2.1v2, University of Karlsruhe. Dec., 2004.

Bouquet, P., Serafini, L., and Zanobini, S. Semantic coordination: A new approach and an application. pp. 130–145, 2003.

Breitman, K., Casanova, and Ruszkowski, W. *Semantic Web: Concepts, Technologies and Applications.* Springer, 2006.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Y. Ontology matching: A machine learning approach. In *Handbook on Ontologies.* pp. 385–404, 2004.

Euzenat, J. and Shvaiko, P. *Ontology Matching.* Springer Publishing Company, Incorporated, 2010.

Jain, P., Yeh, P. Z., Verma, K., Vasquez, R. G., Damova, M., Hitzler, P., and Sheth, A. P. Contextual ontology alignment of lod with an upper ontology: a case study with proton. In *Proceedings of the 8th Extended Semantic Web Conference.* ESWC'11. Springer-Verlag, Berlin, Heidelberg, pp. 80–92, 2011.

Kalfoglou, Y. and Schorlemmer, M. Ontology mapping: the state of the art. *Knowl. Eng. Rev.* 18 (1): 1–31, Jan., 2003.

Lenzerini, M. Data integration: a theoretical perspective. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems.* ACM, New York, NY, USA, pp. 233–246, 2002.

Mcguinness, D. L. and van Harmelen, F. OWL Web Ontology Language Overview. W3C Recommendation, 2004.

Parundekar, R., Knoblock, C. A., and Ambite, J. L. Linking and building ontologies of linked data. In *Proceedings of the 9th International Semantic Web Conference.* ISWC'10. Springer-Verlag, Berlin, Heidelberg, pp. 598–614, 2010.