A Self-Training Approach for Resolving Object Coreference on the Semantic Web

Wei Hu
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, 210093, PR China
whu@nju.edu.cn

Jianfeng Chen
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, 210093, PR China
jf_chen@ymail.com

Yuzhong Qu
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, 210093, PR China
yzqu@nju.edu.cn

ABSTRACT

An object on the Semantic Web is likely to be denoted with multiple URIs by different parties. Object coreference resolution is to identify "equivalent" URIs that denote the same object. Driven by the Linking Open Data (LOD) initiative, millions of URIs have been explicitly linked with `owl:sameAs` statements, but potentially coreferent ones are still considerable. Existing approaches address the problem mainly from two directions: one is based upon equivalence inference mandated by OWL semantics, which finds semantically coreferent URIs but probably omits many potential ones; the other is via similarity computation between property-value pairs, which is not always accurate enough. In this paper, we propose a self-training approach for object coreference resolution on the Semantic Web, which leverages the two classes of approaches to bridge the gap between semantically coreferent URIs and potential candidates. For an object URI, we firstly establish a kernel that consists of semantically coreferent URIs based on `owl:sameAs`, (inverse) functional properties and (max-)cardinalities, and then extend such kernel iteratively in terms of discriminative property-value pairs in the descriptions of URIs. In particular, the discriminability is learnt with a statistical measurement, which not only exploits key characteristics for representing an object, but also takes into account the matchability between properties from pragmatics. In addition, frequent property combinations are mined to improve the accuracy of the resolution. We implement a scalable system and demonstrate that our approach achieves good precision and recall for resolving object coreference, on both benchmark and large-scale datasets.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; D.2.12 [**Software Engineering**]: Interoperability; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance

Keywords

Object coreference, object consolidation, self-training, property combination, data fusion

1. INTRODUCTION

The *Semantic Web* is an ongoing effort by the W3C Semantic Web Activity, with the purposes of actualizing data integration and sharing among different applications and organizations. To date, a number of prominent ontologies have emerged for publishing data in specific domains, such as the Friend of a Friend (FOAF), which define common identifiers for classes and properties, in the form of *URIs*, which have been widely used across data sources.

In the instance level, however, it is still far from achieving agreement among data sources on the use of common URIs to identify a specific *object* [11]. In fact, due to the decentralized and dynamic nature of the Semantic Web, it frequently happens that many different URIs from a variety of sources, more likely originating from different RDF documents, denote one real-world object, i.e., represent the same identity. Such examples exist in the domains of personal profiles, academic publications, encyclopedic or geographical resources, etc.

Object coreference resolution, also known as object consolidation or identification [1], is a task for identifying multiple URIs for the same real-world object, i.e., finding *coreferent URIs* which represent a unique identity. Object coreference resolution is important for data-centric applications, such as fusing distributed descriptions of equivalent RDF resources in data integration systems.

Driven by the Linking Open Data (LOD) initiative, millions of URIs from independent data sources have been explicitly interlinked with `owl:sameAs` statements [3]. However, considering billions of object URIs on the current Semantic Web, we observe that there still exist a large amount of URIs which implicitly represent the same objects but have not been connected with `owl:sameAs` yet. For example, at least 70 URIs returned by Falcons search engine [2] denote a person "Tim Berners-Lee", the director of W3C, but only five of them are linked with `owl:sameAs`.

In the field of Semantic Web, recent studies address this problem mainly from two directions: one is based on utilizing standard OWL semantics, such as `owl:sameAs` [8] and inverse functional properties (IFPs) [11]; while the other is according to the intuition that two URIs represent the same real-world object if they share some similar property-value pairs [7, 13]. Generally speaking, the semantics-based way can infer explicitly coreferent URIs but probably misses a lot of potential candidates, while the similarity-based way is not always accurate due to heterogenous ways for expressing the same thing. Hence, a key issue for resolving object coreference on the Semantic Web is: How to *combine* these two classes of techniques for building bridges between coreferent URIs that we already have and potential candidates?

In this paper, we propose a *self-training* approach to leveraging the semantics-based and similarity-based ways for addressing the problem of object coreference resolution on the Semantic Web. Self-training is a well-known class of semi-supervised learning algorithms, in which a learner continues labeling unlabeled examples and re-training itself on an extended labeled training set [28]. Self-training is suitable for solving our problem, because there are abundant unresolved object URIs, but the number of existing semantically coreferent ones is limited.

Specifically, taking an object URI as input, we firstly establish a *kernel* that consists of a set of semantically coreferent URIs based on `owl:sameAs`, (inverse) functional properties and (max-)cardinalities, and then iteratively *extend* the kernel in terms of discriminative property-value pairs in the descriptions of URIs. The *discriminability* of a property-value pair is learnt based on a statistical measurement, which not only exploits the key characteristics for representing an object, but also takes into account the matchability between properties from pragmatics. Furthermore, *frequent property combinations* are mined to enhance the selection criteria of properties during each iteration, so that the accuracy of the resolution is further improved. We develop a scalable system and evaluate its performance on a benchmark dataset from OAEI 2010 and a large-scale dataset that is collected by Falcons search engine in 2008. The experimental results demonstrate that our approach achieves acceptable F-Measure on both datasets, as compared with the performance of six representative competitors.

The remainder of this paper is structured as follows. The self-training framework of our proposed approach is firstly outlined in Section 2. Section 3 introduces a method to find semantically coreferent URIs in terms of OWL semantics. Section 4 describes our self-training algorithm for resolving object coreference with a statistical measurement to calculate the discriminability of a property-value pair. Section 5 presents a way to mine frequent property combinations for improving the accuracy of the resolution. Experimental results on the benchmark and large-scale datasets are reported in Section 6. Section 7 discusses related work and finally Section 8 concludes this paper with future work.

2. OVERVIEW OF THE APPROACH

The architecture of our proposed approach is outlined in Fig. 1, which starts with an object URI u. After three processing stages, the approach returns a set of coreferent URIs that denote the same object as u.

1. **Building a kernel.** We construct a kernel of semantically coreferent URIs for u based on the OWL semantics of `owl:sameAs`, `owl:InverseFunctionalProperty` (`owl:IFP` for short), `owl:FunctionalProperty` (abbr. `owl:FP`), `owl:cardinality` and `owl:maxCardinality`. The five built-in vocabulary elements in OWL are frequently used to infer the equivalence relation in many systems [19], and combining them together establishes a larger initial labeled training set.

2. **Learning discriminative property-value pairs.** It is an iterative process, which firstly learns discriminative property-value pairs from some labeled coreferent URIs, and then uses the pairs to find more coreferent ones. In accordance with previous works in [10, 13, 17], we assume that coreferent URIs share several common property-value pairs, and certain property-value pairs are more useful for coreference resolution.

   For any two URIs, we extract their involved property-value pairs from the dereference documents,[1] and compare these values with a string matching algorithm I-Sub [22]. If the similarity between two values is larger than a threshold, then the related two properties have a kind of commonality. For a set of coreferent URIs, we select a property pair sharing most matchable values, and assign the most common value to each property in this pair. These two property-value pairs reflect some discriminative characteristics for their denoted object, and they are used to find more coreferent URIs.

3. **Choosing properties based on frequent property combinations.** Some properties are more suitable to use together for describing an object, such as longitude and latitude for a coordinate. If we only choose either of them for identifying coreferent URIs, the results tend to be inaccurate. Therefore, we apply association rule mining to discover frequent property combinations with heuristic refinement. For each learning iteration, if any property in a frequent property combination is chosen, the rest property in the combination with its most common value (if existing in the training set) would be complemented. Consequently, these two properties with associated values are used together for searching new coreferent URIs.

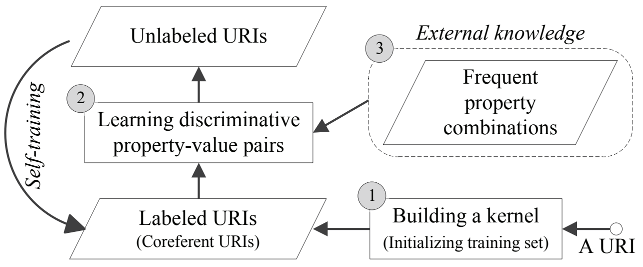
Unlabeled URIs
External knowledge
3
Frequent property combinations
2
Learning discriminative property-value pairs
Self-training
1
Labeled URIs (Coreferent URIs)
Building a kernel (Initializing training set)
A URI

Figure 1: Overview of the proposed approach

Example 1. For illustration purposes, let us consider four RDF documents containing candidate URIs for coreference resolution in Fig. 2. Assuming that `dbpedia:Beijing` is the input object URI for resolution. Through searching for `owl:sameAs` statements, `dbpedia:Beijing` is semantically coreferent with `geo:1816670`. During training, (`rdfs:label`, "Beijing") and (`geo:alternateName`, "Beijing") are learnt in the first iteration as the most discriminative property-value pairs. As a result, `semweb:Beijing` is found. In the second round, (`wgs84_pos:lat`, "40") is the most discriminative pair, and a wrong coreferent result `ex:New_York` is discovered. But considering the frequent property combination {`wgs84_pos:lat`, `wgs84_pos:long`}, `ex:New_York` would not be included any more, because the values of `wgs84_pos:long` are completely different ("116" for Beijing, while "74" for New York).

[1]The act of retrieving a representation of a resource identified by a URI is referred to as dereferencing that URI [15].

```
dbpedia:Beijing  rdfs:label          "Beijing";
                 owl:sameAs          geo:1816670.

    geo:1816670  wgs84_pos:long      "116";
                 wgs84_pos:lat       "40";
                 geo:alternateName   "Beijing";
                 geo:alternateName   "Peking".

 semweb:Beijing  rdfs:label          "Beijing";
                 wgs84_pos:lat       "40";
                 wgs84_pos:long      "116".

    ex:New_York  wgs84_pos:long      "74";
                 wgs84_pos:lat       "40".
```

Figure 2: An illustrating example

3. CONSTRUCTION OF THE KERNEL

Let U be a set of URI references (URIrefs), B be a set of blank node IDs and L be a set of literals. A triple $\langle s, p, o\rangle \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an *RDF triple*. An *RDF graph G* is just a set of RDF triples.

The semantics of `owl:sameAs` indicates that all the URIs linked with this property, in the form of $\langle s, \texttt{owl:sameAs}, o\rangle$, have the same identity, implying that the subject and object should be the same resource.

Definition 1. (The Same-as Relation) Let U be a set of URIs. The same-as relation, denoted by $\mathbb{S}$, is defined as the minimal reflexive, symmetric relation on U, satisfying that: (1) $\forall s \in U$, $\langle s, s\rangle \in \mathbb{S}$; (2) For $s, o \in U$, if there exists a triple $\langle s, \texttt{owl:sameAs}, o\rangle$, then $\langle s, o\rangle \in \mathbb{S}$ and $\langle o, s\rangle \in \mathbb{S}$.

The semantics of an IFP guarantees that a value can only be the value of this property for a single object, that is, two separate objects are indirectly inferred to be identical based on having the same value of that property.

To identify IFPs, we parse ontologies to find the properties whose `rdf:type` is explicitly defined as `owl:IFP`. This is done at the preprocessing time. Note that IFPs can be inferred in a multitude of ways based on OWL semantics. For example, the study in [25] did reasoning over pD* that includes rules for handling `owl:sameAs`, `owl:IFP` and `owl:FP` axioms. But anyone can define anything on the Semantic Web, inferring IFPs across different sources may cause errors and inconsistency. As an example, we could infer `dc:title` as an IFP in the Falcons dataset. The work in [12] studied the reasoning problem of new ontologies published on the Web redefining the semantics of existing entities resident in other ontologies (called *ontology hijacking*), which enlightens us to only use dereferenceable IFPs in our approach for avoiding ontology hijacking.

Definition 2. (The IFP Relation) Let U be a set of URIs. The IFP relation, denoted by $\mathbb{I}$, is defined to be the minimal reflexive, symmetric relation on U, which satisfies the following conditions: (1) $\forall s \in U$, $\langle s, s\rangle \in \mathbb{I}$; (2) For $s_1, s_2 \in U$, if there are an IFP p and two triples $\langle s_1, p, o\rangle$, $\langle s_2, p, o\rangle$, then $\langle s_1, s_2\rangle \in \mathbb{I}$ and $\langle s_2, s_1\rangle \in \mathbb{I}$.

To find the IFP relations, we match the values of objects in the RDF triples with same IFPs as the predicates. If the values are exactly the same by using a trivial string comparison algorithm, we bridge an IFP relation between the subjects of these triples. This approach has been demonstrated to be feasible in [11, 24]. Besides, the lexical forms of some literals can be empty, e.g., the values of `foaf:mbox_sha1sum` are blank in a few triples. We omit these triples for avoiding wrong coreference.

Due to several heterogenous ways for expressing email addresses, we use an ad hoc method to identify identical email addresses for two popular IFPs: `foaf:mbox` and `foaf:mbox_sha1sum`. Given an email address, we compute its sha1sum value and utilize `foaf:mbox_sha1sum` to find new coreferent URIs. We bridge the IFP relations between the URIs using the two IFPs.

The way of using functional properties (FPs) to find coreferent URIs is similar to the way for IFPs. We firstly obtain the dereferenceable properties whose types are explicitly defined as `owl:FP`, and then use these FPs to construct the FP relations.

Definition 3. (The FP Relation) Let U be a set of URIs. The FP relation, denoted by $\mathbb{F}$, is defined to be the minimal reflexive, symmetric relation on U, which satisfies: (1) $\forall o \in U$, $\langle o, o\rangle \in \mathbb{F}$; (2) For $o_1, o_2 \in U$, if there are a FP p and two triples $\langle s, p, o_1\rangle$, $\langle s, p, o_2\rangle$, then $\langle o_1, o_2\rangle \in \mathbb{F}$ and $\langle o_2, o_1\rangle \in \mathbb{F}$.

The cardinality constraint `owl:cardinality` (or `owl:maxCardinality`) is a built-in OWL property which links a restriction class to a data value. A restriction having an `owl:cardinality` (or `owl:maxCardinality`) constraint describes a class of all objects that have exactly (at most) N semantically distinct values for the property concerned, where N is the value of the cardinality constraint. If $N = 1$, its semantics is similar to FPs (however, with respect to a particular class) and can be applied to produce coreferent URIs.

Definition 4. (The Cardinality Relation) Let U be a set of URIs. The cardinality relation, denoted by $\mathbb{C}$, is defined as the minimal reflexive, symmetric relation on U, which satisfies that: (1) $\forall o \in U$, $\langle o, o\rangle \in \mathbb{C}$; (2) For $o_1, o_2 \in U$, if there exist a (max-)cardinality restriction $\langle c, \texttt{owl:onProperty}, p\rangle$, $\langle c, \texttt{owl:cardinality}, \text{“1”}\rangle$ (or $\langle c, \texttt{owl:maxCardinality}, \text{“1”}\rangle$), where c, p are the restriction class and property respectively, and three triples $\langle s, p, o_1\rangle$, $\langle s, p, o_2\rangle$ and $\langle s, \texttt{rdf:type}, c\rangle$, then $\langle o_1, o_2\rangle \in \mathbb{C}$ and $\langle o_2, o_1\rangle \in \mathbb{C}$.

Based on the same-as, IFP, FP and cardinality relations, we define the equivalence relation as follows.

Definition 5. (The Equivalence Relation) Let $\mathbb{S}, \mathbb{I}, \mathbb{F}, \mathbb{C}$ be the same-as, IFP, FP and cardinality relations on a set of URIs U, respectively. $\mathbb{K}$ is the transitive closure on $\mathbb{S} \cup \mathbb{I} \cup \mathbb{F} \cup \mathbb{C}$.

It is worth noting that $\mathbb{K}$ is an equivalence relation on U, because $\mathbb{S}, \mathbb{I}, \mathbb{F}, \mathbb{C}$ are all reflexive and symmetric.

Definition 6. (Kernel) Let U be a set of URIs. For a URI $u \in U$, the equivalence class $[u]_{\mathbb{K}} = \{v \in U \mid \langle u, v\rangle \in \mathbb{K}\}$, under the equivalence relation $\mathbb{K}$, is called a kernel of u.

Based on the definition of the kernel above, it is straightforward to implement a corresponding algorithm for finding semantically coreferent URIs.

4. SELF-TRAINING

For an RDF graph G, we define three operations on G for simplifying our notations:

$$\mathrm{Subj}(G) = \{s \mid \langle s, p, o\rangle \in G\}, \tag{1}$$
$$\mathrm{Pred}(G, s_k) = \{p \mid \langle s_k, p, o\rangle \in G\}, \tag{2}$$
$$\mathrm{Obj}(G, s_k, p_i) = \{o \mid \langle s_k, p_i, o\rangle \in G\}. \tag{3}$$

Our object coreference resolution algorithm is depicted in Algorithm 1, which follows a traditional self-training framework [28]. Inputting a labeled kernel C for an object URI u and a set H of unlabeled URIs, the goal of the algorithm is to iteratively learn the most discriminative property-value pairs for identifying potentially coreferent URIs in H. In the case that there are too many coreferent URIs in C, the algorithm randomly picks up a subset D of C for reducing the computational costs. We choose $|D| = 200$ in terms of the computational capability of our personal computers. The algorithm stops when the iteration times exceeds a maximum number K or all the property pairs have been checked.

Algorithm 1: A coreference resolution algorithm

```
Input: A kernel $C$ for an object URI $u$ in an RDF
       graph $G$, and a set $H$ of unlabeled URIs in $G$.
Output: An extension $E$, after self-training.
1  begin
2      Initialize two empty checked lists $PP$ and $PV$;
3      Copy $C$ to $E$ for training;
4      repeat
5          Create a pool $D$ by randomly choosing at most
           $N$ URIs in $E$;
6          Select the most matchable property pair
           $(p_i, p_j) \notin PP$ by Eq. (4), s.t.
           $p_i \in \bigcup_{s\in D} \mathrm{Pred}(G,s), p_j \in \bigcup_{t\in D, t\neq s} \mathrm{Pred}(G,t)$;
7          if $(p_i, p_j) = NULL$ then break;
8          Assign the most common values $o_i, o_j$ to $p_i, p_j$
           resp. by Eq. (6), s.t. $(p_i, o_i)$ or $(p_j, o_j) \notin PV$,
           $o_i \in \bigcup_{s\in D} \mathrm{Obj}(G,s,p_i), o_j \in \bigcup_{t\in D} \mathrm{Obj}(G,t,p_j)$;
9          if $o_i = NULL$ and $o_j = NULL$ then
10             Push $(p_i, p_j)$ into $PP$, and go back to Line 6;
11         end
12         if $o_i \neq NULL$ then
13             Apply $(p_i, o_i)$ to label a set $P$ of unlabeled
               URIs, s.t. $P = \{s \in H \mid \langle s, p_i, o_i\rangle \in G\}$;
14             if $(p_i, o_i)$ is discriminative by Eq. (7) then
15                 Add $P$ to $E$, and remove $P$ from $H$;
16             end
17         end
18         if $o_j \neq NULL$ then
19             Apply $(p_j, o_j)$ to label a set $Q$ of unlabeled
               URIs, s.t. $Q = \{s \in H \mid \langle s, p_j, o_j\rangle \in G\}$;
20             if $(p_j, o_j)$ is discriminative by Eq. (7) then
21                 Add $Q$ to $E$, and remove $Q$ from $H$;
22             end
23         end
24         Push $(p_i, o_i), (p_j, o_j)$ into $PV$;
25     until iteration times $> K$;
26     return $E$;
27 end
```

Similar to most self-training algorithms, there exist three key measures/parameters that should be discussed: (1) How to measure the discriminability of a property-value pair? (2) How to avoid error accumulation during resolution? and (3) How to determine the maximum iteration times? We answer these questions in the rest of this section.

We propose a three-step way to measure the discriminability of each property-value pair. The intuition behind is that, the more a property-value pair is shared by a set of coreferent URIs, the more likely it represents some discriminative characteristics for the denoted real-world object. However, different URIs often use different properties to describe the same value. For example, `foaf:name` and `dc:title` are both widely used for describing a person's name. Given a set of coreferent URIs, ontology matching techniques [6] could be adopted to discover matchable properties from their values (a so-called *extensional* way), which takes into account the matchability between properties from pragmatics. The most matchable property pair with associated values can be considered as the most important characteristics supported by the training set to identify coreferent URIs.

To formalize, for an RDF graph G, the matchability between two properties p_i, p_j in a labeled set D of G is computed by:

$$\mathrm{Match}(p_i, p_j) = \sum_{\substack{s,t\in D\\ s\neq t}} |\{(o, o') \mid o \in \mathrm{Obj}(G, s, p_i), o' \in \mathrm{Obj}(G, t, p_j), \text{I-Sub}(o, o') > \delta\}|, \tag{4}$$

$$\text{I-Sub}(o, o') = \mathrm{Comm}(\mathrm{Desc}(o), \mathrm{Desc}(o')) - \mathrm{Diff}(\mathrm{Desc}(o), \mathrm{Desc}(o')) + \mathrm{Winkler}(\mathrm{Desc}(o), \mathrm{Desc}(o')), \tag{5}$$

where I-Sub is an improved string comparison method [22], whose novelty is that the similarity between two strings is relevant to their commonalities as well as their differences. Winkler is a correction coefficient. When o is a URI, $\mathrm{Desc}(o)$ extracts its local name, which is a string after the last hash "#" or slash "/" of the URI; when o is a literal, $\mathrm{Desc}(o)$ gets its lexical form. In addition, p_i, p_j can be the same property which is used by different URIs. The most matchable properties are the property pair that has the maximum number of matchable values.

Because a property may have different values, we assign the most common one to each property in a property pair. The matchability of values is also considered, because some value for a property is prevalent in the real world but little supported by a specific training set. For a matchable property pair (p_i, p_j) in a labeled set D of G, the most common value pair (o_i, o_j) for (p_i, p_j) is computed as follows:

$$(o_i, o_j) = \arg\max_{(o,o')} |\{(s, s') \in D \times D \mid \text{I-Sub}(o, o') > \delta, \langle s, p_i, o\rangle \in G, \langle s', p_j, o'\rangle \in G\}|. \tag{6}$$

This equation allows o_i, o_j to be the same value, and s, s' to be the same as well.

The discriminability of a property-value pair is calculated in terms of the number of potentially coreferent URIs that can be found by using such property-value pair. Specifically, let (p_i, o_i) be a property-value pair in G, the discriminability of (p_i, o_i) is calculated as follows:

$$\mathrm{Discr}(p_i, o_i) = \frac{|\{s \in D \mid \langle s, p_i, o_i\rangle \in G\}|}{|\{s \in H \mid \langle s, p_i, o_i\rangle \in G\}|}, \tag{7}$$

where D, H are labeled and unlabeled sets in G, respectively. Here, we use a threshold based on our experiments to decide whether a property-value pair is discriminative or not.

Example 2. Considering Fig. 2, the kernel includes `dbpedia:Beijing` and `geo:1816670`. In terms of Eq. (4), (`rdfs:label`, `geo:alternateName`) is the most matchable property pair in the first iteration, and "Beijing" is the most common value for them. (`rdfs:label`, "Beijing") and (`geo:alternateName`, "Beijing") are the two discriminative property-value pairs for identifying coreferent URIs.

To avoid error accumulation, the selection of discriminative property-value pairs reduces the execution of improper extension. In addition, frequent property combinations are mined for further improvement, which is given in the next section. Regarding the maximum number of iterations, we observed the average number of property-value pairs associated with an object in a large-scale dataset containing 76 million URIs, and found that in average an object is associated with about eight property-value pairs, so we set $K = 10$, which is a little larger than this average.

5. IMPROVEMENT BY FREQUENT PROPERTY COMBINATIONS

In ontology development, some properties are designed to be used together, e.g., `wgs84_pos:long` and `wgs84_pos:lat`; while only use a part of them cannot represent the intended semantics. To improve the accuracy of our coreference resolution, we mine this kind of property combinations and use them as external knowledge to enhance the selection criteria of properties for constructing property-value pairs.

We propose to tailor association rule mining techniques to obtain *binary* associations between properties, which means that each property combination is composed of exactly two different properties. We believe that binary associations are more prevalent and easy to understand, although association rule mining is naturally applicable for n-ary associations.

A binary association rule expresses that the occurrence of a property statistically indicates the presence of another property for the same object URI with certain *confidence*, which can be transformed into a conditional probability. We prefer the *co-occurrence* relation for property combinations, which requires that not only the occurrence of one property implies the other, but vice versa. The co-occurrence relation could be interpreted as an indicator of interdependency of two properties. More specifically, for two properties p_i, p_j in an RDF graph G, the confidence between p_i, p_j is computed as follows:

$$\mathrm{Conf}(p_i, p_j) = \min\{\mathrm{Conf}(p_i \Rightarrow p_j), \mathrm{Conf}(p_j \Rightarrow p_i)\}, \tag{8}$$

$$\begin{aligned}\mathrm{Conf}(p_i \Rightarrow p_j) = \Pr(p_j \mid p_i) &= \frac{\mathrm{Support}(p_i \cup p_j)}{\mathrm{Support}(p_i)}\\ &= \frac{|\{s \in \mathrm{Subj}(G) \mid p_i, p_j \in \mathrm{Pred}(G, s)\}|}{|\{s \in \mathrm{Subj}(G) \mid p_i \in \mathrm{Pred}(G, s)\}|},\end{aligned} \tag{9}$$

when $\mathrm{Conf}(p_i, p_j)$ is greater than a predefined threshold, we say $\{p_i, p_j\}$ is a *frequent property combination*, reflecting its significance in statistics. Different from the goal of conventional association rule mining, we select a high threshold in our case (e.g., 0.98 in our experiments), which tends to find common combinations in data, rather than some surprising or obscure patterns.

Previous studies demonstrated that standard association rule mining techniques can discover numerous spurious patterns when being applied to random data and to real-world data [27]. From our dataset, we also observed a similar phenomenon. For example, several social networking sites such as `hi5.com` and `livejournal.com` exported a large volume of RDF data for describing users, which led to many spurious frequent property combinations, such as {`foaf:name`, `foaf:mbox`}. These combinations are not tightly co-related in semantics, and using them together can cause over-fitting in self-training. Therefore, we propose two heuristic rules to refine the pre-found combinations.

Firstly, we investigate the data ranges of properties defined in their dereference documents, and assume that two properties in a frequent property combination are co-related in semantics if their ranges are compatible. A data range belongs to one of the six categories: a URI, a float, an integer, a string, a date time and a thing, where thing is compatible with the other five disjoint categories. We also remove the frequent property combinations that contain built-in properties in RDF(S), OWL and DC.

Secondly, we perform a statistical analysis for the use of properties and their values. The intuition behind this analysis is that unrelated properties may exhibit divergence in numbers of assigned values associated with objects. For an RDF graph G, let p_i be a property used as a predicate in G, $AV()$ measures the average number of unique values that p_i has, while $AC()$ measures the average cardinality that an object uses p_i:

$$\mathrm{AV}(p_i) = \frac{|\bigcup_{s \in \mathrm{Subj}(G)} \mathrm{Obj}(G, s, p_i)|}{\sum_{s \in \mathrm{Subj}(G)} |\mathrm{Obj}(G, s, p_i)|}, \tag{10}$$

$$\mathrm{AC}(p_i) = \frac{\sum_{s \in \mathrm{Subj}(G)} |\mathrm{Obj}(G, s, p_i)|}{|\{s \in \mathrm{Subj}(G) \mid p_i \in \mathrm{Pred}(G, s)\}|}. \tag{11}$$

If $AC(p_i) \simeq 1$, p_i is called a quasi-functional property in [13]. When two properties are semantically co-related, they should have close values of AVs and ACs, e.g., the difference is less than 0.1 in our experiments.

Example 3. Considering the example in Fig. 2, the average number of values and the average cardinality for `wgs84_pos#lat` are:

$$AV(\texttt{wgs84\_pos\#lat}) = \frac{1}{3}, \quad AC(\texttt{wgs84\_pos\#lat}) = 1.$$

The refined frequent property combinations are stored beforehand. In self-training, when the most matchable property pair (p_i, p_j) is selected (see Line 6 in Algorithm 1), we search the frequent property combinations for p_i, p_j, respectively. If both of them have counterparts in these frequent property combinations, say (p'_i, p'_j), their most common values (if exists) are extracted. Therefore, $\{p_i, p'_i\}$ and $\{p_j, p'_j\}$ with their associated values are used for finding coreferent URIs. It usually happens when p_i, p_j are the same property, and p'_i, p'_j are probably also the same.

Table 1: Statistical Data of Benchmark Dataset

	Data file 1	Data file 2	Ref. mappings
Person1	2,000 URIs	1,000 URIs	500
Person2	2,400 URIs	800 URIs	400
Restaurants	339 URIs	2,256 URIs	112

Sampling. It is very difficult, if not impossible, to analyze billions of RDF triples to calculate accurate AVs and ACs for all properties. As an adaption, we propose a class-based sampling method for dealing with the scalability issue. For a class c, the objects whose types are explicitly defined as c are extracted. If the number of the objects is too small, c is no longer considered because it is insignificant in statistics; while if the number is too large, a subset of the objects are randomly chosen. To some extent, this method avoids the skewed sampling, i.e., guarantees the coverage so that a few common classes, such as `foaf:Person`, do not dominate the sample set. For our large-scale dataset in the experiments, we filter the classes with less than 300 objects, and for each class, we extract at most 1,000 objects to form the sample set. The entire sample set contains 1,199,764 objects, which is used for mining frequent property combinations. However, note that the discussion on different sampling techniques is out of the scope of this paper.

6. EVALUATION

We implemented a scalable system, called *ObjectCoref*, for our proposed method. In this section, we report the experimental results on a benchmark dateset (PR) in OAEI 2010 and on a large-scale dataset collected by Falcons until Sept. 2008. All the tests were carried out on an Intel Core2 Duo 2.4GHz CPU, 4GB memory with Windows 7 and Java 1.6. The datasets were run on four Xeon Quad 2.4GHz CPUs, 8GB memory with Redhat Linux Enterprize Server 5.4 and MySQL 5.0. The experimental results are downloadable at our website,[2] and a part of them about the benchmark test are cited from OAEI 2010 [5].

6.1 Benchmark Test

The PR dataset is a small real dataset, which includes two collections of RDF data files concerning persons (denoted by Person1 and Person2, respectively) and one collection about restaurants. OAEI 2010 organizers provided reference mappings for each collection, where each mapping contains two URIs from different data files that denote the same person or restaurant. The statistics of the PR dataset are listed in Table 1.

The goal of our evaluation on the dataset is twofold. First, we want to test various values for the parameters in ObjectCoref and apply the best ones to the following experiments. Second, we can compare ObjectCoref with other systems on the same dataset. The well-known *Precision*, *Recall* and *F-Measure* were used, where F-Measure is a linear combination of Precision and Recall:

$$\text{F-Measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (12)$$

Because all the URIs in the PR dataset are synthetic, no kernel can be established. Instead, we randomly chose 20 reference mappings from each collection and used them as the

[2]`http://ws.nju.edu.cn/objectcoref/www2011.zip`

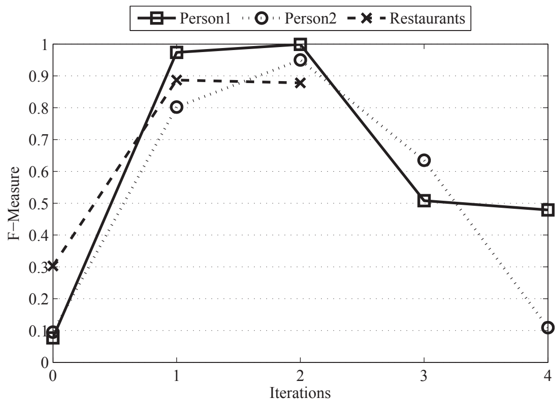

Figure 3: F-Measures versus number of iterations on the benchmark dataset

initial labeled set for training. To fit the dataset, we slightly modified the goal of our self-training algorithm by skipping the assignment of the most common values to properties, since each property only has one value for a URI. Moreover, all the URIs are described by a set of fixed properties, thus frequent property combinations cannot be mined.

Learning curves. The learning curves are shown in Fig. 3, where iteration 0 denotes the initial training set. Based on the figure, we observed that F-Measures drastically risen up in the first one or two iterations, and then dropped sharply. This demonstrates that one or two properties are accurate enough for identifying a person or a restaurant. For identifying the same person, `soc_sec_id` and `phone_number` were learnt; and for a restaurant, `phone_number` was discriminative. If we continued the training, improper properties were chosen, e.g., `age` for Person1 and Person2, which led to many wrong coreferent URIs.

From the learning, we observed that 0.95 is a good value for I-Sub to determine if two strings are similar enough. We also observed that 0.125 is a proper threshold for measuring a property-value pair is discriminative or not, which means that using such property-value pair, if the number of potentially coreferent URIs is eight times more than that in the labeled set, this pair would not be considered for resolution. We used these two thresholds throughout the following experiments.

F-Measure. We compared the results of ObjectCoref with other four coreference resolution systems, namely ASMOV, CODI, LN2R and RiMOM, which also submitted their results on the PR dataset to OAEI. ASMOV [16] and CODI [20] employed similarity-based matchers to obtain coreferent URIs and performed logical inference to remove inconsistent results. LN2R [21] integrated a knowledge-based matcher to find semantically coreferent URIs and adopted a similarity propagation algorithm to generate similarities. RiMOM [18] is a purely similarity-based system, which integrated many matchers to exploit a range of characteristics for both concepts and instances. All of them can only deal with pairwise instances, which are precisely called instance matching systems. We discuss their details in Section 7.

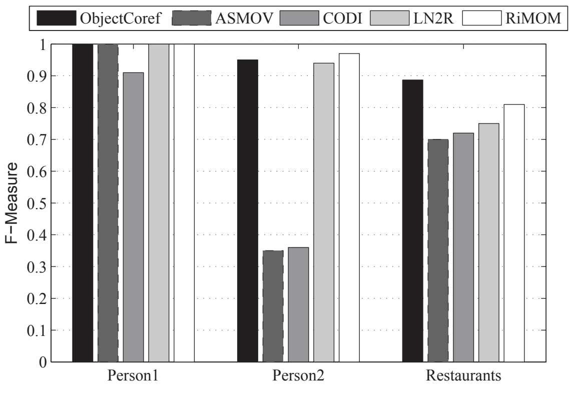

Figure 4: Comparison on F-Measure among ObjectCoref, ASMOV, CODI, LN2R and RiMOM on the benchmark dataset

The comparison results on F-Measure is depicted in Fig. 4. From the figure, we observed that ObjectCoref achieved the best F-Measure in average on the PR dataset. In particular, the Precision of ObjectCoref is quite good (100% in all the three collections), because we learnt the most discriminative properties for the resolution. Notice that our F-Measure on the restaurant track in this paper is slightly better than the result published by OAEI [5], as we corrected a parsing bug for addresses in our programs after participating in OAEI. Furthermore, we took advantage of a small number of reference mappings as the training set and the use of optimal iteration times, which are not against by the OAEI organizers. If other systems can make use of reference mappings, their performance may be improved as well.

6.2 Large-Scale Test

The statistical data of the large-scale dataset is listed in Table 2. The dataset contains 596,418,935 RDF triples (tr.) in 11,719,608 RDF documents referring to 76,389,570 URIs, which equals an average of 7.81 RDF triples per URI. This suggests that each URI is involved in about eight property-value pairs, so we set the maximum iteration times K to 10, which is a little larger than the average. Note that this value is dataset-dependent and no single value would yield similar results on multiple different datasets. We will study how to automatically determine the optimal iteration times in our future work.

By investigating the dataset, 7,880,906 `owl:sameAs` triples without blank nodes were found, where most of them come from `http://bio2rdf.org` and `http://dbpedia.org`. Considering the purposes of ObjectCoref, blank nodes cannot provide any meaning outside their original scopes. So we excluded the same-as (and IFP, FP, cardinality) relations having blank nodes in our experiments. This dataset contains 1,791 IFPs, in which 413 ones (23%) can be dereferenceable. 27,686 RDF triples using the 413 IFPs without blank nodes were retrieved, where most of them are about `foaf:mbox_sha1sum` (11,903) and `foaf:mbox` (2,981). It was found that 11,760 RDF triples hold the IFP relations with others. Regarding FPs, we obtained 11,765 dereferenceable FPs from 21,067 ones in total, and derived 35,652 RDF triples without blank nodes, in which 440 triples have the FP relations with others. Additionally, we discovered 34,635 triples without blank nodes that use 113 dereferenceable properties in cardinalities, where 6,123 ones have the cardinality relations with others.

Table 2: Statistical Data of Large-Scale Dataset

URIs	Same-as tr.	IFP tr.	FP tr.	Cardinality tr.
76,389,570	7,880,906	27,686	35,652	34,635

Table 3: Sample URIs

#	URI	
1	`http://www.w3.org/People/Berners-Lee/card#i`	
2	`http://www.cs.vu.nl/...#Frank+van+Harmelen`	
3	`http://data.semanticweb.org/person/chris-bizer`	
4	`http://dbpedia.org/resource/United_States`	
5	`http://dbpedia.org/resource/New_York_City`	
6	`http://dbpedia.org/resource/Berlin`	[city]
7	`http://dbpedia.org/resource/Semantic_Web`	
8	`http://bio2rdf.org/accession:af048837`	
9	`http://dbpedia.org/resource/Apple_Inc.`	[company]
10	`http://dbpedia.org/resource/Jaguar`	[mammal]

We collected 364,408 query logs from Falcons and chose 10 popular query URIs, which are listed in Table 3. These URIs cover a wide range of real world domains (e.g., people, geography), and all of them have semantically coreferent URIs for establishing kernels. The local names of some URIs have several meanings. For instance, "apple" can be either a fruit or a computer brand. We selected each URI having a clear meaning among its polysemic local names, to find whether or not ObjectCoref can identify the correct coreferent URIs and how well.

In this test, we evaluated Precision and *Relative recall* of ObjectCoref for resolving object coreference on the Semantic Web. Relative recall is the number of coreferent URIs retrieved by one system divided by the total number of unique coreferent ones from all systems, which offers a practical, if imperfect, solution to the problem that the total number of results is unknown. This measure has been widely applied to evaluate the quality of large ontology matching [6]. For Precision, we employed three students to take peer reviews on the coreferent URIs returned by each system. A student judged whether a returned URI is coreferent with the input, in terms of the provided evidences like the equivalence relations, the descriptions in the dereference documents.

Frequent property combinations. By applying the Apriori algorithm to perform association rule mining, we discovered 9,610 frequent property combinations with confidences greater than 0.98. Then, we used the two heuristic rules to refine the combinations and retained 349 ones, which were used as external knowledge for improving the accuracy of our resolution. Some sample frequent property combinations are listed in Table 4, where the average cardinalities are all 1.0,

Table 4: Frequent Property Combinations

Frequent property combinations	AVs	ACs
{`wgs84_pos#long`, `wgs84_pos#lat`}	{.93, .92}	{1.0, 1.0}
{`foaf:surname`, `foaf:givenname`}	{.83, .81}	{1.0, 1.0}
{`foaf:img`, `foaf:depictioin`}	{.52, .52}	{1.0, 1.0}
{`dbpedia:preceded`, `dbpedia:succeeded`}	{.48, .48}	{1.0, 1.0}
{`uniprot:height`, `uniprot:weight`}	{.38, .37}	{1.0, 1.0}

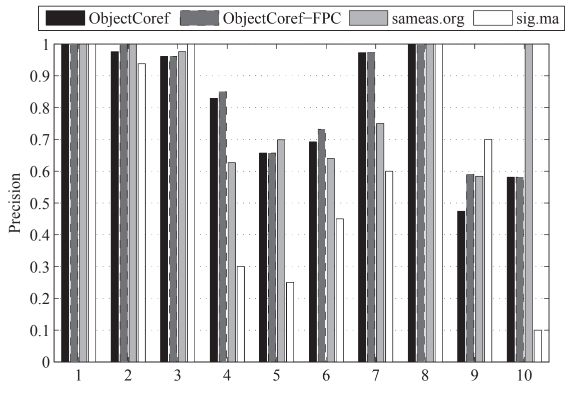

Figure 5: Comparison on Precision among ObjectCoref, ObjectCoref-FPC, sameas.org and sig.ma on the large-scale dataset

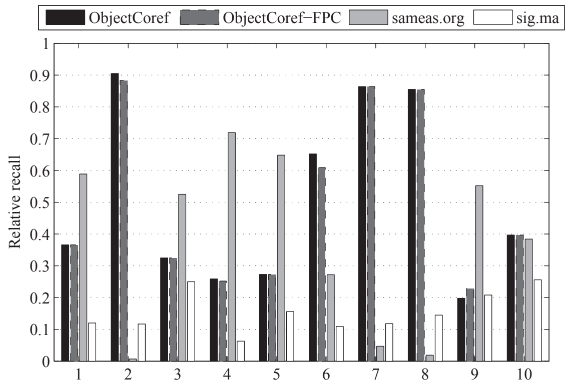

Figure 6: Comparison on Relative recall among ObjectCoref, ObjectCoref-FPC, sameas.org and sig.ma on the large-scale dataset

indicating that each property is used once in average for a single object. For instance, a location usually has only one longitude. On the contrary, the average number of values for each property is less than 1.0, which reflects that different URIs have same property-values.

Precision & Relative recall. In the experiment, we used three systems for comparison. ObjectCoref-FPC represents our system performing refinement based on frequent property combinations, while sameas.org[3] and sig.ma[4] [23] are two online services for object coreference resolution. sameas.org investigated a number of properties that explicitly define the equivalence relation, e.g., `owl:sameAs` and `skos:exactMatch`. sig.ma aimed at data mashup and can only accept keyword queries, so in this experiment we fed it the local names of our sample URIs.

[3]`http://sameas.org/`
[4]`http://sig.ma/`

Table 5: Summary of Average F-Measures

	Benchmark	Large-scale[5]
ObjectCoref	**0.95**	**0.62**
ASMOV	0.68	
CODI	0.66	
LN2R	0.90	
RiMOM	0.93	
sameas.org		0.54
sig.ma		0.21

The Precision and Relative recall of the four systems are depicted in Fig. 5 and Fig. 6, respectively. Based on the figures, we observed that ObjectCoref and sameas.org achieved similar Precisions in most cases, while their Relative recalls are quite different, because all the systems were run on different datasets, and the returned results only had a part of overlap. sig.ma did not perform well, because the keyword-based query brought ambiguity, and it limited the number of returned URIs per input no larger than 20. For the 10 queries, three most matchable properties were `rdfs:label`, `foaf:name` and `dc:title`. For example, (`rdfs:label`, "Tim Berners-Lee") and (`foaf:name`, "Tim Berners-Lee") were two discriminative property-value pairs in #1 for extension.

When frequent property combinations were integrated into the system, the Precisions in four cases were further improved, namely #2, #4, #6 and #9. Notice that the Relative recalls slightly decreased, due to the deletion of some wrong coreferent URIs. In #2, without frequent property combinations, ObjectCoref only used `foaf:firstName` and found wrong persons like "Frank Ohrtmann" or "Frank Leymann", but after considering frequent property combination {`foaf:firstName`, `foaf:lastName`}, such wrong coreferent URIs cannot be found. But in #10, (`rdfs:label`, "Jaguar") was selected as a discriminative property-value pair, which caused wrong resolution. Furthermore, we did not find any frequent property combination for distinguishing the mammal "Jaguar" with the car brand or a chemical package, so the Precision of ObjectCoref-FPC remained the same. The average number of coreferent URIs found by ObjectCoref-FPC is 68.4, while the average size of the kernels is 10.3.

Average resolution time. We randomly chose 5,000 sample URIs from our dataset, and repeated the experiment 10 times to measure the average resolution time for ObjectCoref. The total resolution time is about 12 hours, equating to 8.6 seconds per URI. But the time spent on a few URIs is much longer than this average. It took several minutes to complete an object URI with tens of potentially coreferent URIs. This indicates that, although ObjectCoref performs reasonably efficient when starting with a majority of URIs in our dataset, it is still hard to resolve all the URIs on the Semantic Web based on the current state of ObjectCoref.

Summary. The average F-Measures on the benchmark and large-scale datasets is illustrated in Table 5, depicting that ObjectCoref performed best in both datasets. As compared with the average performance of the other competitors, our approach achieved 18% and 25% increasing in F-Measure, respectively.

[5]The F-Measure on the large-scale dataset is the linear combination of Precision and Relative recall.

7. RELATED WORK

Object coreference resolution is an important task for establishing semantic interoperability and realizing data integration. In the area of Semantic Web, researchers addressed this problem mainly from two directions: one is based upon OWL semantics inference. Glaser, et al. [8] implemented a coreference resolution service (CRS) mainly by `owl:sameAs`. The works in [11, 23] performed large-scale object consolidation in terms of the analysis of IFPs, respectively. Saïs, et al. [21] designed a new language RDFS+ for expressing coreference, which integrated FPs, IFPs and `owl:disjointWith` in OWL as well as SWRL rules in RDFS. The KnoFuss architecture [19] combined the use of `owl:sameAs`, IFPs, FPs and `owl:differentFrom` for resolving coreference. Additionally, some works analyzed the state of `owl:sameAs` in the current Semantic Web or Linked Data [3, 10].

The other class of studies is based on the assumption that URIs are denoting the same real-world object if they share some common property-value pairs. Ferrara, et al. [7] reused an ontology matching tool HMatch to compare the minimal sets of assertions for describing different URIs. Hogan, et al. [13] proposed a statistical approach for identifying "quasi"-key properties for object consolidation over Linked Data. In addition, a number of works called instance matching (e.g., [14, 16, 18, 20, 26]) computed similarities between instances based upon matching their property-values. Except [13], the rest studies aforementioned assumed that the input is just pairwise instances and their computational costs are usually high, so it is difficult to adapt them to Web-scale.

For architecture, our proposed framework is similar to the work in [9], which dedicated a large-scale clustering to ontology terms. It used synonyms to set up a kernel, and then extended the kernel with similar terms in labels and identifiers. Both of us adopted a bootstrapping running mode, but [9] merely executed the extension one time. Furthermore, [9] performed extension only based on labels and local names, while our self-training framework is more adaptive for various domains. In addition, KnoFuss and LN2R [20] searched semantically coreferent URIs and proposed similarity propagation algorithms for refinement.

A key issue in our self-training algorithm is to measure the discriminability for property-value pairs. Hogan, et al. [13] analyzed the global distribution for properties and their associated values to count the discriminability for a property-value pair. Different from [13], our measurement is not static and further considered the matchability between properties, because a property in different domains may have different discriminability, while different properties could be used for expressing a similar meaning. In addition, we involved frequent property combinations to improve the accuracy of the resolution, where the analysis on the average cardinalities of properties was inspired by [13].

Besides the Semantic Web community, identifying duplicate entities, which is also under the names of record linkage, duplicate detection, coreference resolution and many others, has been extensively studied in both database and natural language processing areas [1, 4]. These works are in general treated as similarity-based due to a lack of formal semantics to define equivalence.

8. CONCLUDING REMARKS

The contributions of this paper are summarized as follows:

- We proposed a self-training approach for object coreference resolution on the Semantic Web, which first constructed a kernel of semantically coreferent URIs for a given object URI based on OWL semantics, and then iteratively extended this kernel in terms of discriminative property-value pairs. For further improving the accuracy of the resolution, frequent property combinations were mined and injected in the learning process. To the best of our knowledge, our proposed approach is the first attempt to adopt self-training for bridging the gap between semantically coreferent URIs and potential candidates, which is important for the maturing Web of Data.

- We introduced a statistical measurement to learn the discriminability of property-value pairs, which not only exploited key characteristics for representing an object, but also considered the matchability between properties from pragmatics.

- We applied association rule mining to discover frequent property combinations, which were further filtered by comparing the data ranges of properties in each combination and the distribution information on the use of properties as well as their values. The frequent property combinations avoided error accumulation during the training and improved the accuracy of the resolution.

- We evaluated our proposed approach on both a benchmark dataset in OAEI 2010 and a large-scale dataset that is collected by Falcons search engine in 2008. The experimental results showed that our approach achieved good F-Measure on the two datasets, as compared with the performance of six representative systems.

In the near future, we will perform more experiments for parameter setting in our self-training approach, such as determining the optimal iteration times. In the long term, we hope to abstract our object-driven approach to the class level, which will identify discriminative properties for different classes in different domains. Furthermore, we would like to design the co-training mechanism to improve the robustness of our coreference resolution algorithm. In addition, we will use our approach to analyze the coreference phenomenon in Linked Data.

9. ACKNOWLEDGMENTS

This work is supported in part by the NSFC under Grant 61003018 and 60973024, and also in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20100091120041. We would like to thank Hang Zhang and Min Liu for the active participation in the experiments. We also appreciate the helpful comments from three anonymous reviewers.

10. REFERENCES

[1] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.

[2] G. Cheng and Y. Qu. Searching linked objects with Falcons: Approach, implementation and evaluation. *International Journal of Semantic Web and Information Systems*, 5(3):49–70, 2009.

[3] L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. SameAs networks and beyond: Analyzing deployment status and implications of owl:sameAs in linked data. In *Proceedings of ISWC*, pages 145–160, 2010.

[4] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[5] J. Euzenat, A. Ferrara, C. Meilicke, A. Nikolov, J. Pane, F. Scharffe, P. Shvaiko, H. Stuckenschmidt, O. Svab-Zamazal, V. Svatek, and C. Trojahn. Results of the ontology alignment evaluation initiative 2010. In *Proceedings of ISWC Workshop on OM*, 2010.

[6] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[7] A. Ferrara, D. Lorusso, and S. Montanelli. Automatic identity recognition in the semantic web. In *Proceedings of ESWC Workshop on IRSW*, 2008.

[8] H. Glaser, A. Jaffri, and I. Millard. Managing co-reference on the semantic web. In *Proceedings of WWW Workshop on LDOW*, 2009.

[9] J. Gracia, M. d'Aquin, and E. Mena. Large scale integration of senses for the semantic web. In *Proceedings of WWW*, pages 611–620, 2009.

[10] P. Halpin and P. Hayes. When owl:sameAs isn't the same: An analysis of identity links on the semantic web. In *Proceedings of WWW Workshop on LDOW*, 2010.

[11] A. Hogan, A. Harth, and S. Decker. Performing object consolidation on the semantic web data graph. In *Proceedings of WWW Workshop on i3: Identity, Identifiers, Identification*, 2007.

[12] A. Hogan, A. Harth, and A. Polleres. Scalable authoritative OWL reasoning for the web. *International Journal on Semantic Web and Information Systems*, 5(2):2206–2249, 2009.

[13] A. Hogan, A. Polleres, J. Umbrich, and A. Zimmermann. Some entities are more equal than others: Statistical methods to consolidate linked data. In *Proceedings of ESWC Workshop on NeFoRS*, 2010.

[14] A. Issac, L. van der Meij, S. Schlobach, and S. Wang. An empirical study of instance-based ontology matching. In *Proceedings of ISWC/ASWC*, pages 253–266, 2007.

[15] I. Jacobs and N. Walsh. Architecture of the world wide web, volume one. W3C Recommendation, 2004.

[16] Y. Jean-Mary, E. Shironoshita, and M. Kabuka. Ontology matching with semantic verification. *Journal of Web Semantics*, 7(3):235–251, 2009.

[17] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In *Proceedings of SIGMOD*, pages 205–216, 2003.

[18] J. Li, J. Tang, Y. Li, and Q. Luo. RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1218–1232, 2009.

[19] A. Nikolov, V. Uren, E. Motta, and A. de Roeck. Refining instance coreferencing results using belief propagation. In *Proceedings of ASWC*, pages 405–419, 2008.

[20] J. Noessner, M. Niepert, C. Meilicke, and H. Stuckenschmidt. Leveraging terminological structure for object reconciliation. In *Proceedings of ESWC*, pages 334–348, 2010.

[21] F. Sais, N. Pernelle, and M. Rousset. L2R: A logical method for reference reconciliation. In *Proceedings of AAAI*, pages 329–334, 2007.

[22] G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proceedings of ISWC*, pages 623–637, 2005.

[23] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: Live views on the web of data. In *Proceedings of WWW*, pages 1301–1304, 2010.

[24] G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the open linked data. In *Proceedings of ISWC/ASWC*, pages 552–565, 2007.

[25] J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, and H. Bal. OWL reasoning with WebPIE: Calculating the closure of 100 billion triples. In *Proceedings of ESWC*, pages 213–227, 2010.

[26] S. Wang, G. Englebienne, and S. Schlobach. Learning concept mappings from instance similarity. In *Proceedings of ISWC*, pages 339–355, 2008.

[27] G. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.

[28] Z.-H. Zhou and M. Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2009.