Automatically generating data linkages using class-based discriminative properties

Wei Hu,⁎, Rui Yang, Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, China
Department of Computer Science and Technology, Nanjing University, China

Abstract

A challenge for Linked Data is to link instances from different data sources that denote the same real-world object. Millions of high-quality owl:sameAs linkages have been generated, but potential ones are still considerable. Traditional similarity-based methods to this data linkage problem do not scale well since they exhaustively compare every pair of instances. In this paper, we propose an automatic approach to data linkage generation for Linked Data. Specifically, a highly-accurate training set is automatically generated based on equivalence reasoning and common prefix blocking. The contexts of the instances in the training set, after extracting, are pairwise matched in order to learn discriminative property pairs supporting linkage discovery. For a particular class pair and a pay-level-domain pair, the discriminability of each property pair is measured, and a few property pairs with high discriminability are aggregated in order to be reused in the future to link instances between the same classes and domains. The experimental results show that our approach achieves good accuracy against some complex methods in two OAEI tests and the BTC2011 dataset.

1. Introduction

The Semantic Web (SW) is an effort by the W3C Semantic Web Activity, with the purpose of realizing data integration and sharing among different applications and parties. As of today, many prominent ontologies have been developed for data publishing in various domains, which suggest common classes and properties widely used across data sources.

At the instance level, however, there is a lack of agreement among sources on the use of common URIs to denote a real-world object. Due to the distributed nature of the SW, it frequently happens that multiple instances in diverse sources denote the same object, i.e., refer to an identical thing (also known as URI aliases [1] or coreferents). Such examples exist in the areas of personal profiles, academic publications, media or geographical data, etc.

Data linkage, also referred to as instance matching or object coreference resolution, aims at linking different instances for the same object. It is important to data-centric applications such as heterogeneous data integration or mining systems, SW search engines and browsers. Driven by the Linking Open Data (LOD) initiative, millions of instances have been linked with owl:sameAs explicitly [2], whose semantics defines that all the URIs linked with this property should identify the same resource. But compared to billions of URIs on the SW, there still exists a large amount of instances that potentially denote the same objects without being interlinked yet. For example, at least 70 instances crawled by the Falcons search engine [3] seem to denote Tim Berners-Lee, the director of W3C, but merely six have been linked with owl:sameAs. An analysis on the LOD cloud also indicates that, out of 31 billion RDF statements less than 500 million represent linkages between data sources, and most sources only link to one another.1

⁎ Corresponding author at: State Key Laboratory for Novel Software Technology, Nanjing University, China. Tel./fax: +86 25 8968 0923.
E-mail addresses: whu@nju.edu.cn (W. Hu), ryang@smai.nju.edu.cn (R. Yang), yzqu@nju.edu.cn (Y. Qu).
1 http://lod-cloud.net/state/ (09/19/2011).
In the SW community, traditional work addresses the data linkage problem mainly from two directions: one is by performing equivalence reasoning in terms of standard OWL semantics, e.g., through owl:sameAs and some “key” properties [4,5]; the other is by similarity computation, with the intuition that instances denote the same object if having similar property–values [6,7]. Recent work also uses machine learning and crowdsourcing to cope with complex data linkage tasks [8–11]. Generally speaking, the reasoning-based methods infer explicit linkages but may miss many potentials, while the similarity-based ones often suffer from high computational costs as they exhaustively compare all pairs of instances [12]; many of them have not been aware of the commonalities behind the abstract types of the instances and their publishers. For example, different data publishers prefer social security number, login name, address or even their combinations to disambiguate customers, but hobby or age is less likely to be used. It will facilitate data linkage in the future if such properties can be learnt and reused.

In this paper, we propose an automatic approach, called ADL, which differs from current similarity-based methods in learning a set of important properties for disambiguating instances (referred to as discriminative properties). The methodological steps of ADL, shown in Fig. 1, can be divided into the offline part and the online part:

- For the offline learning, a highly-accurate training set is automatically established. The training set consists of two sets of instance pairs holding the linkages or not, referred to as positive examples and negative examples, resp. The contexts (i.e., a kind of integrated units over RDF triples) for the instances in the training set are extracted in terms of RDF sentences [13], and pairwise matched with a lightweight linguistic matcher V-Doc [14], in order to discover discriminative property pairs, where a discriminative property pair consists of two matchable properties discriminative to link instances. For a specific class pair and a pay-level-domain pair, the discriminability of each property pair is measured by information gain, revealing the global and implicit preference of data publishers on characterizing a type of objects.

- For the online linking, given a new instance as input, the class that it belongs to and its pay-level-domain are firstly extracted, and then the counterpartnering classes and pay-level-domains in the training set are chosen. The instances, with the properties in the related discriminative property pairs, are found out, and their values are matched with that of the input using V-Doc. The similarities from different discriminative property pairs are linearly aggregated with equal weighting, in order to determine whether to generate an instance linkage.

We develop an open source tool and test its accuracy on three cases: the PR and NYT tests in the Ontology Alignment Evaluation Initiative (OAEI) as well as the Billion Triples Challenge (BTC2011) dataset. The experimental results show that, compared with several existing methods, our method achieves good precision and recall with the help of only a few discriminative property pairs. Moreover, the proposed approach is ready to be integrated with other methods, e.g., the found discriminative properties can be used for cost-effective candidate selection [12].

This paper is organized as follows. We define the data linkage problem in Section 2 and discuss related work in Section 3. In Sections 4 and 5, we present our approach to learn class-based discriminative property pairs. Evaluation is reported in Section 6. Finally, Section 7 concludes the paper with future work.

2. Problem statement

Let I be the set of URIs, B be the set of blank nodes and L be the set of literals. A triple <s, p, o> ∈ (I ∪ B) × I × (I ∪ B ∪ L) is called an RDF triple. An RDF graph G is a set of RDF triples, and can be serialized to an RDF document.

For an RDF graph G, a URI u is a class (resp. property) if G entails the RDF triple (u, rdf : type, rdfs : Class) (resp. ⟨u, rdf : type, rdf : Property⟩). If a URI u is not either a class, a property or both, then u is treated as an instance, implying the assumption that classes

\[\text{Offline}\]

\[\text{Online}\]

Fig. 1. Overview of the approach.

2 The pay-level-domain is a sub-domain of a public top-level-domain, for which users usually pay, e.g., the pay-level-domain for www.example.com is example.com. Pay-level-domains allow to identify a realm, where a data publisher is likely to be in control [2].
and properties are disjoint with instances. In this paper, it is assumed that there exists an “oracle” to distinguish them and resolve the definition inconsistency problem.

**Blank nodes** are a type of existentially quantified resources whose meanings exist in the scope of the RDF triples in which they reside. RDF triples that share a blank node constitute an integrated structure, forming a joint semantics of that blank node. If such triples were separated, the semantics would be broken. But current RDF semantics provides no intrinsic mechanism to preserve this kind of structures. In this paper, we use RDF sentences rather than RDF triples as basic units for describing instances. Please refer to [15] for the formal properties of RDF sentence (which was also called the minimum self-contained graph). To formalize, two RDF triples are defined as b-connected if sharing a blank node. An RDF sentence is the maximum closure of the b-connected RDF triples.

**Definition 1. RDF sentence**

Let $G$ be an RDF graph. An RDF sentence, denoted by $st$, is a set of RDF triples, which satisfies the following conditions: (1) $st \subseteq G$; (2) $\forall t_i, t_j \in st, i \neq j, t_i, t_j$ are b-connected; and (3) $\forall t_i \in st, t_j \notin st, t_i, t_j$ are not b-connected.

For an RDF sentence $st \subseteq G$, four operations on $st$ are defined to obtain its subjects, properties and values:

$$\text{Subj}(st) = \{s \in \exists \exists \exists (s_k, p, o) \in \text{st}\},$$

$$\text{Prop}(st, s_k) = \{p \in \exists \exists \exists (s_k, p, o) \in \text{st}\},$$

$$\text{Value}(st, s_k, p_j) = \{o \in \exists \exists \exists (s_k, p_j, o) \in \text{st}\} \cup \bigcup_{o' \in \exists \exists \exists (s_k, p_j, o) \in \text{st}} \text{Value}'(st, o'),$$

$$\text{Value}'(st, s_k) = \{o \in \exists \exists \exists (s_k, p, o) \in \text{st}\} \cup \bigcup_{o' \in \exists \exists \exists (s_k, p, o) \in \text{st}} \text{Value}'(st, o').$$

For an instance, we define its context in terms of the Concise Bounded Description, where the instance appears in the subject of RDF sentences.

**Definition 2. Context**

Let $u$ be an instance. The context of $u$, denoted by $\text{Ctx}(u)$, is a set of RDF sentences $\text{Ctx}(u) = \{st_1, st_2, ..., st_n\}$, where each $st_i (i = 1, 2, ..., n)$ satisfies that $u \in \text{Subj}(st_i)$.

With regard to a given instance for extracting its context, only forward links ($u \in \text{Subj}(st_i)$) are considered. But the direction of a property in RDF is somehow arbitrary, and backward links may also be useful. Although it is not hard to deal with backward links, only forward links are included since predicates and objects in RDF triples mainly describe subjects. The number of triples in a context can be reduced as well.

Besides owl:sameAs, the inverse functional property (IFP), e.g., foaf: homepage, is a type of properties with high discriminability often used to infer instance equivalence. The semantics of an IFP dictates that separate instances are indirectly inferred to be equivalent in terms of having the same value of that property. To discover IFPs, we parse ontologies and find the properties whose rdf:type is explicitly defined as owl:IFP. IFPs can be inferred in a multitude of ways in OWL semantics. For example, Urbani et al. [16] conducted reasoning on pD* that includes rules for handling owl:sameAs, owl:IFP axioms, etc. But anyone can say anything on the SW; inferring IFPs across data sources may cause errors and inconsistency. For example, we even observed dc:title being an IFP in the Falcons dataset. Hogan et al. [7] studied the reasoning problem of new ontologies published on the SW that redefine the semantics of existing classes and properties in other ontologies (called ontology hijacking), suggesting to use dereferenceable URIs to avoid this problem.

A functional property (FP) is a property that can have only one (unique) value for each instance, which can be used to infer the equivalence between instances. Similarly, only dereferenceable FPs are used here. Also, 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 with an owl:cardinality (resp. owl:maxCardinality) constraint describes a class of all instances having exactly (resp. 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, but localized to a particular class.

---

3 In implementation, ontology parsers often assign system-generated internal names to blank nodes. An RDF graph is transformed into an undirected triple graph, where each vertex denotes an RDF triple in the original graph. An edge exists between two triples if they contain blank nodes with the same name. We call that the two RDF triples share a blank node. Each connected component in the triple graph forms an RDF sentence.

4 http://www.w3.org/Submission/CBD/.

5 The URI dereference is a resource retrieval mechanism that uses any of the Internet protocols (e.g., HTTP) to obtain a representation of the resource it identifies, such as an RDF document. The representation retrieved by dereferencing a URI can be considered as the authoritative definition of that URI [7].
The goal of data linkage is to identify different instances referring to the same real-world object. There are a lot of works addressing the data linkage problem, which leads to different viewpoints and definitions. In this paper, it is assumed that the input of data linkage is just one set of instances.

Definition 3. Data linkage

Let \( U = \{u_1, u_2, \ldots, u_n\} \) be a finite set of instances. The data linkage on \( U \) is a function: \( \text{Link}: U \times U \rightarrow [0, 1] \), which assigns a confidence measure in range \([0, 1]\) to weigh any \((u_i, u_j) \in U \times U\).

Given a pre-defined threshold \( \epsilon \in [0, 1] \), a linkage between \( u_i, u_j \) will be generated if \( \text{Link}(u_i, u_j) \geq \epsilon \). The set of all pairs of instances for which the linkages hold is represented by \( D^+ = \{(u_i, u_j) \in U \times U | \text{Link}(u_i, u_j) \geq \epsilon\} \). Likewise, we denote the set of all pairs for which the linkages do not hold: \( D^- = (U \times U) \setminus D^+ \). Typically, the size of \( D^+ \) approximates \( O(|U|^2) \) \([17]\). In some cases, a subset of \( D^+ \) or \( D^- \) is already known prior to linking, which can be used as training data \([9]\).

Example 1. To better understand, we illustrate a running example in Fig. 2. Let us consider the instance \( \text{geo:1816670} \). Several instances can be directly linked, e.g., \( \text{sw:Beijing} \), which are inferred by \( \text{owl:sameAs} \). Also, \( \text{sw:Beijing} \) and \( \text{dbp:Peking} \) can be inferred as equivalent, as a conference event is restricted to have at most one location w.r.t. the ontology in the right part of the figure. (Note that this ontology is constituted by an RDF sentence, where \( \_\text{anonid} \) represents a blank node.) Therefore, \( \text{sw:Beijing} \) and \( \text{dbp:Peking} \) are added in positive examples. Also, \( \text{dbp:Perth} \) is added in negative examples w.r.t. the positives (see Section 4.2 for more details).

Then, we extract the contexts for the instances in the training set. For example, the context for \( \text{geo:1816670} \) is constituted by a set of RDF triples with \( \text{geo:1816670} \) as subject. We pairwise match these contexts to measure the discriminability of all matchable property pairs, e.g., for \( \text{geo:Feature in Geonames vs. dbp:Place in DBpedia, (geo:long, dbp:longitude) and (geo:lat, dbp:latitude)} \) are identified as two discriminative property pairs. We can also find that \( \text{dbp:utcOffset} \) is non-discriminative for \( \text{dbp:Place in DBpedia}. \) The above is done offline.

For the online linking, given an instance of \( \text{geo:Feature in Geonames, e.g., geo:5128581}, \) the values of \( \text{(geo:lat, dbp:latitude) and (geo:long, dbp:longitude)} \) are matched with those of the instances from \( \text{dbp:Place in DBpedia}, \) and the similarities are linearly aggregated to link new instances, e.g., \( \text{dbp:New_York_City} \).

3. Related work

In the SW field, researchers addressed the data linkage problem mainly from two directions: one is by equivalence reasoning. Glaser et al. \([4]\) implemented a Co-reference Resolution Service (CRS) mainly by \( \text{owl:sameAs} \). Hogan et al. \([5,7]\) conducted large-scale object consolidation in terms of the analysis on IFPs. Saïs et al. \([18]\) designed a language RDFS+ for reference reconciliation, which combined FPs, IFPs and \( \text{owl:disjointWith} \) in OWL as well as SWRL rules. The KnoFuss system \([19]\) used \( \text{owl:sameAs, IFPs, FPs and owl:differentFrom} \) to identify coreference. Also, the works in \([2,20]\) investigated the status of \( \text{owl:sameAs} \) in the Linked Data. In general, the reasoning-based approaches are usually accurate but likely to miss many potential data linkages for the current SW.

The other line of work is based on the assumption that instances refer to the same real-world object if sharing similar property–values. Ferrara et al. \([6]\) reused an ontology matching tool HMatch to compare the minimal sets of assertions describing different instances. RiMOM \([21]\) and AgreementMaker \([22]\) also extended with similarity-based matchers for instance matching.

---

**Fig. 2. A running example.**
In general, traditional similarity-based methods assume that the input is two datasets and exhaustively compare every pair of instances in them [12], which often suffer from high computational costs. Our work uses equivalence reasoning to build a training set and similarity computation to learn discriminative property pairs from the training set.

Regarding discriminative properties, Zhishi.links [23] employed an indexing technique to pick up candidate instance pairs, and its indexing scheme was based on manually defined properties, which only used six properties for describing names/aliases for all domains. The works in [7,12,24] proposed to select domain-independent candidates by important properties. They analyzed the global distributions for properties and their associated values to measure the discriminability and coverage of each property–value pair in a dataset. However, they ignored the existing data linkages and did not consider the pragmatics of properties w.r.t. different classes as well as property matching. ObjectCoref [25] applied self-training to learn "local" discriminative property–values for improving coreferencing on a specific object, but it did not characterize the discriminability from a more abstract level. Wang et al. [26] focused on using Markov Random Field to learn the weighting of properties from the TF-IDF based similarities between the instances of two library thesaurus. But their goal was to match thesaurus rather than to link instances, and they assumed that the properties from the two thesauri are the same to simplify the learning process using the same feature space.

In recent years, crowdsourcing attracted many attentions in data linkage. Yang et al. [11] combined the distributed human computation system with Linked Data to create an ecosystem to solve the coreference problem when integrating distributed bibliographic datasets. ZenCrowd [8] employed human workers to improve the quality of identifying entities from natural language text, and introduced a probabilistic framework to make decisions about candidate linkages and to recognize unreliable human workers. Besides, CrowdMap [27] studied micro-task crowdsourcing for ontology alignment. In general, the crowdsourcing-based methods can largely improve the accuracy of linkages with the help of collective intelligence. Additionally, active learning is often introduced to make the best use of human efforts. SILK [28] enabled the derivation of owl:sameAs relations among datasets with customized properties and mapping criteria. It combined active learning and genetic programming to generate link specifications with discriminative properties [9], which required a set of instance linkages beforehand and the computational cost is high. EAGLE [10] is also conducted in a similar way. Different from them, user interaction is not involved in our approach.

Outside the SW community, identifying duplicate entities, also known as record linkage, duplicate detection, entity resolution and many others, has been extensively studied in database [29,30]. Some blocking methods were presented, e.g., BSL [31] and Adaptive Filtering [32], using supervised or unsupervised learning to identify key attributes and reduce the searching space of candidate selection. But these works may not be directly used in the SW, since they did not conform to the RDF data model (e.g., blank nodes) and cannot utilize OWL semantics for reasoning. None of them considered to characterize the blocking rules at the class and domain levels.

Compared with all these works above, our proposed approach is fully-automatic and learns class-based discriminative properties for data linkage by leveraging equivalence reasoning and similarity computation.

4. Training set generation

Due to the large scale of the SW, it is time-consuming to manually build a training set with both broad coverage and good accuracy. Thanks to the LOD initiative, millions of instances have been interlinked with owl:sameAs explicitly. This enlightens us to utilize these equivalence relations to automatically generate a highly-accurate, moderate-scale training set.

4.1. Positive example reasoning

In the first step, the instances holding the linkages (i.e., positive examples in the training set) are exploited by performing equivalence reasoning.

Let U be a set of instances. The same-as relation, denoted by S, is defined as the minimal reflexive, symmetric relation on U, which satisfies that: (1) ∀s∈U, ⟨s,s⟩∈S; (2) ∀s,o ∈ U, if there is an RDF triple ⟨s, owl:sameAs, o⟩, then ⟨s,o⟩∈S and ⟨o,s⟩∈S. The RDF triple ⟨s, owl:sameAs, o⟩ is called a same-as triple.

SKOS is a common data model for sharing and linking knowledge organization via the Web. skos:exactMatch is recommended to link instances, e.g., (zbw:16830–9, skos:exactMatch, dbp:Berlin) gives a high degree of confidence that the two instances can be used interchangeably in a wide range of information retrieval applications. skos:exactMatch is reflexive and symmetric. The RDF triple involving skos:exactMatch is called an exact-match triple, and the exact-match relation is denoted by E.

Given an IFP, two separate instances can be indirectly inferred as the same based on holding the same value of that IFP. To formalize, let U be a set of instances. The IFP relation, denoted by I, is defined as the minimal reflexive, symmetric relation on U, which satisfies that: (1) ∀s ∈ U, ⟨s,s⟩ ∈ I; (2) ∀s1,o ∈ U, if there exist an IFP p and two RDF triples ⟨s1, p, o⟩, ⟨s2, p, o⟩, then ⟨s1,s2⟩ ∈ I and ⟨s2,s1⟩ ∈ I.

To establish the IFP relations, we compare the object values in the RDF triples with the same IFP as predicate. If their object values are completely the same, an IFP relation between the subjects of those IFP triples is constructed. The lexical forms of some literals could be empty, where for example the values of foaf:mbox_sha1sum are blank in a few triples. We omit these triples to avoid wrong linkages. Due to the large number of email addresses and several heterogeneous ways of expression, an ad hoc method is designed to identify identical email addresses between foaf:mbox_sha1sum and foaf:mbox. For an email address,
we calculate its sha1 sum value and use foaf:mbox_sha1sum to find new linkages. This is the only property for which a specific comparison method has been made.

The way of using FPs to link instances is similar to that for IFPs. The RDF triple involving an FP is called as an FP triple, and the FP relation is denoted by $\mathcal{F}$.

Additionally, regarding the cardinality constraint $\text{owl:cardinality}$ (or $\text{owl:maxCardinality}$), if the value of the cardinality equals 1, it is useful to equivalence reasoning. $\mathcal{C}$ is used to denote this specific cardinality relation.

In terms of the same-as, exact-match, IFP, FP and cardinality relations, we define the equivalence relation as follows.

Definition 4. Equivalence relation

Let $\mathcal{S}$, $\mathcal{E}$, $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{C}$ be the same-as, exact-match, IFP, FP and cardinality relations on a set of instances $\mathcal{U}$, resp. The equivalence relation, denoted by $\mathcal{K}$, is defined as the transitive closure on $\mathcal{S} \cup \mathcal{E} \cup \mathcal{I} \cup \mathcal{F} \cup \mathcal{C}$, that is, $\mathcal{K} = (\mathcal{S} \cup \mathcal{E} \cup \mathcal{I} \cup \mathcal{F} \cup \mathcal{C})^+$. Clearly, $\mathcal{K}$ is the equivalence relation on $\mathcal{U}$, because $\mathcal{S}$, $\mathcal{E}$, $\mathcal{I}$, $\mathcal{F}$, and $\mathcal{C}$ are all reflexive and symmetric. Within this paper, the equivalence relation will be represented using the symbol $\sim$. The reasons to choose the five built-in vocabulary elements are that they are significant in size and widely used to infer the equivalence relation in literature (see Sections 6.2 and 3). With more and more vocabularies being published on the SW, we will incorporate new vocabulary elements in our future work.

In terms of the equivalence relation $\sim_{\mathcal{K}}$, the positive examples in the training set are defined as follows.

Definition 5. Positive examples

Let $\mathcal{U}$ be a set of instances. For an instance $u \in \mathcal{U}$, the positive examples of $u$, denoted by $\mathcal{D}^+(u)$, are a set of instance pairs for which the equivalence relations hold, that is, $\mathcal{D}^+(u) = \{(u, v) | \forall u \sim_{\mathcal{K}} v\}$.

A simple method to compute the transitive closures on $\mathcal{U}$ is the Floyd–Warshall algorithm with the time complexity $O(|\mathcal{U}|^3)$. An instance is typically linked with only a few others by equivalence reasoning. For example, we randomly chose 100,000 instances from the BTC2011 dataset and found that an instance links to about 1.5 instances in average. Thus, the practical computational cost for generating positive examples is low.

4.2. Negative example estimation

In the second step, the instance pairs that do not hold the linkages (i.e., negative examples) are constructed for the training set. According to our analysis, there are only a few instances that are claimed to be semantically different in the BTC2011 or Falcons dataset using $\text{owl:differentFrom}$ or $\text{owl:AllDifferent}$ (see Section 6.2). This rare number is not adequate for a moderate-sized training set that covers various classes and domains.

Instead, we approximate the negative examples in terms of the pre-found positives. For any two instances, we consider them probably a negative example if without an equivalence relation. This assumption is based on the number of negative examples which is significantly more than positives [9].

Definition 6. Negative examples

Let $\mathcal{U}$ be a set of instances. For an instance $u \in \mathcal{U}$, the negative examples of $u$, denoted by $\mathcal{D}^-(u)$, are a set of instance pairs for which the equivalence relations do not hold, that is, $\mathcal{D}^-(u) = \{(u, v) | \forall u \in \mathcal{U}, (u, v) \not\in \mathcal{D}^+(u)\}$.

This approximation involves wrong negative examples to some extent, because positive examples are inferred using equivalence reasoning, which may produce false negatives. But considering the significant difference between the sizes of positive and negative examples, the wrong negatives are usually rare. For instance, we manually checked 2000 negative examples in our experiment, and only two wrong ones were observed. Furthermore, the approach proposed in the paper statistically measures for a type of instances, therefore a small number of wrong negative examples do not influence the global discriminability.

In case that too many instance pairs can be negative examples, and most of them are not useful for learning discriminative properties since they are totally orthogonal, we use two heuristic rules to eliminate definitively negative examples. The goal of the elimination is to achieve the balance between the positive and negative examples, more precisely, to keep the size of $\mathcal{D}^-$ and the size of $\mathcal{D}^+$ at the same order of magnitude.

1. Negative examples are chosen from the same classes and namespaces as those of positives, because it is likely for a dataset to use different URIs to distinguish different instances under a particular class (without explicitly specifying that they are semantically equivalent).
2. If the number of instances under a particular class and namespace is still considerable, e.g., foaf:Person, a common prefix blocking scheme is further used, which only keeps the negative examples holding that the local names\footnote{The local name of a URI is a string after the last hash “#” or slash “/” of the URI.} of the involved
instance URIs are different with those of positives, but have at least the first-`
`l common letters (implying that the local 

names of these URIs must have more than l letters). We can increase the length l to filter more instances in negative examples. In case that some local name is just a sequence of random characters, we may use rdfs:label instead. Other feasible filtering schemes include first letter scheme or common word scheme [33]. The comparison on different filtering schemes is out of the scope of this paper.

**Example 2.** Recalling the example in Section 2, it is found that some instances are holding the linkages with geo:1816670 by equivalence reasoning, e.g., dbp:Peking; they form positive examples. To approximate negative examples, we consider other places in DBpedia and only keep instance URIs whose local names are different with “Peking” but have at least l-length common prefix. Assuming l = 2 in this case, the instances like dbp:Perth would be reserved to form negative examples with geo:1816670.

5. Discriminative property discovery

Discriminative property pairs are learnt by comparing the contexts of the instances in the training set. In accordance with many similarity-based approaches, e.g., [7,20,26], it is assumed that the instances that constitute linkages share similar property-values, and a few properties are more important for characterizing the denoted real-world object.

5.1. Discriminative property pair learning

For an instance to be compared, we extract its involved context from the dereferenced document of its URI. Different from the works in [12,23], we believe that the values, from both data type properties and object properties, are useful for linking instances. We use RDF sentences (see Section 2) to guarantee the completeness of the values of object properties when encountering blank nodes.

**Example 3.** Let us see the example in Fig. 3. w3:tbl is the subject of an RDF sentence (not allowed to be blank according to Eq. (1)). It has an object property foaf:based_near, and the values of foaf:based_near are “42.361860” and “—71.091840”. By traversing the blank node, we extract the complete context for w3:tbl.

We compare the contexts of the instances in the training set with a linguistic matcher called V-Doc [14], whose novelty lies in the construction of virtual documents. A virtual document characterizes a set of values V for a property p w.r.t. the context of an instance u. Let v ∈ V,

- If v is a URI, a term vector (precisely, a bag of terms normalized by stemming, removing stop words, etc.) is extracted from v’s local name;
- If v is a literal, a term vector representing its lexical form is extracted;
- If v is a blank node, a term vector is formed as the union (but allowing components repeated) of the term vectors from v’s forward values in the same RDF sentence in the context of u (see Eqs. (3)–(4)).

The virtual document of p w.r.t. u, denoted by V D(p|u), is the union of all its term vectors in V, because p can have multiple values in the context of u. Moreover, V D(p|u) is refined by inverse document frequency (IDF) factors. Hence, a virtual document is represented as a collection of weighted terms based on the TF-IDF model, and the weights are rational numbers.

Given any two properties p_i, p_j w.r.t. two instances u_i, u_j, the similarity between p_i and p_j is computed based on the cosine similarity between their virtual documents (the so-called extensional measure):

\[
\text{sim}(p_i, p_j|u_i, u_j) = \frac{V \ D(p_i|u_i) \cdot V \ D(p_j|u_j)}{|V \ D(p_i|u_i)| \cdot |V \ D(p_j|u_j)|},
\]

where the numerator is the dot product of the two virtual documents, and the denominator is the product of their magnitudes. The time complexity of comparing all pairs of instances in the training set is O(|D|), which is the main time cost in learning.
discriminative property pairs. However, the size of training set is much smaller than the Cartesian product of the number of all instances, so the real time spent on instance comparison is not too long.

A property can be associated with a variety of classes, and for different classes, the discriminability of the same property can be different. Given a property \( p \), we partition the instances involving \( p \) by their classes. Additionally, pay-level-domains are frequently used to identify Linked Data publishers [2], and different publishers have their own preferences on the use of properties, so the instances involving \( p \) under a class \( c \) are further partitioned in terms of their pay-level-domains. More fine-grained methods to identify domains can be seen in [34]. On the other hand, a few properties and their values repeat in many instances, e.g., \((\text{geo:1816670, dbp:Peking}) (\text{geo:long, dbp:longitude})\) \(0.8\), \((\text{geo:lat, dbp:latitude})\) \(0.5\), \((\text{geo:utcOffset, dbp:utcOffset})\) \(1.0\).

In case that there are not enough training examples related to a \((\text{class, pay-level-domain})\) pair affecting the learning accuracy, \(\theta=0.01\) is fixed in the paper. In fact, a few discriminative pairs are enough to link instances, so \(\theta\) acts as a baseline. We rank all the property pairs with their discriminability under a class pair and a pay-level-domain pair, and select the top-\(k\) highest discriminative pairs to link instances online. The value of \(k\) will be determined in the experiment (see Section 6.1).

In case that there are not enough training examples related to a \((\text{class, pay-level-domain})\) pair affecting the learning accuracy, class hierarchies are used to flexibly adjust the classes of instances. Initially, we begin with the class to be measured, and if the instances for this class are too few, we choose its superclass for gathering more instances. The process terminates when the sizes

<table>
<thead>
<tr>
<th>Table 1</th>
<th>An example for measuring the discriminability of property pairs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance pair</td>
<td>Property pair</td>
</tr>
<tr>
<td>(geo:1816670, dbp:Peking)</td>
<td>(geo:long, dbp:longitude)</td>
</tr>
<tr>
<td></td>
<td>(geo:lat, dbp:latitude)</td>
</tr>
<tr>
<td></td>
<td>(geo:utcOffset, dbp:utcOffset)</td>
</tr>
<tr>
<td>(geo:1816670, dbp:Perth)</td>
<td>(geo:long, dbp:longitude)</td>
</tr>
<tr>
<td></td>
<td>(geo:lat, dbp:latitude)</td>
</tr>
<tr>
<td></td>
<td>(geo:utcOffset, dbp:utcOffset)</td>
</tr>
</tbody>
</table>
of positive examples and negative examples w.r.t. a class are both more than a pre-defined number, e.g., 20 in this paper, which guarantees a relatively good diversity of training examples. The class hierarchies can be queried from an external “oracle” such as Falcons [3].

Example 4. Let us see the example in Table 1. We can compute the entropy of $D$ by $H(D) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1.0$. Assuming the similarity threshold $\eta = 0.8$, we can measure the entropy of the property pair $(\text{geo:long}, \text{dbp:longitude})$ by $H(\text{geo:long}, \text{dbp:longitude}) = \frac{1}{2} \log_2 0 + \frac{1}{2} \log_2 0 = 0$. Therefore, the discriminability (i.e., information gain) for $(\text{geo:long}, \text{dbp:longitude})$ is 0, which is non-discriminative to link instances. Additionally, the discriminability of $(\text{geo:lat}, \text{dbp:latitude})$ is dependent on the choice of $\eta$.

Considering $\text{geo:Feature}$ in Geonames vs. $\text{dbp:Settlement}$ in DBpedia, if the instances of $\text{dbp:Settlement}$ in DBpedia in the training set are not enough, then we will query its super-classes and use the super-classes, e.g., $\text{dbp:Place}$, to learn discriminative property pairs.

5.2. Online linking

For an instance to be linked online, its class defined in the dereferenced document, and its pay-level-domain are extracted. If the instance has more than one class, we enumerate all of them during the online process. Then, the counterparting (class, pay-level-domain) pairs learnt offline are selected, and the related top-k discriminative property pairs are queried. The value from each instance involving a discriminative property pair is matched with that of the input w.r.t. each class pair and pay-level-domain pair. At the end, the similarities of different discriminative property pairs are linearly aggregated with equal weighting, in order to decide whether to generate an instance linkage or not. Although some more complex data linkage learning models are proposed, e.g., logistic regression [35] and genetic programming [9], we find that, with the help of discriminative property pairs, some cost-effective aggregation strategies can also achieve good accuracy.

Let $u_i$ be an instance to be linked in a class and pay-level-domain pair $(c_i, d_i)$, and $u_j$ be a candidate in another class and pay-level-domain pair $(c_j, d_j)$. $P$ denotes the top-k discriminative property pairs w.r.t. $(c_i, d_i)$ vs. $(c_j, d_j)$. The confidence measure for linking $(u_i, u_j)$ is defined as follows:

$$\text{Link}(u_i, u_j) = \frac{1}{k} \sum_{(p_i, p_j) \in P} \text{sim}(p_i, p_j | u_i, u_j).$$

Table 2
Statistical data of the OAEI tests.

<table>
<thead>
<tr>
<th>PR</th>
<th>Persons1</th>
<th>Persons2</th>
<th>Restaurants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instances in RDF file 1</td>
<td>2000</td>
<td>2400</td>
<td>339</td>
</tr>
<tr>
<td>Instances in RDF file 2</td>
<td>1000</td>
<td>800</td>
<td>2256</td>
</tr>
<tr>
<td>Reference linkages</td>
<td>500</td>
<td>400</td>
<td>112</td>
</tr>
<tr>
<td>NYT</td>
<td>Locations</td>
<td>Organizations</td>
<td>People</td>
</tr>
<tr>
<td>Instances in NYT</td>
<td>3840</td>
<td>6088</td>
<td>9958</td>
</tr>
<tr>
<td>Instances in DBpedia</td>
<td>$\approx 0.46$ M</td>
<td>$\approx 0.15$ M</td>
<td>$\approx 0.36$ M</td>
</tr>
<tr>
<td>Reference linkages: NYT–DBpedia</td>
<td>1920</td>
<td>1949</td>
<td>4977</td>
</tr>
<tr>
<td>Instances in Freebase</td>
<td>1920</td>
<td>3044</td>
<td>4979</td>
</tr>
<tr>
<td>Reference linkages: NYT–Freebase</td>
<td>1920</td>
<td>3044</td>
<td>4979</td>
</tr>
<tr>
<td>Instances in Geonames</td>
<td>$\approx 10$ M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference linkages: NYT–Geonames</td>
<td>1789</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Table 3
No. of negative examples and generation time for the OAEI tests.

<table>
<thead>
<tr>
<th>PR</th>
<th>Persons1</th>
<th>Persons2</th>
<th>Restaurants</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locations</td>
<td>Organizations</td>
<td>People</td>
<td>Total time</td>
<td></td>
</tr>
<tr>
<td>NYT–DBpedia</td>
<td>2752</td>
<td>2730</td>
<td>1803</td>
<td>112 s</td>
</tr>
<tr>
<td>NYT–Freebase</td>
<td>1986</td>
<td>3937</td>
<td>9846</td>
<td>63 s</td>
</tr>
<tr>
<td>NYT–Geonames</td>
<td>3688</td>
<td>17 s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A threshold $\epsilon \in [0, 1]$ is given to eliminate the instances having little confidence values with $u_i$, i.e., an instance linkage is formed between $u_i, u_j$ iff $\text{Link}(u_i, u_j) \geq \epsilon$. Without loss of generality, we set $\epsilon = 0.5$ in this paper. Usually, an instance only links with a small amount of others, so the time complexity for querying candidates and comparing them is low. In our experiment, the run time in general is less than 1 min for an instance, because it only needs to look up the stored discriminative property pairs, and find a few others with similar property-values.

Two special cases are considered: (1) if $u_i$ has no class, or $(c_i, d_i)$ (and the super-classes of $c_i$) has not been included in the training set, we use \text{rdfs:label} as the only discriminative property and search the instances satisfying that their \text{rdfs:label} values match the one of $u_i$ using V-Doc; and (2) if $(c_i, d_i)$ (or the super-classes of $c_i$) is included in the training set, to extend the searching space not covered by the training set, we also use the top-k frequently-occurred discriminative properties derived from the discriminative property pairs in the training set w.r.t. $(c_i, d_i)$. Typically, the derived discriminative properties are the well-known properties, e.g., \text{rdfs:label}, \text{foaf:name} and \text{dc:title}. We searched the instances satisfying that their values of the discriminative properties match those of $u_i$ using V-Doc. This process is similar to some indexing-based methods without property matching [23,24].

**Example 5.** For the instance pairs in Table 1, assuming that $(\text{geo:long}, \text{dbp:longitude})$ and $(\text{geo:lat}, \text{dbp:latitude})$ are two discriminative property pairs w.r.t. \text{geo:Feature} in Geonames vs. \text{dbp:Place} in DBpedia, the linkage confidence of $(\text{geo:1816670}, \text{dbp:Peking})$ is $\frac{1}{2} \times (1.0 + 0.8) = 0.9$. To link $\text{geo:1816670}$ with the instances whose class and pay-level-domain are not covered by the training set, we find the instances that involve $\text{geo:long}$ and $\text{geo:lat}$ and have similar values to that of $\text{geo:1816670}$.

6. Evaluation

We developed an open source tool for the proposed method ADL. In this section, we will report our experimental results on two OAEI tests (namely, PR and NYT) as well as the BTC2011 dataset. All the tests were conducted on a PC with an Intel Core 2 Duo
2.4 GHz CPU, 4 GB memory, Windows 7 and Java 6. The datasets were stored on an IBM x3850 M2 server with two Xeon Quad 2.4 GHz CPUs, 8 GB memory, Red Hat Enterprise Linux Server 5 and MySQL 5.1. The detailed results and source code are available at our website,\(^7\) and a part of the results on the PR and NYT tests was cited from the OAEI reports\([36,37]\).

### 6.1. OAEI test

#### 6.1.1. Dataset and test goal

We chose the PR and NYT tests from the OAEI in the evaluation. The goal of this experiment is two-fold. Firstly, ADL can be compared with other comparative systems on the same benchmark datasets. Secondly, some key parameters in ADL can be tuned

\(^7\) http://ws.nju.edu.cn/adl.
based on the given golden standards. The organizers of the OAEI provided reference linkages for the tests, where a linkage is constituted by two instances from different datasets referring to the same real-world object. The statistical data of the tests are listed in Table 2. We briefly introduce them as follows:

- The PR test is a small-scale, synthetic test, which consists of two collections of RDF data files about persons (denoted by Persons1 and Persons2, resp.) and one pair of RDF files about restaurants.
- The NYT test is to rebuild the linkages between the NYT dataset and three external large-scale datasets: DBpedia, Freebase and Geonames. We downloaded their RDF dump files and parsed them locally in our experiment. This NYT test was also supplied in random segments for cross-validation of learning systems using training data.

6.1.2. Experimental methodology

We used the conventional precision, recall and F1-measure to evaluate the accuracy of ADL, where F1-measure is the combination of precision and recall. The execution time for training set generation and discriminative property pair learning was also recorded. The NYT test organizers separated the datasets and reference linkages to 10 random partitions, and participants were asked to conduct 10-fold cross-validation. So the final accuracy was obtained by averaging the results over all folds. In accordance with the NYT test, we performed 10-fold cross-validation for the PR test as well. In one validation, we chose 10% disjoint reference linkages in a collection and treated them as the training data to learn discriminative property pairs. Moreover, different from some methods that only linked the instances resident in the references, e.g., [22,38], which largely reduced the candidate amount and simplified the test, the entire DBpedia and Geonames datasets were used in this experiment. But for Freebase, we only imported the instances in the references presently, because it is too large to be parsed and stored on local disks.

6.1.3. Negative examples

We estimated the negative examples complementing to the reference linkages (i.e., positive examples) for each fold. The number of negative examples and their generation time are listed in Table 3. With the help of the filtering method (e.g., the common prefix blocking scheme), the negative examples and positives were kept at a similar size. It was also found that the time cost was not proportional to their sizes. For the synthetic PR dataset, many URIs are similar to each other, thus more time was spent to adjust the number of remaining negative examples.

6.1.4. Parameter analysis

The key parameters in ADL include: the threshold \( \eta \) for property similarities, the number \( k \) of discriminative property pairs for aggregation and the threshold \( \epsilon \) for linking instances. We also compared a few alternatives in our approach, e.g., the information gain measurements.
First of all, we tested the F1-measure with the variation of $\eta$. In Fig. 4, we found that $\eta = 0.8$ is good for determining two properties are similar or not, which can achieve a balance between the accuracy of property matching and the coverage of properties. We fixed $\eta = 0.8$ in the rest of the experiment.

Secondly, we determined the number $k$ of discriminative property pairs for aggregation. Fig. 5 depicts the F1-measure curves. It was observed that the F1-measure drastically rose up with the first one or two discriminative property pairs, and then ascended slowly. This indicates that a few discriminative property pairs (often $k \leq 5$) are accurate enough to characterize an object. If we continued to use more property pairs, improper ones would be chosen, e.g., (category, has_category) for Restaurants, which led to wrong linkages and caused the F1-measure decreasing.

The time for learning discriminative property pairs and the number of property pairs that achieved the highest F1-measure are listed in Table 4. From the table, it was found that the learning time varied between cases, and was mainly affected by the amount of properties and values to be compared. For instance, it took more than 10 times to find discriminative property pairs in NYT–DBpedia than NYT–Freebase, since the instances in DBpedia often have many properties and some properties have very long textual values.

To measure the discriminability of property pairs, we compared the information gain measurement with its variance, the information gain ratio, which is the ratio between the information gain and its intrinsic value. We found that the information gain can give a better ranking for discriminative property pairs, which led to faster convergence of the process.

We tried to set a cutoff $\theta$ for the discriminability of property pairs. Because the sizes of training sets for different classes and pay-level-domains varied, it was difficult to find a uniform cutoff for the discriminability. Nevertheless, the discriminability cutoff $\theta = 0.01$ was used to eliminate highly inaccurate property pairs. We will study how to flexibly adjust it in the future.

Thirdly, we tested various $\epsilon$ for generating instance linkages. Similar to decide the discriminability cutoff $\theta$ for property pairs, we found that it was also hard to use the information gain values as the coefficients in linearly aggregation. Instead, we aggregated the top-$k$ discriminative property pairs ($k = 5$) with equal weighting, and observed that $\epsilon$ was surprisingly stable. If the similarity between two instances is not smaller than $\frac{1}{\epsilon}$, we built a linkage between them. This linkage generation method also implies that an instance is allowed to link to more than one others.

### 6.1.5. Linkage accuracy comparison

The accuracy of ADL was compared with other instance linkage systems, namely ASMOV, CODI, LN2R, ObjectCoref, RiMOM, AgreementMaker, SERIMI and Zhishi.links. We selected them as their evaluation results available on the considered datasets, Table 5 and Table 6.

#### Table 5
Statistical data of the BTC2011 dataset.

<table>
<thead>
<tr>
<th>(Classes, pay-level-domains)</th>
<th>Inst.</th>
<th>Pos.</th>
<th>Neg.</th>
<th>Generate time</th>
<th>Learn time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(foaf:Person, w3.org)</td>
<td>98</td>
<td>137</td>
<td>290</td>
<td>18 min</td>
<td>16 min</td>
</tr>
<tr>
<td>(foaf:Person, dbpedia.org)</td>
<td>53,825</td>
<td>238,645</td>
<td>269,125</td>
<td>5 d</td>
<td>15 h</td>
</tr>
<tr>
<td>(foaf:Person, semanticweb.org)</td>
<td>4970</td>
<td>7066</td>
<td>7961</td>
<td>1 d</td>
<td>1 h and 10 min</td>
</tr>
<tr>
<td>(factbook:Country, fu-berlin.de)</td>
<td>157</td>
<td>4234</td>
<td>3140</td>
<td>1 h and 22 min</td>
<td>3 h and 40 min</td>
</tr>
<tr>
<td>(dbp:City, dbpedia.org)</td>
<td>1850</td>
<td>13,474</td>
<td>8500</td>
<td>8 h and 30 min</td>
<td>1 h and 36 min</td>
</tr>
<tr>
<td>(foaf:Organization, dbpedia.org)</td>
<td>90</td>
<td>30</td>
<td>27</td>
<td>3 min</td>
<td>5 min</td>
</tr>
<tr>
<td>(foaf:Organization, semanticweb.org)</td>
<td>1136</td>
<td>1703</td>
<td>3408</td>
<td>57 min</td>
<td>46 min</td>
</tr>
<tr>
<td>(dbp:Company, dbpedia.org)</td>
<td>10,552</td>
<td>55,689</td>
<td>31,656</td>
<td>16 h and 32 min</td>
<td>2 h and 6 min</td>
</tr>
<tr>
<td>(dbp:EnglishNovels, dbpedia.org)</td>
<td>25</td>
<td>83</td>
<td>75</td>
<td>4 min</td>
<td>5 min</td>
</tr>
<tr>
<td>(swrc:InProceedings, semanticweb.org)</td>
<td>2034</td>
<td>2636</td>
<td>6102</td>
<td>2 h and 1 min</td>
<td>1 h and 11 min</td>
</tr>
<tr>
<td>(linkedmdb:film, linkedmdb.org)</td>
<td>23,440</td>
<td>35,693</td>
<td>70,320</td>
<td>12 h and 51 min</td>
<td>4 h and 58 min</td>
</tr>
<tr>
<td>(yago:WebServices, dbpedia.org)</td>
<td>27</td>
<td>192</td>
<td>125</td>
<td>9 min</td>
<td>6 min</td>
</tr>
<tr>
<td>(dbp:Mammal, dbpedia.org)</td>
<td>3904</td>
<td>16,315</td>
<td>11,712</td>
<td>9 h and 19 min</td>
<td>49 min</td>
</tr>
</tbody>
</table>
where the former five gave their results on the PR test, while the latter three participated in the NYT test. We generally introduce them as follows:

- ASMOV [39] and CODI [40] both employ similarity-based matchers to establish linkages between instances and perform logical inference to remove inconsistent results.
- LN2R [18] integrates a knowledge-based matcher to find semantically equivalent instances (L2R) and applies a similarity propagation algorithm to generate similarities among instances (N2R).
- ObjectCoref [25] proposes a self-supervised framework to learn “local” discriminative property–value pairs for each input instance, and mines frequent property combinations to improve its accuracy.
- RIMOM [21], AgreementMaker [22] and SERIMI [38] are some kinds of purely similarity-based systems, which combine a lot of matchers to exploit a range of characteristics for both concepts and instances.
- Zhishi.links [23] employs indexing techniques on several user-defined properties as well as on homonyms to select candidate instance pairs, allowing it to scale to larger datasets than similar systems. Domain knowledge is integrated for property–value matching.

The accuracy comparison results on F1-measure are shown in Fig. 6. It was observed that ADL achieved the best F1-measure in average in the PR test. In the NYT test, ADL got the best results on the classes of Organizations and People, and the second best result on Locations. All the four systems achieved very good F1-measure (around 0.9) in the NYT People test.

The precision of ADL was pretty good, because on one hand, we found and aggregated the highly discriminative property pairs; on the other hand, the learnt property pairs related to different classes and pay-level-domains to guarantee the adaptability of ADL. Specifically,

- In the PR test, the properties soc_sec_id and phone_number were chosen for linking persons; while phone_number and has_address were picked up for restaurants. It is worth mentioning that RDF sentences contributed to extracting the complete values for has_address. If not to deal with blank nodes, such useful values cannot be utilized. Another reason for the bad accuracy of the rest of the systems is that they involved improper properties in similarity computation, e.g., category on the Restaurants, which led to some wrong linkages.
- In the NYT test, skos:prefLabel, geo:name, fb:type.object.name and rdfs:label were the most discriminative properties. Some properties were combined together, e.g., geo:long and geo:lat are used to denote locations. We observed that the discriminative property pairs that we automatically discovered were largely in accordance with those manually specified in Zhishi.links, verifying the effectiveness of our discriminative property pair discovery. However, Zhishi.links only calculated the name/alias and geographical similarities with domain knowledge, which made it more precise but some potentials were lost. So it performed best on the NYT Locations, while slightly worse on the NYT Organizations and People. For the rest systems, the recall that they achieved was not very stable, since they only used rdfs:label as the discriminative property and missed some candidate linkages.

6.2. BTC2011 test

6.2.1. Dataset and test goal

Different from the OAEI test, in this experiment we aim at linking instances in one large-scale dataset, while the needed training data are unknown beforehand. The BTC2011 dataset was chosen for this purpose. This dataset contains 102,684,277 URIs in 7,364,639 RDF documents. By harvesting it, we identified 285,519 classes and 307,821 pay-level-domains from all the URIs. Some statistical data of the BTC2011 dataset are shown in Table 5. It was found that the bulk (≈79%) of the equivalence relations is derived from the same-as relations. We also counted the number of triples using new OWL2 properties to infer equivalence
6.2.2. Experimental methodology

We selected 20 popular instances from 13 different class and pay-level-domain pairs (see Table 6 and our website). These instances cover a wide range of classes, e.g., person, geography and organization, and they are from several major SW data sources, e.g., DBpedia, SW Dog Food and CIA Factbook. To the best of our knowledge, there is still a lack of reference linkages for our intended test.

In this test, we empirically evaluated the precision and relative recall of our method. Relative recall is the correct number of instance linkages found by one system divided by the total correct number of unique linkages from all systems, offering an approximate solution to the problem that the total number of results is unknown. It has also been adopted to assess the quality of large ontology matching [13].

For precision, we followed the evaluation process in [41]. We invited nine master students major in computer science, who were registered in our SW course this year and had basic knowledge about data linkage, to take peer review on the instance linkages returned by each system. A test case was reviewed by three students, and a student was expected to do 6–7 testing instances. A student firstly judged whether an instance is correctly linked with the input, based on the provided evidences like the equivalence relations, the contexts of the instances, and their dereferenced documents. An instance linkage was scored “1” for correct, “−1” for wrong and “0” for not sure. Then, as the students may have disagreement on the correctness of an instance linkage, the linkage is treated as finally correct if the sum of its scores from all the three students is larger than 0. The union of all correct instance linkages from all systems formed the “golden standard” for testing relative recall. It is worth noting that the evaluation process is time-consuming and tedious. The level of agreement from the nine students on the results and its standard deviation, computed by the free-marginal Fleiss’s $\kappa$ [41], is 0.82 ± 0.13, which indicated a perfect agreement among the students in this experiment.

6.2.3. Training set generation and discriminative property pair discovery

We extracted all instances under the 13 different class and pay-level-domain pairs, and generated positive and negative examples for them. The size of training sets and their generation time are illustrated in Table 6. From the table, it was found that the number of instances in different class and pay-level-domain pairs is skewed, which caused significant difference in computation time; for foaf:person we even spent days to complete the entire process. It is worth noting that the process is offline, and the discriminative property pairs are reusable in the future. During the online linkage, the execution time of ADL in average is about 30 s per testing instance, because it only needs to look up the discriminative property pairs for classes and pay-level-domains, and find other instances with similar property–values.

6.2.4. Linkage accuracy comparison

In this test, three systems were chosen for comparison. SameAs.org\(^8\) and ObjectCoref [25] are two online services and capable of linking instances in one large-scale dataset. Crowdsourcing-based linkage approaches recently attract many attentions, so we implemented a semi-automatic approach, which in a sense followed a simple crowdsourcing-based manner in [11]. As ObjectCoref has been introduced in the previous subsection, we briefly describe the rest of the two systems as follows:

- SameAs.org makes use of at least seven properties that explicitly define the equivalence relations, such as owl:sameAs, rkb:coreferenceData, skos:exactMatch, skos:closeMatch, umbel:isLike, voc:similarTo and obo:hasExactSynonym. We deployed SameAs.org by ourselves on the BTC2011, because it currently runs on its own dataset. Note that half of the testing instances in our experiment were used in the evaluation of ObjectCoref and SameAs.org or mentioned on their websites.

\(^{8}\) http://sameas.org/.
The semi-automatic approach built inverse indexes on various naming properties (e.g., foaf:name, rdfs:label, dbp:name and geo:name) to select candidate instance pairs, and then used the TF-IDF model (as ADL did) to compute the similarities between them. We set an upper bound threshold and a lower bound. If the similarity was larger than the upper bound, the instance linkage was assumed to be correct, and no user interaction was acted. If the similarity was smaller than the lower bound, the instance pair was probably wrong, and no user interaction was conducted either. According to our previous experience, we fixed the upper and lower bounds to 0.7 and 0.3, resp. For the rest of the 204 instance pairs, we issued questions to 10 undergraduate students who did not learn the SW. Each student finished two testing instances disjoint with each other, and in each test she manually picked up the correct ones that she thought. In this implementation, we simply avoided the inconsistency among the students’ feedbacks.

For each testing instance, we exploited the top-5 discriminative property pairs for all pairs of classes and pay-level-domains in the training set. To extend the searching space, we also used the top-5 discriminative properties instead of discriminative property pairs for the counterparting class and pay-level-domain pairs that have not been covered by the training set (see Section 5.2). We observed that the discriminative properties are generally well-known properties, e.g., rdfs:label, foaf:name and dc:title, as IFPs and FPs have not been prevalent on the current SW. For example, there are 1.4 million instances whose types are foaf:Person, but less than 4% of them involve foaf:mbox_sha1sum and foaf:mbox.

Firstly, we compared the precision and relative recall in average of ADL with the training set. In Fig. 7(a), it was found that both the precision and relative recall of ADL are better, since more instances were linked by the discriminative property pairs and the aggregation of the properties filtered some wrong results. In the training set, the instance linkages were correct in most test cases. But in a few cases, a common mistake was the misuse of owl:sameAs as rdfs:seeAlso. We also manually checked 2000 negative examples by ourselves, and found that they were highly accurate (≈99.9%), even in the cases that the positive examples contain a number of wrong results (e.g., dbp:Paul McCartney).

Secondly, we compared the average precision and relative recall of ADL with SameAs.org, ObjectCoref and semi-automatic. The comparison results are shown in Fig. 7(b). In the figure, it was found that ADL achieved good precision and relative recall in average. SameAs.org had the worst recall, because it only performed reasoning to link explicitly equivalent instances. ObjectCoref conducted an iterative learning, so it linked more instances but some of them are not correct, which led to the second best recall but worst precision. The semi-automatic method obtained the best precision thanks to user interaction, but the indexing scheme and lower-bound threshold caused the loss of recall, as a few instance pairs under the lower-bound may still refer to the same object. The standard deviations on F1-measure of ADL, SameAs.org, ObjectCoref and semi-automatic are 0.08, 0.19, 0.23 and 0.21, resp., which indicated that our method was most stable among various cases. Therefore, we empirically conclude that our approach can establish a relatively large amount of accurate instance linkages.

It is also interesting to analyze the portion of the instance linkages given by each system within the golden standard. We collected the instance linkages generated by all the systems and figured how many linkages were provided by exactly one, two, three or four systems. The analysis is depicted in Fig. 8. It shows that every system contributed quite a number of instance linkages to the golden standard, which demonstrated the effectiveness of each system. It was also observed that more instance linkages came from ADL and ObjectCoref, rather than SameAs.org, which indicated that only performing explicit equivalence reasoning would miss many potentials; therefore the similarity-based approaches are needed. Furthermore, this experimental result suggests that a large portion (88%) of instance linkages can be found in terms of only a few discriminative property pairs.

7. Conclusion

Data linkage is important to establish semantic interoperability and realize data integration on the SW. In this paper, we proposed an automatic approach for learning discriminative property pairs to link instances, which characterized common patterns of instances from their abstract types and domains. Our main contributions are summarized as follows:

• We presented an automatic method to build a highly-accurate training set by performing equivalence reasoning and common prefix blocking. The positive and negative examples in the training set are helpful to learn discriminative property pairs.
• We leveraged RDF sentences to extract the complete contexts of the instances in the training set, and discovered matchable properties with a lightweight linguistic matcher. The discriminability of property pairs is measured by the information gain under a specific class pair and a pay-level-domain pair, which can be reused to link instances online.
• We also provided an open source implementation of the approach and conducted evaluation on both the OAEI and BTC2011 datasets. The experimental results demonstrated that, compared with some complex methods, our approach can achieve good precision and (relative) recall with the aggregation of a few discriminative property pairs.

In future work, we look forward to combining our “global” discriminative property pairs with the “local” characteristics of instances, in order to improve adaptability. New approaches will be studied for incremental similarity computation to support dataset update. Additionally, we want to develop more testing instances and reference linkages for the BTC2011 dataset, and submit them to the OAEI to propose a new instance matching track.
Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant Nos. 61370019, 61223003 and 61321491, and in part by the Natural Science Foundation of Jiangsu Province under Grant No. BK2011189. We appreciate the students participating in the evaluation. We also thank the anonymous reviewers for their valuable comments.

References

Wei Hu is a lecturer at the Department of Computer Science and Technology, Nanjing University. He received his B.Sc. degree in Computer Science and Technology in 2005, and his Ph.D. degree in Computer Software and Theory in 2009 both from the Southeast University. His research interests include Semantic Web, ontology engineering and data integration.

Rui Yang is a Master's student at the Department of Computer Science and Technology, Nanjing University. He received his B.Sc. degree in Computer Science and Technology from the Nanjing University in 2013. His research interests include Semantic Web, linked data and recommender systems.

Yuzhong Qu is a full professor and Ph.D. supervisor at the Department of Computer Science and Technology, Nanjing University. He received his B.Sc and M.Sc. degrees in Mathematics from the Fudan University in 1985 and 1988 respectively, and his Ph.D. degree in Computer Software from the Nanjing University in 1995. His research interests include Semantic Web, Web science and novel software technology for the Web. His research is continuously supported by the National Natural Science Foundation of China.