# A Unified Approach to Matching Semantic Data on the Web

Zhichun Wang[a,b,*], Juanzi Li[a], Yue Zhao[c], Rossi Setchi[d], Jie Tang[a]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[b] *College of Information Science and Technology, Beijing Normal University, Beijing, China*
[c] *Department of Computer Science, Columbia University, USA*
[d] *School of Engineering, Cardiff University, Cardiff, Wales, UK*

**Abstract**

In recent years, the Web has evolved from a global information space of linked documents to a space where data are linked as well. The Linking Open Data (LOD) project has enabled a large number of semantic datasets to be published on the Web. Due to the open and distributed nature of the Web, both the schema (ontology classes and properties) and instances of the published datasets may have heterogeneity problems. In this context, the matching of entities from different datasets is important for the integration of information from different data sources. Recently, much work has been conducted on ontology matching to resolve the schema heterogeneity problem in the semantic datasets. However, there is no unified framework for matching both schema entities and instances. This paper presents a unified matching approach to finding equivalent entities in ontologies and LOD datasets on the Web. The approach first combines multiple lexical matching strategies using a novel voting-based aggregation method; then it utilizes the structural information and the already found correspondences to discover additional ones. We evaluated our approach using datasets from both OAEI and LOD. The results show that the voting-based aggregation method provides highly accurate matching results, and that the structural propagation procedure effectively improves the recall of the results.

*Keywords:* semantic web, linked data, ontology matching, instance matching.

## 1. Introduction

In recent years, the Web has evolved from a global information space of linked documents to a space where data are linked as well. Many Linking Open Data (LOD) datasets have been published on the Web, leading to the creation of the Web of Data, a global data space containing billions of assertions [1]. Linked data on the Web are machine-readable, and their meaning is explicitly defined by using ontology classes and properties. Datasets are linked to other external datasets and can in turn be linked by external datasets. Although several rules guide the publication of LOD data, two important issues still require further investigation. The first issue is the schema heterogeneity problem. Although the use of common vocabularies/ontologies such

*Corresponding author. Tel.: +86 010 62773618; fax: +86 010 62781461

*Email addresses:* zcwang@bnu.edu.cn (Zhichun Wang), ljz@keg.cs.tsinghua.edu.cn (Juanzi Li), zhaoy1030@gmail.com (Yue Zhao), setchi@cardiff.ac.uk (Rossi Setchi), jietang@tsinghua.edu.cn (Jie Tang)

as FOAF[1], SIOC[2] and SKOS[3] is encouraged to simplify the processing of LOD data by client applications [2], existing datasets often employ their own schemas, which can be quite different because they are often defined by different organizations. Schema heterogeneity hinders data sharing and data integration. Second, there are fewer established RDF links that connect data than real links between those data. For example, the DBLP and ACM libraries are two datasets in the LOD that contain information about scientific publications. Both datasets contain many duplicate authors and papers, most of which are not linked to each other. RDF links allow client applications to navigate between data sources, discover additional data and combine data on the same individuals that are stored in different locations. Although tools, such as D2RQ[4] and Openlink Virtuoso[5] have been developed to publish LOD data, they do not provide functions for discovering the links between different data sources [3]. Few tools can build semantic correspondences between entities of different datasets at both the schema and instance level. Therefore, the LOD provides a new environment for investigating ontology matching techniques and also raises some challenges for ontology matching.

The problem of ontology matching has been extensively studied in the last decade [4, 5, 6]. Most matching strategies calculate the semantic similarity between any two ontology entities and find the correspondences between them. Different information can be exploited to assess the similarity between ontology entities, e.g., entity name, entity description, taxonomy structure and instance. To perform well, most matching systems, such as RiMOM [7], Falcon-AO [8] and ASMOV [9], combine multiple strategies based on different ontology information. Recently, ontology instance matching has attracted much research interest because many LOD datasets are published on the Web. Instance matching attempts to evaluate the degree of similarity between different descriptions of real objects across heterogeneous data sources and determine whether two object descriptions refer to the same real object in a given domain. Existing systems for instance matching, such as Silk [10] and KonFuss [11] also use multiple strategies to build links between instances.

An important problem in ontology matching is how to combine the results of the multiple matching strategies that are used. Although combining multiple strategies can improve the overall matching results in a set of matching tasks, it cannot guarantee better results in every matching task. Multi-strategy approaches do not outperform their single-strategy alternatives in certain circumstances. Peukert et al. [12] evaluated several similarity combination methods and found that no single strategy returns the best results in all test cases. A good matching result depends on choosing the most appropriate matching and combination method. Furthermore, combination strategies that perform well in ontology schema matching may fail to achieve good results in instance matching because more fields are compared when matching instances. Therefore, an aggregation strategy that performs well in both ontology schema matching and instance matching tasks is needed. Another important issue is how to combine both lexical and structural information to accurately predict the matching results. Some existing approaches use a Vector Space Model to represent merged lexical information on entities and their neighbors and then compute the cosine similarity between the vectors of entities. Other approaches utilize the structural information to propagate the similarities between entities. However, combining the

[1] http://xmlns.com/foaf/spec/
[2] http://rdfs.org/sioc/spec/
[3] http://www.w3.org/TR/skos-reference
[4] http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[5] http://virtuoso.openlinksw.com/

information on neighbors or propagating similarity cannot always improve the quality of matching results because this technique may degrade the performance of matching algorithms. Certain methods should be used to control the use of structural information. For instance, RiMOM [7] defines two factors to determine whether to use structural information in a certain matching task, while Duan et al. [13] proposed a supervised learning approach to determine the degree to which the similarity should be propagated through the structural information.

In this paper, we propose a unified approach for discovering matching correspondences of both schema entities and instances. Given two sets of ontology entities, our approach uses lexical and structural information separately, in two steps, to identify matches between the entities. In the first step, a set of initial matching correspondences are identified using strategies based on the lexical information on the entities; different strategies are combined using a voting-based method to obtain a set of accurate matching results. Additional semantic correspondences are then identified by utilizing the initial matching results and the internal links between the entities. This paper advances the state-of-the-art in this area by making the following contributions:

(1) Development of a novel voting-based aggregation method to combine the matching results of multiple strategies. Instead of extracting matching correspondences after aggregating similarity values, we generate matching results for each individual strategy and then merge them to produce the final result. A voting scheme is used to refine the results by eliminating less possible correspondences. The method generates a set of matching results with very high precision.

(2) Development of a structural matching strategy that only utilizes confirmed matching results. Our approach uses the lexical and structural information separately in two steps. In the first step, a set of initial matching results is identified based on lexical information, and a structural strategy is then used to match entities by comparing the entities that were already matched in their neighborhood. This two-step approach operates effectively and efficiently to identify matching correspondences. The lexical matching strategies produce high-accuracy results, and the structural matching strategy improves the recall of the results.

The rest of this paper is organized as follows. Section 2 presents the background of our work. Section 3 describes our new combination approach and the structure propagation method. Section 4 presents an analysis of the experimental results, and Section 5 discusses some related work. The conclusion is provided in section 6.

## 2. Background

### *2.1. RDF and Linked Data*

The Resource Description Framework (RDF)[6] is a family of W3C[7] specifications that has been widely used as a general method for the conceptual description or modeling of information in Web resources. The RDF describes resources in the form of subject-predicate-object expressions. These expressions are called triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and object.

The RDF Vocabulary Definition Language (RDFS) and the Web Ontology Language (OWL) provide a basis for the creation of vocabularies that can be used to describe entities and their relationships [14]. Vocabularies are collections of classes and properties. Vocabularies are expressed

[6] http://www.w3.org/RDF/
[7] http://www.w3.org/

in the RDF using terms from RDFS and OWL that provide varying degrees of expressivity in modeling domains of interest. Anyone can publish vocabularies in the Web of Data, which in turn can be connected by RDF triples that link classes and properties in one vocabulary to those in another, thereby defining mappings between related vocabularies [1].

The term "Linked Data" refers to a set of best practices for publishing and connecting structured data on the Web [1]. Berners-Lee [15] outlined four basic rules for publishing LOD data on the Web. Figure 1 shows an example of LOD data in the form of a graph that represents three RDF links taken from Tim Berners-Lees FOAF profile. The terms con:assistant, con:phone, rdf:label and foaf:Person are schema information in the graph; they are entities from the Contact[8], RDF[9] and FOAF namespaces, respectively, which specify the meaning of the links. The two URIs in Figure 1 represent two person instances, namely Tim Berners-Lee and his assistant. Therefore, publishing LOD data can simply be considered as the use of certain ontologies to create typed links between things that are represented by URIs.

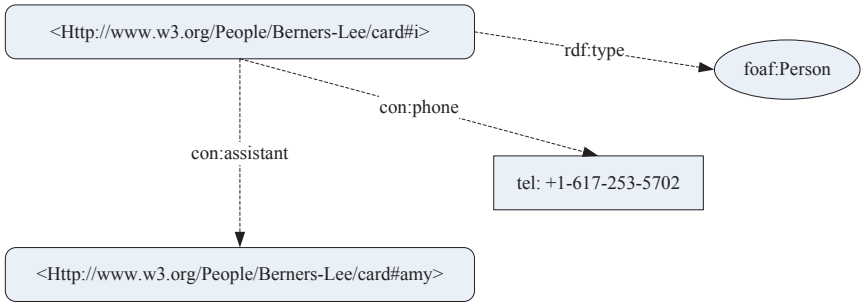



Figure 1: An example of LOD data

### *2.2. Ontology and Ontology Matching*

In computer science, an ontology is a formal, explicit specification of a shared conceptualization [16, 17]. Ontologies can be expressed in several standard languages, including the Web Ontology Language (OWL) [18]. OWL is an ontology language that is recommended by the W3C and provides vocabularies used to define the formal semantics of ontology. Here, we refer to entities in an ontology as classes, properties and instances (or individuals) and define them based on OWL. A class defines a group of entities having common characteristics. Instances are elements of the set of entities modeled by a class. Properties can be divided into object properties and datatype properties; object properties specify relations between two instances while datatype properties specify relations between an instance and RDF literals or XML Schema datatypes. Formally, an ontology is represented as a 6-tuple:

$$O = (C, P, \leqslant_C, \leqslant_P, I, A^O) \tag{1}$$

where $C$ and $P$ denote classes and properties, respectively; $\leqslant_C$ and $\leqslant_P$ are partial orders on classes and properties, respectively; $I$ is a set of instances and $A^O$ is a set of axioms that are

[8]http://www.w3.org/2000/10/swap/pim/contact
[9]http://www.w3.org/1999/02/22-rdf-syntax-ns

Table 1: Features of ontology entities

| Entity | Feature | Type |
|---|---|---|
| Class | label | String |
| | comments | String |
| | properties | Set(Property) |
| | superclasses | Set(Class) |
| | subclasses | Set(Class) |
| | instances | Set(Instance) |
| Property | label | String |
| | comments | String |
| | domain | Set(Class) |
| | range | Set(Class) |
| | superproperties | Set(Property) |
| | subproperties | Set(Property) |
| Instance | label | String |
| | comments | String |
| | data properties | String |
| | direct classes | Set(Class) |
| | object properties | Set(Instance) |

used to associate class and property IDs with either partial or complete specifications of their characteristics and to provide other logical information on classes and properties.

The goal of ontology matching is to identify correspondences between semantically related entities in different ontologies. Here, we use the term entity to refer to a class, property or instance in an ontology. Given a source ontology $O_S$, a target ontology $O_T$ and an entity $e_i$ in $O_S$, the procedure implemented to identify the semantically equivalent entity $e_j$ in $O_T$ is called ontology matching and is denoted as $M$. Formally, for each entity $e_i \in O_S$, the ontology matching $M$ can be represented as [19]:

$$M(e_i, O_S, O_T) = e_j \tag{2}$$

The matching tasks are usually resolved by assessing the similarity between entities in ontologies. Table 1 lists the features used to make the comparison. Here, features of string type are types of *lexical information* because they provide descriptions attached to entities; features with a set of entities are *structural* because they provide links to other entities. Both lexical and structural features should be used to accurately estimate the similarity of entities.

## 3. The Proposed Approach

This section presents the proposed ontology matching approach for both schema and instance matching tasks. The approach divides ontology information into two types: lexical information attached to each entity (e.g., labels, comments and the instances data-type properties) and structural information hidden in the links between the entities, such as the is-a relations between concepts and properties, and the instances object-type properties. To efficiently benefit from the

availability of this information, our approach consists of the following two stages: it first uses the lexical information to identify highly accurate semantic correspondences and then uses a structural matching strategy to identify additional matching results based on the structural information in the dataset and the previously found correspondences. Our approach achieves good results with regard to both ontology schema matching and instance matching problems. We introduce our approach in more detail below.

### *3.1. Discover Matching Correspondences Based on Lexical Information*

The first step of our approach is to find a set of initial matching results based on the lexical information of the entities.

#### *3.1.1. Similarity Metrics and Matching Strategies*

Several lexical similarity metrics can be used to compare entities. Here, we use two similarity metrics: edit-distance-based similarity and vector-based similarity. These two metrics are both based on lexical information; the edit-distance-based similarity metric is used for short strings, while the vector-based similarity metric is preferred for long texts (comprising many words).

**Edit-distance-based similarity:** The edit distance between two strings is the minimal cost of operations (insertion, replacement and deletion of characters) that must be applied to one of the strings to obtain the other string. Given two strings $s$ and $t$, the edit-distance-based similarity between them is defined as:

$$S_e(s,t) = 1 - \frac{|\{ops\}|}{\max(length(s), length(t))} \tag{3}$$

where $|\{ops\}|$ indicates the minimum number of operations required to transform $s$ to $t$, and $length(s)$ and $length(t)$ represent the number of characters in $s$ and $t$, respectively.

**Vector-based similarity:** The vector-based similarity is calculated between the feature vectors of ontology entities. Before the similarity computation, a virtual document is generated for each ontology entity by combining different sources of strings (e.g., labels, comments) attached to the entity. Then the virtual document of each entity is represented as a vector, where the elements in the vector are weights assigned to the words in the virtual document using TF-IDF method [20]. For a word $i$ in an entity's virtual document $j$, the weight of the word is computed as

$$\omega_{ij} = tf_{ij} \cdot \lg \frac{N}{df_i} \tag{4}$$

where $tf_{ij}$ is the number of occurrences of $i$ in $j$, $df_i$ is the number of virtual documents that contain $i$, and $N$ is the total number of virtual documents. The vector-based similarity between two entities is computed as the cosine value between their virtual documents:

$$S_v(d,k) = \frac{\sum_{i=1}^{M} \omega_{id} \cdot \omega_{ik}}{\sqrt{\sum_{i=1}^{M} \omega_{id}{}^2} \cdot \sqrt{\sum_{k=1}^{M} \omega_{ik}{}^2}} \tag{5}$$

where $M$ is the total number of distinct words in all of the virtual documents.

Several types of lexical information can be used to assess the similarity between the entities. In our approach, three matching strategies are used for ontology schema matching, including

the Name-based strategy, Metadata-based strategy and Instance-based strategy; a property-based strategy is used for ontology instance matching.

**Name-based strategy:**

$$M_{name}(e_1, e_2) = S_e(label(e_1), label(e_2)) \tag{6}$$

where $label(e)$ is the value of rdfs:label of an entity; if there is no information for rdfs:label, $label(e)$ corresponds to the last segment of the ontology entity's URI. If there are more than one rdfs:label assertions for an entity, we calculate the edit-distances of all possible label pairs and keep the largest as the similarity of the two entities.

**Metadata-based strategy:**

$$M_{meta}(e_1, e_2) = S_v(meta(e_1), meta(e_2)) \tag{7}$$

where $meta(e)$ is a set of words formed by combining the values of rdfs:label and rdfs:comment of entity $e$.

**Instance-based strategy:**

$$M_{inst}(e_1, e_2) = S_v(inst(e_1), inst(e_2)) \tag{8}$$

If $e$ is a class, $inst(e)$ is the union of the metadata of the instances belonging to class $e$, i.e., $inst(e) = \bigcup_{t_j \in I(e)} meta(t_j)$, where $I(e)$ denotes the set of instances belonging to class $e$, $meta(t_j)$ denotes a set of words in the values of rdfs:label and rdfs:comment of the instance $t_j$. If $e$ is a datatype property, $inst(e)$ is the union of lexical values attached to instances by the property $e$, i.e., $inst(e) = \bigcup_{t_j} prop_e(t_j)$, where $prop_e(t_j)$ denotes the set of words attached to instance $t_j$ by property $e$; if $e$ is an object property, $inst(e) = \emptyset$.

**Property-based strategy:**

$$M_{prop<e,e'>}(t_1, t_2) = \begin{cases} S_e(prop_e(t_1), prop_{e'}(t_2)) & \text{if } min(|prop_e(t_1)|, |prop_{e'}(t_2)|) < 3 \\ S_v(prop_e(e_1), prop_{e'}(t_2)) & \text{if } min(|prop_e(t_1)|, |prop_{e'}(t_2)|) \geqslant 3 \end{cases} \tag{9}$$

Property-based strategy only uses lexical information attached to instances by datatype properties. Here, $prop_e(t_j)$ also denotes the set of words attached to instance $t_j$ by property $e$. In the process of ontology instance matching, property-based matching strategy needs to run multiple times for different datatype property pairs $< e, e^{'} >$ of two ontologies; so multiple similarities can also be obtained for every instance pair using property-based matching strategy.

*3.1.2. Voting-based Strategy Combination*

Given two sets of entities $E_1$ and $E_2$, and $k$ matching strategies, $M_i$ $(i = 1, 2, ..., k)$, our approach first finds a set of matching correspondences by combining the results of the multiple strategies used. Traditional composite ontology matching approaches use several matching strategies to compute the similarities between entities with different techniques and then aggregate those similarities to produce a combined similarity. The final results are then extracted based on the aggregated similarities (see Figure 2). Our approach combines multiple strategies in a different way; namely, it first derives a set of matching results from each single matching strategy and then combines those matching results using a voting method (see Figure 3). This process is described in more detail below.

**(1) Extracting Matching Correspondences from a Single Strategy**

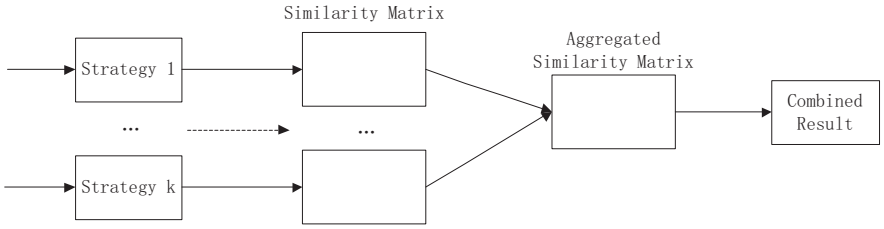



Figure 2: Strategy combination by similarity aggregation

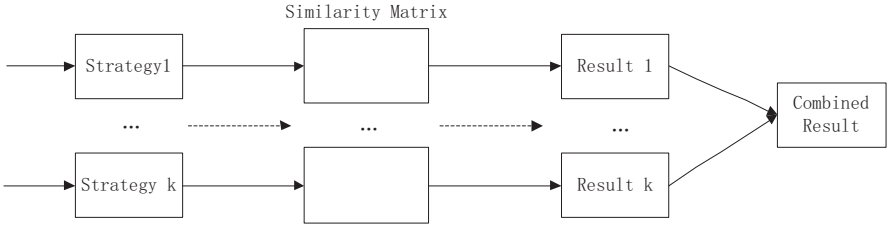



Figure 3: Strategy combination by the voting method

A matching strategy $M_i$ independently computes the similarity of every entity pair $p =< e, e^{'} > (e \in E_1 \ , \ e^{'} \in E_2)$. The result of the similarity computation is an $n \cdot m$ similarity matrix $\mathcal{M}_i$, where $n = |E_1|$ and $m = |E_2|$. The element $x_{jk}$ in the $j$th row and $k$th column of $\mathcal{M}_i$ denotes the similarity between the $j$th entity in $E_1$ and the $k$th entity in $E_2$.

After the similarity computation, we extract the matching results from the similarity matrix. We use the Naive Descending Extraction (NDE) algorithm [21] in combination with a threshold filtering method to select the matching results from the similarity matrix (see Algorithm 1). NDE is based on the 1-1 matching assumption that each entity in $E_1$ can only matches one entity in $E_2$. In addition, we only select extracted matching correspondences that have similarity values greater than a threshold $\sigma$; this helps to exclude possible incorrect matching results. Given two sets of $n$ and $m$ entities, the majority of the computation time is spent in sorting the $n \times m$ similarities in the similarity matrix. In our approach, the Quicksort algorithm [22] is used to sort all of the similarities in descending order. Therefore, the average time cost of NDE is $O((nm)\log(nm))$. The cost in the worst case is $O((nm)^2)$.

The threshold $\sigma$ is determined by applying the k-means clustering algorithm to the similarity values. The algorithm partitions the similarity values into two sets; one set is close to 1 and the other set is close to 0. The boundary between the two cluster sets is then selected as the threshold $\sigma$.

**(2) Combining the Results from Multiple Matching Strategies**

Let $R = \{R_i | i = 1...k\}$ denote a set of matching results obtained by $k$ strategies. We use a voting-based method to combine the strategies and obtain the final matching result. The votes that entity pair $p =< e, e^{'} > (e \in E_1 \ , \ e^{'} \in E_2)$ receives from the $k$ strategies are first counted. Here, we define the votes of an entity pair as the number of strategies whose results contain the pair. Thus, the votes of $p$ can be expressed as $v(p) = |\{R_i | p \in R_i, i = 1...k\}|$. After the votes of all the entity pairs are counted, we apply the NDE algorithm combined with the threshold filtering method to the votes to obtain the final results.

Traditional methods combine strategies by using different mathematical techniques for similarity aggregation, including the Maximum, Minimum, Average, and Weighted Sum techniques. Our voting-based approach does not directly operate on similarity values, and no numerical calculations are made based on the similarity values. In our approach, each matching strategy can be considered as a condition that tests the entity pairs; those pairs that satisfy many conditions are likely to be true matching results. The voting-based method can be applied to any number of strategies, and we do not need to tune the weights of each strategy.

**Algorithm 1** Naive descending matching extraction

**Input:**

Two sets of entities $E_1$ and $E_2$;
The $n \cdot m$ similarity matrix , $\mathcal{M}$;
The threshold of similarity, $\sigma$;

**Output:**

A set of selected matching correspondences, $R$;

$S \leftarrow$Sort all of the similarities in $\mathcal{M}$ in descending order;
$H \leftarrow$ Sorted similarities' indexes in $\mathcal{M}$;
$R \leftarrow \emptyset$;
$i \leftarrow 1$;
**while** $S(i) \geq \sigma$;
    $R \leftarrow R \cup < e_{H(i)_1}, e'_{H(i)_2} >$, $e_{H(i)_1} \in E_1$ and $e_{H(i)_2} \in E_2$;
    Delete the $H(i)_1$th row and the $H(i)_2$th column in $\mathcal{M}$;
    Delete the corresponding similarities and indexes in $S$ and $H$;
    $i = i + 1$;
**end while**;
**return** $R$;

*3.2. Identifying Additional Matching Correspondences using Structural Information*

After obtaining a set of initial matching results based on the lexical information of the entities, we find additional matching correspondences using the structural information. Figure 4 shows the structural links of class, property and instance. Our approach is based on the assumption that two entities are possible matching correspondences if they are related to some previously matched entities. Let the initial matching pairs between $E_1$ and $E_2$ be $R$. The following steps are necessary to identify additional matching correspondences.

(1) Find the set of entities in $E_1$ that do not occur in the matching pairs in $R$ and denote them as $\bar{E}_1$; find the set of entities in $E_2$ that do not occur in the matching pairs in $R$ and denote them as $\bar{E}_2$.

(2) For each entity $e \in \bar{E}_1$, construct a feature set $F(e) = (E_1 - \bar{E}_1) \cap N(e)$, where $N(e)$ denotes the entities linked to $e$; for each entity $e' \in \bar{E}_2$, construct a feature set $F(e') = (E_2 - \bar{E}_2) \cap N(e')$, where $N(e')$ denotes the entities linked to $e'$.

(3) For each candidate matching pair $< e, e' >$, $e \in \bar{E}_1$ and $e' \in \bar{E}_2$, compute the similarity between $e$ and $e'$:

$$Sim(e, e') = \begin{cases} 0 & \text{if } |F(e)| + |F(e')| = 0 \\ \frac{|\{<f,g> | <f,g> \in R, f \in F(e), g \in F(e')\}|}{|F(e)| + |F(e')|} & \text{if } |F(e)| + |F(e')| \neq 0 \end{cases} \quad (10)$$

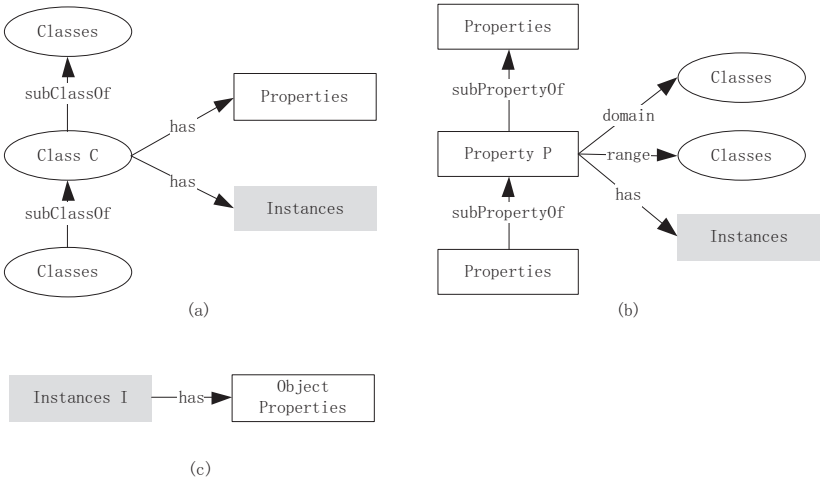



Figure 4: Structural links of (a) classes, (b) properties, (c) instances

(4) Select the matching pairs $R^*$ from the matching candidates using the method in Algorithm 1, and add the new matching results $R^*$ to $R$.

After adding the new matching candidates to the initial result, repeat the above steps until no more matching correspondences can be found or the algorithm reaches a maximum number of iterations. In contrast to the similarity propagation method, our approach only uses the derived correspondences to help identify additional correspondences. This approach can effectively avoid the influence of small similarity values that are also propagated in traditional methods.

## 4. Evaluation

### *4.1. Datasets*

We evaluate our approach using both the OAEI and LOD datasets. OAEI is an annual ontology matching campaign that provides authoritative tests and evaluations of ontology matching technologies. Here, we use the OAEI Benchmark dataset for the ontology schema matching evaluation and OAEI IIMB (including IIMB_SMALL and IIMB_LARGE) for the ontology instance matching evaluation. Test cases in OAEI are artificially generated, so we also use real datasets in the LOD to evaluate our approach. For ontology schema matching, our approach is evaluated using eight popular ontology schemas to determine matching correspondences among them; for instance matching, our approach interlinks the DBpedia and LinkedMDB datasets. Detailed information about these datasets is provided below.

#### *4.1.1. Benchmark Datasets in OAEI 2010*

The benchmark test library of OAEI 2010 contains 111 ontologies in the bibliography domain. Of these ontologies, one is a reference ontology and the rest are ontologies that are systematically generated from the reference ontology. The reference ontology includes 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. The test data are generated by discarding some of the information to evaluate how the algorithm performs when this information is absent [23].

*4.1.2. Matching LOD Vocabularies*

Jain et al. [24] selected eight popular ontologies from the LOD (see Table 2) and built reference matching correspondences for Music-BBC, Music-DBpedia, GEO-DBpedia, SWC-AKT, and SWC-DBpedia, FOAF-SIOC and FOAF-DBpedia matching tasks. In this paper, we use these datasets and the reference matching results established by Jain et al. to evaluate the proposed approach.

Table 2: LOD ontologies (#C denotes the number of concepts)

| Ontology | LOD Datasets | #C |
|---|---|---|
| DBpedia (D) | DBpedia | 204 |
| Geonames (G) | Geonames, Geospecies | 11 |
| Music Ontology (M) | Jamendo, Music, Brainz, DBTunes | 136 |
| BBC Program (B) | BBC Programs, BBC Music | 100 |
| FOAF (F) | FOAF, Music, Brainz | 16 |
| SIOC (S) | DBpedia, Linked-MDB | 14 |
| AKT Reference Ontology (A) | ACM, DBLP | 17 |
| Semantic Web Coreference Ontology (W) | SW Conference Corpus | 177 |

*4.1.3. IIMB Datasets in OAEI 2010*

**IIMB** is composed of a set of test cases. Each case represents a set of instances built from an initial dataset of real LOD data extracted from the web. The test cases are generated by transforming the individual descriptions in the reference dataset. This includes: (i) data value transformation, which simulates the fact that data expressing the same real object in different data sources may be different because of data errors or the usage of different conventional patterns for data representation; (ii) data structure transformation, which simulates the fact that the same real object may be described using different properties/attributes in different data sources; and (iii) data semantic transformation, which simulates the fact that the same real object may be classified differently in different data sources. IIMB is a collection of OWL ontologies consisting of 29 concepts, 20 object properties, 12 data properties and thousands of individuals that are divided into 80 test cases. The 80 test cases are divided into four sets of 20 test cases each. The first three sets are different implementations of the data value, data structure and data semantic transformations, while the fourth set is obtained by combining the other three types of transformations. IIMB is created by extracting data from Freebase, an open knowledge base that contains information approximately 11 million real objects, including movies, books, TV shows, celebrities, locations, and companies.

*4.1.4. Interlinking LinkedMDB and DBpedia*

In addition to the OAEI datasets, two datasets from the LOD[10], namely DBpedia and LinkedMDB, are chosen to evaluate the proposed approach.

**DBpedia**[11] is built by extracting structured content from the information contained in the Wikipedia project. The DBpedia knowledge base currently describes more than 3.5 million things, 1.67 million out of which are classified in a consistent ontology, including 364,000 people, 462,000 places, 99,000 music albums, 54,000 films, 17,000 video games, 148,000 organizations, 169,000 species and 5,200 diseases. The DBpedia knowledge base altogether consists of over 672 million pieces of information (RDF triples); of these, 286 million were extracted from the English edition of Wikipedia, and 386 million were extracted from other language editions.

The **LinkedMDB**[12] is a semantic web database of movies and includes a large number of interlinks to several datasets on the open data cloud and references to related webpages. LinkedMDB is one of the densest examples of interlinking datasets in the Linking Open Data cloud and has as many interlinks and page references as entities. There are more than six million triples in the LinkedMDB dataset, including 85,000 films, 50,000 actors, 17,000 directors, 17,000 writers, and 14,000 producers. There are currently more than 30,000 links between LinkedMDB and DBpedia.

*4.2. Performance Metrics*

We use precision, recall, and F1-Measure to measure the performance of the proposed approach. These measures are defined as follows:

Precision ($p$) is the percentage of correctly discovered matching results among all of the discovered matching results.

$$p = \frac{|A \cap T|}{|A|} \tag{11}$$

Recall ($r$) is the percentage of correctly discovered matching results among all of the correct matching results.

$$r = \frac{|A \cap T|}{|T|} \tag{12}$$

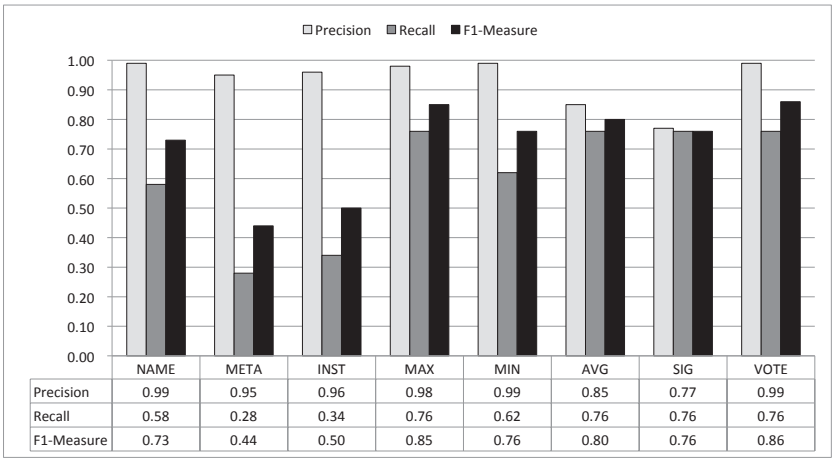
F1-Measure ($F1$) considers the overall results of precision and recall.

$$F1 = \frac{2pr}{p + r} \tag{13}$$

*4.3. Evaluation on the OAEI Benchmark*

We first investigate the effect of the voting-based strategy combination method and then test whether the structural matching strategy can effectively find additional matching results.

[10] http://linkeddata.org/
[11] http://dbpedia.org/About
[12] http://www.linkedmdb.org/




| | NAME | META | INST | MAX | MIN | AVG | SIG | VOTE |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.99 | 0.95 | 0.96 | 0.98 | 0.99 | 0.85 | 0.77 | 0.99 |
| Recall | 0.58 | 0.28 | 0.34 | 0.76 | 0.62 | 0.76 | 0.76 | 0.76 |
| F1-Measure | 0.73 | 0.44 | 0.50 | 0.85 | 0.76 | 0.80 | 0.76 | 0.86 |

Figure 5: Results of different combination methods in the Benchmark test

*4.3.1. Effect of the Strategy Combination Method*

To validate the effect of the voting-based combination method, we compare it with four other commonly used methods. These four counterparts are based on similarity aggregation, in which the similarity values returned by the different strategies are first aggregated and then matching results are selected based on the combined similarities. The other methods are described as follows:

**MAX** aggregates the similarities by selecting the maximum similarity value of any matching strategy;

**MIN** aggregates the similarities by selecting the minimum similarity value of any matching strategy;

**AVG** aggregates the similarities by computing the average similarity value of all matching strategies;

**SIGMOID(SIG)** aggregates the similarities by computing the average similarity values transformed by a sigmoid function.

For each test in the Benchmark, we first compute three similarity matrixes based on three individual matching strategies (i.e., Name-based strategy, Metadata-based strategy and Instance-based strategy). We then use MAX, MIN, AVG, and SIGMOID to aggregate the individual similarity matrixes and compute four combined similarity matrixes. Based on these similarity matrixes, we apply the method in Algorithm 1 to select matching results for the individual and combined strategies. For each of the four combined strategies, we test threshold values ranging from 0.1 to 0.9 and keep the results with the highest F1-Measure.

Figure 5 shows the results of the different combination methods. The results of the individual matching strategies all result in high precisions that are greater than 95%, but their recall results are lower. The recall is significantly improved when multiple matching strategies are combined. Among the five combined approaches that were compared, MIN and VOTE yield the highest precision, but the recall of MIN is low. MAX, AVG, SIG and VOTE achieve similar recall results. In general, VOTE outperforms all of the other methods; it yields the highest precision, recall and F1-measure, which are 99, 76 and 86%, respectively.

*4.3.2. Effect of the Structural Matching Strategy*

We can obtain a set of accurate matching correspondences by combining multiple lexical matching strategies. The goal of the structural matching strategy is to find additional correspondences based on existing correspondences. Here, we compare the results of our approach before and after using the structural strategy.

Table 3 shows the results of the two stages separately. 1XX, 2XX, and 3XX denote three groups of matching tasks in the Benchmark test, while H-mean represents the overall performance. For the tests in the 1XX group, we can already obtain perfect precision and recall using only lexical strategies, so no improvements are gained by using the structural strategy. For the tests in the 2XX group, the precisions in both stages are the same, but the recall increases by 8%. In the 3XX group, both the precision and recall have improved; the precision increases from 92% to 94% and the recall increases from 73% to 76%. The results of both stages demonstrate that the structural strategy can effectively improve the recall while maintaining the same precision. For other tasks, the use of the structural strategy does not degrade the quality of the results, regardless of whether the structural information yields improvements.

Table 3: The results of two steps in the proposed approach on the Benchmark dataset

| Process | Lexical | | | Lexical+Structural | | |
|---|---|---|---|---|---|---|
| Test | Pre. | Rec. | F1. | Pre. | Rec. | F1. |
| 1XX | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2XX | 0.99 | 0.75 | 0.85 | 0.99 | 0.83 | 0.90 |
| 3XX | 0.92 | 0.73 | 0.81 | 0.94 | 0.76 | 0.84 |
| H-mean | 0.99 | 0.76 | 0.86 | 0.99 | 0.84 | 0.91 |

*4.3.3. Comparison with Other Matching Systems*

Here, we compare our approach with the participants in OAEI 2010. Table 4 displays the results of all the participants. Our approach achieves results similar to that of ASMOV, which won the OAEI 2010 competition. Our approach yielded higher precision than ASMOV, although ASMOV achieved better recall.

*4.4. Evaluation on the LOD Ontologies*

In this section, we evaluate our approach by matching seven ontologies in the LOD. The reference matching results provided by [24] specify *subclass* and *equivalence* relationships between classes in different ontologies. We use our approach to find only *equivalence* relations between classes. We have found that many subclass relations in the reference matching correspondences can be inferred from the equivalence classes. Therefore, we use the following rule to generate a set of *subclass* relations based on the *equivalence* classes found by our approach:

$$M(e_i, O_S, O_T) = e_j \Longrightarrow \begin{cases} e_p \succ e_q & e_p \in super(e_i), e_q \in sub(e_j) \\ e_p \prec e_q & e_p \in sub(e_i), e_q \in super(e_j) \end{cases} \tag{14}$$

where $A \succ B$ means that $B$ is a subclass of $A$ and $super(e)$ and $sub(e)$ denote all of the super and sub classes of entity $e$ (including both explicit and implied relations), respectively.

Table 4: Results of participants in the OAEI 2010 Benchmark test

| System | | 1xx | 2xx | 3xx | H-mean |
|---|---|---|---|---|---|
| AgrMaker | Prec. | 0.98 | 0.95 | 0.88 | 0.95 |
| | Rec. | 1.00 | 0.84 | 0.58 | 0.84 |
| AROMA | Prec. | 1.00 | 0.94 | 0.83 | 0.94 |
| | Rec. | 0.97 | 0.46 | 0.58 | 0.48 |
| ASMOV | Prec. | 1.00 | 0.99 | 0.88 | 0.98 |
| | Rec. | 1.00 | 0.89 | 0.84 | 0.89 |
| CODI | Prec. | 1.00 | 0.83 | 0.95 | 0.84 |
| | Rec. | 1.00 | 0.42 | 0.45 | 0.44 |
| Ef2Match | Prec. | 1.00 | 0.98 | 0.92 | 0.98 |
| | Rec. | 1.00 | 0.63 | 0.75 | 0.65 |
| Falcon | Prec. | 1.00 | 0.81 | 0.89 | 0.82 |
| | Rec. | 1.00 | 0.63 | 0.76 | 0.65 |
| GeRMeSMB | Prec. | 1.00 | 0.96 | 0.90 | 0.96 |
| | Rec. | 1.00 | 0.66 | 0.42 | 0.67 |
| MapPSO | Prec. | 1.00 | 0.67 | 0.72 | 0.68 |
| | Rec. | 1.00 | 0.59 | 0.39 | 0.60 |
| SOBOM | Prec. | 1.00 | 0.97 | 0.79 | 0.97 |
| | Rec. | 1.00 | 0.74 | 0.75 | 0.75 |
| TaxoMap | Prec. | 1.00 | 0.86 | 0.71 | 0.86 |
| | Rec. | 0.34 | 0.29 | 0.32 | 0.29 |
| Proposed | Prec. | 1.00 | 0.99 | 0.94 | 0.99 |
| | Rec. | 1.00 | 0.83 | 0.76 | 0.84 |

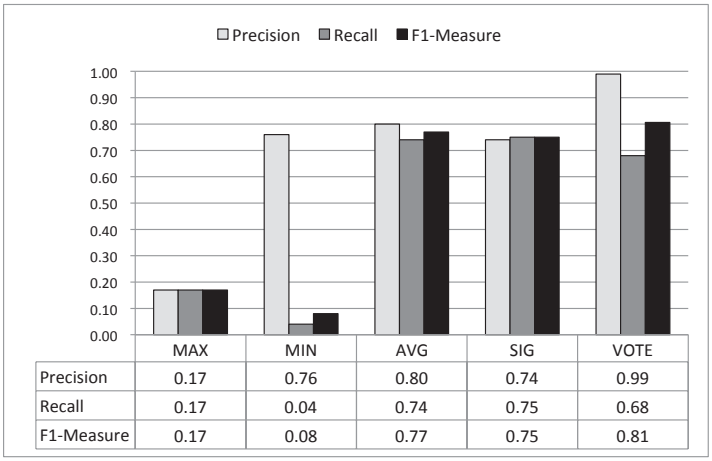
Table 5: Results of matching LOD ontologies.

| | S-Match | | | AROMA | | | BLOOMS | | | Proposed | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Test | Pre. | Rec. | F1. | Pre. | Rec. | F1. | Pre. | Rec. | F1. | Pre | Rec | F1. |
| M-B | 0.04 | 0.28 | 0.07 | 0.00 | 0.00 | 0.00 | 0.63 | 0.78 | 0.70 | 1.00 | 0.29 | 0.45 |
| M-D | 0.08 | 0.30 | 0.13 | 0.45 | 0.01 | 0.02 | 0.39 | 0.62 | 0.48 | 0.96 | 0.08 | 0.15 |
| F-D | 0.11 | 0.40 | 0.17 | 0.33 | 0.04 | 0.07 | 0.67 | 0.73 | 0.70 | 0.98 | 0.52 | 0.68 |
| G-D | 0.23 | 1.00 | 0.37 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.69 | 0.82 |
| S-F | 0.52 | 0.11 | 0.18 | 0.30 | 0.20 | 0.24 | 0.55 | 0.64 | 0.59 | 1.00 | 0.35 | 0.52 |
| W-A | 0.06 | 0.40 | 0.10 | 0.38 | 0.03 | 0.06 | 0.42 | 0.59 | 0.49 | 1.00 | 0.01 | 0.02 |
| W-D | 0.15 | 0.50 | 0.23 | 0.27 | 0.01 | 0.02 | 0.70 | 0.40 | 0.51 | 0.89 | 0.19 | 0.31 |
| Avg. | 0.17 | 0.43 | 0.18 | 0.25 | 0.04 | 0.06 | 0.48 | 0.54 | 0.50 | 0.98 | 0.30 | 0.42 |

Table 5 presents the results of S-Match, AROMA, BLOOMs and our approach for these seven matching tasks. The results of the former three systems are obtained from [24]. The results show that our approach yields matching results with high precision. The recalls are low because we only consider the *equivalence* relations and some inferred *subclass* relations.

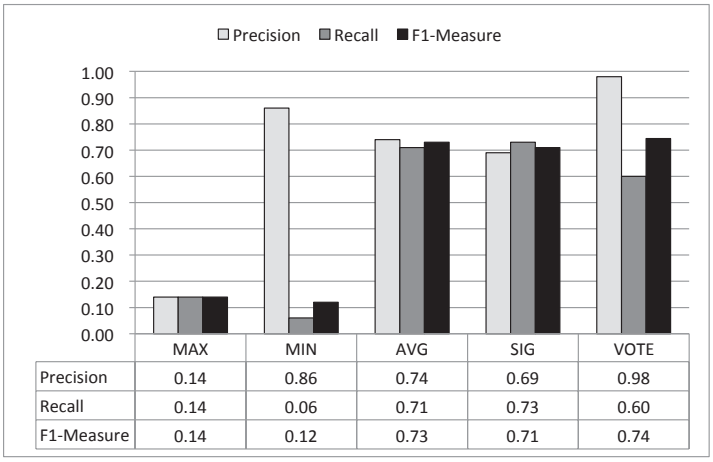
### *4.5. Evaluation on OAEI IIMB*

IIMB includes a set of tests for evaluating instance matching techniques. We test the effect of the strategy combination method and the structural matching strategy on instance matching tasks and then compare our approach with the participants in the OAEI IIMB task.




| | MAX | MIN | AVG | SIG | VOTE |
|---|---|---|---|---|---|
| Precision | 0.17 | 0.76 | 0.80 | 0.74 | 0.99 |
| Recall | 0.17 | 0.04 | 0.74 | 0.75 | 0.68 |
| F1-Measure | 0.17 | 0.08 | 0.77 | 0.75 | 0.81 |

Figure 6: Results of different combination methods on IIMB-SMALL




| | MAX | MIN | AVG | SIG | VOTE |
|---|---|---|---|---|---|
| Precision | 0.14 | 0.86 | 0.74 | 0.69 | 0.98 |
| Recall | 0.14 | 0.06 | 0.71 | 0.73 | 0.60 |
| F1-Measure | 0.14 | 0.12 | 0.73 | 0.71 | 0.74 |

Figure 7: Results of different combination methods on IIMB-LARGE

*4.5.1. Effect of the Strategy Combination Method*

Figure 6 and Figure 7 present the results of the different combination methods on the IIMB_SMALL dataset and IIMB_LARGE dataset, respectively. The results of the methods obtained using these two datasets are similar. The MAX strategy achieves low precision and recall, and the MIN strategy can achieve high precision but low recall. AVG and SIG yield similar results with good precision and recall. The VOTE strategy achieves the highest precision with acceptable recall. In the case of IIMB_SMALL, the precision is 0.99 and the recall is 0.68; for IIMB_LARGE, the precision is 0.98 and the recall is 0.60. Therefore, our voting-based combination method also works well for instance matching tasks.

*4.5.2. Effect of the Structural Matching Strategy*

Table 6 compares the results of the two stages of our approach. Using the structural strategy, the recall in the IIMB_SMALL dataset increased by 0.14 with a decrease in precision of 0.04; in the IIMB_LARGE dataset, the recall increased by 0.20 with a 0.07 decrease in precision.

Table 6: The results of the two steps in the proposed approach on IIMB dataset

| Test | Lexical | | | Structural+Lexical | | |
|---|---|---|---|---|---|---|
| | Pre | Rec | F1 | Pre | Rec | F1 |
| IIMB_SMALL | 0.99 | 0.68 | 0.81 | 0.95 | 0.82 | 0.88 |
| IIMB_LARGE | 0.98 | 0.60 | 0.74 | 0.91 | 0.80 | 0.85 |

*4.5.3. Comparison with Other Matching Systems*

Table 7 shows the results of the participants in the OAEI IIMB task. Our approach achieves better results than ASMOV. The results derived on IIMB_SMALL are similar to those obtained using CODI; however, although CODI and our approach yield the same recall for the IIMB_LARGE dataset, CODI provides higher precision.

Table 7: Results of the OAEI IIMB participants in OAEI 2010

| Test | IIMB_SMALL | | | IIMB_LARGE | | |
|---|---|---|---|---|---|---|
| Algorithm | Pre. | Rec. | F1. | Pre. | Rec. | F1. |
| ASMOV | 0.86 | 0.82 | 0.84 | 0.85 | 0.80 | 0.82 |
| CODI | 0.96 | 0.83 | 0.89 | 0.96 | 0.80 | 0.87 |
| Proposed | 0.95 | 0.82 | 0.88 | 0.91 | 0.80 | 0.85 |

*4.6. Evaluation on Interlinking DBpedia and LinkedMDB*

To evaluate our approach using real LOD datasets, we use our approach to interlink instances in DBpedia and LinkedMDB. We only link films in DBpedia and LinkedMDB. Currently, there are 18,000 established sameAs links between film instances. We randomly selected 500 sameAs links between films and manually verified that they were correct. These 500 verified sameAs links are used as the ground truth for the evaluation.

Here, we compare our approach with that of Silk [10], a link discovery engine that automatically finds RDF links between datasets. We let Silk compute the Jaro distance between the names

of films, the Jaro distance between the names of directors, and the number similarity between the published times of the films. The similarities are aggregated by the Average method in Silk. Table 8 shows the results of the two approaches. Our approach results in high precision and recall in linking real LOD datasets, and it outperforms Silk in both precision and recall.

Table 8: Results on Interlinking DBpedia and LinkedMDB

| Approach | Pre. | Rec. | F1. |
|---|---|---|---|
| Silk | 0.93 | 0.91 | 0.92 |
| Proposed | 0.98 | 0.94 | 0.96 |

### *4.7. Summary*

Our experiments demonstrate that our approach can effectively identify matching correspondences in semantic data at both the schema and instance level. Using the OAEI tests, we validate the effect of the voting-based combination method and the structural matching strategy. The results show that the voting-based method achieves high precision, which averages 0.98 in the OAEI tests. The average improvement of recall using the structural matching strategy is +0.12 for schema matching and +0.17 for instance matching. In tests using the LOD datasets, our approach also achieves high precision in both schema matching and instance matching tasks. The recall of schema matching using the LOD datasets is low because our approach can only find equivalent relations among the entities, and the reference results contain many subsumption relations.

## 5. Related Work

### *5.1. Ontology Matching*

Many ontology matching methods have been proposed over the last decade, including heuristic methods [7, 25, 8], graph based methods [26, 27], learning-based methods[28, 29], probabilistic methods [30, 31], and reasoning-based methods [32, 33]. Here, we refer only to the work most closely related to our study.

COMA [34] is a schema matching tool that supports multiple schema types. It provides a library of matching algorithms and allows the use of different algorithms and combination strategies, such as MAX, MIN, AVG and SIG, but users should select the strategies and combination methods according to the matching problem. In our approach, we use the voting-based combination method to aggregate the results of the multiple matching strategies. We compare the performances of the different combination methods; the results show that our voting-based method outperforms the other algorithms in various tasks.

ASMOV [9] is an automated ontology matching system that combines a comprehensive set of element- and structure-level measures of similarity with a technique that uses formal semantics to verify whether computed correspondences comply with desired characteristics. ASMOV computes four types of similarities, including a lexical similarity, two structural similarities and an external similarity. It combines these similarities using a weighted sum, where the weights are adjusted based on static rules. In our approach, we use lexical and structural information separately in two steps instead of combining them based on a weighted sum. After obtaining a

set of matching results, the ASMOV algorithm uses a semantic verification process to exclude invalid matching pairs. Matching correspondences that cause multiple-entity matching, crisscross matching, disjointness-equivalence contradiction, subsumption incompleteness, and domain and range incompleteness are eliminated. Our approach uses the NDE algorithm to extract matching results and only imposes a 1-1 matching constraint on the matching results.

PRIOR+ [35] is a generic and adaptive ontology mapping approach that is based on propagation theory, information retrieval techniques and artificial intelligence. The approach consists of three major modules, i.e., the IR-based similarity generator, the adaptive similarity filter, the weighted similarity aggregator, and the neural network-based constraint satisfaction solver. The approach first measures both the linguistic and structural similarity of the ontologies in a vector space model and then aggregates them using an adaptive method based on their harmonies, which is defined as an estimator of performance of the similarity assessment. The structural similarity between the entities is determined based on the number of different types of neighbors, while our structural matching strategy compares matched entities in the neighborhood of two entities.

RiMOM [7] is a dynamic multistage ontology matching framework that uses both lexical and structural information of ontologies to compute the similarity between entities. To adaptively combine multiple matching strategies, RiMOM estimates the similarity characteristics of each matching task by computing two factors: the label similarity factor and the structural similarity factor. Our approach does not need to determine whether structural information should be used because our structural matching strategy will not find additional matching results if the structural information is not important.

UFOme [25] is an ontology matching software framework that has been designed to aid expert and non-expert users in designing matching systems. UFOme provides a library of matchers and a strategy prediction module that suggests individual matchers that should be used in a specific task. The strategy prediction module computes three coefficients of two ontologies: the lexical affinity coefficient, the structural affinity coefficient, and the exploiting affinity coefficient. Based on these coefficients, UFOme suggests the optimal weights for different matchers. The strategy prediction in UFOme is similar to RiMOMs weighting strategy. In our approach, the results of different matchers are combined by the voting-based combination method instead of the weighting method. Lexical and structural information is used in two steps in our approach; therefore, different types of information are combined in a more natural way.

Falcon-AO [8][36] is a similarity-based ontology matching system. It employs three matching strategies: V-Doc, I-Sub and GMO. V-Doc builds a virtual document for each entity and then measures their similarity in a vector space model. I-Sub computes the similarities between strings attached to different entities. GMO explores the structural similarity based on a bipartite graph. Falcon-AO also uses a PBM ontology partitioner that accelerates the matching process. Falcon-AO combines lexical and structural information and uses three heuristic rules to integrate the results of three matching strategies. Both V-Doc and GMO use structural information. In contrast to Falcon-AO, our approach uses lexical and structural information in two separate steps and employs the voting-based combination method to aggregate the results.

FOMA [37] achieves high-quality results by using a matching learning method to draw classification rules for combining the results of different similarity metrics. Duan et al. [13] proposed an iterative supervised-learning approach to determine the weights of each matching strategy and the degree to which the information should be propagated to their neighbors. Our approach is unsupervised; therefore, it does not require training examples. Nevertheless, it still achieves good performance in various matching tasks.

IF-Map [38] identifies mappings automatically based on the theory of information flow. It

exploits both schema and instance information to match two ontologies. It first examines their instances to determine whether they can be assigned to concepts in a reference ontology and then uses formal concept analysis to derive an alignment. Our approach uses schema and instance information in a different way. It first finds matching correspondences between ontologies based on different lexical information and then propagates the equivalent relationships among class, property and instance pairs.

### *5.2. Instance Matching*

Silk [10] is a link discovery engine that automatically finds RDF links between datasets. Users must specify which type of RDF links should be discovered between the data sources, as well as which conditions the data items must fulfill to be interlinked. These link conditions can apply different similarity metrics to multiple properties of an entity or related entities that are addressed using a path-based selector language. The resulting similarity scores can be weighted and combined using various similarity aggregation functions. Silk accesses data sources via the SPARQL protocol and can thus be used to discover links between local or remote data sources. Silk provides a library of similarity metrics for comparing different types of information between entities. It also has several similarity combination methods, such as Average, Max, Min and Product. Users should select proper similarity metrics and combination strategies before running Silk. Our approach requires a relatively simple configuration to obtain good results; users only need to specify the property pairs within which comparisons should be made. The voting-based combination method and structural matching strategy can ensure high-quality results.

idMesh [39] is a graph-based algorithm for online entity disambiguation based on a probabilistic graph analysis of declarative links that relate pairs of entities. idMesh derives a factor graph from the entities and the source graphs to retrieve equivalent entities. It first defines a constraint satisfaction factor graph by taking advantage of symmetry and the transitivity of the equivalence relations. It then defines a reputation-based trust management factor graph to maintain the probabilistic trust variables attached to the different sources. Finally, idMesh connects the trust factor graphs to the constraint satisfaction factor graph to create an autocatalytic reinforcement process where constraint-satisfaction aids in the identification of untrustworthy sources and where trust management delivers more reasonable prior values for the link variables. idMesh works based on a given set of matching correspondences and then finds others, while our approach first uses lexical information to find an initial set of matching correspondences and then uses structural information to find additional matches. Therefore, our approach is suitable for more tasks than idMesh.

Raimond et al. [40] proposed an interlinking algorithm for automatically linking music-related datasets on the web by considering the similarities of the web resources and of their neighbors. Their algorithm provides an online linking function based on accessing data through a SPARQL end-point. Jaffri et al. [41] investigated DBLP and DBpedia and found that a large percentage of entities were either conflated or incorrectly linked. They presented several possible solutions for addressing the issue of co-referencing on the Semantic Web, such as the ReSIST and Okkam projects. Rowe et al. [42] presented a methodology for generating this background knowledge by exporting data from multiple Web 2.0 platforms as RDF data models and combining these models for use as seed data. They described two disambiguation techniques: the first uses a semi-supervised machine learning technique known as Self-training, and the second uses a graph-based technique known as Random Walks. They also explained how the semantics of data support the intrinsic functionalities of these techniques. Nikolov et al. [11] presented a data integration architecture called KnoFuss and proposed a component-based approach that allows

flexible selection and tuning of the methods and takes the ontological schemata into account to improve the reusability of the methods.

### *5.3. Summary*

Existing approaches focus on either ontology schema matching or instance matching, and there lacks an approach that performs well on both problems. Systems discussed in Section 5.1 mainly focus on matching schema entities while approaches mentioned in Section 5.2 are proposed specifically to matching instances. Most of the matching strategies and aggregation methods in these approaches are designed for one type of matching task. Some aggregation methods such as AVG, MAX and MIN are used in both ontology schema matching and instance matching approaches (e.g. COMA, SILK), but they are too primitive to guarantee good results, as shown in the experimental analysis in Section 4. In our approach, the matching strategies for both schema entities and instances are all defined on the same similarity metrics; the voting-based aggregation method works well on both ontology schema matching and instance matching problems. Compared to the existing work, our approach provides a unified and effective framework for matching both schema entities and instances.

## 6. Conclusion

In this paper, we propose a unified approach for ontology schema matching and instance matching. Our approach first identifies an initial set of matching correspondences by combining multiple lexical matching strategies using a novel voting-based combination method. Instead of aggregating different similarities, the voting-based combination method merges matching results that are independently extracted from individual matching strategies, which ensures accurate matching results. Then, our approach uses a structural matching strategy to iteratively find additional matching correspondences based on previously discovered ones. The experimental results show that the voting-based strategy combination method achieves high precision and outperforms the four baseline methods (MAX, MIN, AVG and SIG) in terms of F1-Measure; it is also observed in the experiments that the structural matching strategy can effectively improve the recall of the matching results while maintain high precision. The evaluation on the OAEI and LOD datasets validates the effectiveness of our approach in both ontology schema matching and instance matching problems.

## 7. Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 661035004, 61202246, 60973102), the National Basic Research Program of China (No. 2007CB310803), and the THU-NUS Next Research Center.

## References

[1] C. Bizer, T. Heath, T. Berners-Lee, Linked data - the story so far, International Journal on Semantic Web and Information Systems 5 (2009) 1–22.

[2] B. Aleman-Meza, U. Bojars, H. Boley, J. Breslin, M. Mochol, L. Nixon, A. Polleres, A. Zhdanova, Combining rdf vocabularies for expert finding, in: Proceedings of the 4th European conference on The Semantic Web (ESWC '07), volume 4519, pp. 235–250.

[3] O. Hassanzadeh, L. Lim, A. Kementsietsidis, M. Wang, A declarative framework for semantic link discovery over relational data, in: Proceedings of the 18th international conference on World wide web(WWW '09), pp. 1101–1102.
[4] Y. Kalfoglou, M. Schorlemmer, Ontology mapping: the state of the art, The Knowledge Engineering Review 18 (2003) 1–31.
[5] N. Choi, I.-Y. Song, H. Han, A survey on ontology mapping, ACM SIGMOD Record 35 (2006) 34–41.
[6] S. Pavel, J. Euzenat, Ontology matching: State of the art and future challenges, IEEE Transactions on Knowledge and Data Engineering PP (2011).
[7] J. Li, J. Tang, Y. Li, Q. Luo, Rimom: A dynamic multistrategy ontology alignment framework, IEEE Transactions on Knowledge and Data Engineering 21 (2009) 1218–1232.
[8] W. Hu, Y. Qu, Falcon-ao: A practical ontology matching system, Web Semantics: Science, Services and Agents on the World Wide Web 6 (2008) 237 – 239.
[9] Y. R. Jean-Mary, E. P. Shironoshita, M. R. Kabuka, Ontology matching with semantic verification, Web Semantics: Science, Services and Agents on the World Wide Web 7 (2009) 235 – 251.
[10] J. Volz, C. Bizer, M. Gaedke, G. Kobilarov, Discovering and maintaining links on the web of data, in: Proceedings of the 8th International Semantic Web Conference (ISWC '09), pp. 650–665.
[11] A. Nikolov, V. S. Uren, E. Motta, A. N. D. Roeck, Handling instance coreferencing in the knofuss architecture, in: Proceedings of CEUR Workshop, volume 422.
[12] E. Peukert, S. Maßmann, K. König, Comparing similarity combination methods for schema matching, in: Proceedings of Information Integration in Service Architectures Workshop.
[13] S. Duan, A. Fokoue, K. Srinivas, One size does not fit all: Customizing ontology alignment using user feedback, in: Proceedings of the 9th international semantic web conference on The semantic web (ISWC'2010), volume 6496, pp. 177–192.
[14] V. Nebot, R. Berlanga, Finding association rules in semantic web data, Knowledge-Based Systems 25 (2012) 51 – 62.
[15] T. Berners-Lee, Linked data - design issues, http://www.w3.org/DesignIssues/LinkedData.html, 2006.
[16] R. Studer, V. Benjamins, D. Fensel, Knowledge engineering: Principles and methods, Data & Knowledge Engineering 25 (1998) 161 – 197.
[17] N. Guarino, D. Oberle, S. Staab, What is an ontology?, in: Handbook on Ontologies, Springer Berlin Heidelberg, 2009, pp. 1–17.
[18] Owl web ontology language, http://www.w3.org/TR/owl-features/ (2004).
[19] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, K. Wang, Using bayesian decision for ontology mapping, Web Semantics: Science, Services and Agents on the World Wide Web 4 (2006) 243–262.
[20] G. Salton, C.-S. Yang, On the specification of term values in automatic indexing, Journal of Documentation 29 (1973) 351–372.
[21] C. Meilicke, H. Stuckenschmidt, Analyzing mapping extraction approaches, in: Proceedings of the ISWC 2007 Workshop on Ontology Matching.
[22] C. A. R. Hoare, Quicksort, The Computer Journal 5 (1962) 10–16.
[23] Oaei 2010, in: http://oaei.ontologymatching.org/2010/benchmarks/.
[24] A. P. S. K. V. Prateek Jain, Pascal Hitzler, P. Z. Yeh, Ontology alignment for linked open data, in: Proceedings of the 8th International Semantic Web Conference (ISWC '10).
[25] P. Giuseppe, D. Talia, Ufome: An ontology mapping system with strategy prediction capabilities, Data & Knowledge Engineering 69 (2010) 444–471.
[26] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: Proceedings of 18th International Conference on Data Engineering (ICDE'02), pp. 117 –128.
[27] W. Hu, N. Jian, Y. Qu, Y. Wang, Gmo: A graph matching for ontologies, in: K-Cap 2005 Workshop on Integrating Ontologies 2005, pp. 43–50.
[28] M. Ehrig, S. Staab, Y. Sure, Bootstrapping ontology alignment methods with apfel, in: Proceedings of WWW '05 Special interest tracks and posters of the 14th international conference on World Wide Web, volume 3729, pp. 186–200.
[29] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, A. Halevy, Learning to match ontologies on the semantic web, The VLDB Journal 12 (2003) 303–319.
[30] P. Mitra, N. Noy, A. Jaiswal, Omen: A probabilistic ontology mapping tool, in: Proceedings of the 4th international conference on The Semantic Web (ISWC'05), volume 3729, pp. 537–547.
[31] S. Albagli, R. Ben-Eliyahu-Zohary, S. E. Shimony, Markov network based ontology matching, Journal of Computer and System Sciences 78 (2012) 105 – 118.
[32] M. Niepert, C. Meilicke, H. Stuckenschmidt, A probabilistic-logical framework for ontology matching, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI'10), pp. 1413–1418.

[33] E. Jiménez-Ruiz, B. C. Grau, Logmap: Logic-based and scalable ontology matching, in: Proceedings of the 10th international conference on The semantic web (ISWC'11), volume Part I, pp. 273–288.
[34] H.-H. Do, E. Rahm, Coma: a system for flexible combination of schema matching approaches, in: Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02), pp. 610–621.
[35] M. Mao, Y. Peng, M. Spring, An adaptive ontology mapping approach with neural network based constraint satisfaction, Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 14 – 25.
[36] W. Hu, Y. Qu, G. Cheng, Matching large ontologies: A divide-and-conquer approach, Data & Knowledge Engineering 67 (2008) 140–160.
[37] M. Ehrig, Foam - framework for ontology alignment and mapping; results of the ontology alignment initiative, in: Proceedings of Integrating Ontologies Workshop.
[38] Y. Kalfoglou, M. Schorlemmer, If-map: An ontology-mapping method based on information-flow theory, in: Journal on Data Semantics I, volume 2800, Springer Berlin / Heidelberg, 2003, pp. 98–127.
[39] P. Cudre-Mauroux, P. Haghani, M. Jost, K. Aberer, H. De Meer, idmesh: graph-based disambiguation of linked data, in: Proceedings of the 18th international conference on World wide web (WWW'09), pp. 591–600.
[40] Y. Raimond, C. Sutton, M. S, Automatic interlinking of music datasets on the semantic web, in: Proceedings of Linked Data on the Web workshop (LDOW2008).
[41] I. M. Afraz Jaffri, Hugh Glaser, Uri disambiguation in the context of linked data, in: Proceedings of Linked Data on the Web workshop (LDOW2008).
[42] F. C. Matthew Rowe, Disambiguating identity web references using web 2.0 data and semantics, Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 125 – 142.