OMI-DL: An Ontology Matching Framework

Xiulei Liu, Payam Barnaghi, *Member IEEE,* Bo Cheng, Li Wan, Yixian Yang

***Abstract*—This paper focuses on matching ontologies created for similar domains through different sources. Different solutions use lexical, structural or logical processing and analysis to match ontologies. However, an important aspect is also interpreting concepts that entities are presented with and using them in relation to semantics in an ontology. The paper demonstrates analyzing and extending the concepts used to define entities in an ontology, discusses establishing and filtering matching candidates by reasoners, and then describes constructing correspondences between entities from different ontologies using lexical and semantic analysis. The experiments show that our prototype, called OMI-DL, is among the top group over many ontologies adopted from the OAEI benchmark dataset. We also provide an evaluation of the OMI-DL method for matching the DOLCE+DnS Ultralite ontology and the domain ontology developed in the m:Ciudad project.**

***Index Terms*—Ontology Matching, Lexical Analysis, Semantic Analysis, Reasoners**

I. Introduction

ONTOLOGY as an explicit specification of a conceptual model is becoming more and more popular in many areas such as service composition, Semantic Web, and knowledge and data engineering [1], [2], [3]. An ontology provides a structured knowledge representation and makes it sharable and reusable for machines, software agents and human users. With growing interest and applications, an increasing number of ontologies are constructed to describe domain knowledge in different areas. Most of these ontologies are constructed and maintained by different knowledge engineers who have various backgrounds and use different terminologies to describe concepts. This leads to the construction of several ontologies with different terminologies for similar (or even the same) domains. The heterogeneity among different ontologies for representing similar domains limits interoperability across the ontologies. Ontology matching methods deal with this issue [4].

There are various methods which are used to match two ontologies. Some of the recent works are reviewed in [5], [6]. The existing techniques mostly focus on element-level (i.e. entity label attributes), structure-level (i.e. taxonomy structures) and instances information to match ontologies (see [7], [8], [9] for more details). Some of the existing methods (such as [10], [11], [12], [13]) utilize lexical and semantic analysis to match ontologies. At the lexical analysis stage, the words in annotation axioms (such as Label Annotation Axioms and Comment Annotation Axioms) are analyzed using string similarity [11], [12] or deriving the senses of words from external resources [10], [11], [12]. At the semantic analysis stage, alignments are directly inferred based on the results from the lexical analysis stage [10], [13], or similarity measures are calculated between entities using results by submitting a constant number of rules into a reasoner [12], [14].

We propose a comprehensive framework for ontology matching which is called OMI-DL. At the lexical analysis stage, OMI-DL not only analyzes the suitable senses of words, but also extends them by utilizing conceptual-semantics and lexical relations. This helps to include the possible senses of a word in some existing contexts instead of only finding the most suitable representation and supports identification of potential relations between entities. OMI-DL then formally defines the representation of an entity notion for describing the lexical information of an entity. With semantic elements (such as $\sqcup$ and $\sqcap$), the definition not only combines the suitable senses and their extensions from the words in entity label and comment, but also imports the lexical information of the concepts and the logics in the domain and range of an entity property. The definition implies more information than other similar solutions especially when only entity label and comment cannot provide enough information. It also conforms more to semantics described in domain and range of an entity property in contrast to the methods that only calculate similarity measures.

At the semantic analysis stage, OMI-DL not only establishes matching candidates between entities from different ontologies based on the representation of an entity notion by reasoners, but also proposes a semantic iterative filtering process to remove redundant matching candidates based rules, and prove that this is terminable. The filtering process ensures that semantics in integration ontology are consistent to the ones in matched ontology when the integration ontology is established by assembling all of axioms in the matched ontologies and the alignment together. OMI-DL then calculates similarity measures based on filtered matching candidates by reasoners to compute correspondences between entities from different ontologies and also defines the closest subsumer of an entity description based on semantics in both matched ontologies and alignment. This produces supplementary correspondences which do not appear in alignments derived from most of other methodsand ensures the consistency of the derived alignment.

The rest of this paper is organized as follows: Section II describes background information; Section III presents related work; Section IV discusses the OMI-DL framework; Section V explains extensions of the suitable senses and the representation of an entity notion; Section VI demonstrates construction of matching candidates between entities and describes the filtering process; Section VII presents the generation of correspondences; Section VIII demonstrates an evaluation; Section IX concludes the paper and discusses the future work.

Xiulei Liu, Li Wan, Bo Cheng and Yixian Yang are with National Engineering Laboratory for Disaster Backup and Recovery, Beijing University of Posts and Telecommunications, 100876, P.R. China.

Payam Barnaghi is with Centre for Communicastion Systems Research, University of Surrey, Guildford, Surrey, GU2 7XH, UK.

II. Background

The Web Ontology Language (OWL) benefits from well established background and years of Description Logics(DL) research, which is currently a popular ontology representation language. This paper focuses on matching ontologies which are represented in the **OWL** form.

An OWL ontology can be interpreted as knowledge-base semantics (see [15] for more details). The most important aspect for semantics is to determine relations between concepts and between properties such as *include* $\sqsupseteq$, *beIncluded* $\sqsubseteq$, *disjoint* $\perp$ and *equivalent* $\equiv$. In general, they can be processed by **reasoners** such as Pellet, FaCT++ and Racer.

An OWL ontology has three types of entities: concepts, individuals, and object and data properties. Concepts in an OWL ontology can be also specified by three types: defined concepts (name symbols), primitive concepts (base symbols) and anonymous concepts. In this paper, we abstract both data properties and object properties as roles (i.e. properties). An entity label attribute in an OWL ontology is normally defined by knowledge engineers and specified in words, and denotes the value of the *rdfs:label* property.

To match entities from different ontologies, a main source of difficulty is that ontologies are designed with various background knowledge and often in different contexts. This background knowledge does not usually become a part of the ontology specification and thus is not available to matchers [6], [16]. Lack of the background knowledge increases the difficulty of matching tasks as it generates too many ambiguities in interpreting and analyzing senses. Different strategies have been used to tackle this problem based on syntactic and structural analysis [5], [10], [17]. We discuss the extensions of terminologies to describe entities by using WordNet in this work. WordNet groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms which are called "***synsets***". Each *synset* expresses a distinct concept and provides a short and also general definition related to this concept. A *synset* is uniquely identified by an address number called an **offset**. A *synset* also contains a group of synonym words or collocations that form a specific sense, and provides a **usage count** for indicating how often a word appears in this specific sense. For example, in *[Synset: [Offset: 5780748][2][POS: noun] Words: abstraction, abstract – (a concept or idea not associated with any specific instance; "he loved her only in the abstract– not in person")]*, 5780748 is the **offset** and the collocation is *abstraction* and *abstract*. Various senses of a word are described in different *synsets*. *Synsets* are also interlinked by means of conceptual-semantic and lexical relations. The relations are defined as: *also_see, antonym, attribute, cause, and son on*.

III. Related Work

Due to the growth of the Semantic Web and the employment of ontology-based methods in numerous applications, ontology matching has become a focus of research community in recent years. The works presented in [5], [6] provide a comprehensive review of the current approaches and classify them from different views. According to these surveys, there are five kinds of basic data used by most of the ontology matching methods: lexical (or linguistic) information, structural information, semantic information, external resource information and individual information.

The different methods employ various techniques to use one or more kinds of the information above. Some of the techniques include similarity flooding [18], [19], [20], coefficient computation [11], [21], graph matching [22], [23], formal concept analysis [24], machine learning [25], [26], Bayesian decision theory [7], hybrid methods [8], [27], [28], Markov networks [9], optimization techniques [29], [30], or reasoners [10], [11], [12], [10], [31].

Many solutions analyze entity label and entity comment attributes in an ontology and use external data resources to obtain lexical (or linguistic) information; such as UFOme [8], ILIADS [12], ASMOV [11], Lily [23] and MapPSO [29]. Most of the works above generally obtain the usage count of a synset, relations (such as hypernym) or other sets in WordNet to calculate lexical (or linguistic) similarity measures between entities. The values of these similarity measures usually vary between [0, 1]. OMI-DL and S-Match [10] combine the suitable sense of a word with semantic elements to define an entity notion and then use WordNet as a knowledge base to infer semantic relations between entities by utilizing reasoners as shown Section V-D.

However, in practical scenarios, finding the suitable sense of a word in natural language processing by current techniques is still a challenging task (as shown in the example in Section V-B). So OMI-DL not only obtains the suitable sense of a word, it also uses **synset extensions** to overcome the limitations of entity label processing. This helps to include the possible senses of a word in existing contexts instead of only finding the most suitable representation, and supports the identification of potential relations between entities. This process is discussed in Section V-B. Most of the works above do not support the extensions of the suitable senses to compute lexical (or linguistic) similarity measures.

The method to define **an entity notion** in OMI-DL is also different from similar solutions such as S-Match. S-Match defines an entity notion by intersecting (or uniting) the suitable sense of each word only in that entity **label** attribute. In addition to adopting the suitable sense and its extensions in entity label and entity **comment** attributes, OMI-DL extends this method by transforming the representation of an entity notion into a combination statement as discussed in Section V-D. This leads to identification of a set of intersections (or unions) of senses in an entity notion in OMI-DL. This feature increases the possibility of finding more correspondences. Concepts and logics in the domain and range of a property also provide important information to match the properties as explained in Section V-D. S-Match does not analyze the property data, and a few of the methods above only use it as one of the attributes for computing similarity measures. OMI-DL imports the logics and the lexical information of the concepts in the domain and range of a property into the representation of an entity notion as shown in Section V-D. So an entity notion in OMI-DL implies more information than S-Match and other similar solutions especially when only property label and property comment attributes cannot provide enough information. The OMI-DL method also conforms more to semantics described in domain and range of a property in contrast to the methods that only calculate similarity measures.

There are various methods which utilize structural information for matching ontologies. For example, Lily [23] utilizes semantic sub-graphs to represent structural information; ILIADS [12], CSR [25], SSID [32] and iMatch [9] use Jaccard distances, classification-based learning techniques and Markov networks to process structural information respectively; S-Match defines the concept of a node whose logic expression is computed as the intersection of the concepts of all labels from the root node to the node itself; ASMOV [11] computes similarity measures based on structural elements from different entities; OMI-DL uses reasoning results to process structural information.

Most of the methods based on semantic information use reasoning results. They go one step further than the methods that only use lexical (or linguistic) and structural information to calculate similarity measures. The semantic analysis in ontology matching can be divided into two main categories according to deductions: Propositional Satisfiability problem (SAT) and Description Logics (DL) reasoning.

SAT deciders take the Conjunctive Normal Form (CNF) formulas as input [10]. This means that the results obtained based on CNF (in methods such as S-Match) are not used to match OWL ontologies. In general, DL reasoning adopts tableau algorithms which use negation to reduce subsumptions to (un)satisfiability of concept descriptions. This technique, adopted by the methods such as ILIADS and OMI-DL, can process various syntaxes and exploit semantics in an OWL ontology. Reasoners in ILIADS execute only a constant number of axioms (N) that are extracted from a set of rules by heuristic methods. The results obtained by inferring the axioms are used to compute a logical inference similarity measure between entities. So ILIADS cannot check whether all restrictions in ontologies are consistent or not. OMI-DL ensures this by directly using reasoners over all axioms.

OMI-DL also describes **a semantic iterative filtering process** to remove redundant matching candidates by reasoners; because the sense extension and representation of an entity notion lead to inaccuracy when identifying matching candidates. The rules in the filtering process and the format of the matching candidates in OMI-DL are different from those in ASMOV as described in Section VI-B. The verification (or filtering) process in ASMOV mainly focuses on semantic relations in the matching candidates according to similarity values. The filtering process in OMI-DL puts more emphasis on combining semantic relations in the matching candidates together with semantics in matched ontologies.

For example, assuming *101* and *302* are references to sample ontologies as shown in Section IV, in ASMOV, given *<101:Report, 302:Publication, value: 0.36>* and *<101:Manual, 302:Publication, value:0.24>*, if there exists *<101:Report, 101:Manual*, $\equiv$> in *Ontology101*, the two candidates can define a match, OTHERWISE one of them is removed; because there is only $\equiv$ relation between entities whatever the *values* are. Given *<101:Report, 302:Publication,* $\sqsubseteq$> and *<101:Manual, 302:Publication,* $\sqsubseteq$>, OMI-DL first evaluates whether *Ontology101* and *Ontology302* with the two matching candidates are consistent. Then, it computes whether they lead to produce new links between the matched ontologies. The new links do not exist in results obtained by inferring the matched ontologies and a set of matching candidates. If such new links are not produced, the two matching candidates can define a match even if *<101:Report, 101:Manual*, $\equiv$> does not exist in *Ontology101*. This concept is discussed in more details in Section VI-B.

The filtering process in OMI-DL is more efficient compared to the verification process in ASMOV. It increases the precision of the system as shown in Fig. 3 in Section VIII-B1. Other methods discussed above mostly compare a threshold value with similarity values to filter matching candidates.

IV. Ontology Matching Framework

This section describes the OMI-DL framework. The main workflow, illustrated in Fig. 1, is divided into three main stages (separated by the dashed black line): analyzing an entity notion, identifying and filtering matching candidates, and aligning entities. The system receives two ontologies as input, and outputs a set of correspondences (called **an alignment**). The sample ontologies (*Ontology101* and *Ontology302*) are taken from Folders 101 and 302 in the Ontology Alignment Evaluation Initiative (OAEI) benchmark dataset[1]. This paper uses *101 (OR 302): <an entity label attribute>* to refer to entities in the ontologies.

The entity notion analysis stage (shown on the left in Fig. 1) automatically finds the suitable sense of each word described in natural language labels and comments taken from an ontology, extends the suitable sense and then defines the representation of an entity notion. The extensions of the suitable senses support the identification of potential relations between entities. The method of representing an entity notion facilitates inferring relations between entities by using more descriptive information for each concept.

The Tokenizer and Synset Finder components shown in Fig. 1 are similar to Tokenization and elementLevelSenseFiltering described in [10]. We use the Synset Finder component to find the suitable sense of each word and then utilize the Synset Extension component to address the limitation of getting only one sense for each word.

When deriving the suitable sense and its extensions for each word, we first construct the Synset Relation Ontology. We then define an entity notion to express the concept which this entity label and entity comment attributes imply. While inferring, if two entities from different ontologies are identified as a matching candidate, we need to obtain the relations between synsets that are used to define an entity notion. The relations between synsets will be obtained by reasoning over the Synset Relation Ontology.

After defining an entity notion and constructing the Synset Relation Ontology, the following stage (shown in the middle in Fig. 1) will identify and filter matching candidates. The Matching Candidates Generator component determines whether two

[1]http://oaei.ontologymatching.org/2009/benchmarks

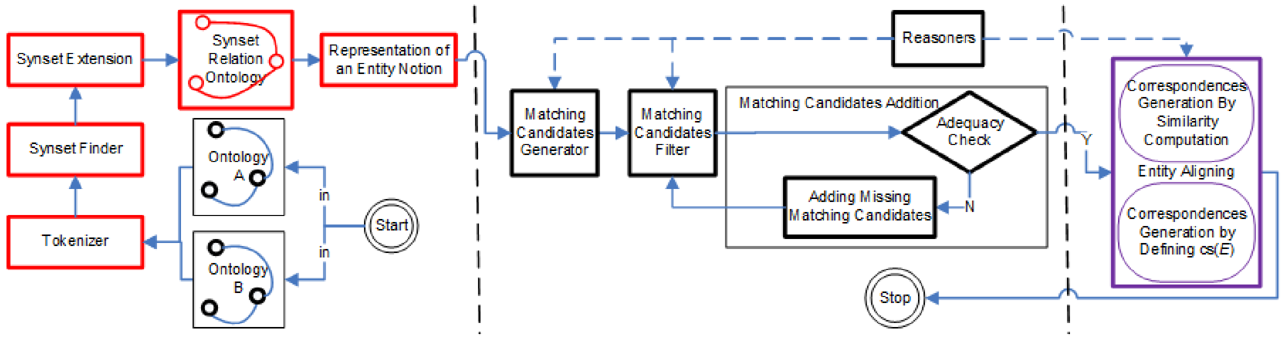

Synset Extension
Synset Relation Ontology
Representation of an Entity Notion
Synset Finder
Tokenizer
Ontology A
Ontology B
in
in
Start
Matching Candidates Generator
Matching Candidates Filter
Reasoners
Matching Candidates Addition
Adequacy Check
Adding Missing Matching Candidates
N
Y
Correspondences Generation By Similarity Computation
Entity Aligning
Correspondences Generation by Defining cs(E)
Stop

Fig. 1: Overview of ontology matching process

entity notions from different ontologies have any semantic relations by using a reasoner (the dashed black connection line as shown in Fig. 1). This process results in a set of matching candidates which cover most of the correspondences in the reference alignments. Collecting all possible matching candidates ensures a high recall for the system.

However, not all of the matching candidates derived from the previous steps are accurate. The cases are illustrated in Table II by a partial view of the matching candidates for the two sample ontologies. The next step which is an iterative process filters these matching candidates and removes redundant matching candidates by using a reasoner. This process increases the precision of the system.

To perform the task, we first filter out redundant matching candidates with the help of a reasoner (the dashed black connection line as shown in Fig. 1) in the Matching Candidates Filter component based on the rules defined in Section VI-B. We then check if some matching candidates are overlooked in the filtering process. The checking is performed according to an adequacy condition defined in Section VI-B. If so, the Matching Candidates Filter component will filter the overlooked matching candidates.

The entity aligning stage (shown on the right in Fig. 1) generates correspondences by two methods. The first method computes similarity values between all possible pairs of entities (one from each ontology) by considering the filtered matching candidates, semantic information in an ontology, and string similarity. It then selects correspondences by comparing the similarity values with a heuristically defined threshold. The second method infers supplementary correspondences according to the closest subsumer of an entity description which is implemented with the help of reasoners (the dashed black connection line as shown in Fig. 1). The supplementary correspondences satisfy semantics in the matched ontologies and also semantics in the correspondences generated by the previous step. This improves the recall and precision of inferred alignments which have been discussed in Sections VIII-B1 and VIII-C.

V. Analysis of Entity Notions

This section introduces the formal representation of an entity notion. An entity notion is composed of the suitable sense and its extensions of each word describing an entity. So OMI-DL firstly analyzes and extends the suitable sense of a word; secondly it determines relations between the extracted synsets from different ontologies; finally it formally defines an entity notion. The suitable sense extension and representation of an entity notion help to identify more matching candidates. This improves the recall of the system.

In descriptions from entities, **stop words** such as "*am, is, are, have, has, string*" are omitted. However, special words such as "*or, without, no, else*", denoted as "**SpWdSet**", are processed (see Section V-D for more details).

A. *Analyzing Synsets*

A word might have a different sense in different contexts. So before aligning entities, the Synset Finder component automatically finds the suitable synset (sense) of each word taken from entity label and entity comment attributes in an ontology. It has two sub-processes: Finding Related Synsets and Selecting Suitable Synsets.

The Finding Related Synsets sub-process is inspired by the element-level-sense-filtering technique used in [10]. The key idea is that we check whether any two synsets from two different words in an ontology are connected by any relation in WordNet. If so, we put each synset into a set of the **related synsets** of the word to which it is related.

To choose the suitable synsets, the Selecting Suitable Synsets sub-process first calculates the critical value for each synset in a set of the related synsets of a word as shown below.

$CriticalValueOfSynsetA = word.countForSynsetA * \varphi + usageCountOfSynsetA$

in which *SynsetA* is a synset in a set of the related synsets of the word; *word.countForSynsetA* is the number of times that *SynsetA* is selected as the related synset; *usageCountOfSynsetA* is the usage count of *SynsetA* which is obtained from WordNet; φ is defined experimentally. In our experiments, based on heuristics we found that $\varphi = 6$ maximizes the number of words which have correct synsets.

The Selecting Suitable Synsets sub-process then chooses a synset in a set of the related synsets of a word which has the highest critical value as the **suitable synset** of this word.

However, WordNet does not cover all words in entity label and comment attributes (such as "*url*" discussed in Section VIII-C). Consequently these words will not have the suitable synsets. WordNet has also some omissions as shown in [33].

TABLE I: Axiom conversion rules

WordNet Relation	Symbol	ClassAxiom
Hypernym	$\sqsupseteq$	SupClass Axiom
Hyponym	$\sqsubseteq$	SubClass Axiom
Holonym	$\sqsupseteq$	SupClass Axiom
Antonym	$\perp$	DisjointClass Axiom
Similar to	$\equiv$	EquivalentClass Axiom
Coordinate	$\perp$	DisjointClass Axiom

B. WordNet Synset Extension

The Synset Finder component uses a statistical approach to select the most frequently used synset of a word with the information from WordNet. However, in practical scenarios, using natural language processing to obtain the most suitable synset of a word in some domains is a challenging task.

Defining a suitable synset does not mean that we only use one synset for each word. There could be more than one synset to be used because we cannot determine which synset of a word is best according to the data derived from WordNet and matched ontologies. Knowledge engineers may use different terminologies to construct similar concepts in different ontologies. Given the two statements <*Book, publishedBy, IEEE*> and <*Book, hasPublisher, IEEE*>, here *published* and *publisher* have the same role but there is no relation between the synsets for *published* and *publisher*. The Synset Extension component shown in Fig. 1 addresses this issue.

The Synset Extension component defines a set of **extended synsets** of a word which have special relations (including *pertainym, derived_from_adj, derivationally, related*) with the suitable synset of this word in WordNet. This helps to include different possible synsets of a word in different contexts instead of only finding one suitable synset. Although this function does not guarantee that the best representative concepts for words will be found, it broadens the scope of the synset representation for the words. In our experiments, the extended synsets of *published* contain the suitable synset of *publisher*; and vice versa.

The extended synsets include terminologies that support the identification of potential relations between entity pairs. This will improve the recall of the system. However, as in classic information retrieval scenarios, higher recall could result in lower precision [34]. For example, the extended synsets of the word *published* in the property *101:howPublished* contain *publisher, publishing house, publishing firm, publishing company – (a firm in the publishing business)*, denoted as *synsetP*, and the suitable synset of the word *organization* in the property *302:organization* is *organization, organisation, system – (an ordered manner; orderliness by virtue of being methodical and well organized)*, denoted as *synsetO*. Because there is *holonym_mem* relation between *synsetO* and *synsetP* in WordNet, this leads to the definition of a *beIncluded* relation between *101:howPublished* and *302:organization* using our method as shown in Table II.

C. Construction of Synset Relations

After determining the suitable synsets and their extensions of all words in ontologies, we define semantic relations (if there are any) between any two synsets, one from each ontology, and import the relations into the Synset Relation Ontology. This ontology is composed of the defined semantic relations and the extracted synsets. It is also the knowledge-base for the matching candidates inferring process described in Section VI-A.

For establishing the Synset Relation Ontology, we first need to infer the relations between the synsets respectively from different ontologies, including the suitable synset and its extensions of a word. The relations including *Hypernym*, *Hyponym*, *Holonym*, *Antonym*, *Similar to* and *Coordinate*, are defined in WordNet.

While finding the relations between two synsets, we convert these relations into axioms according to the rules shown in Table I and then add the axioms into the Synset Relation Ontology. For example, given the synset *A [Synset: [Offset: 32311][610][POS: noun] Words: school – (an educational institution;)]* for the word *School*, and another synset *B [Synset: [Offset: 42323][26][POS: noun] Words: institution, establishment – (an organization founded and united for a specific purpose;)]* for the word *Institution*, if there exists a *Hypernym(B,A)* relation in WordNet (which means that *B* is Hypernym of *A*), we translate this relation into *SubClassAxiom (School32311, Institution42323)* and then add this axiom into the Synset Relation Ontology.

We also represent a set of the **informative synsets** of each word which are the basis for defining an entity notion when constructing the Synset Relation Ontology. Each synset in this set is from the suitable synset or its extensions and has some relation with another synset from the other ontology. It is probable that a set of the informative synsets of a word do not include all of the suitable synset and its extensions of this word and could only include a part of them.

From the process of establishing the Synset Relation Ontology, it can be seen that the informative synsets of words map the concepts in this ontology and the semantic relations between the obtained synsets map the axioms in this ontology. This ontology only includes concepts and does not specify properties. The Synset Relation Ontology, when matching *Ontology101* and *Ontology302*, includes 907 concepts, 1202 subClass axioms and 125 equivalentClass axioms.

D. Representation of an Entity Notion

This section formally define the representation of an entity notion by combining the suitable senses and their extensions with semantic elements and logics in the domain and range of a property. It presents more information and also conforms more to semantics described in domain and range of an entity property.

It is denoted as **EN**(E) in which E is an entity. **EN**(E) is a set in which every element is the OWL Object Intersection of several synsets from the Synset Relation Ontology. The Synset Relation Ontology is composed of a part of the WordNet synsets and the relations among them, which are extracted according to the data in ontologies as shown in V-C. So only a subset of the WordNet synsets are related to an entity notion.

We define **W**(*E*) below before describing an entity notion.

$$\mathbf{W}(E) = \{e|e = La(E) \bigotimes (\Sigma_n^i \biguplus C_n^i(Co(E)))\} \quad (1)$$

1. $0 \le i \le n$
2. $n = Co(E).size$

In Formula 1, *La(E)* is a set of words in the Label Annotation Axiom of an entity, and *Co(E)* is a set of words in the Comment Annotation Axiom of an entity, where the stop words are deleted. We take as an example the concept *101:Book* shown in the sample *Ontology101*. The label and comment of the *101:Book* concept is "*Book*' and "*A book that may be a monograph or a collection of written texts*', respecitvely. So *La(101:Book)*={*Book*} and *Co(101:Book)* = {*book, monograph, or, collection, written, text*}. $C_n^i(A)$ represents a set in which every element is composed of *i* elements selected in *n* elements of *A*. This can be represented with a combination statement. For example, C_6^1(*Co(101:Book)*) = {{*book*}, {*monograph*}, {*or*}, {*collection*}, {*written*}, {*text*}}.

$\uplus$ denotes the merging of two sets into one set. For example, C_6^1(Co(*101:Book*)) $\uplus$ C_6^6(Co(*101:Book*)) = {{*book*}, {*monograph*}, {*or*}, {*collection*}, {*written*}, {*text*}, {*book, monograph, or, collection, written, text*}}.

$\otimes$ denotes the formation of a new set in which every element includes *La(E)* and all of the elements of the set which is an element in $\Sigma_n^i \uplus C_n^i(Co)$, and repeated words are deleted. For example, *La(Co(101:Book))*$\otimes C_6^6$(*Co(101:Book)*) = {{*Book, monograph, or, collection, written, text*}}.

Each element in **W**(*E*) includes all words in the label of *E* and several words in the comment of *E*.

Based on the description of **W**(*E*), we formally define **EN**(*E*) as follows:

$$\mathbf{EN}(E) = \{s|s = s_1 \sqcap s_2 \sqcap s_i \sqcap s_{i+1} \cdots \sqcap s_n\} \quad (2)$$

1. $e \in \mathbf{W}(E)$
2. $n = e.size$
2. $0 \le i \le n$
3. $W_i \in e$
4. $s_i = \begin{cases} special\ symbol & \text{if } \mathrm{W_i} \in \text{ SpWdSet} \\ \in W_i.InformativeSynsets & \text{otherwise} \end{cases}$

In Formula 2, *e* is an element of **W**(*E*) defined in Formula 1; W_i is an element of *e*; W_i*.InformativeSynsets* is a set of the informative synsets of W_i; s_i is an element in this set when W_i does not belong to the SpWdSet, or it is an element in the SpWdSet; $\sqcap$ denotes the intersection of several synsets.

There are four attributes for an element in **EN**(*E*);

1. The element is an instance of OWL Object Intersection.
2. Operands in the element must contain one and only one synset in a set of the informative synsets of each word in the label of *E*, except when this word belongs to the SpWdSet.
3. Operands in the element may contain one synset in a set of the informative synsets of each word in the comment of *E*.
4. Operands in the element are concepts from the Synset Relation Ontology. Namely, the element is an anonymous concept in the Synset Relation Ontology.

An entity label attribute takes the key role in an entity notion, so each element in **EN**(*E*) must contain the synsets of the words in this entity label attribute. An entity comment attribute only takes the supplementary role, so each element in **EN**(*E*) may contain the synsets of the words in this entity comment attribute. This is reasonable, because when a knowledge engineer creates an ontology, s/he will select necessary words as an entity label attribute and may also further describe this entity in an entity comment attribute.

We also process words in the SpWdSet (such as "*or, without, no, else*") when defining an entity notion. For example, we will interpret *or* using $\sqcup$ in "*reference or conference*", and we will interpret *without* using $\sqcap$ and $\neg$ in "*book without report*". If s_i in Formula 2 includes symbols in the SpWdSet, we need to process it further. Given the element in **EN**(*101:Book*): {*Book, monograph, "**or**", collection*}, we first obtain {*book56434 $\sqcap$ monograph534534 $\sqcap$ "**or**" $\sqcap$ collection23244*} according to Formula 2, and then, after processing the special symbol "***or***", we replace it with {*book56434 $\sqcap$ (monograph534534 $\sqcup$ collection23244)*}.

While defining an entity notion for a property, we also consider concepts and logics in the domain and range of this property. This provides important information in some cases, e.g., in the two properties shown below:

PropertyA: <*Art **book** String*>

PropertyB: <*Book $\sqcup$ Report **title** String)*>

The bold words are a property label, the left items are a domain, and the right ones are a range. If we only use the property label attribute "*book*", it is not enough to express the notion for *PropertyA*. For a property, we use lexical information from concepts and logics in the domain and range of this property to address the limitation of only using property label and property comment attributes. The process is performed in four steps, as shown below:

1. Obtain the words in the concept labels in the domain and range of the property *P*, denoted as *D* and *R*;
2. Calculate $\Sigma_n^i \uplus$ $\mathrm{C}_n^i(D)$ and $\Sigma_n^i \uplus$ $\mathrm{C}_n^i(R)$ according to the logics in the domain and range (the logics in the range are only used for an object property);
3. Insert $\Sigma_n^i \uplus$ $\mathrm{C}_n^i(D)$ and $\Sigma_n^i \uplus$ $\mathrm{C}_n^i(R)$ into *Co(P)* in Formula 1;
4. Calculate Formulas 1 and 2 to obtain **EN**(*P*);

We take as an example *PropertyB* above to explain the steps.

a) derive *D* = {*Book, Report*}. Data types of the range of a property, such as "*String, Char*", are omitted, so *R* in *PropertyB* is an empty set.

b) calculate $\Sigma_n^i \uplus$ $\mathrm{C}_n^i(Book \sqcup Report)$,
when *i = 1*, C_2^1*(Book $\sqcup$ Report)* = {{*Book*}, {*Report*}};
when *i = 2*, C_2^2*(Book $\sqcup$ Report)* = {{*Book $\sqcup$ Report*}};
$\Sigma_n^i \uplus$ C_n^i*(Book $\sqcup$ Report)* = {{}, {*Book*}, {*Report*}, {*Book $\sqcup$ Report*}}.

c) compute *Co(P)* = {{}, {*Book*}, {*Report*}, {*Book $\sqcup$ Report*}, {*title*}}

d) to get the synset of *Book $\sqcup$ Report*, one must use the union of two synsets, a synset obtained for *Book* and another obtained for *Report*. So the synset of *Book $\sqcup$ Report* is *SynsetOfBook $\sqcup$ SynsetOfReport*. The rest of this step is similar to those explained for Formulas 2 and 1.

As shown above, the method to define an entity notion in OMI-DL is also different from similar solutions such as S-Match. S-Match defines an entity notion by intersecting (or uniting) the suitable sense of each word only in that entity label attribute. In addition to adopting the suitable sense and its extensions in entity label and entity comment attributes, OMI-DL extends this method by transforming the representation of an entity notion into a combination statement. This leads to identification of a set of intersections (or unions) of senses in an entity notion in OMI-DL. This feature increases the possibility of finding more correspondences. Concepts and logics in the domain and range of a property also provide important information to match the properties. S-Match does not analyze the property data, and a few of the methods above only use it as one of the attributes for computing similarity measures. OMI-DL imports the logics and the lexical information of the concepts in the domain and range of a property into the representation of an entity notion. So an entity notion in OMI-DL implies more information than S-Match and other similar solutions especially when only property label and property comment attributes cannot provide enough information. The OMI-DL method also conforms more to semantics described in domain and range of a property in contrast to the methods that only calculate similarity measures.

Although this method can enhance the descriptions to some extent, it leads to some inaccuracy. Given *(Academic ⊔ LecturesNotes **101:school** String)* and *(Publication **302:notes** String)*, according to the steps above, we obtain one element in **EN**(*302:school*): *((academic15216 ⊔ (lecture8932 ⊓ notes65380)) ⊓ school46212)*, and one element in **EN**(*101:notes*): *(publication65022 ⊓ notes65380)*, and we have the two axioms *(publication65022 ⊒ notes65380)* and *(notes65380 ⊒ academic15216)* in the Synset Relation Ontology. So we can infer **EN**(*101:notes*) ⊒ **EN**(*302:school*) which leads to inconsistency in the derived alignment. We filter out redundant matching candidates such as **EN**(*101:notes*) ⊒ **EN**(*302:school*) using the process described in Section VI-B.

VI. Identifying and Filtering Matching Candidates

This section describes generating matching candidates between entities by using a reasoner over the Synset Relation Ontology when deriving an entity notion, and then discusses filtering the matching candidates through an iterative process. The matching candidates are the basis for producing correspondences. The filtering process increases the precision of the system.

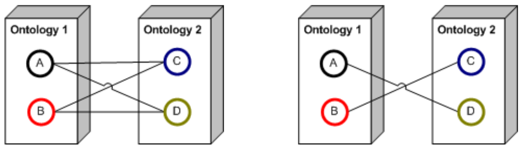
A. Identifying Matching Candidates

Once we get an entity notion, we can identify a matching candidate (**MC** for short) using a reasoner over the Synset Relation Ontology. Matching candidates express logical relations between the senses of words in the labels and comments of all possible pairs of entities, one from each ontology. It is denoted as below:

$$\mathbf{MC}(A, B) = < A, B, Relation > \quad (3)$$

TABLE II: A partial view of matching candidates for the sample ontologies

	101: Entity	302: Entity	Relation
1	101:Collection	302:Publication	beIncluded
2	101:School	302:TechReport	include
3	101:Book	302:Book	equivalent
4	101:Booklet	302:Book	beIncluded
5	101:author	302:author	equivalent
6	101:author	302:firstauhtor	include
7	101:howPublished	302:organization	beIncluded



(a) **MC**s before filtering (b) **MC**s after filtering

Fig. 2: Matching candidates (**MC**s) filtering

where A and B are the same type and are called a source entity from a source ontology denoted as ***MC.left*** and a target entity from a target ontology denoted as ***MC.right*** respectively. **MC(*A*, ⁎)** includes a set of matching candidates where a source entity in each **MC** is A. A similar symbol **MC(⁎, *A*)** is also defined. *Relation* in Formula 3 includes: *disjoint* ⊥, *include* ⊒, *beIncluded* ⊑ and *equivalent* ≡.

Given **EN**(*E1*) and **EN**(*E2*), firstly infer relations between any two elements, which are anonymous classes in the Synset Relation Ontology and are obtained from **EN**(*E1*) and **EN**(*E2*) respectively, by reasoners; secondly we record the count for each relation; finally we select the relation with the maximum count. If there is no relation between two entities, **MC** for them will not be established.

This step will produce a set of the original matching candidates, denoted as **MCO**, as partially shown in Table II for the sample ontologies.

B. Filtering Matching Candidates

Not all of the matching candidates are accurate and some of them could be redundant as shown in Sections V-B and V-D. The matching candidates also need to ensure semantic consistency in matched ontologies and themselves. Based on the above considerations, we filter the matching candidates derived from the previous step and keep the most relevant matching candidates. This process helps to increase the precision of the system. We first introduce the filtering rules and then describe the iterative filtering process.

The rules for filtering are described in the following.

1 If one entity in an ontology has more than one candidates that match to multiple entities in another ontology, these matching candidates should neither lead to inconsistency nor directly produce any new links among two matched ontologies.
2 Select the matching candidates which have stronger relations if Rule 1 is broken. Their order from stronger relations to weaker relations is ≡, ⊒, ⊑ and ⊥ respectively.

Rule 1 ensures that semantics in integration ontology are consistent to the ones in matched ontology when the integration ontology is established by assembling all of axioms in the matched ontologies and the alignment together. Rule 2 is developed by intuition and experiment.

With the help of Fig. 2, we explain the above rules. The cubes represent two matched ontologies, denoted as *Ontology 1* and *Ontology 2*. The circles represent entities, denoted as *A*, *C*, *B* and *D*. The lines between two cubes represent matching candidates.

In Fig. 2a, *A* has two matching candidates with *C* and *D*. If the matching candidates are <*A, D, equivalent*> and <*A, C, include*>, we can infer <*D, C, include*> for *Ontology 2*. If <*D, C, include*> does not exist in *Ontology 2*, this directly produces a new link which knowledge engineers have not considered in *Ontology 2*. It breaks Rule 1. According to Rule 2, <*A, D, equivalent*> is selected rather than selecting <*A, C, include*>.

If the matching candidates <*A, D, equivalent*> and <*A, C, disjoint*>, and <*D, C, include*> exist in *Ontology 2*, we can see <*D, C, disjoint*> conflicts with <*D, C, include*> in *Ontology 2*. This leads to ontology inconsistency (refer to [15] regarding inconsistency in an ontology) and breaks Rule 1; so <*A, C, disjoint*> should be deleted based on Rule 2. The filtered matching candidates for this example are depicted in Fig. 2b.

The Matching Candidates Filter component first checks if **MC**s in **MC**(A,⋇) lead to inconsistency and then deletes those matching candidates that cause conflict. After this, the matching candidates are denoted as Consistent **MC**(A,⋇). If equivalent **MC**s exist in Consistent **MC**(A,⋇), we will then directly discard non-equivalent matching candidates, and then keep only one equivalent matching candidate. Given <*A, D, equivalent*>, <*A,E, equivalent*> and <*A, C, beIncluded*>, we discard the non-equivalent matching candidate <*A, C, beIncluded*>. If <*D, C, beIncluded*> or <*E, C, beIncluded*> does not exist in *Ontology 2*, it would be discarded based on Rule 1 and 2. If <*D, C, beIncluded*> and <*E, C, beIncluded*> exist in *Ontology 2*, it would still be discarded. According to < *A, D, equivalent*> and <*D, C, beIncluded*> or < *A, E, equivalent*> and <*E, C, beIncluded*>, <*A, C, beIncluded*> will be obtained in next step. After discarding the non-equivalent matching candidates for Consistent **MC**(A,*), we filter out the equivalent matching candidates, which have a lower string similarity value, to keep only one equivalent matching candidate for Consistent **MC**(A,*). For this, we adopt a token-based vector space similarity measure called Jaccard Similarity[2].

We also filter **MC**s in Consistent **MC**(A, *) for the cases that some or all of the relations *include, beIncluded, disjoint* exist together in Consistent **MC**(A, *). They can lead to breaking Rule 1. For example, given <*101:author, 302:author, equivalent*> and <*101:author, 302:firstauhtor, include*>, there is no relation between *302:author* and *302:firstauthor* in *Ontology302*. So we delete the second one according to Rule 2. When matching *Ontology101* and *Ontology302*, at first we obtain more than 100 **MC**s. However by filtering the results only 46 **MC**s are left. The filtered matching candidates are denoted as **MCF**.

The process in the Matching Candidates Filter component is bidirectional. It could also lead to deletion of some correct matching candidates. To describe this issue, let's assume the case shown in Fig. 2a as an initial state before describing the Matching Candidates Addition component.

The Matching Candidates Filter component first deletes *AC* and *BC* from *Ontology 1* and then removes *AD* from *Ontology 2*. It is clear that the matching candidate *AC* in Fig. 2a could be correct but it will still be deleted. The Matching Candidates Addition process, which includes the Adequacy Check and Adding Missing Matching Candidates components, addresses this issue.

The Adequacy Check component checks whether any matching candidates are missing. This is implemented by checking whether Condition 1 below is satisfied.

Condition 1. $\forall$**MC** $\in$ **MCO**, it must $\exists$**MC**$^{'}$ $\in$ **MCF** which makes **MC**.left = **MC**$^{'}$.left or **MC**.right = **MC**$^{'}$.right. (1)

The Adding Missing Matching Candidates component marks the missing matching candidates according to Formula 4, which are denoted as **MissingMC**. It then submits them along with **MCF** into the Matching Candidates Filter component. The process will run until Condition 1 above is satisfied.

$$\mathbf{MissingMC} = \{MC | MC \in MCO \wedge \forall MC^{'} \in MCF \Rightarrow MC^{'}.left \neq MC.left \wedge MC^{'}.right \neq MC.right\} \quad (4)$$

The filtering process is iterative as shown in Fig. 1. It is terminable as described in Proof 1 (We use ^{i} to present related symbols in Iteration *i*).

Proof 1. *We assume that the process is in Iteration i now. According to Condition 1 and Formula 4, if the size of* ***MissingMC***$^{i-1}$ *is 0, the process has been stopped at Iteration i-1.*

MCF^{i-1} *does not break the rules and* ***MissingMC***$^{i-1}$ *is probable to break them. According to Formula 4,* ***MissingMC***$^{i-1}$ *does not affect* ***MCF***$^{i-1}$ *for the rules, so the Matching Candidates Filter component only filters* ***MissingMC***$^{i-1}$ *and keeps* ***MCF***$^{i-1}$*. So* ***MCF***^{i}*=****MCF***$^{i-1}$*+ filtered* ***MissingMC***$^{i-1}$*.*

The size of the filtered ***MissingMC***$^{i-1}$ *will not be 0, because the size of* ***MissingMC***$^{i-1}$ *is not 0 and the Matching Candidates Filter component does not delete all* ***MC****s in* ***MissingMC***$^{i-1}$*. So* ***MCF***$^{i-1}$ *is a subset of* ***MCF***^{i}*.*

MCO *remains unchanged and* ***MCF***$^{i-1}$ *is a subset of* ***MCF***^{i}*, so according to Formula 4,* ***MissingMC***^{i} *is a subset of* ***MissingMC***$^{i-1}$ *and the size of* ***MissingMC***^{i} *is smaller than the size of* ***MissingMC***$^{i-1}$*. When i tend to n (integer), the size of* ***MissingMC***^{i} *converges to 0. This means Condition 1 is satisfied and the process will stop.*

As shown above, OMI-DL describes a semantic iterative filtering process to remove redundant matching candidates by reasoners. The rules in the filtering process and the format of the matching candidates in OMI-DL are different from those in ASMOV as described in Section VI-B. The verification (or filtering) process in ASMOV mainly focuses on semantic relations in the matching candidates according to similarity values. The filtering process in OMI-DL puts more emphasis on combining semantic relations in the matching candidates together with semantics in matched ontologies. The filtering

[2]http://www.dcs.shef.ac.uk/~sam/stringmetrics.html

process in OMI-DL is more efficient compared to the verification process in ASMOV. It increases the precision of the system as shown in Section VIII-B1. Other methods discussed above mostly compare a threshold value with similarity values to filter matching candidates.

VII. Entity Aligning

This section presents correspondences generation by calculating similarity measures. Some correspondences are also produced by defining the closest subsumer of an entity description to enhance the overall performance of the system.

A. Correspondences Generation by Similarity Computation

The similarity measure between two entities A and B, denoted as *S(A,B)*, is computed by utilizing the filtered matching candidates, semantic information in an ontology and string similarity. The sum of three kinds of similarity measures is used for this purpose: lexical similarity measure denoted as *LS(A,B)*, semantic similarity measure denoted as *SS(A,B)*, and string similarity measure denoted as *SM(A,B)*. *SM(A,B)* is obtained by using the string similarity function described in Section VI-B; if A and B exist in **MCF**, LS(A,B) is ℓ ($0 \leq \ell \leq 1$), otherwise it is 0; *SS(A,B)* is described below.

Firstly, all the parent concepts of A and B are obtained. Secondly, the properties of A and A's parent concepts are obtained (denoted as *PS(A)*) and the properties of B and B's parent concepts are obtained (denoted as *PS(B)*). Thirdly, if two properties, one from *PS(A)* and one from *PS(B)*, appear together as some matching candidate in **MCF** which does not have *disjoint* relation, the counter is increased by one. The process examines any two properties respectively from *PS(A)* and *PS(B)* in a similar way. Fourthly, *SS(A,B)* is calculated by dividing the counter by multiplication of the sizes of *PS(A)* and *PS(B)*.

Any two entities from different ontologies which have the same type are related to one correspondence as defined below,

$$\mathbf{C}(A,B) = < A, B, Relation, S(A,B) > \qquad (5)$$

where *Relation* represents the matching candidate relation if A and B appear together as a matching candidate in **MCF**. Otherwise it is defined as "*No Relation*".

We compare the value of the similarity measure in each correspondence derived from the previous step with a threshold μ. The correspondences which have a lower value than the threshold are removed. The rest will be kept as a part of the alignment.

B. Correspondences Generation by Defining cs(E)

For aligning ontologies, we not only utilize the semantics in matched ontologies, but also the ones in alignments from the last stage. We introduce a set of new correspondences by defining the closest subsumer of an entity description. The definition future exploits semantics between matched ontologies and alignments, and ensure the consistency of the derived alignment. These correspondences are important because they are generally correct and do not appear in the alignments obtained from most of the other solutions as shown in Section VIII-B1. This method enhances the precision and recall of the system.

Definition 1. *An entity (a concept or a property) description C is the* ***closest subsumer*** *of an entity description E (**cs(E)** for short) in an ontology if it satisfies:*

1. $C \sqsubseteq E$ *and*
2. *if* $C' \sqsubseteq E$, *then* $C' \not\sqsupseteq C$;

These new correspondences are produced by two steps. We select special correspondences derived from the previous component by referring to those that have *equivalent* relation. We then combine the special correspondences with the closest subsumer of an entity description, as explained in Definition 1, to produce new correspondences.

Given a special correspondence *<A, B, equivalent, value>*, we calculate *cs(A)* and *cs(B)*. For $\forall C \in cs(B)$, the new correspondence will be established, namely *<A, C, included, 0>*. For $\forall D \in cs(A)$, the new correspondence will be established, namely *<D, B, beIncluded, 0>*. We implement *cs(E)* mainly by using a reasoner.

We take as an example *101:Book* and *302:Book* to explain this process. Given the correspondence *<101:Book, 302:Book, equivalent, 1.0>*, we first obtain *cs(101:Book)* = {*101:Collection, 101:Proceedings, 101:Monograph*} and then compose the new correspondences in relation to *302:Book*:

1. *<101:Collection, 302:Book, beIncluded, 0.0>*;
2. *<101:Proceedings, 302:Book, beIncluded, 0.0>*;
3. *<101:Monograph, 302:Book, beIncluded, 0.0>*.

Because *cs(302:Book)* is an empty set, no new correspondence is established for it. Generally, the three items above do not appear in the alignments proposed by most of other solutions.

As shown above, the correspondences inferred by *cs(E)* is the results of the semantics from the matched ontologies and the equal correspondences in the last stage. The definition limits the number of the result to the closest subsumer of the equal correspondences.

VIII. Evaluation

This section provides an evaluation of OMI-DL. It is implemented using MIT Java WordNet Interface[3], OWL-API[4], Pellet[5], and Alignment API[6] to interact with WordNet, process an ontology, execute inferences and compare alignments.

We evaluate OMI-DL against several ontology matching solutions namely: SOBOM, DSSim, AROMA, Lily, ASMOV, RiMOM, GeRoMe, Aflood, Kosimap, MapPSO, Amaker and TaxoMap. For comparative purposes, we have also used edna (a simple distance algorithm applied to labels).

We use three metrics: precision (**Prec.**), recall (**Rec.**) and F-Measure (**FMeas.**) as defined in [35] to evaluate the alignments. In our experiments, the metrics are computed using the **ExtGroupEval** evaluation class provided by the Alignment API.

[3]http://projects.csail.mit.edu/jwi/
[4]http://owlapi.sourceforge.net/
[5]http://clarkparsia.com/pellet
[6]http://alignapi.gforge.inria.fr/

A. *Test Dataset*

1) Ontologies from the Benchmark Dataset: We evaluate OMI-DL by using the benchmark dataset[7] in the domain of bibliography from the OAEI dataset. The reference alignment of each test case is provided on the OAEI web site.

In general, an ontology in the benchmark dataset includes around **37** concepts, 72 properties and **108** axioms. There are totally **110** ontologies in this dataset **except** *Ontology102* which is not found in the results of other solutions. We classify these ontologies into four groups as shown below according to entity label and entity comment attributes in an ontology.

G_1 includes ontologies that have similar lexical (or linguistic) and structural information. The ontologies in G_1 are taken from Folders 101 to 104 (except Folder 102). G_4 consists of four real life ontologies from Folders 301 to 304. Entity label and comment attributes in each ontology in G_2 do not contain random strings or strings in another language rather than English. G_3 consists of the rest of the ontologies in the benchmark dataset. All or a part of entity label and entity comment attributes in each ontology in G_3 contain random strings or strings in another language rather than English.

2) Ontologies from Real World Projects: In addition to evaluation of OMI-DL based on the benchmark dataset, we also use three sets of ontologies from real world projects.

In the first set, one is the content description ontology from the *m:Ciudad* project[8] called UDL-CD, and the other is DOLCE+DnS Ultralite[9] (refered to as DUL in this paper). UDL-CD provides concepts to describe mobile services and relates them to different real world entities, activities and events. The concepts in UDL-CD are adopted from different common ontologies and vocabularies such as DBPedia and CyC. DUL provides a set of upper level concepts that can be the basis for easier interoperability among many middle and lower level ontologies. UDL-CD includes **5806** concepts, 72 properties and **8937** axioms, and DUL includes 76 concepts, 109 properties and 1047 axioms.

The ontologies in the second set are related to the domain of organizing conferences and are taken respectively from the *Conference Management Toolkit*[10] and the *Conference Accelerator*[11]. In this paper, the ontologies are respectively called CMT and CAW. CMT includes 30 concepts, 59 properties and 207 axioms, and CAW includes 39 concepts, 66 properties and 377 axioms.

The ontologies in the third set are related to mouse anatomical terms, namely the *Mouse Genome Informatics*[12] and *Ontology at the MPI for Evolutionary Anthropology*[13]. In this paper, the ontologies are respectively called MGI and MEA. MGI includes **5466** concepts, 3 properties and **11552** axioms, and MEA includes **2744** concepts, 3 properties and **8289** axioms.

[7]http://oaei.ontologymatching.org/2009/benchmarks/
[8]http://www.mciudad-fp7.org/
[9]http://ontologydesignpatterns.org/wiki/Ontology:DOLCE%2BDnS_Ultralite
[10]http://msrcmt.research.microsoft.com/cmt/
[11]http://webistem.com/en/
[12]http://www.informatics.jax.org/searches/AMA_form.shtml
[13]https://onto.eva.mpg.de/

The evaluation of alignments of these ontologies are manually produced by our colleagues at the University of Surrey and Beijing University of Posts and Telecommunications. None of these individuals is associated with the OMI-DL system .

B. *Evaluation Results*

The evaluation takes three aspects into account: 1) the impact of different components on the results; 2) comparing OMI-DL with other existing solutions; 3) evaluation results for ontologies from real world projects.

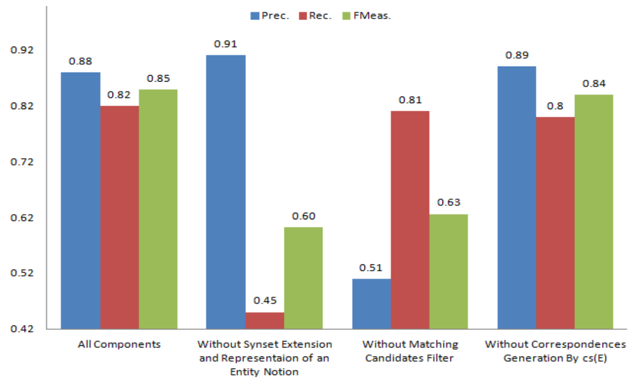


Fig. 3: Effect of using each of the components

1) Effect of Using Each of the Components: There are mainly four components which affect the performance of the system: Synset Extension, Representation of an Entity Notion, Matching Candidates Filter and Correspondences Generation by Defining *cs*(*E*). We have run the system with excluding some of the components at a time to analyze the effect of the various components in the performance over the four real life ontologies in G_4. It is clear that including all the components produces better results compared with the case that some components are excluded as shown in Fig. 3. The evaluation also demonstrates to what extent they affect the results.

It can be seen that the Synset Extension, Representation of an Entity Notion and Matching Candidates Filter components improve the results better than the Correspondences Generation by Defining *cs*(*E*) component. Without the Synset Extension and Representation of an Entity Notion components, although Prec. is increased from 88% to 91% (3%), Rec. and FMeas. are decreased from 82% to 45% (-37%) and from 85% to 60% (-25%) respectively in comparison with the case that the components are not excluded. This case violates the basis for producing matching candidates, so Rec. is very low. It can be seen that Synset Extension and Representation of an Entity Notion components help to increase the Rec. of the system.

By excluding the Matching Candidates Filter component, Prec., Rec. and FMeas. vary from 88% to 51% (-37%), from 82% to 81% (-1%) and from 85% to 63% (-22%) respectively. It can be seen that the Matching Candidates Filter component helps to increase the Prec. of the system. It is interesting to note that without the Matching Candidates Filter component, the Rec. is decreased. This component removes some matching candidates that could be correct correspondences at the end, but it does not add any new matching candidates. This shows that the existence of some matching candidates will affect the construction of correspondences in the rest of the system.

The Correspondences Generation by Defining $cs(E)$ component has 2% increase in Rec. and 1% decrease in Prec. in comparison with the case that all the components are included. All of the correspondences inserted by this component conform to semantics in two matched ontologies. However, the reference alignments do not include all reasonable correspondences and not all correspondences in the reference alignments are incoherent [36][14]. The correspondences inserted by this component are coherent according to our analysis. For example, *<101:Deliverable, 303:Report, beIncluded>* and *<101:Report, 303:ProjectReport, included>* produced by this component are coherent and reasonable, but they do not appear in the reference alignment while matching *Ontology101* and *Ontology303*.

2) Comparison with Other Existing Solutions: We have compared our method with SOBOM, DSSim, AROMA, Lily, ASMOV, RiMOM, GeRoMe, Aflood, Kosimap, MapPSO, Amaker and TaxoMap. We found that OMI-DL is one of the top performers in many test cases and is the best in some of them as shown in Table III and Table IV.

While evaluating ontologies in G_1, most solutions including OMI-DL obtain high performance (around 100% Prec. and Rec.) as shown in Table III. This is due to the fact that these ontologies have similar structural and lexical (or linguistic) information.

OMI-DL has high Prec., FMeas. and especially higher Rec. (99%) while evaluating ontologies in G_2 as shown in Table III. However, while evaluating ontologies in G_3, OMI-DL does not provide high results, especially in Rec. (only 44%), and ASMOV, Lily, RiMOM and Aflood have higher performance. Fig. 4 represents the differences in Rec., Prec. and FMeas. of each solution while evaluating ontologies in G_2 and G_3. The results reflect that the various solutions have a different degree of dependency on lexical (or linguistic) or structural information. As shown in Fig. 4, the performance (especially Rec.) of most of the solutions decreases with the lack of lexical (or linguistic) data (especially with G_3 that contains ontologies with random and often not meaningful words). It is also found that some solutions, especially OMI-DL whose Prec. Rec. and FMeas. are decreased from 97% to 83% (-14%), 99% to 44% (-55%) and 98% to 57% (-41%), are more dependent on lexical (or linguistic) information than ASMOV, Lily and RiMOM.

However, the main reason for the low performance in OMI-DL in this case is that our solution is designed to match the senses of English words. When entity label and entity comment attributes in an ontology contain random strings (which do not cover any meanings) or are specified in another language than English, OMI-DL cannot exploit the suitable sense and its extensions of a word using WordNet and cannot efficiently produce matching candidates by inferring entity notions which reflect lexical (or linguistic) information from entities. This causes some components (such as the

[14]This affects the evaluation results of our system. However, for consistency reasons, we followed the reference alignments.

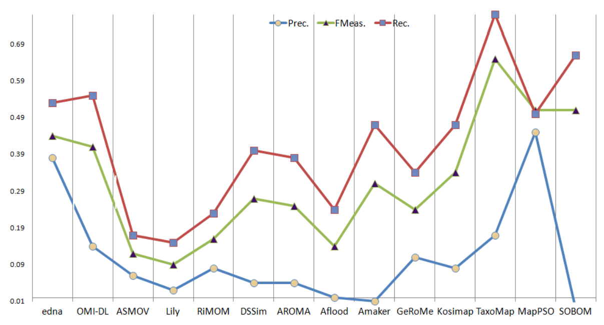


Fig. 4: Difference in performance between G_2 and G_3

Correspondences Generation By Similarity Computation and Correspondences Generation by Defining *cs(E)* components) which analyze structural or semantic information not to perform efficiently as they do not have access to sufficient inputs. Another reason for affecting the performance in OMI-DL is that we cannot find some correspondences together as we focus highly on consistency in matched ontologies as shown in Section VIII-C. In some testing cases, correspondences are correct but do not appear in the reference alignments as described in Section VIII-B1. As shown in Table IV, OMI-DL performs highly for the test cases that ontologies include meaningful words and statements to describe entity label and entity comment attributes (e.g. Test Cases *221–247* and *301–304*) even though their lexical (or linguistic) and structural information are altered.

The common feature of many of the existing solutions (such as Lily, ASMOV and RiMOM) is that they concurrently calculate various similarity measures between entities which reflect lexical and structural information. The calculations of the different similarity measures do not depend on each other. These solutions then combine the values of these similarity measures by different methods (such as assigned weights in ASMOV and a dynamic multi-strategy in RiMOM). For example, ASMOV combines the values of the similarity measures by a formula as: $\omega_1 S_1$+...+$\omega_n S_n$ where ω_i represents an assigned weight and S_i represents the value of a similarity measure. It then compares the sum with a threshold μ to produce correspondences. If an ontology does not include some kinds of information, ASMOV can deal with this problem by adjusting μ and ω_i. So, solutions such as Lily, ASMOV and RiMOM still have high performance while evaluating ontologies in G_3 as shown in Table III. This could be due to the adjustment factor that can tune them for a special set of data. However, this will limit their flexibility usability in applying to various ontologies.

Although lexical (or linguistic) and structural information for ontologies in G_2 and G_4 vary, sufficient information can be extracted. As shown in Table III, the difference between Rec. in OMI-DL and the highest value is only of 1% while evaluating ontologies in G_2. Rec. in OMI-DL is one of the best for the four real life ontologies in G_4 as shown by Row 4 in Table III.

This is due to the fact that the synset extension and representation of entity notions enhance the possibility of finding more correspondences. The Correspondences Generation by

TABLE III: Experiment comparison with other solutions

Algo.	edna	OMI-DL	Kosimap	TaxoMap	GeRoMe	DSSim	AROMA
Test data	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.
G_1	0.96 1.00 0.98	1.00 1.00 1.00	0.99 0.99 0.99	1.00 0.34 0.51	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00
G_2	0.71 0.99 0.83	0.97 0.99 0.98	0.97 0.96 0.97	0.98 0.98 0.98	1.00 1.00 1.00	1.00 0.96 0.98	0.97 0.98 0.98
G_3	0.33 0.46 0.39	0.83 0.44 0.57	0.89 0.49 0.63	0.81 0.21 0.33	0.89 0.66 0.76	0.96 0.56 0.71	0.93 0.60 0.73
G_4	0.47 0.82 0.60	0.88 0.82 0.85	0.72 0.50 0.59	0.77 0.31 0.44	0.68 0.60 0.64	0.94 0.67 0.78	0.85 0.78 0.81
Algo.	Amaker	ASMOV	Lily	RiMOM	Aflood	SOBOM	MapPSO
Test data	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.	Prec. Rec. FMeas.
G_1	0.98 0.98 0.98	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	1.00 1.00 1.00	0.97 0.98 0.98	1.00 1.00 1.00
G_2	0.98 0.98 0.98	1.00 1.00 1.00	0.99 0.99 0.99	0.99 1.00 0.99	0.99 0.99 0.99	0.97 0.94 0.95	0.98 0.98 0.98
G_3	0.99 0.51 0.67	0.94 0.83 0.88	0.97 0.84 0.90	0.91 0.77 0.83	0.99 0.75 0.85	1.00 0.28 0.44	0.53 0.48 0.47
G_4	0.92 0.79 0.85	0.81 0.82 0.81	0.84 0.81 0.82	0.81 0.82 0.81	0.90 0.81 0.85	0.92 0.55 0.69	0.54 0.29 0.37

Defining *cs(E)* also helps to increase the Rec. in OMI-DL. Row 4 in Table III also shows that edna has high Rec. (82%), but its Prec. is low (47%).

In this case, Prec. in OMI-DL is lower in contrast to the high Rec. in OMI-DL. The difference between Prec. in OMI-DL and the highest Prec. obtained via other solutions for G_2 is 3% and for G_4 is 6% (as shown in Talbe III). Row 4 in Table III shows that SOBOM and DSSim have high Prec. but very low Rec., because they only generate around 50 correspondences, and other solutions produce around 70 correspondences.

Considering Rec. and Prec., even if OMI-DL has lower Prec. than some solutions, it has high FMeas. value. while evaluating ontologies in G_1, G_2 and especially G_4 (as shown in Table III). Row 4 in Table III shows that the improvement for FMeas. in OMI-DL is 25% over edna, 4% over ASMOV, 3% over Lily, 4% over RiMOM, 7% over DSSim, 4% over AROMA, 20% over GeRoMe, 25% over Kosimap, 40% over TaxoMap, 48% over MapPSO and 16% over SOMBOM. Row 4 in Table III also shows that Aflood, Amaker and OMI-DL have the same FMeas. (85%). But the difference between Prec. and Rec. in OMI-DL (6%) is smaller than Aflood (9%) and Amaker (13%). So Pec. and Rec. in OMI-DL are more balanced.

It is worth noting that the ExtGroupEval class in the Alignment API does not consider semantic relations in correspondences when it computes the metrics. OMI-DL produces correspondences containing semantic relations (such as <*Monograph*, *Book*, *beIncluded*>) which are not found in other solutions. This is important in some applications (such as ontology integration).

3) Evaluation of Ontologies from Real World Projects: This subsection demonstrates applying our method to ontologies with broader scope of concepts. We provide an evaluation of matching ontologies from real world projects. Prec., Rec. and FMeas. are shown in Table V for each case[15].

When evaluating matching DUL and UDL-CD, OMI-DL maps $< DUL : classifies >$ to $< UDL - CD : category >$. The main reason is that the extensions of a suitable synset of the word "*classify*" contain a suitable synset of the word "*category*". This leads to the establishment of the matching candidate based on their notions. In CMT-CAW set, we can infer <*CMT:Chairman, CAW:Chair, equivalent*> when matching CMT and CAW, because OMI-DL assign the right synsets to *Chairman* and *Chair* respectively.

OMI-DL can also map each element in *cs(CMT:Person)*

[15]The detailed results can be accessed from: http://tinyurl.com/cufbg4u

TABLE IV: The results of OMI-DL for the benchmark dataset

Test Case	Prec. Rec. Fmeas.	Test Case	Prec. Rec. Fmeas.
101	1.00 1.00 1.00	103	1.00 1.00 1.00
104	1.00 1.00 1.00	201	NaN 0.00 NaN
201-2	0.93 0.79 0.86	201-4	0.88 0.60 0.71
201-6	0.69 0.42 0.53	201-8	0.91 0.22 0.35
202	1.00 0.01 0.02	202-2	0.92 0.79 0.85
202-4	0.87 0.60 0.71	202-6	0.69 0.42 0.53
202-8	0.91 0.22 0.35	203	1.00 1.00 1.00
204	0.98 0.94 0.96	205	0.69 0.34 0.46
206	0.83 0.35 0.49	207	NaN 0.00 NaN
208	0.98 0.94 0.96	209	0.69 0.32 0.44
210	0.83 0.35 0.49	221	1.00 1.00 1.00
222	0.96 1.00 0.98	223	0.92 1.00 0.96
224	1.00 1.00 1.00	225	1.00 1.00 1.00
228	1.00 0.97 0.98	230	0.94 1.00 0.97
231	1.00 1.00 1.00	232	1.00 1.00 1.00
233	1.00 0.97 0.98	236	1.00 0.97 0.98
237	0.96 1.00 0.98	238	0.92 1.00 0.96
239	0.97 0.97 0.97	240	0.79 1.00 0.88
241	1.00 0.97 0.98	246	0.97 0.97 0.97
247	0.79 1.00 0.88	248	1.00 0.01 0.02
248-2	0.94 0.79 0.86	248-4	0.89 0.60 0.72
248-6	0.77 0.42 0.55	248-8	0.91 0.22 0.35
249	1.00 0.01 0.02	249-2	0.92 0.79 0.85
249-4	0.87 0.60 0.71	249-6	0.69 0.42 0.53
249-8	0.91 0.22 0.35	250	NaN 0.00 NaN
250-2	0.81 0.79 0.80	250-4	0.71 0.61 0.66
250-6	0.47 0.42 0.44	250-8	1.00 0.21 0.35
251	1.00 0.01 0.02	251-2	0.96 0.80 0.87
251-4	0.95 0.60 0.74	251-6	0.85 0.42 0.56
251-8	0.91 0.23 0.36	252	1.00 0.01 0.02
252-2	0.84 0.79 0.81	252-4	0.84 0.79 0.81
252-6	0.84 0.79 0.81	252-8	0.84 0.79 0.81
253	1.00 0.01 0.02	253-2	0.94 0.79 0.86
253-4	0.89 0.60 0.72	253-6	0.77 0.42 0.55
253-8	0.91 0.22 0.35	254	NaN 0.00 NaN
254-2	0.90 0.79 0.84	254-4	0.83 0.61 0.70
254-6	0.64 0.42 0.51	254-8	1.00 0.21 0.35
257	NaN 0.00 NaN	257-2	0.81 0.79 0.80
257-4	0.71 0.61 0.66	257-6	0.47 0.42 0.44
257-8	1.00 0.21 0.35	258	1.00 0.01 0.02
258-2	0.96 0.80 0.87	258-4	0.95 0.60 0.74
258-6	0.85 0.42 0.56	258-8	0.91 0.23 0.36
259	1.00 0.01 0.02	259-2	0.84 0.79 0.81
259-4	0.84 0.79 0.81	259-6	0.84 0.79 0.81
259-8	0.84 0.79 0.81	260	0.00 0.00 NaN
260-2	0.88 0.79 0.84	260-4	0.86 0.62 0.72
260-6	0.71 0.41 0.52	260-8	0.88 0.24 0.38
261	0.00 0.00 NaN	261-2	0.63 0.79 0.70
261-4	0.63 0.79 0.70	261-6	0.63 0.79 0.70
261-8	0.63 0.79 0.70	262	NaN 0.00 NaN
262-2	0.90 0.79 0.84	262-4	0.83 0.61 0.70
262-6	0.64 0.42 0.51	262-8	1.00 0.21 0.35
265	0.00 0.00 NaN	266	0.00 0.00 NaN
301	0.90 0.76 0.83	302	1.00 0.69 0.81
303	0.70 0.83 0.76	304	0.94 0.95 0.94

TABLE V: Evaluation for matching real world ontologies

Num.	Ontologies	FMeas.	Prec.	Rec.
1	DUL UDL-CD	0.65	0.93	0.50
2	CMT CAW	0.67	0.78	0.58
3	MGI MEA	0.84	0.89	0.79

to $< CAW : Person >$ such as *<CMT:User, CAW:Person, beIncluded>* and *<CMT:ConferenceMember, CAW:Person, beIncluded>* (based on the correspondence *<CMT:Person, CAW:Person, equivalent>*). For the same reason, we can also infer *<DUL:NaturalPerson, UDL-CD:Person, beIncluded>* and *<DUL:SocialPerson, UDL-CD:Person, beIncluded>*.

Because both MGI and MEA have almost the same entity label attributes and have not any other information, this leads to that the same words from different ontologies have the same synsets at the same contextes and the entities from different ontologies have the same entity notions. So the words for the source entity in a correspondence are similar to those used for the target entity in the correspondence when matching MGI and MEA, such as *<MGI:Colon, MEA:colon, equivalent>* and *<MGI:Bone, MEA:bone, equivalent>*.

Overall, the real world ontologies provide broader concepts and descriptions, and the entity label attributes are also described in different terminologies. As the results show, OMI-DL performs efficiently when it is applied to ontologies with broader scope in the real world applications.

C. Analysis And Discussion

To demonstrate the strengths and limitations of OMI-DL, we use the sample ontologies (*Ontology101* and *Ontology302*) in Section IV as an example to discuss the derived alignment.

As shown in Table IV, OMI-DL has 100% in Prec. and 69% in Rec. when matching the two sample ontologies. The correspondences which appear in the reference alignment but are not found by OMI-DL are listed in the following:

1. *<101:Chapter, 302:InBook, equivalent>*;
2. *<101:Reference, 302:Resource, equivalent>*;
3. *<101:Booklet, 302:Publication, beIncluded>*;
4. *<101:LectureNotes, 302:Publication, beIncluded>*;
5. *<101:Academic, 302:Publication, beIncluded>*;
6. *<101:Informal, 302:Publication, beIncluded>*;
7. *<101:Report, 302:Publication, beIncluded>*;
8. *<101:Part, 302:Publication, beIncluded>*;
9. *<101:Deliverable, 302:Publication, beIncluded>*;
10. *<101:Unpublished, 302:Publication, beIncluded>*;
11. *<101:proceedings, 302:booktitle, equivalent>*;
12. *<101:url, 302:softCopyURI, equivalent>*;
13. *<101:isPartOf, 302:booktitle, equivalent>*;
14. *<101:book, 302:booktitle, equivalent>*;

The Synset Extension and Representation of an Entity Notion components generate Items 3 to 10. They appear in the original matching candidates. However, according to the filtering process based on the rules as shown in Section VI-B, we remove them. If the filtering process is not applied, we will also obtain some other correspondences which do not appear in the reference alignment (see Fig. 3 and Table II). So the Matching Candidates Filter component enhances the Prec. of the system, but we still need to improve the rules in the filtering process to produce a higher Rec. value.

For Item 2, relations between *101: Reference* and *302: Resource* are not found. Because relations between their labels are not found in WordNet. For Item 12, *url* cannot be found in WordNet. So using other external resources such as Linked Open Data [37] may improve the performance. The Tokenizer component is not able to process *booktitle* into two words *<book, title>*. So Items 11, 13 and 14 are missing in the derived alignment. The enhanced string and label processing can solve this problem and will be a part of future work.

OMI-DL ensures the consistency of the derived alignment, so we cannot find some of the correspondences above together. For example, *<101:proceedings, 302:booktitle, equivalent>*, *<101:isPartOf, 302:booktitle, equivalent>*, and *<101:book, 302:booktitle, equivalent>* cannot be found together, because this leads to *<101:isPartOf* ***equivalent*** *101:proceedings* ***equivalent*** *101:book>*, and it also conflicts with the axiom in *0ntology101*, namely *<101:isPartOf, 101:proceedings, Include>*. OMI-DL also cannot find *<101:Chapter, 302:InBook, equivalent>* and *<101:InBook, 302:InBook, beIncluded>* together, because we can infer *<101:InBook, 101:Chapter, beIncluded>* which is not claimed and is not reasonable in *Ontology101*. If those incoherent correspondences [36] in the reference alignments are removed, OMI-DL can obtain an even higher performance.

Using the synset extension and representation of an entity notion improves the Rec. of the system; for example, OMI-DL obtains *<101: date, 302: publishedOn, beIncluded>* which does not appear in the alignments provided by most of other solutions. By defining the closest subsumer of an entity description, OMI-DL obtains *<101:Collection, 302:Book, beIncluded>* and *<101:Monograph, 302:Book, beIncluded>* which also do not appear in the alignments provided by most of other solutions.

IX. Conclusion And Future Work

This paper proposes a comprehensive framework for ontology matching called OMI-DL. It analyzes the senses of a word and extends them to define the representation of an entity notion in an ontology. OMI-DL generates matching candidates between entities and filters out redundant matching candidates using reasoners. Our method constructs correspondences between entities according to the closest subsumer of an entity description and similarity measures. We have conducted evaluations and compared the results with other methods. The results empirically prove its strengths.

Future work will focus on improving the performance in OMI-DL by extending the scope of analysis methods. Using other common terminologies and structured data on the Web (such as DBpedia and Linked Open Data resources) will provide broader resources to describe and represent concepts and enable the concepts to be extended. We also plan to include individual analysis in our method. This will enhance the results of our algorithm when lexical and semantic analysis methods fail to provide high confidence in establishing correspondences. Future work will also focus on scalability issues and provide matching for large scale ontologies.

References

[1] K. Xiangping, L. Deyu, and W. Suge, "Research on domain ontology in different granulations based on concept lattice," *Knowledge-Based Systems*, vol. 27, no. 0, pp. 152–161, 2012.

[2] X. Liu, A. Bouguettaya, J. Wu, and L. Zhou, "Ev-lcs: A system for the evolution of long-term composed services," *Services Computing, IEEE Transactions on*, vol. 6, no. 1, pp. 102–115, First 2013.

[3] J. Zhang, D. Kuc, and S. Lu, "Confucius: A tool supporting collaborative scientific workflow composition," *Services Computing, IEEE Transactions on*, vol. 7, no. 1, pp. 2–17, Jan 2014.

[4] J. Gracia and E. Mena, "Semantic heterogeneity issues on the web," *Internet Computing, IEEE*, vol. 16, no. 5, pp. 60–67, 2012.

[5] P. Shvaiko and J. Euzenat, "Ten challenges for ontology matching," in *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems*, ser. LNCS, no. 5332, Mexico, 2008.

[6] ——, "Ontology matching: State of the art and future challenges," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 1, pp. 158–176, Jan 2013.

[7] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang, "Using bayesian decision for ontology mapping," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 4, pp. 243–262, 2006.

[8] G. Pirro and D. Talia, "Ufome: An ontology mapping system with strategy prediction capabilities," *Data & Knowledge Engineering*, vol. 69, no. 5, pp. 444–471, 2010.

[9] S. Albagli, R. Ben-Eliyahu-Zohary, and S. E. Shimony, "Markov network based ontology matching," *Journal of Computer and System Sciences*, vol. In Press, Accepted Manuscript, pp. –, 2011.

[10] F. Giunchiglia and P. Shvaiko, "Semantic matching," *Knowl. Eng. Rev.*, vol. 18, pp. 265–280, September 2003.

[11] Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka, "Ontology matching with semantic verification," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 3, pp. 235–251, 2009.

[12] O. Udrea, L. Getoor, and R. J. Miller, "Leveraging data and structure in ontology integration," in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2007, pp. 449–460.

[13] M. Bonifacio, A. Dona, A. Molani, and L. Serafini, "Context matching for electronic marketplaces: a case study," *The Knowledge Engineering Review*, vol. 18, no. 04, pp. 317–328, 2003.

[14] S. Khan and M. Safyan, "Semantic matching in hierarchical ontologies," *Journal of King Saud University - Computer and Information Sciences*, no. 0, pp. –, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157814000111

[15] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003, ch. Basic Description Logics, pp. 46–99.

[16] C. Quix, P. Roy, and D. Kensche, "Automatic selection of background knowledge for ontology matching," in *Proceedings of the International Workshop on Semantic Web Information Management*, ser. SWIM '11. New York, NY, USA: ACM, 2011, pp. 51–57.

[17] P. Maio and N. Silva, "An extensible argument-based ontology matching negotiation approach," *Science of Computer Programming*, no. 0, pp. –, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642314000264

[18] P. Xu, Y. Wang, and T. Zang. (2009) Alignment results of sobom for oaei 2009. [Online]. Available: http://dit.unitn.it/~p2p/OM-2009/oaei10_paper12.pdf

[19] C. E. Yap and M. H. Kim, "Instance-based ontology matching with rough set features selection," in *IT Convergence and Security (ICITCS), 2013 International Conference on*, Dec 2013, pp. 1–4.

[20] G. Acampora, V. Loia, and A. Vitiello, "Enhancing ontology alignment through a memetic aggregation of similarity measures," *Information Sciences*, vol. 250, no. 0, pp. 1 – 20, 2013.

[21] M. Houshmand and E. Khorram, "Similarity aggregation in ontology matching based on reliability maximization," in *Electrical Engineering (ICEE), 2011 19th Iranian Conference on*, may 2011, pp. 1–6.

[22] Y. Qu and G. Cheng, "Falcons concept search: A practical search engine for web ontologies," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 41, no. 4, pp. 810–816, july 2011.

[23] W. Peng, "Research on the key issues in ontology mapping," Ph.D. dissertation, Dept. Electron.Eng., SouthEast Univ., NanJing, China, 2009.

[24] R. Chen, C. Bau, and C. Yeh, "Merging domain ontologies based on the wordnet system and fuzzy formal concept analysis techniques," *Applied Soft Computing*, vol. 11, no. 2, pp. 1908–1923, 2011.

[25] V. Spiliopoulos, G. A. Vouros, and V. Karkaletsis, "On the discovery of subsumption relations for the alignment of ontologies," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 1, pp. 69–88, 2010.

[26] X. Xue, Y. Wang, and A. Ren, "Optimizing ontology alignment through memetic algorithm based on partial reference alignment," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3213 – 3222, 2014.

[27] J. Li, J. Tang, Y. Li, and Q. Luo, "Rimom: A dynamic multistrategy ontology alignment framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 1218–1232, 2008.

[28] A. Buccella, A. Cechich, D. Gendarmi, F. Lanubile, G. Semeraro, and A. Colagrossi, "Building a global normalized ontology for integrating geographic data sources," *Computers & Geosciences*, vol. 37, no. 7, pp. 893–916, 2011.

[29] J. Bock and J. Hettenhausen, "Discrete particle swarm optimisation for ontology alignment," *Information Sciences*, no. -, pp. –, 2010.

[30] X. Wang and Q. Xu, "An improved ant colony optimization for ontology matching," in *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, vol. 4, march 2011, pp. 234–238.

[31] M. Kolli and Z. Boufaida, "A description logics formalization for the ontology matching," *Procedia Computer Science*, vol. 3, no. 0, pp. 29–35, 2011.

[32] Y. Jiang, X. Wang, and H.-T. Zheng, "A semantic similarity measure based on information distance for ontology alignment," *Information Sciences*, vol. 278, no. 0, pp. 76 – 87, 2014.

[33] T. Richens, "Anomalies in the wordnet verb hierarchy," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, August 2008, pp. 729–736.

[34] H. Deng, I. King, and M. R. Lyu, "Enhanced models for expertise retrieval using community-aware strategies," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2011.

[35] J. Euzenat and P. Shvaiko, *Ontology matching*. Heidelberg (DE): Springer-Verlag, 2007.

[36] C. Meilicke, "The relevance of reasoning and alignment incoherence in ontology matching." in *ESWC*, ser. Lecture Notes in Computer Science, vol. 5554. Springer, 2009, pp. 934–938.

[37] V. Mascardi, A. Locoro, and P. Rosso, "Automatic ontology matching via upper ontologies: A systematic evaluation," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 5, pp. 609 –623, 2010.