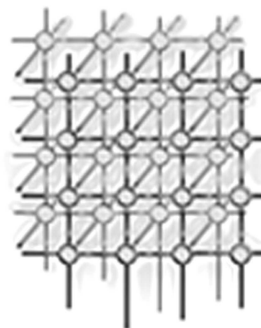


# Ontology Mapping: As a Binary Classification Problem



Ming Mao<sup>\*†¶</sup>, Yefei Peng<sup>‡¶</sup>, Michael Spring<sup>§</sup>

<sup>†</sup>*SAP Labs, 3410 Hillview Ave, Palo Alto, CA 94306, USA*

<sup>‡</sup>*Yahoo! Labs, 4401 Great America Pkwy, Santa Clara, CA 95054, USA*

<sup>§</sup>*School of Information Sciences, University of Pittsburgh, 135 N. Bellefield Ave, Pittsburgh, PA 15260, USA*

---

## SUMMARY

Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies. Ontology mapping is critical to achieve semantic interoperability in the WWW. To solve the ontology mapping problem, this paper proposes a non-instance learning-based approach that transforms the ontology mapping problem to a binary classification problem and utilizes machine learning techniques as a solution. Same as other machine learning based approaches, a number of features (i.e., linguistic, structural and web features) are generated for each mapping candidate. However, in contrast to other learning-based mapping approaches, the features proposed in our approach are generic and do not rely on the existence and sufficiency of instances. Therefore our approach can be generalized to different domains without extra training efforts. To evaluate our approach, two experiments (i.e., within-task vs. cross-task) are implemented and the SVM algorithm is applied. Experimental results show that our non-instance learning-based ontology mapping approach performs well on most of OAEI benchmark tests when training and testing on the same mapping task; and the results of approach vary according to the likelihood of training data and testing data when training and testing on different mapping tasks.

KEY WORDS: ontology mapping; binary classification; machine learning; Semantic Web

---

\*Correspondence to: Ming Mao, SAP Labs, 3410 Hillview Ave, Palo Alto, CA, USA

†E-mail: ming.mao@sap.com

‡E-mail: ypeng@yahoo-inc.com

§E-mail: spring@pitt.edu

¶This work was done when the authors had affiliation with University of Pittsburgh.



## 1. INTRODUCTION

The World Wide Web (WWW) is widely used as a universal medium for information exchange. However, semantic interoperability (i.e., information systems can exchange data and reuse the exchanged data with their intended meanings) in the WWW is still limited due to the heterogeneity of information, which occurs at three levels, i.e., syntax, structure and semantic level [1]. Syntactic heterogeneity is the simplest heterogeneity problem caused by the usage of different data formats. Standardized formats such as XML<sup>†</sup>, RDF/RDFS<sup>‡</sup> and OWL<sup>§</sup> have been widely used to describe data in a uniform way that makes automatic processing of shared information easier. However standardization does not overcome structural heterogeneity that occurs as a result of the way information is structured even in homogeneous syntactic environments. For example, one source might model trucks but only classify them into a few categories; while the other source might make very fine-grained distinctions between types of trucks based on their physical structure, weight, purpose, etc. Manually encoded transformation rules as well as some middleware components have been used to solve structural heterogeneity problems [2]. Though sophisticated solutions to syntactic and structural heterogeneity have been developed, the problem of semantic heterogeneity is still only partially solved. Semantic heterogeneity occurs whenever two contexts do not share the same interpretation of information (e.g. homonyms and synonyms). For example, Swoogle (<http://swoogle.umbc.edu/>) returns 346 documents when searching for *spring* (Based on the results retrieved from Swoogle in July, 2007). The top ranked results show that the same term has many different meanings, e.g. one spring means the season, the other spring means the ground water etc. Though the approaches such as using synonym sets, term networks, concept lattices, features and constraints have been proposed as solutions for solving semantic heterogeneity among different information systems [1], they are not sufficient to solve the problem of semantic heterogeneity in the WWW environment.

The vision of the Semantic Web [3] provides many new perspectives and technologies to overcome the limitation of the WWW. Ontology, a formal, explicit specification of a shared conceptualization [4], has been suggested as a way to solve the problem of semantic heterogeneity, and thus enable semantic interoperability between different web applications and services. Given the reality of multiple ontologies over many domains, ontology mapping that aims to find semantic correspondences between similar elements of different ontologies has been the subject of research in various communities [5][6][7][8].

Different techniques have been examined in ontology mapping, e.g., using linguistic techniques to measure the lexical similarity of concepts in ontologies [9][10][11][2], treating ontologies as structural graphs [12] [13], taking the advantage of information retrieval techniques [14][15] and artificial intelligent model [16][17][18], applying heuristic rules to look for specific mapping patterns [19][20], and learning to map ontologies through machine learning

---

<sup>†</sup><http://www.w3.org/TR/2004/REC-xml-20040204>

<sup>‡</sup><http://www.w3.org/TR/2004/REC-rdf-primer-20040210>

<sup>§</sup><http://www.w3.org/TR/2003/CR-owl-features-20030818/>



techniques [21][7][22]. Comprehensive surveys of ontology mapping approaches can be found in [23][24].

Some machine learning based ontology mapping approaches, such as [21], heavily rely on instances, and thus they simply cannot work if no instances are provided. Though instances provide important information to ontology mapping and, in some cases, they are even critical to derive correct mappings, we have to face the reality that no or very few instance are not available in real world cases. For example, in the situation that two web services are trying to communicate, normally no instances are provided in the service descriptions that need to be mapped. However, on the other hand, human being can derive mappings without instances by utilizing linguistic information, domain knowledge, and ontology structures, and etc. These facts motivate our work on a machine learning based ontology mapping approach that does not rely on the existence and sufficiency of instances, i.e., such an approach should be able to work when no instances are available, but also can make use of instances, if they are provided, as well as linguistic information, structure information, and background knowledge.

In this paper, we propose a non-instance machine learning based ontology mapping approach. The insight of our approach is to treat ontology mapping problem as a binary classification problem, which, on one hand, can take advantage of machine learning techniques; and on the other hand, can overcome the limitation of traditional learning-based approaches, i.e., first they heavily rely on the availability of instance data when measuring the similarity of classes/attributes; secondly, they require new training data to rebuild their model when domain changes and thus restrict the universality of their model. However it should be noted that, though our approach does not rely on the existence and sufficiency of instances, it can make use of instances if existing. This makes our approach very flexible.

To evaluate our approach, we adopt the benchmark tests from OAEI ontology matching campaign 2007. We follow the evaluation criteria of OAEI, calculating the precision, recall and f-measure of each test case. Experimental results show that our non-instance learning-based ontology mapping approach performs well in most of OAEI benchmark tests when training and testing on the same mapping task; and the results of approach vary according to the likelihood of training data and testing data when training and testing on different mapping tasks. The rest of the paper is organized as follows. Section 2 defines the ontology mapping problem as a binary classification problem in a formal way. Section 3 proposes a non-instance machine learning based approach to solve the ontology mapping problem. Section 4 evaluates our approach using the benchmark tests from OAEI campaign in 2007. Section 5 reviews the related work in the ontology mapping area. Finally conclusions are made and some future works are given in Section 6.

## 2. PROBLEM DEFINITION

Ontology is a formal, explicit specification of a shared conceptualization in terms of classes, attributes and relations [23]. Ontologies are typically represented as taxonomic trees that include classes, properties, and relations, and associated with instances. Two sample bibliographic ontologies are shown in Figure 1, in which the ellipses indicate classes (e.g., “Reference”), the dashed rectangles indicate properties (e.g., “publisher”), the lines with

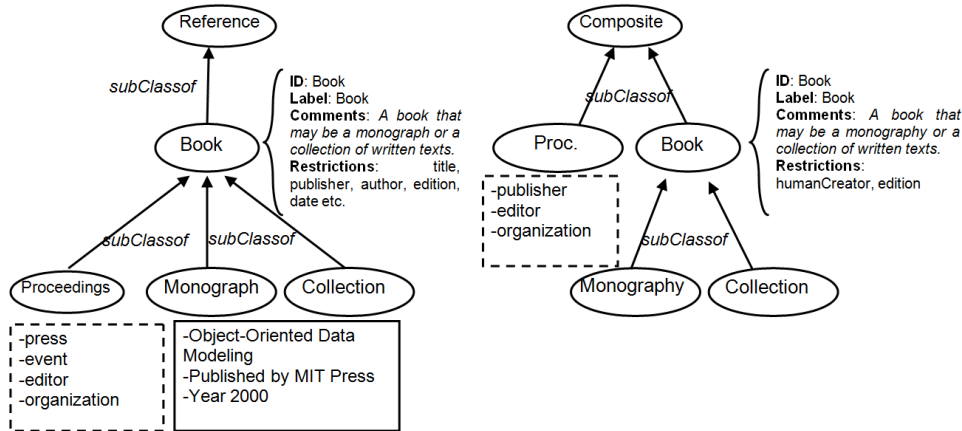


Figure 1. Two sample bibliographic Ontologies

arrowhead indicate “subClassof” relation between two classes, and the solid rectangles indicate instances of class (e.g., “Object-oriented data modeling”). Each class and property can also have descriptive information (e.g., ID, label, comment) and restrictions (e.g., title, publisher) as indicated in the brace next to “Book”.

The process of ontology mapping is to find semantic correspondences between similar elements in two homogeneous ontologies, and many ways can be used to judge the quality of a mapping result. According to Euzenat and Shvaiko [25], the problem of ontology mapping can be defined as follows. Given two ontology  $O_1$  and  $O_2$ , an alignment between  $O_1$  and  $O_2$  is a set of correspondences:  $\langle e_1, e_2, r, l \rangle$ , with elements  $e_1 \in O_1$  and  $e_2 \in O_2$  being the two matched entities,  $r$  is a relationship holding between  $e_1$  and  $e_2$ , and  $l$  expresses the level of confidence in this correspondence.  $O_1$  and  $O_2$  also represent matchable elements in ontology  $O_1$  and  $O_2$  respectively.

In this paper, relationships are limited to “=” relationship, elements are limited to “classes” and “properties” of ontology. To be consistent with OAEI campaign, only two levels of correctness are considered: correct or incorrect, which can be depicted as a binary set  $\{0, 1\}$ . According to the statement, mappings in Figure 1 can be evaluated as followings:  $m(Book_{right}, Book_{left}) = 1$ ,  $m(Proceedings, Proc) = 1$ ,  $m(Monograph, Monography) = 1$ ,  $m(Proceedings, Talks) = 0$ ,  $m(Proceedings, Monography) = 0$ , and etc.

We can rewrite  $\langle e_1, e_2, r, l \rangle$  as  $m(e_1, e_2) = l$ ,  $l \in \{0, 1\}$ , where  $e_1$  is element from ontology  $O_1$ ,  $e_2$  is element from ontology  $O_2$ . We also simplify  $m(e_1, e_2)$  as  $m_{e_1, e_2}$ . Then the ontology mapping problem can be transformed as a binary classification problem: Given  $e_1$  and  $e_2$ , to predict  $m(e_1, e_2)$ , which is the correct label associated with the pair. More details of binary classification problem are presented in 3.3.



### 3. APPROACH

In this section, we propose a non-instance learning-based approach for solving ontology mapping problem. Previous learning-based approaches have been reported in [21][7][22]. However these approaches either has a limitation that it heavily relies on the availability of instance data when measuring the similarity of classes/attributes, or requires new training data to rebuild their model when domain changes and thus restrict the universality of the model. To overcome the limitations, we treat the ontology mapping problem as a binary classification problem. We learn a generic mapping model, which does not require the existence of instances and domain constraints. To learn a model, a variety of features that can reflect the characteristics of mapping pairs are generated, and then the SVM algorithm is applied.

#### 3.1. Overview of the Non-instance Learning-based Approach

Generally speaking, our approach has the following steps, see detailed description in 4.4.1 and 4.4.2.

1. Generate various domain independent features (i.e., linguistic, structural and web features) to describe the characteristics of ontologies.
2. Randomly generate training and testing set for OAEI benchmark tests.
3. Train a SVM model on training set.
4. Classify testing data on the trained SVM model.
5. Extract mapping results of testing data using naive descendant extraction algorithm [26].
6. Evaluate testing data against ground truth.
7. Finally, repeat step 2-6 10 times and get the average evaluation result to eliminate bias.

#### 3.2. Feature Generation

Applying machine learning techniques to ontology mapping context raises the question of what types of information should be used in the learning process. Many different types of information can contribute toward deciding the correspondence of a mapping pair. Two principles are followed to select features:

1. The feature should not be limited to instances. It could be generated from classes, properties and/or instances in ontologies.
2. The feature should be general enough and domain independent so that the model could be generalized to other applications regardless of the variety of domain.

In our approach, 3 categories, i.e., linguistic features, structural features and web features, and total 26 features are generated for each mapping pair.

##### 3.2.1. Linguistic Features

Linguistic features are selected according to the principle described in [27]. Table I lists 16 linguistic features that have been used in our approach. These features can be divided into two types:



1. Isolated characteristics of elements in mapping pair, e.g. length of elements, number of tokens, etc.
2. Syntactic characteristics of a mapping pair, e.g. (normalized) length difference between elements, Levenshtein edit distance between two elements, the proportion of word change between elements, number of common tokens in the pair, the cosine similarity of the profile of elements, etc.

From description in the table, it is easy to understand how linguistic features are calculated based on the characteristics of a mapping pair except the profile similarity between two elements. Here we briefly introduce how we calculate ProfileSim, for more details please refer to [14][28]. The profile similarity is generated in three steps.

1. Profile Enrichment. For each element in the ontology, we generate a profile (i.e., a combination of the element's descriptive information) to represent it and thus enrich its information. In particular, the profile of a class = the class's ID + label + comments + other restriction + its properties' profiles + its instances' profiles. The profile of a property = the property's ID + label + its domain + its range. Then the *tf \* idf* weight [1] is assigned for each profile based on the whole collection of all profiles in the ontology.
2. Profile Propagation. To exploit the neighboring information of each element, the profile of the element's ancestors, descendants and siblings will be passed to that of the element with different weights.
3. Profile Mapping. Finally the cosine similarity between the profiles of two elements of  $e_1$  and  $e_2$  is calculated in a vector space model using the following equation, where  $V_{e_1}$  and  $V_{e_2}$  are two vectors representing the profile of element  $e_1$  and  $e_2$  respectively,  $n$  is the dimension of the profile vectors,  $V_k^{e_1}$  and  $V_k^{e_2}$  are  $k$ th element in the profile vector of element  $e_1$  and  $e_2$  respectively,  $\|V_{e_1}\|$  and  $\|V_{e_2}\|$  are the lengths of the two vectors respectively.

$$ProfileSim(e_1, e_2) = \frac{V_{e_1} \bullet V_{e_2}}{\|V_{e_1}\| \|V_{e_2}\|} = \frac{\sum_{k=1}^n V_k^{e_1} * V_k^{e_2}}{\sqrt{\sum (V_k^{e_1})^2} \sqrt{\sum (V_k^{e_2})^2}}$$

### 3.2.2. Web Features

Edit distance, which defines the strings similarity by the minimum number of insertions, deletions and substitutions that require transforming one string into the other, is a commonly used method to calculate the similarity of terms. However edit distance encounters trouble where synonyms exist. For synonyms, the intuitive way is to check auxiliary information in a thesaurus, e.g. WordNet. However WordNet suffers some drawbacks as well. First, semantic similarity between words changes across domains. Even though a thesaurus may contain a sufficiently wide range of common words (As of 2006, WordNet contains about 150,000 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs), sometimes it does not cover special domain vocabulary. For example, though apple is frequently associated with computers on the Web, this sense of apple is not listed in WordNet. Second, new words are



Features	Description
$le_1$	the length of $e_1$
$le_2$	the length of $e_2$
ldiff	the length difference between $e_1$ and $e_2 = le_1 - le_2$
absldiff	the absolute value of length difference = $abs(ldiff)$
ldiffn	the length difference normalized by the length of $e_1 = ldiff/le_1$
lengthratio	the absolute value of ldiffn : $abs(ldiffn)$
ntoken1	the number of tokens in $e_1$
commonw	the number of tokens in common in $e_1$ and $e_2$
editdist	the normalized Levenshtein edit distance of $e_1$ and $e_2$
worddist	the proportion of word changed from $e_1$ to $e_2$
word_pov	the proportion of tokens in common at beginning of $e_1$
word_suf	the proportion of tokens in common at the end of $e_1$
char_pov	the proportion of characters (utf8 bytes) in common at the beginning of $e_1$
char_suf	the proportion of characters in common at the end of $e_1$
digit_dropped	a boolean value, identifying whether a digit has been dropped from $e_1$ to $e_2$
ProfileSim	the cosine similarity of the profile of $e_1$ and that of $e_2$ , the profile is the combination of all linguistic information (e.g. name, label and comments) of an element

Table I. Linguistic features

continually created and new senses are assigned to existing words. Thesauri usually can not capture these new words and senses in time. As an alternative, the Web has a huge amount of information and the newest words/senses are stored. To overcome the problem of edit distance comparison and WordNet sense limitation, a new Web feature, i.e., WebDice coefficient has been proposed in [29].

Webdice coefficient is page count based co-occurrence measure and a variant of Dice coefficient [1]. Page count of a query is the number of pages that contain the query terms obtained from a Web search engine such as Yahoo. Page count can be considered as a global measure of co-occurrence of query terms. For example, the page count of the query “spring” AND “last name” is 1,550,000 and the page count of the query “spring” AND “season” is 57,300,000 in Google (Based on the results retrieved from <http://www.google.com> in July, 2007). The about 40 times more numerous page counts for “spring” AND “season” indicate that “spring” is more semantically similar to “season” than “last name”. Using page count alone to measure the co-occurrence of two terms does not always accurately express semantic similarity. This is because page count ignores the position of a word appearing in a page, and thus may incorrectly count pages, in which both words appear but far away from each other and without any relevance. To overcome the drawback, one must consider the page counts not just for query  $X$  AND  $Y$  but also for the individual words  $X$  and  $Y$  to access semantic similarity between  $X$  and  $Y$ .



In [29], Webdice is defined as following, where the notation  $H(X)$  denote the page counts for query  $X$  in a search engine,  $H(X, Y)$  denotes the page counts for the conjunction query  $X$  AND  $Y$ ,  $c$  is a predefined threshold (e.g.  $c = 5$ ) to reduce the adverse effects caused by random co-occurrences. Because of the scale and noise in Web data it is possible that a page contains two words purely accidentally. So the webdice feature of a pair of element is defined as:

$$WebDice(e_1, e_2) = \begin{cases} 0 & \text{if } H(e_1 \cap e_2) \leq c \\ H(e_1) + H(e_2) & \text{otherwise} \end{cases}$$

### 3.2.3. Structural Features

Structural information is important in estimating the similarity of ontologies. Different ontology elements (i.e., classes, properties) have different structural characteristics. For example, the structural features of a class may include the number of its subclasses and super classes, the similarity of its subclasses and super classes, the depth from the root to the class itself etc. The structural features of a property may include the similarity of its sub properties and super properties, the similarity of its domain and range etc. Structural features that are used in our approaches for a mapping hypothesis between classes (e.g.,  $C_1$  and  $C_2$ ) and properties (e.g.,  $p_{1i}$  and  $p_{2j}$ ) are listed in Table II.

### 3.3. Binary Classification

For each pair of potential mapping elements  $e_1$  and  $e_2$ , a feature vector  $X_{e_1, e_2} = X(e_1, e_2)$  will be generated by methods described in 3.2. For all pairs of  $e_1$  and  $e_2$ , where  $e_1 \in O_1$  and  $e_2 \in O_2$ , data set  $D$  includes all potential mapping pairs and their labels:  $D = (X_i, c_i) | X_i \in R^p, c_i \in \{0, 1\}_{i=1}^n$ .  $c_i$  is either 1 or 0, indicating the class to which the feature vector  $X_i$  belongs. Each  $X_i$  is a  $p$ -dimensional real vector.  $n$  is total number of potential mapping pairs. Now the problem of ontology mapping is transformed to feature space as a binary classification problem. Given a subset of data  $D_T \in D$ , a binary classification model could be trained,  $f : X \rightarrow C$ , where  $C \in \{0, 1\}$ .

It's true that sometimes a degree of mapping makes more sense than binary classification. However, to our knowledge, currently in ontology mapping research area all ground truth is given in a binary format, i.e., a pair of elements is either a correct mapping or not, which makes evaluation more straightforward. Hence our approach also outputs binary mapping result.

Support vector machine (SVM) is a sophisticated statistical machine learning method. It constructs a hyperplane in a high dimensional feature space, which can be used for classification, regression or other tasks. Like other machine learning methods, SVM automatically combines different types of features to maximize training error, so no need to manually assign weights to different features.

SVM method will output a decision function,  $g : X \rightarrow h$ , where  $h = [-1, 1]$  is confidence value. By default, a decision boundary of 0 could be used to transform  $h$  into binary decision. But this method may generate one-to-many mappings with conflict. To generate one-to-one mapping, an extra step needs to be taken to extract mappings from the set of all  $\langle e_1, e_2, g(X_{e_1, e_2}) \rangle$ .



Element	Feature	Description
Classes	DirPropNumDiff	The normalized difference between the numbers of the classes' direct properties
	DirPropSim	The edit distance based similarity between the classes' direct properties, i.e., $DirPropSim = Avg_i(max_j(EditDistSim(p_{1i}, p_{2j})))$ , where $p_{1i}$ and $p_{2j}$ are direct properties of class $C_1$ and $C_2$ .
	chNumDiff	The normalized difference between the numbers of the classes' subclasses.
	chSim	The edit distance based similarity between the classes' subclasses, i.e., $chSim = Avg_i(max_j(EditDistSim(subC_{1i}, subC_{2j})))$ , where $subC_{1i}$ and $subC_{2j}$ are subclasses of class $C_1$ and $C_2$ .
	Passim	The edit distance based similarity between the classes' super classes, i.e., $paSim = Avg_i(max_j(EditDistSim(paC_{1i}, paC_{2j})))$ , where $paC_{1i}$ and $paC_{2j}$ are super classes of class $C_1$ and $C_2$ .
	depDiff	The normalized difference between the depth to root of the classes
Properties	domainSim	The edit distance based similarity between the properties' domain
	rangeSim	The edit distance based similarity between the properties' range
	motherSim	The edit distance based similarity between the properties' mother class

Table II. Structural features

According to Meilicke and Stuckenschmidt [26], a mapping extraction function is a function  $t : O \times O \times M \rightarrow M$  iff for all  $O_1, O_2 \in O$  and for all  $m \in M$  we have  $t(O_1, O_2, m) \in m$ , where  $O$  denotes the set of ontologies and  $M$  denotes the set of possible mappings between two ontologies  $O_1, O_2 \in O$ ,  $m$  is the set of tuples  $\langle e_1, e_2, l_{12} \rangle$ ,  $e_1 \in O_1$ ,  $e_2 \in O_2$ ,  $l_{12}$  is confidence of the mapping. Naive descendant extraction algorithm [26] transforms many-to-many mapping into one-to-one mapping. First all mapping tuples in  $m$  is sorted descending by confidence. Then the algorithm iterates over  $m$  removing elements from  $m$ , and adding some tuples to final result. In each iteration, the tuple with highest confidence is removed from  $m$  and added to final result. All tuples which have one common element with the selected tuple are removed from  $m$ . The final result is a list of tuples with one-to-one mapping which is a subset of  $m$ . All tuples in final result set will be treated as correct mapping.



Tests	Description
#101-104	$O_R$ and $O_T$ have exactly the same or totally different names
#201-210	$O_R$ and $O_T$ have the same graph but different linguistics in some level
#221-247	$O_R$ and $O_T$ have the same linguistics but different graph
#248-266	Both graph and linguistics are different between $O_R$ and $O_T$
#301-304	$O_T$ are real world cases, which we have more interest in

Table III. The overview of OAEI benchmark tests

## 4. EVALUATIONS

To evaluate our approach we adopt benchmark tests from OAEI ontology matching (OM) campaign 2007. The reasons why we pick these tests are: 1. The annual OAEI OM campaign has become an authoritative contest in the area of ontology mapping, and thus attracts many participants including both well-known ontology mapping systems and new entrants. 2. The campaign provides uniform test cases for all participants so that the analysis and comparison between different approaches is practical. 3. The ground truth of benchmark tests is open. Thus we can use it to comprehensively evaluate different components of our approach.

### 4.1. Test Ontologies

Our test ontologies are benchmark tests ontologies from OAEI ontology matching campaign 2007, originating from the bibliography domain. The OAEI benchmark tests include one reference ontology dedicated to the very narrow domain of bibliography, multiple test ontologies manually discarding various information from the reference ontology in order to evaluate how algorithms behave when information is lacking, and 4 real world bibliographic ontologies that are generated by MIT, UMBC, University of Karlsruhe and INRIA respectively. More specifically, benchmark tests can be divided into 5 groups as shown in Table III. All benchmark tests can be downloaded at <http://oaei.ontologymatching.org/2007/benchmarks/>.

### 4.2. Evaluation Criteria

We follow the evaluation criteria used by the OAEI ontology matching campaign 2007. The evaluation is based on the results provided by participants. All results include a set of mapping pairs, which are expressed in the format as shown in Figure 2, where the element of Address in entity1 can be mapped to the element of Address in entity2, with the measure of 1.0 and “=” relation.

The OAEI benchmark tests are open tests, which mean the expected results are provided for all participants. The standard information retrieval evaluation measures, i.e., precision, recall and f-measure, are computed against the reference alignment. The precision, recall and f-measure are defined as follows.

$$Precision\ p = \frac{\#correct\_found\_mappings}{\#all\_found\_mappings}$$



```
<map>
  <cell>
    <entity1 rdf:resource='http://oaei.ontologymatching.org/2006/benchmarks/101/onto.rdf#Address'/>
    <entity2 rdf:resource='http://oaei.ontologymatching.org/2006/benchmarks/101/onto.rdf#Address'/>
    <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
    <relation>=</relation>
  </cell>
</map>
```

Figure 2. A sample of mapping pair

$$\text{Recall } r = \frac{\#correct\_found\_mappings}{\#all\_possible\_mappings}$$
$$F - \text{measure } f = \frac{2 \times p \times r}{p + r}$$

### 4.3. Experimental Design Motivation

Two experiments were designed. The motivation of them is:

- The 1st experiment investigates how the approach performs in the situation where people have manually marked some mapping results for a specific mapping task, but they need help from automatic mapping tools to find the rest of mappings.
- The 2nd experiment investigates whether a model trained on one mapping task can work on another mapping task(s). Moreover, we are interested in which benchmark test(s) is more suitable as a training model. The motivation for the 2nd experiment is: in most ontology mapping cases, no ground truth is available for a specific mapping task, but a general model has been learned that can be used to find mappings. Thus, to save users' time and effort, we want to find out mapping results using the existing model.

### 4.4. Experimental Methodology and Results

#### 4.4.1. 1st Experiment - Within-task

The 1st experiment investigates if machine learning methods work well in the situation where some manually marked mapping results are available for a specific mapping task. The motivation of the 1st experiment comes from the real world case: A user is working on an ontology mapping task with the help of computers. Since the size and complexity of the mapping task are large, the user can not manually find all mapping pairs. But fortunately the user is able to mark some of them (e.g., e1 maps to e2). Therefore, if we can utilize users' previous effort, we can help them find the rest of mappings by using some machine learning techniques. The methodology of the 1st experiment is:



1. For each OAEI benchmark test, we generate candidate mapping pairs by simply combine all elements from two ontologies.
2. For each mapping candidate, we mark down their correctness according to the reference alignment (i.e. the ground truth). Simultaneously we generate various features (i.e., linguistic, structural and web features) to describe the characteristics of the mapping pair.
3. We split all mapping pairs into two groups (i.e., one is for training purpose and the other is used as testing set) by randomly choosing (e.g. 50% vs. 50%). We train two SVM models (i.e., SVM-Class and SVM-Property) on training set using SVM-Light<sup>¶</sup> package.
4. We classify testing data on two models.
5. We extract mapping results of testing data using naive descendant extraction algorithm [26] and evaluate the results against reference alignment.
6. Finally to eliminate the bias caused by randomly choosing mapping pairs to generate training and testing data in step 3, we repeat step 3-5 10 times and report the average result as our final result. In the experiment, two SVM models (i.e., SVM-Class model for classes and SVM-Property model for properties) are trained separately due to the difference between the structure of classes and properties. As a result, the mapping pairs of classes are tested on SVM-Class model and the mapping pairs of properties are tested on SVM-Property model. Moreover, since the number of negative examples is much larger than the number of positive examples in training data, we use a fixed cost factor (i.e. 10) in SVM-Light to equalize the distribution and ensure training errors on positive examples outweigh those on negative examples.

Figure 3 shows the average f-measure of classes of each OAEI benchmark task tested on SVM-Class model. Figure 4 shows the f-measure of properties of each OAEI benchmark task tested on SVM-Property model, in which the f-measures of benchmark tests #226, #233-#237, #240-#247, #250, #254-#257, #260-#266 are 0, which is because there is no property existing for those tests. For comparison purpose both Figure 3 and Figure 4 include the f-measure of classes/properties running by PRIOR+ approach [28], a non learning based ontology mapping approach. The observations from Figure 3 and Figure 4 are:

1. On Test #101-#104 and #221-#247, both SVM-Class model and SVM-Property model perform as well as PRIOR+. This is because the linguistic information of these test ontologies is highly similar with that of the reference ontology and there is much less interference such as randomly generated name of classes/properties. Thus it is easy for both SVM-Class and SVM-Property model to catch useful features like edit distance that can contribute to learning models.
2. On Test #201-#210, both SVM-Class and SVM-Property model perform relatively worse than the PRIOR+ (especially on #201, #202, #208, #209). This is because the linguistic information changes too much on these tests so that it is hard to catch its linguistic and web characteristics in the training model. Meanwhile the structural feature is relatively weak.

---

<sup>¶</sup><http://svmlight.joachims.org/>

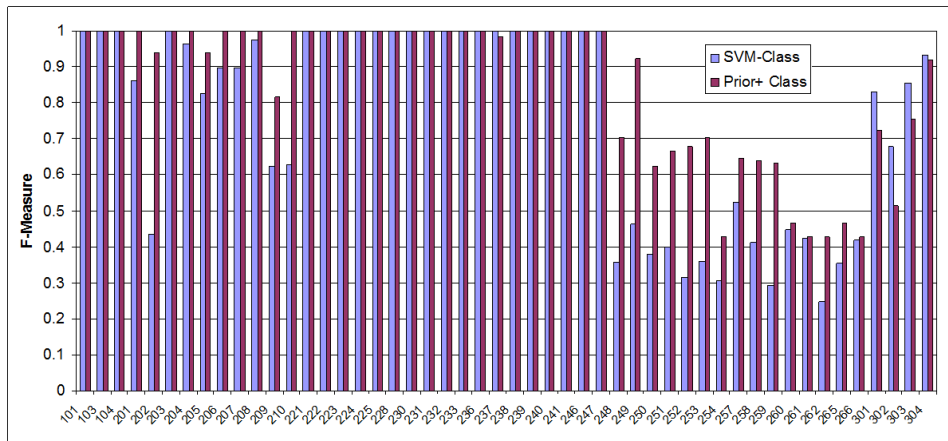


Figure 3. Results of classes on SVM-Class model on all benchmark tests (Within-task)

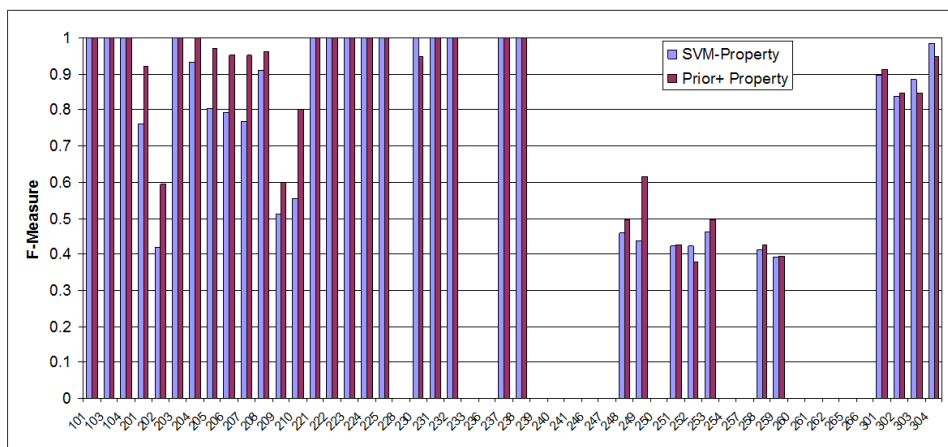


Figure 4. Results of properties on SVM-Property model on all benchmark tests (Within-task)

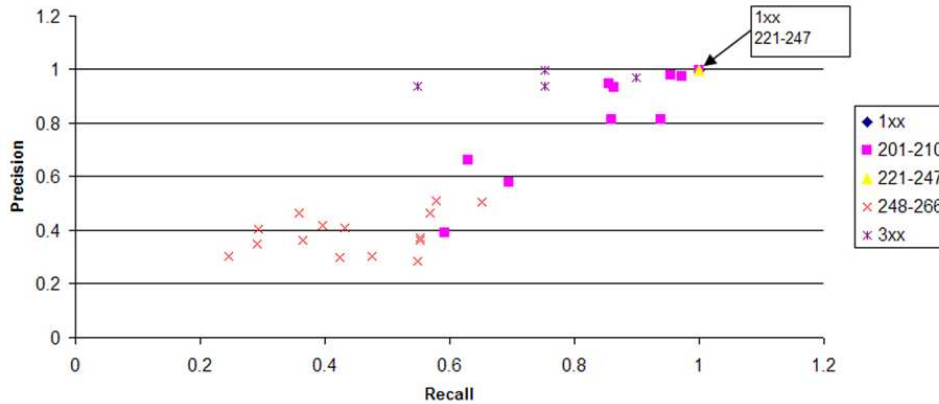


Figure 5. The precision-recall graph of SVM-Class model over all benchmark tests

- On Test #248-#266, both SVM-Class and SVM-Property model perform much worse than the PRIOR+. This is because there is no name and no comments in the test ontologies at all, i.e., both linguistic features and web features are totally unavailable. The only feature available for SVM models is structural, which is relatively weak. Meanwhile, the PRIOR+ benefits from the profile enrichment process that integrates instance information, which keeps all descriptive information, to both classes and properties.
- On real world cases #301-304, the SVM-Class model performs much better than the PRIOR+ and the SVM-Property model performs similarly as the PRIOR+ (i.e., slightly better on #301 and #302 but slightly worse on #303 and #304). The reason is our learning based approach utilizes Web feature to explore synonymous relations between concepts in ontologies. By contrast the PRIOR+ approach does not integrate any auxiliary thesaurus for such a purpose.

Furthermore, we investigate the distribution of the precision and recall of the SVM-Class model and SVM-Property over all benchmark tests. The results in Figure 5 and Figure 6 show that the precision and recall of test group (i.e., #1xx, #221-247) on SVM-Class model and SVM-Property model are close to each other. The precision and recall of #202, #209 and #210 are different from other tests in the group of #201-#210. This is because though the linguistic difference exists in all test ontologies in the group, the unavailability of the linguistic information on the comments of #202, #209 and #210 is more severe and thus be anomalous with others. The differences between the classes of #248-#266 and #3xx result in different precision and recall on SVM-Class model. Such differential is smaller on SVM-Property model due to the smaller differential between their properties.

To summarize, our conclusions on within-task experiment are: For learning-based approach (within-task), the performance is good when mapping task is relatively easy (i.e., #1xx and #221-247). When mapping task is more difficult, its performance is not as good as the PRIOR+

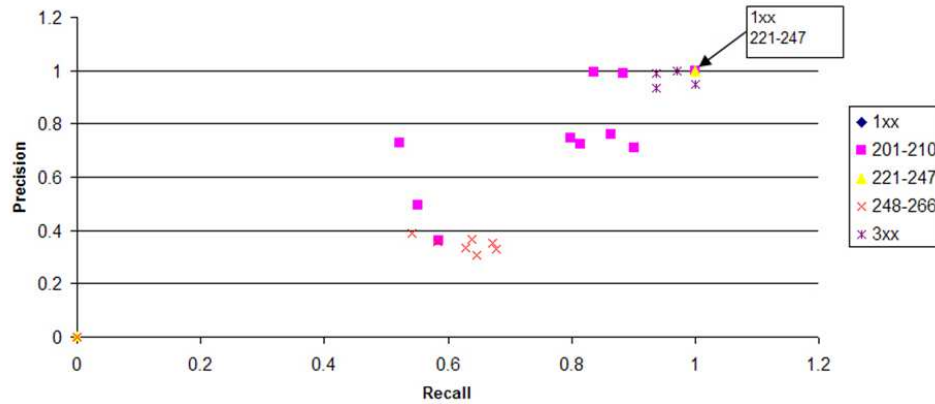


Figure 6. The precision-recall graph of SVM-Property model over all benchmark tests

approach (i.e., #201-#210 and #248-#266). But the performance of this approach is better than the PRIOR+ on real world cases, which shows the features used in this approach make more sense on real world cases than on artificially constructed cases.

#### 4.4.2. 2nd Experiment - Cross-task

The 2nd experiment investigates whether a model trained on one mapping task can work on another mapping task(s). Moreover, we are interested in which benchmark test(s) are more suitable as training models. The motivation for the 2nd experiment is based on the fact that in most ontology mapping cases, no ground truth is available. However we may have a model, which is already trained on another ontology mapping task. Thus, to save users' time and effort, we would like to find out mapping results using the existing model. The methodology of the 2nd experiment is:

1. Same as step 1 in 1st experiment.
2. Same as step 2 in 1st experiment.
3. We train two SVM models (i.e. SVM-Class and SVM-Property for each benchmark mapping task, except #228, #233, #236, #239-#247, #250, #254, #257, and #260-#266, using SVM-Light package. This is because no properties exist in these test ontologies, and thus no SVM-Property model can be trained on them. And thus it does not make sense to test mapping tasks with both classes and properties on the model trained without property.
4. We classify testing data of all the other benchmark tests (excluding the one that has been used in training model) using the SVM models.
5. We extract mapping results of using naive descendant extraction algorithm and evaluate the results against the reference alignment.

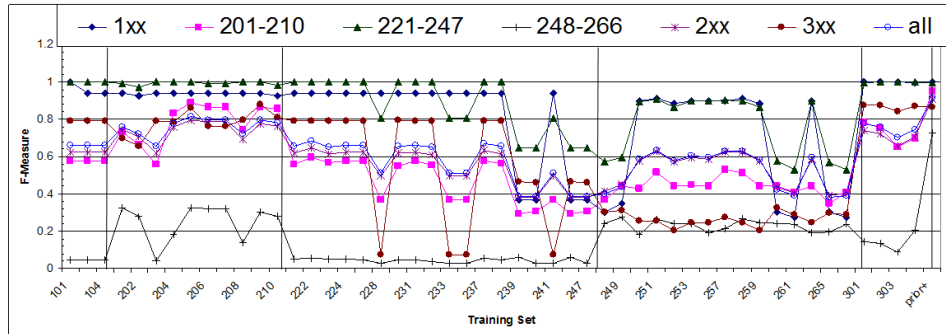


Figure 7. Testing results of benchmark tests (Cross-task)

6. Finally we repeat step 3-5 10 times and report the average f-measure of a group of testing data (e.g., #1xx, #2xx, #3xx etc.) on each training model as our final result.

Figure 7 shows the average f-measure tested on different data sets (i.e., all tests, #1xx, #2xx, #3xx, and more specific #201-#210, #221-#238, #248-#259). The observations from Figure 7 are:

1. Generally speaking, the PRIOR+ outperforms the learning-based approach (i.e., the cross-task) over all benchmark tests. The PRIOR+ holds the highest f-measure .92. Whereas no f-measure of any learning-based models is higher than .82. This suggests that none of OAEI benchmark tests is the best to be chosen to train the model that will then be used by all other benchmark tests to predict their mapping results. More specifically, training on #101-#104 and #221-#238 but testing on all other benchmark tests obtains worse results than training on other benchmark tests. This is because benchmark tests #1xx and #221-#238 are easy ontology mapping tasks. They have highly similar linguistic features, which leads the machine learning model to rely on some simple but specific features too much (e.g. the editdist feature), and thus it does not perform well when linguistic similarity changes a lot on other benchmark tests. For example, many benchmark tests have removed name and comments so that there is no linguistic information available at all. In this case, testing such data set on the model trained on #1xx and #221-#238 will hurt due to the significant difference between the characteristics of two mapping tasks. Training on #201-#210 and #301-#304, the f-measure is better than training on #101-#104 and #221-#238 due to the balance between different kinds of features in the training data. Finally the f-measure when training on #248-#259 is better than training on #1xx and #221-#238 but worse than training on #201-210 and #3xx. This is because #1xx and #221-#238 mapping tasks are too easy and simple to build a training model; meanwhile, #248-#259 have very limited linguistic information in themselves, which is unlike the diverse characteristics existing in #201-#210 and #3xx.



2. The f-measures when training on different benchmark tests but testing on #1xx are good except on #248-#259. The overall good performance on 1xx is because their simple characteristics are easy to catch in almost all training sets.
3. The f-measures when training on different benchmark tests but testing on #201-210 vary. Like the results over all benchmark tests, training on #101-#104 and #221-#238 obtains worse results than training on other benchmark tests because all these tests have highly similar linguistic features, which leads the machine learning model to rely on some simple but specific features too much (e.g. the editdist feature), and thus it does not perform well when linguistic similarity changes a lot on other benchmark tests. Furthermore, training on #201-#210 (except #203) and testing on themselves get the best f-measure compare to training on all other benchmark tests. We believe it is because the training data carry the same characteristics as the testing data most. However the performance of #203 is different to #201-#210 but similar to #1xx and #221-238 is because though no comments exist in #203 no change has been made to any name of its classes and properties or to its structure. Such characteristic makes #203 highly similar to #1xx and #221-#238 from both linguistic view and structural view. The f-measure when training on #248-#259 is better than the f-measure when training on #1xx and #221-#238 but worse than the f-measure when training on #201-210 and #3xx. This is because, on the one hand, #1xx and #221-#238 mapping tasks are too easy and too simple to build a training model; on the other hand, #248-#259 have very limited linguistic information in themselves, which is unlike the diverse characteristics existing in #201-#210 and #3xx. Finally, training on #301-#304, the f-measure is better than the f-measure training on all other benchmark tests except #201-210 themselves. This is because the diversity and balance of different kinds of features existing in #3xx.
4. The f-measures when training on each benchmark test but testing on #221-#247 are similar as that of #1xx. The overall good performance on #221-#247 is because their simple characteristics are easy to catch in almost all training sets.
5. The f-measures when training on each benchmark test but testing on #248-#259 is relatively bad. The f-measure when training on #1xx, #203, #221-#238, and #3xx but testing on #248-#259 is poor. This is because all these benchmark tests include more or less linguistic information. Compare to #248-#259, these benchmark tests are relatively simple and easy. Thus, training on relatively simple and intuitive mapping tasks will result in relatively simple and intuitive model, which is not suitable to complex situations. Whereas the f-measure when training on #201-#210 (except #203) and #248-#259 themselves is better than the f-measure when training on other benchmark tests due to the similarity between the training data and testing data.
6. Lastly, the f-measures when training on each benchmark test but testing on #301-#304 vary. The f-measure when training on #3xx themselves is the best compare to training on all other benchmark tests and the PRIOR+ approach except that the f-measure on #303 is slightly worse than on the PRIOR+. Training on #203 and #248-#259 obtained worst results in this case. This is because though #203 and #248-#259 are difficult mapping tasks their difficulties are manually made and are very different from what we can meet in real world cases. Therefore the model trained on these tests can not contribute to the real world cases. Finally our conclusions on cross-task experiment are: For learning-



based cross-task approach, the performance is good when training data and testing data share similar characteristics. If the testing mapping task is very simple, it's easy to catch characteristics in the training model and thus get good performance with more difficult training task. Meanwhile if both training and testing tasks are difficult but with different characteristics, the performance is not as good as other approaches.

## 5. RELATED WORK

Different approaches have been explored to solve ontology mapping problem such as heuristic and rule-based approach [9][20], graph-based approach [12][13], and machine learning approach [21][7][22]. In the following part we briefly review these three ontology mapping approaches and compare our approach with other machine learning approaches. Comprehensive surveys of the state-of-the-art ontology mapping approaches can be found in [23] and [24].

### 5.1. Heuristic and Rule-based Approach

Heuristic and rule-based approach usually uses various rules to exploit schema information for ontology mapping. Sample rules include “two elements match if they have the same name (allowing synonyms) and the same number of sub elements”, “similar-enough natural language definitions (comments) should also provide some evidence of concept similarity” and “when a concept can be aligned to more than one alternative, only those whose super-concepts are somewhere aligned to the super-concepts of the target concept are considered.” etc. Heuristic and rule based approach are relatively inexpensive and fast because it does not require training data and it operates on schema only instead of data instances. Moreover heuristic and rule base approach can quickly and concisely capture user knowledge such as phone number and zip codes. However this approach cannot exploit data information such as word occurrence frequency and distribution effectively and it can not take advantage of previous matching efforts as well. Especially it faces serious problem whenever no rule exists. For example, it is hard to distinguish movie description and user comments. Sample systems that apply heuristic and rule-based method include [9] and Prompt [20].

### 5.2. Graph-based method

Graph-based methods treat ontologies as graphs and compare the corresponding sub-graphs. Sample systems that apply graph method for ontology mapping include Anchor-Prompt [13] and Similarity Flooding [12]. The assumption in Anchor-Prompt is if two elements are similar and there are paths connecting them, then the elements in these paths are often similar as well. Though both linguistic and structural information are explored, it is time-consuming and requires equal-typed structure. For example, it can not handle mapping problem between two ontologies that have a flat structure in one and deep structure in the other. Similarity Flooding assumes that the similarity of two nodes depend on the similarity between their adjacent nodes. Though it is a generic graph matching algorithm, it only works for directed labeled graphs and relies on the contribution of the adjacency (i.e., if nodes are visually close



but structurally far away, it doesn't work). Like Anchor-Prompt, Similarity Flooding requires equal-typed structure in the to-be-mapped ontologies.

### 5.3. Machine Learning Based Approach

Machine learning approach usually uses various machine learning techniques to exploit both schema and data information. Compare to heuristic and rule based approach, machine learning based approach is efficient when the concepts in ontologies are associated with many instances, and it works better if many values of instances are text rather than references to other instances. Though machine learning approach can exploit data information and past matching activities, and figure out new rules by learning from training sets, it is usually time-consuming and requires training dataset. Moreover it faces difficulty in learning certain type of knowledge such as phone number and zip code. Sample systems that apply machine learning method include GLUE [21], APFEL [7], and PSN [22].

In GLUE [21], a well-known machine learning based ontology mapping system, to measure the similarity of concepts the author needs to calculate the joint probability distribution of the concepts that heavily rely on the availability of instance. However, in most cases instances are just unavailable or insufficient, and it is more common to have references between instances than text description. Furthermore, the target of the GLUE is every element in the target ontology, which makes the model unable to be generalized to any application where domain has changed. Therefore they need new training data to re-build the model for each domain, which is usually unavailable. GLUE is useful when ground truth mapping has been obtained between multiple ontologies and one public ontology, and all ontologies have enough instances to train machine learning model. Then the model could be applied when new ontology is to be mapped to the same public ontology. When there are no enough instances, this approach simply does not work. This limits the usage scope of GLUE. Our approach does not require instance level training data, is a good complement to those methods relying heavily on training data, and thus could be applied more widely, especially when enough training data is not available.

Another approach using machine learning techniques for ontology mapping is APFEL [7], an advanced version of QOM [19]. In APFEL, the authors focus on calculating various similarities based on expert encoded rules, while machine learning approach has been merely used to integrate all these similarity measures. As described in the paper the machine learning based similarity integration approach can easily be replaced by other methods. In the contrast, the function of machine learning in our approach is to calculate mapping similarity, which is the key of our approach and cannot be replaced.

A new neural network architecture PSN (Partial Shared Network) [22] is proposed to solve ontology mapping problem. The idea is to train multiple tasks (ontology structures) simultaneously on a partially shared feed forward network. Each ontology has its own input bank and output bank, middle part of the network is shared by all ontologies. The shared part of neural network is supposed to capture the mapping of ontology structures. After training, if an element from one ontology is presented at its input, correspondence in another ontology could be obtained by examining the output bank of the ontology. The limitation of the PSN is: 1. Only structure information is used to train the network. 2. The approach is tested on a



toy problem only, which does not have complex relationships and thus it is hard to predict its performance on real ontology mapping problem.

To summarize, our approach is a pure machine learning approach compared with APFEL; it can be used more widely than GLUE; and more mature than PSN.

## 6. CONCLUSIONS AND FUTURE WORK

In this paper, we examine a non-instance machine learning based ontology mapping approach, which overcomes the limitations of previous learning-based ontology mapping approaches that either rely on the availability of sufficient instances or are domain-dependent. In the approach we treat the ontology mapping problem as a binary classification problem; generate a number of generic features; utilize these features to build training model; and conduct two experiments to investigate the performance of machine learning techniques in different situations. We evaluate our approach on the well known OAEI benchmark tests. The experiment results show that our approach performs well on most of tests when training and testing on the same mapping task; and the results of approach vary according to the likelihood of training data and testing data when training and testing on different mapping tasks.

Future work may include: 1. Leverage different features so as to achieve a robust semantic similarity measure. 2. Do feature selection procedure by maximizing the f-measure. 3. Perform an active learning with Support Vector Machine algorithm, etc.

## ACKNOWLEDGEMENTS

We appreciate the comments from anonymous reviewers very much.

## REFERENCES

1. van Rijsbergen CJ. *Information Retrieval*. 2 edn., Butterworths: London, 1979.
2. Tang J, Li J, Liang B, Huang X, Li Y, Wang K. Using bayesian decision for ontology mapping. *Web Semant.* 2006; 4(4):243–262. doi:http://dx.doi.org/10.1016/j.websem.2006.06.001.
3. Berners-Lee T, Hendler J, Lassila O. The semantic web: Scientific american. *Scientific American* May 2001; (284(5)):34–43.
4. Gruber T. A translation approach to portable ontology specifications. *Knowledge Acquisition* 1993; 5(2):199–220.
5. Noy N. Semantic Integration: A survey of ontology-based approaches. *SIGMOD Record* 2004; 33(4):65–70.
6. Doan A, Halevy AY. Semantic-integration research in the database community. *AI Mag.* 2005; 26(1):83–94.
7. Zhuge H, Xing Y, Shi P. Resource space model, owl and database: Mapping and integration. *ACM Trans. Internet Technol.* 2008; 8(4):1–31. doi:http://doi.acm.org/10.1145/1391949.1391954.
8. Zhuge H. *The Web Resource Space Model*. Springer, 2008.
9. Hovy E. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. *Proc. of the First Int. Conf. on Language Resources and Evaluation (LREC)*, 1998; 535–542.
10. Mao M, Peng Y, Spring M. A harmony based adaptive ontology mapping approach. *SWWS*, Arabnia HR, Marsh A (eds.), CSREA Press, 2008; 336–342.
11. Qu Y, Hu W, Cheng G. Constructing virtual documents for ontology matching. *WWW*, Carr L, Roure DD, Iyengar A, Goble CA, Dahlin M (eds.), ACM, 2006; 23–31.
12. Melnik S, Garcia-Molina H, Rahm E. Similarity flooding: a versatile graph matching algorithm. *Proc. 18th International Conference on Data Engineering (ICDE)*, San Jose, CA, US, 2002; 117–128.



13. Noy NF, Musen MA. Anchor-prompt: Using non-local context for semantic matching. *In Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001; 63–70.
14. Mao M, Peng Y, Spring M. A profile propagation and information retrieval based ontology mapping approach. *SKG '07: Proceedings of the Third International Conference on Semantics, Knowledge and Grid*, IEEE Computer Society: Washington, DC, USA, 2007; 164–169, doi: <http://dx.doi.org/10.1109/SKG.2007.172>.
15. Mao M. Ontology mapping: An information retrieval and interactive activation network based approach. *ISWC/ASWC, Lecture Notes in Computer Science*, vol. 4825, Aberer K, Choi KS, Noy NF, Allemang D, Lee KI, Nixon LJB, Golbeck J, Mika P, Maynard D, Mizoguchi R, *et al.* (eds.), Springer, 2007; 931–935.
16. Mao M, Peng Y. An adaptive ontology mapping approach with neural network based constraint satisfaction. *Journal of Web Semantics* 2010; **8**(1):14–25, doi:10.1016/j.websem.2009.11.002.
17. Mao M, Peng Y, Spring M. Neural network based constraint satisfaction in ontology mapping. *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, AAAI Press, 2008; 1207–1212.
18. Mao M, Peng Y, Spring M. Integrating the iac neural network in ontology mapping. *WWW '08: Proceeding of the 17th international conference on World Wide Web*, ACM: New York, NY, USA, 2008; 1223–1224, doi:<http://doi.acm.org/10.1145/1367497.1367737>.
19. Ehrig M, Staab S. QOM: Quick ontology mapping. *Proceedings of the 3rd International Semantic Web Conference (ISWC)*, 2004; 683–697.
20. Noy NF, Musen MA. Prompt: Algorithm and tool for automated ontology merging and alignment. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press / The MIT Press, 2000; 450–455.
21. Doan A, Madhavan J, Dhamankar R, Domingos P, Halevy A. Learning to match ontologies on the semantic web. *The VLDB Journal* 2003; **12**(4):303–319, doi:<http://dx.doi.org/10.1007/s00778-003-0104-2>.
22. Peng Y, Munro PW, Mao M. Learning to map ontologies with neural network. *OM, CEUR Workshop Proceedings*, vol. 551, Shvaiko P, Euzenat J, Giunchiglia F, Stuckenschmidt H, Noy NF, Rosenthal A (eds.), CEUR-WS.org, 2009; 256–257.
23. Euzenat J, Barrasa J, Bouquet P, Dieng R, Ehrig M, Hauswirth M, Jarrar M, Lara R, Maynard D, Napoli A, *et al.* D2.2.3: State of the art on ontology alignment. *Technical Report*, NoE Knowledge Web project deliverable 2004. [Http://knowledgeweb.semanticweb.org/](http://knowledgeweb.semanticweb.org/).
24. Kalfoglou Y, Schorlemmer M. Ontology mapping: the state of the art. *The Knowledge Engineering Review* 2003; **18**(1):1–31.
25. Euzenat J, Shvaiko P. *Ontology Matching*. Springer-Verlag New York, Inc.: Secaucus, NJ, USA, 2007.
26. Meilicke C, Stuckenschmidt H. Analyzing mapping extraction approaches. *In Proc. of the ISWC 2007 Workshop on Ontology Matching, Busan, Korea*, 2007; 25–36.
27. Jones R, Rey B, Madani O, Greiner W. Generating query substitutions. *WWW '06: Proceedings of the 15th international conference on World Wide Web*, ACM: New York, NY, USA, 2006; 387–396, doi: <http://doi.acm.org/10.1145/1135777.1135835>.
28. Mao M, Peng Y. The prior+: Results for oaei campaign 2007. *OM, CEUR Workshop Proceedings*, vol. 304, Shvaiko P, Euzenat J, Giunchiglia F, He B (eds.), CEUR-WS.org, 2007; 219–226.
29. Bollegala D, Matsuo Y, Ishizuka M. Measuring semantic similarity between words using web search engines. *WWW '07: Proceedings of the 16th international conference on World Wide Web*, ACM: New York, NY, USA, 2007; 757–766, doi:<http://doi.acm.org/10.1145/1242572.1242675>.