A Concept Hierarchy based Ontology Mapping Approach

Ying Wang, Weiru Liu, and David Bell

School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast, BT7 1NN, UK
{ywang14, w.liu, da.bell}@qub.ac.uk

Abstract. Ontology mapping is one of the most important tasks for ontology interoperability and its main aim is to find semantic relationships between entities (i.e. concept, attribute, and relation) of two ontologies. However, most of the current methods only consider one to one (1:1) mappings. In this paper we propose a new approach (CHM: Concept Hierarchy based Mapping approach) which can find simple (1:1) mappings and complex (m:1 or 1:m) mappings simultaneously. First, we propose a new method to represent the concept names of entities. This method is based on the hierarchical structure of an ontology such that each concept name of entity in the ontology is included in a set. The parent-child relationship in the hierarchical structure of an ontology is then extended as a set-inclusion relationship between the sets for the parent and the child. Second, we compute the similarities between entities based on the new representation of entities in ontologies. Third, after generating the mapping candidates, we select the best mapping result for each source entity. We design a new algorithm based on the Apriori algorithm for selecting the mapping results. Finally, we obtain simple (1:1) and complex (m:1 or 1:m) mappings. Our experimental results and comparisons with related work indicate that utilizing this method in dealing with ontology mapping is a promising way to improve the overall mapping results.

1 Introduction

Research and development on ontology mapping (or matching) has attracted huge interests (e.g., [1–6]) and many mapping methods have been proposed. Comprehensive surveys on recent developments of ontology mapping can be found in [7, 8].

Considerable efforts have been devoted to implement ontology mapping systems, especially one to one mappings. However, complex mappings (m:1, 1:m and m:n) are also pervasive and important in real world applications. In [7], an example was given to illustrate the importance of complex mappings in schema mapping research. We believe that the same issue exists in ontology mapping. Therefore, it is very important to find simple and complex mapping results in a natural way.

To address this problem in this paper, we first propose a new method to represent entities in ontologies. Traditionally, the concept names of entities are used

directly. This representation method does not consider the hidden relationships between concept names of entities, so it cannot reflect the complete meaning of the concept names of entities. When computing the similarities between entities based on this representation method, the result is hardly accurate. So it is significant to have a better method to represent entities. In this paper, we propose a new representation method for entities. For the multi-hierarchical structure of ontology, we view it as a concept hierarchy. For the example given in Figure 1(a), we observe that for each concept (in this paper, *concept*, *concept node* and *entity* represent the same thing) in this concept hierarchy, its complete meaning is described by a set of concept names. In other words, there is a kind of *semantic inclusion relationship* among these concepts. For instance, a branch from **CS**, **Courses** to **Graduate Courses** in Figure 1(a), **CS** means the department of computer science, **Courses** means the courses offered by the department of computer science and **Graduate Courses** means **Graduate Courses** is a kind of **Courses** and is offered by the department of computer science, i.e. **CS**, so the semantics of **Courses** can be completed by extending **Courses** to **{CS, Courses}**. Identically, we can extend the concept **Graduate Courses** to **{CS, Courses, Graduate Courses}**. Actually, a branch from one concept node to the root node indicates a complete meaning for this concept node. So for any concept name of entity C in an ontology, we can represent it by a new method as follows. First, we find the branch which has the concept C. Second, we collect those concepts along the path between C and the root node to form a set. We use this new set to represent entity C.

Once each entity is represented by a set of words, we compute the similarities between entities. In this paper, we separate the similarity values into two types: one is the similarities between entities which belong to one ontology, another is the similarities between entities which belong to two different ontologies. Here, we choose the *Linguistic-based matcher* (which uses domain specific thesauri to match words) and the *Structure-based matcher* (which uses concept-hierarchy theory) to compute similarities (we utilize Linguistic-based matcher because the performance of this matcher is good for similar or dissimilar words. Please refer to [9] for details).

As a result, we obtain a set S_1 consisting of mapping candidates such that from each entity in ontology O_1, a similarity value is obtained for every entity in ontology O_2. Following this, we select the best mapping entity in O_2 for each entity in O_1 and these best mapping results constitute another set S_2. In S_2, we search all the mapping results to see if there exist multiple source entities in O_1 that are mapped to the same target entity in O_2. If so, we apply a new algorithm based on Apriori algorithm [10] to decide how many source entities in O_1 should be combined together to map onto the same entity in O_2. Our study shows that this method significantly improves the matching results as illustrated in our experiments.

The rest of the paper is organized as follows. Section 2 describes the similarity measures used. Section 3 illustrates how to select final mapping results by using our new algorithm. Section 4 gives the background information about the

experiments and the results. Section 5 discusses related work and concludes the paper with discussions on future research.

2 Ontology Mapping

Ontology mapping can be done based on similarities, so we need to leverage the degree of the similarity between any two entities no matter these entities are in one ontology or from two ontologies. In this section, we describe our notion to measure the similarity between entities in detail. In this paper, we use *concept node* to denote an *entity* in ontology and we compute the similarity between concept nodes to indicate the similarity between entities. We compute the similarity of two concept nodes, e_i and e_j, denoted as $Sim(e_i, e_j)$:

$$Sim(e_i, e_j) = \begin{cases} \omega_{ls} sim_{ls}(e_i, e_j) + \omega_{ss} sim_{ss}(e_i, e_j) & e_i, e_j \in \text{same ontology} \\ sim_{ls}(e_i, e_j) & e_i, e_j \in \text{different ontologies} \end{cases} \tag{1}$$

where ω_{ls} and ω_{ss} are two weight coefficients, reflecting the relative importance of the components. In our approach, we think both of the components are equally important, so we assign them both with coefficient 0.5. $sim_{ls}(e_i, e_j)$ and $sim_{ss}(e_i, e_j)$ denote the similarities obtained from the linguistic-based matcher and structure-based matcher respectively.

2.1 Extension for Concept Nodes

When using different methods to compute the similarity values between two names of entities in ontologies, such as *edit distance-based method* [11], or *Jaro-Winkler distance-based method* [12] etc, we discover that these methods are too simple to reflect the semantic relationship between those entities.

Example 1. Figure 1(a) provides a simple ontology which describes a department of computer science and shows its concept hierarchy. It is clear that if we only use one method (*edit distance-based method* or *Jaro-Winkler distance-based method*) to compute the similarity between those entities, such as "CS" and "People", we cannot obtain good results because these two names of entities are not very similar directly but are related indirectly.

As shown in Figure 1(a), we found that the hierarchical structure is very similar to the *concept hierarchies* in multi-hierarchy association rules mining [10]. In this kind of mining, a concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts. According to the concept hierarchies, "People" actually means "People in the department of Computer Science", i.e. "People" and "CS" should be denoted as: {CS People} and {CS} separately. So we can denote all the concept names of entities in an ontology by a new approach in terms of the *inclusion relationship* between these concept names from the root node to leaf nodes and then Figure 1(a) can be changed to Figure 1(b).

We now give precise similarity measures between entities. As stated above, concept names of entities have been expanded to concept sets, such as Figure

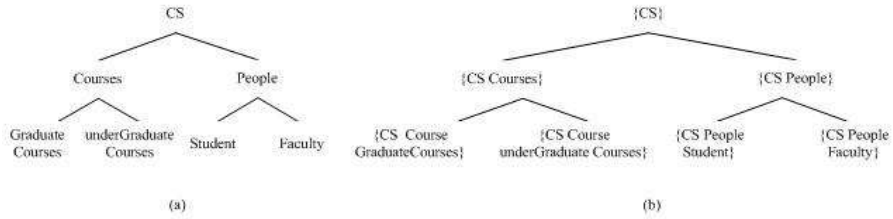


Fig. 1. (a) An ontology which describes a department of computer science; (b) A new method to represent the ontology where we expand all the concepts

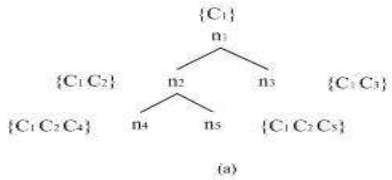


Fig. 2. An expanded ontology model where each node represents a concept

2, so we compute the similarity between any two sets by adopting a general method for computing similarities between composite words. We will introduce the method in the subsection: Calculating Similarities of Ontology Entities. So Equation(1) can be modified as:

$$Sim(E_i, E_j) = \begin{cases} \omega_{ls} sim_{ls}(E_i, E_j) + \omega_{ss} sim_{ss}(E_i, E_j) & E_i, E_j \in \text{same ontology} \\ sim_{ls}(E_i, E_j) & E_i, E_j \in \text{different ontologies} \end{cases} \tag{2}$$

where E_i, E_j are the concept sets for the concept nodes separately.

2.2 Linguistic-based Matcher

We employ the *Linguistic-based matcher* as our similarity measure and in our paper the linguistic similarity between two concept nodes is denoted as $sim(e_i, e_j)$. *Linguistic-based matcher* uses common knowledge or domain specific thesauri to match words and this kind of matchers have been used in many papers [13, 14].

The concept names of entities are usually composed of several words, so first we adopt Lin's matcher to compute the similarity between two words and then we use another method to compute the similarity between concept names based on the revised version of Lin's matcher.

Lin's Matcher: Lin's matcher is a kind of *Linguistic-based matcher*. In this paper, we use an electronic lexicon WordNet for calculating the similarity values between words. Lin in [15] proposed a probabilistic model which depends on corpus statistics to calculate the similarity values between words using the

WordNet. This method is based on statistical analysis of corpora, so it considers the probability of $word_1$ ($sense_1$) and $word_2$ ($sense_2$) and their most specific common subsumer $lso(w_1, w_2)$ appearing in the general corpus. However, since the words in given ontologies are usually application specific, this general corpus statistics obtained using the WordNet can not reflect the real possibility of domain-specific words. To improve Lin's method, we propose to calculate a punishment coefficient according to the ideas in the path length method [16]. The path length method regards WordNet as a graph and measures the similarity between two concepts (words) by identifying the minimum number of edges linking the concepts. It provides a simple approach to calculating similarity values and does not suffer from the disadvantage that Lin's method does, so we integrate Lin's method and a punishment coefficient to calculate the similarity values between words. First, we outline Lin's approach. The main formulas in this method are as follows: $sim_{Lin}(s_1, s_2) = \frac{2 \cdot \log(p(s_1, s_2))}{\log(p(s_1)) + \log(p(s_2))}$, $p(s) = \frac{freq(s)}{N}$ and $freq(s) = \sum_{n \in words(s)} count(n)$ where: $p(s_1, s_2)$ is the probability that the same hypernym of sense s_1 and sense s_2 occurs, $freq(s)$ denotes the word counts in sense s, $p(s)$ expresses the probability that sense s occurs in some synset and N is the total number of words in WordNet.

The punishment coefficient which is based on the theory of path length of WordNet is denoted as: $\frac{1}{2}\alpha^l$. Its meaning is explained as follows: α is a constant between 0 and 1 and is used to adjust the decrease of the degree of similarity between two senses when the path length between them is deepened and l expresses the longest distance either sense s_1 or sense s_2 passes by in a hierarchical hypernym structure. Because sense s_1 and sense s_2 occupy one of the common branches, this value has to be halved.

Therefore in our method, the similarity value calculated by Lin's method is adjusted with this coefficient to reflect more accurate degree between two senses s_1 and s_2. The revised calculation is:

$$sim_{new}(s_1, s_2) = \frac{2 \cdot \log(p(s_1, s_2))}{\log(p(s_1)) + \log(p(s_2))} \bullet \frac{1}{2}\alpha^l \tag{3}$$

Word w_1 and word w_2 may have many senses, we use $s(w_1)$ and $s(w_2)$ to denote the sets of senses for word w_1 and word w_2 respectively as $s(w_1) = \{s_{1i} \mid i = 1, 2, ..., m\}, s(w_2) = \{s_{1j} \mid j = 1, 2, ..., n\}$, where the numbers of senses that word w_1 and word w_2 contain are m and n. We then choose the maximum similarity value between two senses from the two sets of senses for words w_1 and w_2, so the similarity between words is: $sim(w_1, w_2) = \max(sim_{new}(s_{1i}, s_{2j}))$, $1 \leq i \leq m, 1 \leq j \leq n$

Calculating Similarities of Ontology Entities We compute similarities between names of ontology entities based on the word similarities obtained from the two matchers separately. The names of ontology entities are composed of several words, so we split a phrase (name of entity) and put the individual words into a set and then we deal with these words as follows: first, we calculate similarities of every pair of words within both sets by using one of the matchers

(Linguistic-based matcher or Structure-base matcher). After that, for each word in one set, compute similarity values between this word and every word from the other set and then pick out the largest similarity value. Finally attach this value to the word. Repeat this step until all of the words in the two sets have their own values. Finally, we compute the final degree of similarity of names using the sum of similarity values of all words from two sets divided by the total counts of all words.

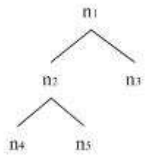
2.3 Structure-based Matcher

Ontology can be regarded as a model of multi-hierarchy, so in terms of the structure we propose a *Structure-based Matcher* which determines the similarity between two nodes (entities) based on the number of children nodes. We first introduce the method.

An ontology is usually designed in such a way that its topology and structure reflects the information contained within and between the concepts. In [17], Schickel-Zuber and Faltings propose the computation of the *a-priori score* of concept c, $APS(c)$, which captures this information. Equation (4) gives the definition of the a-priori score of concept c with n descendants as:

$$APS(c) = \frac{1}{n+2} \tag{4}$$

To illustrate the computation of the a-priori score, consider the simple example shown in Figure 3 where n_i represents the concept node in the ontology and N_d is the number of descendants for each concept node n_i. First, the number of descendants of each concept is computed. Then, Equation (8) is applied to compute the a-priori score of each concept in the ontology.



(a)

Concepts	Nd	APS(n_i)
n_4	0	1/2
n_5	0	1/2
n_2	2	1/4
n_3	0	1/2
n_1	4	1/6

(b)

Fig. 3. (a) An ontology model where each node represents a concept; (b) The a-priori scores for those concepts

It is very easy to find that the concept becomes more generalized as we travel up the ontology, and the a-priori score decreases due to the increasing number of descendants. That is the a-priori score reflects the information of each concept node, i.e., the higher score the concept has, the more information the concept

expresses. So it is possible to estimate the similarity between two concept nodes by finding the overlapping part of information between two concepts.

After obtaining the a-priori score for each concept node, we use the following definition to calculate the similarity as the structure-based matcher.

Given two a-priori scores $APS(n_i)$ and $APS(n_j)$ for two concept nodes n_i and n_j respectively, the similarity between n_i and n_j is defined as [17]:

$$sim_{ss}(n_i, n_j) = \frac{\min(APS(n_i), APS(n_j))}{\max(APS(n_i), APS(n_j))} \tag{5}$$

Example 2. From Figure 3(b), we can get the $APS(n_i)$ value for each node n_i and then we can compute the similarity between any two nodes. For instance, $sim_{ss}(n_1, n_4) = \frac{1/6}{1/2} = 1/3$.

3 Selection of the Best Mapping Results

For each entity e_i in O_1, we apply the linguistic-based matcher for computing the similarities between this entity and every member of O_2 and find the best mapping for this entity. Let S denote the set that contains the best mapping candidate in O_2 for every entity in O_1. In S, there may exist complex mapping results, i.e. several entities in O_1 map to the same entity in O_2. Our task is to decide where several entities in O_1 should be mapped to the same entity in O_2.

DCM framework [18] is a schema matching system and it is focused on dealing with matching multiple schemas together. In this framework, there is a AprioriCorrMining algorithm for discovering correlated items. In [18], correlated items are defined as the mapping results. This algorithm is to find all the correlated items with size $l+1$ based on the ones with size l in multiple schemas. It first finds all correlated two items and then recursively constructs correlated $l+1$ items from correlated l items. In this paper, our aim is to make sure if several entities in O_1 should be combined together to map to the same entity in O_2, so we regard the entities in O_1 as the items and attempt to find if they are correlated. We try to obtain the most correlated items directly, but AprioriCorrMining algorithm is not suitable for our objective, so we propose an improved algorithm named RevisedAprioriMining based on AprioriCorrMining algorithm.

First, for each entity of O_2 in set S, we collect its mapping entities of O_1 and input these source entities and use the RevisedAprioriMining to find whether these entities are really correlated and can be combined together to map one entity in O_2. As shown in Algorithm 1, first we find the incorrelate entities in V based on the similarities between them and store them into X (Line 4-8). Next, for each item in X, we have to construct different entities groups in which two entities of one item in X can not happen together (Line 9-16). When this algorithm is complete, we obtain a set V that stores the entity groups. Each entity group is a different combination of correlated entities. We search the set V to find the largest entity groups (in terms of cardinality). Since there may exist more than one such group, i.e. the number of entities in these groups are

the same, we select one such group by using the formula below:

$$G_e = \arg max_{k=1}^{l}(\sum_{i=1}^{n} sim(e_i, e_j)), e_i \in O_1, e_j \in O_2 \quad (6)$$

where G_e denotes the entity group which stores the combined entities, l is the number of entity groups in V and n is the number of entities in each group.

Algorithm 1 REVISED APRIORI MINING:

Input: Input entities in O_1: $Z = \{e_1, e_2, ..., e_n\}$, Threshold T
Output: Combined entity groups $V = \{V_1, V_2, ..., V_m\}$

$X \leftarrow \emptyset$
Create two queues $A \leftarrow \emptyset, V \leftarrow \emptyset$
$V = V \cup \{Z\}$
for all $e_i, e_j \in Z, i \neq j$ **do**
  if $sim(e_i, e_j) < T$ **then**
    $X \leftarrow X \cup \{\{e_i, e_j\}\}$
  end if
end for
for each item $\{e_i, e_j\} \in X$ **do**
  $A = V, V = \emptyset$
  for each set V_s in A **do**
    $A = A \backslash \{V_s\}$
    Remove e_i and e_j respectively from V_s, then V_s is changed into two different sets V_p and V_q
    $V = V \cup \{V_p\}$, $V = V \cup \{V_q\}$
  end for
end for
return V

4 Experiments

4.1 Dataset

We have implemented our approach in Java and now we present the experimental results that demonstrate the performance of our methods using the OAEI 2007 Benchmark Tests. In our experiments, we only focus on classes and properties in ontologies.

Generally, almost all the benchmark tests in OAEI 2007 describe Bibliographic references except Test 102 which is about wine and it is totally irrelevant to other data. We choose twenty-five test data for testing. All of these twenty-five test data can be divided into four groups [19] in terms of their characteristics: Test 101-104, Test 201-210, Test 221-247 and Test 301-304. A brief description is given below.

- **Test 101-104**: These tests contain classes and properties with either exactly the same or totally different names.
- **Test 201-210**: The tests in this group change some linguistic features compared to Test 101-104. For example, some of the ontologies in this group have no comments or names, names of some ontology have been replaced with synonyms.
- **Test 221-247**: The structures of the ontologies have been changed but the linguistic features have been maintained.
- **Test 301-304**: Four real-life ontologies about BibTeX.

In our evaluation, we take **Test 101** as the reference ontology. All the other ontologies are compared with **Test 101**.

4.2 Comparison of Experimental Results

We now compare the outputs from our system (denoted as CHM) to the results obtained from *ASMOV, DSSim, TaxoMap* and *OntoDNA* algorithms which were used in the 2007 Ontology Alignment Contest [1], and there are fifty tests totally. However, in some of ontologies, the names of entities are represented in scramble or in French, so the similarities between the names of entities can not be computed by our linguistic-based matcher. We ignore the comparisons of these ontologies. The details of experimental results are given in Table 1. In Table 1, p for precision, r for recall, f for f-measure, *Best* and *Worst* denote the values $\frac{f(SIM)}{f(Best\ or\ Worst)}$ between CHM and *Best* system or *Worst* system in one row. If the value equals to 1, it means these systems obtain the same results. If the value is smaller than 1, it means CHM presents worse than other systems. Otherwise, it shows CHM is better than others.

Table 1. Comparison of experiment results

Groups	Datasets	CHM			ASMOV			DSSim			TaxoMap			OntoDNA			Best	Worst
		p	r	f	p	r	f	p	r	f	p	r	f	p	r	f	f	f
Test 101-104	101	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	1	1
	103	100	100	100	100	100	100	100	100	100	100	34	51	94	100	97	1	1.96
	104	100	100	100	100	100	100	100	100	100	100	34	51	94	100	97	1	1.96
Test 201-210	203	100	100	100	100	100	100	100	100	100	NaN	0.00	NaN	94	100	97	1	∞
	204	86	84	85	100	100	100	96	91	93	92	24	38	93	84	88	0.85	2.24
	205	47	44	46	100	100	100	94	33	49	77	10	18	57	12	20	0.46	2.56
	208	86	83	85	100	100	100	95	90	92	NaN	0	NaN	93	84	88	0.85	∞
	209	49	41	45	92	90	91	91	32	47	NaN	0	NaN	57	12	20	0.49	∞
Test 221-247	221	82	82	82	100	100	100	100	100	100	100	34	51	93	76	83	0.82	1.61
	222	89	92	91	100	100	100	100	100	100	100	31	47	94	100	97	0.91	1.94
	224	100	100	100	100	100	100	100	100	100	100	34	51	94	100	97	1	1.96
	225	100	100	100	100	100	100	100	100	100	100	34	51	94	100	97	1	1.96
	228	100	100	100	100	100	100	100	100	100	100	100	100	53	27	36	1	2.78
	230	73	90	81	99	100	99	97	100	98	100	35	52	91	100	95	0.82	1.56
	231	100	100	100	100	100	100	100	100	100	100	34	51	94	100	97	1	1.96
	232	82	82	82	100	100	100	100	100	100	100	34	51	93	76	84	0.82	1.61
	233	52	52	52	100	100	100	100	100	100	100	100	100	53	27	32	0.52	1.63
	236	100	100	100	100	100	100	100	100	100	100	100	100	53	27	32	1	3.13
	237	93	97	95	100	100	100	100	100	100	100	31	47	94	100	97	0.95	2.02
	239	88	100	94	97	100	98	97	100	98	100	100	100	50	31	38	0.94	2.47
	241	58	58	58	100	100	100	100	100	100	100	100	100	53	27	32	0.58	1.81
	246	88	100	94	97	100	98	97	100	98	100	100	100	50	31	38	0.94	2.47
Test 301-304	301	43	45	44	93	82	87	82	30	44	100	21	35	88	69	77	0.51	1.26
	302	34	53	42	68	58	63	85	60	70	100	21	35	90	40	55	0.6	1.2
	304	51	49	50	95	96	95	96	92	94	93	34	50	92	88	90	0.53	1

Overall, we believe that the experimental results of our system are good. Although on individual pair of ontologies, our results are less ideal than the *ASMOV* system and *DSSim*, however, our results are better than *TaxoMap* system and *OntoDNA* system on most pairs of matching. The performances of these three different approaches, i.e., *ASMOV, DSSim* and our system *CHM* are good for almost the whole data set from Test 101 to Test 246, but our system does not perform well for Test 205, Test 209, Test 233 and Test 241. The performance of all these five systems are not very good for the data set from Test 301 to Test 304. Below we analyze the reasons for this.

[1] http://oaei.ontologymatching.org/2007/results/

One-to-one Mapping Results

- More effective results:
  - The two ontologies to be matched contain classes and properties with exactly the same names and structures, so every system that deploys the computation of similarities of names of entities can get good results, for instance, Test 101 vs 103 and vs 104.
  - Most of the results of Test 221-247 are good because the linguistic features have been maintained. However, the structures of them have been changed, so the performance of our system has been affected.
- Less effective results:
  - Test 201-210 describe the same kind of information as other ontologies, i.e. publications, however, the class names in them are very different from those in the reference ontology Test 101, especially Test 205 and 209, so our system does not obtain good results.
  - Our method is based on the hierarchical structure of an ontology, but for Test 233 and Test 241, these two ontologies have only one layer. When computing the similarity between two concepts in Test 233 and Test 101, such as **MastersThesis** in Test 233 and **MastersThesis** in Test 101. First, our method extends **MastersThesis**. Test 233 only has one layer, so **MastersThesis** can not be changed. Test 101 has three layers, so **MastersThesis** is extended to **{MastersThesis, Academic, Reference}**. The similarity value is reduced and does not reflect the true similarity between these two concepts.

Table 2. Comparison of complex mapping results

Group	Datasets	CHM		
		p	r	f
Test 301-304	301	55	55	55
	302	71	42	53
	304	33	50	40

Complex Mapping Results In Test 301-304, there exists inclusion relationships between entities, for example, **Collection<Book**. Several source entities have the inclusion relationships to one target entity separately, so we take this mapping as complex (m:1) mapping.

Tests 301-304 are real-life BibTeX ontologies which also include different words compared to Test 101 describing publications so the results are similar to Test 205, so we do not get good similarity results from this data set. However we still find some complex mappings (m:1) by using our algorithm to discover the best mapping results, such as for Test 302 vs Test 101, we get **{Collection, Monograph, Book}** mapping to **Book**.

5 Related Work and Conclusion

Related Work: Ontology matching is important and has received significant attention. However, existing matching methods mostly focus on simple (1:1) mappings. Here we present some related work on complex matching.

RiMOM [20] is a general ontology mapping system based on Bayesian decision theory and the approach divides the process of discovering complex mapping (m:1 mapping) into two steps: mapping entities discovery and mapping expression discovery. We mainly focus on mapping entities discovery. In mapping entities discovery, the system aims at finding whether there are multiple source entities mapped onto one target entity. If there are multiple source entities that are mapped onto one target entity, then it will combine those source entities. It does not consider any hidden relationship between those source entities, so it may cause wrong complex mappings.

PBM [21] and BMO [22] are two approaches to focusing on complex mappings so far. PBM is a method for partition-based block matching that is practically applicable to large class hierarchies. It first partitions the two large class hierarchies into blocks separately and then constructs the mappings between these blocks. It does not consider the relationship between the classes which belong to two large hierarchies respectively. It partitions these two hierarchies separately instead of partitioning one hierarchy by referring another one. This may produce wrong mappings. It is not always reasonable to partition a hierarchy itself into blocks, so this is one of the reason that it can not obtain very good results.

BMO tries to utilize a partition method to handle complex mappings just like PBM. But it is different on the process of partition. BMO puts the two ontologies together for partitioning and then obtains the block mapping results directly. It considers the correspondence between different ontologies and avoids too much manual intervention. BMO is a method that implements complex matching, but it failed to consider many issues, such as if it finds multiple complex mapping results, then what method should be used to combine them, etc.

Conclusion: In this paper, we proposed a new representation for concepts in ontology and then utilized two matchers (*Linguistic-based matcher* and *Structure-based matcher*) to deal with ontology mapping. In the *Linguistic-based matcher*, we improved Lin's method which computes similarity values between words. In the *Structure-based matcher*, we adopted the structure of ontology to calculate the similarity between two entities. Following this, we investigated how we can obtain reasonable ontology mapping results. We apply our new algorithm to search complex ontology mapping (m:1 mapping) from a set of ontologies used for ontology mapping competitions. The experimental results demonstrated that it is efficient and feasible for dealing with ontology mapping.

References

1. Ehrig, M., Sure, Y.: Ontology mapping - an integrated approach. In Proceedings of the 1st European Semantic Web Symposium (ESWS'04). (2004) 76–91
2. Ehrig, M., Staab, S.: Qom - quick ontology mapping. In Proceedings of the 3rd International Semantic Web Conference (ISWC'04). (2004) 683–697
3. Noy, N.F., Musen, M.A.: Prompt: Algorithm and tool for automated ontology merging and alignment. In Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI'00). (2000) 450–455

4. Noy, N.F., Musen, M.A.: Anchor-prompt: Using non-local context for semantic matching. In Workshop on Ontologies and Information Sharing at the 17th International Joint Conference on Articial Intelligence (IJCAI'01). (2001)
5. Kalfoglou, Y., Schorlemmer, W.M.: Information-flow-based ontology mapping. In Proceedings of the International Federated Conferences (CoopIS/DOA/ODBASE'02). (2002) 1132–1151
6. Su, X., Gulla, J.A.: Semantic enrichment for ontology mapping. In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB'04). (2004) 217–228
7. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. Journal of VLDB. **10**(4) (2001) 334–350
8. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal of Data Semantics **4** (2005) 146–171
9. Wang, Y., Liu, W., Bell, D.: Combining uncertain outputs from multiple ontology matchers. In Proceedings of the 1st International Conference on Scalable Uncertainty Management (SUM'07). (2007) 201–214
10. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. (2000)
11. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys **33**(1) (2001) 31–88
12. Winkler, W.E.: The state of record linkage and current research problems Statistical Society of Canada, Proceedings of the Survey Methods Section (1999) 73–80
13. Tang, J., Liang, B., Li, J.: Multiple strategies detection in ontology mapping. In Proceedings of the 14th international conference on World Wide Web (WWW'05) (Special interest tracks and posters). (2005) 1040–1041
14. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01). (2001) 49–58
15. Lin, D.: An information-theoretic definition of similarity. In Proceedings of the 15th International Conference on Machine Learning (ICML'98). (1998) 296–304
16. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of 14th International Joint Conference for Artificial Intelligence (IJCAI'95). (1995) 448–453
17. Schickel-Zuber, V., Faltings, B.: OSS: A semantic similarity function based on hierarchical ontologies. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07). (2007) 551–556
18. He, B., Chang, K.C.: Automatic complex schema matching across Web query interfaces: A correlation mining approach. ACM Transactions on Database Systems **31**(1) (2006) 346–395
19. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In Proceedings of 15th World Wide Web Conference (WWW'06). (2006) 23–31
20. Tang, J., Li, J., Liang, B., Huang, X., Li, Y., Wang, K.: Using Bayesian decision for ontology mapping. Journal of Web Semantics **4**(4) (2006) 243–262
21. Hu, W., Zhao, y., Qu, y.: Partition-based block matching of large class hierarchies. In Proceedings of the 1st Asian Semantic Web Conference (ASWC'06) (2006) 72–83
22. Hu, W., Qu, Y.: Block Matching for Ontologies. In Proceedings of the 5th International Semantic Web Conference (ISWC'06). (2006) 300–313