A novel algorithm for ontology matching

Ismail Akbari and Mohammad Fathian

Journal of Information Science 2010; 36; 324 originally published online Mar 19, 2010;
DOI: 10.1177/0165551510361432

A novel algorithm for ontology matching

Ismail Akbari and Mohammad Fathian

Industrial Engineering Department, Iran University of Science and Technology, Tehran, Iran

Abstract.

Ontology matching is an essential aspect of the Semantic Web with a goal of finding alignments among the entities of given ontologies. Ontology matching is a necessary step for establishing interoperation and knowledge sharing among Semantic Web applications. In this study we present an algorithm and a tool developed based on this algorithm to find correspondences among entities of input ontologies. The proposed algorithm uses a new lexical similarity measure and also utilizes structural information of ontologies to determine their corresponding entities. The lexical similarity measure generates a bag of words for each entity based on its label and description information. The structural approach creates a grid for each node in the ontologies. The combination of lexical and structural approaches creates the similarity matrix between the source and target ontologies. The proposed algorithm was tested on a well known benchmark and also compared to other algorithms presented in the literature. Our experimental results show the proposed algorithm is effective and outperforms other algorithms.

Keywords: lexical similarity; ontology matching; Semantic Web; similarity measure; structure similarity

1. Introduction

The great success of the current world wide web leads to a new challenge: a huge amount of data is being interpreted by humans only with limited machine support. Berners-Lee, inventor of the web, suggests enriching it by machine-processable information which supports the users in their tasks. For instance, today's search engines are already quite powerful, but still too often return excessively large or inadequate lists of hits. Machine-processable information can lead the search engine to the relevant pages and can thus improve both precision and recall [1]. Here, the Semantic Web comes into play. It stands for a vision in which computers can find, read, understand, and use data over the world wide web; much the same way as people do to accomplish users' goals in a more effective way.

The Semantic Web is supposed to make data located anywhere on the web accessible and understandable to people and machines. It aims at exploiting formal knowledge at the world scale. It is, particularly, based on ontologies. Ontologies are shared models of a domain that encode a view which is common to a set of different parties [2]. Web ontologies written by RDF Schema [3] or OWL [4] play a crucial role in the Semantic Web and form its backbone. Ontology matching has been proposed as a possible solution for the knowledge sharing and reuse, by providing a formal mechanism

Correspondence to: Ismail Akbari, Iran University of Science and Technology, Tehran, Iran.
Email: Akbari@ind.iust.ac.ir

for defining the semantics of data. It plays a key role in expansion and utilization of Semantic Web-based applications [5–7]. In recent years a lot of ontologies have been created in different domains. Ontology matching allows data and knowledge expressed in different languages and formats to be shared. It usually takes two ontologies as input and generates an alignment as output. The alignment is a set of correspondences between semantically related entities such as named classes, object properties, and data properties of input ontologies.

A more formal definition can be stated as follows [8]: "Given two ontologies O and O', an alignment (matching) between O and O', is a set of correspondences (i.e. 4-tuples) $<e, e', r, n>$ with $e \in O$ and $e' \in O'$ being the two matched entities, r being a relationship holding between e and e', and n expressing the level of confidence [0..1] in this correspondence".

In this paper we present a combinatorial approach for finding correspondences among entities (i.e. named classes, object properties and data properties) of given ontologies based on their structural and lexical similarities. First, we determine lexical similarities among named classes (nodes in ontology), object properties (edges in ontology) and data properties (attributes) using a new distance measure which creates a bag of words for each entity in the given ontologies. The bags are then compared to each other to calculate the lexical similarity between two entities with the same type (e.g. two named classes) from two ontologies. So, using this distance measure we obtain three different lexical matrices which respectively correspond to the similarities of named classes, object properties and data properties of the source and target ontology. Second, we find similarities among nodes (named classes) of source and target ontologies based on their ontology structures. In other words, we create a grid for each node using its neighbours and neighbours of its neighbours in the source and target ontology to structurally compare them. Finally, in this phase, the three lexical and one structural similarity matrices obtained in the previous phases will be combined using an additional technique to generate the overall similarity matrix.

The rest of the paper is organized as follows. Section 2 reviews related works. In Section 3 we discuss the proposed algorithm in detail. Section 4 provides an illustrative example. Experimental results are presented in Section 5 and also the tool that was developed based on this algorithm is described. Section 6 includes conclusions and remarks.

2. Related work

Several methods have been proposed for finding alignments among entities of ontologies [9–27]. In this section we review the approaches that have been proposed for ontology matching. These approaches can be split into four categories: lexical, semantic, structural and combinatorial.

Lexical approaches are string-based methods for identifying similar entities in given ontologies. For example, these may be used to identify identical classes in the source and target ontologies based on the similarity of their label or description [9–13]. These techniques consider strings as sequences of letters. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts. A comparison of different string matching techniques, from distance functions to token-based functions can be found in [14]. Some examples of string-based techniques which are extensively used in matching systems are prefix, suffix, edit distance, and n-gram [9].

Prefix and suffix approaches take as input two strings and check whether the first string starts (ends) with the second one. These approaches are efficient in matching cognate strings and similar acronyms (e.g. int and integer) [15–18]. Edit distance approaches take two strings as input and compute the edit distance between them. Edit distance is the number of insertions, deletions and substitutions of characters required to transform one string into another, normalized by the length of the longer string. For example, the MLMA+ algorithm [19] uses Levenshtein distance [20] as an edit distance to compute the lexical similarity between two entities. N-gram-based approaches take two strings as input and compute the number of common n-grams (i.e. sequences of n characters) between them [15, 16]. For example, 3-grams for the string 'nikon' are 'nik', 'iko', 'kon'. Thus, the distance between 'nkon' and 'nikon' based on 3-grams would be 1/3.

In semantic approaches usually one or more linguistic resources such as lexicons and thesauri are used to identify synonym entities [9, 11–13]. These approaches usually use a common knowledge or a domain-specific thesaurus to match words based on linguistic relations between them (e.g. synonyms, hyponyms). In this case, the names of ontology entities are considered as words of a natural language. Some approaches use common knowledge thesauri to obtain the meanings of terms used in ontologies. For example, WordNet [21] is an electronic lexical database for English (and other languages), where various senses of words are put together into sets of synonyms. Relations between ontology entities can be computed in terms of bindings between WordNet senses [22–24]. Other approaches use domain-specific thesauri which usually store some specific domain knowledge, which is not available in the common knowledge thesauri (e.g. proper names), as entries with synonym, hypernym and other relations [17].

Structural approaches identify similar classes (nodes) by looking at their links to other classes from the same ontology and also their properties. The main idea is that two classes of the source and target ontologies are similar if they have the same neighbours (structures) and the same attributes [8, 12]. For example, GMO [7] is a structural algorithm which uses bipartite graphs to represent ontologies. It measures the structural similarity between graphs by a new measurement. However, GMO takes a set of matched pairs, which are typically previously found by other approaches, as an external input in its matching process. Some other structural matchers that are used to compare ontologies are matchers based on children, leaves and relations. In children and leaves matchers, two non-leaf entities are considered as similar if they have highly similar children or leaves, respectively [15, 17]. In relation matchers, similarity computation between nodes can also be based on their (object property) relations [25, 26].

Combinatorial approaches mix up two or more of the above-mentioned approaches (i.e. combination of lexical, semantic and structural approaches) to get better results. The MLMA+ algorithm [19] and its improved version [27] are combinatorial approaches which use a neighbour search technique that works at two levels. At the first level, two lexical similarity measures are applied to the input ontologies. These measures are the name similarity measure which uses Levenshtein distance [20] and the linguistic similarity measure which uses WordNet [21]. At the second level, the algorithms will apply a structural similarity measure to find the best possible matching solution.

3. A combined approach to align ontologies

In this section we describe our combinatorial method for matching ontologies based on their lexical and structural similarities. The proposed method comprises three phases to find alignments between two given source and target ontologies. At the first and second phases, the input ontologies are lexically and structurally matched, respectively. Then at the third phase the initial results obtained from the previous phases are combined to generate the overall results. The details of these three phases are explained in subsequent subsections.

3.1. Lexically matching entities between two ontologies

Lexical similarity matching approaches are string-based methods for identifying similar entities in given ontologies. Here, we separately find the lexical similarity among entities (named classes, object properties and data properties) of the source and target ontologies. Therefore, this phase will produce three distinct lexical similarity matrices as its output.

We introduce a new similarity distance measure to determine the lexical similarity of input ontologies. Assume that we want to calculate lexical similarity between two strings s and t. These strings can be a single word or a text containing several statements. At first, we tokenize each string using delimiters and then convert it to a bag of words. Any non-alphabetical character in the given strings will be considered as a delimiter. For example, if string s contains two non-alphabetical characters then we consider these two as delimiters and remove them from string s. As a result, s will be tokenized into three words. After converting each of the strings s and t to a bag of words, every word that is common to the two bags will be removed from both of them. After that, if there is nothing left in the first and second bags, the distance between the two input strings will be zero. Otherwise, all characters left in each bag will be concatenated and the Levenshtein similarity distance [20] is calculated between the two resulting words. After calculating the distance between strings *s* and *t*, their similarity will be obtained from the following equation:

$$Lexical\_Similarity(s, t) = 1 - {distance(s,t)}/{\max\_len(s,t)} \qquad (1)$$

where *distance* is the distance between strings *s* and *t* as described above, and *max_len* is the maximum of lengths of strings *s* and *t*.

Consider the following example:

s = *"Part Of"*
t = *"is_part _of"*
bag_of_words(*s*) = {*"Part", Of"*}
bag_of_words(*t*) = {*"is", "part", "of"*}

After creating the bags of words, we remove two words *"part"* and *"of"* from both bags. So, we will have the following bags:

bags_of_words (*s*) = {}
bags_of_words (*t*) = {"is"}

Finally, similarity between strings *s* and *t* will be:

$$Levenshtein\_distance("", "is") = 2$$

$$Lexical\_similarity(s, t) = 1 - \left(\frac{distance(s,t)}{max\_len(s,t)}\right) = 1 - \left(\frac{Levenshtein\_distance("","is")}{max(7,10)}\right) = 1 - {2}/{10} = 0.80$$

We separately calculate lexical similarity among named classes, object properties and data properties of the two input ontologies using the aforementioned measure and then generate three distinct lexical similarity matrices.

3.2. Structurally matching entities between two ontologies

In this section we show how corresponding nodes are found in input ontologies, based on their structure. Figure 1 shows the flowchart of this approach. Below are the steps to create the structural similarity matrix between source and target ontologies:

1. Create a neighbour matrix for each ontology.
2. Make an array of linked lists for each ontology based on its neighbour matrix.
3. Calculate structural similarity among nodes of the source and target ontologies using their linked lists and generate an initial structural similarity matrix.
4. Improve the initial structural similarity matrix using three additional operations.

In the following the above steps are described in detail. To compare input ontologies structurally, we generate a grid for each node of each ontology. These nodes will be compared with one another based on their grids. This grid is modelled using an array of linked lists (i.e. a two-dimensional array). The important aspect of this approach is the number of neighbours a node has and the way they are related to each other and to this node (see Figure 2).

We consider two nodes in an ontology as neighbours if they are related to each other through one of ‘is-a’ (subclass or superclass), ‘equivalent to’ or ‘disjoint with’ relations or through an object property relation. For example, if class A is a subclass of class A', then A and A' are neighbours. For another example, given an object property p, its domain and range nodes will be neighbours. Based on this assumption, we calculate a neighbour matrix for any given ontology. Each element of the neighbour matrix is either 1 or 0 which shows that its row and column nodes are or are not neighbours, respectively.

Now for each node in the source or target ontology we calculate a grid using the neighbour matrix of its ontology. This gird is represented by an array of linked lists. Each row in the neighbour matrix of an ontology corresponds to a node of that ontology. The number of 1’s in each row shows the number of neighbours which that node has. Also if, for example, node A in a neighbour matrix has n neighbours then its corresponding array of linked lists will have n rows. Each row (linked list) in this array represents neighbours of that neighbour.

In other words, the first column of *A′s* grid shows its neighbours and also how many neighbours any of these neighbours have (see the first column of the matrix in Figure 3(b)). The j th row of *A′s* grid (for $j = 1,2,\ldots,n$) corresponds to the j th neighbour of A and also shows the neighbours of that neighbour and how many neighbours each one has. See the rows of the matrix in Figure 3(b).

In the third step we calculate the initial structural similarity matrix between input ontologies by using their array of linked lists. Each element of the matrix corresponds to the degree of structural similarity between a node from the source ontology and a node from the target ontology. We compare each node A of the source ontology with all the nodes A' of the target ontology when *num_of_neighbors(A)* – *num_of_neighbors(A′)* ≤ 1. As an alternative, we can also compare all nodes of the source ontology with all nodes of the target.

To compare a node A of the source ontology which has n neighbours with a node A' of the target ontology we calculate $n + 1$ probabilities. The first probability, called p_0, is calculated based on comparing neighbours of these two nodes (the first column of each node’s array). The other probabilities p_i (for $i = 1, 2, \ldots, n$) are calculated by comparing neighbours of these neighbours. Each row i of the first array is compared to all rows of the second array and accordingly the best matching row having the highest probability is considered as giving the probability p_i. In fact, p_i shows the best matching neighbour of the node A' with the ith neighbour of node A. When all probabilities are calculated, the average of these $n + 1$ probabilities forms the structural similarity of the two nodes.

After all elements of the similarity matrix are computed, the following three operations are applied to improve the initial computed matrix:

- If two nodes from the source and target ontologies are similar, their neighbours’ similarity will be increased by a predefined *bias* value.
- If two nodes from the source and target ontologies have n matched neighbours, their similarity for a predefined *bias* value will be increased by $n \times$ *bias*.
- If two nodes from the source and target ontologies have n common data type properties, their similarity, for a predefined *bias* value, will be increased by $n \times$ *bias*.

After conducting some experiments we obtained 0.1 as a suitable value for the *bias* factor. Thus we have used this value in our experiments. By applying a threshold value on the initial structural similarity matrix, we determine which nodes from source ontology are matched to which nodes of the target ontology. In general, in the process of generating the final structural similarity matrix, four different similarity matrices are generated. The initial similarity matrix is the first matrix and the other three matrices are generated by applying the three operations defined above to gradually improve the initial matrix. The last generated matrix will be considered as the final structural similarity matrix.

3.3 Combination of the lexical and structural similarity results

In subsections 3.1 and 3.2 the details of the proposed method for calculating lexical and structural similarities among entities of the source and target ontologies were discussed. As shown, the proposed

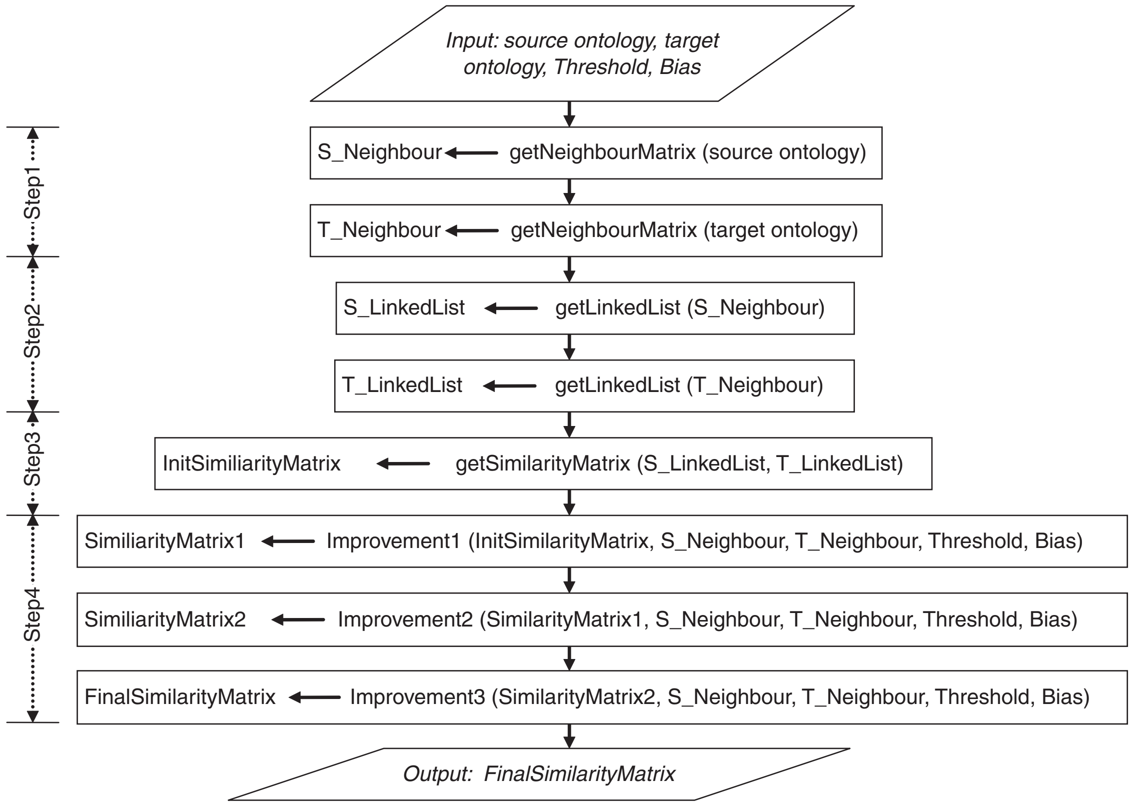
Input: source ontology, target ontology, Threshold, Bias
S_Neighbour ← getNeighbourMatrix (source ontology)
T_Neighbour ← getNeighbourMatrix (target ontology)
S_LinkedList ← getLinkedList (S_Neighbour)
T_LinkedList ← getLinkedList (T_Neighbour)
InitSimiliarityMatrix ← getSimilarityMatrix (S_LinkedList, T_LinkedList)
SimiliarityMatrix1 ← Improvement1 (InitSimilarityMatrix, S_Neighbour, T_Neighbour, Threshold, Bias)
SimiliarityMatrix2 ← Improvement2 (SimilarityMatrix1, S_Neighbour, T_Neighbour, Threshold, Bias)
FinalSimilarityMatrix ← Improvement3 (SimilarityMatrix2, S_Neighbour, T_Neighbour, Threshold, Bias)
Output: FinalSimilarityMatrix
Step1
Step2
Step3
Step4

Fig. 1. The schematic diagram of the structural similairty calculation algorithm.

method produces three lexical matrices for similarity of named classes, object properties and data properties, and a structural similarity matrix for named classes. Now, we examine the acquisition of the overall similarity results by combining these matrices.

To determine similar named classes of the source and target ontologies, the weighted mean of the structural similarity matrix and the first lexical similarity matrix (i.e. the similarity matrix of named classes) is used:

$$NamedClasses\_Similarity = {}^{\alpha \times Lexical\_NC\_Matrix + \beta \times Structural\_Matrix}\!\big/_{\alpha + \beta} \qquad (2)$$

where *NamedClasses_Similarity* is the overall similarity among named classes of the source and target ontologies, *Lexical_NC_Matrix* is the lexical similarity matrix of named classes, *Structural_Matrix* is the structural similarity matrix, and α and β are the weights assigned to lexical and structural matrices respectively. In our experiments if two given ontologies are more lexically similar than structurally then α and β coefficients will hold values 0.6 and 0.4 respectively, otherwise α = and β = 0.6.

To improve the second and the third lexical similarity matrices which correspond to the object and data properties, the following procedure is applied. For each pair of object or data properties from the source and target ontologies, if they have a lexical similarity value equal to or greater than 0.50 then:

- If they have aligned domains, their similarity will be increased by the predefined bias value.
- If they have aligned (or the same, for data properties) ranges, their similarity will be increased by the predefined bias value.

After this task, these matrices will reflect the overall similarity among object properties and data properties, respectively. We refer to the *NamedClasses_Similarity* matrix to decide which domains and ranges of the object and data properties are aligned.

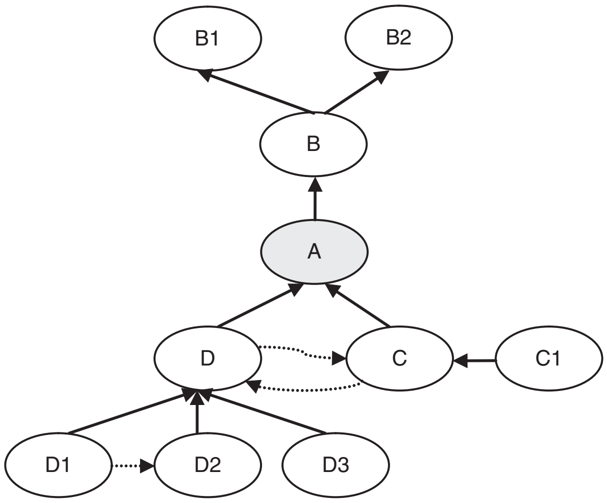

Fig. 2. An example ontology to show how the structural similarity matching phase works.

4. Illustrative example

An illustrative example is presented in this section to further clarify the way in which the structural similarity matching method of the proposed algorithm works. Consider Figure 2 as a sample ontology.

In Figure 2 solid arrows show the 'is-a', 'equivalent' or 'disjoint' relations and dotted arrows show the object property relations. As the ontology has 10 nodes, the neighbour matrix of this ontology will be a 10 × 10 matrix, see Figure 3(a). Using the computed neighbour matrix, an array of linked lists is created for each node to indicate its neighbours and the neighbours of its neighbours and their connections. In the neighbour matrix shown in Figure 3(a), for example, the first row corresponds to node *A* of the ontology. It shows that node *A* has three neighbours. Figure 3(b) shows the calculated array of linked list for node *A*. The first column of the array implies that node *A* has three neighbours each with three, three and five neighbours (corresponding to nodes *B*, *C* and *D*) respectively. The first row of the array implies that the first neighbour of node *A* (which is node *B*) itself has three neighbours each with one, one and three neighbours (corresponding to nodes *B*1, *B*2 and *A*) respectively, and so on.

Now, for structurally comparing node *A* of the example ontology with another node in a given target ontology, we would need to calculate four probabilities p_i (for $i = 0, 1, \ldots, 3$). To calculate p_o, the first column of *A*'s array is compared to the first column of the other node's array. This is basically the structural comparison of the two nodes based on their neighbours. To calculate p_i (for $i =$ 1, 2, 3), the *i* th row (excluding the first element) of *A*'s array is compared to all rows (excluding the first elements) of the target node's array and the best matching row of the target node's array determines the value of p_i. Finally, the average of the four calculated probabilities determines the structural similarity between node *A* and the target node.

The structural similarities among all nodes of the source ontology and all nodes of the target ontology constitute the initial structural similarity matrix. After calculating this matrix, the three operations introduced in subsection 3.2 are applied to generate the final structural similarity matrix.

5. Experiments and results

We have evaluated the performance of our proposed algorithm using the quality factor. For the quality of matching results, we compare our results with six algorithms presented in the Ontology

$$Neighbour = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad Linked\_list\,(A) = \begin{bmatrix} 3 & 1 & 1 & 3 & & \\ 3 & 3 & 5 & 1 & & \\ 5 & 3 & 3 & 2 & 2 & 1 \end{bmatrix}$$

(a) *(b)*

Fig. 3. (a) The neighbour matrix, and (b) the array of linked lists of node A of the ontology in Figure 2.

Alignment Evaluation Initiative 2008 (OAEI-08) [28] and also with the MLMA+ algorithm. The OAEI is an annual campaign for ontology matching systems to identify their strengths and weaknesses. It is a coordinated international initiative that organizes the evaluation of an increasing number of ontology matching systems. Its main goal is to compare systems and algorithms on the same basis and to allow anyone to draw conclusions about the best matching strategies [28]. The evaluation organizers provide a systematic benchmark test suite with pairs of ontologies to align as well as expected (human-based) results. These ontologies are described in OWL-DL and serialized in the RDF/XML format. The expected alignments are provided in a standard format expressed in RDF/XML and described in [29]. We have developed a tool based on the proposed algorithm and applied it to the OAEI-08 benchmark test suite [29]. We used standard information retrieval metrics to assess the results of our tests:

$$precision = \frac{\#correct\_found\_alignments}{\#found\_alignments} \tag{3}$$

$$Recall = \frac{\#correct\_found\_alignments}{\#existing\_alignments}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

We have categorized the test cases into five groups: test cases #101–104, #201–210, #221–247, #248–266 and #301–304. The average precision, recall and *f*-measure values obtained for each group using the proposed algorithm are presented in Table 1.

We used the Jena package [30] in our prototype tool to parse ontologies of the OAEI-08 benchmark test suite and considered the threshold value *th* = 0.70 and bias value *bias* = 0.1. In test case #101–104 there is good lexical and structural information, thus we have obtained the best results. In test cases #201–210 the source and target ontologies have identical structures and in test case #221–247 there is good lexical information in the source and target ontologies. So, we have got good results in these test cases too. Because the test case #248–266 has poor lexical and structural information we have acquired the worst results. Test case #301–304 has four real ontologies, as the results show; the proposed algorithm exhibits a good performance for this test case as well. We have compared our multi-level and combinatorial ontology matching approach with some systems like CIDER, DSSim, GeRoMe, MapPSO, SPIDER, TaxoMap [19, 28, 31–34], all participants of the OAEI-08 benchmark test suite and also with MLMA+ algorithm [19]. The results of applying these six algorithms on the OAEI-08 benchmark test suite are available and presented in [28]. The comparison results are

Table 1
Average performance of the proposed algorithm on the OAEI-08 benchmark test suite

	#101–104	#201–210	#221–247	#248–266	#301–304	Average
Precision	0.98	0.96	0.93	0.46	0.87	0.84
Recall	0.95	0.92	0.88	0.41	0.84	0.80
F-measure	0.96	0.93	0.90	0.43	0.85	0.81

presented in Table 2. The benchmark test cases have been placed into three categories: 1xx, 2xx and 3xx test cases. Table 2 shows the average precision and recall, the total average (which is a harmonic average) and also the *f*-measure value of these three categories. See equation (3).

As it can be observed from the results presented in Table 2, the proposed algorithm has a better *f*-measure value than the other systems and implies that it is more effective than the other systems. The proposed algorithm has also gained a better recall value than other systems. But, it has a lower precision value than TaxoMap, CIDER and DSSim systems. However, these systems have almost gained the worst recall and f-measure values among all algorithms. In fact they have sacrificed the recall metric for obtaining a better precision.

6. Conclusions

In this paper we presented an ontology matching algorithm that finds correspondences among entities of given ontologies based on their structure and lexical information. This algorithm works at three phases: lexical, structural, and combinatorial. For determining lexical similarity among entities, we introduced a new similarity measure which converts each entity's available lexical information, such as its label or description, to a bag of words which are then used for finding their similarities. In the first phase we obtain three lexical similarity matrices by comparing named classes, object properties, and data properties of the two ontologies. In the second phase, to structurally compare ontologies, we generate a grid for each node in the source and target ontologies and then compare them based on their grids. The important aspect of the structural approach is the number of neighbours a node has and how they are related to one another and to the node itself. Each node's grid is created using neighbours of that node and neighbours of those neighbours and is shown by a two-dimensional array. The initial structural similarity matrix is calculated by comparing these arrays. After generating this matrix, it is improved by applying the three operations described in subsection 3.2. Finally, in the third phase we compute the weighted mean of the lexical and structural results. Their weights imply their importance. We have performed our algorithm on the benchmark test suite of the OAEI-08 campaign and have had promising results. Also we compared our algorithm to some of the systems which participated in the OAEI-08 contest and, as Table 2 shows, it has a better *f*-measure.

Table 2
Comparison of average precision and recall values of our approach with some of the systems that participated in the OAEI-08 campaign

System	TaxoMap		MapPSO		GeRoMe		SPIDER		CIDER		DSSim		MLMA+		Our approach	
test	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
1xx	1.0	0.34	0.92	1.0	0.96	0.79	0.99	0.99	0.99	0.99	1.0	1.0	0.91	0.89	0.98	0.95
2xx	0.95	0.21	0.48	0.53	0.56	0.52	0.97	0.57	0.97	0.57	0.97	0.64	0.57	0.52	0.78	0.74
3xx	0.92	0.21	0.49	0.25	0.61	0.40	0.15	0.81	0.90	0.75	0.90	0.71	0.68	0.65	0.87	0.84
Average	0.91	0.22	0.51	0.54	0.60	0.58	0.81	0.63	0.97	0.62	0.97	0.67	0.69	0.65	0.86	0.83
F-measure	0.35		0.52		0.58		0.70		0.75		0.79		0.66		0.84	

References

[1] G. Stumme, A. Hotho and B. Berendt, Semantic Web mining state of the art and future directions, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web 4* (2006) 124–143.

[2] T. Gruber, Towards principles for the design of ontologies used for knowledge sharing, *International Journal of Human-Computer Studies* 43(5/6) (1995) 907–928.

[3] A. Gómez-Pérez and O. Corcho, Ontology languages for the semantic web, *IEEE Intelligent Systems Journal* 17(1) (2002) 54–60.

[4] J. Pan and I. Horrocks, RDFS(FA): Connecting RDF(S) and OWL DL, *IEEE Transactions on Knowledge and Data Engineering* 19(2) (2007) 192–206.

[5] H. Wang and C. Wang, Ontologies for universal information systems, *Journal of Information Science* 21 (1995) 232–239.

[6] N. Arch-Int and P. Sophatsathit, A semantic information gathering approach for heterogeneous information sources on WWW, *Journal of Information Science* 29 (2003) 357–374.

[7] W. Hu, N. Jian, Y. Qu and Y. Wang, GMO: A graph matching for ontologies, *K-Cap 2005 Workshop on Integrating Ontologies* 2005 (Banff, Alberta, Canada) 41–48.

[8] M. Ehrig and J. Euzenat. Relaxed precision and recall for ontology matching, *K-Cap 2005 Workshop on Integrating Ontologies* 2005 (Banff, Alberta, Canada) 25–32.

[9] P. Shvaiko and J. Euzenat, A survey of schema-based mapping approaches, *Journal on Data Semantics* 4 (2005) 146–171.

[10] A. H. Doan, J. Madhavan, P. Domingos and A. Halevy, Learning to map between ontologies in the semantic web, *VLDB Journal, Special Issue on the Semantic Web* (2003).

[11] E. Rahm and P.A. Bernstein, A survey of approaches to automatic schema matching, *The VLDB Journal* 10 (2001).

[12] Y.R. Jean-Marya, E.P. Shironoshita and M.R. Kabuka, Ontology matching with semantic verification, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 7(3) (2009) 235–251.

[13] L.S. Xiao and R. Ellen, Automated schema mapping techniques: an exploratory study, *Research Letters Information Science* 4 (2003) 113–136.

[14] W. Cohen, P. Ravikumar and S. Fienberg, A comparison of string metrics for matching names and records, *Proceedings of the Workshop on Data Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining (KDD)* (2003).

[15] H.H. Do and E. Rahm, COMA – a system for flexible combination of schema matching approaches, *Proceedings of the Very Large Data Bases Conference* (2001) 610–621.

[16] F. Giunchiglia, M. Yatskevich and P. Shvaiko, Semantic matching: algorithms and implementation, *Journal on Data Semantics* 10 (2007) 1–38.

[17] J. Madhavan, P. Bernstein and E. Rahm, Generic schema matching with Cupid, *Proceedings of the Very Large Data Bases Conference* (2001) 49–58.

[18] S. Melnik, H. Garcia-Molina and E. Rahm, Similarity flooding: a versatile graph matching algorithm, *Proceedings of the International Conference on Data Engineering* (2002) 117–128.

[19] A. Alasoud, V. Haarslev and N. Shiri, An empirical comparison of ontology matching techniques, *Journal of Information Science* 35(4) (2009) 379–397.

[20] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, *Soviet Physics Doklady* 10 (1966) 707–710.

[21] G.A. Miller, WordNet: A lexical database for english, *Communications of the ACM* 38 (1995) 39–41.

[22] P. Bouquet, L. Serafini and S. Zanobini, Peer-to-peer semantic coordination, *Journal of Web Semantics* 2(1) (2004) 81–97.

[23] G. Pirro, A semantic similarity metric combining features and intrinsic information content, *Journal of Data and Knowledge Engineering* 68 (2009) 1289–1308.

[24] F. Giunchiglia, P. Shvaiko and M. Yatskevich, S-Match: an algorithm and an implementation of semantic matching, In *Proceedings of the European Semantic Web Symposium* (2004) 61–75.

[25] A. Maedche and S. Staab, Measuring similarity between ontologies, In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management* (2002) 251–263.

[26] X. Dong, J. Madhavan and A. Y. Halevy, Mining structures for semantics, *SIGKDD Explorations* 6(2) (2004) 53–60.

[27] I. Akbari, M. Fathian and K. Badie, An Improved MLMA+ Algorithm and its Application in Ontology Matching, *Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)* (Malaysia, 2009).

[28] P. Shvaiko, J. Euzenat, F. Giunchiglia and H. Stuckenschmidt, Results of the ontology alignment evaluation initiative 2008, In *Ontology Matching Workshop*, The 7th International Semantic Web Conference. (Karlsruhe, 2008).
[29] http://oaei.ontologymatching.org/2008/benchmarks/ (accessed 21 March 2009).
[30] http://jena.sourceforge.net/ (accessed 3 March 2009).
[31] J. Gracia and E. Mena, Ontology matching with CIDER: evaluation report for the OAEI 2008, *ISWC Workshop on Ontology Matching* (2008).
[32] M. Nagy, M. Vargas-Vera and P. Stolarski, DSSim Results for OAEI 2008, *The 7th International Semantic Web Conference* (Karlsruhe, 2008).
[33] J. Bock and J. Hettenhausen, MapPSO Results for OAEI 2008, *The 7th International Semantic Web Conference* (Karlsruhe, 2008).
[34] F. Hamdi, H. Zargayouna, B. Safar and C. Reynaud, TaxoMap in the OAEI 2008 alignment contest, *The 7th International Semantic Web Conference* (Karlsruhe, 2008).