

# Block Matching for Ontologies

Wei Hu and Yuzhong Qu

School of Computer Science and Engineering, Southeast University,  
Nanjing 210096, P. R. China  
{whu, yzqu}@seu.edu.cn

**Abstract.** Ontology matching is a crucial task to enable interoperation between Web applications using different but related ontologies. Today, most of the ontology matching techniques are targeted to find 1:1 mappings. However, block mappings are in fact more pervasive. In this paper, we discuss the block matching problem and suggest that both the mapping quality and the partitioning quality should be considered in block matching. We propose a novel partitioning-based approach to address the block matching issue. It considers both linguistic and structural characteristics of domain entities based on virtual documents, and uses a hierarchical bisection algorithm for partitioning. We set up two kinds of metrics to evaluate of the quality of block matching. The experimental results demonstrate that our approach is feasible.

## 1 Introduction

Web ontologies written in RDF [12] or OWL [19] play a prominent role in the Semantic Web. Due to the decentralized nature of the Web, there always exist multiple ontologies from overlapped domains or even from the same domain. In order to enable interoperation between Web applications using different but related ontologies, we need to establish mappings between ontologies for capturing the semantic correspondence between them.

The common relationship cardinality of mappings between concepts, relations or instances (we uniformly name them as *domain entities*) of ontologies is 1:1. However, mappings between sets of domain entities are more pervasive. In particular, 1:1 mappings can be viewed as a special case of mappings between sets of domain entities. In this paper, a *block* is a set of domain entities. A *block mapping* is a pair of matched blocks from two ontologies in correspondence. We refer to the process of discovering block mappings as *block matching*.

Block matching is required in many occasions. Two discriminative examples are illustrated as follows.

*Example 1.* Given two ontologies (denoted by  $O_1$  and  $O_2$ ),  $O_1$  contains three domain entities: **Month**, **Day** and **Year**; while  $O_2$  contains a single domain entity: **Date**. We can see **Month**, **Day** and **Year** are parts of **Date**. So it is more natural to match the block {**Month**, **Day**, **Year**} in  $O_1$  with the block {**Date**} in  $O_2$ .

*Example 2.* When two ontologies being compared, block matching can provide us a general picture at a higher level to explore macroscopical correspondences between the main topics assigned to these two ontologies, and it may also help to generate some new focused ontologies from the original block mappings.

The blocking matching problem can be transformed to a special kind of partitioning problem. Usually, it is required that blocks in either of the two ontologies should be disjointed with each other. So all the block mappings essentially compose a partitioning of all domain entities from the two ontologies with the requirement that each partition should contain at least one domain entity from each of the two ontologies. From the viewpoint of partitioning, the cohesiveness within each block mapping should be high; while the coupling crossing different block mappings should be low. Therefore, in addition to the inherent difficulties in discovering the high quality mappings, the block matching problem is exacerbated by having to consider the partitioning quality of the block matching.

Nowadays, quite a lot of algorithms have been proposed in literature addressing the ontology matching problem. GLUE [6], QOM [7], OLA [8], S-MATCH [10], HCONE-MERGE [13], PROMPT [14], V-DOC [20] and I-SUB [23] are such works. However, these algorithms cannot solve the block matching problem since they are targeted to find 1:1 mappings. To our knowledge, the block matching problem has only been addressed before in PBM [11]. But it merely partitions two large class hierarchies separately without considering the correspondence between them. Certainly, the mapping quality is not satisfied. In addition, it just copes with mappings between classes, thus it is not a general solution for ontology matching.

In this paper, we propose a new partitioning-based approach to address the block matching problem. Partitioning entails both developing a relatedness measure and choosing an appropriate partitioning algorithm. We consider both linguistic and structural characteristics of domain entities based on virtual documents for the relatedness measure [20]. The novelty of this measure is that both the mapping quality and the partitioning quality can be guaranteed simultaneously. We present a hierarchical bisection algorithm for partitioning, which can provide block mappings at different levels of granularity. We also describe an automatic process to extract the optimal block mappings with a given number of block mappings. Besides, we assume the mappings between blocks is 1:1 in order to avoid the combinatorial explosion of the search space.

The remainder of this paper is organized as follows. Section 2 sketches out our approach. Section 3 introduces the computation of the relatedness among domain entities by virtual documents. Section 4 presents a hierarchical bisection algorithm based on the relatedness, and describes a method to automatically extract the optimal block mappings for a flat partitioning. Section 5 sets up two kinds of metrics to evaluate of the quality of the block matching generated by our approach. Section 6 discusses some related works on ontology matching as well as some related works on ontology partitioning. Finally, Section 7 provides concluding remarks.

## 2 Overview of the Approach

The overview of the approach is illustrated in Figure 1. Generally speaking, our approach starts with two ontologies to be compared as input, and then after four processing stages, the output returns block mappings between the two ontologies.

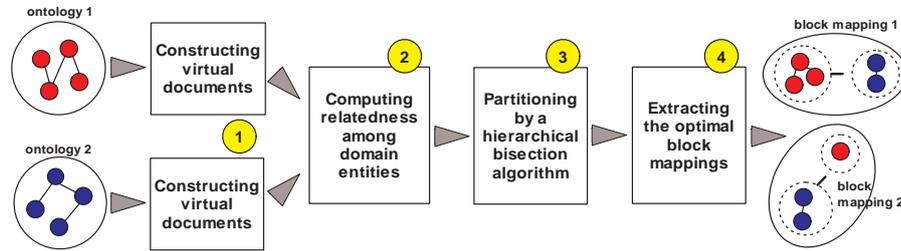


Fig. 1. The overview of the approach

1. *Constructing virtual documents.* The process constructs virtual document for each domain entity of the input ontologies. We make use of the virtual documents as the features of domain entities to be compared. The virtual document of a domain entity consists of a collection of weighted words; these words come from not only the local descriptions (e.g., labels) but also the neighboring information to reflect the intended meaning of the entity.
2. *Computing relatedness among domain entities.* The process sets up the relatedness for any two domain entities by computing the similarity between the virtual documents of them in correspondence. More precisely, it includes the comparison among domain entities within each of the two ontologies as well as those crossing the two ontologies. Therefore, the linguistic and structural characteristics are both revealed simultaneously in a uniform process.
3. *Partitioning by a hierarchical bisection algorithm.* The hierarchical bisection algorithm acts on the set of domain entities from the two ontologies. It recursively partitions the unrelated or dissimilar domain entities into disjoint blocks mappings. As a result, the similar ones are fallen into the same block mapping (containing the domain entities from the two ontologies). The algorithm returns a dendrogram (a typical type of tree structure) consisting of layers of block mappings at different levels of granularity.
4. *Extracting the optimal block mappings.* The process finds the optimal block mappings in the dendrogram derived from the hierarchical bisection algorithm for a flat partitioning with a given number of block mappings.

We will further describe each process in the next two sections.

### 3 Relatedness among Domain Entities

In this section, we construct virtual documents for the domain entities declared in OWL/RDF ontologies. Then, we compute the relatedness among the domain entities by calculating the similarity among the virtual documents.

#### 3.1 Construction of Virtual Documents

The RDF graph model is the foundation of the Semantic Web ontologies, and OWL ontologies can also be mapped to RDF graphs [19]. Therefore, we uniformly use the RDF graph model to represent ontologies.

An RDF graph is a set of triples (statements). An RDF triple is conventionally written in the order (subject, predicate, object). A node in an RDF graph may be a literal, a URI with an optional local name (URI reference, or URIref), or a blank node. Please note that a predicate is always a URIref, and a literal cannot be a subject.

In the field of Information Retrieval, the content of a document might be represented as a collection of tokens: words, stems, phrases, or other units derived or inferred from the text of the document. These tokens are usually weighted to indicate their importance within the document which can then be viewed as a vector in a high dimensional space. In this paper, a virtual document represents a collection of weighted tokens, and the weights are rational numbers. To simplify the expression, we use the term *a collection of words* instead of a collection of weighted tokens.

As a collection of words, the virtual document of a domain entity contains not only the local descriptions but also the neighboring information to reflect the intended meaning of the entity.

- *Local descriptions.* For a literal node, the local description is a collection of words derived from the literal itself. For a URIref, it is a collection of words extracted from the local name, rdfs:label(s), rdfs:comment(s) and other possible annotations. For a blank node, it is a collection of words extracted from the information originated from the forward neighbors. A weighting scheme is incorporated in the formation of the description.
- *Neighboring Information.* We capture different kinds of neighbors (subject neighbors, predicate neighbors and object neighbors) by distinguishing the places the nodes occurred in triples. The descriptions of these neighbors are integrated as neighboring information in the virtual document of a domain entity to reflect the structural information of the domain entity.

For formal definitions, please refer to [20].

#### 3.2 Computation of Relatedness

The similarity among virtual documents of domain entities is calculated in the Vector Space Model (VSM) [17]. In this model, the virtual document of a domain

entity is considered to be a vector. In particular, we employ the TF/IDF [21] term weighting model, in which each virtual document can be represented as follows:

$$(tf_1 \cdot idf_1, tf_2 \cdot idf_2, \dots, tf_n \cdot idf_n), \quad (1)$$

where  $tf_i$  is the frequency of the  $i$ th word in a given virtual document and  $idf_i$  is the distinguishability of the word in such document w.r.t. the whole. So the TF/IDF term weighting model gives prominence to the words close by related to the given virtual documents, which to some extent exposes the latent features of the virtual documents.

The similarity between virtual documents is measured by the cosine value between the two vectors  $\vec{N}_i$  and  $\vec{N}_j$ , corresponding to two virtual documents  $D_i$  and  $D_j$  in the Vector Space Model. The measure is defined as follows:

$$sim(D_i, D_j) = \cos(\vec{N}_i, \vec{N}_j) = \frac{\sum_{k=1}^d n_{ik}n_{jk}}{\sqrt{(\sum_{k=1}^d n_{ik}^2)(\sum_{k=1}^d n_{jk}^2)}}, \quad (2)$$

where  $d$  is the dimension of the vector space, and  $n_{ik}$  ( $n_{jk}$ ) is the  $k$ th component of the vector  $\vec{N}_i$  ( $\vec{N}_j$ ). If the two virtual documents do not share any words, the similarity will be 0.0. If all the word scores equal completely, it will be 1.0.

After computing the similarity among virtual documents within each of the two ontologies as well as crossing the two ontologies, we can obtain a relatedness matrix, denoted by  $W$ . The matrix has the following block structure:

$$W = \begin{pmatrix} W_{11} & W_{12} \\ W_{12}^T & W_{22} \end{pmatrix}, \quad (3)$$

where  $W_{11}$  is a matrix representing the relatedness among domain entities within the ontology  $O_1$ , and  $W_{22}$  is similarly defined for the ontology  $O_2$ .  $W_{12}$  is a matrix representing the relatedness among domain entities between  $O_1$  and  $O_2$ . Please note that we assume that the relatedness among domain entities is symmetric in our approach.

The relatedness matrix  $W$  has two features. Firstly, both of linguistic and structural relatedness within each of the two ontologies are reflected in  $W_{11}$  and  $W_{22}$ , respectively. For example, to exhibit structural relatedness within  $O_1$  or  $O_2$ , each domain entity collects its neighboring information, i.e., the local descriptions of the subject, predicate or object neighbors, and then the structural affinity between any two entities is revealed through shared words obtained from neighborhood relationship in Vector Space Model. In other words, two entities within  $O_1$  or  $O_2$  are more related if they co-occur in more statements. Secondly, linguistic relatedness crossing ontologies is characterized by  $W_{12}$ . This matrix is one of the most important key points in this paper.

## 4 Partitioning for Block Matching

In this section, we present a hierarchical bisection algorithm based on the relatedness among domain entities. Besides, we describe a method to automatically find the optimal block mappings with a given number of block mappings.

### 4.1 The Hierarchical Bisection Algorithm

The objective of a partitioning solution is seeking to partition the set of vertices  $V$  into disjoint clusters  $V_1, V_2, \dots, V_n$ , where by some measure the cohesiveness among the vertices in a cluster  $V_i$  is high; while the coupling crossing different clusters  $V_i, V_j$  is low. In the context of this paper, we seek to partition domain entities of two ontologies into block mappings, so that the relatedness among the domain entities in a block mapping is high, and that crossing different block mappings is low.

The partitioning approach we present in this paper is a hierarchical bisection algorithm. In each bisection, it partitions the domain entities into two disjoint block mappings  $B_1, B_2$ . We adopt the min-max cut (*Mcut*) function [5] as the criterion function. It minimizes the relatedness between the two block mappings meanwhile maximizes the relatedness within each block mapping. The *Mcut* function is defined as follows:

$$Mcut(B_1, B_2) = \frac{cut(B_1, B_2)}{W(B_1)} + \frac{cut(B_1, B_2)}{W(B_2)}, \quad (4)$$

where  $cut(B_1, B_2)$  is the sum of the relatedness among domain entities across  $B_1$  and  $B_2$ .  $W(B_1)$  is the sum of the relatedness within  $B_1$  and  $W(B_2)$  is similarly defined. The optimal bisection is the one that minimizes the *Mcut*.

The optimal solution of *Mcut* is NP-complete. However, the relaxed version of this objective function optimization can be well solved in a spectral way. Roughly speaking, spectral partitioning makes use of the eigenvalues and eigenvectors of the relatedness matrix to find a partitioning. The merit of the spectral methods is the easiness in implementation and the reasonable performance. Furthermore, they do not intrinsically suffer from the problem of local optima.

In addition, our approach is a hierarchical approach. The reason is that it is usually difficult to specify the exact partitioning for a given domain, and there may not be a single correct answer. The block mappings in each bisection form a dendrogram. The dendrogram provides a view of the block mappings at different levels of granularity, which allows flat partitions of different granularity to be extracted. In Section 4.2, we make use of the dendrogram to extract the optimal block mappings.

The algorithm is illustrated in Table 1. The input of the algorithm is a relatedness matrix  $W$ . During a run, it recursively bisects a matrix into two submatrices by searching the minimum *Mcut*. In the end, it returns a dendrogram consisting of layers of block mappings at different levels of granularity. The eigenvector corresponding to the second smallest eigenvalue in Step 3 is also called the Fielder

vector [9]. It provides a linear search order (Fielder order). The discussion on the properties of the Fielder vector is out of the scope of this paper. In Step 4, we set a parameter  $\epsilon$  to limit the minimum number of domain entities in each block mapping, which can decrease the recursion times. In our experiments, we set the parameter  $\epsilon$  to 10.

**Table 1.** The hierarchical bisection algorithm

---

**Algorithm.** The hierarchical bisection algorithm.

---

**Input.** A relatedness matrix  $W$ , and a parameter  $\epsilon$ .

**Output.** A dendrogram consisting of layers of block mappings.

1. Initialize a diagonal matrix  $D$  with the row sums of  $W$  on its diagonal.
  2. Solve the eigenvalues and eigenvectors of  $(D - W)$ .
  3. Let  $v$  be the eigenvector corresponding to the second smallest eigenvalue.
    - 3.1 Sort  $v$  so that  $v_i < v_{i+1}$
    - 3.2 Find the splitting point  $t$  such that  
 $(A, B) = (\{1, \dots, t\}, \{t + 1, \dots, |v|\})$  minimizes the  $Mcut$ .
  4. Let  $W_A, W_B$  be the submatrices of  $W$ , respectively.
    - 4.1 Recurse (Steps 1–3) on  $W_A$  until  
 the number of domain entities within  $W_A$  is less than  $\epsilon$ .
    - 4.2 Recurse (Steps 1–3) on  $W_B$  until  
 the number of domain entities within  $W_B$  is less than  $\epsilon$ .
- 

The time complexity of Steps 1–4 is  $O(m+n)$ , where  $m$  denotes the number of nonzero components in  $W$  and  $n$  denotes the number of domain entities (equals to the row (or column) dimension of  $W$ ). The most time-consuming step is Step 2. Usually, the time complexity of eigenvalue decomposition is  $O(n^3)$ . Since we only need a vector with the second smallest eigenvalue, the time complexity can be decreased to  $O(m+n)$  via the Lanczos method [16].

## 4.2 Extraction of the Optimal Block Mappings

So far, we have constructed a dendrogram by the hierarchical bisection algorithm presented above. In some cases, we would like to obtain a flat partitioning with a given number of block mappings  $k$ . For this purpose, we need to extract the optimal block mappings from the dendrogram. In this paper, we use the dynamic programming method proposed in [2, 5].

Let  $opt(B_i, p)$  be the optimal block mappings for  $B_i$  using  $p$  block mappings.  $B_l, B_r$  denotes the left and right children of  $B_i$  in the dendrogram, respectively. Then, we have the following recurrence:

$$opt(B_i, p) = \begin{cases} B_i & \text{if } p = 1 \\ \arg \min_{1 \leq j < p} g(opt(B_l, j) \cup opt(B_r, p - j)) & \text{otherwise} \end{cases}, \quad (5)$$

where  $g$  is the objective function, which is defined as follows:

$$g(\{B_1, B_2, \dots, B_p\}) = \min\left(\frac{\text{cut}(B_1, \overline{B_1})}{W(B_1)} + \frac{\text{cut}(B_2, \overline{B_2})}{W(B_2)} + \dots + \frac{\text{cut}(B_p, \overline{B_p})}{W(B_p)}\right). \quad (6)$$

By computing the optimal block mappings from the leaf nodes in the dendrogram firstly, we can finally gain  $\text{opt}(B_{\text{root}}, k)$ , which includes  $k$  optimal block mappings for a flat partitioning.

## 5 Evaluation

We have implemented our approach in Java, called BMO, and then evaluated its performance experimentally. Due to lack of space, we cannot list all the details about our experiments. The test cases and all the experimental results can be downloaded from our website <sup>1</sup>. Please note that in our evaluation, we focus on the domain entities at the schema level, i.e., we just consider the classes and properties in ontologies. However, it is worthy of noting that our approach can easily be extended to the ontologies containing instances.

### 5.1 Case Study

In our evaluation, we choose two pairs of ontologies: **russia12** and **tourismAB**. They can be downloaded from the website <sup>2</sup>. The reasons for selecting them as test cases are: (i) they are from real world domains and famous in the field of ontology matching, (ii) their sizes are moderate. If the sizes of ontologies are too small, it is unnecessary to partition them into blocks; while if the sizes are too large, they are not appropriate for human observation, and (iii) they have reference files contains aligned domain entity pairs. Short descriptions of the two pairs of ontologies are given below.

- **russia12**. The two ontologies are created independently by different people from the contents of two travel websites about Russia. **russia1** contains 151 classes and 76 properties. **russia2** contains 162 classes and 81 properties. The reference alignment file contains 85 aligned domain entity pairs.
- **tourismAB**. The two ontologies are created separately by different communities describing the tourism domain of Mecklenburg-Vorpommern (a federal state in the northeast of Germany). **tourismA** contains 340 classes and 97 properties. **tourismB** contains 474 classes and 100 properties. The reference alignment file contains 226 aligned domain entity pairs.

<sup>1</sup> <http://xobjects.seu.edu.cn/project/falcon/>

<sup>2</sup> <http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/>

## 5.2 Experimental Methodology and Evaluation Metrics

Let us recall that the ideal block mappings should have both high mapping quality and high partitioning quality. In order to measure the two kinds of quality, two experiments are designed to evaluate the effectiveness of BMO. The first experiment is to measure the mapping quality of block mappings. The other one is to assess the partitioning quality of block mappings. Besides, we also make a comparison between BMO and PBM [11].

In the first experiment, we evaluate the mapping quality of the computed block mappings by observing the *correctness* with the variation of the number of the block mappings. The rationale is that the higher the quality of the block matching is, the more aligned domain entity pairs could be found in the block mappings.

Let  $B$  be a set of the computed block mappings ( $|B| = n$ ).  $B_i$  denotes the  $i$ th block mapping in  $B$ . Let  $R$  be a set of aligned domain entity pairs in a reference alignment file ( $|R| = r$ ).  $R_p$  denotes the  $p$ th aligned domain entity pair in  $R$ . The correctness of  $B$  is defined as follows:

$$correctness(B) = \frac{1}{r} \sum_{i=1}^n |\{R_p | R_p \subseteq B_i, 1 \leq p \leq r\}|. \quad (7)$$

Intuitively, the correctness of  $B$  increases when the number of the block mappings decreases. In particular, when  $n = 1$ , the correctness of  $B$  is 1.0. However, it is clear that merely considering the metric of the correctness is not sound. We need to evaluate the quality of block mappings in other aspects.

In the second experiment, we focus on evaluating the partitioning quality of the block matching. Three volunteers are trained to set up manual block mappings. We assess the partitioning quality of the computed block mappings by comparing with the manual ones. In this experiment, we set  $n$  equal to the number of block mappings of the manual ones. This kind of measurement is widely adopted in the field of Data Clustering.

We use two well-known metrics to compare the gained block mappings with the manual ones. The first metric is *f-measure*. The other one is *entropy*. Before introducing the two metrics, we firstly defined two basic operations (*precision* and *recall*), which are used to compare a gained block mapping with a manual one. Let  $C$  be the set of the manual block mappings ( $|C| = m$ ).  $C_j$  denotes the  $j$ th block mapping in  $C$ .  $|B_i|$  returns the number of domain entities in  $B_i$ , and  $|C_j|$  is defined analogously.  $|B_i \cap C_j|$  calculates the mutual domain entities in both  $B_i$  and  $C_j$ . The precision and recall of a computed block mapping  $B_i$  referring to  $C_j$  are defined as follows respectively:

$$prec(B_i, C_j) = \frac{|B_i \cap C_j|}{|B_i|}, \quad (8)$$

$$reca(B_i, C_j) = \frac{|B_i \cap C_j|}{|C_j|}. \quad (9)$$

The f-measure is defined as a combination of the precision and recall. Its score is in the range  $[0, 1]$ , and a higher f-measure score implies a better partitioning

quality. The f-measure of the set of computed block mappings  $B$  is defined as follows:

$$f\text{-measure}(B) = \frac{1}{\sum_{i=1}^n |B_i|} \cdot \sum_{i=1}^n f\text{-measure}(B_i) \cdot |B_i|, \quad (10)$$

$$f\text{-measure}(B_i) = \max_{1 \leq j \leq m} \frac{2 \cdot \text{prec}(B_i, C_j) \cdot \text{reca}(B_i, C_j)}{\text{prec}(B_i, C_j) + \text{reca}(B_i, C_j)}. \quad (11)$$

The other metric is the entropy. It considers the distribution of domain entities in block mappings and reflects the overall partitioning quality. A lower entropy score implies a better partitioning quality. The best possible entropy score is 0; while the worst is 1. The entropy of the set of the computed block mappings  $B$  is defined as follows:

$$\text{entropy}(B) = \frac{1}{\sum_{i=1}^n |B_i|} \cdot \sum_{i=1}^n \text{entropy}(B_i) \cdot |B_i|, \quad (12)$$

$$\text{entropy}(B_i) = -\frac{1}{\log m} \cdot \sum_{j=1}^m \text{prec}(B_i, C_j) \cdot \log(\text{prec}(B_i, C_j)). \quad (13)$$

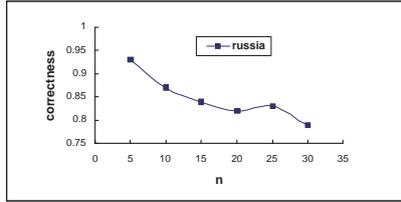
In the last experiment, we compare both the mapping quality and the partitioning quality of BMO with PBM [11]. Because PBM merely copes with mappings between classes, for comparing the mapping quality, we remove the aligned property pairs in the reference alignment files, only retaining 70 aligned class pairs in `ruissia12` and 190 aligned class pairs in `tourismAB`. In addition, we construct the manual block mappings only between classes to evaluate the partitioning quality of these two approaches.

### 5.3 Discussion on Experimental Results

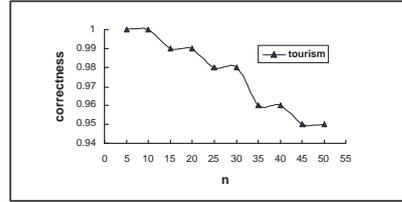
Firstly, the correctness with the variation of the number of the block mappings (denoted by  $n$ ) is depicted in Figure 2. We can see that in the two test cases, when  $n$  increases, the correctness of the block mappings decreases. We can also find that in most situations, the correctness of the results is fine. In particular, in `tourismAB`, when  $n = 50$ , the correctness is still larger than 95%. It demonstrates that the mapping quality of the block mappings computed by BMO is high. In addition, the correctness does not decrease drastically as  $n$  increases. It implies that BMO is stable with a pretty good accuracy.

Secondly, by setting the number of the required block mappings (25 block mappings for `ruissia12`, and 26 block mappings for `tourismAB`), we can compare the the results of BMO with the manual ones to evaluate the partitioning quality. The partitioning quality of the computed block mappings are shown in Table 2. Both the f-measure and the entropy are moderate.

Finally, the comparison results of the mapping quality and the partitioning quality between BMO and PBM are presented in Table 3. Both the number of the required block mappings for `ruissia12` and for `tourismAB` are 13. From the table,



(a) russia12



(b) tourismAB

**Fig. 2.** The correctness with the variation of the number of the block mappings  $n$ **Table 2.** The partitioning quality of BMO

	number	f-measure	entropy
russia12	25	0.61	0.28
tourismAB	26	0.52	0.22

we can see that the partitioning quality between the two approach are almost the same. However, the mapping quality of the BMO approach is far beyond the one of PBM. For example, in **russia12**, the correctness of the BMO approach is 0.84; while the one of PBM is merely 0.57.

**Table 3.** The comparison between BMO and PBM

	approach	number	correctness	f-measure	entropy
russia12	BMO	13	0.84	0.56	0.37
	PBM	13	0.57	0.65	0.33
tourismAB	BMO	13	0.98	0.67	0.31
	PBM	13	0.66	0.57	0.30

Based on the observations above, we can make a preliminary and empirical conclusion that our approach is feasible for achieving a good mapping quality as well as a good partitioning quality.

## 6 Related Work

In this section, we firstly discuss some related works on ontology matching, and then we present some related works on ontology partitioning.

### 6.1 Ontology Matching

Despite many works (e.g., [6–8, 10, 11, 13, 14, 20, 23]) have addressed the ontology matching (also called ontology mapping or alignment) problem, there exist

very few approaches raising the issue of block matching. PBM [11] is the only work we know so far that considers the block matching problem. It exploits block mappings between two class hierarchies by firstly partitioning them into blocks respectively, and then constructing the mappings between blocks via the pre-defined anchors generated by the string comparison techniques. The weakness of that work is that it ignores the correspondence between the two hierarchies when doing partitioning, so the mapping quality is not satisfied. Furthermore, it just copes with mappings between classes, so it might not be applicable to ontology matching in general.

In the field of schema matching (please see [18] for a survey), iMAP [4] is semi-automatically discovers both the 1:1 and complex mappings (e.g.,  $\text{room-price} = \text{room-rate} * (1 + \text{tax-rate})$ ). It embeds two new kinds of domain knowledge (overlapped data and external data) to find complex mappings. However, iMAP may not be a universal solution, because it is not easy to specify the domain knowledge in some cases. ARTEMIS [1] is another work which vaguely presents the idea of block matching. It firstly computes the 1:1 mappings between two ontologies by using WordNet, and then constructs block mappings from the 1:1 mappings via a clustering algorithm. This is similar to the framework of our approach. But it is clear that the method always suffers from the high computational complexity for calculating the 1:1 mappings. More importantly, it discards both the linguistic and structural characteristics in each of the two ontologies, thus the partitioning quality cannot be guaranteed.

## 6.2 Ontology Partitioning

From another viewpoint, our method partitions two ontologies into blocks throughout the process of searching the block mappings. So it might be broken down into the category of ontology partitioning. [3, 15, 22, 24] are some representative works. However, these works only provide a flat partitioning on a single ontology; while our work supports a hierarchical view with different levels of granularity, and partitions two ontologies at the same time. But, we should note that these ontology partitioning techniques might also be used to find block mappings by partitioning two ontologies separately, and then matching these blocks. This is just the method adopted in PBM. Although this kind of methodology could deal with the block matching problem between large-scale ontologies, as shown in our experiments, the mapping quality is usually not so good as BMO's.

## 7 Concluding Remarks

In summary, the main contributions of this paper are as follows.

- We discussed the block matching problem and suggested both the mapping quality and the partitioning quality should be considered in block matching.

- We proposed a relatedness measure based on virtual documents that simultaneously importing both linguistic and structural characteristics of domain entities.
- We presented a hierarchical bisection algorithm to provide block mappings at different levels of granularity. Also, we described a method to automatically extract the optimal block mappings for a flat partitioning.
- We set up two kinds of metrics to evaluate of the quality of block matching. The experimental results demonstrated that our approach is feasible.

The work reported here is a first step towards block matching for ontologies, and many issues still need to be addressed. In future work, we plan to find other possible approaches to block matching, and compare them with each other. Furthermore, in order to make steady progress on the block matching problem, it is valuable to set up systematic test cases for block matching. Another issue is block matching for very large-scale ontologies.

## Acknowledgements

The work is supported in part by the NSFC under Grant 60573083, and in part by the 973 Program of China under Grant 2003CB317004, and also in part by the JSNSF under Grant BK2003001. We are grateful to Prof. Jianming Deng and Dr. Yanbing Wang for their valuable suggestions. We also thank Gong Cheng, Yuanyuan Zhao and Dongdong Zheng for their work in the experiments related to this paper. In the end, we appreciate anonymous reviewers for their precious comments.

## References

1. Castano, S., De Antonellis, V., and De Capitani Di Vimercati, S.: Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*. **13(2)** (2001) 277–297
2. Cheng, D., Kannan, R., Vempala, S., and Wang, G.: A divide-and-merge methodology for clustering. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS'05)*. (2005) 196–205
3. Cuenca Grau, B., Parsia, B., and Sirin, E.: Combining OWL ontologies using  $\varepsilon$ -connections. *Journal of Web Semantics*. **4(1)** (2005)
4. Dhamankar, R., Lee, Y., Doan, A. H., Halevy, A., and Domingos, P.: iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the 23th ACM International Conference on Management of Data (SIGMOD'04)*. (2004) 383–394
5. Ding, C. H. Q., He, X., Zha, H., Gu, M., and Simon, H. D.: A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*. (2001) 107–114
6. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A. Y.: Learning to match ontologies on the semantic web. *VLDB Journal*. **12(4)** (2003) 303–319

7. Ehrig, M., and Staab, S.: QOM - quick ontology mapping. In Proceedings of the 3rd International Semantic Web Conference (ISWC'04). (2004) 683–697
8. Euzenat, J., and Valtchev, P.: Similarity-based ontology alignment in OWL-Lite. In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04). (2004) 333–337
9. Fiedler, M.: A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*. **25** (1975) 619–633
10. Giunchiglia, F., Shvaiko, P., and Yatskevich, M.: S-Match: An algorithm and an implementation of semantic matching. In Proceedings of the 1st European Semantic Web Symposium (ESWS'04). (2004) 61–75
11. Hu, W., Zhao, Y. Y., and Qu, Y. Z.: Partition-based block matching of large class hierarchies. In Proceedings of the 1st Asian Semantic Web Conference (ASWC'06). (2006) 72–83
12. Klyne, G., and Carroll, J. J. (eds.): Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/rdf-concepts/>
13. Kotis, K., Vouros, G. A. , and Stergiou, K.: Towards automatic merging of domain ontologies: The HCONE-merge approach. *Journal of Web Semantics*. **4(1)** (2005)
14. Noy, N. F., and Musen, M. A.: The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*. **59** (2003) 983–1024
15. Noy, N. F., and Musen, M. A.: Specifying ontology views by traversal. In Proceedings of the 3rd International Semantic Web Conference (ISWC'04). (2004) 713–725
16. Parlett, B. N.: *The symmetric eigenvalue problem*. SIAM Press. (1998)
17. Raghavan, V. V., and Wong, S. K. M.: A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*. **37(5)** (1986) 279–287
18. Rahm, E., and Bernstein, P.: A survey of approaches to automatic schema matching. *VLDB Journal*. **10** (2001) 334–350
19. Patel-Schneider, P. F., Hayes, P., and Horrocks, I. (eds.): OWL web ontology language semantics and abstract syntax. W3C Recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/owl-semantics/>
20. Qu, Y. Z., Hu, W., and Cheng, G.: Constructing virtual documents for ontology matching. In Proceedings of the 15th International World Wide Web Conference (WWW'06). (2006) 23–31
21. Salton, G., and McGill, M. H.: *Introduction to modern information retrieval*. McGraw-Hill. (1983)
22. Seidenberg, J., and Rector, A.: Web ontology segmentation: analysis, classification and use. In Proceedings of the 15th International World Wide Web Conference (WWW'06). (2006) 13–22
23. Stoilos, G., Stamou, G., and Kollias, S.: A string metric for ontology alignment. In Proceedings of the 4th International Semantic Web Conference (ISWC'05). (2005) 623–637
24. Stuckenschmidt, H., and Klein, M.: Structure-based partitioning of large concept hierarchies. In Proceedings of the 3rd International Semantic Web Conference (ISWC'04). (2004) 289–303