International Journal of Engineering & Technology, 7 (4.31) (2018) 290-297

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Attributes Correspondence Discovery in Ontology Instance-based Matching and RDF Data Linkage using Clustering Method

Mansir Abubakar[1], Hazlina Hamdan[2]*, Norwati Mustapha[3], Teh Noranis Mohd Aris[4]

Faculty of Computer Science and Information Technology, University Putra Malaysia,
43400 UPM Serdang Selangor Darul Ehsan
** Corresponding author Email: hazlina@upm.edu.my*

Abstract

One important aspect of ontology instance matching process is elements or attributes discovery. It specifies element correspondences in order to produce potential matching elements; otherwise, all elements of a class in the source ontology have to be compared with all elements of class in the target ontology. This heavy comparison is time-consuming and resulted in the poor performance of the matching system and makes the matching incomplete. Matching two or more ontologies and *RDF* datasets requires complete instance matching so as to establish logically equivalent relation among semantically related entities of the data sources. This deems challenging because of the existence of semantic heterogeneity and presence of irregular data in the RDF data sources which makes elements discovery and feature value extractions difficult. Thus, we proposed a four-step elements discovery method that utilizes unsupervised *K-Medoids* clustering algorithm in discovering potential matching elements pairs. To ensure generalization, we take unsupervised Canopy Clustering method to be the baseline for our evaluation. In terms of scalability, our method outperforms the baseline method with approximately 99% in both *Pair Completeness* and *Reduction Ratio* as against 60% and 86% respectively in the baseline. In mapping pattern generation, our method also outperforms the baseline algorithm with the overall F-Measure of ~91% against ~85%. The result of comparism with other methods justifies the significance effect of clustering attributes in the initial stage of the instance matching which can save about 50% of the comparism.

Keywords: *Ontology Matching; Instance Matching; Clustering Method; Feature Value Identification; Attribute Discovery; Mapping Generation.*

1. Introduction

Ontology matching refers to the finding relationships between entities of two or more ontologies represented in *Resource Description Framework (RDF)* data model to determine whether or not these ontologies can represent a single real-world entity [1]. Recent development in the areas of ontology matching focuses deeply on how matching could be done via the instances (individuals) of candidate ontologies rather than relying on ontology's classes. Instance matching problem identifies pairs of equivalent individuals (instances) that represent single real-world object [2]. In semantic web, instance matching result called alignment, identifies *owl:sameAs* relationships within pairs of instances in the candidate ontologies. The plain impression about the Instance-based matching is regarded as the high the significant of overlap in similar instances of two objects. The issue here is that how one can define the degree of significance of that overlap [3].

Current ontology matching systems suffer in determining the meaning (Semantics) of information as a result of high expressivity of heterogeneity [4]. This is because the systems can hardly discover potential attribute's correspondence at an initial stage of the matching as in the work presented by [5]. Most of the approaches depend on supervision and manual configuration in determining the potential attributes to be considered for a matching. To address the problem of supervision, we introduced four-step unsupervised learning method to classify elements of corresponding classes so that matching can perform in a complete unsupervised mode.

A web search engine requires to group documents according to the similarity of their attribute values. Business personnel require grouping their business centers based on the available history of their purchases. In these kinds of situations, the objects are subjected by feature value vectors (text, images, etc.) [6]. Therefore, in each situation, there are reasonable similarity measures that can be computed. In large-scale datasets, computing similarity pairs within objects tend to be tedious with traditional methods, in which all instances in the corresponding classes have to be compared. This may lead to a high time of execution. Thus, efficient methods are needed to overcome the problem. We proposed a clustering-based element's discovery method that can partition candidate data sets into homogeneous groups in order to produce potential attributes correspondences.

Since instance's similar elements are expected to be aligned together, it would be appropriate if the matching process should encompass a special component that would allow grouping instances attributes into similar and dissimilar clusters, so that attributes that exhibit similarity are considered for a matching to avoid unnecessary comparison of instances elements. Likewise, attributes that are not similar are grouped into a dissimilar cluster so that matching process should eliminate visiting that clusters during matching. Thus, this paper proposes a clustering algorithm that can cluster instances elements into similar and dissimilar

groups in generating potential matching element pairs. The research is limited to the enhancement of K-Medoids clustering algorithm [7] and development of cluster mapping algorithm to achieve its desired objective, all data sets used for the evaluations are standard data provided in the OAEI ontology matching campaign for the instance matching track evaluations. The result obtained is reasonably scalable with varying data size and different test cases domain in terms of *Pairs Completeness (PC)* and *Reduction Ratio (RR)*. Thus, our method outdoes the baseline in grouping class's attributes in terms of performance. The results show that *CC* algorithm cannot be predicted in loosely structured *RDF* data linkage and knowledge-based development.

The rest of the paper is organized as follows: Related works in section two, objectives in section three, methodology in section four, data sets and experimental setup in section five, discussions of results in section six, and finally concludes the paper alongside future work directions in section seven.

2. Literature Review

Several studies[8], [9], [10] were conducted to find the lasting solution to the matching of different ontologies and *RDF* data interlink, but only a few emphasize on elements discovery using unsupervised learning so as to provide a complete intelligent matching process [11].

In [12], a simple approach that has common nature in identifying similarity among concepts of ontologies was proposed. The matching process was steered in line with its algorithms and it discovered the similarities both manual and automatic form. In their method, there were differences in the respective super-concepts, sibling-concepts, sub-concepts as well as their interaction with other objects. Furthermore, they proposed the technique of the granularity in the level of similarities; it does not also consider the primitive characteristics of concepts during similarities measurement.

Machine learning models for instance matching based on some similarity metrics of instances was developed by [13]. In this technique, matching instance pairs may have some common features in the similarity metrics of each pair. Sharing some significant words is a common feature of the matching instance pairs in this model. They design a similarity vector independent of property matching to represent such features. Based on this vector, train a learning model to classify the instance pairs as matching or non-matching. To minimize the demand for training data and promote the performance, they tried to use existing instance matching information *in Linked Open Data (LOD)* for help. However, this approach does not match large-scale instance datasets efficiently as a result of pair-wise comparison which creates wider matching space.

Similar work by [14], introduced a new matching approach based on searching *Wikipedia* pages associated with the ontology terms: the classes extracted from these pages are then organized into graphs and used to match the ontology terms. This work is the extension of AgreementMaker ontology matching system. Even though, the approach allows the maximum number of comparisons between the concepts in the source and target ontologies. Therefore, advancement in attribute discovery method is required in order to minimize the number of these comparisons, such as identifying only those instances of the ontologies that need to be aligned.

In the study proposed by [15], a large-scale instance matching, *Via Multiple Indexes* (called *VMI*) by using multiple indexes and candidate selection have minimally reduces the number of comparisons. The objective of *VMI* was to match large-scale instance datasets efficiently and generates as many matching results (alignment) as possible with high quality. Precisely, *VMI* uses the vector space model to represent instances' descriptive information. It creates two types of vectors for each instance, one for names and labels of the instance and the other for descriptive information and information from neighboring instances. They built inverted indexes for these types of vectors and select matching candidates according to the indexes. In doing so, VMI is able to avoid pair-wise comparison and reduces the matching space greatly so that the matching efficiency can be improved. Then VMI compares the match pairs from user-specified properties to filter the primary candidates and improve the precision. This study has limited entity classification capability as its algorithm did not employ meta-classification strategies in the instance matching process.

A solution is proposed by [16] to resolve the following problems: Ontology heterogeneity problem, difficulty in identifying core ontology entities, missing domain or range information and machine learning for core ontology entity extraction. To solve the above problems, the study introduced the Framework named, *In-Tegrating Ontologies (FITON)*, which reduces the ontology heterogeneity in the linked datasets, retrieves core ontology entities, and automatically enhances the integrated ontology by adding the domain, range, and annotations. The main weakness of this study is that it cannot retrieve more missing links by applying similarity matching methods on the object values in the link discovery process. Its performance also declined with the growth in data size.

An approach based on Information Retrieval (IR) techniques and an indexing strategy called *ServOMap* is proposed by [17] to address the challenge of scalability and efficiency of matching techniques. One of the originalities of the approach is the reduction of the search space through the use of an efficient searching strategy over the built indexes to be matched. It showed that the introduction of a general purpose background knowledge and machine learning strategy for contextual similarity computing has a positive effect on matching performance. However, *ServOMap* is able to provide only equivalence mappings, which is a setback when dealing with some matching tasks as the recall could be negatively affected if the reference alignment is contained of subsumption and disjoint relationships. Minimally supervised instance matching system that offers a practical compromise between the performance and that of supervised systems was presented in [18]. To maximize its performance on unseen data, the system employs a meta-classification strategy called boosting. The classifier was used for probabilistic instance matching, where the classifier scores for each instance pair according to its likelihood of being a matching pair. Given the low degree of supervision, the overall output is not expected to have high quality. Instead, the system uses a small percentage of the most confidently labeled instance pairs to iteratively self-train itself in a semi-supervised fashion.

To further improve the high-quality expectation in the matching output, [19] proposed ontology matching that applies technique closely related to our approach, with the focus on the discovery of the equivalence, disjunction relations as well as the subsumption relations between the concepts of ontologies to align. The value of these measures often determines the similar/dissimilar entities of the matched ontologies. The measures defined just the equivalence and disjunction relations in this study, which did not address all ontology instance matching issues, such as interoperability or data integration among the attributes values. It is also minimally supervised method as it requires some external effort to train the system.

In [5], similarity measure integration conceptual relationship (SIMCR) is proposed, it was based on semantic relation distance within the concepts of *WordNet* and other ontologies. It considered many conceptual relations for it to refine the matching. The overall advantage of this approach is that both the semantic similarity measure and the web integration service description have significantly supported effectiveness which is among the serious challenges of ontology matching but did not scale well in varying data sizes. It's over consideration to ontology's concepts to refine the matching could not allow it to generate potential attributes that can pave a way to complete alignment generation as a system's output. Complete instance-based matching can only be possible when the matching depends heavily on the instances' elements rather than concepts' of the ontologies classes.

Due to the rapid growth in the data volume, velocity, and variety, it is found to be difficult to achieve better scalability/optimization in the matching of ontologies from the large-scale data sources, more specifically at their instance level, and the existing matching approaches. One general weakness of most approaches is a trade-off between effectiveness and efficiency in matching which is the major factor affecting the scalability of the matching systems[20]. According to the literature investigation, these problems are associated with the neglect to the early stage of the matching process, which is an insufficient utilization of method for discovering potential matching elements. In this paper, we proposed an unsupervised clustering algorithm that can cluster instances elements into similar and dissimilar groups prior to generating potential matching element pairs. This method will drastically reduce the number of comparison between ontologies class elements in generating alignment thereby reducing the cost of execution in the overall matching process.

2.1. Canopy Clustering (CC) Method

Methods of clustering have recently been successfully applied blocking high dimensional datasets with their applications to reference matching [6]. The idea of this method involves using simple, approximate distance measures to split the data into overlapping groups called *canopies*. The algorithm performs its clustering by only measuring precise distance between points commonly available in a canopy. With canopies methods, the number of clustering problems that seem impossible tune realistic. In contrast to sorted Neighborhood technique, which involves scanning the sorted records *n* from the two input files that have fixed dimension of the window, *w*. Every record pair that falls within the window is the candidates to be compared. It requires *n x w* highest comparisons. The error rate triggered by this method is unfavorably dependent on the choice of sorting keys [21]. Multiple Canopies did not depend on blocking key but receive distance function as its input. For this reason,[13] termed it as *instance-based blocking* method in contrast to feature-based blocking methods such as *Sorted Neighborhood* technique. The technique works well with different token-based measures of similarity that include *cosine similarity* and *Jaccard similarity* [22]. To test its performance on semi-structured *RDF* data, we decided to apply it as the baseline to our clustering approach.

2.2. K-Medoids Clustering Method

K-medoids is an unsupervised clustering method [20, 21], it is able to group large-scale data set into *Voronoi cells* according to data coordinates in the Euclidean space. It has time complexity $O(n^2)$, where *n* is the data volume. The goal of the K-Medoids clustering algorithm is to partition and group numerical and non-numerical attributes into many groups according to the coordinates of the elements in order to minimize the total distance between each sample point and its center. The method used in this paper utilizes this approach to create clusters because of its tolerance to irregular data which is inevitable in any *RDF* data format. One important requirement of any clustering algorithm is a good similarity measure that determines the distance between the entities to be grouped together. Achieving this good similarity is always challenged by the presence of redundant and irregular data in large RDF graph data [25]. Irregular data is the data that deviate from the normal form of the data set. It is a well-known existing issue of *RDF* datasets. Therefore, modifying a method that is tolerable to data irregularity is of credible importance when clustering with *RDF* data.

3. Objectives

The primary objective of this research is to develop a clustering-based element's discovery method that can partition candidate data sets into homogeneous groups in order to produce potential attributes correspondences. These attribute correspondence will serve as an input to the next step in the matching system pipeline. The method reduces the number of comparisons between matching candidates of say *ontology A* and *ontology B*, thereby reducing the time of producing alignments since attributes that exhibit similarity falls into a similar cluster and attributes that exhibits dissimilarity falls into the dissimilar cluster. Our algorithm will also reduce the search space in the course of matching ontology instances in contrast to blocking component found in most of the existing systems. Another objective of this approach is to test the scalability of our algorithm in different test cases that belong to different domains with varying data sizes.

4. Research Methodology

Prior to developing a matching pattern in instance-based ontology matching, an identifying group of element correspondence across the candidate classes of both source and target ontologies is necessary. We use an unsupervised clustering method to accomplish this task in order to avoid incessant supervision. This method reduces the number of potential elements to be considered for mapping which significantly minimizes the computational cost. Matching without clustering as in the case of [26] requires that all elements of class in a target ontology has to comparisons check with entire elements of corresponding source ontology class. Being element of every element correspondence share common feature values, matching elements with similar value features would justify the element correspondences. Value features describe the pattern of values [25]. Assuming a class element "email" it is feature value is "@" which appears to be common to each email address, likewise, the value feature of date is "YY/YY/YYYY". This indicates that in each date there are eight (8) numbers and two (2) symbols "/ or -" in its presentation. Thereafter, a group of elements from the corresponding classes' can be linked based on the identified features grouped according to the feature type. Feature type is feature value that expounds the values of the similar type. There is as many feature type as possible in *RDF/OWL* datasets that are hard to count. This is because many types of attributes that describes the real-world are in existence, also these type can only distinguish elements of the same type but not of different type. It is, therefore, assumes that there are only two element types (numerical and non-numerical) of each class of ontology. Figure 1 shows an excerpt of Person1 ontology that is used to form our property table as shown in Table 1.

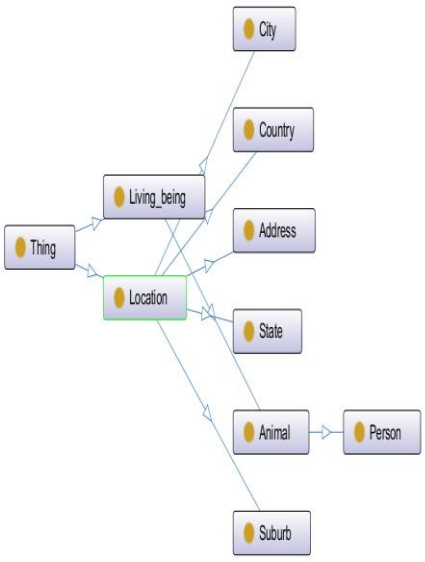

Fig. 1: Excerpt of Person Ontology: *people_1.owl* in *PR* data set

From the property table, we can see that the class elements or attributes can generally be classified into either numerical or non-numerical element. This classification would pave a way to the identification of elements' feature values to be applied in clustering. We adopted four statistical features to differentiate Non-numerical element type of each ontology class [25]. These features are based on the characteristics of both numerical and non-numerical elements. The feature values are identified via the construction of property table from the ontology's *RDF* triples (Table 1). We drive the property table from the "*Person1 ontology*" (Figure 1) in *PR* data set; the ontology consists of *person11* and *person12* designed in *RDF* data format.

Table 1: Property table for person11.rdf of Person in *PR* data set

	:Person11	**:Address**	**:State**	**:Suburb**
Attributes				
:has_given_name	:Ella	-	-	-
:has_surname	:Lunson	-		
:has_house_number	-	:173	-	-
:has_street	-	:dixon drive	-	-
:has_post_code	-	:6044	-	-
:has_date_of_birth	:19140322	-	-	-
:has_phone_number	:0494813466	-	-	-
:has_soc_sec_id	:8706620	-	-	-
:has_age	:25			
:is_in_state	-	-	:wa	-
:is_in_subarb	-	-	-	:ashcroft

The feature values for Numerical elements are:

1. Value of element's range (maximum)
2. Value of the attribute's range (minimum)
3. Average value of element's range
4. Deviance of element's range

Assume we have two properties, one regarding the age of people as in our property table (Table 1) and the other regarding the people weight in another property table. If the age is presented as 25, 30, 32, 40, in "*years*" as in our property table and the weights are presented as 53, 40, 46, 80, in "*kilogram*". The points (25, 30, 32, 40) and (52, 40, 46, 80) are quite far from each other in *Euclidean space*. Therefore, the two properties (people's age and people's weight) will be clustered into separate groups of the element. The feature values for non-numerical (sequence of characters) elements are:

1. Ratio of capitalized characters
2. The Ratio of special characters, like "@" in each email address.
3. Average length of characters
4. Average number of words

These statistical feature values are used to differentiate elements based on their non-numerical characteristics. Assume properties of people's "given name" and "street address" in our property table. The Property value of name is usually shorter than property value address in word and property value name are in most cases have capitalized the first letter as well. Properties values like email address must contain a special character "@" in it, therefore any of such property can be distinguished by the special value feature. A potential element refers to the feature pairs crosswise two linked clustered sets. We proposed 4 steps novel discovery process towards discovering element pairs using unsupervised clustering-based method (Figure 2).

4.1. Proposed Attributes Discovery Method

Elements or attributes discovery which specifies element correspondences in order to produce potentials matching elements is one important aspect of instance matching. It can reduce the heavy comparism between the different ontology classes; otherwise, all elements of a class in the source ontology have to be compared with all elements of class in the target ontology. This heavy comparison is time-consuming which can be reduced drastically by the application of appropriate elements discovery method.

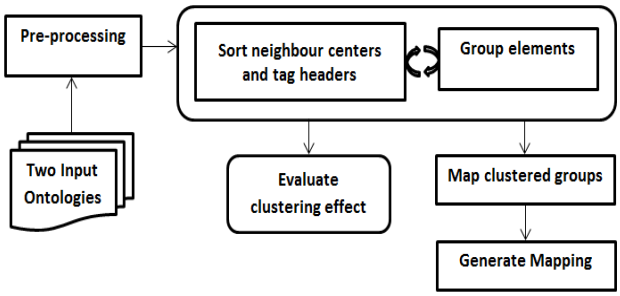

Fig. 2: Clustering-based attributes discovery process flow

Step 1: Datasets Pre-processing

The input data sets in our method are ontologies represented in *Resource Description Framework (RDF)* data format. These ontologies are categorized as source and target ontologies. The source ontology is the ontology of a given domain that satisfied to be mapped into the different ontology of the same domain. The later is the one described as target ontology. All data sets are downloaded from standard dataset sites more specifically, *OAEI* campaigns which is compliance with the *W3C* standards. We pre-processed our data by applying basic text filters: parenthesis and punctuations were eliminated; all words are converted to lower case. Sentences and phrases were tokenized by spaces so as the generated similarity between sets of words. Therefore, sentence or phrase *p* of *m* words defined as:

$$w = [wi], \tag{1}$$

where: w = a word, $i = 1, 2, 3 \ldots n$

Step 2: Clustering

This step performs basic clustering activity. The nearest neighbor approach is applied to sort element correspondences across two similar candidate classes of the given ontologies. This method treats elements of each corresponding class as a neighbor, thereafter assign a tag that distinguishes the two classes. The element of each class can be clustered based on its assigned tag. Due to the fact that web dataset (e.g. RDF and OWL format) contains irregular data, we proposed to apply K-Medoids clustering for elements classification of each class. This method has high-level of tolerance to irregular data present in data sets. Irregular data is the data that has distinct value feature with the regular data of the data set. If methods, such as k-means [27] are applied, it is normally influenced much by these irregular data. The coordinates of skew (irregular) data tend to be far from the coordinates of an ordered data in the Euclidean space.

The algorithm requires setting the initial set up of numbers $N = 20$ if the total number of elements is 50 or greater. If the values of elements are less than 50, N is set to the exact number of the element. The clustering algorithm is formally defined in algorithm 1.

Algorithm 1: Modified K-Medoid Method

```
1 Input: sample points set M representing numerical/non-numerical elements in a
class
2 Output: clusters of corresponding element based on their value features
        Initialize number N of clustered groups with iteration time T = 20;
        C = the set of N centres computed by the Forgy method;
        Clustering sample points into the nearest centres with KMedoids (M, C);
        Compute TD;
        t = 1;
        While (t<T) do
        Cnew = the set of N centres computed by the Forgy method;
        Clustering sample points into the nearest centres with KMedoids (M,
        Cnew);
        Compute TDnew;
        if TDnew<TD then
                TD=TDnew;
                        C = Cnew;
        t++;
        While (|C|>2) do
        Clustering sample points into the nearest neighbour centres with KMedoids
        (M, C);
                Compute new centres Cnew of each clustering group;
        Compute ad; /*Compute the average distance between centres*/
                C = Cnew;
        if (∃ ci,cj ∈C, distance(ci,cj)<12·ad) then
                C=MergeCenters(C, 12·ad); /*Merge nearby centres in C*/
                if (|C|<2) then
                                break;
                        else
                    break;
        Output C.
```

In the context of our clustering algorithm, the similarity function is the function that compares instances from each *RDF* graph as input and produces real value [1, 0] as an output. If similarity exists, the function will return 1, otherwise returns 0. This can generally be modelled as:

$$S(O, O') = \begin{cases} 1 & if\ O = O' \\ 0 & if\ O \neq O' \end{cases} \quad (2)$$

We applied two distance measures to calculate the distance between any two coordinates. These are *Euclidean distance* and *Cosine similarity*.

Euclidean distance [28], this function defines the Euclidean space distance between two sample points. It is formally defined as:

$$dis(x, y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2} \quad (3)$$

Cosine similarity [29], this indicates the similarity of two sample points in their orientation.

$$\cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i x\, B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2 \, x \, \sum_{i=1}^{n}(B_i)^2}} \quad (4)$$

We combined these two measures to express the distance between two coordinates with a multipurpose view. Therefore, the distance between two points is calculated as the average value of these two measures. Since web data contains irregular data, we chose to use K-medoids clustering algorithm to classify elements of each class. K-medoids clustering is an unsupervised learning technique that can partition large-scale data into *Voronoi cells* based on the Euclidean space coordinates [28]. This algorithm has $O(k^2)$ as its complexity in time, where *k* is the data size.

Step 3: Map Clustered Groups

Having classified the element of each class into different groups through clustering (Step 2), it is easy to construct potential matching elements correspondences by mapping group of corresponding classes *C* in *O1* and *C'* in *O2*. The mapping here should be 1:1, where each cluster of the *C* class mapped to the similar cluster of the C' class based upon the coordinates of cluster centers. Note that numerical and non-numerical elements are disjointedly clustered. Therefore numerical elements of each class are only mapped to a numerical element of the corresponding class, similarly for non-numerical element class. Algorithm 2 describes the mapping method used in this work.

Algorithm 2: Clusters Mapping

1. ***Input:*** *Class properties P and P' from property tables*
2. ***Output:*** *One-to-one mapping element pairs É*

Initialize matrix M of |P| and |P'|;

Invoke similarity function to form M;

Map P and P' if similarity exist in M;

Repeat step 2 if step 3 fails, else;

P semantically equivalent to P';

Output É.

Step 4: Generate Potential Elements

The final stage of the method is constructing potential attributes based on the group mappings performed in step 2 and cluster mapping is done in step 3. Assume *n* group of string (non-numerical) attributes belongs to class *C*, represented by $c1, c2, c3, \ldots cn$. We have adopted four statistical value features identified by [25] for constructing the coordinates of the elements as described in section IV. Clustered group's center is represented by (xi, yi, zi, wi), where $i = 1, 2, 3, \ldots, n$. Let *m* be groups of string (non-numerical) elements in C' class, denoted by $c'1, c'2, c'3, \ldots c'm$. The clustered group's centers is $((x'j, y'j, z'j, w'j)$, where, $j = 1, 2, 3, \ldots m$. Thus, the distance between c_i and c'_j is shown by their coordinates distance, indicated as $(d(ci, c'j))$ the groups mapping can be based on the given function:

$\{arg\ min\ d(ci, c'j)\}$,

where: *arg min* describes the index value *(i, j)* with a small value of $(d(ci, c'j)$. *i, j = 1, 2, 3 …, n*

This indicates that in each group c_i in *C* class, the mapping group is the corresponding c_j in *C'* class that has the nearest center. With the sets of mapped groups of elements, potential elements correspondences would then be generated.

This method reduces the number of comparisons between matching candidates of say *ontology A* and *ontology B*, thereby reducing the time of producing alignments, since attributes that exhibit similarity falls into a similar cluster and attributes that exhibits dissimilarity falls into the dissimilar cluster. Our algorithm will also reduce the search space in the course of matching ontology instances in contrast to blocking component or linkage specification component of the traditional matching process [30] which requires domain expert or learning from prior knowledge.

5. Data Sets and Experimental Setup

Table 2 illustrates the test cases used in this evaluation; all test cases are structurally heterogeneous, real-world and standard data provided in *IM@OAEI* campaign. We chose to evaluate our method with varying large-scale test cases for efficiency and scalability determination. The data sets are open access in *OAEI* website alongside their necessary information. All experiments were run on *Python Jupyter Notebook 4.3.1* and *Weka* Machine learning tool.

Table 2: Datasets statistics

Matching Task	Ontology's Classes	# of Instance	# of Instance Pairs
IIMB-Small	Film:Science_fiction Film:Science_fiction	581 222	581x222= 128982
Sanbox	owl:NamedIndividual owl:NamedIndividual	363 367	363x367= 133221
Person	Person_1: Person Person_2: Person	600 400	600x400= 240000
SABINE	Source: Topic Target: Topic	706 1127	706x1127= 795662
IIMB-Large	Film:Science_fiction Film:Science_fiction	2150 1568	2150x1568= 3371200
IM-Similarity	Similarity_a:Book Similarity_b:Book	1675 1658	1675x1658= 2777150
IM-Identity	identity_a: Book identity_b: Book	1330 2649	1330x2649= 3523170

5.1. Scalability Evaluation

To avoid the trade-off between efficiency and effectiveness, we evaluate the algorithm by applying the blocking metrics of *Pair*

Completeness (PC) and *Reduction Ratio (RR)*. To ensure generalization, we take unsupervised blocking method, *Canopies* (Canopy Clustering or CC) by [6] to be our baseline. *CC* algorithm is incorporated in *Weka* platform. The experiment will determine if simple clustering algorithms are warranted in real-world cases by running *CC* on each test case. The main experiment reported in the results and discussion section evaluates the K-medoids clustering effect against the state-of-the-art trigrams-based *CC* algorithm baseline. This method would mines trigrams from each element value in the dataset, and then cluster attributes by computing the similarity between trigram sets of values. Similar attributes in one cluster and dissimilar attributes in the other cluster.

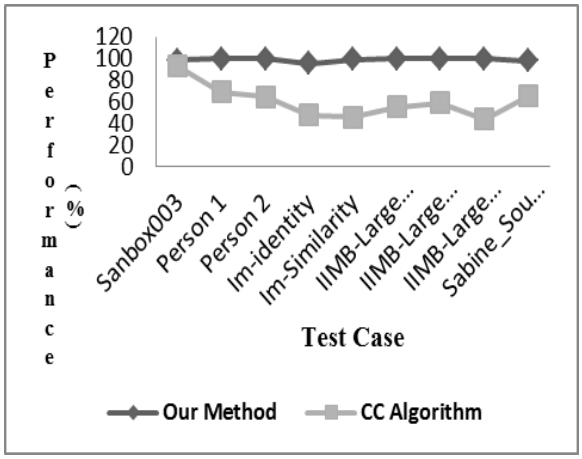

Fig. 3: Scalability Performance in terms of *PC* of our method against the baseline *CC algorithm* in different test cases. (PC, stand for Pair Completeness. All values are presented in percentage)

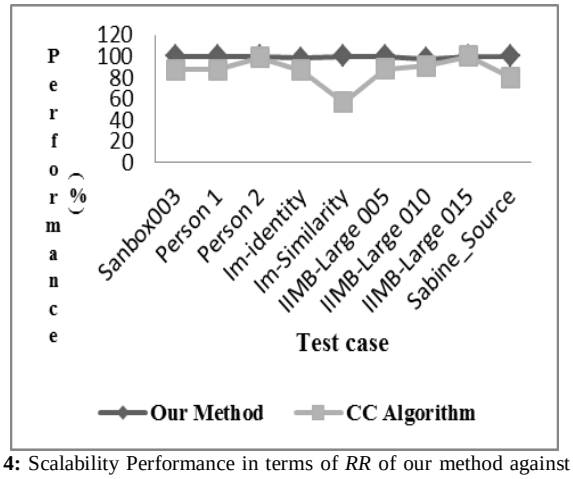

Fig. 4: Scalability Performance in terms of *RR* of our method against the baseline *CC algorithm* in different test cases. (*RR*, stand for Reduction Ratio. All values are presented in percentage)

The average PC of 60% and RR below 86% is low compared to our state-of-the-art method with the average of 99% in both *PC* and *RR*, (Figure 3 and 4). Our method tries as much as possible to maintain near-linear scalability as an increase in the size of the test cases data does not affect its performance in contrast to the baseline *CC*. Therefore, the result indicates that the performance of *CC* algorithm cannot be predicted in *RDF* data and knowledge-based applications.

Table 3: Run Time results of Different Clustering Algorithms measured in Nine Test Cases (all run-times are measured in seconds)

Test Case	Canopy Method	Our Method	Simple Kmeans	Xmeans
Sandbox003	0.61	0.30	0.41	1.55
Person_1	0.61	0.29	0.32	1.52
Person_2	0.61	0.29	0.32	1.53
Im-Identity	0.61	0.28	0.32	1.52
Im_Similarity	0.61	0.30	0.32	1.52
IIMB_Large_005	0.61	0.28	0.32	1.47
IIMB_large_010	0.41	0.28	0.31	1.52
IIMB_Large015	0.41	0.28	0.32	1.51
Sabine_Source	0.39	0.28	0.31	1.51
Average	**0.54**	**0.29**	**0.33**	**1.52**

5.2. Mapping Pattern Evaluation

We use a test cases *Person* to evaluate the mapping pattern. In order to acquire maximum and reasonable F-measure, we consider only properties of the mapping task. The prefix of the test case use is given as *http://www.okkam.org/ontology_person1.owl#* to indicate *Person_1andhttp://www.okkam.org/ontology_person2.owl#* to represent *Person_2* in the *PR* data set. In our property table of a *person_1* (Table 1), there are six attribute correspondence to be considered for a mapping: *given_name, Surname, Soc_Sec_ID, date_of_birth, age, and Phone_number*. These same properties are found in the corresponding ontology class person_2 for mapping consideration. Table 4 shows the maximum performance of our mapping pattern based on the properties outlined.

To evaluate the performance of our technique in terms of mapping generation, we employed the popular measurement metrics used in information and data retrieval systems; these are *precision and recall*. We also measure the harmonic mean *(F-measure)* of these metrics. Systems with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision will return very few results, but most of its predicted labels are correct when compared to the training labels. The best system with high precision and high recall will return many results, with all results labeled correctly. Example of "Precision-Recall" performance is shown in "Precision-Recall Curve" (Figure 5). Therefore, we target high precision as well as high recall in our mapping pattern in order to have an ideal matching system. The best result of the mapping is recorded and shown in Table 3.

Table 4: Mapping Result in *Person_1* and *Person_2* of *PR* data sets

Mapping Properties	1st Property	2nd Property	3rd Property	4th Property	5th Property	6th Property
Precision	1.00	0.93	0.93	0.97	0.90	0.96
Recall	1.00	0.95	0.95	0.94	0.90	1.00
F-measure	1.00	0.92	0.99	0.94	0.91	0.98

In the result table, the 1st property represents given_name↔given_name mapping in the two data sets person_1 and Person_2, 2nd property represent *surname↔surname mapping,* the 3rd property stands for *soc_sec_number*↔soc_sec_number mapping, 4th property stand for date_of_birth↔date_of_birth mapping, 5th property represent age↔age mapping and 6th property stands for phone_number↔phone_number mapping. Figure 5 shows the Precision-Recall Curve that indicates the mapping precision against the mapping recall of the 1st property (given_name↔given_name) mapping between person_1 and Person_2 ontologies in which the presentations of the property "*given_name*" of a class "*:Person*" in both ontologies appeared to be "*Ella"* that makes the classes to be structurally and semantically equivalent. Sample property mapping pattern is given as:

1. The property mapping from the ontology_1 property birth_date to the ontology_2 property birthdate:

{PropertyMapping
Ontology_1Property = birth_date

Ontology_2Property = birthdate}

2. The property mapping for the first value of the spouses' property:

{PropertyMapping
Ontology_1Property = spouses
Ontology_2Property = firstSpouse
Select = first}

3. The property mapping from the ontology_1 property phone_number to the ontology_2 property contactNumber:

{PropertyMapping
Ontology_1Property = phone_number
Ontology_2Property = contactNumber}

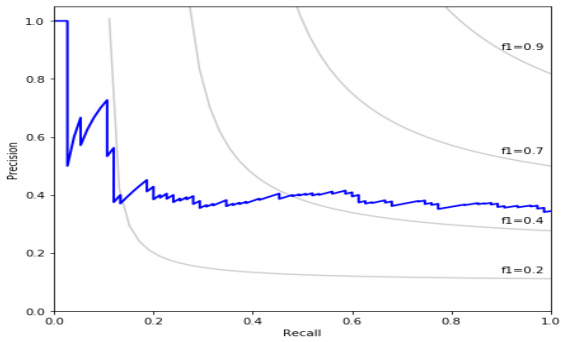

Fig. 5: Example of Precision-Recall Curve: 1st Property (given_name↔given_name)

5.3. Evaluation of Different Methods

To validate our method, we measure its mapping performance against state-of-the-art algorithms, notably those present in weka machine learning platform. The result obtained shows that our method outperforms all algorithms compared as can be viewed in Table 5. The high F-measure of 0.913 in our method against 0.849 in Canopy CC demonstrated a significant improvement over the baseline canopy CC performance in generating mapping pairs.

Table 5: Result of Comparison with State-of-the-art Algorithms

Algorithm	Precision	Recall	F-Measure
Our Method	**0.951**	**0.879**	**0.913**
Canopy	0.795	0.912	0.849
Simple K-Means	0.518	0.518	0.518
Xmeans	0.556	0.059	0.106

5.4. Comparison with State-of-the-art Mapping Frameworks

We compare the performance measures of our method with mapping frameworks that participates in 2014 OAEI annual campaign. The metrics shown in Figure 6 testify that our method performs better than the high performed participant RiMOM-IM with the F-score of 0.5581 as against our state-of-the-art method with the maximum F-score of 0.913. Both data and the results compared are publicly available at *OAEI* website.

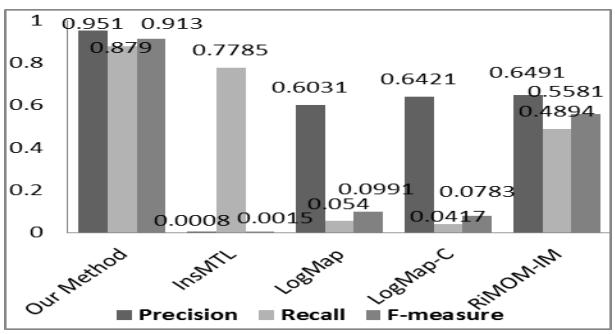

Fig. 6: Performance Comparison with *OAEI2014 Instance Matching Track* Participants

6. Discussions of Results

We have evaluated our algorithm in the perspective of its performance scalability against the baseline *CC* algorithm. The objective is to have a linear stability in our algorithm with varying data size. The baseline CC achieves above 90% PC in one test case (Sandbox003) only, the graphs also shows unpredictable attitude in both *PC* and *RR* compared to the performance of our algorithm that achieves above 97% throughout the 9 test cases. The *RR* of *CC* algorithm recorded above 90% in 3 test cases and its performance generally exhibits much deviation as the trade-off between PC and RR resulted in a low performance of the *CC* algorithm in maintaining scalability with varying test data. Furthermore, the algorithm ran quite slow on large data sets, and the threshold parameter had to be tuned separately in each run.

The result (Table 3) shows that our method outperforms all the algorithms compared in regards to the running time. We ran each test case 10 times and recorded the maximum running time. In all test cases, our approach performs best with the highest running time of 0.3 seconds in *Sandbox003* and *Im-Similarity* test cases. These results justify the performance scalability of our method in terms of execution time considering the fact that Sandbox003 test case contains 133,221 instance pairs compared to *Im-Similarity* test case with 2,777,150 instance pairs, but recorded same running time. The result also affirms to our effort of reducing the entire running time when producing alignments, alongside constructing domain-independence ontology matching system. In comparison to the baseline *CC* method [6], the result indicates that the running time of our approach also recorded significance difference with about 50% difference in their running time averages. This indicates that our method can generate mapping pairs as faster as 50% difference with the baseline canopy algorithm.

The proposed mapping pattern was evaluated gains the state-of-the-art mapping frameworks [31], [32] and [33] that participates in an annual ontology alignment campaign in 2014, were our method outperform all participant in both precision and recall of the mapping which results in recording the highest F-measure of 0.913 against the high performed participant RiMOM [33] with F-measure 0.5581.

7. Conclusion and Future Work

In ontology matching and RDF data interlink, element correspondence discovery is an initial priority considering the nature of the growth in *LOD* that makes data homogeneity a challenging issue. Without an efficient element discovery method, all elements of a candidates instance in source dataset have to be compared with all elements of an instance in a target dataset. These heavy comparisons resulted to the computational complexity of the matching or interlinking system. In this paper, we proposed a four-step method of discovering elements via clustering the classes attributes into either numerical or non-numerical group to determine the inter-cluster distance between the classes elements of the given two ontologies, and also establish within-cluster similarity relation pairs that can be perfectly mapped together. The aim of adapting K-medoids clustering algorithm is to group numerical and non-numerical attributes into many groups according to the coordinates of the elements in order to minimize the total distance between each sample point. Our evaluation results on PC and RR shows that the performance of our method outperforms the baseline *CC algorithm* and can be predicated in *RDF* data and knowledge-based applications. Yet, the execution time of our approach outperforms the baseline and other popular algorithms present in Weka platform. Our method also demonstrated better performance in mapping generation with high F-measure against the state-of-the-art mapping frameworks. However, there is still need for further refinement of the algorithm, especially, in selecting initial clusters to aim 1.0 maximum F-score.

In future, we need to evaluate our method in as many test cases as possible that belongs to a big data category to ensure absolute scalability of the method. There is also need to select initial clusters with a different method than the *Forgy method* applied in our approach, as well as to test the complexity independent to alignment generation. In our ongoing project, we hope to determine whether the disjunctive pattern is required to express the mapping pattern of several categories of ontologies that are represented in RDF data format.

Acknowledgment

This work is supported by University Putra Malaysia grant (Putra Grant, No: 9569200) and Al-Qalam University Katsina (AUK), Katsina State Nigeria under the University's Staff Development Unit.

References

[1] Gracia J, Mena E, Semantic heterogeneity issues on the web, *IEEE Internet Computing*, Vol. 16, No. 5, (2012), pp. 60–67.

[2] Li L, Xing H, Xia H, Huang X, Entropy-Weighted instance matching between different sourcing points of interest, *Entropy*, Vol. 18, No. 2, (2016), pp. 1–15.

[3] Isaac A, Van der Meij L, Schlobach S, Wang S, An empirical study of instance-based ontology matching, *Belgian/Netherlands Artificial. Intelligence Conference*, (2008), pp: 317–318.

[4] Castano S, Ferrara A, Montanelli S, Varese G, Ontology and instance matching, *Lecture Notes Computer Sciience (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 6050, (2011), pp. 167–195.

[5] Chen F, Lu C, Wu H, Li M, A semantic similarity measure integrating multiple conceptual relationships for web service discovery, *Expert Systems with Application*, Vol. 67, (2017), pp. 19–31.

[6] Mccallum A, Ungar LH, Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching, Vol. 64, No. (2013), pp. 213–223.

[7] Choi DW, Chung CW, A K-partitioning algorithm for clustering large-scale spatio-textual data, *Information System*, Vol. 64, No. June 2016, (2017), pp. 1–11.

[8] Pawar P, Tokmakoff A, Ontology-based context-aware service discovery for pervasive environments, *Proc. First IEEE Int. Workshop on Service Integration Pervasive Environ.*, 2006.

[9] Arch-Int N, Arch-Int S, Semantic Ontology Mapping for Interoperability of Learning Resource Systems using a rule-based reasoning approach, *Expert Systems with Applications*, Vol. 40, No. 18, (2013), pp. 7428–7443.

[10] Liu L, Yang F, Zhang P, Wu JY, and Hu L, SVM-based ontology matching approach, *International Journal of Automative Computing*, Vol. 9, No. 3, (2012), pp. 306–314.

[11] Bilgin AS, Singh MP, A DAML-based repository for QoS-aware semantic Web service selection, *Proc. - IEEE International Conference on Web Services*, (2004), pp. 368–375.

[12] Farooq A, Ahsan S, Shah A, An Efficient Technique for Similarity Identification between Ontologies, *Computing*, Vol. 2, No. 6, (2010), pp. 147–155.

[13] Rong S, Niu X, Xiang EW, Wang H, Yang Q, A Machine Learning Approach for Instance Matching Based on Similarity Metrics, *11th International Semantic Web Conference*, (2012), pp. 1–16.

[14] Cruz IF, Palmonari M, Caimi F, Stroe C, Building linked ontologies with high precision using subclass mapping discovery, *Artificial Intelligence Review*, Vol. 40, No. 2, (2013), pp. 127–145.

[15] Li J, Wang Z, Zhang X, Tang J, Large scale instance matching via multiple indexes and candidate selection, *Knowledge-Based Systems*, Vol. 50, (2013), pp. 112–120.

[16] Zhao L, Ichise R, Ontology Integration for Linked Data, *Journal of Data Semantics*, Vol. 3, No. 4, (2014), pp. 237–254.

[17] Diallo G, An effective method of large scale ontology matching, *Journal of Biomedical Semantics*, Vol. 5, No. 1, (2014), p. 44.

[18] Kejriwal M, Miranker DP, Semi-supervised instance matching using boosted classifiers, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9088, (2015), pp. 388–402.

[19] Gherbi S, Khadir MT, Inferred Ontology Concepts Alignment Using Instances and an External Dictionary, *Procedia Computer Scince*, Vol. 83, No. Ant, (2016), pp. 648–652.

[20] Teh Noranis MA, Mansir A, Hazlina H, Norwati M, Instance-Based Ontology Matching : A Literature Review, in *Recent Advances on Soft Computing and Data Mining, Advances in Intelligent Systems and Computing*, (2018), pp. 456–469.

[21] Noy NF, McGuinness DL, Ontology Development 101: A Guide to Creating Your First Ontology, *Stanford Knowledge Systems Lab.*, (2001), pp. 25, 2001.

[22] Cao F, Huang JZ, Liang J, A fuzzy SV-k-modes algorithm for clustering categorical data with set-valued attributes, *Applied Mathematical Computing*, Vol. 295, (2017), pp. 1–15.

[23] Hopke PK, The Use of Sampling to Cluster Large Data Sets, *Chemom. Intelligent. Laboratory Sytems*, Vol. 8, (1990), pp. 195–204.

[24] Huang Z, Extensions to the k -Means Algorithm for Clustering Large Data Sets with Categorical Values, *Data Mining Knowledge Discovery*, Vol. 304, No. 2, (1998), pp. 283–304.

[25] Fan Z, Euzenat J, Scharffe F, Learning concise pattern for interlinking with extended version space, *Proc. - 2014 IEEE/WIC/ACM International. Joint. Conference of Web Intelligence and Intelligent Agent Technology Workshop, WI-IAT 2014*, Vol. 1, No. 3, (2014), pp. 189–204.

[26] Cerón-Figueroa S, *et al*., Instance-based ontology matching for e-learning material using an associative pattern classifier, *Computers in Human Behaviour*, Vol. 69, (2017), pp. 218–225.

[27] Wagstaff K, Rogers S, Schroedl S, Constrained K-means Clustering with Background Knowledge, *Expert Systems with Applications*, (2001), pp. 577–584.

[28] Greenacre M, Primicerio R, Measures of distance between samples: Euclidean, *Multivariant Analytic Ecological Data*, (2013), pp. 47–59.

[29] Garcia E, Co A, Cosine Similarity Tutorial, *Information Retrieval Intelligence*, (2015), pp. 4–10.

[30] Noy N, Ontology Mapping and Alignment, *3rd Summer School. Ontology Engineering*, (2005), pp. 48.

[31] Jiménez-Ruiz E, Cuenca GB, LogMap: Logic-based and scalable ontology matching, *Lecture Notes in Computer Science (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, Vol. 7031 LNCS, No. PART 1, (2011), pp. 273–288.

[32] Khiat A, Benaissa M, InsMT / InsMTL Results for OAEI 2014 Instance Matching, in *CEUR Workshop Proceedings (Vol. 1545,). CEUR-WS.*, (2014), pp. 158–161.

[33] Shao C, Hu LM, Li JZ, Wang ZC, Chung T, Xia JB, RiMOM-IM: A Novel Iterative Framework for Instance Matching, *Journal of Computer Science Technoogy*, Vol. 31, No. 1, (2016), pp. 185–197.