IJST, Transactions of Electrical Engineering, Vol. 36, No. E2, pp 95-108…
Printed in The Islamic Republic of Iran, 2012
© Shiraz University

PRE-PROCESSING ONTOLOGIES TO IMPROVE THE RESULTS OF MATCHERS[*]

B. BEHKAMAL[**1], M. NAGHIBZADEH[2] AND R. ASKARI MOGHADAM[3]

[1]Young Researchers Club and Elites, Mashhad Branch, Islamic Azad University, Mashhad, I. R. of Iran
Email: behkamal@mshdiau.ac.ir
[2]Dept. of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, I. R. of Iran
[3]Faculty of New Sciences and Technologies, University of Tehran, Tehran, I. R. of Iran, P.O.Box:14395-1561

Abstract– Ontology matching is one of the most important topics in many fields of semantic web research. The focus of ontology matching is on discovering semantic correspondence between entities of differently modelled ontologies of the same domain. Although there have been many efforts on developing ontology matching tools, the results of matching process is not appealing. In this paper, a new approach is proposed in order to get a better result from matching process. In this approach a pre-processing phase is added to matchers for analysing and modifying input ontologies. Refactoring rules are utilized on detected inappropriate patterns. Matching algorithms are then applied on the refactored ontologies. To evaluate our proposed approach two well-known matchers, RiMOM, and ASMOV, are used. Evaluation results show that our proposed approach improved the quality of the matching process with respect to standard evaluation measurements such as precision, recall, and F-measure.

Keywords– Ontology matching, ontology alignment, ontology patterns, naming patterns, ontology refactoring, semantic web

1. INTRODUCTION

Semantic web is a web with formalized knowledge and data that can be processed by humans and computers. It provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. The growth rate of data and information resources on the web makes the need for data integration critical. Semantic web designers try to find appropriate solutions for detecting the semantic interoperability between concepts which are modelled in the same domains. One approach which is proposed to tackle this is the ontology matching technique. The aim of this process is to discover interchangeable and synonymous concepts and relations of the ontological content. The matching process takes two ontologies as inputs and then determines the relationships between them as an output. Each of the ontologies is composed of a set of separate entities, like classes, properties, and individuals. The output relations could be equivalence, subsumption, overlapping, and disjointness [1]. Ontology matching is an important operation in traditional applications, such as ontology integration, schema integration, data warehouses, e-commerce, query mediation, distributed expert system, and others. Typically, these applications are characterized by heterogeneous structural models that are analyzed and matched either manually or semi-automatically at the time of design. Thus far, many different matchers have been proposed which use diverse methods to find the associations between ontological components. Despite serious research efforts to solve matching problems, matchers still suffer from severe difficulties

[*]Received by the editors April 11, 2012; Accepted December 29, 2012.
[**]Corresponding author

associated with the quality of matching results. This is a challenge in the field of ontology matching namely, uncertainty, which is pointed out in [2]. Some issues dealing with the uncertainty in ontology matching have been addressed in previous research [3-7].

In this paper, an approach is proposed in order to reduce the uncertainty in alignment results. In this approach, a pre-processing phase is added to matchers. At first, we analyze some matchers [8-15] and alignments achieved from them. To accomplish the approach, many input ontologies are analyzed in order for inappropriate patterns modelled by various developers to be detected. Then, refactoring rules are applied on detected patterns to repair the input ontologies. Next, our work is evaluated by running some matchers on the repaired ontologies. Experimental results show that better results and high quality alignment are achieved from matchers on refactored ontologies as opposed to the original ones.

The rest of the paper is organized as follows: Section 2 presents some introductory definitions of concepts which are relevant to this literature. Section 3 gives an overview of related works and Section 4 describes the problem definition. Section 5 elaborates on the proposed approach. In Section 6, the experimental results are presented. Finally, Section 7 provides the conclusion and future work.

2. BACKGROUND

The following are some preliminary definitions and terms which are relevant to this literature and used throughout this paper.

a) Ontology

An ontology *O* contains a set of entities related to a number of relations. Entities of ontology can be divided into components as follows [8]:

- Classes (*C*): Classes define the concepts within the ontology
- Individuals (*I*): Individuals denote the object instances of classes
- Literals (*L*): Literals represent concrete data values
- Data types (*T*): Data types determine the possible types of those values
- Object properties (*OP*): Include the definitions of possible associations between two individuals
- Data type properties (*DP*): Include the definitions of possible associations between one individual and a literal

There are four specific relations in ontology matching: equivalence, subsumption, disjointness, and membership.

b) Matching process

Matching is the process of finding relationships or correspondences between entities of different ontologies. The matching operation determines the alignment A' for a pair of ontologies o and o'. There are some other parameters that can extend the definition of the matching process, namely: (i) the use of an input alignment *A*, which is to be completed by the process; (ii) the matching parameters *p*, e.g. weights, thresholds; and (iii) external resources used by the matching process *r*, e.g. common knowledge and domain specific thesaurus.

Generally, matching algorithms can be classified based on (I) input of the algorithms, (II) characteristics of the matching process, and (III) output of the algorithms. The input dimension focuses on the input type on which algorithms operate. Algorithms can be classified depending on the data/conceptual models in which ontologies or schemas are described. The matching process can be based on its general properties. In particular, this depends on the approximate or exact nature of its computation. The output of an algorithm is related to the form of the alignment. For example, the correspondence between ontology entities is one-to-one or not at all. Another dimension concerns the kind of relations between entities that a system can provide. Most of the systems focus on equivalence (=), while a few others are able to provide more expressive results (e.g., equivalence, subsumption and incompatibility) [1, 16].

c) Correspondence

A correspondence between an entity *e* belonging to ontology *o* and an entity *e'* belonging to ontology *o'* is a 5-tuple <*id, e, e', R, conf*> where: *id* is a unique identifier of the correspondence; *e* and *e'* are the entities (e.g. properties, classes, individuals) of *o* and *o'*; *R* is a relation and *Conf* is a confidence measure (typically in the [0, 1] range) holding for the correspondence between the entities *e* and *e'* [17].

d) Alignment

The alignment of ontologies o and o' is a set of correspondences between two or more (in the case of multiple matching) ontologies. The alignment is the output of the matching process between the entities of o and o' [1].

e) Refactoring

Refactoring is recognized as a change made to the internal structure of the software in order to make it easier to understand and to modify without changing its observable behaviour [18].

3. RELATED WORKS

In this paper, we attempt to combine three apparently distant areas: ontology matching, ontology patterns, and ontology refactoring. Accordingly, in this section, some research conducted in each of these areas is described.

Research in ontology matching has been burgeoning since the early 2000's. So far, most articles on the ontology matching field have focused on the method of matching processes and have introduced some matchers with diverse algorithms. Here, are introduced some matchers having high ranks in the ontology alignment evaluation initiative (OAEI). ASMOV (Automated Semantic Matching of Ontologies with Verification) [8] is a novel algorithm using the lexical and structural characteristics of two ontologies to iteratively calculate a similarity measure between them, derive an alignment, and then verify it to ensure that it does not contain semantic inconsistencies. RiMOM [11] is a dynamic multi-strategy ontology alignment framework that combines multiple strategies to improve matching efficiency. The key intuition in this framework is that similarity characteristics between ontologies may vary widely. This approach considered both the textual and structural characteristics of ontologies. RiMOM is a framework based on risk minimization of the Bayesian decision. It employs multiple ontology alignment strategies and sets the combination weight. Another system is Falcon-AO [9], a practical ontology matching system with good performance that acts based on a number of remarkable features. It is an automatic ontology matching system that uses multiple elementary matchers (V-Doc, GMO and PBM), coordination rules and the similarity combination strategy. PROMPT [15] algorithm consists of an interactive ontology merging tool and a graph-based mapping called Anchor-PROMPT. Anchor-PROMPT [14] uses linguistic “anchors” as a starting point and analyses these anchors in terms of the structure of the ontologies. GLUE [19] discovers mappings through multiple learners which analyse the taxonomy and the information within concept instances of ontologies. S-Match [12] is a deductive technique for semantic ontology matching which employs a number of elemental level matchers to express ontologies as logical formulas and then a propositional satisfiability (SAT) solver to check the validity of these formulas.

Generally, all of the above matching algorithms are classified into two levels: elemental and structural. Elemental level matching techniques compute matching elements by analysing entities in isolation and ignoring their relations with other entities. Structural level techniques compute matching elements by analysing how entities appear together in a structure and considering the relation of concepts in taxonomy tree [16].

Previous works on ontology patterns in the field of ontology matching are also limited [20-23]. Ontology patterns have been used in many fields such as ontology engineering, but they have rarely been applied to the field of ontology matching. Ontology patterns are mainly inspired by software engineering and knowledge engineering [24]. Here, we describe some previous works in the field of ontology matching by considering ontology patterns. The paper in [20] involves testing the impact of ontology refactoring on the results of three matcher algorithms, namely HMatch, Falcon-AO, and ASMOV. In this paper, some modelling errors via name structure analysis were found, to which three refactoring operations were applied. By considering semantic structures, authors in [21] analysed collections of OWL ontologies in order to determine the number of occurrences of several combined name and graph patterns. These structures ranged from simple subsumption to more complex constructions. The goal of this paper is to facilitate automatic alignment among different models by finding such patterns in the given ontologies. In [22], the authors concentrate on the detection and mutual matching of semantic structures in ontologies. The authors use the equivalence relation, as well as analysing homogeneous correspondence. Research in [23] presents a simple method of tracking name patterns over OWL ontology taxonomies. This method helps to detect several probable taxonomic errors and modelling inconsistencies with respect to their set-theoretic interpretations.

Although ontology refactoring is employed in many different areas [25-28], the impact of ontology refactoring on the ontology matching field is rarely discussed [20, 22, 29]. In [25], the authors focus on the detection of anomalies as an important criterion for verification. In this paper, some approaches for the syntactic verification of ontologies are explained and definitions are extended with respect to the existence of rules. Furthermore, novel measures are introduced for detecting the parts of the ontology that may create problems for maintainability. This paper [26] proposed an approach for refactoring multimodal knowledge on the basis of a generic data structure, KDOM, in order to support the representation of multimodal knowledge. Moreover, how this data structure was created from given documents (i.e. the most general mode of knowledge) was explained, along with how different refactoring could be performed by considering various levels of formality. In [28], the authors present the semantic knowledge wiki, Know WE, used to capture and share ontological knowledge for the effective elicitation of problem solving knowledge. Also, a distributed knowledge acquisition process and refactoring phase are shown. In [29], a semi-automatic process for lifting meta-models into ontologies is proposed that allows creating the semantic integration of modelling languages. In so doing, implicit concepts in the meta-model are changed to explicit concepts in the ontology. Thus, the implementation of a certain modelling language towards the explicit reification of the concepts is covered by this language. The application of refactoring patterns on the resulting ontologies could improve automation support for semantic integration tasks. The paper [27] presents a method to develop conceptual schemas as refinements of more general ontologies. For obtaining final conceptual schemas, three activities are performed: refinement, pruning, and refactoring. The refinement phase is done to execute a set of additive operations to the ontology to create necessary elements. Afterwards, in the pruning phase, some unnecessary elements are deleted. Then, a pruned ontology is obtained. At the end, the pruned ontology can be improved by using refactoring operations to obtain the final conceptual schema.

4. PROBLEM DEFINITION

Due to the vast growth of the numerous information resources on the web, handling the heterogeneity concepts on the web is becoming more challenging. In the semantic web, data essentially comes from diverse ontologies and the process of information across ontologies is not possible without knowing the semantic relations between them. Therefore, ontology matching becomes a critical problem in many application domains, such as ontology integration, data warehouses, e-commerce, query mediation, catalogue matching, and more. All these applications can profit from ontology alignment, that is, a set of correspondences between the entities of two ontologies. Each correspondence represents a kind of relationship between entities or, more generally, between whole semantic structures. Various solutions were proposed to solve problems dealing with the automatic integration of distributed resources. Therefore, ontology matchers use diverse methods to find correspondences.

Despite many efforts in the field of ontology matching, alignments achieved from matchers have not had high quality. A higher quality could be achieved by manually designing alignments, which is a very demanding method. On the other hand, manual and semi-automatic methods are not very efficient because they are time-consuming. Therefore, most recent techniques for overcoming matching issues concentrate on a fully automatic approach. One challenge in ontology matching, especially for fully automatic matchers, is uncertainty in ontology matching, as pointed out in [2]. One reason for this occurrence is that, in fully automatic techniques, human interaction is limited. Therefore, the result of this matching process is not very acceptable for users. This paper proposes an approach to facilitate this situation and to achieve optimum results from the matching process, especially for fully automatic matchers. In the proposed approach a pre-processing phase is added to matchers in order to find the different lexical and structural patterns. In this phase, as explained in Section 5, various patterns are detected, on which the refactoring operations are applied to repair them. This renders ontologies easier to understand by both humans and matchers and also helps avoid some common mistakes in the alignment results. Therefore, better results from the matching process are achieved by applying refactored ontologies as opposed to the original unrepaired ones.

5. THE PROPOSED APPROACH

The goal of ontology matching is generating correct correspondences from the matching process. In some cases, either incorrect correspondences are discovered or correct correspondences are not found. Therefore, in this section, an approach is proposed for improving the quality of the matching results. The aim of this approach to improve the alignment result by adding a pre-processing phase to matchers. The work flow is illustrated in Fig. 1. In the pre-processing phase, at first, a comprehensive survey on some ontologies is performed in order to find inconsistencies in ontologies. Then various lexical and structural patterns are detected. Afterward, some refactoring operations are applied to repair the incompatible ontologies. Finally, this process is evaluated by two high level matchers in OAEI2009 [30] and OAEI 2008 [31], ASMOV and RiMOM. Experimental results indicate that better outcomes can be achieved by applying the matching process on refactored ontologies as opposed to original ones.

a) Ontologies considered in the study

We chose some ontologies from the conference track and benchmark track of the OAEI data set. These ontologies are used by the most popular matchers participating in the OAEI contest. The conference track, which is about conference organization, contains 16 ontologies with the same domain. These ontologies are suitable for the ontology matching task because of their heterogeneous character of origin. They have been developed by the OntoFarm project [32].

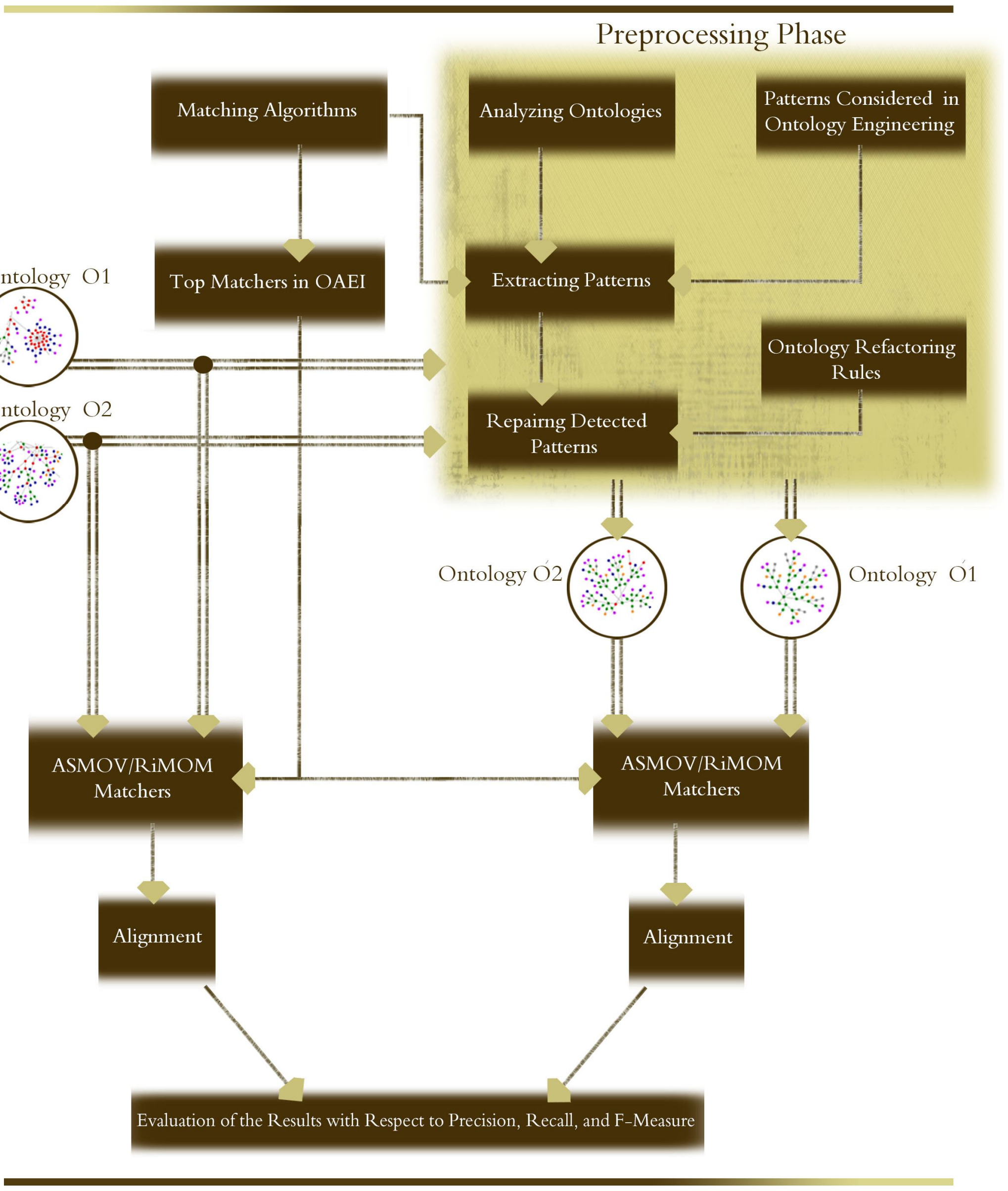


Fig. 1. Proposed approach diagram

b) The proposed patterns

Some non-uniform patterns used in the design of ontologies negatively affect the ontology matching process. In this section, four such patterns are detected. Three patterns are at the elemental level and one is at the structural level. Patterns in the elemental level are lexical patterns and the one in the structural level is structural. Since entity naming in ontologies plays a vital role in the matching process, names of entities such as classes and properties in OWL ontologies are analyzed first. We realized that developers use different styles for naming compound words, thus matchers have difficulties in finding the relations between compound nouns. Therefore, in the pre-processing phase, compound words are analyzed more than simple words. In this phase, a uniform model is set for assimilating the style naming of compound words. The result improved the speed of matching process because assimilation is done prior to the actual matching process.

In OWL ontologies, the names of concepts are separated by delimiters. This process is named tokenization [21]. During our survey, we realized that there are two kinds of tokenization: underscore (e.g.

General_Chair), and the change of a lowercase letter to an uppercase one (e.g. General Chair). In the Conference Track dataset, some ontologies such as cocus, conference, confious, conf Of, crs-dr, iasted, ekaw, open Conf, paperdyne, and sigkdd, used an underscored character for tokenization and some others such as cmt, edas, micro, and my Review, used the change of the first lowercase letter to an uppercase one, for tokenization.

In this paper, the first pattern is determined by considering the delimiters which are applied to compound words. For example, in the edas and ekaw ontologies from the Conference Track, many correspondences are realized like,<”edas#sessionChair”,”ekaw#session_Chair”>, <“edas#Rejected Paper”,”edas#Rejected_Paper”>, <”edas#Social Event”,“edas#Social Event”>, <”edas#Accepted Paper”, “ekaw#Accepted_Paper”> etc., which cannot be discovered by RiMOM matcher. Therefore, in the pre-processing phase we apply refactoring rules as explained in Subsection 5.3 for assimilating the compound words in one uniform style.

In the second pattern, hierarchical naming for compound words is analyzed. Developers use different rules for modeling the knowledge in ontologies, so they apply various forms for compound words in hierarchical naming. As a result, in hierarchical naming, the name of the parent which is used in the naming of the children appears in different orders. For example, in some ontologies of the Conference Track, there are compound names like <Author Paper, Paper Author>, <Rejected Paper, Paper Rejected>, <Accepted Paper, Paper Accepted> that use the name of parent “Paper” in the descendent name in different orders. In this situation, some popular matchers such as ASMOV, RiMOM, and Prior+ cannot properly recognize hierarchical compound names. Therefore, a protocol for all compound naming entities is proposed in which the parent name, which is repeated in the child name, should be at the end of a compound noun. This pattern is applied for naming the class-subclass concepts.

The third pattern focuses on naming data and object properties in OWL ontologies. To apply this pattern, some stop words such as "has", "is", and "was" are added at the beginning of some property names. This is done by considering the peer entities in two ontologies for obtaining uniformity in the properties of input ontologies. By setting resembling names for these different kinds of style naming in data and object properties, better results of the matching process are possible. For example, by applying RiMOM matcher on two ontologies, cmt and ekaw, we realized that some correspondences, such as<cmt#title, confOf#has Title>and <cmt#email, conf Of#has Email>, could not be found in unfixed ontologies. However, after applying the above pattern on input ontologies and then applying RiMOM, these correspondences were easily found.

The fourth pattern is a structural pattern which is proposed by considering how entities appear together in a structure. This pattern is introduced by taking into account the relation of concepts in the taxonomy tree. Since developers have different viewpoints on defining ontologies in similar domains, taxonomic structure of ontologies is often different and confusing. Accordingly, they use diverse hierarchy and granularity for defining similar entities in ontologies. For example, in the Conference and Ekaw ontologies of the Conference Track, two different granularities in concept naming for a similar concept of “author” are identified. In the Conference data set, three levels of granularity for “author” were found including: “contribution_regular-author”, “contribution_co-author”, and “Conference _1th-author”, while in Ekaw data set, only one level of granularity for “author” is reported. To resolve this problem, a protocol was proposed. Based on this protocol, names of entities in ontologies should be defined in similar granularity for similar concepts. In order to set this protocol the idea of conceptual summarization is used [33].

Experimental results in Section 6 show that, in most ontologies, there is a significant number of occurrences of the aforementioned patterns. Therefore, repairing the input ontologies by considering the patterns proposed in this section can improve the result of the alignment. Furthermore, the mentioned patterns can be used for creating/updating new versions of ontologies.

c) Refactoring

All cases of modelling errors that were detected via some patterns mentioned above can be repaired by some refactoring operations. The detection of the mentioned patterns is the starting point for refactoring. Generally, refactoring is a process for performing some changes in the internal structure of software in order to make it easier to understand and make modifications without changing its discernible behaviour. In this literature, refactoring process is used in the ontology matching area. Thus, some revisions in ontologies were made with a semi-automatic process for achieving new versions of ontologies that are more understandable by users and matchers. These versions of ontologies can be used more effectively by different ontology matching tools. We implemented refactoring rules for knowledge which are modelled by different methods. There are three general refactoring operations: adding operation (ADD), restructuring operation (RS), and renaming operation (RN) [20]. Here, we use RN for lexical patterns and RS for structural patterns in order to repair ontologies as explained in Subsection 5.2.

For reforming the three lexical patterns which were detected in the previous step, we apply a renaming operation to unify knowledge concepts. More desirable results in the lexical similarity of the matching tools can be achieved by applying rename operations (RN) for the name of classes in ontology. This is done by considering the name of these classes in a peer ontology which has the same taxonomic structure. Furthermore, by considering the domains and ranges of classes and the relations between concepts in the taxonomy tree, restructuring operations (RS) are applied. So, the structural similarity phase of matchers can produce a better result by transforming one part of ontology into another one.

Experiments were carried out on seven pairs of ontologies from the Conference Track. The reason for choosing these seven pairs among the other ontologies is described in Subsection 6.1. The percentage of RN and RS operations applied on these seven pairs of ontologies is illustrated in Figure 2. As can be seen in Figure 2, in four pairs of ontologies, <cmt-confOf>, <cmt-ekaw>, <conference-ekaw>, and <edas-ekaw>, RN operations are applied more than RS operations, because many different lexical patterns exist in these pairs. Besides, in the other ontology pairs, <cmt-sigkdd>, <conference-confOf>, and <confOf-sigkdd>, RS operations are utilized more than those of RN. This is because these pairs of ontologies have different hierarchical structures.

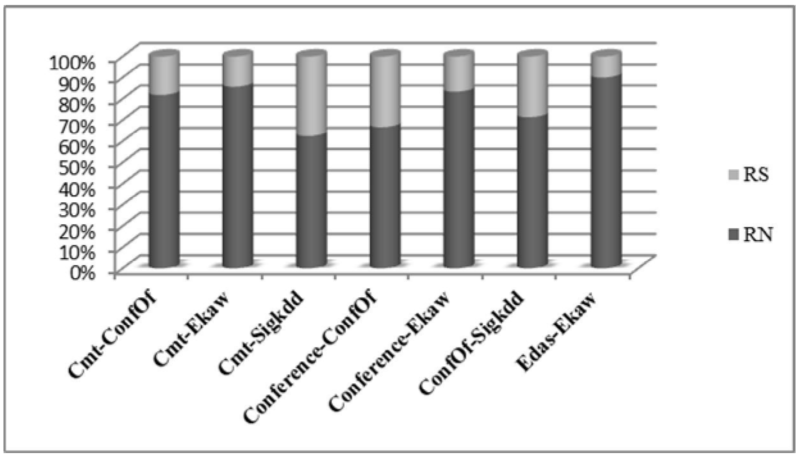


Fig. 2. Comparison of percentages of refactoring operations on different pairs

d) Implementation

Our implementation is based on the employment of Java language with Jena API in Net Beans IDE. Furthermore, protégé and the Ontology Pre-Processor Language (OPPL) were used for manipulating ontologies written in OWL. OPPL is a domain-specific macro language, based on the Manchester OWL Syntax. OPPL instructions can add/remove entities and add/remove axioms to/from entities in OWL ontology. The OPPL Instruction Manager is a Java library that processes OPPL instructions to make changes in OWL ontology. This language is also suitable for defining independent modeling macros (e.g. Ontology Design Patterns) that can be applied across ontologies [34].

e) Illustrative example

Presented in this section are two illustrative examples to clarify the proposed approach by testing the work of two matchers. In the first example, our approach was tested using the ASMOV matcher. Figure 3 illustrates different styles in class naming and various taxonomic structures for defining the same concepts in a part of two ontologies, namely *ConfOf* and *Sigkdd*.

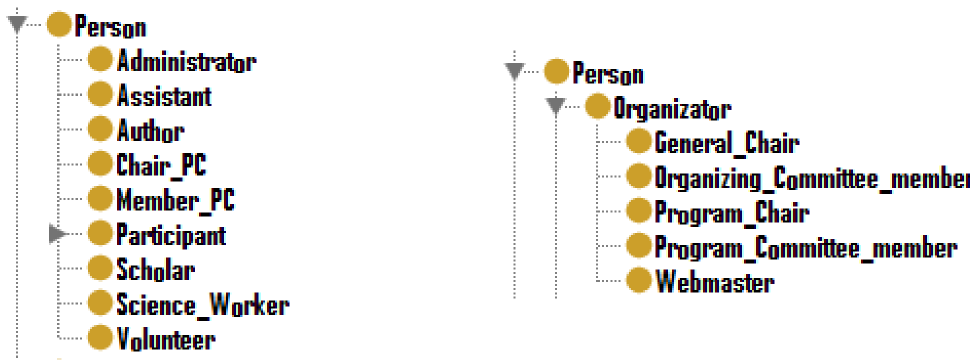


Fig. 3. Different styles in class naming and different taxonomic structures in two ontologies

Figure 4 illustrates correspondences found by ASMOV after applying refactoring operations (RN and RS) on *confOf* and *sigkdd*. The lines in this figure show correspondences existing in reference alignment which could be found by ASMOV after the refactoring operations.

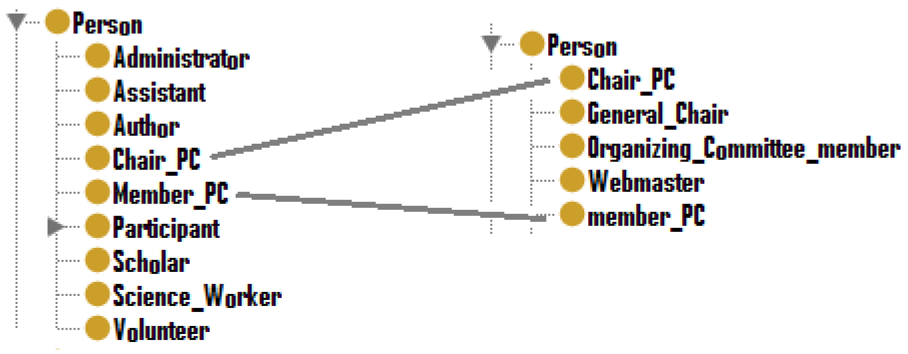


Fig. 4. Correspondences found by ASMOV after refactoring

In the second example our approach was tested using the RiMOM matcher. Figure 5 illustrates different styles in class naming and tokenization in original ontologies. These different styles in concept naming led to many problems in calculating the lexical similarity phase in RiMOM. Therefore, these kinds of difficulties are addressed by applying refactoring operations for various lexical patterns.

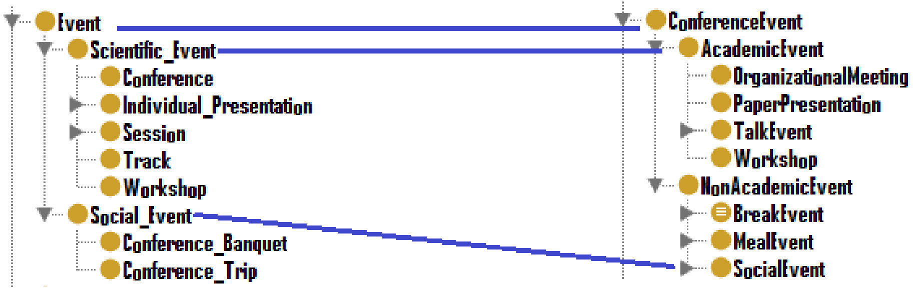


Fig. 5. Different styles in class naming and various tokenization in two ontologies

After repairing these ontologies in the pre-processing phase, RiMOM located these correspondences. Figure 6 shows the alignment detected by RiMOM after utilizing refactoring operations.

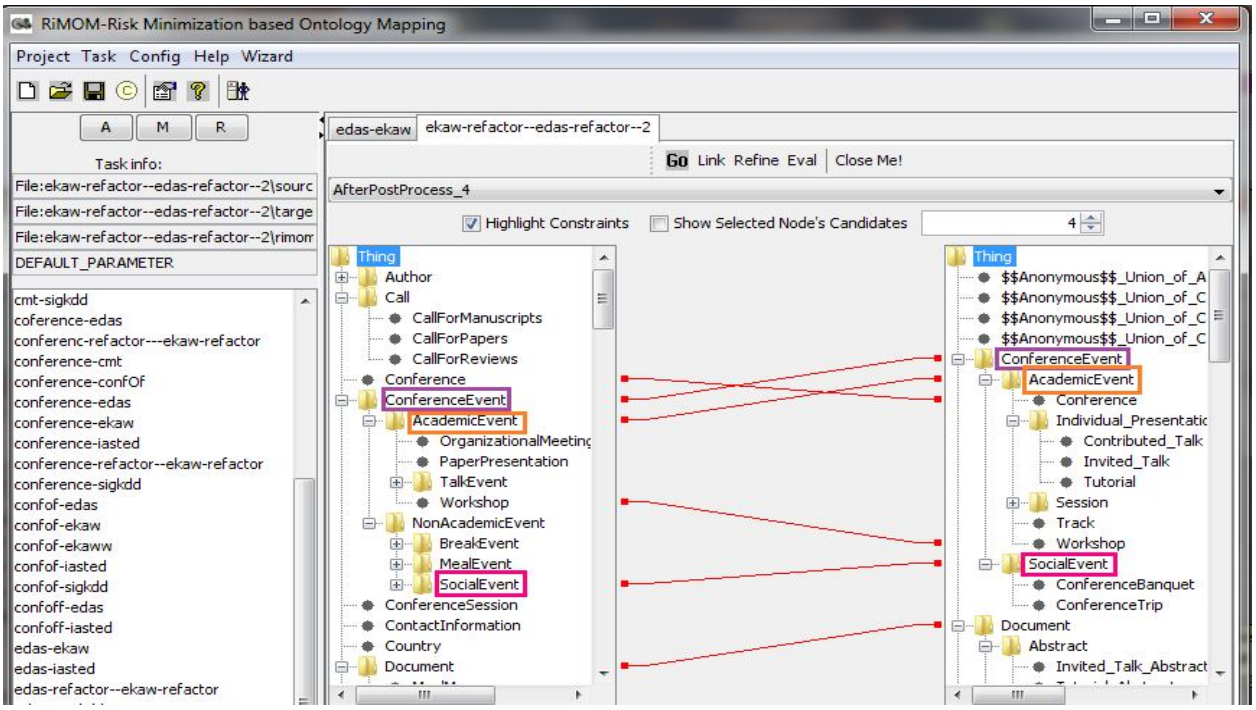


Fig. 6. Correspondences found by RiMOM after refactoring

6. EXPERIMENTAL RESULTS

a) Datasets

As mentioned in Subsection 5a, ontologies used in our experiments were a part of OAEI. The OAEI offers several tracks and subtracts concentrated in different types of matching problems. Our approach was tested on the *Conference Track*. They are described in OWL-DL and published in the RDF/XML format. These data sets are well known to the organizers and have been used in many ontology matching evaluations. Because it is more heterogeneous, the *Conference* dataset can be viewed as a much more challenging test case in contrast to other ontologies of OAEI, such as the *Benchmark* dataset. Our experiment was carried out on six out of sixteen ontologies of the *Conference Track*. These ontologies were*cmt*, *confOf*, *ekaw*, *conference*, *edas*, and *sigkdd*. The reason for selecting these six ontologies among others is that reference mapping (also referred to as the gold standard) is available for all possible combinations of these selected ontologies. To evaluate the accuracy of the matching process, it is necessary to determine both the number of correctly found correspondences and the number of incorrectly found correspondences. This is done by using a reference alignment between two ontologies that has been previously extracted by human experts.

b) Evaluation of the proposed approach

The evaluation of the ontology matching results is an important activity called ontology alignment evaluation. In 2004 this process was established by the Ontology Alignment Evaluation Initiative (OAEI) for the purpose of evaluating world-wide matchers. Two top level matchers of the OAEI 2008 and OAEI 2009, RiMOM and ASMOV, were employed for evaluating our proposed approach. For measuring how accurate an alignment is, three traditional measures borrowed from information retrieval, namely precision, recall, and F-measure were used. These three measures are adapted for ontology alignment evaluation [1].

Precision is defined as the number of correctly found correspondences divided by the total number of found correspondences. Recall is defined as the number of correctly found correspondences divided by the number of reference alignments. A perfect precision score of 1.0 signifies that every correspondence computed by the algorithm was correct (correctness), whereas a perfect recall score of 1.0 means that all correct correspondences were found (completeness).

Precision and recall are defined in (1), (2) [35].

$$precision = \frac{number\ of\ correctly\ found\ correspondences}{number\ of\ all\ found\ correspondences} \tag{1}$$

$$recall = \frac{number\ of\ correctly\ found\ correspondences}{number\ of\ all\ reference\ alignment} \tag{2}$$

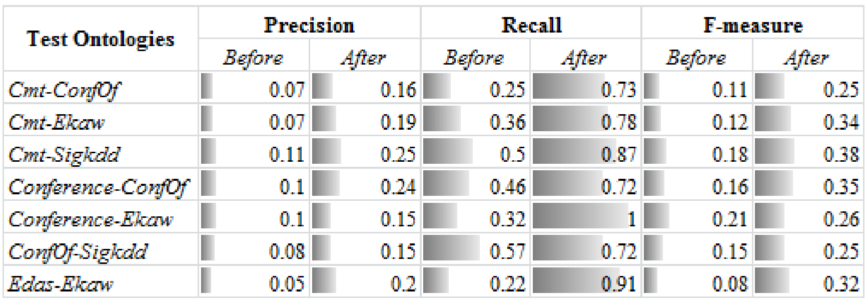
F-measure represents a compromise between precision and recall and it is calculated as (3), where the β parameter represents the influence of recall and precision. F-measure is mainly set to 1, which implies precision and recall are of the same importance.

$$f - measure(\beta) = \frac{(B^2 + 1) \times precision \times recall}{(B^2 \times precision) + recall} \tag{3}$$

$$By\ setting\ \beta = 1, (3)\ leads\ to \tag{4}$$

$$f - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{5}$$

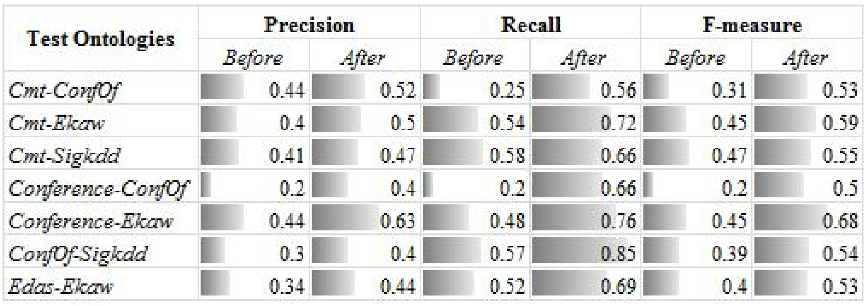
The first evaluation focuses on the RiMOM matcher [36]. Experiments were performed on seven pairs of ontologies from the *conference track*. First, alignments by RiMOM were automatically generated for these pairs of ontologies:*cmt-ekaw, cmt-confOf, cmt-sigkdd, edas-ekaw, conference-confOf, conference-ekaw, and confOf-sigkdd*. Then, the quality of the output alignment was judged by regarding the reference alignment which was available at the beginning of the process, and the three metrics of precision, recall, and F-measure. The first column in Fig. 6 provides the results of RiMOM before the application of the proposed approach. The second column in Fig. 6 presents the results achieved after the implementation of the proposed approach. This result is achieved by running RiMOM on repaired ontologies and generating alignments automatically. The alignments between the original ontology and the refactored one reveal that the proposed approach improves results in comparison with standard evaluation measures.



Test Ontologies	Precision		Recall		F-measure	
	Before	*After*	*Before*	*After*	*Before*	*After*
Cmt-ConfOf	0.07	0.16	0.25	0.73	0.11	0.25
Cmt-Ekaw	0.07	0.19	0.36	0.78	0.12	0.34
Cmt-Sigkdd	0.11	0.25	0.5	0.87	0.18	0.38
Conference-ConfOf	0.1	0.24	0.46	0.72	0.16	0.35
Conference-Ekaw	0.1	0.15	0.32	1	0.21	0.26
ConfOf-Sigkdd	0.08	0.15	0.57	0.72	0.15	0.25
Edas-Ekaw	0.05	0.2	0.22	0.91	0.08	0.32

Fig. 7. The effect of implementation of the proposed approach on the results using RiMOM

For the second evaluation, we concentrated on the ASMOV matcher [37]. Experiments were carried out on the previous seven pairs of ontologies. We automatically generated alignments by ASMOV for those pairs of ontologies before and after implementing the proposed approach. The results obtained before applying the proposed approach and those after are illustrated in Fig. 7. The results show the improvement of the matching results of ASMOV by the implementation of the proposed approach.



Test Ontologies	Precision		Recall		F-measure	
	Before	*After*	*Before*	*After*	*Before*	*After*
Cmt-ConfOf	0.44	0.52	0.25	0.56	0.31	0.53
Cmt-Ekaw	0.4	0.5	0.54	0.72	0.45	0.59
Cmt-Sigkdd	0.41	0.47	0.58	0.66	0.47	0.55
Conference-ConfOf	0.2	0.4	0.2	0.66	0.2	0.5
Conference-Ekaw	0.44	0.63	0.48	0.76	0.45	0.68
ConfOf-Sigkdd	0.3	0.4	0.57	0.85	0.39	0.54
Edas-Ekaw	0.34	0.44	0.52	0.69	0.4	0.53

Fig. 8.The effect of implementation of the proposed approach on the results using ASMOV

7. CONCLUSIONS AND FUTURE WORKS

In this paper, an approach has been presented for overcoming the uncertainty in the challenge of ontology matching. This is to add a pre-processing phase to matchers for improving the results of the matching process. In the pre-processing phase, input ontologies were analyzed in order to find unexpected errors which have been modeled in input ontologies. As a result, some lexical and structural patterns were detected on which refactoring operations like RN and RS operations were applied to repair them. This approach makes ontologies easier to understand by both humans and matchers and avoids some common mistakes in the alignment results of the matching process. The proposed approach evaluated the usage of two high level matchers, RiMOM, and ASMOV. Our experiments were carried out on some ontologies of the *conference track*. Experimental results show that our proposed approach improved the quality of the matching process with respect to standard evaluation measurements, i.e. precision, recall, and F-measure.

For future research, new solutions can be proposed for overcoming this challenge or others which exist in the field of ontology matching. Furthermore, our approach can be tested on other matching tools, especially those participating in the OAEI contest. Moreover, some detectable patterns for discovering errors of ontologies and other refactoring operations for repairing them can be extended.

Acknowledgement: I would like to thank Ondrej Svab Zamazal from the University of Economics, Prague for the helpful suggestion provided during this research.

REFERENCES

1. Euzenat, J. & Shvaiko, P. (2007). *Ontology matching.* Berlin: Springer, pp. 51-54.
2. Shvaiko, P. & Euzenat, J. (2008). Ten challenges for ontology matching. Proceedings of the Ontologies, Databases and Applications of Semantics (ODBASE) International Conference, pp. 1164-1182.
3. Madhavan, J., Bernstein, P. A., Domings, P. & Halevy, A. (2002). Representing and reasoning about mappings between domain models. *Proceedings of the 18th National Conference on Artifical Intelligence (AAAI),* pp. 80-86.
4. Dong, X., Halevy, A. & Yu, C. (2007). Data integration with uncertainty. *Proceedings of the 3rd International conference on very large data bases (VLDB),* pp. 687-698.
5. Gal, A. (2006). Managing uncertainty in schema matching with top-k schema mappings. *Journal on Data Semantics VI, LNCS,* Vol. 6, No., pp. 90-114.
6. Castano, S., Ferrara, A., Lorusso, D., Nath, T. H. & Moller, R. (2008). Mapping validation by probabilistic reasoning. *Proceedings of the 5th European semantic web conference on The semantic web: research and applications (ESWC),* pp. 61-75.
7. Gal, A., Anaby-Tavor, A., Trombetta, A. & Montesi, D. (2005). A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, Vol. 14, pp. 50-67.
8. Jean-Marya, Y. R., Shironoshitaa, E. P. & Kabuka, M. R. (2009). Ontology matching with semantic verification. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 7, pp. 235-251.
9. Hu, W. & Qu, Y. (2008). Falcon-AO: A practical ontology matching system. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 6, pp. 237-239.
10. Maoa, M., Peng, Y. & Spring, M. (2010). An adaptive ontology mapping approach with neural network based constraint satisfaction. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web,* Vol. 8, pp. 14-25.
11. Li, J., Tang, J., Li, Y. & Luo, Q. (2009). RiMOM: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, pp. 1218-1232.
12. Giunchiglia, F., Shvaiko, P. & Yatskevich, M. (2004). S-Match: an algorithm and implementation of semantic matching. *Proceedings of the first European Semantic Web Symposium (ESWS)*, Vol. pp. 61-75.
13. Seddiqui, M. H. & Aono, M. (2009). Anchor-flood: results for OAEI 2009. *Proceedings of the ISWC 2009 Workshop on ontology matching*, pp. 127-134.
14. Noy, N. F. & Musen, M. A. (2001). Anchor-PROMPT: using non-local context for semantic matching. In Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 63-70.
15. Noy, N. F. & Musen, M. A. (2000). PROMPT: algorithm and tool for automated ontology merging and alignment. *Proceedings of the 17th International Conference on Artificial Intelligence (AAAI),* pp. 450-455.
16. Shvaiko, P. & Euzenat, J. (2005). A survey of schema-based matching approaches. *Journal on Data Semantics IV.* Vol. 6, pp. 146-171.
17. Mascardi, V., Locoro, A. & Rosso, P. (2009). Automatic ontology matching via upper ontologies: A systematic evaluation. *IEEE Transactions on Knowledge and Data Engineering*. Vol. 22, pp. 609-623.
18. Ostrowski, D. A. (2008). Ontology refactoring. Proceedings of the IEEE International Conference on Semantic Computing; pp. 476-479.
19. Doan, A., Madhavan, J., Domingos, P. & Halevy, A. (2004). *Ontology matching: a machine learning approach.* Handbook on Ontologies, Berlin, Springer-Verlag, pp. 385-404.
20. Sváb-Zamazal, O., Svátek, V., Meilicke, C. & Stuckenschmidt, H. (2008). Testing the impact of pattern-based ontology refactoring on ontology matching results. *Proceedings of the ISWC 2008 Workshop on Ontology Matchin*, pp. 229-233.

21. Svab-Zamazal, O. & Svatek, V. (2008). Analysing ontological structures through name pattern tracking. *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management, Springer LNCS*, pp. 213-228.
22. Svab-Zamazal, O. & Svatek, V. (2009). Towards ontology matching via pattern-based detection of semantic structures in OWL ontologies. *Proceedings of the Znalosti Czecho-Slovak Knowledge Technology conference*.
23. Svab-Zamazal, O. & Svatek, V. (2007). Tracking name patterns in OWL ontologies. Proceedings of the 5th International EON Workshop Located at the 6th International Semantic Web Conference (ISWC), pp. 61-70.
24. Clark, P., Thompson, J. & Porter, B. W. (2000). Knowledge patterns. Proceedings of the international Conference on Principles of Knowledge Representation and Reasoning (KR2000), pp. 591-600.
25. Baumeister, J. & Seipel, D. (2006). Verification and refactoring of ontologies with rules. Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW), pp. 82-95.
26. Reutelshoefer, J., Baumeister, J. & Puppe, F. (2009). A data structure for the refactoring of multimodal knowledge. *Proceedings of the 5th Workshop on Knowledge Engineering and Software Engineering*, pp. 33-45.
27. Caralt, J. C. (2004). Ontology-driven information systems: pruning and refactoring of ontologies. Proceedings of the Doctoral Syposium of 7 th International Conference on the Unified Modeling Language.
28. Baumeister, J., Reutelshoefer, J., Haupt, F. & Nadrowski, K. (2008). Capture and refactoring in knowledge wikis. *Proceedings of the 2nd Workshop on Scientific Communities of Practice*.
29. Kapsammer, E., Kargl, H., Kramler, G., Reiter, T., Retschitzegger, W. & Wimmer, M. (2006). Lifting metamodels to ontologies - a step to the semantic integration of modeling languages. *Proceedings of the ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems*, pp. 528-542.
30. Jean-Mary, Y. R., Shironoshita, E. P. & Kabuka, M. R. (2009). ASMOV: Results for OAEI 2009. Proceedings of the 4th International Workshop on Ontology Matching collocated with the 8th International Semantic Web Conference, pp. 152-160.
31. Zhang, X., Zhong, Q., J., Li, J. T., Xie, G. & Li, H. (2008). RiMOM. Results for OAEI 2008. Proceedings of the 7th International Workshop on Ontology Matching collocated with the 7th International Semantic Web Conference (ISWC), pp. 182-190.
32. Svab, O., Svatek, V., Berka, P., Rak, D. & Tomasek, P. (2005). OntoFarm: towards an experimental collection of parallel ontologies. *In Poster Proceedings of the International Semantic Web Conference*, pp. 1-3.
33. Gavagsaz, E., Naghibzadeh, M. & Jalali, M. (2011). Conceptual summarization using ontologies and nearest neighborhood clustering. *Proceedings of the International Conference on Semantic Technology and Information Retrieva*l, pp. 1-7.
34. Egaña, M., Stevens, R. & Antezana, E. (2008). Transforming the axiomisation of ontologies: the ontology pre-processor language. *Proceedings of the 4th International Workshop, OWL: Experiences and Directions (OWLED).*
35. Houshmand, M., Naghibzadeh, M. & Araban, S. (Reliability-based similarity aggregation in ontology matching. *Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2010)*, pp. 744-749.
36. Behkamal, B., Naghibzadeh, M. & Moghadam, R. A. (2010). The impact of refactoring based on naming pattern detection on RiMOM result. *Proceedings of the 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010)*, pp. 374-378.
37. Behkamal, B., Naghibzadeh, M. & Moghadam, R. A. (2010). Using pattern detection techniques and refactoring to improve the performance of ASMOV. *Proceedings of the 5th International Symposium on Telecommunications (IST'2010)*, pp. 979-984.