A Framework for Improving the Performance of Ontology Matching Techniques in Semantic Web

Kamel Hussein Shafa'amri
Princess Sumaya University for Technology
Amman, Jordan

Jalal Omer Atoum
Princess Sumaya University for Technology
Amman, Jordan

***Abstract*—Ontology matching is the process of finding correspondences between semantically related entities of different ontologies. We need to apply this process to solve the heterogeneity problems between different ontologies. Some ontologies may contain thousands of entities which make the ontology matching process very complex in terms of space and time requirements. This paper presents a framework that reduces the search space by removing entities (classes, properties) that have less probability of being matched. In order to achieve this goal we have introduced a matching strategy that uses multi matching techniques specifically; string, structure, and linguistic matching techniques. The results obtained from this framework have indicated a good quality matching outcomes in a low time requirement and a low search space in comparisons with other matching frameworks. It saves from the search space from (43% - 53%), and saves on the time requirement from (38% - 45%).**

Keywords- Ontology matching; RDF statements;Semantic web; Similarity Aggregation.

I. INTRODUCTION

In the current World Wide Web (WWW) computers and machines have no idea about the semantic of the information that are transferred through the web; the transferred information are not machine understandable. The role of computers is only to present the transferred information using web browsers [19].

However, the next generation of the WWW is called a Semantic Web. The role of the computers in the Semantic Web is not only to present the information, but for the computers to read and process the information in the WebPages, and extract knowledge from this information.

The computer can understand the information in the Semantic Web by using a data structure called Ontology. Ontology provides a knowledge representation in a particular domain; it defines concepts (classes and properties) in a given domain, and shows the relationships between the defined concepts [1], [19].

Different people may develop different ontologies that describe a particular domain; this causes heterogeneity problems between ontologies that describe the same domain. In general different ontologies for a specific domain may use different data formats, modeling languages and structures to represent certain knowledge. The heterogeneity problem leads to inability to get accurate search results in semantic web. For example, some ontologies define a car as a "car" and another ontologies define a car as an "automobile", so if we write a keyword "car" in a semantic web search engine then the result of the search engine will be a list of all WebPages that are based on ontologies that define car as a "car", and this list will not contain the WebPages that are based on ontologies that define a car as "automobile".

In general, ontology provides knowledge in a certain domain to help the machines to make intelligent decisions. Ontology consists of four components: concepts, object properties, data properties and Individuals. In ontologies, we can define RDF statements. An RDF statement consists of three elements [1], [19]: Resource (Subject or Domain), Object Property (Predicate or Property), Value (Object or Range).

To solve the heterogeneity problem between ontologies, we must apply a process called ontology matching process. Ontology matching is the process of finding correspondences between semantically related entities of different ontologies. These correspondences stand for different relations such as equivalence, more general, or disjointness, between ontologies entities.

II. BACKGROUND

There are several types of matching techniques that are used to find the correspondences entities between ontologies. These techniques are string-based techniques, language-based techniques and structural techniques [9].

Several ontology matching systems were developed to find matched entities between different ontologies, such as Naive Ontology Mapping (NOM) [7], PROMPT [15], Anchor-PROMPT [14] and GLUE [5]. These previous systems take two ontologies as inputs and use different ontology matching techniques to test all entities of the first ontology with all entities of the second ontology in order to find the matched entities between the input ontologies [6]. So the search space and time requirements of these previous systems are very large.

To reduce the search space and time requirement of the ontology matching process, we present in this paper an ontology matching framework (system). This framework uses a multi matching techniques specifically; string, structure, and linguistic matching techniques, and depends on some important features of ontologies; such as RDF statements and class hierarchies.

III. PROPOSED ONTOLOGY MATCHING FRAMEWORK

A. System Overview

As shown in Fig.1, the proposed matching system (PMS) takes two ontologies as input, and determines the matched entities (e.g. classes, object properties, data properties) between these two ontologies. PMS compares entities of the same type.

Specifically, it compares classes of ontology1 with classes of ontology2, object properties of ontology1 with object properties of ontology2, and data properties of ontology1 with data properties of ontology2. PMS uses three types of matching techniques, string and linguistic techniques in a combined framework called "structure matching".

In order to reduce the search space of the matching process, PMS is dependent on RDF statements and class hierarchies which are the base components in ontologies. This PMS matches RDF statements of ontology1 with RDF statements of ontology2, and matches class hierarchies of ontology1 with class hierarchies of ontology2.

The output of PMS is a set of matched entities with their similarity values (confidence measures) each between 0 and 1. All the entities that have similarity values greater than a pre-defined threshold are considered to be correct matched entities. And the output, also, includes the relationship types between the matched entities (equivalence and subsumption). The matching relationship is dependent on similarity values. Our PMS focuses on one-to-one (1:1) and many-to-many (m: m) match relationships, since they are the most commonly used.

B. Structure of Ontology Matching Framework

The proposed system PMS has three stages: pre-processing, matching process and post-processing.

1) Pre-processing

In this stage, PMS extracts the features of the input ontologies. As shown in Fig.2, the system will read all RDF statements and put them in a list that is called RDF statements list. Each element in the RDF statements list will be one RDF statement (subject, object properties, object).

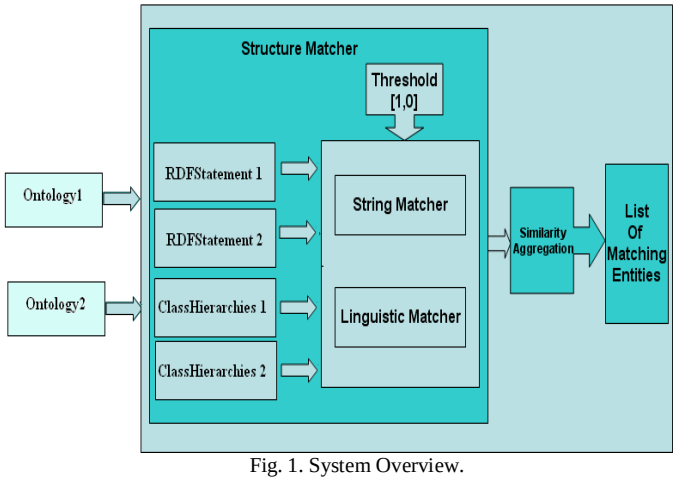

Fig. 1. System Overview.

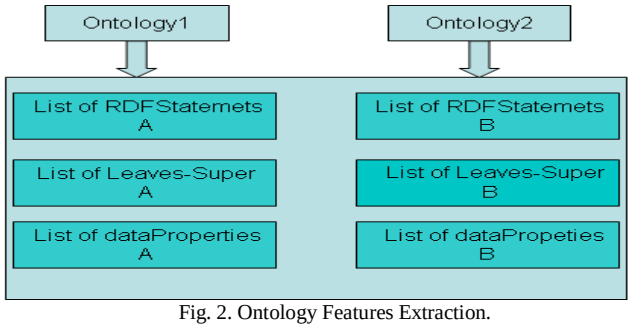

Fig. 2. Ontology Features Extraction.

Then, the system reads all leaves classes and their super-classes and put them in a list that is called leaves-super list. Each element in the leaves-super list will be an object that contains a leaf class and its super classes. Finally, the system reads all ontology classes and their data properties and put them in data properties list. Each element in data properties list will be an object that contains class and its data properties.

2) Matching Process

Structure matching consists of two stages; the first one involves matching of RDF statements and the second one involves matching of class hierarchies. Matching class hierarchies also consists of two sub stages, matching of leaves-super classes and matching of class-data properties.

a) Similarity Aggregation

In order to combine the similarity values of string matcher and linguistic matcher, we use the following similarity aggregation function [4]:

$$Sim_{agg}(e1,e2) = W_S \times Sim_S(e1,e2) + W_L \times Sim_L(e1,e2)$$

Where e1 is an entity of ontology1 and e2 is an entity of ontology2, Simagg( ) is similarity combination of string similarity Sims ( ) and linguistic similarity SimL ( ), Ws is a string weight and WL is a linguistic weight. Ws, WL∈ [0, 1] and Ws + WL = 1. We used Ws = 0.3 and WL = 0.7. This means that the linguistic matcher is more important than the string matcher.

b) Matching RDF Statements

As mentioned earlier, an RDF statement has three components (Subject, Object property, Object). PMS will match every RDF statement in RDF-statements-list-A of ontology1 with every RDF statement in RDF-statements-list-B of ontology2 as illustrated in Fig.3.

PMS computes the similarity aggregation value of the subject of an RDFstatement-A with the subject of an RDFStatement-B, if their similarity aggregation value is greater than the threshold value (matched subject) then it will compute the similarity aggregation value of the object property of an RDFStatement-A with the object property of an RDFStatement-B. If their similarity aggregation value is greater than the threshold value (matched object property) then it will compute the similarity aggregation value of the object of an RDFStatement-A with the object of an RDFStatement-B.

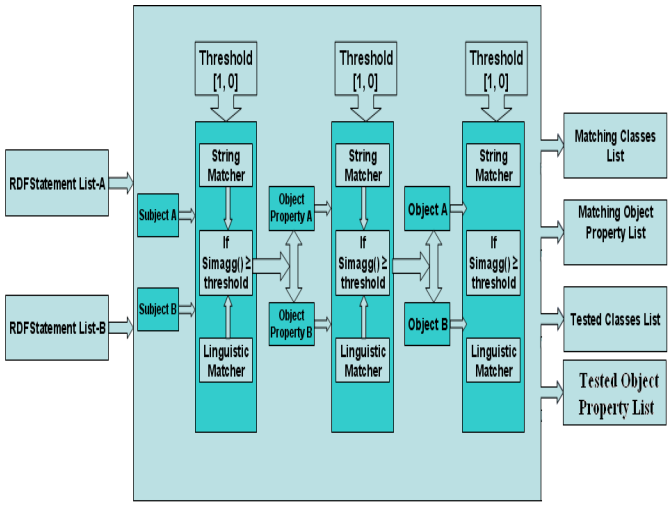

Fig. 3. RDF Statements Matching Process.

Also, PMS adds the tested subjects and objects to the tested classes list, and adds the tested object properties to the tested object properties list. Then, PMS checks if the object, subject classes, and object properties were tested before, by searching the tested corresponding list. These operations are done to prevent computing similarity aggregation values for classes (subjects and objects) and object properties more than once.

Finally, the system will add matched subjects and objects with their similarity aggregation values to the matched classes list, and will add matched object properties with their similarity aggregation values to the matched object properties list.

Matching RDF statements using this scenario will reduce the search space of the matching process for the following reasons:

- PMS ignores the object properties and the objects of RDF statements if the subjects of RDF statements are not matched.
- PMS ignores the object of RDF statements if the subjects or the objects properties of RDF statements are not matched.

The outputs of the matching RDF statement process are the following four lists:

1) Matched classes list.
2) Matched object properties list.
3) Tested classes list.
4) Tested Object Properties List.

c) Matching Class Hierarchies

In PMS, there are two types of class hierarchies, the first one leaves-superClasses and the second one class-data property. Each type has its own matching process.

- *Matching leaves-superClasses*

In this stage, PMS reads four lists; two lists are outputted from the previous process (Matching RDF Statements), which are the matched classes list and the tested classes list. And the other two lists are the leaves-superList-A of ontology1 and the leaves-superList-B of ontology2. Then, PMS will apply the matching process as illustrated in Fig.4.

PMS computes the similarity aggregation value of the leaf in leaves-superList-A with the leaf of leaves-superList-B, if their similarity aggregation value is greater than the threshold value (matched leaves) then it will compute the similarity aggregation value for all super classes of the matched leafs.

Also, PMS adds the tested leaves classes and super-classes to tested classes list. Then, PMS checks if the leaves and super-classes were tested before, by searching the tested corresponding list. These operations are done to prevent computing similarity aggregation values for classes more than once.

Matching leaves-super classes using this scenario will reduce the search space of the matching process because PMS ignores the super-classes if the leaves are not matched.

The outputs of the matching leaves-superClasses process are the following two lists:

1) Matched classes list.
2) Tested classes list.

- *Matching Class-data Property*

In this stage, as illustrated in Fig.5, PMS reads three lists: matched classes list, dataProperties list-A of ontology1 and dataProperties list-B of ontology2. Data Properties list contains objects. Each object presents a class and its data properties. PMS check every pair of matched classes in the matched classes list, if they have data properties.

Also, PMS adds the tested data properties to tested data properties list. Then, PMS checks if the data properties were tested before, by searching the tested corresponding list. These operations are done to prevent computing similarity aggregation values for data properties more than once.

Again, matching class-data property using this scenario will reduce the search space of the matching process because the system will ignore the data properties of non-matching classes, and the system will try to match the data properties of matched classes only.

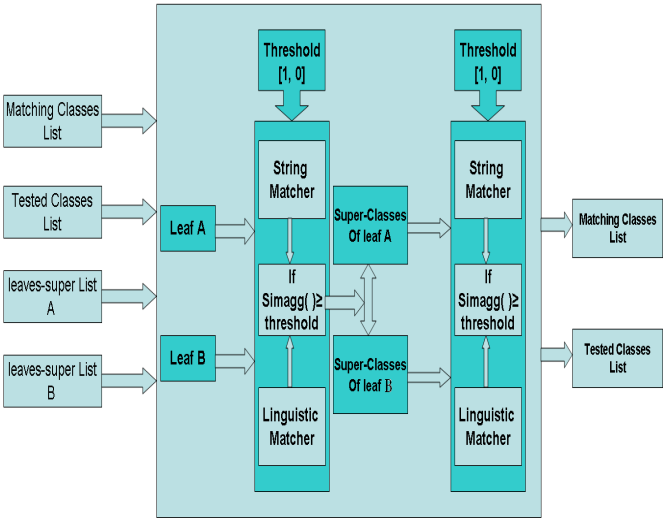

Fig. 4. Leaves-SuperClasses Matching Process.

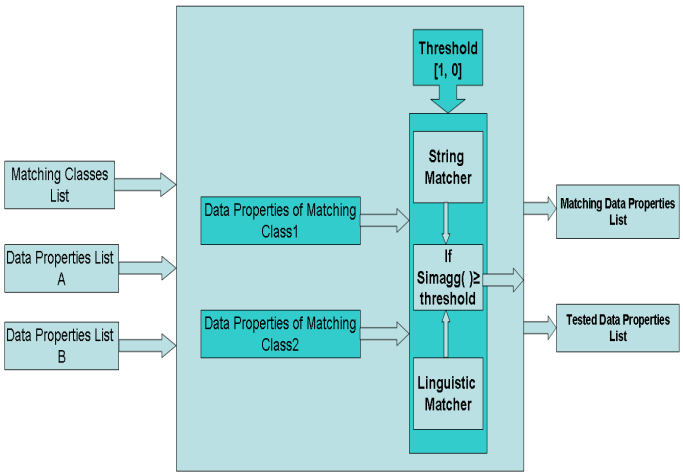

Fig. 5. Data Properties Matching Process.

The outputs of the matching Class-data Property process are the following two lists:

1) Matched data properties list.
2) Tested data properties list.

d) Final Outputs of the Matching Process

The final outputs of the PMS will be three lists as follows:

- Matched classes list.
- Matched object properties list.
- Matched data properties list.

3) Post-Processing

In this stage, the PMS assigns matching relationship R to the matched entities (Classes, Object properties, Data properties) according to their similarity value [9].

- If Simagg (e1, e2) = 1 then R is the equivalence relation.
- If Simagg (e1, e2) ≥ threshold then R is a subsumption relation.

Where Simagg (e1, e2) is the similarity aggregation value between matched entities e1 and e2.

4) String Matcher Implementation

For string matchers, the PMS uses Levenshtein distance similarity measure [17] and soundex similarity measure [17], [22] in combined manner as follows:

$$Sim_S(e1,e2) = \frac{Levenshtein\ distance(e1,e2) + Soundex(e1,e2)}{2}$$

5) Linguistic Matcher Implementation

For linguistic matchers, the PMS uses Wordnet similarity measures of Wu & Palmer similarity measure [18] and path similarity measure [20] in combined manner as follows:

$$Sim_L(e1,e2) = \frac{Wu\ \&\ Palmer(e1,e2) + Path\ (e1,e2)}{2}$$

IV. Proposed Framework Evaluation

There are two types of evaluations that are used to evaluate the PMS. The first evaluation is done by counting the number of tested entities that were tested and by computing the time requirement that are needed by the system to find the matched entities. The second type of evaluation is based on compliance measures to evaluate the quality of the matched results.

A. Compliance Measures

Following the work in [8], Compliance Measures are used to evaluate the degree of compliance of the results of matching algorithms. Compliance measures consist of three measures Precision, Recall and F-measure. These measures are used to evaluate the quality of the matching process and its results. Precision and Recall are based on the comparison of an expected result provided by a reference alignment and the effective result of the evaluated system. Finally, F-measure combines the measures of Precision and Recall as single efficiency measure.

B. Traditional Matching System

For the purpose of evaluating our PMS we have developed a matching system called (Traditional Matching System) TMS that is based on the work of some existing ontology matching systems such as NOM [7], PROMPT [15], Anchor-PROMPT [14] and GLUE [5]. The TMS matches all classes of the first ontology with all classes of the second ontology, and matches all object properties of the first ontology with all object properties of the second ontology, and matches all data properties of the first ontology with all data properties of the second ontology. The goal of developing the TMS is to compare it with the PMS.

C. Conference Dataset

The conference dataset, shown in Table 1, has been proposed in OAEI 2010 and it includes seven ontologies that are dealing with conference organization. These ontologies have been developed within OntoFarm project [21], and are quite suitable for ontology matching task because of their heterogeneous character. Every ontology in this dataset has a number of classes, a number of object properties and a number of data properties. The matching process will be done on each pairs of these ontologies.

TABLE 1. CONFERENCE DATASET [21].

Ontology Name	# of classes	# of object properties	# of data properties
Cmt	29	49	10
Conference	59	46	18
ConfOf	38	13	23
Edas	103	30	20
Ekaw	73	33	0
iasted	140	38	3
sigkdd	49	17	11

D.

E. TMS vs. PMS

The comparison of the PMS and the TMS **was** done on the same computer system ((Intel (R) Core (TM) 2 Duo CPU, 2.4GHz, 3 GB RAM) and Windows 7)). We had applied the matching process using the PMS and the TMS **between** each pair of ontologies of the conference dataset at different threshold values (0.5, 0.7, 0.85, and 1).

1) Search Space and Time Requirement Evaluation

Figures 6 and 7 present the average number of tested entities and the average time requirement at different threshold values (0.5, 0.7, 0.85, and 1) that are needed by both systems TMS and PMS to match all the pairs of Conference dataset ontologies.

We can notice from these two Figures, that the number of tested entities and the time requirement that were needed to find the matched entities in the TMS are larger than the number of tested entities and time requirement that were needed by the PMS, this is due to the fact that the TMS tests more entities than the PMS.

Furthermore, we can notice that the number of tested entities and the time requirement for the TMS remain the same regardless of the threshold values. Whereas, in the PMS they are inversely dependent on the threshold value.

2) Compliance Measures Evaluation

Figures 8, 9 and 10, present the average compliance measures results **(**Precision, Recall and F-measure) of all the matched pairs of Conference dataset ontologies, at different threshold values (0.5, 0.7, 0.85, and 1) for both TMS and PMS.

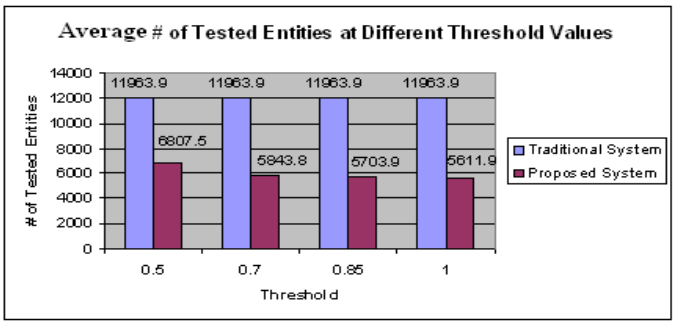

Fig. 6. Average Number of Tested Entities.

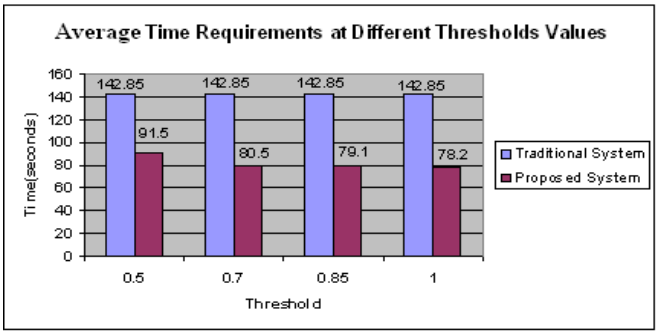

Fig. 7. Average Time Requirement.

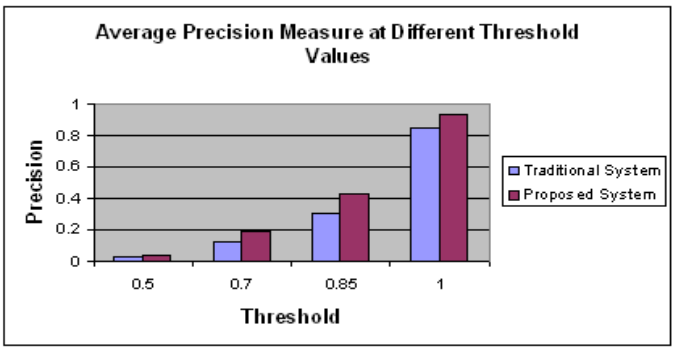

Fig. 8. Average Precision Measure.

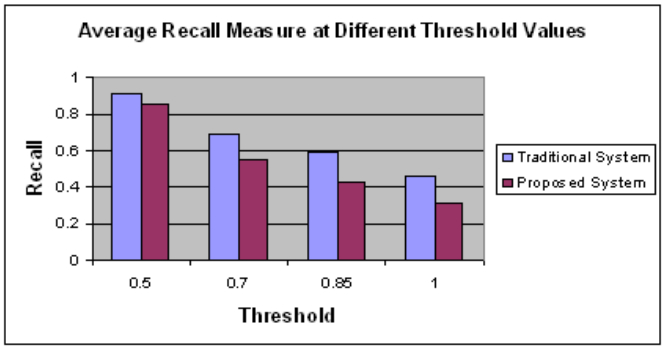

Fig. 9. Average Recall Measure.

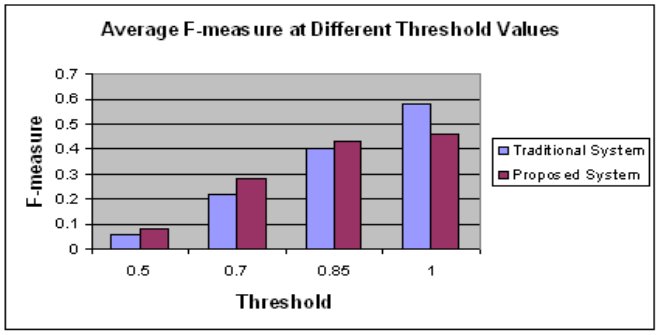

Fig. 10. Average F-measure.

We can notice from these Figures the following:

- At all thresholds values the Precision value of the PMS is better than the Precision value of the TMS. Hence, the PMS returns more accurate matching results than the TMS.
- At all thresholds values the Recall value of the TMS is better than the Recall value of the PMS. This means that the TMS returns more matched entities that are existed in reference alignment R than the PMS;
- At Threshold values (0.5, 0.7, and 0.85) the F-measure value of the PMS is better than the F-measure of the TMS. But at threshold value 1 the F-measure value of the TMS is better than the F-measure of the PMS; this is due to the fact that F-measure is dependent on Recall and Precision values.

F. Comparison with other Existing Matching Systems

We had made a comparison between the PMS and other matching systems that participated in OAEI 2010 in terms of Precision, Recall and F-measure. These systems are AgrMaker [11], AROMA [3], ASMOV [12], CODI [13], Ef2Match [2], Falcon [10] and GeRMeSMB [16]. This comparison is done at threshold values of 0.5 and 0.7. Figures 11 and 12 show the results of this comparison.

We can notice from these Figures the following:

- The PMS at threshold 0.5 and at threshold 0.7 has the lowest Precision value, because the PMS returns the largest number of matched entities but a few of them are existed in the reference alignment.
- The PMS has the highest Recall value at threshold 0.5. This means that the PMS returns the largest number of matched entities that are existed in reference alignment than the other matching systems.
- The PMS has a good Recall value between the Recall values of the other systems at threshold 0.7.
- The PMS has a low F-measure value at threshold 0.5 and at threshold 0.7.

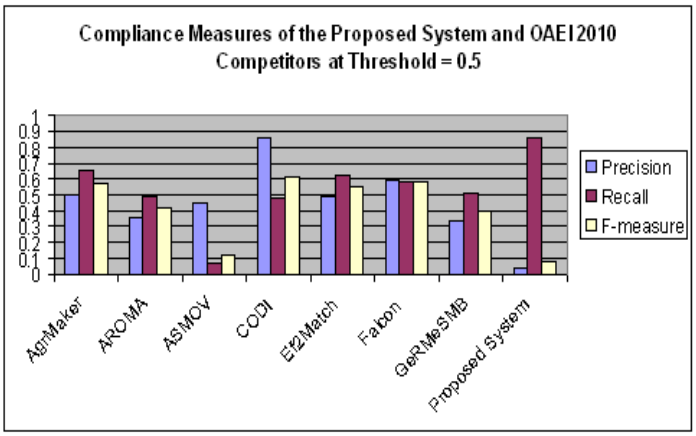

Fig. 11. Comparison with Other Matching Systems at Threshold = 0.5.

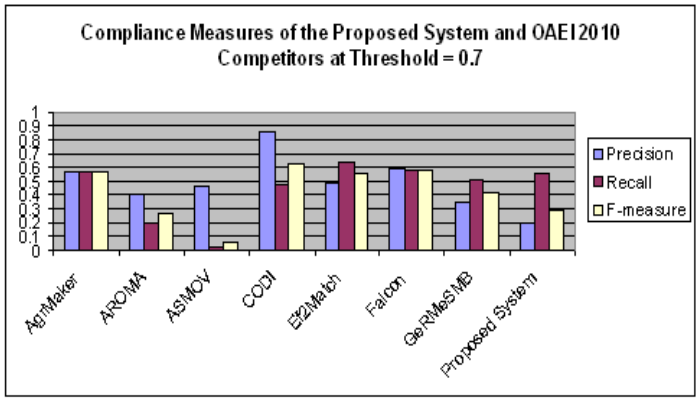

Fig. 12. Comparison with Other Matching Systems at Threshold = 0.7.

V. Conclusion

The main goal of this paper was to reduce the complexity (search space and time requirement) of the ontology matching process. This paper have introduced an ontology matching framework that reduces the search space and the time requirement of the matching process by removing entities (classes, properties) that have less probability of being matched. The proposed ontology matching framework had used a multi matching techniques in order to find the correspondences entities between ontologies.

The proposed matching framework saves (43% - 53%) from the number of tested entities (search space). Furthermore, the proposed matching framework saves on time requirement of the matching process from (38% - 45%) in comparisons with other matching frameworks.

The drawback of the proposed matching framework is that it can't find all possible alignments entities between ontologies, due to the fact that the PMS doesn't test all entities of the matching ontologies. Hence, the PMS is recommended to be used in matching large ontologies, since it will produce a huge number of matching entities that could be enough for web searching using semantic web.

References

[1] G. Antoniou and F. Harmelen , "A semantic web prime", (2nd ed). London: Massachusetts Institute of Technology, 2008.

[2] W. Chua and J. Kim, "Eff2Match results for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.

[3] J. David F. Guillet and H. Briand , "Matching directories and OWL ontologies with AROMA", proceedings of the 15th ACM international conference on Information and knowledge management, ACM, New York, USA, pp. 831-831, 2006.

[4] H. Do and E. Rahm, "COMA - A system for flexible combination of schema matching approaches", proceedings of the very larged data bases conference (VLDB), Hong Kong, China, pp610-621, 2002.

[5] A.Doan, J. Madhavan, P. Domingos and A.Halevy, "Ontology matching: a machine learning approach" In: S. Staab, and R. Studer (Ed), handbook on ontologies in information system, (pp.385–404), Berlin: Springer-Verlag, 2003.

[6] M. Ehrig and S.Staab, "QOM: quick ontology mapping", Proceedings of the international semantic web conference (ISWC), Hiroshima, Japan, pp. 683–697, 2004.

[7] M. Ehrig and Y. Sure. "Ontology mapping - an integrated approach", proceedings of the 1st european semantic web symposium (ESWS), vol. 3053, pp.76–91, Heraklion: Springer-Verlag, 2004.

[8] J. Euzenat, "Semantic precision and recall for ontology alignment evaluation", proceedings of international joint conference on artificial intelligence (IJCAI), Hyderabad, India, pp. 248-253, 2007.

[9] J.Euzenat and P. Shvaiko, "Ontology matching", (1st ed.). Berlin: Springer-Verlag, 2007.

[10] W. Hu, J. Chen, G. Cheng and Y. Qu , "ObjectCoref & Falcon-AO: results for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.

[11] C. Isabel. S. Cosmin, C. Michele, F. Caimi, M. Palmonari, F. Antonelli and K. Ulas, "Using agreementmaker to align ontologies for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.

[12] Y. Jean-Mary P. Shironoshita and M. Kabuka, "ASMOV results for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.

[13] J. Noessner and M. Niepert, "CODI: Combinatorial Optimization for Data Integration – results for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.
[14] N. Noy and M. Musen, "Anchor-PROMPT: using non-local context for semantic matching" ,proceedings of international joint conference on artificial intelligence (IJCAI), Seattle, US, pp. 63–70, 2001.
[15] N. Noy and M. Musen, "The PROMPT suite: interactive tools for ontology merging and mapping". international journal of human-computer studies, vol. 59(6), pp. 983–1024, 2003.
[16] C. Quix, A. Gal, T. Sagi and D. Kensche, "An integrated matching system: GeRoMeSuite and SMB Results for OAEI 2010", proceedings of the 5th international workshop on ontology matching, Shanghai, China, 2010.
[17] M. Taye, "Ontology alignment mechanisms for improving web-based searching", unpublished doctoral dissertation, De Montfort University, United Kingdom, England, 2009.
[18] Z. Wu and M. Palmer, "Verb semantics and lexical selection". proceedings of 32nd annual meeting of the association for computational linguistics (ACL), Las Cruces, US, pp. 133–138, 1994.
[19] L. Yu, "Introduction to the semantic web and semantic web services", (1st ed.). Florida: Taylor & Francis Group, 2007.
[20] Pedersen, Ted. "Wordnet similarity", from, http://search.cpan.org/dist/WordNet Similarity/lib/WordNet/Similarity/path.pm .
[21] Ontology alignment evaluation initiative, http://oaei.ontologymatching.org/ .
[22] "Soundex", from http://en.wikipedia.org/wiki/Soundex.