A Comparative Analysis of Ontology and Schema Matching Systems

K. Saruladha
Computer Science Department, Pondicherry Engineering College, Puducherry, India

Dr. G. Aghila
Computer Science Department, Pondicherry University, Puducherry, India

B. Sathiya
Computer Science Department, Pondicherry Engineering College, Puducherry, India

ABSTRACT
In a distributed and open system, such as the semantic web and many other applications like information integration, peer- peer communication, etc., the heterogeneity among the data increases enormously. To solve the heterogeneity issue various matching techniques are proposed and large-scale matching needs especially to be supported for different kinds of ontologies and XML schemas due to their increasing use and size, e.g., in life science applications, e-business and web. In this paper the techniques which are scalable like early pruning, partitioning, parallelization and some renowned scalable matching techniques are discussed. In addition to it, a brief comparison of the discussed matching techniques is also presented.

General Terms
Artificial Intelligence, Semantic Web, Knowledge sharing, Knowledge Representation.

Keywords
Similarity Measure, Schema Matching, Ontology Matching, Ontology Alignment.

1. INTRODUCTION
Matching technique finds semantic correspondences between cartesian product of entity pair of the given two ontologies or schema. In general relational and XML database is called as Schema and the database represented by semantic language like Web Ontology Language (OWL) or Resource Description Language (RDF) is called as Ontology. The correspondences of the matching technique may stand for equivalent, subsumption, or disjoint relation between the ontology entities. The result of matching technique is the set of semantic correspondences called alignments. Fig.1 depicts the general framework for the schema matching techniques which is also applicable for ontology matching techniques. As shown in figure the input of the matching technique is two schemas which are first imported into a format suitable for processing. For a faster computation the schema may be preprocessed, e.g., preparing neighbour list for each entity to fasten the structural similarity calculation, removing redundant information from schema, etc. The similarity value is calculated for all cartesian product entity pairs one from each of two preprocessed schemas using a match workflow consisting of set of matcher (e.g., lexical, structural and semantic matcher). The output of the matcher workflow is the semantic similarity value matrix. The semantic value ranges from 0 to 1, i.e., value 1 indicates equivalent and value 0 means disjoint relation. The matcher workflow can be sequential, parallel, iterative or in some mixed fashion as shown in Fig.2. The similarity value obtained from the match workflow can be aggregated to obtain the final alignment. For large scale schema matching the workflow will slightly vary to incorporate techniques like early pruning and partitioning of schema to reduce the search space, parallelization, etc.

According to Shvaiko P and Euzenat J [1] one of the toughest challenge for matching system is handling large scale schemas or ontology and on the basis of Rahm. E [2] work domain where large scale matching is necessary include e-business [3], web data [4], life science ontologies [5], medicine [6] and web directories [7]. Even though the advancements are made in the current matching systems, they are still unable to achieve good effectiveness (correctness and completeness of the alignment) and good efficiency (time and space efficiency) and the effectiveness is measured by precision and recall while the efficiency is measured by execution time and memory used.

The remainder of this paper is organized as follows: Section 2 defines the terminologies and notations used in this paper. Section 3 discusses various large scale matching technique. Section 4 gives a brief comparison of the large scale matching technique. Finally, Section 5 concludes the paper.

2. PREMILINARIES
Definition 1. (Ontology) An ontology O is a set of RDF triples T. Any RDF triple t ($t \in T$) denotes a statement of the form <subject,predicate,object> Any node in an RDF triple may be a URI with an optional local name (URIref), a literal, or a blank node (having no separate form of identification). A predicate is always an URIref, and a literal cannot be a subject. The node can be a class or a property and for simplicity, classes and properties are uniformly called entities. There exist hierarchical relationships between each entity and its sub-entity. The ontology also consists of set of axioms A. I is a set of instances of concepts and properties.

Definition 2. (Ontology matching) [8] Let O, O' be two ontologies. Matching O with O' finds a set of alignments A= {a1, a2,..,an}. Each ai (i=1, 2,..., n) is a 5-tuple: <id,e1,e2,u,v> where id is an unique identifier, e1 is an entity in O, e2 is an entity in O', u is an equivalence , subsumption or disjointness relationship holding between e1 and e2 and v is a similarity between e1 and e2 in the [0,1] range.

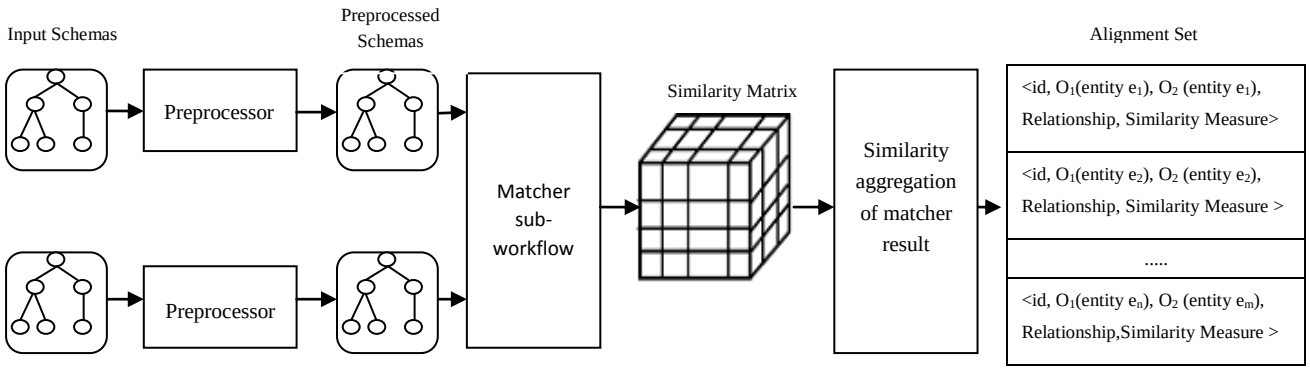

Fig 1: General workflow of matching techniques

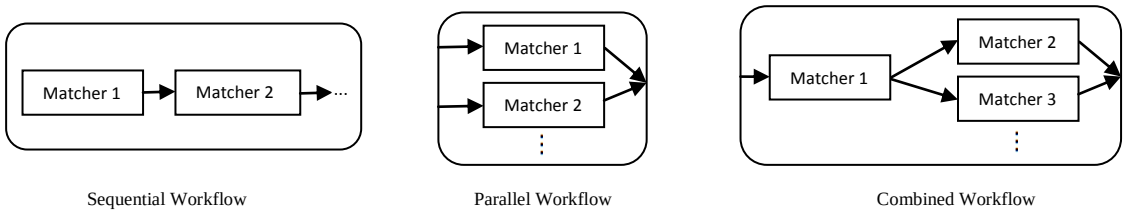

Fig 2: General sub-workflow of matchers

3. APPROACHES USED FOR LARGE SCALE MATCHING TECHNIQUE

In this section, we discuss about various large scale matching technique. The matching techniques are

1) Early pruning strategy based matching technique
   a. Eric peukert et al. schema and ontology matching algorithm
   b. **Q**uick **O**ntology **M**atching algorithm (QOM)
2) Partition based matching technique
   a. Anchor flood ontology matching algorithm
   b. Coma++ schema and ontology matching algorithm
   c. Falcon-AO ontology matching algorithm
   d. Taxomap ontology matching algorithm
3) Parallel matching technique
   a. Gross et al. ontology matching algorithm
4) Other matching tool
   a. AgreementMaker schema and ontology matching tool
   b. ASMOV **(A**utomated **S**emantic **M**atching of **O**ntologies with **V**erification) ontology matching tool
   c. RiMOM ontology matching tool

Fig. 3 depicts the classification of the large scale matching techniques based on techniques adapted for the scalability.

3.1 Early Pruning Matching Technique

The early pruning technique reduces the search space for matching by eliminating certain entities from matching process e.g., one matcher can prune entity pairs whose semantic correspondence value is very low, thus reducing search space for the subsequent matcher. This idea can be accomplished by both sequential and iterative matcher workflow. Eric Peukert et al. [9] System implemented both matcher workflows and QOM [10] implemented iterative workflow.

3.1.1 *Eric Peukert et al. Matching Technique*

Eric Peukert et al. proposed a rewrite match workflow based schema and ontology matching algorithm which is a rule-based optimization technique. The rewrite of workflow is done to improve performance. The match workflow consist of the filter operators to eliminate dissimilar entity pairs (whose similarity value is less than some threshold) from intermediate match results and thus reducing search space for further matchers. The threshold is either statically predetermined or dynamically derived from the similarity threshold used in the previous match workflow to select alignment.

XML schemas, meta models or ontologies format data can be the input. These input formats are converted into a standard internal format. The designer must choose and design the matching steps which is represented as a graph where the vertices represent matcher from the library and edges represent data flow and execution order. The matcher workflow graph designed by the designer can be parallel or serial or iterative or a combined strategy. The designer use the matcher workflow rewrite recommendation system to increase efficiency of the matching process. These matcher workflow rewrite recommendation system is similar to cost-based rewrites in database query optimization e.g, the use of rewrite rules are similar to the use of predicate push-down rules for database query optimization which reduce the number of input tuples for joins and other expensive operators. A simple cost model is used to choose which parts of a matching workflow to rewrite. These rewrites can be accepted or rejected by the user. Once the matching process graph is finalized the graph can be executed to obtain the alignment.

The drawbacks of this method are as follows. The matcher library consists of only few numbers of matchers. Rewrite rules, only focus on efficiency improvements but not on effectiveness improvement.

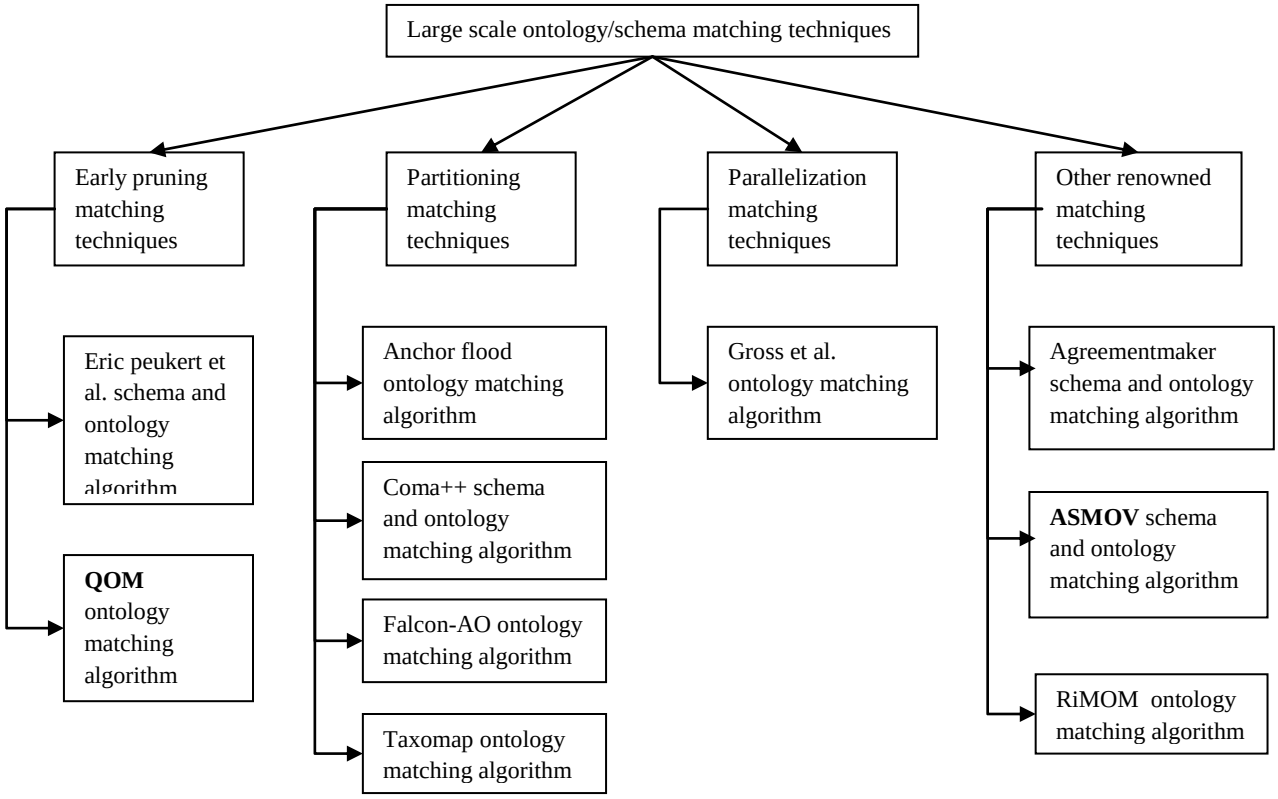

Fig. 3 Classification of large scale Ontology/schema matching techniques

3.1.2 QOM Matching Technique

QOM is the iterative ontology matching algorithm which is a variation of the **N**aive **O**ntology **M**atching (NOM) algorithm dedicated to improve the efficiency of the system. The basic difference is QOM reduces the search space by eliminating less promising matching entity pairs using heuristic and dynamic programming approach with marginally compromising on effectiveness. QOM matcher consisting of the following steps to match the ontologies.

The input to QOM must be in RDF format. The process of determining whether two entities are same or not is based on the entity features. This step constructs the search space of ontology entities by computing the cartesian product of entity pairs using entity features of both ontologies. The entity pairs obtained will be pruned using heuristics strategies to reduce the search space. The selected matching pairs will be processed for similarity computation and their interpretation. The heuristics strategies are based on hierarchy of entities, entity neighbour to the alignments obtained from the previous iteration, labels of entities, random pick or combination of the above strategies. The similarity between the above selected entity pairs can be measured by wide range of similarity functions like Dice coefficient, Levenshtein's edit distance and cosine similarity value. Based on the feature of the entity any of the above similarity function can be chosen. To aggregate the similarity value QOM uses weighted average of similarity values for a given entity pair. The weight of the similarity values is calculated by sigmoid function which emphasizes high individual similarities and deemphasizes low individual similarities. For the next iteration the matching pairs with similarity value less than a threshold and which violates bijectivity (one-to-one and onto constraint) of the matching are discarded. Initial iteration uses lexical knowledge while later iterations use structure knowledge for matching. The number of iteration required irrespective of ontology size based on their experiment is ten. Based on the evaluation QOM is nearly twenty times faster than NOM.

The drawback of QOM is that, effectiveness is marginally lower than NOM since not all cartesian entity pair is evaluated.

3.2 Partition Based Matching Technique

The partition based matching technique divides the ontology to form partitions and execute a partition wise matching between the two ontologies. The partitioning is performed in such a way that each partition of first ontology is matched with only small subset of the partitions (ideally, only with one partition) of the second ontology. The entities of the dissimilar partition pairs can be eliminated from further matching process thus reducing the search space to achieve better efficiency. Space complexity of the matching process is also reduced. Four partition based methods anchor flood [11], COMA++ [12], Falcon-AO [13],Taxomap [14], will be discussed below.

3.2.1 *AnchorFlood Matching Technique*

AnchorFlood is an ontology matching algorithm with a dynamic partition based matching which avoids the priori partitioning of the ontologies. The internal structure, external structure and Jaro-Wingler string distance is used in measuring semantic similarity value between the entity pairs. The initial input to the algorithm is the set of anchors, where an anchor is defined as an

exact string match of concepts, properties or instances pair. Preprocessing of ontologies is carried out to normalize the textual contents of entities. The algorithm starts with an anchor and collects neighboring entity of the chosen anchor across ontologies. The segment pair constructed from the above collected neighbors is processed to produce alignment pairs. The process repeats until "either all the collected entity is explored, or no new aligned pair is found". The anchor will be discarded as misalignment if the constructed segment of the anchor does not produce sufficient alignments.

The drawbacks of this method are as follows. Ignores certain distantly placed aligned pairs since segments are constructed from neighbor of anchors. Semantic similarity between entities is not explored. Only exact string match of concepts, properties and instances are considered as anchors which lead to inefficient alignments.

*3.2.2 COMA++ (**CO**mbination of **MA**tching algorithms)*

COMA++ is a schema matching tool based on parallel composition of matchers. COMA++ is a framework which is equipped with a suite of matcher libraries. The matching result of a matcher could be combined with other matcher results. Further the effectiveness of the different matchers could be evaluated.This method can process XML schema, relational database and OWL ontology. First the schema is processed to identify the relevant components (node, path or fragment) for matching. Then depending on the component chosen a matcher workflow is used to compute component similarities. Finally similarity combination methods are used to find the correspondence between components.

A Fragment is defined as a rooted sub-graph down to the leaf level in the schema graph. The fragment component based matching process is capable of handling large scale schema. The three available fragment type are schema, subschema and shared. User can also select their own fragment through GUI. Based on the fragment type, the schema is fragmented and fragment pairs one from each of the two ontology is matched to find the most similar fragment pair. The search for similar fragments is some kind of light-weight matching, e.g., based on the similarity of the fragment roots. Next similar fragment pair is matched element wise using matcher in the matcher library to find the alignments.

The drawbacks of this method are as follows. COMA++ is developed for XML schemas and hence it is not suitable for complex ontology graph. It use relatively simple heuristic rules to partition the input schemas resulting often in too few or too many partitions and to find similar fragment pair only the root node features are used which will lead to less matching quality.

3.2.3 Falcon-AO Matching Technique

Falcon-AO is an ontology matching tool which aims at finding alignments of the given two OWL/RDF ontologies. The matcher library of Falcon-AO consist of V-Doc [15] and I-Sub [16] which are light-weight linguistic matchers, GMO [17] an iterative structural matcher and **P**artition **B**ased **M**atcher finds the mappings among blocks of large scale ontologies by using divide and conquer technique. The ontology is preprocessed to make it into a RDF graph. The main steps of PBM are partitioning the two input ontology independently, and executing a pair wise partition matching taken one from each of the two ontologies. The similar partition pairs will be further processed for element level matching using VDOC and GMO. The phase of Falcon-AO are as follows.

Partitioning ontologies: For both the input ontology the structural proximities between entities are calculated based on how closely they are related in the hierarchies. The ontology clusters are formed based on structural proximities using modified ROCK [18] clustering algorithm. RDF Blocks are constructed from the clusters by assigning each RDF triple to a cluster in which at least two entities are contained.

Matching blocks: The alignment with high similarities is referred as anchors. A light-weight string comparison technique, I-SUB, is firstly employed to exploit anchors between two full ontologies and then the blocks from the two ontologies are matched based on the distribution of the anchors.

Discovering alignments: V-DOC adopts a linguistic approach to ontology matching. It associates with each ontology entity a bag of words which is built from the entity label, the entity annotations as well as the labels of connected entities. The similarity between entities is based on TFIDF [19]. GMO is an iterative and bipartite structural graph matcher. It starts by considering the RDF representation of the ontologies as a bipartite graph which is represented by its adjacency matrix (A and A'). The distance between the ontologies is represented by a distance matrix (X) and the distance (or update) equations between two entities are simply a linear combination of all entities they are adjacent to, i.e., $X^{t+1} = AX^tA'^T + A^TX^tA'$. This process can be bootstrapped with an initial distance matrix. However, the real process is more complex than described here because it distinguishes between external and internal entities as well as between classes, relations and instances. Similarity combination is a heuristic strategy to tune the thresholds of the above two matchers.

The drawbacks of this method are as follows. The entire ontology needs to be processed to find anchors and thus efficiency of Falcon-AO is reduced. Maximum number of entity in a cluster is determined by the GMO matcher and the clustering algorithm terminates abruptly if it reaches maximum number which will lead to poor clustering.

3.2.4 Taxomap Matching Technique

Taxomap is an ontology matching algorithm consisting of two partitioning algorithm namely Anchor Partition Partition (APP) and Partition Anchor Partition (PAP) which have been designed to take the alignment objective into account in the partitioning process. The most structured ontology is referred as target ontology and the less structured is referred as source ontology. PAP is suitable for structured vs unstructured ontology matching and APP is suitable for structured vs structured ontology matching. The entity pair one from each of two ontology which has identical labels is called as anchors which will be used in both PAP and APP. The alignment is based on lexical and structure (subclass) similarity measure.

Anchor Partition Partition (PAP):

1. Identify the set of anchors across ontologies.
2. Partition both the target ontology and source ontology by

modifying PBM matcher in order to take into account shared anchors.
3. Align blocks that share maximal number of anchors.

Partition Anchor Partition (PAP):

1. Use PBM matcher to partition the target ontology into set of blocks.
2. Identify the set of anchors between two ontologies. This set will be the center of the future block which will be generated from the source ontology.
3. Use PBM matcher to partition the source ontology around the identified centers.
4. Align each block with the corresponding block.

The drawbacks of this method are as follows. The effectiveness of this method depends on the availability of identical labels across ontologies. Only labels and hierarchy structure is used for matching and hence comparatively less recall.

3.3 Parallel Matching Technique

Parallelization is a straight-forward method to increase the efficiency of large-scale matching by executing matcher in parallel on several processors. The two kinds of parallel matching are inter-matcher and intra-matcher parallelization. Inter-matcher parallelization deals with parallel execution of independently executable matchers while intra-matcher parallelization deals with internal decomposition of individual matchers or matcher parts into several match tasks that can be executed in parallel. Gross et al. [20] matching system implements both inter-matcher and intra-matcher parallelization which will be discussed below

3.3.1 Gross et al. Matching Technique

Gross et al. proposed a parallel ontology matching system with a distributed infrastructure to incorporate intra-matcher and inter-matcher parallelism. The element- level and structure-level matching are also parallelized. For intra-matcher parallelism a size-based partitioning algorithm has been proposed by this system leading to better load balancing, limited memory consumption and scalability without reducing the effectiveness of match results.

The context attributes for each entity is first generated. Then the ontology is partitioned into set of partition based on the size-based partitioning algorithm achieving intra-matcher parallelism. Now the partition pairs are constructed one from each of the two ontologies. Each partition pair is assigned a processor. Within each processor the partition pair is parallel processed by the element level, structure-level, instance-based matchers achieving inter- matcher parallelism. The alignments from all the matcher can be aggregated to output the final alignment.

The drawbacks of this method are as follows. The matcher library consists of only few numbers of matchers. The partition algorithm uses only simple strategies for partitioning which can be improved.

3.4 Other Matching Technique

3.4.1 *AgreeementMaker Matching Technique*

AgreementMaker [21] is a schema and ontology matching algorithm consisting of wide range of matcher for lexical and structural feature of the ontology. Both serial and parallel matcher workflow is provided. The strength of the system lies in GUI which enable user to choose, control and execute the iterative matchers and their results. Through GUI the user can choose matcher from the matcher library based on the matching granularity (element wise, structural wise and instance wise), dominant features of input schema, etc.

The input ontology can be in XML, RDFS, OWL, or N3 format. The system consists of three layers. First matcher layer use the entity features (e.g., label, comments, annotations, and instances) and compute the similarity value using the syntactic and lexical matchers. The resulting similarity values are combined based on weighted average method. Second matcher layer use structural properties of the ontologies to compute the similarity value. Finally, third matcher layer combine the similarity value of first and second matcher layer.

The drawback of this system is that, the end user should be a sophisticated domain expert because to choose, control and execute the matcher the user must have domain knowledge.

3.4.2 *ASMOV Matching Technique*

ASMOV [22] **(A**utomated **S**emantic **M**atching of **O**ntologies with **V**erification) is an iterative ontology matching algorithm. The ASMOV achieves high effectiveness by post processing the alignment to remove the alignments which are semantically inconsistent. It also uses WordNet and the Unified Medical Language System (UMLS) to increase the effectiveness. But postprocessing of the alignments and use of external dictionary lead to more execution time and thus reduced efficiency. The input ontology should be in OWL-DL format. The input of ASMOV is two ontologies and an optional pre-determined alignment set. The similarity between entities belonging to two ontologies is computed by matching the string, structure and instances of the entities. It then uses the optional input alignment to adjust any calculated measures. A similarity among the entity pairs belonging to two ontologies computed as explained above is stored in a similarity matrix. For each entity choose the maximum similarity value as pre-alignment from the similarity matrix.

This pre-alignment must undergo a process of semantic verification, which is an extensive postprocessing to eliminate potential inconsistencies among the set pre-alignment. Five different kinds of inconsistencies are checked. One such inconsistency rule is Multiple-entity correspondences, e.g., if two alignments (a, b) and (b, c) exist then there must also be alignment like (a, c), if not the above two alignment cannot be verified and hence removed. The output of the above step is the semantically verified similarity matrix, which is then test against a termination condition. . If this condition is true, no more iteration is needed and the process stops. The resulting alignment is final alignment set.

The drawbacks of this system are as follows. When the given two ontologies are dissimilar effectiveness is decreased. Sometime semantic verification system eliminates too many alignments leading to less recall. For each iteration the ASMOV needs polynomial time, thus leading to inefficiency.

3.4.3 *RiMOM Matching Technique*

RiMOM [23] is the automatic ontology matching algorithm developed with dynamic selection of matchers for ontology

	QOM	Eric Peukert et al.	COMA++	Falcon-AO	Taxo-map	Anchor Flood	Gross et al.	RiMOM	ASMOV	Agreement Maker
XML Schemas as Input	No	Yes	Yes	No	No	No	No	No	No	(Yes)
Ontologies as Input	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GUI	No	Yes	Yes	(Yes)*	No	No	No	No	No	Yes
Linguistic Matcher	Yes	N/K	Yes	Yes	Yes	Yes	Yes	(Yes)	Yes	Yes
Structural Matcher	Yes	N/K	Yes	Yes	Yes	Yes	Yes	(Yes)	Yes	Yes
Instance Matcher	Yes	N/K	Yes	No	No	No	Yes	(Yes)	Yes	Yes
Early pruning	Yes	Yes	No	No	No	No	No	No	No	No
Schema/ Ontology partition	No	No	Yes	Yes	Yes	Yes	Yes	No	No	No
Parallel matching	No	(Yes)	(Yes)	No	No	No	Yes	No	No	No
Dynamic matcher selection	No	No	No	No	No	No	No	Yes	No	No
Mapping Reuse	No	Yes	Yes	No	No	No	No	No	No	No
OAEI participation	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Use of external dictionary	Yes	No	Yes	No	No	No	No	Yes	Yes	Yes

***** optional

Table 1. A Comparison of Ontology/Schema Matching Techniques

alignment tasks. The input ontology should be in OWL format. It considers lexical, structural and instance similarities. Based on the features of the input ontology and the predefined rules, appropriate matchers are chosen to apply for the matching task. RiMOM consist of six steps as follows

Ontology Preprocessing and Feature Factors Estimation. For each entity of both ontologies, generate the features of the entity like its name, label, children, etc. Then the label and structure similarity of the entity pairs are calculated which will be used in the following step.

Strategy (Matcher) Selection. The basic idea of strategy selection is that if two ontologies have some feature in common, then matcher based on these feature information are employed with high weight; and if some feature factors are two low, then these matcher may be not employed. The entities are first linguistically matched; structural matching is only applied if the schemas exhibit sufficient structural similarity.

Single strategy execution. The selected strategies from the above step is use to find the alignment independently. Each strategy outputs an alignment result.

Alignment combination. In this phase RiMOM combines the alignment results obtained by the selected strategies. The combination is conducted by a linear interpolation method.

Similarity propagation (optional). If the two ontologies have high structure similarity factor, RiMOM employs to find new alignment according to the structural information.

Alignment refinement. It refines the alignment results from the previous steps. It defined several heuristic rules to remove the unreliable alignments.

The drawback of RiMOM is its inefficiency for dealing with large scale ontologies. Eventhough it shows a very good effectiveness for large scale ontologies it consume long time and large amount of memory since it does not incorporate search space reduction techniques like early pruning, partition of ontology or parallel technique.

4. AN ANALYSIS OF DIFFERENT LARGE SCALE MATCHING TECHNIQUES

In this section we present an analysis of different large scale ontology/schema matching technique. Table 1 provides a brief comparison among ten match techniques which were discussed in the above section. The parameters used for comparison are

1) Input Format
    a. Ontology
    b. XML
2) GUI
3) Type of matcher
    a. Linguistic based matcher
    b. Structural based matcher
    c. Instance based matcher
4) Scalability techniques
    a. Early pruning techniques
    b. Partitioning techniques
    c. Parallelization techniques
5) Dynamic matcher selection
6) Mapping reuse
7) OAEI participation
8) Use of external dictionary

The analysis depicts that all matching techniques accepts ontology as input and only few matching techniques like Peukert et al., COMA++ and AgreementMaker accepts XML as input. GUI is provided only by Eric Peukert et al., COMA++, Falcon-AO and AgreementMaker. Eric Peukert et al., matching technique uses the matcher library and the types of matcher in the library are not mentioned. So we are unable to consider the Eric Peukert et al., matching technique for matcher type analysis. Linguistic and structural matcher is incorporated by all the matching technique, whereas the instance matcher is not implemented by Falcon-AO, Taxomap and AnchorFlood matching technique. The scalability technique early pruning is implemented by the QOM and Eric Peukert et al., matching technique, Schema/Ontology partitioning is applied in AnchorFlood, COMA++, Falcon-AO and Taxomap and parallelization is applied in Eric Peukert et al., COMA++ and Gross et al. matching technique. RiMOM is the only system to implement dynamic matcher selection. The Alignment/mapping reuse is incorporated only in COMA++ and Eric Peukert et al., matching technique. All the matching technique except QOM, Eric Peukert et al. and Gross et al. participated in **OAEI** (**O**ntology **A**lignment **E**valuation **I**nitiative) which is a benchmark competition for evaluating the new proposed matching techniques. QOM, COMA++, ASMOV, AgreementMaker and RiMOM uses external dictionary like WordNet , UMLS to increase the efficiency of matching.

5. CONCLUSION

The various schema and ontology matching techniques that could be used for large scale matching is discussed in this paper. The goal of this analysis paper is to present broad overview of matching techniques which are used to increase the effectiveness of the matching process by postprocessing of alignments, dynamic matcher selection, and full user control of match workflow through GUI. And also the efficiency of the matching process by early pruning strategy, partitioning of ontology/schema and parallelization of matching process. The analysis depicts that there is always a tradeoff between good effectiveness and good efficiency in the existing matching systems. Hence, we are working for a better effective and efficient large scale ontology matching technique which will combine the advantages and eliminate the disadvantage of the matching techniques discussed in this paper.

6. REFERENCES

[1] Shvaiko P, Euzenat J, "Ten challenges for ontology matching," *Confederated International Conference on the Move to Meaningful Internet Systems*, pp. 1164–1182,2008.

[2] Rahm E, "Towards Large-Scale Schema and Ontology Matching," *Schema matching and mapping*, Bellahsene Z, Bonifati A Rahm E, eds. New York: Springer Heidelberg, pp. 3–27, 2011.

[3] Smith K, Morse M, Mork P et al., "The role of schema matching in large enterprises," *Proceedings of CIDR*, 2009.

[4] Su W, Wang J, Lochovsky FH, "Holistic schema matching for web query interfaces," *Proceedings of international conference on extending database technology (EDBT)*. Springer, Heidelberg, pp 77–94, 2006.

[5] Kirsten T, Thor A, Rahm E, "Instance-based matching of large life science ontologies," *Proceedings of data integration in the life sciences (DILS)*. LNCS, Springer, Heidelberg, vol. 4544, pp 172–187, 2007.

[6] Zhang S, Mork P, Bodenreider O, Bernstein PA, "Comparing two approaches for aligning representations of anatomy," *Artif Intell Med* vol. 39, no. 3, pp 227–236, 2007.

[7] Avesani P, Giunchiglia F, Yatskevich M, "A large scale taxonomy mapping evaluation," *Proceedings of international conference on semantic web (ICSW)*. LNCS, vol. 3729. Springer, Heidelberg, pp 67–81, 2005.

[8] P. Shvaiko, J. Euzenat, " A survey of schema-based matching approaches," *Journal on Data Semantics IV, LNCS, Springer*, vol. 3730, pp. 146-171, 2005.

[9] Peukert E, Berthold H, Rahm E, "Rewrite techniques for performance optimization of schema matching processes," *Proceedings of 13th international conference on extending database technology (EDBT)*. ACM, NY, pp 453–464, 2010.

[10] Ehrig M, Staab S, "Quick ontology matching," *Proceedings of international conference semantic web (ICSW)*. LNCS, vol 3298. Springer, Heidelberg, pp 683–697, 2004.

[11] Hanif MS, Aono M, "An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size," *Journal of Web Semantics*, vol. 7, no. 4, pp. 344–356, 2009.

[12] Do HH, Rahm E, "Matching large schemas: Approaches and evaluation," *Journal of Information System*, vol. 3, no. 6, pp. 857–885, 2007.

[13] Hu W, Qu Y, Cheng G, "Matching large ontologies: A divide-and-conquer-approach," *Journal of Data Knowledge Engineering*, vol. 67, no. 1, pp. 140–160, 2008.

[14] Hamdi F, Safar B, Reynaud C, Zargayouna H, "Alignment-

based partitioning of large-scale ontologies," *Advances in knowledge discovery and management*, 1st ed. New York: Springer Heidelberg, pp. 251–269, 2009.

[15] Y. Qu, W. Hu, G. Cheng, "Constructing virtual documents for ontology matching," *Proceedings of the 15th International World Wide Web Conference*, ACM Press, pp. 23–31, 2006.

[16] Stoilos G, Stamou G, Kollias S, "A string metric for ontology alignment," *Proceedings of the 4th International Semantic Web Conference*, LNCS, Springer, vol. 3729, pp. 624–637, 2005.

[17] W. Hu, N. Jian, Y. Qu, Y. Wang, "GMO: a graph matching for ontologies," *Proceedings of the K-CAP Workshop on Integrating Ontologies*, pp. 41–48, 2005.

[18] Guha S, Rastogi R, Shim K, "ROCK: a robust clustering algorithm for categorical attributes," *Proceedings of the 15th International Conference on Data Engineering*, pp. 512–521, 1999.

[19] Yuzhong Qu, Wei Hu, and Gong Chen, "Constructing virtual documents for ontology matching," *Proc. 15th International World Wide Web Conference (WWW)*, Edinburgh (UK), pp 23–31, 2006.

[20] Gross A, Hartung M, Kirsten T, Rahm E, "On matching large life science ontologies in parallel," *Proceedings of 7th international conference on data integration in the life sciences (DILS)*. LNCS, vol 6254. Springer, Heidelberg, 2010.

[21] Cruz IF, Antonelli FP, Stroe C, "AgreementMaker: Efficient matching for large real-world schemas and ontologies," *PVLDB*, VLDB Endowment, vol. 2, no. 2, pp 1586–1589, 2009.

[22] Jean-Mary YR, Shironoshita EP, Kabuka MR, "Ontology matching with semantic verification," *J Web Sem* vol. 7, no. 3, pp. 235–251, 2009.

[23] Li J, Tang J, Li Y, Luo Q, "RiMOM:A dynamic multistrategy ontology alignment framework," *IEEE Trans Knowl Data Eng* vol. 21 , no. 8, pp. 1218–1232, 2009.