International Journal of Computer Applications (0975 – 8887)
Volume 37– No.12, January 2012

A Study of Different Ontology Matching System

Mohammed Muzaffar Hussain
Research Scholar
Sri Chandrashekhendra Saraswathi Viswa
Maha Vidyalaya University,
Enathur, Kanchipuram – 631 561, India.

Dr. S.K. Srivatsa
Senior professor,
St. Joseph's College of Engg,
Jeppiaar Nagar, Chennai-600 064

ABSTRACT

Ontology matching is a key interoperability enabler for the Semantic Web, as well as a useful tactic in some classical data integration tasks. It takes the ontologies as input and determines as output an alignment, that is, a set of correspondences between the semantically related entities of those ontologies. These correspondences can be used for various tasks, such as ontology merging and data translation. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to inter-operate. In this paper we present an overview of recently proposed matching techniques which is participated in OAEI and Ontology matching tools which achieve high match efficiency with respect to conference track of OAEI 2010.In particular we discuss lessons learned on strong points and remaining weaknesses of various matching techniques is summarized.

General Terms

Classes, Objects, Properties

Keywords

Similarity Measure, Ontology Matching, Ontology Alignment.

1. INTRODUCTION

1.1 GENERAL

The World Wide Web (WWW) is widely used as a universal medium for information exchange. However, semantic interoperability in the WWW is still limited due to the heterogeneity of information [1]. Ontology, a formal, explicit specification of a shared conceptualization [2], has been suggested as a way to solve the problem. With the popularity of ontologies, ontology mapping, aiming to find semantic correspondences between similar elements of different ontologies, has attracted many research attentions from various domains.

1.2 ONTOLOGY MATCHING

Ontology matching is the process of automatically finding the relationship between the elements or concepts of two or more formal ontologies. Computed correspondences typically need to be validated and corrected by users to achieve the correct match mappings. Match mappings are needed in many areas, in particular for data integration, data exchange, or to support schema and ontology evolution.

1.3 TECHNIQUES

Different techniques have been examined in ontology mapping [3] [4] [5], e.g.,

1. Terminological (String based(name similarity), Language based (lemmatisation), Linguistic resources(lexicons))
2. Structural (Constrained based(lexicons),Taxonomy based (subsumption)
3. Extensional (Data analysis)
4. Semantic (Upper level ontologies,Model based (DL reasoner))
5. Instance based approach.
6. Use of auxiliary.
7. combination strategies
8. Filtering Techniques

1.4 ALIGNMENTS [6]

Using ontologies is the rich way to attain interoperability among heterogeneous systems within the Semantic web. However, as the ontologies underlying two systems are not necessarily compatible, they may in turn need to be consistent. Ontology equalization requires most of the time to find the correspondences between entities (e.g., classes, objects, properties) occurring in the ontologies. We call a set of such correspondences an alignment .We have designed a format for expressing alignments in a uniform way. The goal of this format is to be able to share on the web the available alignments. The format is expressed in RDF, so it is freely extensible.

1.5 REFERENCE ALIGNMENTS

Reference alignment is a correspondence which is correct made by domain experts. All the reference alignments are available for the conference track in OAEI. The complex and laborious task of generating the reference alignment has been conducted by a combination of computational methods and an extensive manual evaluation with the help of domain experts.

1.6 MEASURES [3]

Instance matching algorithms are evaluated according to the following parameters.

– Precision: the number of correct retrieved mappings / the number of retrieved mappings.
– Recall: the number of correct retrieved mappings / the number of expected mappings.
– F-measure: 2 · (precision · recall) / (precision + recall).

1.7 SOME OF THE PARTICIPANTS OF OAEI [7]

- AgreementMaker (AgrMaker)
- FALCON
- ASMOV
- CODI
- AROMA

2. AGREEMENTMAKER [8]

The AgreementMaker method is specific in that it attributes a powerful user program, a bendable and extensile architecture, and merged evaluation engine that relies on inbuilt quality measures, and semi-automatic and automatic methods.

2.1 Matching Algorithms

2.1.1 Syntactic

Base Similarity Matcher (BSM)

BSM is a basic string matcher that computes the similarity between concepts by examination all the strings connected with them.

Parametric String Matcher (PSM)

PSM is a more in-depth string matcher, which for the contention is set to use a substring measure and an edit distance measure.

Vector-based Multi-word Matcher (VMM)

VMM compiles a virtual document for every concept of an ontology, change the resulting strings into TF-IDF vectors and then computes their similarity using the cosine similarity measure

Advance similarity matcher

ASM is a string-based matcher that computes mappings between source and target concepts (including their properties) by comparing their local names, and providing better similarity evaluation in particular when compound terms are used. ASM outper-forms generic string-based similarity matchers because it is based on a deeper linguistic analysis.

2.1.2 Structural

Descendants Similarity Inheritance Matcher (DSI)

This matcher is based on the idea that if two nodes are similar, then their descendants should be similar.

Siblings Similarity Contribution Matcher (SSC)

The Group Finder Matcher (GFM) is another structural matcher that separate out the mappings provided by another matcher (the input matcher).It determine groups of idea and properties in the ontologies and expect that two concepts (or properties) that exist to two groups that were not mapped by the input matcher will likely have different meanings and should not be mapped.

2.2 STRONG FACTOR OF AGRMAKER

Offers a user interface built on an extensible architecture. This architecture allows to configure the matching process to a high degree. For that purpose AGREEMENTMAKER uses internally different methods and similarity measures that can be combined in different ways. Because of using three layers in the Agreementmaker approach, we are getting the best results. Otherwise if we are using only string matcher methods then we will get the best results for some entities. Finally, third layer matcher combines the results of two or more matchers so as to obtain a unique final matching in two layers.

2.3 WEAK FACTOR OF AGRMAKER

A component dedicated to the detection and avoidance of incoherence is not mentioned.

2.4 RESULT

In order to get the result for the AGREEMENTMAKER.I have used the alcomo software[9] and netbeans IDE 7.0[10] for the results of conference track and the results are mentioned in terms of F-Measure. And the results are represents in the tables 1. And sample interface is listed in fig 1.I have used the ontology, alignment, and reference alignment from OAEI web site [11].

Table 1. Results for the conference track for AGRMAKER using alcomo

ONTO-LOGY	Con-ference	ConOf	Edas	Ekaw	Iasted	sigkdd
cmt	48.78	55.17	64.51	46.15	57.14	82.75

3. FALCON [12]

Falcon is an infrastructure for Semantic Web applications, which aims at providing fundamental technologies for finding, aligning and learning ontologies, and, ultimately for capturing knowledge from the Web via an ontology-driven approach.

3.1 Matching Algorithms (Matchers)

Linguistic matching

V-Doc takes a linguistic approach to ontology matching. Its originality is the idea of build virtual documents. Fundamentally, as an accumulation of heavy words, the virtual document of a domain entity (e.g., a class or a property) in an ontology contains not only the local descriptions, but also the adjacent information to indicate the knowing meaning of the entity. Document similarity can be calculated via traditional vector space techniques, and further be used in certain similarity-based approaches to ontology matching. Specifically, the RDF graph structure is used to obtain the description information from adjacent domain entities.

Structural matching

GMO is an repetitive structural matcher. It uses RDF bipartite graphs to represent ontologies and computes structural similarities between domain entities and between statements (triples) in ontologies by longhand propagating similarities in the bipartite graphs. GMO takes a set of external alignments as input, which are typically found previously by other matchers (in current implementation, the external alignments are the ones with full similarities that are from V-Doc and I-Sub), and incrementally return extra alignments as output. The performance of GMO improves as the precision of external alignments increases.

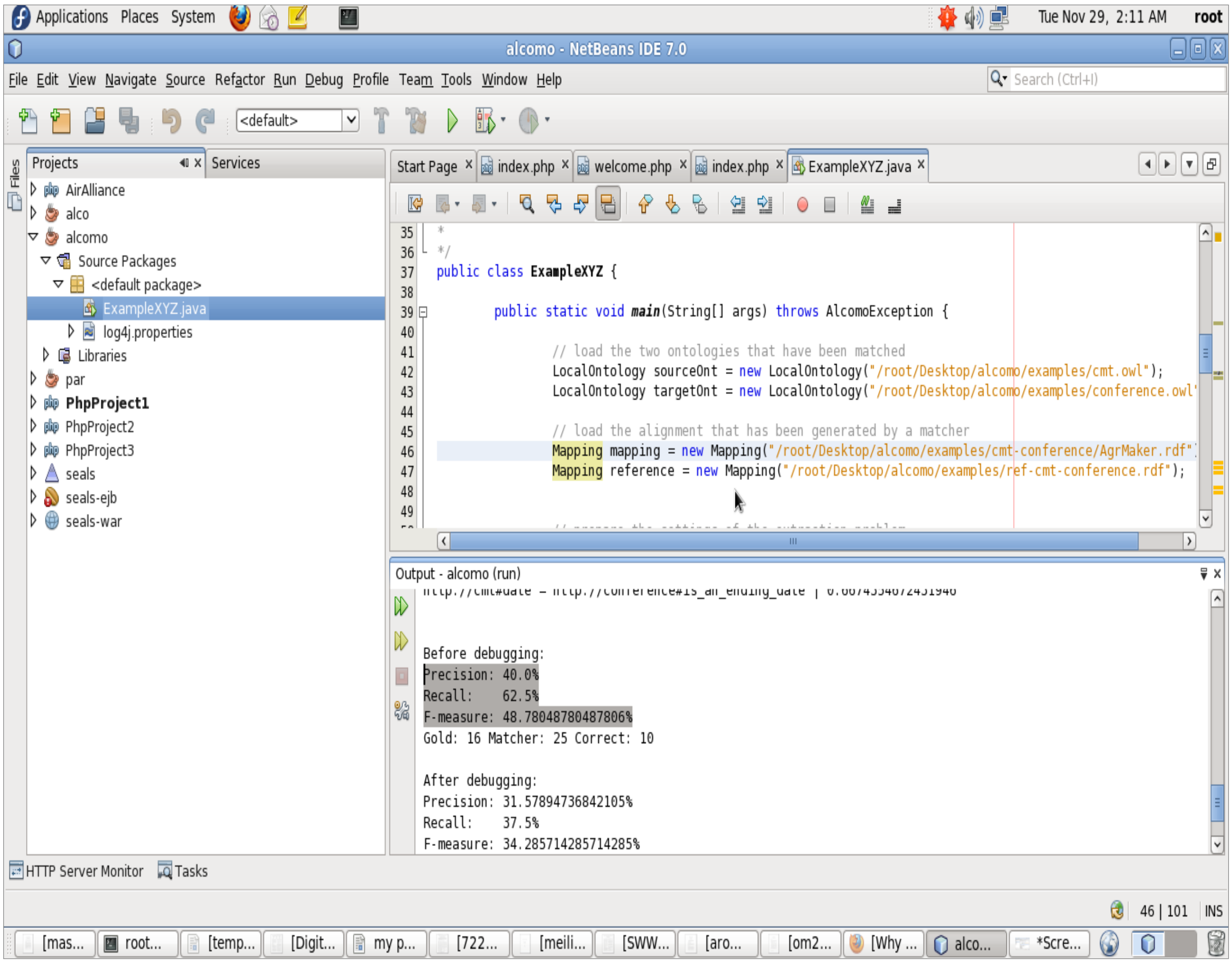
Partition-based block matching of large-scale ontologies

Large-scale ontologies hike a big challenge to present ontology matching systems because of their property and their large nature. PBM uses a divide-and-conquer approach to finding block mappings between large-scale ontologies , which has two major advantages: (1) it avoids our matching system suffering from lack of memory; and (2) it decreases the execution time without loss of quality, because it is likely that large portions of one or both input ontologies have no matching counterparts.

In special, PBM firstly divider domain entities of each ontology into a number of little clusters based on their structural proximity (e.g., the distance between classes in the class hierarchy, and the overlapping between the domains of properties), and then builds blocks by assigning RDF sentences to the clusters.RDF sentences can provide more integrated syntactic and semantic structures than RDF statements, because they can encapsulate blank nodes into them. Finally, blocks are matched via anchors (i.e., pre-found alignments by I-Sub) and only block pairs with high similarities are further matched by V-Doc and GMO.

Coordination rules

Due to the mixed ways in explicit semantics and the various reasoning capabilities of ontology languages, ontologies are often represented differently. So, it is necessary to align ontologies before executing elementary matchers. Falcon-AO implements 21 coordination rules to eliminate worthless axioms and reduce structural heterogeneity between the ontologies to be matched. Generally, three categories of coordination rules can be assigned to elementary matchers : (1) removing excess statements; (2) inferring omitted statements, e.g., the ones involving owl:inverseOf; and (3) build List structures, e.g., using the rdfs:member property to describe the relationship between a list and each of its members, instead of RDF collection vocabularies(rdf:first, rdf:rest and rdf:nil).



```java
 *
 */
public class ExampleXYZ {

        public static void main(String[] args) throws AlcomoException {

                // load the two ontologies that have been matched
                LocalOntology sourceOnt = new LocalOntology("/root/Desktop/alcomo/examples/cmt.owl");
                LocalOntology targetOnt = new LocalOntology("/root/Desktop/alcomo/examples/conference.owl'
                
                // load the alignment that has been generated by a matcher
                Mapping mapping = new Mapping("/root/Desktop/alcomo/examples/cmt-conference/AgrMaker.rdf"
                Mapping reference = new Mapping("/root/Desktop/alcomo/examples/ref-cmt-conference.rdf");
```

```
http://cmt#date = http://conference#is_an_ending_date | 0.6674554672431946


Before debugging:
Precision: 40.0%
Recall:    62.5%
F-measure: 48.78048780487806%
Gold: 16 Matcher: 25 Correct: 10

After debugging:
Precision: 31.57894736842105%
Recall:    37.5%
F-measure: 34.285714285714285%
```

Fig 1: sample interface of alcomo software in Netbeans IDE 7.0

Similarity combination strategy

Similarity combination is an crucial and hard issue in structure ontology matching systems. Falcon-AO develops an approach to step by step tune up the thresholds (cutoffs) based on the measures of both the linguistic comparability and the structural compare, which makes Falcon-AO robust in a variety of matching scenarios.

The linguistic compare is calculated by examining the dimension of the candidate alignments against the minimum number of domain entities in the ontologies. The notion is that if the number of alignments is close to the number of domain entities in the smallest ontology, then we are almost done with matching, and it is not necessary to run GMO anymore. The structural equivalence is measured by comparing which built-in properties are used in the ontologies, and how often. Furthermore, it estimates the number of correct alignments from GMO in proportion to the ones from V-Doc and I-Sub.

Falcon-AO considers these two kinds of comparison to mechanically determine the similarity combination strategy. For example, if the linguistic comparability is high, Falcon-AO would lower the thresholds of V-Doc and I-Sub, so that more alignments from V-Doc and I-Sub can be combined to the final alignments.

3.2 Strong Factor of Falcon

As analyze to other systems, Falcon-AO has three strengths: (1) it can fulfill various matching tasks, especially matching large-scale ontologies; (2) it can stably achieve very good precision and recall on both systematic and blind tests. (3) It is economic; all the tasks can be carried out in a sensible time only on an ordinary personal computer. For small ontologies, Falcon-AO can complete the matching process within several seconds, even for large-scale ontologies, Falcon-AO can accomplish them within a few hours.

3.3 Weak Factor of Falcon

A component for generating coherent alignments is to our knowledge not implemented in the FALCON system.

3.4 Results

In order to get the result for the FALCON_AO.I have used the alcomo software and netbeans IDE 7.0 for the results of conference track and the results are mentioned in terms of F-Measure. And the results are represents in the tables 2.I have used the ontology, alignment, and reference alignment from OAEI web site.

Table 2. Results for the conference track for FALCON AO using alcomo

ONTO-LOGY	Con-ference	ConOf	Edas	Ekaw	Iasted	sigkdd
cmt	52.94	44.44	69.23	54.54	66.66	83.33

4. ASMOV [13]

Automated Semantic Matching of Ontologies with Verification (ASMOV) is a good rule that uses lexical and structural characteristics of two ontologies to repetitive calculate a similarity measure between them, derives an alignment, and then verifies it to ensure that it does not contain semantic inconsistencies.

4.1 Matching Algorithms (Matchers)

4.1.1 Similarity calculation

A lexical (or terminological) similarity

The lexical property attribute consists of the entire human-readable message provided in ontology. Three such lexical features are considered in OWL ontologies: the id, the label, and the comment.

Entity-set similarity

Entity-set similarity uses greddy selection algorithm in order to obtain a set of correspondences between the first and second ontology. This algorithm iteratively chooses the largest correspondence in S and eliminates every other similarity for first and second from S, until all dissimilar are eliminated.

Relational similarity

The relational similarity is computed by union the similarities between the parents and children of the entities being analyze. As classes or properties may include multiple parents and children, the similarity calculation is calculated as the average of the similarities of all parents or children, in order to restrict the results between 0 and 1.

Internal similarity

The internal similarity is calculated differently for classes and properties in the ontology.

Extensional similarity

The extensional similarity measure for two classes is calculated in the same way as the children hierarchical similarity.

To determine extensional similarity between properties, all individuals that contain a value for a given property are analyzed to determine a list of possible matches. Only properties which are both object and both datatype can have an extensional similarity; otherwise, the similarity is undefined.

4.1.2 Semantic verification process

Pre-alignment extraction

In order to perform semantic verification, a pre-alignment is first extracted from the similarity matrix that results from the similarity calculations**.** This pre-alignment is obtained using a greedy algorithm.

Semantic verification

The pre-alignment is then passed through a process of semantic verification, designed to verify that certain axioms inferred from an alignment are actually asserted in an ontology, removing correspondences that lead to inferences that cannot be verified.

4.2 Strong Factor in Asmov

In ASMOV algorithms they have using the concepts called semantic verification, the advantages of semantic verification is if we have small amount of information in the ontology then we can calculate the measures.

4.3 Weak Factor in Asmov

ASMOV still needs to improve its ability to work with very large ontologies and resources. While some disk-based storage of partial results has been implemented, the entire contents of the ontologies still needs to loaded in memory prior to performing the matching process. This needs to be further improved to use permanent storage in order to enable the alignment of very large ontologies. We also need to continue the implementation of the ability to infer assertions in order to utilize them for similarity measurement and semantic verification. In addition, we are also working in the improvement of the general scalability of the ASMOV algorithm for the processing of ontologies with a large number of entities. Finally, we need to reexamine the use of an appropriate threshold value to optimize accuracy.

4.4 Results

In order to get the result for the ASMOV.I have used the alcomo software and netbeans IDE 7.0 for the results of conference track and the results are mentioned in terms of F-Measure. And the results are represents in the tables 3.I have used the ontology, alignment, and reference alignment from OAEI web site.

Table 3. Results for the conference track for ASMOV using alcomo

ONTOL-OGY	Con-ference	ConOf	Edas	Ekaw	Iasted	sigkdd
cmt	47.61	37.83	55.55	43.74	36.36	54.05

5. CODI [14]

The question of linking entities in heterogeneous and localized data repositories is the impulsive power behind the data and knowledge integration effort. We describe our probabilistic-logical alignment system CODI (Combinatorial Optimization for Data Integration). The system supply an indicative structure for the alignment of individuals, concepts, and properties of two heterogeneous ontologies. CODI vantage both logical schema information and lexical similarity measures with a well-defined semantics for A-Box and T-Box matching. The alignments are computed by solving corresponding combinatorial optimization problems.

5.1 Matching Algorithms (Matcher)

Cardinality Constraints

A performing often practical in real-world setting is the mixture of a functional one-to-one alignment.. Within the Markov Logic framework, we can include a set of difficult cardinality constraints, confining the alignment to be functional and one-to-one.

Coherence Constraints

Incoherence occurs when axioms in ontologies advantage to logical contradictions. Distinctly, it is desired to avoid incoherence during the alignment process. All present approaches that put a focus on alignment coherence remove correspondences after computing the alignment. Within the

Markov Logic framework we can incorporate incoherence reducing constraints during the alignment process.

Stability Constraint

Various formulations to ontology matching transfer alignment grounds derived from structural relationships between concepts and properties. These methods leverage the concept that present evidence for the equivalence of concepts C and D also makes it more likely that, for example, child concepts of C and D are equivalent. One such approach to grounds propagation is similarity flooding. As a reciprocal idea, the general notion of stability was introduced, expressing that an alignment should not introduce new structural knowledge.

Combination of Different Similarity Measures

Here lexical string similarity measures significantly increase. In a first step we collect and standardize all string information like ids, labels and annotations from the entities. During the normalisation process we split tokens into separate words if necessary (e.g.hasAuthor is transformed to has Author), replace special characters with spaces, and remove few words like a or the according to a stop-words list

5.2 Strong Factor of Codi

CODI performs concept, property, and instance alignments. It combines logical and structural information with a-priori similarity measures in a well-defined way by using the syntax and semantics of Markov logic. The system therefore not only aligns the entities with the highest lexical similarity but also enforces the coherence and consistency of the resulting alignment.
The overall results of the young system are very promising. Especially when considering the fact that there are many optimization possibilities with respect to the lexical similarity measures that have not yet been investigated. The strength of the CODI system is the combination of lexical and structural information and the declarative nature that allows easy experimentation. We will continue the development of the CODI system and hope that our approach inspires other researchers to leverage terminological structure for ontology matching.

5.3 Weak Factor of Codi

Because Coherence and consistency are taken into account by a set of hard constraints inspired by the patterns the system takes much time to produce the results. Hence time complexity is little bit high compare to other matching system.

5.4 Results

In order to get the result for the CODI.I have used the alcomo software and netbeans IDE 7.0 for the results of conference track and the results are mentioned in terms of F-Measure. And the results are represents in the tables 4. I have used the ontology, alignment, and reference alignment from OAEI web site.

Table 4. Results for the conference track for CODI using alcomo

ONTO-LOGY	Con-ference	ConOf	Edas	Ekaw	Iasted	sigkdd
cmt	36.36	38.09	76.19	62.5	88.88	72.72

6. AROMA [15]

AROMA is a hybrid, extensional and asymmetric matching approach designed to find out relations of equivalence and subsumption between entities, i.e. classes and properties, issued from two textual taxonomies (web directories or OWL ontologies). Our approach makes use of the association rule paradigm [Agrawal et al., 1993], and a statistical interestingness measure. AROMA relies on the following assumption: An entity A will be more specific than or equivalent to an entity B if the vocabulary (i.e. terms and also data) used to describe A, its descendants, and its instances tend to be included in that of B.

6.1 Matching Algorithms (Matchers)

The pre processing stage represents each entity

The first stage constructs a set of relevant terms and/or datavalues for each class and property. To do this, we extract the vocabulary of class and property from their annotations and individual values with the help of single and binary term extractor applied to stemmed text. In order to keep a morphism between the partial orders of class and property subsumption hierarchies in one hand and the inclusion of sets of term in the other hand, the terms associated with a class or a property are also associated with its ancestors.

Association rules between entities

The second stage of AROMA discovers the subsumption relations by using the association rule model and the implication intensity measure. In the context of AROMA, an association rule a → b represents a quasi-implication (i.e. an implication allowing some counter-examples) from the vocabulary of entity a into the vocabulary of the entity b. Such a rule could be interpreted as a subsumption relation from the antecedent entity toward the consequent one. For example, the binary rule car → vehicle means:"The concept car is more specific than the concept vehicle". The rule extraction algorithm takes advantage of the partial order structure provided by the subsumption relation, and a property of the implication intensity for pruning the search space.

Cleaning and enhancing the resulting alignment

The last stage concerns the post processing of the association rules set. It performs the following tasks: deduction of equivalence relations, suppression of cycles in the alignment graph, suppression of redundant correspondences, selection of the best correspondence for each entity (the alignment is an injective function), the enhancement of the alignment by using equality and a string similarity -based.

6.2 Strong Factor of Aroma

AROMA is a time efficient matcher compare to other matching alignments. AROMA takes very less memory space in order to find alignments.

6.3 Weak Factor of Aroma

On anatomy track the precision is also degradated due to the subomption correspondences it returns**.** One way for doing that is to tune the parameters and also to had some structural matcher

6.4 Results

In order to get the result for the AROMA.I have used the alcomo software and netbeans IDE 7.0 for the results of conference track and the results are mentioned in terms of F-Measure. And the results are represents in the tables 5. I have used the ontology, alignment, and reference alignment from OAEI web site.

Table 5. Results for the conference track for CODI using alcomo

ONTO-LOGY	Con-ference	ConOf	Edas	Ekaw	Iasted	sigkdd
cmt	36.84	21.42	46.15	26.08	33.33	46.15

7. RESEARCH DIRECTION

Agreementmaker

Agreementmaker using of linguistic matching and structural matching, finally joining the results of both the matcher and getting the measures. In this matcher, it is not describe about any new specific concepts. All the concepts used in the AGREEMENTMAKER are common strategies which are there in ontology matching. In this algorithm it is not mentioned what is the size of the ontology. What happened if the size of the ontology is very big? If semantic verification implies in this algorithm, then we would have got the good result.

Falcon

In falcon they have used many new concepts like partition-based-block matching of large scale ontologies, similarity combination strategies and coordination rule .In coordination rule, saying that redundant statement removing. In many cases we can see that many names are same and other information is same. If we remove redundant statement there is lot of chances are there going out of statement which is necessary for our ontology matching. Falcon used that reconstructing the list structure; if we do this there is chances are there that many classes are losing its property.

8. COMPARISION OF DIFFERENT ONTOLOGY MATCHING SYSTEM

Table 6 showing five different ontology matching systems which is participated in OAEI

Matcher	Linguistic	Structural	contribution
AGREEMENTM-AKER	Label,comments,annotation & instance matching	DSI & SSC	User interface,control,combin-ation & LWC
FALCON	V-Doc	GMO	V-Doc,Coordination rule
ASMOV	Lexical elements (id, label, and comments)	relational structure (ancestor-descendant hierarchy),internal structure	semantic verification
CODI	lexical similarity lexSim	Internal structural matching.	Markov Logic Framework
AROMA	classes and properties, by a set of terms	-----	association rules between entities

Asmov

In asmov we have similarity verification and semantic verification. In semantic verification they remove correspondence that is less likely to be satisfiable based on the information present in the ontologies. In some cases that removed correspondence is useful in order to get the good measures

Codi

CODI uses both soft and hard constraints that guide the matching process. It defines a matching problem as an optimization problem that takes care of both types of constraints. If we ignore the soft constraints the correspondences filtered out by CODI nearly coincide with a global optimal diagnosis. CODI has generated the best alignments in terms of f-measure for the CONFERENCE track of OAEI 2010. Further improvements have to aim at generating good alignments with higher recall. Alignments generated by CODI have a very low degree of incoherence; however, CODI cannot guarantee coherence of the generated alignments in general

Aroma

Divides the matching process in three successive steps. The final step aims at cleaning and enhancing the resulting alignment. This is done, for example, by removing redundant correspondences. While cycles in the alignment graph are suppressed in this step, the suppression of incoherence is not mentioned.

9. CONCLUSION

In this study paper we have discussed about many ontology matching system and discussed about the advantages and disadvantages. we have provided the results of the different ontology matching system with conference track and we have the f-measures by using alcomo software and we have given the research direction for all ontology matching system in which we have said about how to increasing the efficiency and effectiveness of the ontology matching system

10. REFERENCES

[1] G.Widerhold, Mediators in the architecture of future information systems,IEEE Computer 25(3)(1992).

[2] Gruber,T. (1993). “A Translation Approach to portable ontology specification.” Knowledge cquisition 5920:199-220

[3] Jérôme Euzenat and Pavel Shvaiko. Ontology Matching. Springer,eo 2007

[4] Willem Robert van Hage. Evaluating Ontology-Alignment Techniques. PhD thesis, Vrije Universiteit Amsterdam, 2008.

[5] He Tan and Patrick Lambrix by A method for recommending ontology alignment strategies

[6] The Ontology Alignment Source http://alignapi.gforge.inria.fr/

[7] OAEI Paticipants

http://oaei.ontologymatching.org/2010/results/conference/index.html

[8] 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.

[9] ALCOMO Software http://web.informatik.uni-mannheim.de/alcomo/

[10] NetBeans IDE http://netbeans.org/

[11] Alignment, Reference alignments, ontology http://oaei.ontologymatching.org/2010/results/conference/index.html

[12] Wei Hu and Yuzhong Qu. Falcon-AO: A practical ontology matching system. Journal of Web Semantics. 6(3): 237-239, 2008. (System Paper, SCI & EI index, JCR 2008 IF: 3.023)

[13] Yves R. Jean-Mary , E. Patrick Shironoshita , Mansur R. Kabuka Ontology matching with semantic verification

[14] Jakob Huber, Timo Sztyler, Jan Noessner, and Christian Meilicke CODI: Combinatorial Optimization for Data Integration – Results for OAEI 2011

[15] Jérôme David-AROMA results for OAEI 2011