Automatic Ontology Merging by Hierarchical Clustering and Inference Mechanisms

Nora Maiz, Muhammad Fahad, Omar Boussaid, Fadila Bentayeb
(ERIC Laboratory, University of Lyon2, Bron, France
firstname.lastname@univ-lyon2.fr)

Abstract: One of the core challenges for current landscape of ontology based research is to develop efficient ontology merging algorithms which can resolve the mismatches with no or minimum human intervention, and generate automatic global merged ontology on-the-fly to fulfil the needs of automated enterprise business applications and mediation based data warehousing. This paper presents our approach of ontology merging in context of data warehousing by mediation that aims at building analysis contexts on-the-fly. Our methodology is based on the combination of the statistical aspect represented by the hierarchical clustering technique and the inference mechanism. It generates the global ontology automatically by four steps. First, it builds classes of equivalent entities of different categories (concepts, properties, instances) by applying a hierarchical clustering algorithm. Secondly, it makes inference on detected classes to find new axioms, and solves synonymy and homonymy conflicts. This step also consists of generating sets of concept pairs from ontology hierarchies, such as the first component subsumes the second one. Third, it merges different sets together, and uses classes of synonyms and sets of concept pairs to solve semantic conflicts in the global set of concept pairs. Finally, it transforms this set to a new hierarchy, which represents the global ontology.

Keywords: Ontology Merging, Similarity Measure, Hierarchical Clustering and Inference, Data Warehouse design, Data Mining
Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

1 Introduction

Decision tools are more and more used in modern companies to conduct analysis and take decisions at data that originate from distributed and heterogeneous data sources. Therefore, data integration is crucial since the analysis context, also called data cube, is built using data from different data sources in the same company or shared with other companies or on the web. There are two main strategies for data integration, i.e., data warehousing [Inmon, 92; Kimball, 98] and mediation [Goasdoue, 00; Huang, 00; Lamarre, 04]. The goal of former strategy is to build a centralized database that contains all data coming from different data sources modeled in multidimensional way promoting on-line analytical processing. This approach is characterized by its performance in terms of query response time since the data is warehoused that facilitates the decision processes. But, when data changes over time, decisional tools necessitate several updates for sound decision making. The updating processes are achieved using data warehouse refreshment strategies that cause much additional cost. To tackle this problem, we propose mediator approach to construct a virtual data warehouse to process analysis context on-the-fly.

Mediator approach consists of defining three elements; data source schemas as

local schemas, the mediator layer as a global schema and correspondence rules between local schemas. Querying data from their real sources to make decision consists of defining decisional queries. For formulating decisional queries for the mediator, we must first define a global schema that allows execution of this kind of queries. To fulfill this task, we need a powerful strategy of query transformation from the global schema language to the data sources languages. Furthermore, the obtained results from different data sources are combined to build the data cube on-the-fly. Since that we are in the analysis and decision domain, we are interested more by the pertinence of query results, so a simple search of data is not sufficient and we must proceed to a semantic research based on the semantic of terms used in the global schema or in the user query. This requires usage of ontologies as a support to data source semantic representation to ensure the knowledge sharing between different heterogeneous data sources or between different users. In this context, we propose an initial strategy for the definition of the global schema of the mediation system that aims at data searching. We use the classification technique to build the concept classes for the global ontology starting from the local ontologies that are representative of local data sources. The clustering technique based on semantic similarity is used to define clusters of concepts in the global ontology by merging local ontology classes. It takes into account the concept context that is defined as a set of roles that link this concept to other ones.

The remainder of this paper is structured as follows. Section 2 throws a light on state-of-the-art systems. Section 3 presents the methodology of our ontology merging system that exploits hierarchical clustering and inference mechanisms. Section 4 discusses the experimental validation of our approach. Finally, we draw some conclusions and show ongoing research aspects in section 5.

2 State-of-the-Art on Ontology Mapping and Merging

There are many approaches and systems for ontology alignment and mapping in research literature. IF-Map exploits instance similarity approach based on the formal concepts analysis for mapping of source ontologies by considering common reference ontology [Kalfoglou, 03]. GLUE integrates the instance matching with machine learning approach, and calculates the probabilities of concept matching by analyses of taxonomic structure for ontology integration [Doan, 04]. OBSERVER aims to work with semantic heterogeneities between distributed data repositories, and translates user queries from different ontologies using inter-ontology relationships (mappings) and retrieves desired data [Mena, 00]. QOM exploits the heuristic based dynamic programming approach for choosing only promising candidate mappings, and thus reduces the runtime complexity [Ehrig, 04]. OLA transforms ontologies to OWL-Graphs, and use Valtchev's similarity measure to compare entities belonging to the same category (Object property, datatype property, etc.) for find alignments between them [Euzenat, 04]. Besides these approaches, there are some semantic ontology matching techniques, such as CtxMatch [Bouquet, 06], S-Match [Giunchiglia, 04] and ASMOV [Jean-Marya, 09]. CtxMatch and S-Match follows the same methodology by translating concepts into the Description Logic (DL) formulas and then solves the propositional satisfiability problems with the help of available DL reasoners. Mascardi et al. make use of upper ontologies as semantic bridges for matching source

heterogeneous ontologies [Mascardi, 10].

For ontology merging, there are very few approaches contributed in the research literature. The semi-automatic interactive tools PROMPT and AnchorPROMPT [Noy, 03], and Chimaera [McGuinness, 00] exploit concept labels and to some extent the structure of source ontologies for ontology merging. These tools have no ability to find correspondences between concepts that are semantically equivalent but modeled with different names.. FCA-Merge is an algorithm for ontology merging that defines an ascending formal method of ontologies merging based on a set of natural language documents [Stumme, 01]. They use techniques for natural language treatment and concepts formal analysis to derive the concept lattice. The later is explored and transformed to ontology with the human intervention. H-Match and Merge, a dynamic ontology matching algorithm developed in the Helios framework, adopts another interesting approach by using linguistic and contextual affinity of concepts in peer-based systems [Castano, 04].

Our research on ontology merging topic has two folds. First, our semantic based ontology merger, DKP-OM, follows the hybrid approach and uses various inconsistency detection algorithms in initial mapping found in first steps [Fahad, 07]. Our hybrid strategy makes it possible to find all possible mappings, and semantic validation of mappings gives very promising final results by ignoring the incorrect correspondences which don't satisfy the test criteria. Secondly the contribution presented in this paper, automatic merging of local ontologies by clustering and inference mechanisms, for building analysis context (merged global ontology) on-the-fly in the context of data warehousing by ontology mediation approach [Maiz, 07]. The previous approaches in research literature use ontologies in XML (Extensible Markup Language), RDF (Resource Description Framework) or OWL-Lite (Ontology Web Language) format, and are not capable for automatic generation of global merged ontology. In addition, majority of them use similarity measures that cover at least the ontology structure and use a stabilization threshold to stop the alignment process, which limits the semantic propagation resulting reduction in precision. Moreover, these approaches support only two ontologies to be aligned, contrary to the reality where several ontologies need to be aligned in the same system for their share and reuse especially in case of data warehouse design. So, we need a new approach that takes into account the scalability by supporting several ontologies at the time. It is the case of our approach that fulfills these challenges as explained below.

3 Ontology Integration by Hierarchical Clustering and Inference

The main idea in our approach is to combine the power of the statistical approach represented by the hierarchical clustering algorithm with the inference mechanism offered by the semantic language OWL-DL. For generation of automatic global merged ontology, we apply the clustering algorithm on different categories of ontological entities (concepts, properties, instances) to find classes of equivalent entities belonging to different local ontologies as shown in Figure 1. For each class, we make inference to discover the new axioms representing the new relationships between entities in the same class or between different classes of the same category. After that, we make use of different classes and axioms to build the global ontology. The methodology starts by aligning the local ontologies by finding similar entities

belonging to different ones. Then, we use the result of the ontology alignment to merge local ontologies automatically. The next sections discuss various aspects of our methodology in detail.

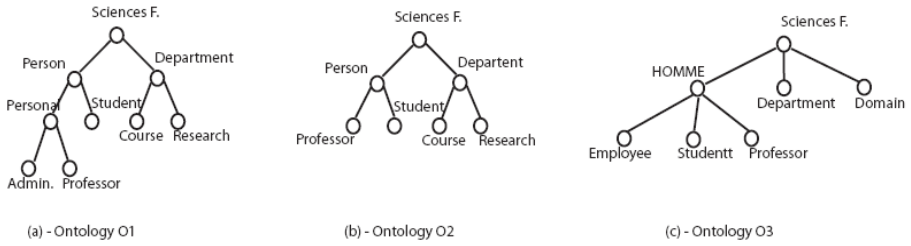

Figure 1: Examples of three local ontologies of the same domain

3.1 Ontology Alignment Strategy

3.1.1 Clustering Algorithm

Ontology. The concept Ontology can be defined with different manners according to its type and use. In our case, we define an ontology as a triplet (*C, R, I*), where *C* is the set of concepts or OWL-classes, *R* is the set of relationships between concepts or OWL-properties and *I* is the set OWL-instances.

Concept. A concept is an attribute vector V_i defined as, $V_i = (T_i, At_1, ..., At_k, P_1, ..., P_j)$ where T_i is the concept term, $At_i(i=1, ..., k)$ are attributes that describe the concept. Finally, Pm (m=1, ...,j) represent concept properties. They can be owl datatype properties or object properties. Concept term and attributes are used to compute the similarity between different concepts.

Similarity measure. Similarity measure allows managing the semantic equivalence or independence between entities. It is based on the concept terminology, properties and its neighborhood. In fact, there is a high probability of semantic equivalence of two concepts which have the same terminology, the same properties and the same relationships with other similar concepts in neighborhood. For computing the similarity between concepts, we must start by computing similarity between different pairs of attributes ($Attribute_i$, $Attribute_j$) where the first attribute belongs to the first concept and the second attribute to the second one. Similarity between two attributes $Attribute_i$ and $Attribute_j$ named $Sim(Attribute_i, Attribute_j)$ is a terminological similarity based on Wordnet thesaurus. Wordnet Java API returns the synonym set (Synset) of a given term, and to find similarity between two terms (attributes) At_1 and At_2, it will be necessary to perform a breadth-first search starting from the Synset of At_1 to the Synsets of Synset of At_2, and so on, until At_2 is found. Once the similarity between different pairs of attributes is computed, we must define a similarity threshold in the order to eliminate all pairs that are not similar and to take only into account those that have a high similarity. Then, attributes similarity measure between two concepts C_i and C_j is calculated as shown in equation 1. We define A, the set of all selected attributes of concepts.

$$Sim_A(C_i, C_j) = \Sigma_{(k=1, ..., Card(A))} \Pi_{ik} Sim(Attribute_{ik}, Attribute_{jm}) \quad (1)$$

Where, $Attribute_{ik}$ (or $Attribute_{jm}$) is the k^{th} attribute of the concept C_i (or C_j). It can be a term, a property or a relationship between this concept and its neighbors. Π_{ik} is the k^{th} attribute weight, which is fixed by the user. Equation (2) and (3) represent the local similarity Sim_L between two concepts and the global similarity Sim_G respectively based on property similarity Sim_p and neighborhood similarity Sim_v . Figure 2 shows the Similarity *GSim* between two concepts with input and output.

$$Sim_L(Ci, Cj) = Sim_T(Ci, Cj) + Sim_A(Ci, Cj) \quad (2)$$

$$Sim_G(Ci, Cj) = Sim_L(Ci, Cj) + Sim_p(Ci, Cj) + Sim_v(Ci, Cj) \quad (3)$$

```
Input:
  C1= V1(T1, \Pi_{11}, At11 ..., \Pi_{1i}, At1i);
  C2= V2(T2, \Pi_{21}, At21, ...,\Pi_{2j}, At2j);
  Similarity threshold S;

Output:
  Similarity GSim between Ci and Cj;
begin GSim := 0
 For each {Attribute k in V1}
       For each {Attribute m in V2}
            Simil := Sim(Attribute_{k}, Attribute_{m});
       IF {Simil > S}
            GSim := GSim+ Simil * Max (\Pi_{k},\Pi_{m});
end.
```

Figure 2: Similarity GSim between two concepts

3.1.2 Hierarchic clustering of ontological entities

The clustering algorithm use different categories of entities (concepts, properties, etc.). We explain here only the case of concepts which is similar for other entities as well. We explain the clustering algorithm that uses the set of concepts and the similarity measure to define synonym concept classes. A synonym concept class is a set, which contains only semantically equivalent concepts. The goal of clustering algorithm is to devise the set *C* of all concepts belonging to all candidate ontologies, to *M* sets of equivalent concepts. For that, clustering algorithm implements the definition of the agglomerative hierarchical clustering mechanism that exploits the similarity measure, which we defined previously to compute the semantic similarity between different pairs of concepts.

Clustering Algorithm Application. The clustering algorithm is based on the use of a similarity matrix in the algorithm of Figure 3. The first row and the first column of the matrix contain the concepts of different ontologies. Each cell in the matrix contains a number that represents the similarity value between the two concepts of the matrix. The first step is to compute the similarity between different pairs of concepts and to load its value in the corresponding cell of the similarity matrix.

$$Sim(SYN_i, C_j) = Min(Sim(C_1, C_j), ..., Sim(C_i, C_j)) \quad (4)$$

After that, the algorithm will search from the maximal value of similarity in the matrix and keep the pair of concepts corresponding to this value. The first class will contain the selected two concepts. The class built will be considered as an element or an individual for the algorithm. For that, it updates the matrix by re-computing the similarity value between the new class and other concepts. The similarity between a

class *SYNi* that contains *j* elements (*C1,... ,Cj*) and an other element *Ck* is defined in equation 4. The algorithm continues it's iterations until it obtains a representative set of classes. The similarity between elements of the same class is maximal and the similarity between different classes is minimal. The result of the algorithm is a set *SYN* that contains *M* sets *SYNi*. Each set *SYNi* contains equivalent concepts belonging to different ontologies. The maximal cardinality of a set *SYNi* is the cardinality of the set *C* of all concepts, and the minimal cardinality is one.

Algorithm 1: Concepts Hierarchic Clustering Algorithm

1: **Input**:
2: O_i (i=1…n): candidate ontologies to be merged
3: C = C_i (i=1... k) / set of concepts of candidate ontologies
4: M_o : set of singleton of C
5: MatrSim[n+m+s , n+m+s] : Similarity matrix;
6: Similarity threshold S;
7: **Output**:
8: Clusters SYN_i of equivalent concepts
9: Initialize the sub matrix M_1[n , n], M_2[n+1-n+m , n+1-n+m] and M_3[n+m+1-n+m+s,n+m+1-n+m+s] of MatrSim with X.

10: FOR (1 <= i <= n+m+s) do
11: FOR (1 <= j <= n+m+s) do
12: IF MatrSim[i, j] <> X Then
13: MatrSim[i, j] ⟵ GSim(C_i, C_j)
14: ENDIF
15: ENDFOR
16: ENDFOR
17: Max ⟵ 0
18: REPEAT
19: FOR (1 <= i <= n+m+s) do
20: FOR (1 <= j <= n+m+s) do
21: IF MatrSim[i, j] > Max Then
22: Max ⟵ Matrsim[i, j]
23: ENDIF
24: ENDFOR
25: ENDFOR
26: IF Max > seuil Then
27: M_i ⟵ M_{i-1} U Ci, C_j - C_i, C_j
28: MatrSim[i, j] ⟵ X
29: ENDIF
30: Update MatrSim by taking into the count the new class
31: Make inference using the new relationship in the new class
32: Until GSim(C_i, C_j) < seuil

Figure 3: Concept Hierarchical Clustering Algorithm

Example. We consider three parts of three different heterogeneous ontologies showed in Figure 1. We start by defining *C*, i.e., the set of all concepts. Then after the application of the algorithm of hierarchical clustering, we obtain *SY N*, the set of nine *SY Ni* as follows.

C *= {SciencesF., Person, Department, Personal, Student, Course, Research, Admin., Professor; sciencesF., person, Departement, Student, Course, Research, Professor, sciencesF., HOMME, Department, Employee, Student, Course, Research, Domain,*

Professor}
SYN *= {{SciencesF., SciencesF., SciencesF.}, {Person, Person, HOMME}, {Department, Department, Department}, {Student, Student, Student}; {Professor, Professor, Professor}, {Employee, personal}, {Course, Course}, {Admin.}, {Domain}}*

3.1.3 Inference

The mechanism of inference is used to discover the implicit relationships between different entities belonging to different classes of different categories. For example, if we have two classes *Person* and *Homme* belonging to two different ontologies, the clustering algorithm merge the two previous classes to build only one. On the other hand, if the class *Person* is an antecedent of an other class *Student*, and at the same time the class *Homme* is an antecedent of an other class *Student* which is similar to the first class *Student*, our inference mechanism detect that the property link the two classes *Person* and *Student* is similar to the property link the two classes *Homme* and *Student*. This is an example of the mechanism of inference that allows extracting all implicit relationships between different entities belonging to different ontologies. It allows to find all relations like *owl*:*SameAs* property, the *owl*:*equivalent* property and the *owl*:*subclassof* property, and similar other entities.

3.2 Our Ontology Merging Strategy

Figure 4 presents our ontology merging methodology comprises of four steps. In the first step, we used the hierarchical clustering algorithm in order to form a set of concept classes. Each class contains the synonym concepts belonging to different ontologies. The result of the algorithm is a set *SY N*, which is composed of *N* subset *SY Ni*. Each subset contains the synonym concepts of different ontologies. The goal of this step is to find all synonym concepts in different ontologies representing the local data sources. To realize this task, we need a similarity measure for computing similarity between the two concepts by taking into account their structure and terminology.

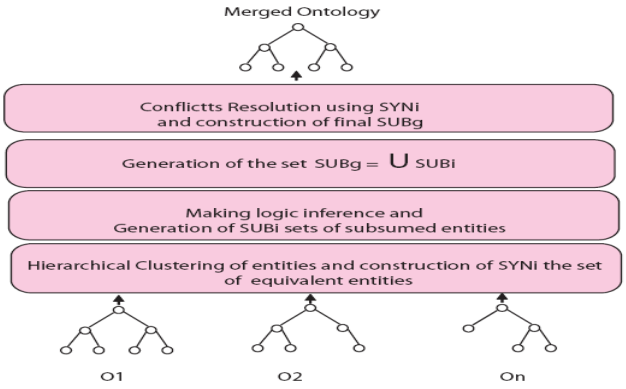

Figure 4: General schema of the ontologies merging approach

Once the concept clusters are built, we pass them to the second step that generates *SUB*, the set of pairs(*Father, Son*) of different ontologies. For this, we start

by generating the set of SUB_i (i = 1, …, P), where each SUB_i corresponds to an ontology O_i. After that, we merge the sets SUB_i to obtain the global set *SUB*. The goal of this step is to keep the hierarchy of different ontologies in order to deduce the merged ontology hierarchy. The third step consists in using classes *SY Ni* to find the final set *SUB* by replacing each concept by the general concept corresponding to the *SY Ni*, which contains it. The goal of this step is to prepare the set *SUB* that allows semantic conflicts resolution. The result of this step is a set of pairs that can be similar or different. In the first case, we eliminate all the instances of the same pair and we keep only one. At the end of this step, we obtain a set *SUB* that contains all pairs of general concepts and the hierarchy of the global ontology. Finally, we use directly *SUB* to build the global ontology. The detail of each step based on similarity measure defined previously is as follows.

Concepts merging and general concepts definition. In this phase, we proceed to assign each class *SY Ni*, a representative concept (term), which can be one of the class element or another concept more general. Before assigning the new term to the class, we must verify that it is not yet affected to another class. In this case, we must change it for the considered class. Like this, we do not solve only the synonymy conflicts, but also homonymy conflicts. After that, we define correspondences tables to save correspondences between the new general concept and class elements. These correspondences will be used for other tasks like query rewriting.
Example. Once we built the set *SY N*, which contains subsets *SY Ni* of synonym concepts, we replace each *SY Ni* by the general concept assigned to the correspondent class. In our example, the set *SY N* becomes as follows.
SY N = {sciencesF., Person,Department, Student, Professor, Personal, Course, Admin., Domain}
The general concept *Cg* attributes are the union of all concepts attributes belonging to the corresponding subset *SY Ni*. The link between the general concept and the class elements will be saved in the correspondences tables.

Generation of the set *SUBg*. This step consists in generating from different ontology hierarchies, the set *SUBg* of pairs $(Ci,Cj) \in Oi$, *where* (i = 1,..,P) and P is the number of ontologies to be merged) where *Ci* is the parent and *Cj* is its child in the ontology hierarchy. The set *SUBg*is used to define the global ontology hierarchy, as follows.
Generation of subsets *SUBi*. The first phase consists in defining subsets *SUBi*, (i = 1,…,P). Each subset *SUBi* corresponds to an ontology *Oi*. The subsets definition is done by a simple traversal of different ontologies hierarchies and we take the node with its child. At the end of this phase, we obtain *P* subsets *SUBi* where each one corresponds to ontology *Oi*. The obtained subsets contain semantic conflicts that we solve in the next step using the set *SY N* generated in the previous step.
Example. : In our example, the three subsets SUB_1, SUB_2 and SUB_3 that correspond to the three ontologies are as below.
SUB_1 = {(*sciencesF., Person*), (*sciencesF.,Department*), (*Person, Personal*), (*Person, Student*), (*Department,Course*), (*Department,Research*), (*Personal, Admin.*), (*Personal, Professor*)}
SUB_2 = {(*sciencesF., Person*); (*sciencesF.,Department*), (*Person, professor*), (*Person, Student*), (*Department, Course*), (*Department, Research*)}

SUB_3 = {(*sciencesF., Homme*); (*sciencesF., Department*), (*sciencesF., Domain*), (*Homme, Personal*), (*Homme, Student*), (*Homme, Professor*)}

SUBi merging. After the generation of the subsets *SUBi*, we merge them to obtain a set *SUB* that contains all pairs of concepts (Parent, Child) belonging to different ontologies to be merged. *SUB* is so defined as below in equation 5.

$$SUBg = U_{i=1,...,P}\ SUBi \qquad (5)$$

The union operation defined in the precedent formulas is the classic union that takes only one instance of the element. But in our case, we can not compare elements coming from different subsets and to find similarity between pairs of concepts. The set *SUB* contains all redundant occurrences of all pairs of concepts belonging to different subsets *SUBi*. For eliminating redundancies, we use the set of synonym concepts *SY N* defined previously to find similar or equivalent pairs.

Example. The union of the three precedent subsets *SUBi* is the set *SUBg* as below.
SUBg = {(*SciencesF., Person*), (*SciencesF.,Department*),(*Person, Personal*), (*Person, Student*), (*Department,Course*), (*Department, Research*), (*Personal, Admin.*); (*Personal,Professor*), (*sciencesF., Person*), (*sciencesF., Department*), (*Person, Professor*), (*Person, Student*), (*Department, Course*), (*Department, Research*), (*sciencesF., HOMME*), (*sciencesF., Department*), (*sciencesF., Domain*), (*Homme, Employee*), (*HOMME, Student*), (*HOMME, Professor*)*}*

Use of *SYNi* to generate *SUBg*. The generated set *SUBg* contains redundant structures, and to eliminate them we use our knowledge store extracted from the concepts population of different ontologies. Knowledge extraction is realized by the application of the clustering algorithm defined in the first step according to the semantic equivalence. The equivalence found between concepts helps us to eliminate redundancies with two steps as follows.

1. Replace concepts in *SUBg* by their general concept. In *SUB*, we traverse all pairs of concepts, one by one and for each component of the pair we find the corresponding class in *SY N* that contains this concept. After that, we replace the pair component by the class name. We do this for all pairs in the set *SUB*. We obtain so a set *SUB*, which contains different pairs that we can compare them each to other.
Example. : *SUBg* in our example becomes as below.
SUBg = {(*sciencesF., Personne*), (*sciencesF.Departement*), (*Personne, Salarie*), (*Personne, Etudiant*), (*Departement, Cours*), (*Departement, Recherche*), (*Salarie, Admin.*), (*Salarie, Enseignant*), (*sciencesF.,Personne*), (*sciencesF.,Departement*), (*Personne, Enseignant*), (*Personne, Etudiant*), (*Departement, Cours*), (*Departement, Recherche*), (*sciencesF., Personne*), (*sciencesF.,Departement*), (*sciencesF., Domaine*), (*Personne, Salarie*), (*Personne, Etudiant*), (*Personne,Enseignant*)*}*

2. Removing redundancies in *SUB*. This phase consists in traverse *SUBg* and comparing pairs of concepts two by two. The similar pairs are removed in order to take into account only one occurrence of the pair in *SUBg*. Finally, these pairs are used to build the hierarchy of global ontology.
Example. : Finally, we obtain set *SUBg* after removing redundant elements as follows.

SUBg = {(sciencesF., Personne), (sciencesF., Departement), (Personne, Salarie), (Personne, Etudiant), (Departement, Cours), (Departement, Recherche), (Salarie, Admin.), (Salarie, Enseignant), (Personne, Enseignant), (sciencesF.,Domaine)}

Merged ontology building. In this step, we use *SUBg* built previously to get the global ontology hierarchy. For that, we start by traversing *SUBg* until finding the concept, which is not the child of any one of other concepts belonging to the same set of concepts. This concept represents the tree root. The second component of the pair represents the first direct child of the selected concept (root). The selected pair will be marked. Then, we seek another pair that has the tree root as first component, the second component of this pair will represent the second child of the tree root. We repeat this search until we have more pairs that contain the concept root. After that, we take the first child of the root concept and we proceed in the same way to find their children in the set *SUBg*. We continue with all concepts in the tree until marking all pairs in *SUBg* to get the merged ontology.

4 Experimental Validation of OMerSec

To validate our approach, we used different variations of the geographic ontology built for the project FoDoMust [Ont, 10]. These variations are showed in the Figure 5a. The basic ontology of geographic objects is composed of 39 concepts, 110 instances and 28 properties and 14 axioms. Figure 5b resumes the ontologies data set statistics.

Ont	Characteristics
1	The basic ontology
2	The hierarchy of concepts is reduced
3	The entities terms are replaced by their synonyms
4	The hierarchy of concepts is reduced differently

	NbClass.	NbProp.	NbInst.	NbAxiom.
Min	24	17	15	12
Max	39	28	110	14
Average	31	22	43	13

Figure 5: Variations made by us in (a), Data set statistics in (b)

Our experiments are performed using *Eclipse* platform with the free reasoner *Pellet* and the framework *Jena*. For computing similarities between concepts, we use the model similarity explained previously. It takes into account the term, attributes, relations and neighbours similarity. The last one makes it recursive. To limit the number of neighbours taken into account, we performed some tests to measure the optimal value of the diameter of neighbourhood or the path length between a concept and its neighbours that find the optimal similarity. In our case, we observed that the optimal value which gives the best similarity is the second neighbour of the concept. The experimental evaluation of our approach is conducted into two steps. First, we aligned with the help of domain experts the different ontologies manually. The mappings found in this step are considered as the alignment reference. The comparison of the alignment reference with the automatic one produce three sets, i.e., *AFound*, *AExpected* and *ACorrect*. The first one represents the entity pairs aligned with the alignment approach. The second one represents the set of entity pairs aligned

in the alignment reference and the third represents the intersection of the two previous sets that are *AFound* and *AExpected*. Using these three sets of entity pairs, we calculated the three following quality measures, i.e., *Precision*, *Recall* and *Fallout*. The *Precision* is the ratio between *ACorrect* and *AFound*, the *Recall* is defined as the ratio between *ACorrect* and *AExpected* and the *Fallout* is defined as the ratio of the difference between *AFound* and *ACorrect*. We used the candidate ontologies to measure the quality of our alignment system OMerSec, and compared our results with two other approaches that are, COMA++ and FCA-Merge. Our approach has shown the best precision results as shown in the Figure 6.

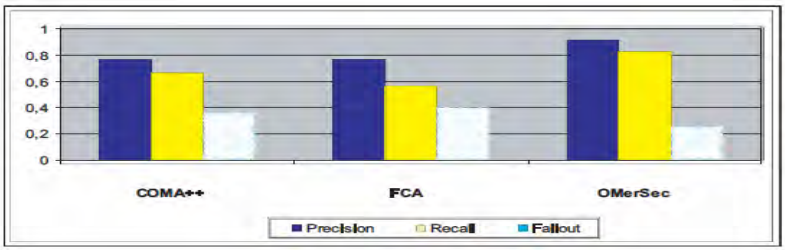

Figure 6: Comparison of our system with others

5 Conclusion and Future Directions

This paper presents the methodology of our automatic ontology merging system that deals in building analysis contexts on-the-fly for data warehouse design by defining a mediation system based on ontologies. Our approach exploits clustering algorithm and an inference mechanism offered by the language OWL. We start by clustering different entities belonging to different local ontologies and making inference on initial axioms to find others which are implicit for the user. After that, we use information in local ontologies to validate those axioms, and build global ontology from them. The proposed methodology benefits in determining and overcoming differences between local ontologies in order to allow the reuse of such ontologies, and the data annotated using these ontologies, throughout different heterogeneous semantic multi-vendor applications. One of our ongoing researches is to apply optimization strategies to enhance the performance of overall system.

References

[Bouquet, 06] Bouquet, P., Serafini, L., Zanobini, S., Sceffer, S.: Bootstrapping semantics on the web: meaning elicitacion from schemas, In Proc. 15th Intl World Wide Web Conference (WWW), 505–512 2006

[Castano, 06] Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications, Journal on Data Semantics, JoDS 3870, Springer Berlin, 25-63 2006

[Doan, 04] Doan, A., Madhaven, J., Domingos, P., Halevy, A.: Ontology matching: A machine learning approach, Handbook on Ontologies in Information Systems, Springer-Verlag, 397-416 2004

[Ehrig, 04] Ehrig, M., Staab, S.: QOM - quick ontology mapping, In Proc. Third International Semantic Web Conference (ISWC2004), Springer, LNCS 3298, 683–696 2004

[Euzenat, 04] Euzenat. J. and Valtchev, P.: Similarity-based ontology alignment in owl-lite, In Proc. 16th ECAI-04, Valencia, Spain, 333–337 2004

[Fahad, 07] Fahad, M., Qadir, M.A., Noshairwan, M.W., Iftakhir, N.: DKP-OM: A Semantic Based Ontology Merger, In Proc. 3rd International Conference I-Semantics 2007, Graz, Austria, 313-322 2007

[Giunchiglia, 04] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: S-Match: an algorithm and implementation of semantic matching, In Proc. 1st European SemanticWeb Symposium, LNCS 3053, 61–75 2004

[Goasdoue, 00] Goasdoue, F., Lattues, V., Rousset, M.C.: The use of carin language and algorithms for information integration: The picsel system, Int. J. Cooperative Inf. Syst., 9(4), 383-401 2000

[Huang, 00] Huang, H.C., Kerridge, J.M., Chen, S.L.: A query mediation approach to interoperability of heterogeneous databases, In Australasian Database Conference, 41-48 2000

[Inmon, 92] Inmon, W.H., Building the Data Warehouse. John Wiley & Sons, Inc., New York, USA, 1992.

[Kalfoglou, 03] Kalfoglou, Y., Schorlemmer, M.: If-map: an ontology mapping method based on information flow theory, Journal of data semantics, Springer Berlin / Heidelberg, LNCS 2800, 98–127 2003

[Kimball, 98] Kimball, R.: The operational data warehouse, DBMS, 11(1), 14-16 1998.

[Klein, 01] Klein, M.: Combining and relating ontologies: an analysis of problems and solution, In Proc. Workshop on Ontologies and Information Sharing (IJCAI-01), Seattle, USA, 53-62 2001

[Lamarre, 04] Lamarre, P., Cazalens, S., Lemp, S., Valduriez, P.: A flexible mediation process for large distributed information systems, In CoopIS/DOA/ODBASE (1), LNCS vol. 3290, Springer, 19-36 2004

[Maiz, 07] Maiz, N., Bentayeb, F., Boussaid, O.: Ontology based mediation system. In Proc. 18th Information Resource Management Association International Conference (IRMA 07), Canada, May 2007

[Mascardi, 10] Mascardi, V., Locoro, A., Rosso, P.: Automatic Ontology Matching Via Upper Ontologies: A Systematic Evaluation. IEEE Transaction on Knowledge and Data Engineering. Vol. 2 (5), pp. 609-623. 2010.

[McGuinness, 00] McGuinness, D., Fikes, L., Rice, J., Wilder, S.: An environment for merging and testing large ontologies, In Proc. 7th Intl. Conference on Principles of Knowledge Representation and Reasoning, Breckenridge, CO, USA, 483–493 2000

[Mena, 00] Mena, E., Illarramendi, A., Kashyap, V., Sheth, A., P.: OBSERVER: An approach for query processing in global information systems based on interoperation across preexisting ontologies, International Journal DAPD, 8(2), 223-271 2000

[Noy, 03] Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. IJHCS, Elsevier, vol. 59(6), 983–1024 2003

[Ont, 10] http://lsiit-old.u-strasbg.fr/afd/sites/fodomust/fr-accueil.php