Establishing Global Ontology by Matching and Merging

Susan F. Ellakwa[1], Passent M. El-Kafrawy[2], Mohamed Amin[2], and El Sayed ElAzhary[1]

[1] Central Lab for Agricultural Expert Systems (CLAES), ARC, Giza, Egypt
fisalsusan@yahoo.com, sayed@claes.sci.eg
[2] Mathematics and CS Department, Faculty of Science, Menoufia University, Egypt
passentmk@gmail.com

Abstract. Ontology is used for communication between people and organizations by providing a common terminology over a domain. This work presents a system of establishing global ontology from existing ontologies. Establishing ontology from scratch is hard and expensive. This work establishes ontology by matching and merging existing ontologies. Ontologies can be matched and merged to produce a single integrated ontology. Integrated ontology has consistent and coherent information rather than using multiple ontologies, which may be heterogeneous and inconsistent. Heterogeneity between different ontologies in the same domain is the primary obstacle for interoperation between systems. Heterogeneity leads to the absence of a standard terminology for any given domain that may cause problems when an agent, service, or application uses information from two different ontologies. Integrating ontologies is a very important process to enable applications, agents and services to communicate and understand each other.

Keywords: Artificial Intelligence, Knowledge Representation, Ontology, Matching, Merging.

1 Introduction

The term ontology refers to a wide range of formal representations, including taxonomies, hierarchical terminology vocabularies or detailed logical theories describing a domain [1]. One commonly used definition is based on the original use of the term in philosophy, where ontology is a systematic account of Existence. For artificial intelligence (AI) systems, what "exists" is that what can be represented [2]. "An Ontology is a formal, explicit specification of a shared conceptualization [3]. *Conceptualization* refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. *Explicit* means that the type of concepts used, and the constraints on their use, are explicitly defined. *Formal* refers to the fact that the ontology should be machine-readable. *Shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group. This paper presents a system to establish

V.V. Das, E. Ariwa, and S.B. Rahayu (Eds.): SPIT 2011, LNICST 62, pp. 76–83, 2012.

dynamic global ontology in specific domain from existing ontologies by matching and merging. Global ontology allows users to avoid querying the local ontologies one by one, and to obtain a result from them just by querying a global ontology. Global ontology has standard and shared terminology. It is consistent and coherent. It has no redundancy. There are a large variety of languages for expressing ontologies. Fortunately, most of these languages share the same kinds of entities, often with different names but comparable interpretations. Source ontologies in the proposed system have been expressed in XML language. Ontology language in the proposed system deal with the following kinds of entities: Concepts, properties, and values according to Common KADS Methodology [4]. In this system, we introduce an ontology matching and merging problem and propose a solution technique called Multi-Matching and Merging Algorithm (MMMA) (table 1.a,1.b), which uses a multi search algorithm to find the correspondences between entities in the input ontologies and to merge these ontologies. An important feature of this technique is that it benefits from existing individual match methods and combines their results to provide enhanced ontology matching. This system proposes a new technique in matching; it performs three iterations, each iteration manipulates one type of entities. The first iteration manipulates the concepts, while the second iteration handles the properties, and the third iteration handles the values. In each iteration, the system uses hybrid matchers which are combined in a sequential composition. This multilevel decomposition reduces redundancy alignments and speeds up the system's final alignments. The system uses different kinds of matchers to cover different kinds of alignments to reduce redundant entities of resulted merged ontology. Using variety of matchers solve the string and language matching problem. This system extracts entities in two ontologies which have same string or same meaning. The system uses thresholds to reduce useless alignments and involves user to confirm alignments. This system can merge the ontologies in hierarchy structure. This paper consists of five sections; first section is introduction, second section shows definition for matching and merging, third section introduces related work, fourth section presents the proposed system and fifth section is conclusion and future work.

2 Ontology Matching and Merging

Matching is the process of finding relationships or correspondences between entities of different ontologies. Alignment is a set of correspondences between two or more (in case of multiple matching) ontologies. The alignment is the output of the matching.

The matching process can be seen as a function f which, from a pair of ontologies to match o and o', an input alignment A, a set of parameters p and a set of oracles and resources r, returns an alignment A' between these ontologies: A'=f (o, o', A, p, r)

The proposed system uses the matching techniques; string-based technique [5] (String equality method, Substring method and Prefix/suffix method) and language-based technique [5] (tokenization method, Stopword elimination method and WordNet [6] method) as blocks on which a matching solution is built. Each of these

methods is called a matcher. Each matcher gives its similarity. Once the similarity between ontology entities is available, the alignment remains to be computed.

Merging is a first natural use of ontology matching, it consists of obtaining a new ontology o'' from two matched ontologies o and o' so that the matched entities in o and o' are related by the alignment. Merging can be presented as the following operator: Merge (o, o', A') = o''

When the ontologies are expressed in the same language, merging often involves putting the ontologies together and generating bridge or articulation axioms. Merging does not usually require a total alignment: those entities which have no corresponding entity in the other ontology will remain unchanged in the merged ontology. Ontology merging is especially used when it is necessary to carry out reasoning involving several ontologies. It is also used when editing ontologies in order to create ontologies tailored for a particular application.

3 Related Work

Several tools exist for ontology establishment, ranging from fully manual to fully automated. Many of the semi-automated ontology merging and matching tools are listed in this section. PROMPT [7] begins with the linguistic-similarity matches for the initial comparison, but generates a list of suggestions for the user based on linguistic and structural knowledge and then points the user to possible effects of these changes. OntoMorph [8] provides a powerful rule language for specifying mappings, and facilitates ontology merging and the rapid generation of knowledge-base translators. It combines two powerful mechanisms for knowledge-base transformations such as syntactic rewriting and semantic rewriting. Syntactic rewriting is done through pattern-directed rewrite rules for sentence-level transformation based on pattern matching. Semantic rewriting is done through semantic models and logical inference.

4 System for Establishing Global Ontology

This section presents a new semi-automated system for establishing global ontology by merging pre-existing ontologies. This technique consists of two main components: matching process and merging process.

4.1 System Structure

The structure of the two main components, matching process and merging process, are shown in fig.1. Ontology matching tries to identify similarities between heterogeneous ontologies and to automatically create suitable mappings for merging. Matching is an essential aspect of merging and could also be used to initiate merging. Ontology merging is the process that will create a single global coherent ontology by unifying two or more existing ontologies.

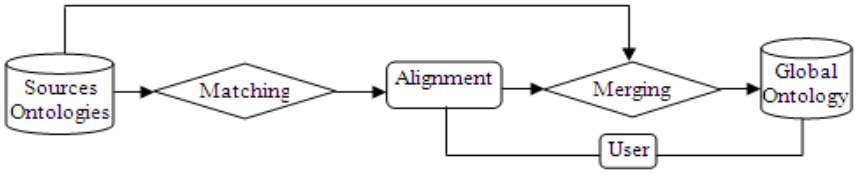

Fig. 1. Framework for establishing global ontology

4.2 System Components

As mentioned before the system is composed of two main components: matching process is shown in fig.2 and merging process is shown in fig.3

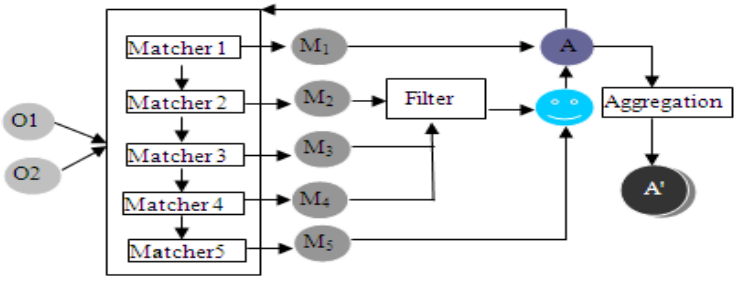

Fig. 2. Matching process

Matching Process: The previous matchers are the building blocks on which the matching solution is built. Once the similarities between ontology entities are available, the alignment can be computed. Matching strategy is built by organizing the combination matchers, aggregating the results of matchers (basic methods) in order to compute the compound similarity between entities, involving users in the system and extracting the alignments from the resulting similarity.

Matcher composition is a global method to combine local methods (or basic matchers) in order to define the matching algorithm. A way of composing matchers in the proposed system uses sequential composition. In sequential composition, combination of matchers is more classically used to improve an alignment. In the proposed system, it consists of five matchers; each matcher extracts additional alignment without redundancy, the input of each matcher depends on the output of the previous matcher. The inputs of the system are two ontologies o1, o2 and initial alignment A. Entities of source ontology are concepts C, properties P and values V. The input of a matcher is the matched entities of the last matcher and the unmatched entities. The matched entities are to be aggregated in final alignment A'. This cycle performs three times; first iteration for extracting matched concepts, second iteration for extracting matched properties of the matched concepts and third iteration for extracting matched values of the matched properties. In each iteration, all matchers

are sequentially applied to entities. First matcher (Matcher1) based on equality string method, it searches for identical terms, the output is M1 (similarity matrix). Second matcher (Matcher2) based on substring method. The input of this matcher is the unmatched entities of previous matcher, the output is M2. M2 should be filtered according to a threshold, it should be determined by the system or the user, and then the user discards the unaccepted correspondences. Third matcher (Matcher3) based on prefix method. The input of this matcher is unmatched entities of previous matchers, the output is the M3. M3 should be filtered according to the pre-determined threshold, and then the user discards the unaccepted correspondences. Fourth matcher (Matcher4) is based on suffix method. The input of this matcher is unmatched entities of previous matchers, the output is the M4. M4 should be filtered according to the pre-determined threshold, and then the user discards the unaccepted correspondences. Fifth matcher (Matcher5) based on WordNet method; it searches for terms which have the same meaning. The input of this matcher is unmatched entities of previous matchers, the output is the M5. M5 can be filtered by the user. This matcher uses tokenization method and stopword elimination method. The output of the five matchers in the first iteration is matched concepts which aggregated in A (initial alignment) to be the input of the second iteration. The output of matchers in second iteration is the matched properties which aggregated in A (initial alignment) to be the input of the third iteration. The output of matchers in third iteration is matched values. Matched concepts, Matched properties, Matched values are aggregated in A'(final alignment).

Merging Process: Consists of five operations (fig.3): *Determine unmatched entities, Select concepts, Merge hierarchical classification, Collect properties and Collect values.* The input of this process is the source ontologies o1, o2 besides the output of the matching process A'. The output is the merged ontology o'. *Determine unmatched entities* operation identifies unmatched concepts C' and its properties P' and its values V'. *Select concepts* operation selects a concept from its correspondence. *Merge hierarchical classification* determines concept location in the hierarchy structure. *Collect properties* determines properties of the selected concept from its correspondence. *Collect values* determines values of a property from its correspondence. The output of the system is the merged ontology of two source ontologies o1, o2. The system can merge more than two ontologies by matching and merging two ontologies and the output can be matched and merged with the another ontology, and so on.

In this paper we studied different matching techniques; we presented a novel framework to support matching and merging. We discussed different matchers to ontology matching. To obtain better quality matching results, we extended the multi-matching strategy by introducing a multi-level matching strategy, each matcher introduces new alignment based on its method and the system collects these alignments. This system can manipulate small and large ontologies, it manipulates ontologies in hierarchy structure, and the merged ontology has no redundancy and no inconsistency.

Table 1.a. Matching part of Multi-Matching and Merging Algorithm (MMMA)

```
/*Matching*/
/*Matching Concepts*/
List of Ontologies [o1, o2]
List of concepts (LC1) of o1 [c1, c2… cn]
List of concepts (LC2) of o2 [c1, c2… cm]
Number of Matchers = 5
List of concept alignments is A
A = [ ], L = n, W = m, Mat = 0
Repeat
Mat = Mat + 1, I = 0
   Repeat
   I = I + 1
   Select concept cI of o1
   J = 0
   K = 0
     Repeat
     J= J + 1
     Select concept cJ of o2
     IF match (cI, cJ)
     THEN {A= [(cI, cJ|A],K=1, L = L – 1,
     W = W – 1,
     LC1 = SUBSTRACT (LC1, cI),
     LC2 = Subtract (LC2, cJ)}
     Until J = W OR K = 1
   Until I = L
Until mat=5
/*Matching Properties*/
A1 = A, A2 = [ ], Mat = 0
Repeat
Mat=Mat+1
    Repeat
    A1 = [H | Tail]
    H = (c1, c2)
    Get PI of c1
/* PI is the list of properties of c1 from o1*/
    Get PJ of c2
/* PJ is the list of properties of c2 from o2*/
        Repeat
        PI = [HPI| T1]
        PJ = [HPJ | T2]
        If   match (HPI, HPJ)
        T HEN {A2 = [[(c1, HPI), (c2, HPJ)]
        A2], PI = [T1],  PJ = [T2]}
        Else PJ = [T2]
        IF PJ = [ ] THEN PI = [T1]
        Until PI = [ ]
    A1 = [Tail]
    Until A1 = [ ]
Until mat=5
/*Matching Values*/
A3 = A2, A4 = [ ], Mat = 0
Repeat
Mat = Mat + 1
    Repeat
    A3 = [H | Tail]
    H = [(C1, P1), (C2, P2)]
    Get VI of P1/*VI: list of values of P1 */
    Get VJ of P2/*VJ:list of values of P2 */
        Repeat
        VI = [HVI| T1], VJ = [HVJ | T2]
        IF match (HVI, HVJ) THEN {A4 =
        [[(C1,P1, HVI), (C2, P2, HVJ)] |
        A4],
        VI = [T1], VJ = [T2]}
        Else VJ = [T2]
        IF VJ = [ ]  THEN VI = [T1]
        Until VI = [ ]
    A3 = [Tail]
    Until A3 = [ ]
Until Mat = 5
A'=append (A, A2, A4)
/*Matching Function*/
A, B, C, X, Yare strings
σ1(X,Y) = (2 * |X|) / (|X|+|Y|)
σ2(X,Y)= |C| / (|X|+|Y|)    /*C is the longest
common string for X and Y */
T is the threshold of similarity
Function:  match(X, Y)
{Case Mat=1: /* matcher1 is executed*/
IF σ1 (X, Y) = 1
THEN match(X, Y) = TRUE
Case Mat=2: /* matcher2  is executed*/
IF (X⊂Y  AND σ 1(X, Y) >=  T  AND
correspondence of X, Y is accepted from
user)
THEN match(X, Y) = TRUE
Case Mat =3: /* matcher3 is executed*/
IF (X= conc(C, A) AND Y= conc(C, B),
σ2(X, Y) >= T, AND correspondence of X,
Y is accepted from user)
THEN match(X, Y) = TRUE
Case Mat =4: /* matcher4 is executed*/
IF (X= conc(A, C) AND Y= conc (B, C),
σ2(X, Y) >= T, AND correspondence of X,
Y is accepted from user)
THEN match(X, Y) = TRUE
Case Mat =5: /* matcher5 is executed*/
IF (WordNet(X, Y), correspondence of X, Y
is accepted from user)
THEN match(X, Y) = TRUE }
/* WordNet(X, Y) means that X and Y are
synonyms */
```

Table 1.b. Merging part of Multi-Matching and Merging Algorithm (MMMA)

```
/ * Merging */
List of concept alignments is A
List of property alignments is A2
LC = [ ], LC2= [ ]
  /* Select Concepts */
Repeat
A = [(C1, C2)|Tail], LC = [C1, LC],
A = [Tail]
Until A = [ ]
Selected Concepts = LC
  /* Unmatched Concepts */
Get concepts C O1 of o1
Get concepts C O2 of o2
Unmatched concepts of o1 (UCO1) =
CO1 DIFFERENCE LC
Repeat
A = [(C1, C2)|Tail],
LC2 = [C2, LC2],
A = [Tail]
Until A = [ ]
Matched Concepts of o2 = LC2
Unmatched concepts of o2 (UCO2) =
 CO2 DIFFERENCE LC2
C' = UCO1 ∪ UCO2
  /* Collect Properties */
I = 0, LCP = [ ]
Repeat
A2 = [[(C1, P1), (C2, P2)] | Tail]
X = C1
IF I > 0 AND X < > C
THEN OldP= P/*P set of properties of c2*/
IF I = 0 OR X < > C
THEN {Get P of C2 from o2,
        SUBSTRACT (P, P2, PY), P = PY}
IF (I > 0 AND X = C)
THEN {SUBSTRACT (P, P2, PY),
P = PY}
IF (I > 0 AND X < > C) THEN {Get Prop
of C from o1, A5 = Prop of C ∪OldP}
I = I + 1, C = C1, A2 = [Tail]
IF A2 = [ ]
THEN {Get Prop of C from o1,
        A5 = Prop of C∪P,
        LCP = [(C, A5) | LCP] }
Until A2 = [ ]
  /* Collect Values */
LCPV = [ ], LPV = [ ], I = 0
Repeat
A4 = [[(C1, P1, V1), (C2, P2, V2)] | Tail]
Z = C1
IF (I > 0 AND Z < > C)
THEN {LCPV = [(C, LPV) | LCPV]
        LPV = [ ], I = 0}
X = P1
IF (I > 0 AND X < > P)
THEN OldV = V
IF (I = 0 OR X < > P)
THEN {Get Val of P2 from o2,
SUBSTRACT (V, V2, VY),V = VY}
IF (I > 0 AND X = P)
THEN {SUBSTRACT (V, V2, VY),
V = VY}
IF (I > 0 AND X < > P)
THEN {Get Val of P from o1, A6 = Val of
P ∪ OldV, LPV = [(P, A6) | LPV]}
I = I + 1, P = P1, A4 = [Tail]
IF A4 = [ ]
THEN{Get Val of P from o1, A6 = Val of
P ∪ V, LPV = [(P, A6) | LPV]}
C = C1
Until A4 = [ ]
  /* Merge Hierarchical Classification*/
Two ontologies o1, o2
Offspring of o1 is Co1
  /* Co1 is a list of concepts */
Offspring of o2 is Co2
A is the alignment concepts of o1, o2
 /* A is a list of matched concepts */
Co11 = Co1
Repeat
Co1 = [H | Tail]
IF match (H, C)
/*H is a concept in o1,C is a concept in o2
*/
THEN {Get Offspring OC of C,
        Link OC with H,
        Co11 = [Co11| OC],
        Co2 = Subtract(C, Co2),
        Co2 = Subtract (OC, Co2)}
Co1 = [Tail]
Until Co1 = [ ] OR Co2 = [ ]
Repeat
Co11 = [H | Tail]
IF match (H, C)
 /* H, C are two concepts in Co11 of o1*/
THEN {Get Offspring OC of C,
        Link OC with H,
        [Tail] = Subtract(C, [Tail])}
Co11 = [Tail]
Until Co11 = [ ]
```

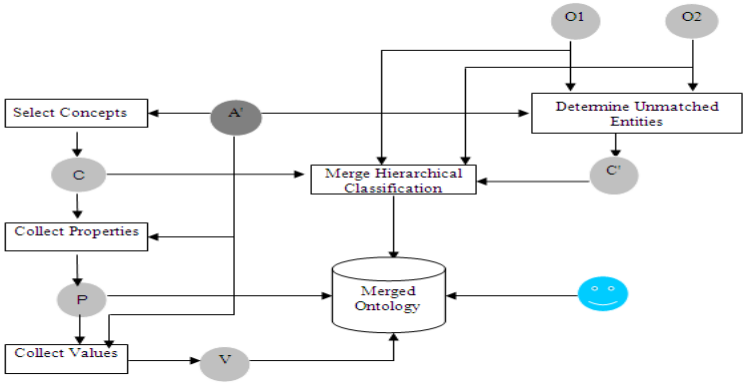

Fig. 3. Merging process

5 Conclusions and Future Work

This paper presents a system to build a global ontology from different ontologies in the same domain. This work presents Multi-Matching and Merging Algorithm (MMMA) for reusing and sharing existing ontologies by matching and merging. We are working on implementation currently now. The system will have graphical user interface to allow browsing to get ontologies to be matched and merged. It allows user to confirm alignments, edits source ontologies, edits merged ontology and gives information about ontologies and their entities.

References

1. Noy, N., Klein, M.: Ontotlogy Evolution: Not the Same as Schema Evolution. Knowledge and Information Systems 6(4), 428–440 (2004); also available as SMI technical report SMI-2002-0926
2. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–220 (1993)
3. Borst, P., Akkermans, H., Top, J.: Engineering ontologies. International Journal of Human-Computer Studies 46, 365–406 (1997)
4. Wielinga, B.J.: Expertise Model Definition Document. University of Amsterdam (1994)
5. Euzenat, J., Shvaiko, P.: Ontology matching, 333 p. Springer, Heidelberg (2007)
6. Pedersen, T., Patwardhan, S., Patwardhan, S.: WordNet:Similarity – Measuring the Relatedness of Concepts. In: Proc. of 19th National Conf. on AI, San Jose (2004)
7. Noy, N., Musen, M.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: Proc. of 17th National Conference on Artificial Intelligence (AAAI), Austin, Texas, pp. 450–455 (2000)
8. Chalupsky, H.: Ontomorph: A Translation System for Symbolic Knowledge. Principles of Knowledge Representation and Reasoning (2000)