Integrated Ontology for Agricultural Domain

Susan F. Ellakwa
Central Lab for Agricultural Expert Systems (CLAES), ARC, Giza, Egypt

El-Sayed El-Azhary
Central Lab for Agricultural Expert Systems (CLAES), ARC, Giza, Egypt

Passent El-Kafrawy
Mathematics and Computer Science Department, Faculty of Science,Menoufia University, Egypt

ABSTRACT
Ontologies provide a shared and common understanding of a domain that can be communicated between people and across application systems. An ontology for a certain domain can be created from scratch or by merging existing ontologies in the same domain. Establishing ontology from scratch is hard and expensive. Multiple ontologies of different systems for the same domain may be dissimilar, thus, various parties with different ontologies do not fully understand each other in spite of these ontologies are for the same domain. To solve this problem, it is necessary to integrate these ontologies. Integrated ontology, should be consistent and has no redundancy. This work presents a semi-automated system for building an integrated ontology by matching and merging existing ontologies. The proposed system has been applied on the agricultural domain for Faba Bean crop to get a dynamic integrated ontology, it can be applied also on all crops whatever field crops or horticulture crops. Source ontologies in the proposed system have been implemented in XML language. CommonKADS Methodology has been used in building the target ontology. CommonKADS Methodology deals with the following kinds of entities: Concepts, properties, and values. The proposed system proposed a technique to solve the matching and merging problems by using a multi-matching technique to find the correspondences between entities in the source ontologies and merging technique which deals with concepts, properties, values and hierarchical classifications. The outcome of the proposed system is an integrated ontology in hierarchical classification of the concepts.

Keywords
Artificial intelligence; knowledge representation; ontology; matching; merging

1. INTRODUCTION
An Ontology is a formal, explicit specification of a shared conceptualization [1]. There are several reasons for developing ontology, first of all, sharing common understanding of the structure of information among people or software agents. The second reason is to enable reuse of knowledge. The third reason is to make domain assumptions explicit. Fourth reason is to separate domain knowledge from the operational knowledge. Fifth reason is to analyze domain knowledge. Sixth reason is to increase interoperability among various domain of knowledge. Seventh reason is to enhance scalability of new knowledge into the existing domain. Finally, searching and reasoning a specific knowledge in a domain knowledge.

The starting point for creating ontology could arise from different situations. An ontology can be created from scratch; from existing ontologies; from a corpus of information sources; or a combination of the latter two approaches.

Multiple ontologies need to be accessed from different systems; these ontologies are dissimilar for the same or overlapping domains, thus, various parties with different ontologies do not fully understand each other. To solve these problems, it is necessary to use integrating ontologies. Ontology integration aims to building ontologies from other ontologies to get integrated ontology. Integrated ontology should be consistent, coherent and has no redundancy

Establishing ontology from scratch is hard and expensive. This work presents a semi-automated system for building integrated ontology by matching and merging existing ontologies. The proposed system has been applied on agricultural domain to get a dynamic integrated ontology. Dynamic means that: the integrated ontology can be modified by adding, deleting, or editing some terms when needed. The proposed system 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 five matchers (exact method, substring method, prefix method, suffix method, wordnet method) sequentially to cover different kinds of alignments (matching entities) and to make the integrated ontology perfect and has no redundancy. The system uses thresholds in substring, prefix, and suffix methods to reduce useless correspondences and involves user to confirm alignments.

2. 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 [2] 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 [3] 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.

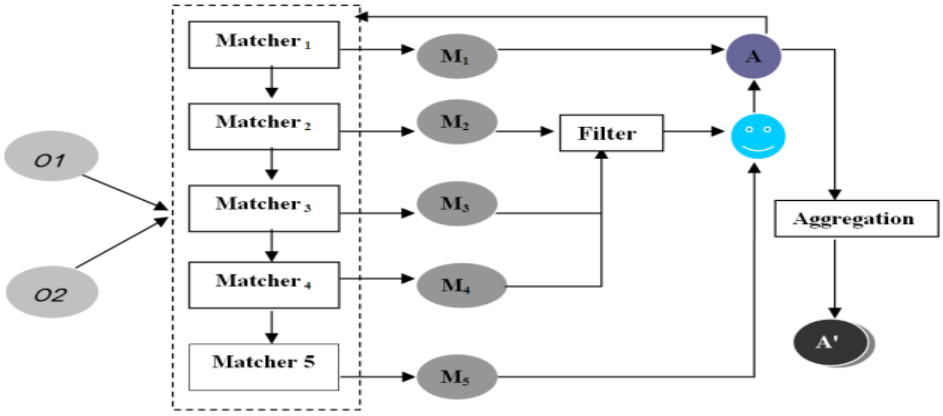

Fig.1: Matching Process

Semantic rewriting is done through semantic models and logical inference. A concept hierarchy management for ontology alignment and merging is provided in Hierarchical Concept Alignment system (HICAL) [4], where one concept hierarchy is aligned with another concept in another concept hierarchy. HICAL uses a machine-learning method for aligning multiple concept hierarchies, and exploits the data instances in the overlap between the two taxonomies to infer mappings. It uses hierarchies for categorization and syntactical information, not similarity between words, so that it is capable of categorizing different words under the same concept. Another system that employs machine learning techniques to find ontology mappings is GLUE [5]. If given two ontologies, for each concept in one of the ontologies, GLUE finds the most similar concept in the other one. GLUE works with several similarity measures that are defined with probabilistic definitions. Multiple learning strategies exploit different types of information from instances or taxonomy structures. GLUE can also use common sense knowledge and domain constraints instead of relaxation labeling. It is a well-known constraint optimization technique adapted to work efficiently. Quick Ontology Mapping (QOM) [6] is based on the hypothesis that mapping algorithms can be streamlined so that the loss of quality is marginal, but the improvement of efficiency is tremendous for the ad-hoc mapping of large-size light-weight ontologies. A generic ontology mapping system, called LILY [7], is based on the extraction of semantic
subgraph. LILY exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. After that, a subsequent similarity propagation strategy is applied to produce more alignments if necessary.
Finally, LILY uses the classic image threshold selection algorithm to automatically select the threshold, and then extracts final results based on the stable marriage strategy. LILY has different functions for different kinds of tasks: for example, Generic Ontology Matching method (GOM) is used for common matching tasks with small size ontologies; Large scale Ontology Matching method (LOM) is used for matching tasks with large size ontologies; and Semantic Ontology Matching method (SOM) is used for discovering the semantic relations between ontologies. The two limitations of LILY are that it requests the user to manually set the size of subgraph according to different mapping tasks and the efficiency of semantic subgraph is very low in large-scale ontologies.
DKP-AOM [8] is a system that exploits linguistic, synonym and axiomatic matching to find correspondences
between concepts. In addition, it employs test criterion for the detection of semantic inconsistencies that originates when concepts contradict according to their subsumption or disjointness in local ontologies criterion. In this way, it detects explication and conceptualization mismatches between heterogeneous ontologies and promotes a larger pool of knowledge and information to be integrated to facilitate new reliable communication and reuse.
The RiMOM [9] system integrates multiple strategies, such as edit distance, statistical learning, and three similarity propagation-based strategies. Then, it applies a strategy selection method in order to decide on which strategy it will rely more. As a result, RiMOM combines the conducted alignment. RiMOM offers three possible structural propagation strategies: concept-to-concept propagation strategy (CCP), property-to-property propagation strategy (PPP), and concept-to-property propagation strategy (CPP). To choose between them, RiMOM uses heuristic rules. For example, if the structure similarity factor is lower than some threshold, then RiMOM does not use the CCP and PPP strategies, but uses CPP. The basic idea of CCP, PPP, and CCP is to propagate the similarities of (concept pairs or property pairs) across the concept/property hierarchy structure. For instance, in CCP, similarities of concept pairs are propagated across the concept hierarchy structure.
In [10] shows that recent studies on ontology merging show that due to conceptualization and mismatches between local ontologies, fully automatic merging is unattainable
oMap [11] deploys a number of matchers in order to find the correspondences between entities of the input ontologies. The matchers include a string similarity measure, learning methods used on instance data, and a matcher that propagates preliminary weights through the ontology constructors used in the definitions of ontology entities. At the end, the results are aggregated using a weighted average.
H-Match [12] takes OWL ontologies as its input. Internally, these input ontologies are represented by graphs using the H-model representation. Moreover, H-Match computes two types of similarities: linguistic and contextual. These are then combined using weighting schemas to yield a final measure, called semantic similarity. In determining the contextual similarity, H-match considers neighboring concepts, e.g., linked through the taxonomy of the actual concept

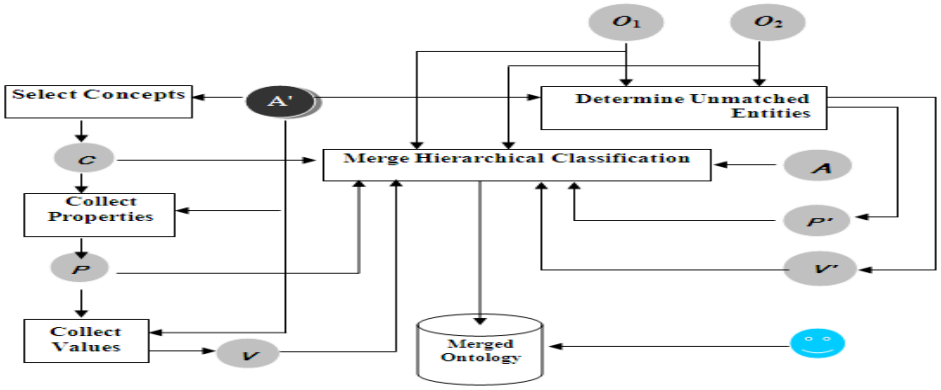

Fig.2: Merging Process

.

Our system consists of two techniques: matching and merging, matching consists of five matchers, each matcher manipulates kind of heterogeneity between two ontologies to detect inconsistency. These matchers executes sequentially to reduce redundancy and speed up system execution time, user is involved to confirm mappings (correspondences) between two ontologies and user can select concepts, properties and values from correspondences. Threshold is involved to reduce unmeaning mappings. Merging is fully automatic process. It handles hierarchy classifications of two ontologies effectively to get merged ontology with new hierarchy classification.

3. THE PROPOSED SYSTEM

This section presents a system for establishing dynamic ontology from two or more existing ontologies in the same domain. The end users of the system may be experts or specialists in the domain. This system introduces matching and merging technique, which uses a multi-level search to find the correspondences between entities in the input ontologies. An important feature of this technique is that it uses several match methods sequentially and combines their results. There are a large variety of languages for expressing ontologies [13]. Fortunately, most of these languages share the same kinds of entities, often with different names but comparable interpretations. In the proposed system, source ontologies have to be expressed in XML language. Ontology language in the proposed system deals with Concepts, properties, and values.

3.1 Matching Process

Fig. 1describes matching process. The 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 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 extract 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 unmatched entities of the last matcher. 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. Each iteration, all matchers are sequentially applied to entities. First matcher (matcher1) based on exact 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, the threshold 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).

3.2 Merging Process

Fig. 2 describes merging process, it consists of five operations: Determine unmatched entities, Select concepts, Merge hierarchical classification, Collect properties and Collect values.

The input of this process is the two source ontologies o1, o2, the alignment A' (the output of the matching process), A (the matched concepts). The output is the merged ontology. *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.

4. MULTI-MATCHING and MERGING AlGORITHM (MMMA)

This section presents MMMA (Multi-Matching and Merging Algorithm) in detail which describe matching and merging process in the proposed system.

Matching Algorithm consists of three parts: Matching concepts (Fig. 3), Matching properties (Fig. 4), and Matching values (Fig. 5).

Merging Algorithm consists of five parts: Select concepts (Fig. 6), Collect properties (Fig. 7), Collect Values (fig 8), Unmatched entities (Fig. 9), and Merge hierarchical classification (Fig. 10))

```
Input Ontologies is o1, o2
LC1 is the List of concepts of o1 [c1, c2… cn]
LC2 is the List of concepts of o2 [c1, c2… cm]
Number of Matchers = 5
A is the List of concept alignments
A = [ ]
L = n
W = m
Mat = 0
Repeat
Mat = Mat + 1
I = 0
    Repeat
     I = I + 1
     Select concept cI of LC1
     J = 0
     K = 0
            Repeat
            J= J + 1
            Select concept cJ of LC2
            IF match (cI, cJ)
            THEN {A= [(cI, cJ)|A],
            K=1,
            W = W – 1,
            LC1 = SUBTRACT (LC1, cI),
            LC2 = SUBTRACT (LC2, cJ)}
           Until J = W  OR  K = 1
    Until I = L OR W = 0
Until mat=5
```

Fig.3: Matching Concepts

```
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*/
       PJJ = PJ
          Repeat
          PI = [HPI| T1]
          PJ = PJJ
             Repeat
              K = 0
              PJ = [HPJ | T2]
                IF   match (HPI, HPJ)
                   THEN {A2 = [[(c1, HPI), (c2, HPJ)] | A2],
                    PJJ = DIFFERENCE(PJJ, HPJ),  K = K + 1},
                        PJ = [T2]
                         Until K = 1 OR PJ = [ ]
             PI = [T1]
              Until PI = [ ] OR PJJ = [ ]
       A1 = [Tail]
      Until A1 = [ ]
Until mat=5
Get A2
```

Fig. 4: Matching Properties

```
A3 = A2
A4 = [ ]
Mat = 0
Repeat
Mat = Mat + 1
      Repeat
      A3 = [H | Tail]
      H = [(C1, P1), (C2, P2)]
      Get VI of P1                /* VI is the list of values of P1 */
      Get VJ of P2                /* VJ is the list of values of P2 */
      VJJ = VJ
         Repeat
         VI = [HVI| T1],
          VJ = VJJ,
           k=0
            Repeat
             VJ=[HVJ|T2]
             IF match(HVI,HVJ)
             THEN {A4 = [[(C1, P1, HVI), (C2, P2, HVJ)] | A4],
                      VJJ = DIFFERENCE(VJJ, HVJ)  ,k=1}
                VJ=[T2]
                UNTIL VJ=[ ] OR k=1
             VI= [T1]
            UNTIL V1 = [ ] OR VJJ = [ ]
     A3 = [Tail]
     Until A3 = [ ]
Until Mat = 5
A'= A4
```

Fig.5: Matching Values

```
LC1 = [ ]        /* selected concepts of o1 */
LC2 = [ ]        /* selected concepts of o2 */
B = A'
Repeat
B = [[(C1, P1, V1), (C2 ,P2, V2)]|Tail],
IF user select C1
THEN
      IF C1 ∉ LC1
     THEN  LC1 = [C1, LC1]
IF user select C2
THEN
      IF C2 ∉ LC2
     THEN  LC2 = [C2, LC2]
B = [Tail]
Until B = [ ]
```

Fig.6: Select Concepts

```
LP1 = [ ]              /* properties of selected concepts of o1 */
LP2 = [ ]              /* properties of selected concepts of o2 */
Repeat
LC1 = [C|Tail]
Get list of selected properties of C (LP)    /*properties of C
selected by user */
LP1 = [(C, LP)|LP1],
LC1 = Tail,
Until LC1 = [ ]
Repeat
LC2 = [C|Tail]
Get list of selected properties of C (LP)    /*properties of C
selected by user */
LP2 = [(C, LP)|LP2],
LC2 = Tail,
Until LC2 = [ ]
```

Fig.7: Collect Properties

```
LV1 = [ ] /* values of selected properties of selected concepts of
o1 */
LV2 = [ ] /* values of selected properties of selected concepts of
o2 */
Repeat
LP1 = [(C, [P|Tail1])|Tail2]
      Repeat
         LP1 = [(C, [P|Tail1])|Tail2]
         Get list of selected values of P (LV)
         /*values of P selected by user */
         LV1 = [(C, [(P, LV)|Tail3)]|LV1],
         LP1 = [(C, [Tail1])|Tail2]
      Until Tail1 = [ ]
LP1 = Tail2,
Until Tail2 = [ ]
Repeat
LP2 = [(C, [P|Tail1])|Tail2]
      Repeat
         LP2 = [(C, [P|Tail1])|Tail2]
         Get list of selected values of P (LV)
           /*values of P selected by user */
         LV2 = [(C, [(P, LV)|Tail3)]|LV2],
         LP2 = [(C, [Tail1])|Tail2]
      Until Tail1 = [ ]
LP2 = Tail2,
Until Tail2 = [ ]
```

Fig 8: Collect Values

```
Get list of concepts C1 of o1
Get list of concepts C2 of o2
Get list of properties P1 of each concept of o1
Get list of properties P2 of each concept of o2
Get list of values V1 of each property of each concept of o1
Get list of values V2 of each property of each concept of o2
Get matched concepts MC1  of o1
Get matched concepts MC2 of o2
Get matched properties MP1 of each concept of o1
Get matched properties MP2 of each concept of o2
Get matched values MV1 of each property of each concept of o1
Get matched values MV2 of each property of each concept of o2

Unmatched concepts UC1 of o1 = DIFFERENCE(C1, MC1)
Unmatched concepts UC2 of o2 =DIFFERENCE(C2, MC2)
Unmatched properties UP1 of each concept of o1 =
DIFFERENCE(P1, MP1)
Unmatched properties UP2 of each concept of o2 =
DIFFERENCE(P2, MP2)
Unmatched values UV1 of each property of each concept of o1 =
                              DIFFERENCE(V1, MV1)
Unmatched values UV2 of each property of each concept of o2 =
                              DIFFERENCE(V2, MV2)
```

Fig.9: Determine Unmatched Entities

```
o1, o2 are two source ontologies
Hierarchical Classification of concepts of o1 is HCo1
Hierarchical Classification of concepts of o2 is HCo2
A is the alignment concepts of o1, o2    /* A is a list of matched
concepts */
HCo = Append(HCo1, HCo2)
Repeat
A = [(X, Y)|Tail]
IF user select X
THEN
    {Get Offspring concepts OY of  Y,
     Link OY with  X,
     Delete Y from HCo}
IF user select Y
THEN
     {Get Offspring concepts OX of X
      Link OX with Y,
      Delete X from HCo }
A = [Tail]
Until A = [ ]
Get HCo
```

Fig. 10: Merge Hierarchical Classification

5. ESTABISHING DYNAMIC ONTOLOGY FOR AGRICULTURAL DOMAIN

Domain of Agriculture has huge data and large information which consists of a lot of terms and relations among them. The advantages of ontology in agricultural domain can be summarized as follows: Standardization of agricultural terms, Knowledge sharing, reusing knowledge of agricultural domain. Several ontologies have been built for some topics for some agricultural crops. These ontologies can be integrated for each crop by the proposed system to get global ontology for each crop.

This section presents two ontologies of faba bean crop of food legume crops of field crops obtained from Central Laboratory of Agricultural Expert Systems (CLAES-ARC) and the merged ontology of the two ontologies. The proposed system has been applied on the two ontologies by matching and merging them. Fig.11 shows the screen of browsing the two ontologies. Fig.12 and Fig.13 show the two ontologies. Fig.14 presents the selection of threshold value. Fig.15 shows the alignment of accepted matched concepts. Fig.16 shows the alignment of accepted matched properties. Fig.17 shows the alignment of accepted matched values.

Fig.18 shows number of all system and accepted alignments for concepts, properties and values. "Alignment Match File" is a link to present all system alignments while "Accepted Matched File" is a link to present all accepted alignments. Fig.19 shows the screen to rename the file of the merged ontology and to merge the source ontologies. Fig.20 shows number of concepts, properties and values for the source ontologies and merged ontology, "Print Ontology" is a link to present the merged ontology in HTML file (show Fig.21). "Global ontology file" is a link to present the merged ontology in XML file (show Fig.22). Fig.23 shows the hierarchical classification of concepts and their properties and values.

6. CONCLUSION

This paper presents a dynamic ontology for agricultural domain. It presents a system used for building an ontology from different other ontologies in the same domain.

The proposed system has a capability to modify, delete, or add entities from integrated ontology. So the integrated ontology is dynamic, it means that it can be modified when needed by using the proposed system. The proposed system has been

applied on the agricultural domain for faba bean crop to get a dynamic integrated ontology, it can be applied also on all crops. This system uses Multi-Matching and Merging Algorithm (MMMA) to get the integrated ontology. It uses five matchers to match between entities in all cases. The using of five matchers makes the matching process perfect and prevents redundancy and inconsistency. The proposed system has been implemented by using ASP.NET C#. This system can be applied on ontologies of different domains. The source ontologies and the target ontology of the proposed system are in XML language. The manipulated ontologies have been represented using CommonKADS methodology [14]. Domain of Agriculture has huge data and large information which consists of a lot of terms and relations among them. The advantages of ontology can be summarized as follows: Standardization of agricultural terms, Knowledge sharing, reusing knowledge of agricultural domain.

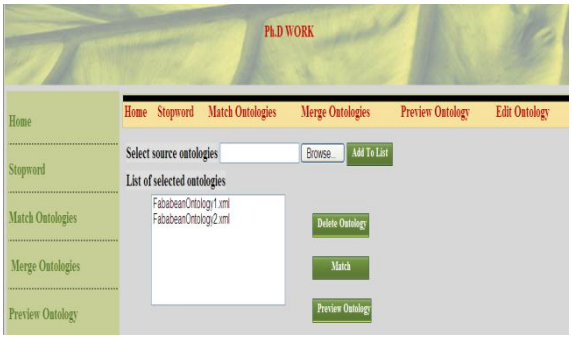

Fig.11: Matching Window

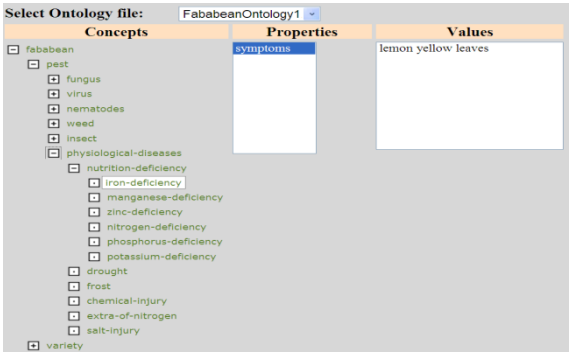

Fig.12: Part of first ontology (FababeanOntology1)

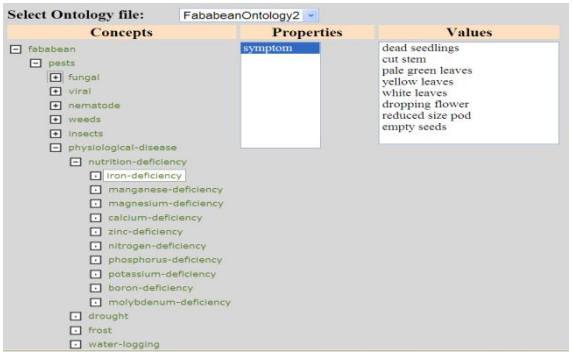

Fig.13: Part of second ontology (FababeanOntology2)

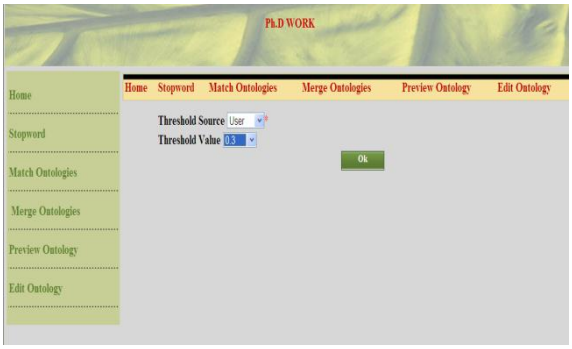

Fig.14: Threshold Window

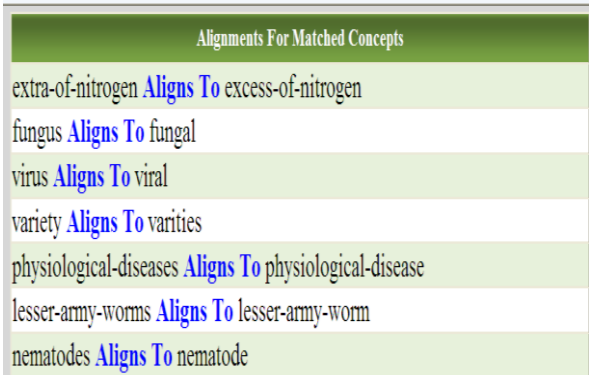

Fig.15: Part of Concept Alignment

Fig.16: Part of Property Alignment

First Value	Alignment matched values	Second Value
chocolate-spot . symptoms . small reddish brown spots on stem	Align To	chocolate-spot . symptom . small or elongated reddish brown spots on stem
chocolate-spot . symptoms . reddish brown spots on leaves	Align To	chocolate-spot . symptom . chocolate spots on leaves
drought . symptoms . wilted leaves	Align To	drought . symptom . gray leaves
drought . symptoms . wilted leaves	Align To	drought . symptom . wilting leaves
chemical-injury . symptoms . white spots on leaves	Align To	chemical-injury . symptom . white or brown spots on leaves
chemical-injury . symptoms . brown burning spots on leaves	Align To	chemical-injury . symptom . white or brown spots on leaves
green-aphid . symptoms . distorted pods	Align To	green-aphid . symptom . malformed pods
pea-aphid . symptoms . distorted pods	Align To	pea-aphid . symptom . malformed pods
lesser-army-worms . symptoms . cutting leaves	Align To	lesser-army-worm . symptom . inner eating in leaves
cutworm . symptoms . wilted leaves	Align To	cutworms . symptom . wilting leaves

Fig.17: Part of Value Alignment

	Accepted Alignment	System Alignment
Concepts Alignment Number	58	75
Properties Alignment Number	82	110
Values Alignment Number	136	168
Download File	Accepted Matched File	Alignment Match File

Fig.18: Final Alignment

Fig.19: Merge Window

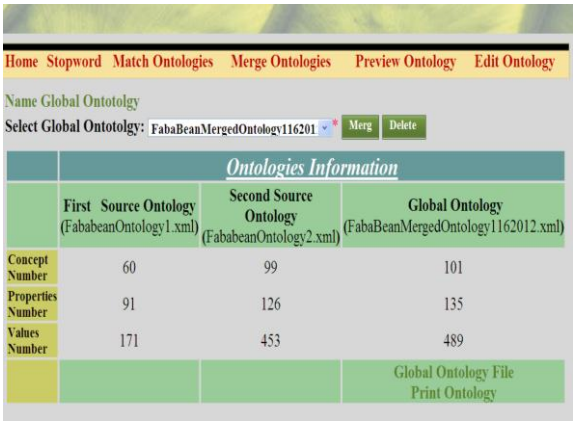

Fig.20: Merged ontology and ontologies information

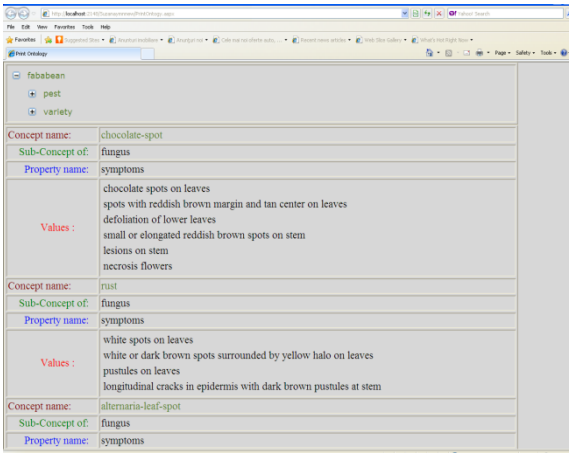

Fig.21: Part of merged ontology in HTML file

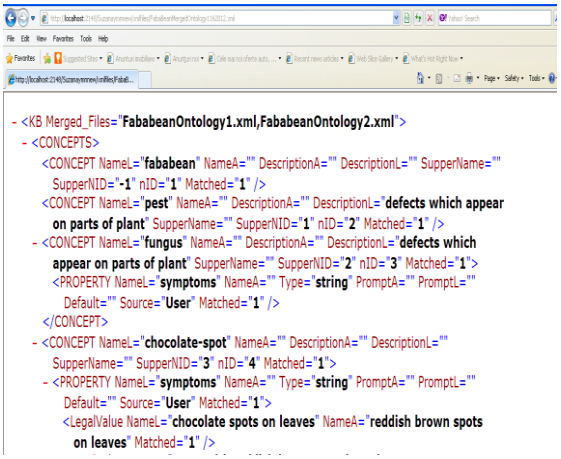

Fig.22: Part of merged ontology in XML file

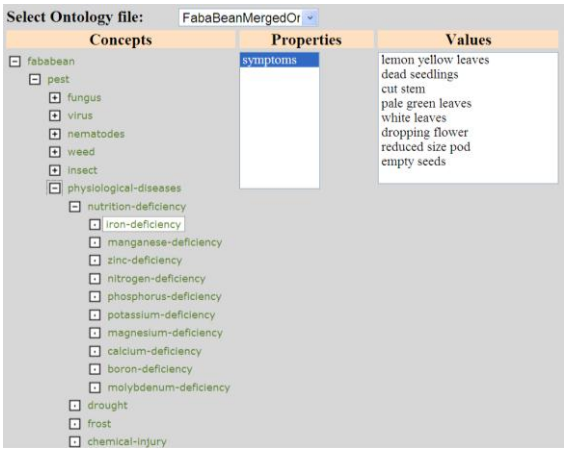

Fig.23: Part of merged ontology in hierarchical classification

7. REFERENCES

[1] Borst P, Akkermans H, Top J.1997. Engineering ontologies, International Journal of Human-Computer Studies 46:pp.365–406.

[2] N. Noy and M. Musen. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, Proc.of17th National Conference on Artificial Intelligence,(AAAI,), pp. 450–455, Austin, Texas.

[3] H. Chalupsky. 2000.Ontomorph: A Translation System for Symbolic Knowledge, Principles of Knowledge Representation and Reasonin.

[4] R. Ichise, H. Takeda, and S. Honiden. 2001. Rule Induction for Concept Hierarchy Alignment, Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI).

[5] An-Hai Doan, J. Madhavan, Pedro Domingos, and Alon Halevy. 2003. "Learning to map ontologies on the semantic web", In Proceedings of the International World Wide Web Conference (WWW), pages 662–673.

[6] Marc Ehrig and Steffen Staab. 2004. "QOM: Quick ontology mapping", In Proceedings of the 3rd International Semantic Web Conference (ISWC), pages 683–697.

[7] Peng Wang and Baowen Xu.: Lily: Ontology alignment results for oaei 2009.In Shvaiko.

[8] Muhammad Fahad, Nejib Moallaa, Abdelaziz Bouras.2011. Towardsensuring Satisfiability of Merged Ontology. International Conference on Computational Science, ICCS. Procedia Computer Science 4 (2011) 2216–2225.

[9] Li, Y., Zhong, Q., Li, J., and Tang, J.2007.Results of ontology alignment with RiMOM. In Proc. International workshop on Ontology Matching (OM), Busan, Korea. Pp. 227-235. November 11.

[10] K. Kotis, G.A. Vouros, K. Stergiou, Towards automatic merging of domain ontologies: The HCONE-merge approach, Web Semantics: Science, Services and Agents on the World Wide Web, 4(1) (2006) 60-79.

[11] Straccia, U. and Troncy, R. oMAP: Combining classifiers for aligning automatically OWL ontologies. In Proc. 6th International Conf. on Web Information Systems Engineering (WISE), pp. 133-147, New York (NY US) 2005.

[12] Castano, S., Ferrara, A., and Montanelli, S. Matching ontologies in open networked systems: Techniques and applications. Journal on Data Semantics, V: 25-63, 2006.

[13] Steffen Staab and Rudi Studer. 2004 . Handbook on ontologies. International handbooks on information systems. Springer Verlag, Berlin (DE).

[14] Bon J. Wielinga.1994. Expertise Model Definition Document, University of Amsterdam.