Conception of Ontology for Security in Health Care Systems

Dr. J. Indumathi

Department of Information Science and Technology,
Anna University, Chennai, Tamilnadu, India

Abstract- The insidious and omnipresent scenery of the network united with mounting concerns about computer-generated intimidation stipulate instant solutions for securing the network communications. By equipping the software agents with the knowledge and by investing meta-data information sources we can provide the users with effectual information Services. So far, the investigation in network security first and foremost focused on securing the information rather than securing the infrastructure itself. Given the widespread intimidation state of affairs, there is a gripping want to enlarge architectures, algorithms, and protocols to apprehend a trustworthy network infrastructure. In order to attain this aspiration, the foremost and leading step is to develop an ample perceptive of the security threats and existing solutions as stated in this paper. A way of building Ontologies that proceeds in a bottom-up fashion is presented, defining concepts as clusters of concrete XML objects. Clusters are being generated, which are formed based on the structure of the input XML documents. The learning domain is a more general concept of security and health care system. On today's global information infrastructure, manual knowledge extraction is often not an option due to the sheer size and the high rate of change of available information. A bottom-up method for ontology extraction and maintenance intended at impeccably harmonizing current ontology design practice, where, as a rule, Ontologies are designed top-down.

I. Introduction

Conventionally, the loom to community-oriented ontology building has been a mutual one aimed at valorizing the involvement of each community member in the knowledge creation activity. Metadata extraction and merging is carried out by hand by individual users as a part of their daily activities, possibly taking sample data items into account. However, some drawbacks do exist. Cooperatively building and maintaining ontology takes a substantial amount of time, even in the presence of a careful process management. For this reason, recent research twisted once again too automatic and semiautomatic (as opposed to cooperative) metadata construction and maintenance. Automatic techniques for building and integrating metadata have been studied for many years by the database Community. More recently, several techniques specifically aimed at learning Ontologies from text corpora or database repositories were proposed.

In security-sensitive contexts, the ontological connection between data and its connotation frequently utters how the data is to be used. In this sense the ontology develops into a security policy. Establishing a risk management process over the effectiveness and efficiency of the security controls has to be done. This is often an effort-consuming intrusion, especially for large organizations, which has not yet been appropriately assisted by automated processes. Deem a patient ontology for healthcare applications. Only administrators may access the healthcare records of the patient ontology. That is, different users are granted access to different parts of the ontology. We have used languages such as XML (eXtensible Markup Language), OWL to specify such security policies.

II. Related Work

The Onion system [9] was born as an attempt to reconcile Ontologies underlying different biological information systems. However, Onion is aimed at merging fully fledged, competing Ontologies rather than at enriching and developing an initial ontology based on emerging domain knowledge. The FCAMERGE[4] technique is much closer to ours, inasmuch it follows a bottom-up approach, extracting instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances, classical mathematical techniques like Formal Concept Analysis (FCA) are then applied to derive a lattice of concepts to be later manually transformed into ontology. Focusing on feature taxonomies, Gupta et al. recently proposed a bottom-up learning procedure called TAXIND [11] for learning taxonomies. An important contribution toward bridging the gap between conventional text-retrieval and structure-aware techniques was given by Bordogna and Pasi [12], whose work, however, does not specifically address Web and XML data.

Well-known research approaches to ontology learning include the Pattern-based extraction, Conceptual clustering, Association rules mining, Ontology extraction and merging. Approaches to content-based clustering (see [14] for details) can be classified as Hierarchical Algorithms, Iterative Algorithms, Meta-Search Algorithms

III. Problem Description

The project is all about crafting Ontology for the security of the domain of our interest in a semi-automatic way. First we collect XML documents (representing domain knowledge) related to security aspects like Security issues, Algorithms, Policies. Classify the gathered documents using fuzzy clustering techniques. Here we adopt both structure as well as content based clustering techniques the classification of the documents. This will help for a generating a better taxonomy with characteristics like cardinality, sub-class of relations, part-of relations. Using these clustered documents generate ontology metadata suitable for enriching and updating an existing ontology. Next we design Ontology for Health Care System. Merging of the above-formed Ontologies gives Secured Ontology for the Health Care System.

Formation of a fuzzy bag for each and every input XML document is presented here. Also clustering of these fuzzy bags based on their structure is also being done. While traditional ontology learning approaches focus on hyperonymic and association relations, our technique takes advantage of structural information, including the elements' cardinality. In particular, our approach is aimed at detecting four ontological relations which are subclass-of, value restriction, cardinality restriction, and Associated-to.

IV Architecture of the Proposed Work

Figure 1 shows the architecture, which contains three major steps for building taxonomy of the domain knowledge. Forming Fuzzy bag, performing structural based clustering and content based clustering. XML documents are composed of a sequence of nested elements (possibly repeated) called *tags* (*xi*), each containing another tag, a value or both of them. In our approach we only take into consideration tag names and not their value. The XML documents are nested representation of data; we can use Fuzzy techniques to encode a nested object structure in a flat representation, keeping some memory of the structure in the membership values. The first step of our fuzzy extraction process consists of encoding XML documents, (i.e. instances of our simplified XML data model) as fuzzy multi-sets, i.e. fuzzy bags.

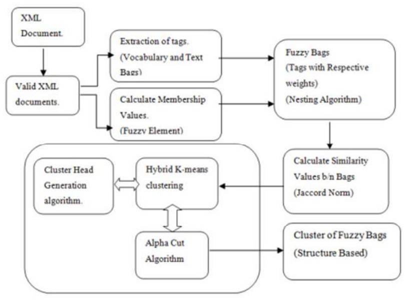

XML Document.
Valid XML documents.
Extraction of tags. (Vocabulary and Text Bags)
Calculate Membership Values. (Fuzzy Element)
Fuzzy Bags (Tags with Respective weights) (Nesting Algorithm)
Calculate Similarity Values b/n Bags (Jaccord Norm)
Cluster Head Generation algorithm.
Hybrid K-means clustering
Alpha Cut Algorithm
Cluster of Fuzzy Bags (Structure Based)

Figure 1. Overall System architecture

A fuzzy bag is a collection of elements with multiple occurrences and having degrees of membership. Bags and fuzzy sets can be viewed as particular cases of fuzzy bags. Fuzzy bags are a straightforward extension of fuzzy sets, where each element can have multiple instances. In a fuzzy set each element is associated to a membership value. For example, a fuzzy bag can be obtained when some attributes are removed from a fuzzy set of topples. This is illustrated by the query: find the salaries of young employees which requires a projection (salary) of a fuzzy set of persons (the young employees) and delivers a fuzzy bag.

The encoding of an XML tree as a *fuzzy bag* can be performed as follows: supposing that every tag *xi* has associated a membership degree μ(*xi*), initially equal to 1 (μ(*xi*) = 1), we divide every membership degree by its nesting level *L*, obtaining a lossy representation of the tree, taking into consideration the original nesting level of tags, but not the brotherhood relations. The *membership value* is an index of the position of the element in the document; it is used to keep memory of the document structure and the nesting level of the tag. This process is called *encoding process*: many encoding techniques can be used; the simple one we just described is called *flattening*. In order to compare the fuzzy bags representing the XML documents belonging to the data flow, we have used a *measure of resemblance*:

S (A, B) = M (A /B) / M (A / B)   [M: Modulus, A, B: Fuzzy Bags]   (1)

Similarity values obtained from the comparison of this subset of documents are then used to populate a matrix. Here, an α-cut has been performed and original fuzzy values have been replaced by crisp value 1 where the similarity value is higher than threshold α. we compute a collection of blocks composed of documents close to a pattern, again based on similarity. Though our blocks are not the result of proper clustering, since the chosen similarity measure lacks the mathematical property of a distance, we shall call them *clusters* in the following. Intuitively, patterns within a valid cluster are more similar to each other than they are to a pattern belonging to a different cluster.

It is necessary to extract a simple description of each cluster. This process of *data abstraction* produces a compact description of each cluster in terms of cluster prototypes or representative patterns, that we call *cluster-heads*. A good candidate to be cluster-head should be the smallest fuzzy bag in the cluster (i.e. the one with the smallest number of elements whose membership is greater than 0) or, using a (non symmetrical) measures of inclusion, the bag more included in each other bag belonging to the same cluster. The latter generates a new fuzzy bag as the union of all fuzzy bags in the cluster. This way, a cluster head does not necessarily coincide with a real data item since each tag is taken with the cardinality and membership degree it has in the bag where it appears with the highest cardinality.

V. Implementation

The algorithms and procedures we followed to perform the above-specified tasks are being explained below in detail:

Nesting Algorithm Fuzzy Bag Generator

- Input the parameters Text bag, Fuzzy bag, Lroot.
- Loop the following steps until all the tags of theXMLdoc are traversed.
- Create fuzzy element to store the Tags and its nesting level (weight).
- Membership= weight/Lroot [M= V / L], where Lroot=1 when we are at the second nesting level of theXMLdocument
- Increment the Lroot value by one until the loop ends.

Intersection of two fuzzy bags

- Take input as two Fuzzy bags.

- Extract the elements from the two bags which are in common, stored in temporary bag called result.
- Loop until the end of result bag elements are traversed;
- Store weights of elements of two input bags in two lists and sort them.
- Obtain the intersection of weights of two lists and store it in weights Intersection.
- Traverse through the passed in lists and obtain a third list whose size is the size of the shorter list and whose element 'i' is equals to the minimum between weightsBag1.get(i) and weightsBag2.get(i).
- End the loop and return the result.
- Given for example the lists L1 = [.5, .3] and L2 = [.9, .2, .1] the result will be L3 =[.5, .2].

Union of two fuzzy bags.

- Take input as two Fuzzy bags.
- Extract the elements from the two bags and perform union and store them in temporary bag called result.
- Loop until the end of result bag elements are traversed;
- If the element is present in only one of the bags just copy it into the result.
- If the element found in two bags do
- Store the weights of elements of two bags in two lists and compute union, which is stored in UnionWeights.
- Traverses the two lists (which must be sorted in descending order) and return a third list whose size is the size of the longer list and whose element 'i' is equal to the maximum between weightsBag1.get(i) and weightsBag2.get(i).
- End if, end Loop and return the result.
- Given for example the lists L1 = [.5, .3] and L2 = [.9, .2, .1] the result will be L3 = [.9, .3, .1].

Division of two fuzzy bags.

- Take input as two Fuzzy bags
- A fuzzy number which represents membership value and multiplicity.
- Compute union on the membership values and store it as union.
- Sort the above list in any order.
- Get division multiplicity for mult1= (union, f_1) and $mult_2$= (unon,f_2).
- Loop until the end of union list;
- Get the membership values in to m_1 and m_2.
- If one of the elements is null do not add the element to the number.
- Else union.get (i) = Value (m_1) / Value (m_2).

Jaccord Norm Algorithm.

- This algorithm is used to compute he similarity value between two fuzzy bags. Given two bags A and B computes the following norm: S (A, B) = |intersection (A, B)| / |union (A, B)|. Where A, B are two fuzzy bags and S is similarity value between them.
- Use Average approximation algorithm to round off the float digit obtained while computing the division of two membership values.

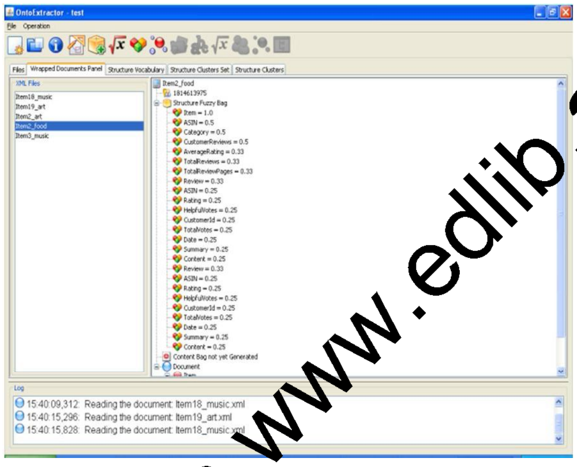
K-means Clustering Algorithm.

- Choose an initial cluster and drop the first bag in to it.
- Compute the cluster head, which is equal to the union of member fuzzy bags.

- Compute the similarity value between cluster heads and one un-clustered bag by using Jaccord Norm.
- Choose an initial alpha cut value.
- If the similarity value is greater than alpha drop it into the same list otherwise create a new cluster for the bag.
- Repeat the above procedure for different alpha values.
- Choose the most suitable cluster from these to further processing.

VI. Simulation Results

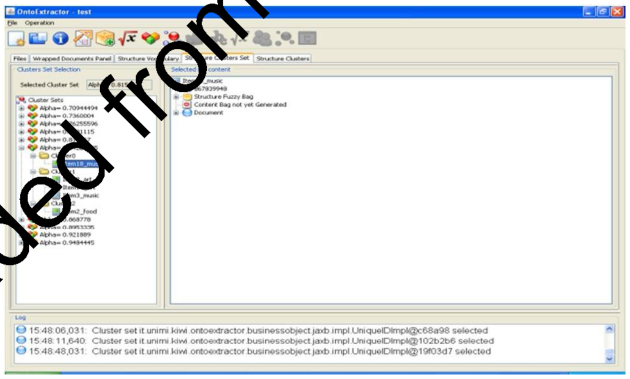
The simulation of the algorithms presented in the design part produce two major outputs. One is the creation of a Fuzzy Bag for each input XML document and the other is a Structure based Clustered fuzzy Bags. These are shown in the following figure which is produced by the tool OntoExtractor.



```
Item18_music
Item19_art
Item2_art
Item2_food
Item3_music
```

```
Item2_food
  1814613975
  Structure Fuzzy Bag
    Item = 1.0
    ASIN = 0.5
    Category = 0.5
    CustomerReviews = 0.5
    AverageRating = 0.33
    TotalReviews = 0.33
    TotalReviewPages = 0.33
    Review = 0.33
    ASIN = 0.25
    Rating = 0.25
    HelpfulVotes = 0.25
    CustomerId = 0.25
    TotalVotes = 0.25
    Date = 0.25
    Summary = 0.25
    Content = 0.25
    Review = 0.33
    ASIN = 0.25
    Rating = 0.25
    HelpfulVotes = 0.25
    CustomerId = 0.25
    TotalVotes = 0.25
    Date = 0.25
    Summary = 0.25
    Content = 0.25
  Content Bag not yet Generated
  Document
```

```
15:40:09,312: Reading the document: Item18_music.xml
15:40:15,296: Reading the document: Item19_art.xml
15:40:15,828: Reading the document: Item18_music.xml
```



```
Cluster Sets
  Alpha= 0.70944494
  Alpha= 0.7360004
  Alpha= 0.76255596
  Alpha= 0.[illegible]1115
  Alpha= 0.8[illegible]7
  Alpha= [illegible]
    Cluster0
      Item18_music
    Cluster1
      [illegible]_art
      Item[illegible]
      Item3_music
    Cluster2
      Item2_food
  Alpha= 0.868778
  Alpha= 0.8953335
  Alpha= 0.921889
  Alpha= 0.9484445
```

```
Item[illegible]_music
  [illegible]867839948
  Structure Fuzzy Bag
  Content Bag not yet Generated
  Document
```

```
15:48:06,031: Cluster set it.unimi.kiwi.ontoextractor.businessobject.jaxb.impl.UniqueIDImpl@c68a98 selected
15:48:11,640: Cluster set it.unimi.kiwi.ontoextractor.businessobject.jaxb.impl.UniqueIDImpl@102b2b6 selected
15:48:48,031: Cluster set it.unimi.kiwi.ontoextractor.businessobject.jaxb.impl.UniqueIDImpl@19f03d7 selected
```

Figure 2 a Fuzzy Bag Representations; 2 b. Structural Clusters Generated With different alpha values

In the figures 2 a and 2 b., the list of tags with their respective weights are being displayed. Also the document is being displayed in the form of tree. We perform clustering for different alpha cut-off values so that we were not restricted to choose an initial cluster of fuzzy bags for further processing.

VII. Conclusion and Future Work

Even though a mammoth literature on information extraction is available, developing and maintaining ontology-based metadata is still more an art than a science. Speedy evolution of obtainable information is difficult to control and often potentially useful, emerging concepts linger disregarded. This project addressed this predicament by a bottom-up, semiautomatic fuzzy technique for generating Ontologies from semi structured data flows.

Further clustering the bags based on their content helps in engendering the classes organized in to hierarchy, which further can be deciphered in to OWL standard knowledge representation format by taking each concept as a candidate class connected by a subclass-of relation to all concepts included in it.

References

1. Ceravolo, P., Damiani, E., & Viviani, M. (2007). Bottom-up extraction and trust-based refinement of ontology metadata. *Knowledge and Data Engineering, IEEE Transactions on*, *19*(2), 149-163.
2. Tsoumas, B., & Gritzalis, D. (2006, April). Towards an ontology-based security management. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on* (Vol. 1, pp. 985-992). IEEE
3. Kim, A., Luo, J., & Kang, M. (2005). *Security ontology for annotating resources*(pp. 1483-1499). Springer Berlin Heidelberg.
4. Schmitt, I., & Saake, G. (1998, August). Merging inheritance hierarchies for database integration. In *Cooperative Information Systems, 1998. Proceedings. 3rd IFCIS International Conference on* (pp. 322-331). IEEE.
5. Karyda, M., Balopoulos, T., Dritsas, S., Gymnopoulos, L., Kokolakis, S., Lambrinoudakis, C., & Gritzalis, S. (2006, April). Ontology for secure e-government applications. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on* (pp. 5-pp). IEEE.
6. Cui, Z., Damiani, E., Leida, M., & Viviani, M. (2005, January). OntoExtractor: a fuzzy-based approach in clustering semi-structured data sources and metadata generation. In *Knowledge-Based Intelligent Information and Engineering Systems* (pp. 112-118). Springer Berlin Heidelberg.
7. Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology
8. Rocacher, D., & Bosc, P. (2005). The set of fuzzy rational numbers and flexible querying. *Fuzzy Sets and Systems*, *155*(3), 317-339
9. Mitra, P., & Wiederhold, G. (2001, November). Algebra for semantic interoperability of information sources. In *Bioinformatics and Bioengineering Conference, 2001. Proceedings of the IEEE 2nd International Symposium on* (pp. 174-182). IEEE.
10. Ceravolo, P., Nocerino, M. C., & Viviani, M. (2004, January). Knowledge extraction from semi-structured data based on fuzzy techniques. In *Knowledge-Based Intelligent Information and Engineering Systems* (pp. 328-334). Springer Berlin Heidelberg.
11. Gupta, K. M., Aha, D. W., & Moore, P. (2004). Learning feature taxonomies for case indexing. In *Advances in Case-Based Reasoning* (pp. 211-226). Springer Berlin Heidelberg.
12. Bordogna, G., & Pasi, G. (2001, December). A user-adaptive indexing model of structured documents. In *Fuzzy Systems, 2001. The 10th IEEE International Conference on* (Vol. 2, pp. 984-989). IEEE.