Using Hash based Bucket Algorithm to Select Online Ontologies for Ontology Engineering through Reuse

Nadia Imdadi
Dept. of Computer Science
Jamia Millia Islamia a Central University,
New Delhi India

Dr. S.A.M. Rizvi
Dept. of Computer Science
Jamia Millia Islamia a Central University,
New Delhi India

ABSTRACT
Semantic web has added a new layer called the knowledge representation layer over the web that may be used to describe a resource on the web. Schemas of domains are represented in machine processable languages known as the ontologies over the semantic web. Construction of ontologies for the semantic web has become a relevant research issue. Ontology engineering has been focus of research in the field of AI since the 70's and with the rising number of ontologies on the web; it should be convenient to reuse these ontologies to build newer ones. In this paper a hash based bucket algorithm is presented for identification of relevant online ontologies, to create ontology that will represent a domain without having to go through building of ontology from the scratch thereby reducing the time, efforts and costs of ontology engineering.

General Terms
Semantic Web Ontologies, Ontology Engineering through Reuse.

Keywords
OWL Ontologies, Hash-based Bucket Algorithm, Ontology Engineering

1. INTRODUCTION
In the context of semantic web ontologies are formal representation of a domain authored in a semantic web language such as OWL [1] that supports reasoning. The basic idea behind semantic web is to provide mechanism where resources on the web can interoperate to arrive at conclusions from the existing web resources. Several semantic web languages have evolved since the formation of W3C [2] a consortium that has over the years developed languages, based on description logics, which make reasoning over web resources possible.

Ontologies play crucial role in the semantic web and provide the infrastructure necessary for semantic web applications such as information retrieval based on question answering system like the [3] or software agents to cooperate and collaborate with different resources to suggest solutions based on user query. As ontologies are central to the scheme of things for semantic web's success their construction has become a central research issue for the semantic web [4].

Development of ontologies has been focus of research in the area of artificial intelligence (AI) since the 70's as knowledge representation of knowledge bases to facilitate intelligent and informed decision making capabilities to systems. Despite several advances ontology engineering process remains a time consuming and costly affair as it involves skills specific to domain as well as knowledge of knowledge representation language. Broadly speaking following stages in ontology construction have been identified 1) acquisition of domain knowledge resources 2) selection of relevant resources 3) integration of the resources and, 4) evaluation of the created ontology. Most of these stages have remained predominantly semiautomatic as human intelligence is needed to identify concepts that represent some context belonging to particular domain. Despite the nature of work associated with the process of ontology engineering a higher degree of automation may be achieved by making some of these stages/their sub stages independent of human intervention.

In [5, 6] a way to automate some stages of ontology construction was explored. The stages identified for automation were ontology acquisition, filtering and integration. Automation of the first stage of ontology building where a which is based on an initial set of concepts, given by an expert as input to a system that will make use of, or reuse existing online resources for the purpose of ontology building. Using existing online ontologies for ontology engineering was proposed in [7], based on which a framework (Figure 1) was forwarded in [8].

2. MODULAR APPROACH TO ONTOLGOY BUILDING
A modular approach to building ontology is the underlying methodology of the framework. A bottom up approach to ontology construction is employed as any domain consists of concepts, which in turn are collection of few terms, properties and relations amongst these terms. Thus based on the key terms an input matrix is created that is then used for concept extraction from knowledge resources which may be locally or globally imported. The bottom approach is suitable as concepts may be present across multiple online semantic repositories and therefore these will have to be identified, extracted and integrated using appropriate strategy.

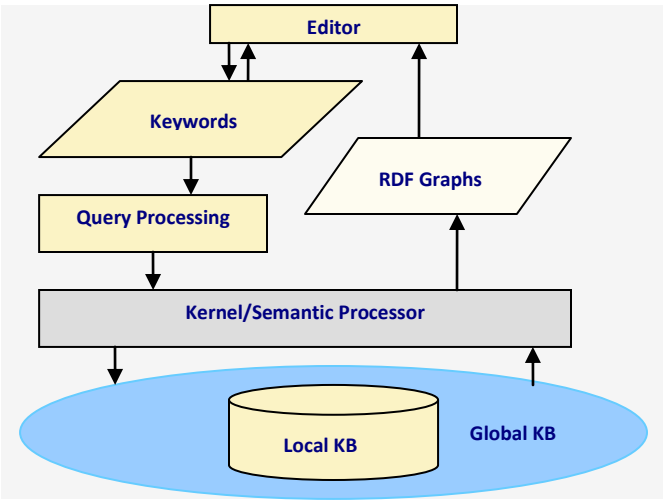

Fig 1: Framework for Ontology Building

2.1 Word Sense Disambiguation in Input Query Processing

In [9] it is deliberated that authoring or querying knowledge on the semantic web is not easy as it requires people to manual selection concepts from is very tedious. Writing complex RDF statements and queries is also a difficult task for many as it requires familiarity with the appropriate ontologies and the terms they define. Using natural language words and vocabulary is considered a more convenient option for users. A system that allows retrieval based on word or set of words is forwarded. The approach is based on analyzing the co-occurrences of ontologies and the terms (cluster of identifiers) defined for each ontology. However, it is noted that structure information about which property word is applied on which class word, which could reduce ambiguity of words, is not considered in order to gain computational time advantage.

However, in this proposed framework appropriate property word will be selected for a given class word to yield a better and more relevant result set at the time of integration of concepts. During processing of input query, synonyms and morphological variants of English words have been restricted to one or at most two synsets using the WordNet [10]. A set of keywords is used consisting of concept name is called the input matrix. For e.g. if set of concept is represented by following terms/names: wine, region, flavor, co lour, vintage- then property words which may be associated with flavor are sour or sweet. This form of input reduces the result set and increases the accuracy of retrieval as most likely other namesake namespaces.

2.2 Global Query Service Routine Handling

After analyzing the type of data structure underlying Swoogle's [11] database, it is realized that Swoogle supports keyword based search on its index of ontology terms. When a keyword is searched a list of namespaces where the keyword is defined is returned. Since multiple keyword based searches are not supported by Swoogle API, namespaces against each concept name input is retrieved and a list of those namespaces with maximum number of occurrences of a group of concept name inputs is prepared. Appropriate strategy to limit the size of resulting relevant namespace list is needed in order to keep computational time frame within acceptable limits as Swoogle indexes over 3 million semantic web documents. A strategy has been formulated to select namespaces according to the coverage of most of the terms from the input matrix and is discussed in the following sections.

3. ACQUISITION AND SELECTION OF DOMAIN KNOWLEDGE

3.1 Input Model

In order to retrieve concepts for building ontology by reuse we apply a modular approach with some hypothesis with which to begin with:

Premise 1: A concept may be identified when a couple of words/terms appear together. This is true both when we have natural language running text and even in case of structure schemas like ones found in a database or a namespace- our domain of interest. For instance book, publisher, author, title & topic appearing together in a namespace would mean that these terms are more likely to be is related to library, or academics, and not travel industry where book may be to reserve, publisher from printing etc.

Premise 2: Since a word may have more than one sense an attribute or property associated with it can be used to identify its context. Again an attribute for the book is its issn number or edition.

Premise 3: In cases of structured information like that of namespaces, context can be identified as super class (parent-of) and sub class are known. Here an example is Reading Material-Book – fiction.

The form of input to be used for querying the semantic web to locate namespaces is decided based on the above premises and as the nature of the semantic repositories where ontologies are represented in form of rdf graphs. As authoring these rdf graphs is not an easy task, methodology to make use of existing ontologies should cut done the job of knowledge engineer significantly. The challenge is to develop strategies to first identify relevant namespaces and then to extract and integrate from these multiple namespaces to build an ontology.

As noted earlier access to namespaces on the semantic web is possible as some semantic web document search engines have been developed. Swoogle is one such engine and it provides a web service interface using which its database of indexed semantic web documents can be accessed. Querying is allowed through a REST [12] interface and several options based on different parameters are possible where search services are powered by Apache Lucene.

3.1.1 Input Matrix

Using premise 1 a list of namespaces is retrieved where input is in form of concept matrix (cm), which consists of set of keywords say K. It is assumed that to form a concept representation say C at least 3 terms should appear in a namespace, say n. We have fixed the maximum number of terms necessary to form a concept at 5, in order to keep the computational time in check.

To get a larger result set based on premise 2 the cm is expanded with list of same sense synonyms say S, which again has been restricted to two for each term in the cm for a C.

Therefore, if a cm is given by [t1, t2, t3, t4, t5] where t1-t5 is term names then the expanded cm version is

[t1, t1s1, t1s2]

[t2, t2s1, t2s2]

[t3, t3s1, t3s2]

[t4, t4s1, t4s2]

[t5, t5s1, t5s2]

Where tns1, tns2 for 0<n<6, are synonym variants. These variants are pulled up from Wordnet Db to allow user to select sense of the word, and the synonyms.

3.2 Hash Based Bucket Algorithm

Upon approval of input cm a httpclient method is used to execute Swoogle Api against the swoogle database. Each query gives a list of namespaces where the term is defined. The maximum limit imposed by Swoogle Api for result retrieval is 1000 for each query string, and therefore maximum number of namespaces retrieved can be up to 1000 only.

3.2.1 Data Structure

A query is executed for each term in a cm. In order that premise 1 holds it is required that the namespaces, URI's, with at least 3 or more terms be clubbed together. For instance a namespace URI appears in all the results for each term query then it may be deduced that it has all the terms and therefore is more like to represent the concept.

The challenge is to select a data structure as the problem is to find common links across multiple lists of namespaces. Now picking up one link and then searching for it in other lists and repeating the process for all the links from all the lists would be time and resource consuming process, in order to minimize the complexity and computational time hashing technique can be used as it allows a quick search in a data structure. Advantage of using hash technique is that a linear search through all the elements, in this case namespaces is not needed.

A hash function and table algorithm becomes an obvious choice for data structure because of the following:

- It can be customized according to the nature of the problem which in this case requires namespaces common in the results sets to be clubbed together or alternatively it requires clubbing together the terms forming a concept against each namespace.

- A hash table can be implemented as an array of buckets which are sequences of nodes that hold elements with same hash code this suits our requirement as we can compute hash code, h, for namespace say n, and store the terms as node in a bucket

In other words a hash code is used as an array index into the hash table and multiple objects with same hash code are stored at the same array position. Customizing the bucket algorithm to solve the present problem the terms of a concept are taken as objects and the array index is used to identify a namespace. A namespace occurring for the first time for a term will point to a new location or index in the array and the term will be the first bucket element. If the namespace is returned for the next term then the term would be inserted as the second bucket element and so on and so forth. For each namespace returned as against term query the above procedure is repeated. In the end array of buckets is created which hold the key of namespace and the number of terms to which the namespace is common.

3.2.2 Treating Synonyms

A synonym term is added to above array of buckets only when it's key term or variant synonym is not present in the list, that is, for a key term t1, t1s1 or t1s2 is only added at the exclusivity of the other two.

3.2.3 Output

A namespace is considered to be a candidate of interest only if it contains 3 or more terms and so using the array bucket we can find the candidate namespaces.

3.3 Criteria for Selection of Candidate Namespace

A weight of 0.2 is assigned to a key term and a weight of 0.1 to a synonym variant, thus if all the key terms are present then a weight of 1 is assigned to the candidate namespace else according to the composition of the concept formed from key terms and the synonym variants. Note for any namespace to qualify as a candidate at least 3 key terms must be present. Based on the above criteria several lists of namespaces may be created with varying weights.

3.4 Aggregation: Final Step in Stage of Ontology Acquisition

Initial Input: Set of concepts cm Є C based on which a list of ranked namespaces for each cm is created. Aggregation of weights of Namespaces present across all cm's in C is performed, for example a namespace is present across 4 cm's and has weights of (1, 0.6, 0.8, and 0.9) then aggregation gives a value of 3.3.

After aggregation function computes weights of namespaces across all concepts the namespace with the maximum weight is considered the most promising candidate and is selected as the starting namespace from where concept clustering seeds are formed.

4. ACQUISITON OF NAMESPACES FROM DOMAIN – FOOD: AN EXAMPLE

To demonstrate the how the previous algorithm executes a set of terms grouped roughly into five concepts are selected as input from the domain food.

Table 1 gives a set of concepts with corresponding expansion done using WordNet Db, a dash in the synonyms column signifies that no appropriate synonym meeting the same sense requirement was found.

Table 1. Input Matrix from Domain – Food

Concept(cm)	Term	Synonym 1	Synonym 2
Concept 1			
Term 1	Consumable	-	-
Term 2	Edible	Comestible	-
Term 3	Dessert	Sweet	Afters
Term 4	Meat	-	-
Term 5	Meal	Repast	-
Concept 2			
Term 1	Course	-	-
Term 2	Red	-	-
Term 3	White	-	-
Term 4	Spicy	Hot	-
Term 5	Fowl	Bird	-
Concept 3			
Term 1	Pizza	-	-
Term 2	Sauce	-	-
Term 3	Tomato	-	-
Term 4	Cream	-	-
Term 5	Topping	-	-
Concept 4			
Term 1	Fish	-	-
Term 2	Bland	Flat	-
Term 3	Non Bland	-	-
Term 4	Seafood	-	-
Term 5	Oyster	Huiter	-
Concept 5			
Term 1	Grape	-	-
Term 2	Fruit	-	-
Term 3	Wine	Vino	-
Term 4	Sweet	-	-
Term 5	Sour	-	-

Table 2. Weighted Namespaces

Namespace	Aggregate Weight	Concept Coverage
http://www.csd.abdn.ac.uk/~yzhang/food.rdfs	**1.8**	**1, 2**
http://www.ksl.stanford.edu/projects/wine/iw/PASTA-WITH-NON-SPICY-RED-SAUCE-F/IW2.daml	1.6	3,4
http://www.cs.man.ac.uk/~drummond/cs646/examples/pizzas2_5.owl	1.6	3,4
http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml	1.2	1,3
http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine	1.2	1,5
http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf	1.2	3,5

Table 2 contains the final result i.e. after aggregation of weights across all the concepts has been performed. The last column gives an account of the concepts that were found in that particular namespace. Although the first namespace carries the highest weight it is highlighted as it is not accessible, and this may be considered as one of the drawbacks of using online semantic repositories. Manual study of these ontologies suggest that the concepts are present in the said ontologies, and their definitions, attributes and relationships such as class hierarchy can be extracted and reused for the purpose of building ontologies.

5. FUTURE WORK

After the discovery and acquisition of ontologies the next stage is of evolving a methodology for integrating information found across set of ontologies retrieved. It is noted here that the area of ontology integration has been explored, for detailed surveys can be found in [13, 14] with some solutions with varying degree of automation and with a general consensus that this can only be done semi-automatically. However, through this framework it attempted that the process of be automated to a higher degree of integration and it is viable as it is to be done on available

resources that are already in a structured form, this is a definitive advantage as it cuts down on the preprocessing stage in the cycle of ontology development.

Presently, methods for concept building are being explored to evolve a strategy for building and integrating OWL ontologies discovered by using the above method. Important contributions by Michalski and Stepp in [15] who gave the notion of concept building and Visser et al [16] based their work on slots and frames may be adapted successfully in case of OWL ontologies that are based on class, properties and relation between classes.

6. CONCLUSION

An effective strategy to identify and rank online semantic repositories for the purpose of ontology engineering through reuse is explained using a hash based bucket algorithm. Using this technique an important aspect, that of discovery and acquisition in the cycle of ontology development is automated. Furthermore, important contributions in the field of concept building are identified that may form the basis for integration of the structured information gathered from the online ontologies.

7. REFERENCES

[1] www.w3.org/2004/OWL/, W3C, Web Ontology Language (OWL).

[2] www.w3.org/, W3C, World Wide Web Consortium

[3] Vanessa, L., Motta, E., and Victoria U. 2006. Poweraqua: Fishing the Semantic Web.

[4] Zhu, L., Yang, Q., and Chen, Wei. 2009. Research on Ontology Intergration Combined with Machine Learning. IEEE.

[5] Imdadi, N. and Rizvi, S.A..M. 2010. Framework for Automatic Reuse of Existing Online Semantic Resources by Facilitating Concept Extraction Using Word Sense Disambiguation in Computational Linguistics Techniques. Conf. Proceedings SWWS.

[6] Imdadi, N. and Rizvi, S.A..M. 2010. Automating Reuse of Semantic Repositories in the Context of Semantic Web. 2010. (CCIS) Springer.

[7] Harith A. 2006. Position paper: ontology construction from online ontologies. WWW.

[8] Imdadi, N. and Rizvi, S.A..M. 2008. Framework for Automatic Semantic Integration of Semantic Repositories. Conf. Proceedings SEEC.

[9] Lushan, H., Finin, T., Yesha, Y. 2009. Finding Appropriate Semantic Web Ontology. ISWC.

[10] WordNet, http://wordnet.princeton.edu/

[11] Ding, L., Finin, T., Joshi, A., Pan, R., Cost R. S., Peng, Y., Reddivari, P., Doshi, V., and Sachs, J. 2004. Swoogle: a search and metadata engine for the semantic web. In Proceedings of the thirteenth ACM international conference on Information and knowledge management. ACM.

[12] Fielding, R.T., Architectural Styles and the Design of Network-based Software Architectures, PhD dissertation Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender

[13] Aufaure, M., Grand, B. L., Soto, M., Bennacer, N. 2005. Metadata- and Ontology-Based Semantic Web Mining. Idea Group Publication.

[14] Noy, F.N. 2004. Semantic Integration: Survey on Ontology Based Approaches, SIGMOD Record

[15] Michalski, R.S., Stepp, R. 1983. Learning from Observation: Conceptual Clustering. Chapter In Book Machine Learning: An Artificial Approach. Tioga Publishing Co.

[16] Valentina, A.M.T. Pepijn, R.S.V. Donato, M. and Dean M. J. 1999. Computer Assisted Ontology clustering for Knowledge Sharing.