International Journal of Computer Applications (0975 – 8887)
Volume 68– No.1, April 2013

Towards Simple Semantic Annotation

Alaa Qasim Mohammed Salih
Aston University/School of Engineering & Applied Science
Oakville, 2238 Whitworth Dr., L6M0B4, Canada

ABSTRACT

One of the initial requirements for the automatic annotation system is the simplicity and easy to be used by client so that untrained users can interact with the system effectively to create annotations for their documents. To fulfill this requirement, a graphical user interface must be designed. The interface of any system acts as a communication channel between the user and the system. A badly designed interface could result in costly mistakes, inefficient working and create an unpleasant atmosphere for the user. Therefore, to design a successful interface the abilities and limitations of both the computer and the human must be taken into account.
This paper provides an insight into the ideas and thoughts to create a system that meets all of the requirements needed. The process of loading information and data integration is described in order to provide the reader with an idea of the how the system was created. Program code examples will be given to illustrate the key features and mechanisms used.

Keywords

Annotation, metadata, Ontology, Semantic Web, server, XML, RDF and OWL

1. INTRODUCTION

Semantic Web (SW) is the vital proposal that is promoted by the World Wide Web Consortium (W3C).It deals with facilitating the data source to provide the next generation Internet infrastructure such that giving significant meaning, make the client and computer to work in cooperation with each other can be provided by the information [1].

A set of semantically annotated Web resources may give a vision to the semantic web. The web resource may be any type of picture, text or representation of a person. The description of semantics of the resources can be provided by semantic annotation. This show more interest by the software companies in order to extend the predefined target. It is too important to mention that not most users are expert in dealing ontology[2]. They do not have the ability to read, understand how the use ontology and sort through.

There are a set of guidelines and recommendation proposed in [2] to provide good practice to achieve text annotations. The important guidelines are:

1. Easy recovering the original text annotation through taking away the annotations to it added.
2. Facilely extraction of annotation from annotated text.
3. Thorough documentation to be supplemented with eachannotated text.

The integration of data with systems uses other ontologies may lead to apply multiple ontologies. This will lead to use some operations on the ontologies. This is considered as very difficult tasks and cannot be achieved automatically.Fig.1 shows how the way of collecting data from the source, build the semantic metadata and the ontology.

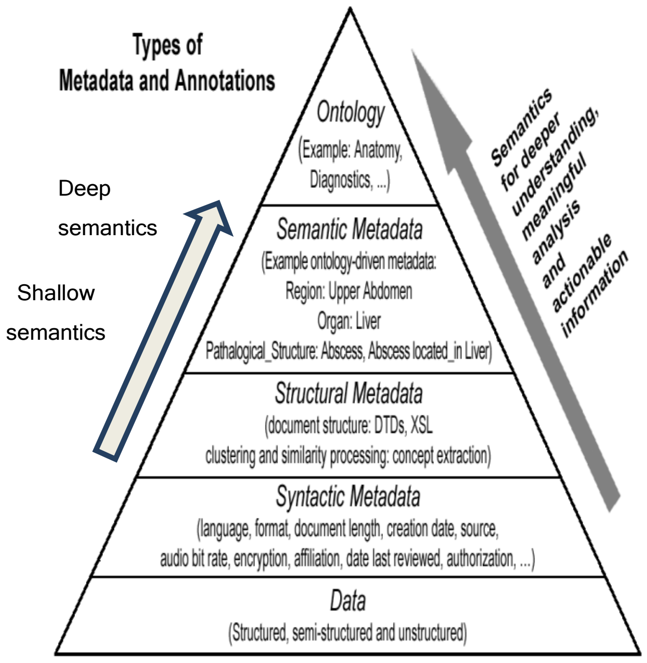

Types of
Metadata and Annotations
Ontology
(Example: Anatomy, Diagnostics, ...)
Semantic Metadata
(Example ontology-driven metadata:
Region: Upper Abdomen
Organ: Liver
Pathalogical_Structure: Abscess, Abscess located_in Liver)
Structural Metadata
(document structure: DTDs, XSL
clustering and similarity processing: concept extraction)
Syntactic Metadata
(language, format, document length, creation date, source,
audio bit rate, encryption, affiliation, date last reviewed, authorization, ...)
Data
(Structured, semi-structured and unstructured)
Deep semantics
Shallow semantics
Semantics for deeper understanding, meaningful analysis and actionable information

Fig.1: From Syntax to Semantics

In this paper, we will give an overview of how the design was realized. Some of the key features of the implementation will be described This chapter will also give an insight into the ideas and thought processes the author has been through to create a system that meets all of the requirements previously defined. The process of loading information and data integration is described in order to provide the reader with an idea of the how the system was created. Program code examples will be given to illustrate the key features and mechanisms used.

2. USER INTERFACE DESIGN

One of the initial requirements for the automatic annotation system was that it must be simple and easy to use so that untrained users can interact with the system effectively to create annotations for their documents [1]. To fulfill this requirement, a graphical user interface must be designed.

The interface of any system performances as a communication channel between the client and the system play a key role in the design. A badly designed interface could result in costly mistakes, inefficient working and create an unpleasant atmosphere for the user. Therefore, to design a successful interface the abilities and limitations of both the computer and the human must be taken into account.

2.1 Human-Computer Interaction Principles

To assist with the design of the user interface, various principles from the field of Human-Computer Interaction
(HCI) have been applied. The HCI can be defined as a field related with many areas *i.e.* evaluation, design and implementation of interactivecomputing systems for
human usage. It also provides studying of main phenomena adjoining them [3].

By far the most evolved and important sense to humans is vision. Therefore, the interface must be visually pleasing and organized in a way that seems naturally appropriate for the task to be completed [3]. This will strongly increase the usability of the automatic annotation system. The information presented to the user should be structured and not cluttered. A cluttered screen is difficult for the user to interpret and requires a lot of extra processing by the brain to make sense of the information. Too much cluttering of information could easily irritate the user, which may then affect their performance.

Consistency is another important factor of interface design. All messages output to the user should use similar language style and the overall structure of the interface itself should not change over time. Humans like to know when they have completed a task as this means they can let go of some pieces of information held in memory that are not relevant anymore [3]. Therefore, suitable messages should be generated when individual tasks have been completed to help provide the user with closure.
The controls of any system are extremely vital, as they are the fundamental method for the user to communicate with the system. The controls should be natural in respect to direction and they should be located in close proximity [3]. The functionality of controls should appear obvious to the user so that they can quickly get to grips with the system.

If an interface is to act as a communication channel between the user and the system, it is clear that it will need to display the information required by the user to operate the system. For the automatic annotation system, the possible annotations generated must be presented to the user so that they can then be accepted or rejected. This means theinterface must display information related to the
possible annotations and also the controls that will allow the user to operate the system (e.g. accept, reject annotation). In addition, the information related to each annotation should be editable, so that the user can use their own knowledge to adjust possible annotations whenever necessary.

2.2 Key Features of the Interface

Some of the main features of the interface design are listed below.

- Each section of information about a possible annotation is encapsulated so that it is obvious to the user what the information relates to. This is also aided by clear headings above each section of information.
- The information related to each class can be edited by changing the value of the annotation or by selecting a different class for it to relate to.
- Properties may be added to an individual class by selecting the property value and the type of property.
- The controls of the system have simple names and are located together at the bottom of the interface. This is a logical position for the controls as the user will need to process the information above the controls first before using them.

An example on semantic tagging ontology is shown in Fig.2.

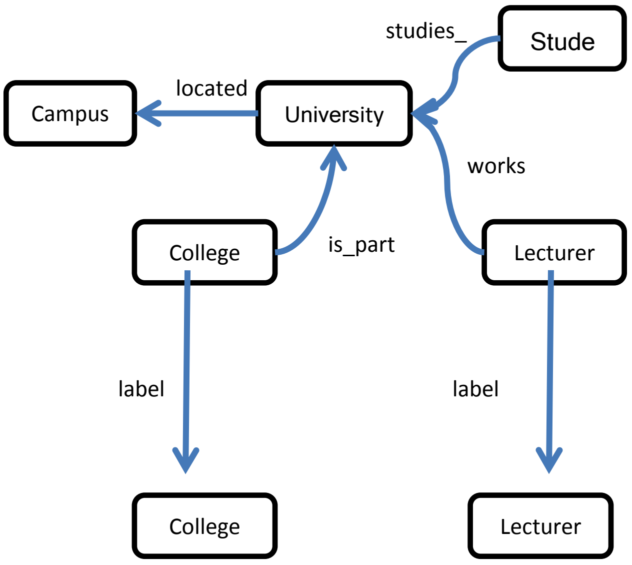


Fig.2: Semantic tagging ontology

2.3 Method

The method focuses on representing the documents succinctly and explicitly through extracting only the related resultant semantics from the document. The methods also consider integrating the semantic annotations within Ontology which allow distinguish between the same words in different contexts that gives different meaning. For example, the chosen texts about *Ford motor company* have
Ontology to express and represent the concepts presenting in the specified domain related to their relationships and attributes. The specific domain ontology will assist the extraction process. The guidance to the modelling process and decoupling of the knowledge base from the required documents is provided by the proposed framework. The following methods are used in the system:

1. LoadOntologyFromURI
   This method loads the ontology from the URI specified and extracts the class information into the JTree in a hierarchical manner.
2. LoadOntologyFromFile
   This method loads the Ontology from the specified file in the local system and extracts the class information into the JTree in a hierarchical manner.
3. ShowClass
   This method will extract the class names in Ontology document in a heirarchichal order.
   This method uses Jena library to extract knowledge from Ontology document.

4. LoadURL
   LoadURL method downloads the contents from the HTML page and shows it to the user in a JEditorPane.
5. AnnotatePage
   This method makes entry into the second JTree for all true results from the search method. Search Method is used to search for each class description in the JTree to find any match in the loaded web page.
6. SearchObject
   This method searches the whole content loaded in the JEditopane for each object or phrase passed as parameter. It will return true if a match is found.

3.CHOICE OF PROFRAMMING LANG - GUAGE

The logical choice for the type of programming language to develop the system was an object oriented one due to the use of an OOD methodology [5]. From early on in the design stage it was decided that the automatic annotation system would be created using Java. By choosing to use Java, a range of benefits were gained including:

- Simplicity - Java has been designed to be compact and does not include some features found in other languages like pointers and multiple inheritances. Also the Java Virtual Machine automatically handles many complex issues like garbage collection. As a result Java is an easy language to use for building small and large-scale systems.
- Platform Independence - One of the key design features of Java is that it is platform independent. Java source code is turned into simple binary instructions like other languages such as C++. However, whereas C++ source is refined to native instructions for a particular family of processor, Java source is compiled into a universal format (bytecode) that is executed by the Java Virtual Machine [5]. Java also familiar in implementing Jena source code. Jena API is fully compatible with RDF data model and OWL ontologies and so this would provide an efficient and well-defined mechanism for creating and manipulating ontologies written in OWL.
- Security-The safe language to be used is Java. It offersnumerous layers of protection from dangerously faulty code including compilers and a bytecode verifier that ensure applications built with Java only execute legitimate code [6]. The language is also type safe, meaning that an object cannot be mistakenly viewed as an incompatible type.

Like any programming language, Java does have some disadvantages. The main one being speed of execution. This is mainly due to the architecture of Java and the many security mechanisms it provides. Despite this, the author feels that the benefits provided by Java clearly outweigh any disadvantages.

3.1 Code Conventions

Standard code conventions were adhered to in the implementation stage for a number of vital reasons. Firstly a standardcoding styleimproves thereadability of
software, allowing other people unfamiliar with the software to fully understand it quicker [6]. The majority of software is developed by the author, so a standard coding style helps increase maintainability as code is clearer and simpler to understand. The code conventions used are subdivided into separate categories and a few of the essential conventions are listed below:

Naming Conventions

- The class names have to be nouns. Where in varied case the first letter of each internal word capitalized. They should also use whole words and be simple and descriptive [4].
- Methods have to be verbs. Where in varied case the first letter lowercase and the first letter of each internal word capitalized [4].
- Variables should be short yet meaningful, in varied case with a lower case first letter [4].

Comments

- Comments could be provided to give indications of code and make available further information that is not available in the code itself. Comments should contain only relevant information to understand the program[4].

Programming Practices

- Variable to be assigned to the same value in a single statement must be avoided.
- Coding literals directly must be avoided except the numerical constant 1, 0, and 1. These constant may appear in for loop for counting purpose.

3.2 Integrating with WordNet

This sub section outlines how the automatic annotation system was integrated with WordNet. To provide an automatic annotation system, the extracting knowledge capabilities of WordNet along with the interface and associated classes developed by the author had to be integrated with Ontology. This would allow the automatic annotation system to make use of the powerful ontology browser and web browser provided by Jena API. These are fundamental parts of the system as they allow the annotator to view web pages and also provide a graphical representation of ontologies.

One of the main benefits of WordNet for developers is its plugin interface. This provides a simple way to interact with the various components of the system and also allows the system to be easily extended in the future. Each of the components of WordNet has a specific plugin interface that can be used query the component for information.

3.2.1 ExtractingKnowledge based Ontologies

The focusing of semantic annotation was in isolated annotations of web pages. However, achievement has been done on annotation of pages with semantic information by semantic web in order to enrich the content of web pages.

The existing information on the web is as natural language documents. IE provide a promising approach to access this knowledge with reducing the documents to tabular from the documents to be retrieved by clients. However, such methods still remains not easy for practical purposes due to time consumed and efforts required to achieve annotations. The certain entities in any text documents are identifying by many IE systems depending on predefined templates.
However, there is a limitation in using vocabulariesstructures by web document and it is very difficult for any IE systems to overcome this challenge.

In this sub-section description on how the extracting knowledge component was implemented. During the design phase of the work, it was decided that WordNet would be used to extract knowledge from documents. The program code used to control

WordNet based on parse tree. An example code fragment to show how to check and create a WordNet database is shown in Fig.3.

```
System.getProperties().put("wordnet.database.di
r","C:\\Wordnet\\2.1\\dict");
BrowserFramefrm= new BrowserFrame();
frm.setVisible(true);
publiccheckWordNet(String wordForm) {
WordNetDatabase database =
WordNetDatabase.getFileInstance();
Synset[] synsets
=database.getSynsets(wordForm);
```

Fig.3: Code Fragment to Check a WordNet Database

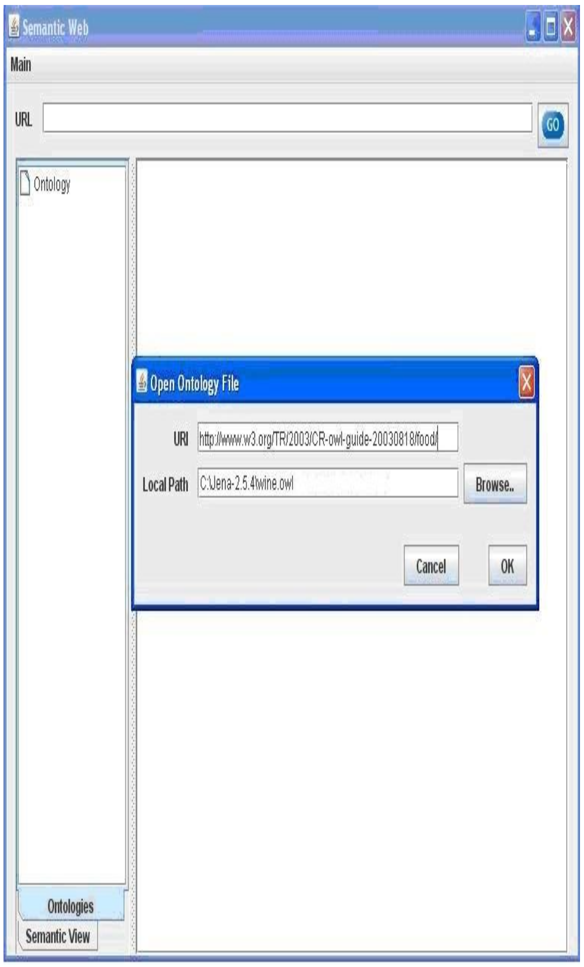
3.2.2 Loading Ontology

There are various methods to write down the ontology, and different thoughts as to what breeds of definition should go in one. The kinds of the application will guide and drive the contents of the ontology. In this section, explanation will be provided on how to load the ontology.

Reusing and sharing of ontologies is supported by OWL through making it portable for one ontology to import another one. All of the properties, classes and individual definitions which are in the imported ontologies are accessible to be used in the importing ontology. The mechanism of how owl:imports will work to be compatible to resolve the position of the required ontology and provide its URI
Ontology in this method is load from the URI or from the specified file in the local system specified and extracts the class information into the JTree in a hierarchical manner. The screen shot of how to load Ontology is shown in Fig.4.

3.2.3 Controlling Tree Model

In order to use WordNet to analyze a specific document, a tree model must be created. This model is provided by the framework and it is used to set entity. An example code fragment to show how to create a new Tree model is shown in Fig.5.An instance of the tree model is created using the import om.hp.hpl.jena.rdf.model.ModelFactory, the method specified by the abstract Model Factory. This approach is commonly known as the factory pattern. It assists to model an interface for generating an object which at formation time can let its subclasses choose which class to instantiate.
This mechanism is widely used in Java programming as it allows classes to be instantiated using different implementationsOver time specific implementations may be updated and new ones may be added to improve the functionality of a system.



URI http://www.w3.org/TR/2003/CR-owl-guide-20030818/food/

Local Path C:\Jena-2.5.4\wine.owl

Fig.4 Load Ontology Screen Shot

The parameters supplied to the tree model method are the resource class name, the parameter values for the resource, the features of the resource and the name of the resource which is the version of jena.rdf supplied in the Framework being used.

```
jSplitPane1.setDividerLocation(150);
jTabbedPane1.setTabPlacement(javax.swing.J
TabbedPane.BOTTOM);
trvOntology.setModel(newDefaultTreeModel(ne
wDefaultMutableTreeNode(newannot
("Ontology"))));
trvOntology.addMouseListener(new
```

Fig.5: Code fragment to create a new Tree model

The parameter vales and the feature values are actually just empty DefaultMutableTreeNode objects as these are updated later in the execution process. The screen shot below shows how to extend tree as shown in Fig.6.

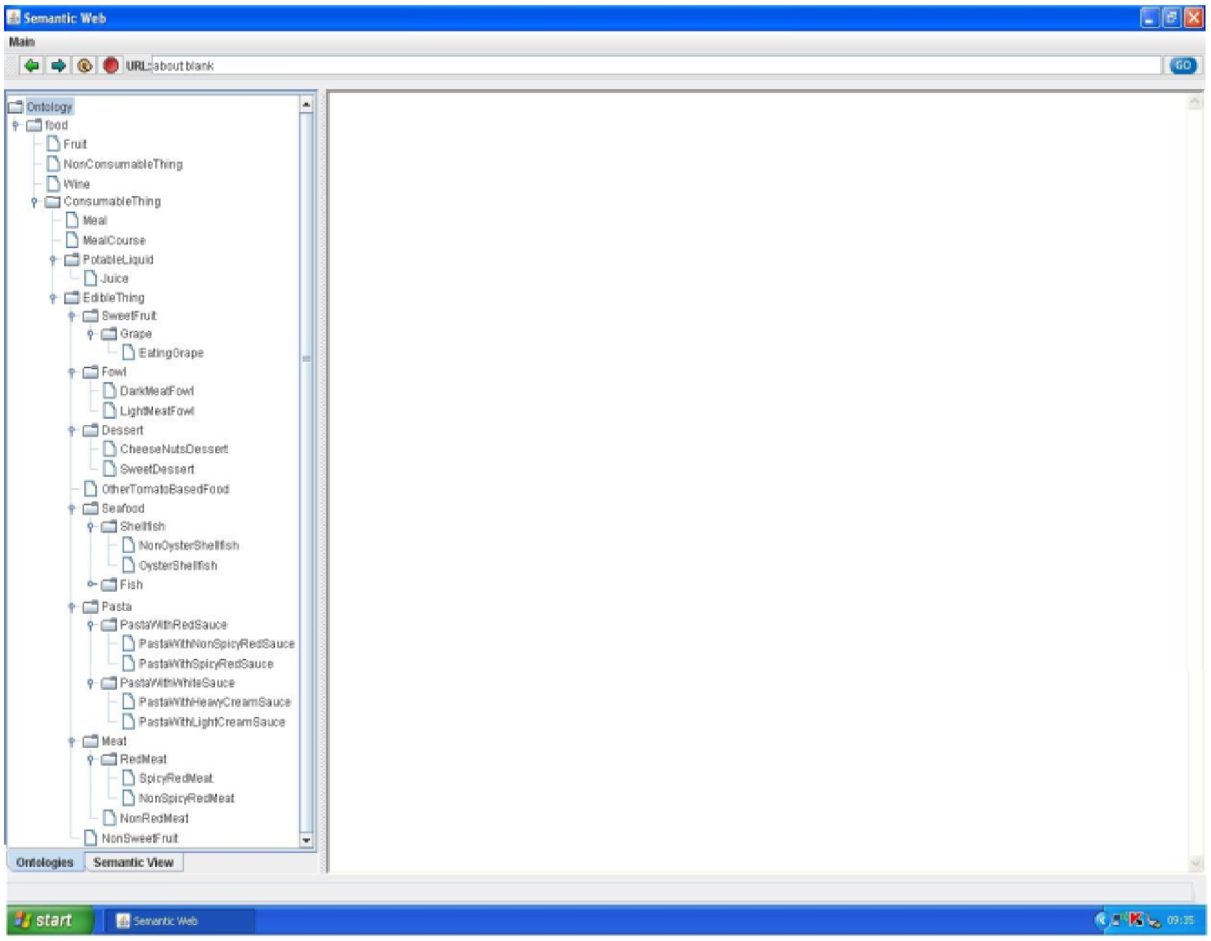


Fig.6: Ontology Tree Screen Shot

4. CONCLUSION

During the last few years, many formalization of WordNet as Owl ontology have been developed but the complete number of resulting concept classes is the main challenge to turn WordNet into OWL ontology.

This paper describes how to support the annotation processes through developing method to achieve the following:

- Collect sentences for each noun pair where the nouns exist.
- Extract patterns automatically from the parse tree and parse the \sentences.
- Train a hypernym/hyponym classifier based upon these features.
- Dependency tree considering the following relation: (word1, category1:Relation: category2, word2).

Theproposed method is focusing on representing the documents succinctly and explicitly through extracting only the related resultant semantics from the document. The utilization of WordNet improves the retrieval of information. The specific domain ontology will assist the extraction process. The guidance to the modelling process and decoupling of the knowledge base from the required documents is provided by the proposed framework.

5. REFERENCES

[1] Ala'aQasim Al-Namiy, (2009); "Towards Automatic Extracted Semantic Annotation (ESA)", IEEE International Conference on Information Processing, 2009. APCIP 2009. Asia-Pacific, Shenzhen, China. Conference on. Issue Date: 18-19 July 2009. Volume: 2, Page(s): 614 – 617.

[2] McEnery, A. M., Wilson, A. (2001) *CorpusLinguistics: An Introduction.* Edinburgh: Edinburgh UniversityPress.

[3] John M. Carroll, (2010); "Conceptualizing a possible discipline of human–computer interact-ion", Journal of Interacting with Computers, Vol. 22, Issue 1, January, pp. 3–12.

[4] J.C.C, (1997) ; Java Code Conventions, http://www.ingce.unibo.it/~mviroli/teaching/fondLA2003/slides/CodeConventions.pdf

[5] J. Domingue, M. Dzbor, and E. Motta. Magpie: Supporting Browsing and Navigation on the Semantic Web, Proceedings ACM Conference on Intelligent User Interfaces (IUI), pp. 191–197, 2004.

[6] Deng ZhiHong, TangShiwei, Yang Dongqing,Semantic-Oriented Integration The role ofOntology in Web Information Integration,Computer Applications, 2002, vol.2, no.1,pp. 15-17.