# Towards Exploring Semantic Similarity based on WordNet Semantic Dictionary

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

## ABSTRACT
The main challenge in the area of Information Retrieval (IR) and Natural Language Processing (NLP) is the characteristics of synonymy and polysemy that exist in words of natural language. The capability of natural language interfaces to the semantic search engine can be improved by both the knowledge extraction and semantic data. The combining of information can make the integration and sharing of distributed data sources easily. This will assist the user to have the required information efficiently and easily.

In this paper, some concerns of evolving algorithm to capture the semantic similarity among sentences based on WordNet semantic dictionaryis presented. The proposed algorithm will be relying on a number of resources including Ontology and WordNet.

## Key words
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 [2].

The SW technology provides a countless support for the computer to realize, represent and structure the data, on the web, with the various annotating tools available. However, most of them are semi-automatic and are not easy to use by non-technical users, who are unfamiliar with the syntax of the language [1].

However, the user can utilize the data successfully with the assistance of the semantic Web annotation technique. One of the main methods used to create metadata is the Semantic Web**.** The improving of web searches and assisting the searching process in order to sense data on the
web is the main aim of using this technique. The semantic web map is shown in Fig.1.

The main challenge in the area of Information Retrieval (IR) and Natural Language Processing (NLP) is the characteristics of synonymy and polysemy that exist in words of natural language [2].

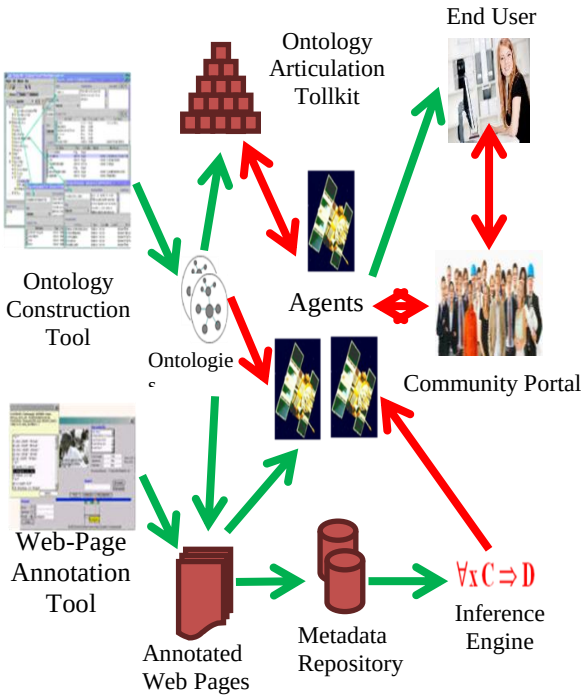


**Fig.1: Semantic Web Map**

The capability of natural language interfaces to the semantic search engine can be improved by both the knowledge extraction andsemantic data. The combining of information can make the integration and sharing of distributed data sources easily [10]. This will assist the user to have the required information efficiently and easily. The automatic annotation system allows an annotator to create new annotations fora specific web page automatically byusing Knowledge Extraction techniques to generate possible annotations.

In this paper, some concerns of evolving algorithm to capture the semantic similarity among sentences based on WordNet semantic dictionary. The proposed algorithm will be relying on a number of resources including Ontology and WordNet will be presented.

## 2. METHODOLOGY

The NLP technology is considered as one branch of the linguistics [5]. It applies the technology provide by the computer to recognize human language processing efficiently. The possibility of semantic data store and multilingual ontology mapping was made with NLP. The relationships of the ontology and NLP offer a chance that Wordnet senses would match the results in a given query.

The process of finding numerically the way of semantically related two words in natural language processing (NLP) is one of the applications used to calculate measures for it. The definition of semantic measures can be used for defining between full tests or between lexically expressed word senses. There are three different types of measures known as [3].

1. Semantic similarity: defined usually by considering lexical relations of synonymy (*i.e.*.<car, automobile>) and hypernymy (the meaning of a word is incorporated by more general term, like <car, vehicle>).
2. Semantic relatedness: it is a general concept than semantic similarity since any type of lexical or functional association can be covered. Unlike entities, manyrelationships may be still related like meronymy (or "part of" relation, as in <noise,mouth>), antonymy (opposite meanings, as <tall, short>).
3. Semantic distance: defined as the inverse of semantic relatedness. This means that two terms are semantically close if they are semantically related.

In the remainder of this paper the explore Semantic Similarity based on WordNet Semantic Dictionary is firstly discussed.
In the remainder of this paper the methodology and approach used in this paper is reviewed in Sec. II and then about the SensesAlgorithm in Sec. II.
In Sec. IV presentation onhow to build the Semantic Contextual Meaning is also presented. In sec. V the process design is discussed. Finally, conclude the work.

## 2.1 Explore Semantic Similarity based on WordNet Semantic Dictionary

The aim of this algorithm is to capture the semantic similarity among sentences based on WordNet semantic dictionary. The four parts of Speech verb, adverb, adjective and noun can be linked together by WordNet which establish the connections between these types. The specific meaning of the word can be represented by synset. The synonyms and word explanation is included in the synset. The sense is the specific meaning of single word under one type of parts of Speech. The group of sensesis equivalent to synsets. The concept is defined bythe gloss owned bythe synset.

For example, the words day, daytime and light constitute a single synset that has the following gloss: the time after sunrise and before sunset while it is light outside. The explicit semantic relations are used to connect synsets each other. Several of these relations considered as a kind of holonymy and a part of hyponym *i.e.* penguin is a kind of mammal, penguin is a hyponym of mammal and mammal is a hypernym of penguin. Analogously, Blubber is a part of a penguin and that blubber may have as meronym of penguin and penguin is a holonym of blubber.

WordNet organizes the senses in the order of the most frequently used to the least frequently used for one particular word.
The algorithm shown in Fig.2 considers the definition of the relations of hyponymy, hypernym and holonym as follows:

- hyponymy "The specific term used to designate a member of a class. X is a hyponymy of Y if X is a (kind of) Y" [WordNet].
- hypernym "The generic term used to designate an whole class of specific instances. Y is a hypernymof X if X is a kind of Y" [WordNet].
- holonym "The name of the whole of which the meronym names a part. Y is a holonym of X if X is a part of Y" [WordNet].

Thus conclude entity *i.e.* word { , word} * { , } and other word, for example hyponym may consider ("mammal", "Penguin"), "Penguin" is a hyponym of "Mammals".

In another description, wordi = Xand word0 = Y→ (X, Y) is proposed, that implies to 'pairs of noun'. A noun X is a hypernym /hyponym of a noun Y if Y is a subtype or instance of X. An automatic classifier for the hypernym / hyponym relation "Arthur Conan Doyle" is a hyponym of "author" and on the opposing "author" is a hypernym of "Arthur Conan Doyle", "desk is a hyponym of "furniture", and so on.

**Algorithm 2** checkWordNet_ BuildMeanning (Synset)
Where Synset[ ] array of synonymsvzxs
String word
loadWordNet Database
Synset[] synsets =
database.getSynsets(word);
// Get word and its definitions for synsets retrieved
if (word == null || word.equals(" "))
return -1;
else
for all wordi , i ≥ 1
if hyponymy (wordi, word0) → hypo
// "The specific term used to designate a member of a class. X is a hyponymy of Y if X is a (kind of) Y" [WordNet]
else
if hypernym (wordi, word0) → hype
// "The generic term used to designate a whole class of specificinstances. Y is a hypernym of x if x is a kind of y" [WordNet].
else
ifholonym (wordi, word0) → holon
// "The name of the whole of which the meronym names a part. Y is a holonym of X if X is a part of Y" [WordNet] .
matcher.appendReplacement (sb, " word ");
matcher.appendTail(sb);
return find;

**Fig.2: Algorithm to Check WordNet and Build Meaning**

# 3. BUILDING SEMANIC RELATIONS ACROSSHIERARCHICAL CLASSIF - ICATIONS

The main feature of this algorithm shown in Fig.3 is to discover semantic relations across hierarchical classifications based on the particulars details explained in [1].

The semantic relation existing between two entities will simply be the output of the algorithm.Subsequently, the Build-Semantic–Relation algorithm is applied to each entity *e* of the spotlight. The reduction of these senses associated to entity which is unfitted with the meaning stated bytheentity e isthemain taskof this algorithm. It uses the Ontology heuristic rules, to obtain the domain information. This information relate to the relations between the entity e and the senses related to other entities in the focus.

The methods in this algorithm (line 1 to 22) determine the properties of each entity with lexical knowledge. The entity

refers to the class of things and sub entity refers to sub class, consider the example mentioned in section 4.3, the method associates entity London (#1) into two senses 'Capital city of England' (#2) and 'city of London, Ontario, Canada' (#3).

The structure knowledge also contains the entity province name Ontario (#4), which is an ancestor of city (#3) and the European state entity England (#5) which is an ancestor of city (#2). From above the algorithm is able to indicate that the entity London (#1) has a sense city name England (#5), for which the relation England is the hyperonym of London' by analyzing the information about the sense of an ancestor entity holds, meaning that 'London is in England'. The R–ONTOLOGY method allows accessing and navigating Ontology toward discovering relations between senses. As a result, the word 'London' denotes to the 'city in England' and not the 'city in province Ontario'.

The conclusion of this algorithm that the linguistic mediation represented by human lead to use WordNet as a shared structure.

**Algorithms Conclusion**

In these algorithms, a batch of methods has been developed to support the annotation processes as follows:

- 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)

**Process Design**

It has been observed that it is easily sharing both the results of the concern process and pre-populated knowledge using NLP systems with Ontology support. The most important issue is to focus on the varying of the distribution of entities from domain to domain [2].

The automatic annotation system allows an annotator to create new annotations for a specific web page automatically

by using Knowledge Extraction techniques to generate possible annotations.

## 3.1 Implementing the Algorithm

In this stage, the most important step is to find the word description, using WordNet which provides complex word descriptions. In this work the description of a word can be a textual definition, more general terms, more specific terms, or a definition in a specific language or domain.

To implement the suggested algorithm Ontology needs to be created. It could be produced using the Jena Framework, as Jena is able to query and store Ontology [2,7]. Furthermore, Jena method of OntDocManageraddAltEntry enables relationships between stored Ontologies, thus identifying the location of Ontology inside the database.

**Algorithm 3** Build-Semantic–Relation (*IST,GO, SynSet, SUBF*)

*where IST is internal structure*
*GO is a General Ontology Concepts*
*SynSet[ ][ ] array of synonyms*
*SUBF is substructure of IST*

*entity e is a Generic Word*string *x, y*

**Variable Identifications**

Relation R, $R^1$ **,** $R_x$ **,** $R_y$*initialized to Null*

Where

R → inherits all properties of Ontology concepts

$R^1$ → inherits all properties of Ontology concepts in reverse order

Sentence Synset[ ][ ]

1 **for** each entity *e* in SynSet[i] do
2 **for**each ancestor i in IST **do**
3 R- ONTOLOGY(*e, GO*) →R // Read Ontology
4 **if** hype (*x,y*) → R // Access hypernym
5 **else** R1=Null
6 **if** hypo (*x,y*) → R // Access hyponym
7 **else** R2=Null
8 **if** holon (*x,y*) → R // Access holonym
9 **else** R3=Null
10 **if**(($R_1$≠*Null* & R=hype) Λ ($R_2$≠*Null* &*R*=hypo) Λ ($R_3$≠*Null*&*R*=holon)) then
***return*** *Property R*(x, y) *i.e* return x ⊂ y ;
11 **else**remove *entity* from *SynSet*[i]; //inherits all the properties of Ontology concepts in reverse order
12 **for** each entity e in SynSet[i] do
13 **for** each descendant *i* of *SUBF***do**
14 R- ONTOLOGY(*e, GO*) →$R^1$ ; Read Ontology
15 **if** hype (x,y) → $R^1$
16**else** R1=Null
17 **if** hypo (x,y) → $R^1$
18**else** R2=Null
19 **if** holon (x,y) → $R^1$
20**else** R3=Null
21 **if** (($R_1$≠*Null* & $R^1$=hype) Λ ($R_2$≠*Null* & $R^1$=hypo) Λ ($R_3$≠*Null* & $R^1$=holon)) then
***retur***n$R^1$ (*y, x*) *i.e.*return*y*⊂ x // Return properties in reverse order
22 **else**remove*entity* from *SynSet*[*i*];

**Fig.3: Algorithm to build Semantic Relations**

The implementation procedure starts by reading a RDF/OWL document into a Jena model; that gives an API for handling the information. Once the description of a word is recognized in the RDF document, the word available in a HTML document will be highlighted / underlined which in turn shows the description extracted from the RDF/OWL document. For example 'Dr. Tony Beaumont' is identified in HTML page and the description of 'Dr. Tony Beaumont' is available in RDF document as shown in Fig. 4.

From the above example, the following deduction should be available:

- An HTML page P with some term T of interest.
- An RDF document R which describes the term T from a set of one or more RDF descriptions.

To find T of interest in P, the system parse P that says "the term T appears on this page will be highlighted, italics, colour, and/or larger font *i.e.* Dr Tony Beaumont "as mentioned in example above. To find R of T if that term has RDF description, then itdisplay the properties asannotation in a new

page. Once parsing the HTML page to annotate 'Dr Tony Beaumont', it should:
- display the description of 'Dr Tony Beaumont', the author and date of creation for that page in a tab or new window.
- produce a new HTML page which is P with T annotated
- store specific metadata in a specific Ontology which will be achieved by adding new annotation rules.

This work handling Jena integrated with SPARQL [9] to create a rule-based system through GeneriRuleReasoner to store the derivation data. The reason for this is to answer user's queries about the derivation of derived statements.

```
<rdf:RDF
xmlns:rdf="http://www.w3. org/1999/ 02/22-rdf- syntax-ns#"
xmlns:lib="http://keg.cs.aston.ac.uk/stfDtls/">
<rdf:Description about="Dr Tony Beaumont" >
<lib:creator>Alaa Al Naimy</lib:creator>
<lib:pages>10< /lib:pages>
</rdf:Description>
```

**Fig.4: Example of Implementation Procedure**

The system stores the derivation data in the database as the reasonerrun [3]. SPARQL considered as a query language for getting information from RDF graphs [Jena]. It provides the requirements for querying by triple designs, optional patterns, disjunctions, conjunctions and supports queries like "*show me all the projects on semantic annotation*". The *projects* will mapped by the semantic annotations. The resultant data from the project database related to the semantic annotation area will be used to identify only *projects.* The SPARQL queries results can be obtained and presented in several different forms [8].

Integrating semanticannotation within Ontology allows distinction between the same words in different contexts that give different meanings *e.g.* the searching process will be easy to distinguish the word "Mississippi" the state, from "Mississippi" the river because the annotation will be with references to various concepts in the Ontology. This will improve the current information retrieval search anomalies.
This work brings in user application that annotates web pages of user choice and connects to an annotation server. The annotation server uses an Annotation engine with an embedded Jena repository, which then transfers the results of the annotation to the annotation server.

**T**he strength of this implementation is to integrate a Knowledge Extraction platform and Ontology which provides flexibility for the formats and methods uses [4]. It also supports the HTML browser to display an integrated open APIs of the Ontology browser along with the documents.

**System Functionality**
From looking back at the functional requirements [1]it is clear that the system needs to provide the following functionality:

**Open an Ontology**
This method read Ontology from a file or URI and then displays the Ontology to the user. The Ontology should be validated before being displayed to make sure it is a legal Ontology that makes sense.

**Open a Web Page**
This method display a web page to the user for a specific URL entered. The URL should be validated to check if it is a legal URL.

## 3.2 Extract Knowledge from Web Page

The system import semantic package and loads WordNet Database to get File Instance. This method extracts information from a specific web page. The main components of WordNet (Token, Scanner, pointer, Lexer Generator) will functional to extract knowledge and take responsibility to convert a series of characters into a serious of tokens. The information will display to the user via the interface.

**Automatically Generate PossibleAnnotations**
Once the system is started, it loads semantic package and uses it to extract knowledge from the current web page as shown in Fig.5. As only one web page may be annotated at a time, WordNet should only ever be loaded once. It would be very inefficient to have multiple copies of WordNet loaded into memory, therefore the system checks to see if WordNet has already been loaded and if so it moves straight onto the Knowledge Extraction phase. If an error occurs trying to load WordNet, a suitable message is output stating that WordNet could not be loaded. Similarly if an error occurs whilst extracting information, a message is output stating that the Knowledge Extraction could not be completed as shown in Fig. 5.

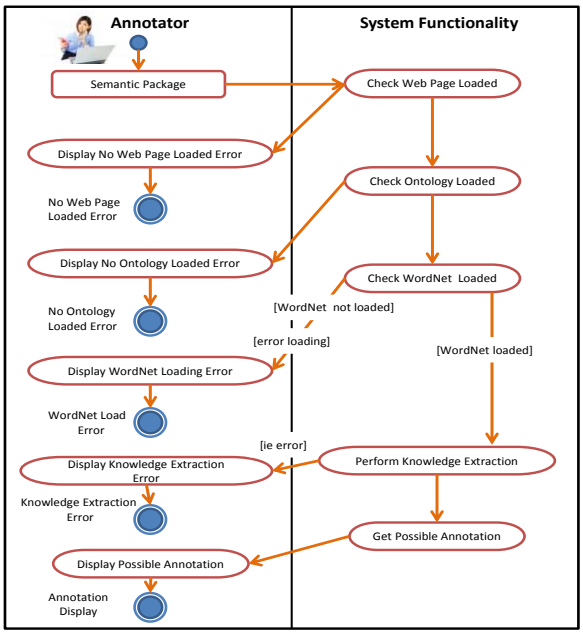


**Fig.5: Activity Diagram for generating possible annotations**

**Post Generated Annotations**

This method allows the annotator to create actual annotations from the possible annotations generated by the system shown in Fig.6.

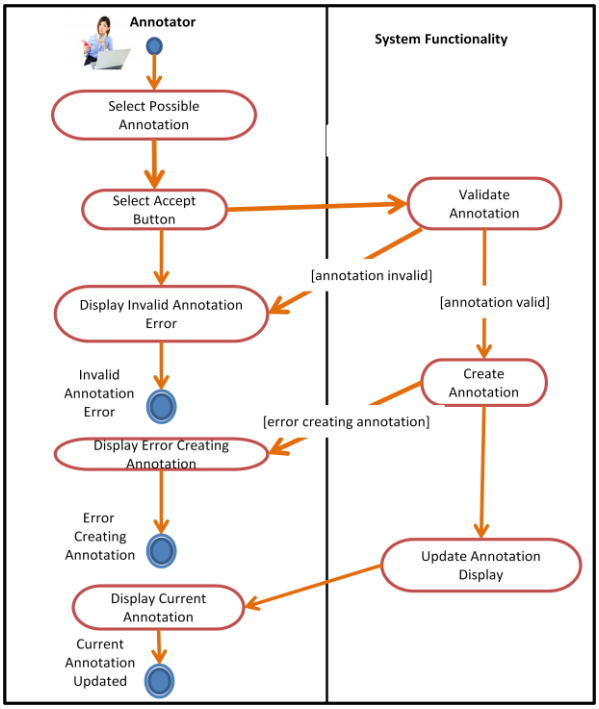


**Fig.6: Activity Diagram for Accepting Possible Annotations**

The annotator must first select one or more possible annotations and then select the "Accept" button. Each selected annotation is validated first to make sure it is valid. If it is invalid, an error is displayed to the annotator but if it is valid the annotation is created and the annotation display updated. The annotation is actually created by writing a new instance of a class or property in RDF. This instance will utilize the vocabulary defined in the Ontology currently being used. Due to ontological commitment, only an instance of a class that is already defined in the current Ontology being used can be created. Also any properties created must already be defined for the specific class instance they are being added too.

If an error occurs whilst creating the annotation, a suitable error message is displayed to the annotator. This method allows multiple annotations to be accepted at the same time by repeating this process for each annotation selected.

**Reject Annotations**

This method allows the annotator to reject possible annotations generated by the system shown in Fig.7. The main benefit of this method is to remove any possible annotations that the annotator does not think need to be created.

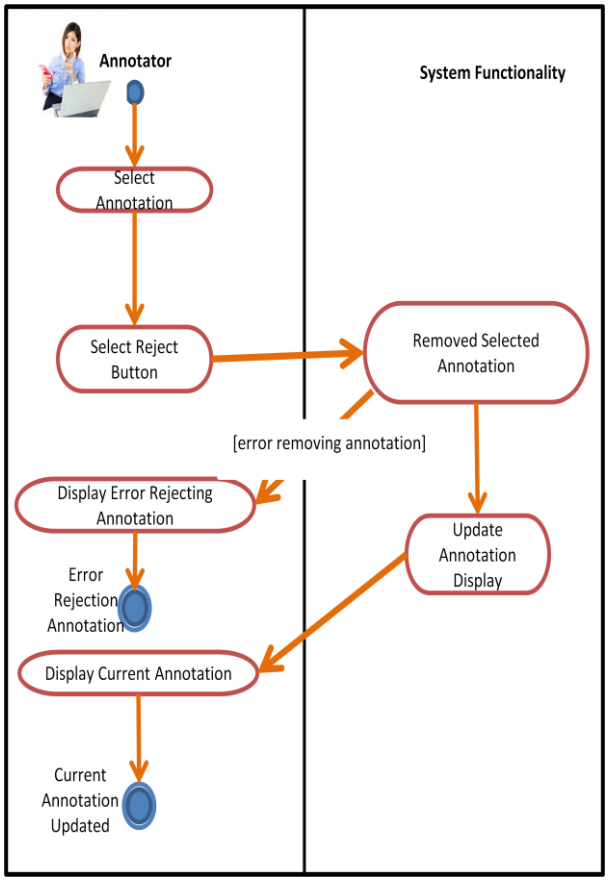


**Fig.7:Activity Diagram for Rejecting PossibleAnnotations**

For a large web page, many possible annotations may be suggested. Therefore, this method lets the annotator remove information that is deemed not useful. The annotator must first select one or more possible annotations and then select the "Reject" button. Each selected annotation is removed from the current list of possible annotations. If an error occurs whilst trying to remove a possible annotation, a suitable error message is displayed to the annotator. The annotation display is then updated to reflect the changes made. This method allows multiple annotations to be rejected at the same time by repeating this process for each annotation selected.

**Save Annotations**

This method allows the annotator to save any created annotations. The user must first select the "Save" button. A save dialog box is displayed to the annotator asking them to enter a filename and select a location to save the annotations. The annotations that have been previously accepted are then saved. This is achieved by writing the RDF already created by the system to represent the annotations to a specific file. If an error occurs whilst trying to save the annotations, a suitable error message is displayed to the annotator. The procedure to implement save annotation is shown in Fig.8.

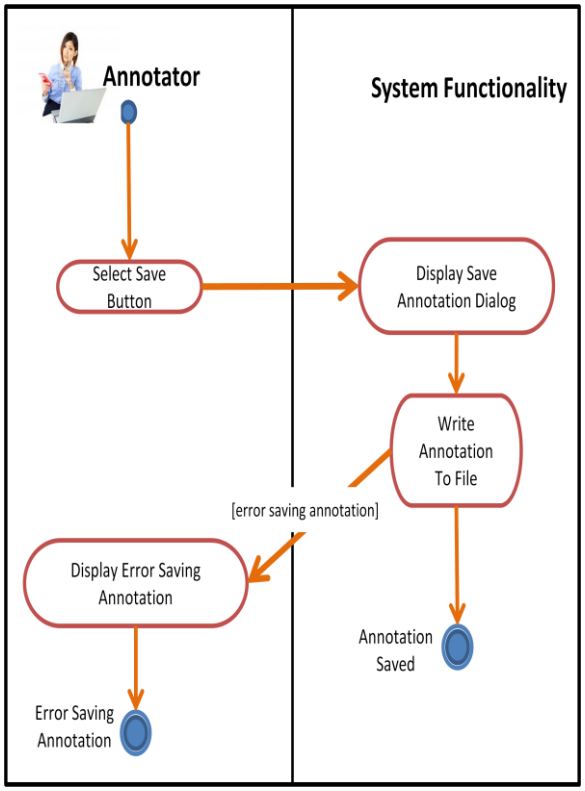


**Fig.8: Activity Diagram for Saving Annotations**

## 4. Conclusion

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).

Our method focuses on representing the documents succinctly and explicitly through extracting only the related resultant semantics from the document. 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] AtanasKiryakov, Borislav Popov, Ivan Terziev, DimitarManov and DamyanOgnyanoff, (2005); "Semantic Annotation, Indexing, and Retrieval ", Elsevier's Journal of Web Semantics, Vol. 2, Issue (1), 2005.

[3] Brian McBride, (2002). "Jena: A Semantic Web Toolkit", Journal of IEEE Internet Computing, Vol.6, no.6, November/December 2002, pp. 55-59.

[4] Budanitsky, A., Hirst, G.: Evaluating WordnetBased Measures of Semantic Distance. Computational Linguistics 32(1), 13–47 (2006).

[5] Guo and Ren. Towards the Relationship BetweenSemantic Web and NLP, Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering, 2009. NLP-KE 2009, (Sept 24-27),Dalian, China, pp. 1-8.

[6] Jung AeKwak and Hwan- Seung Yong, "Ontology Matching Based On Hypernym, Hyponym, Holonym, and Meronym Sets in WordNet", International Journal of Web & Semantic Technology (IJWesT), April 2010.

[7] McBride, B., (2001). "Jena: Implement-ing the RDF Model and Syntax Specification", Hewlett Packard Laboratories, Filton Road, Stoke Gifford, Bristol, UK. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol.40/mcbride.pdf

[8] Stefan Bischof, Stefan Decker, Thomas Krennwallner, and Axel Polleres, (2012); "Mapping between RDF and XML with XSPARQL", Journal on Data Semantics, Vol.1, no.3, September, pp. 147-185.