International Journal of Information Technology and Management(IJIT) Vol. 1, No. 1

Towards From Manual to Automatic Semantic Annotation: Based on Ontology Elements and Relationships

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

ABSTRACT

Automatic annotation is offering a standards base for retrieving information from web services. It has been observed that many existing protocol e.g. Annotea did not support the fully automatic annotation directly or the process to be carried out needs professional developers (i.e. non-trivial protocol) such as KIM.
In this paper a description of the architecture of the proposed system is given and a figurative structure is supplied. The diagrams that represent the structure will be described along with the main resources usage.

Index 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.
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. 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 figure 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.

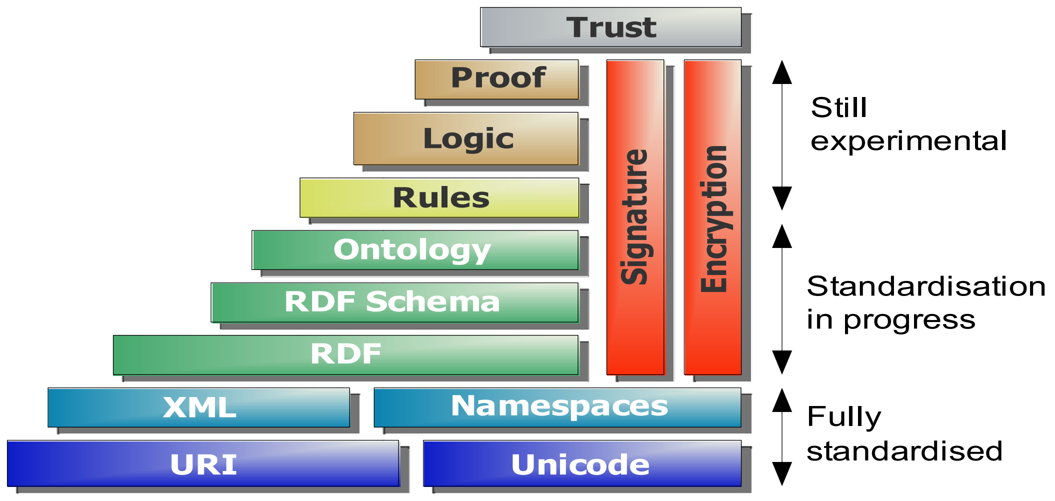

Figure 1 Semantic Web Stack

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. 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.

In this paper, we will present 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.

2. PROPOSED SYSTEM ARCHITECTURE

The objective of this research is to create an effective automatic annotation platform which develops a way of automatically generating metadata to semantically annotate web documents which improve information retrieval. The proposed system must be easily understood by non-technical users who may not be familiar with the technical language used to create Ontologies [11].
The proposed system provides an ontological similarity to determine the relations between words in sentences and concepts in Ontology. I found out that the significance meaning of the term similarity is ambiguous because of its use in many diverse contexts, such as biological, logical, statistical, taxonomic, psychological, semantic, and many more contexts, to solve the ambiguities, WordNet must be used to provide a lexical Ontology.

The suggested automatic annotation architecture [2] is shown in Figure 2. It shows the layout of the main components of the proposed system and also the input and output data.

This component will be the foundation of the system. It should provide all of the functionality required to create annotations automatically. This will include viewing Ontologies and browsing web pages. This component of the system will analyze web pages and extract specific information found within the text. It will be developed using features provided by the Jena Toolkit.

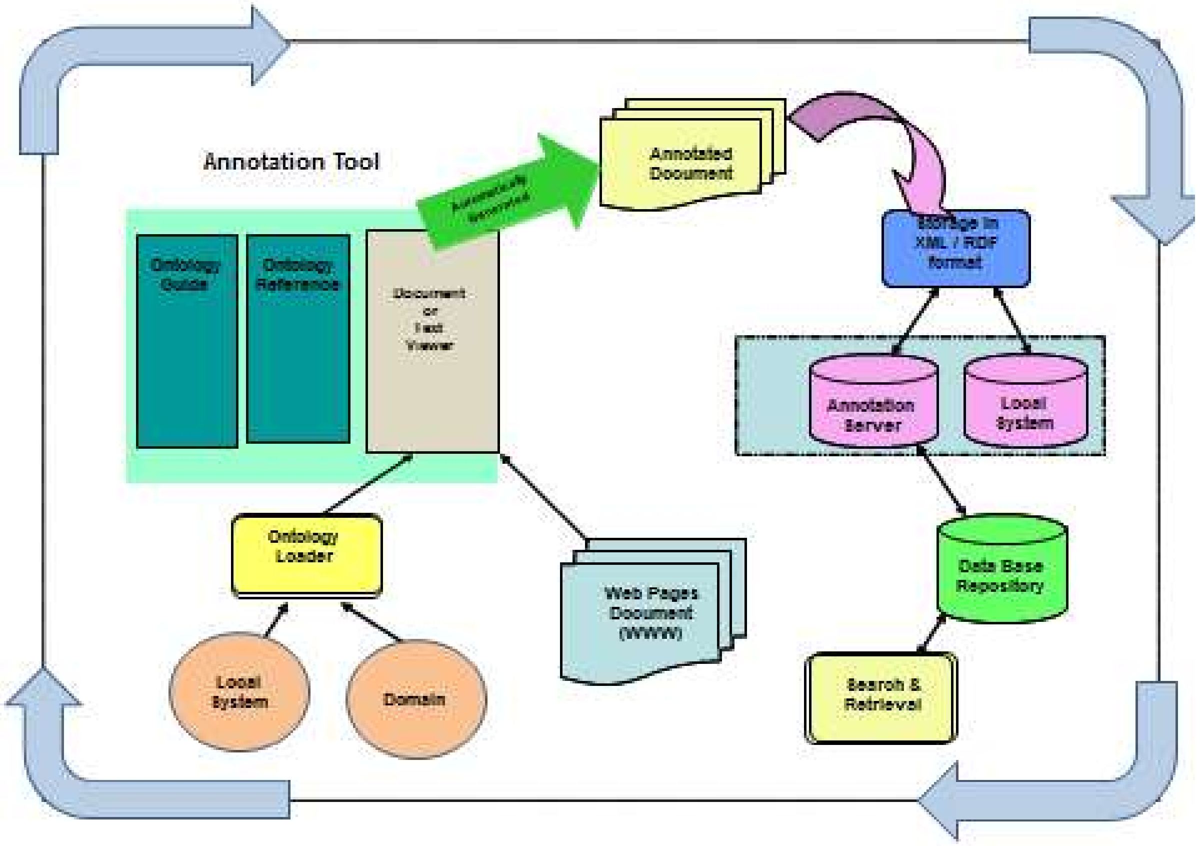

Figure 2 Architecture of the Proposed System

To fulfill the requirements of annotation process the proposed system will start with analyzing, parsing and extracting specific information found within the text and automatically link it to related Ontology (*i.e.* classes, entities and instance) and their properties. A number of general classes will be addressed in such a way that enables the Ontology to offer support to the web context. These classes will appear in different domains in text forms. The upper level Ontology will be involved to provide description to these classes along with their fundamental attributes and relations. The description of the entity is considered as a future of the representation of the Semantic Web after providing the relations, attributes and entity types encoded in the Ontology. Through identification, description and interconnection of the entities, this will make them flexible and be in standard format.

The verification of the results proposed by the parser will be expected by the system that gives a record of the properties suggested by the Ontology content. The proposed system is considered to be efficient as it is a self-learning system. The annotator has the privilege to add a suitable concept or instance in case some of them are not defined in the Ontology. The steps for the annotation in this system are shown in Figure 3.

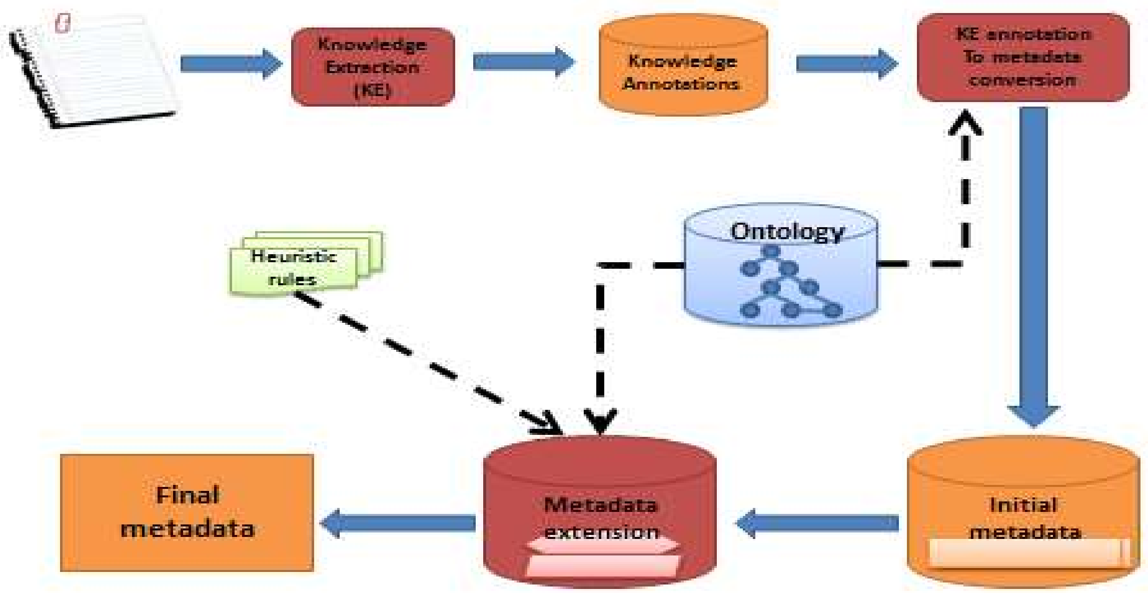

Figure 3 Steps of the Annotation Process

In order to realize this design an algorithm will need to be created.

- The significance of the algorithm (this is described in more detail in chapter four)is that it will automatically infer the meaning of the information from text that contains the search words in any text as an outcome of the input resources *i.e.*:
- Text, sentences as seen in traditional web pages - that are scanned using WordNet.
- Ontology document: representing knowledge - Entities (*i.e.* organization, companies, places name, people, and numerical expressions) and their relationships are described in RDF(s) within a domain. The Ontology documents are arranged hierarchically with rich semantic relationships between terms and attributes which organize information within the database. Entities are typically tagged, both syntactically and semantically.
-

The basis of this structure is to match the same topic (words or concepts) in the sentence with entity/sub entity in the Ontology document.

2.1. Resources

The main purpose of developing an efficient algorithm that will extract knowledge from text using Ontologies and WordNet, is that it will:

1. Provide a formal definition of the Ontology component " *i.e.* entity" based on class hierarchy;
2. Research ways of scanning text in order to extract knowledge from a traditional web page using WordNet;
3. Compare the Ontologies component "*i.e.* entity" within the information that was extracted from the text;
4. Describe and identify the relationships between the Ontology concepts, entities and objects that the proposed system will rely upon to produce automatic annotations;
5. Convert the result into knowledge.

The significance of such an algorithm is that it will automatically infer the meaning of a piece of text. The Ontology will allow us to link that text to other sources of information from the Semantic Web. Therefore both WordNet and Ontology resources are required to underpin this research.

2.1.1 Ontology Elements and Relationship

The representation of Ontology in this system signified a specific vocabulary that provides an importance meaning planned describing certain reality, plus a set of clear assumptions about the vocabulary words [13]. It also deeply relies on the Ontology approach to organize the fundamental data for wide-ranging and machine understanding [2].

Ontology editors vary in their range of functions, which are themselves dependent on the following factors:

a. the supported ontology language(s),
b. the underlying knowledge model (e.g. based on frames or description logics),
c. single-user or multi-user mode, x web-based or locally installed application, x commercial or free tool,
d. desired interoperability with other tools (e.g. merging and reasoning tools).

Among the classical tools for ontology engineering we find those that have to be locally installed as well as Web-based editors. Many of them support little more than one specific ontology language and are designed for individual users or groups – but not for broad Web communities. Several tools are freely available, which often also means that they are being developed as part of a certain research project. They are thus in some cases still under construction, or in other cases there is no longer any user support or maintenance. The most commonly needed methods in ontology engineering, and thus typical components of ontology editors, are:

a. Basic editing functionalities: creating, deleting and renaming concepts, instances, properties, comments and restrictions.

b. Import and export functionalities: saving and storing ontologies in different file formats and importing ontologies in certain formats for editing and modification.

c. Import and export functionalities: saving and storing ontologies in different file formats and importing ontologies in certain formats for editing and modification.

The main components of entities and relationships of Ontology will be the foundation of the proposed system. The entities that form the nodes of Ontology are referred to as concepts, classes and their instances. The relationships are known as properties, relations, or roles [10]. They are also referred to as attributes. Relationships will be used to make statements that specify associations between entities in the Ontology. We consider that each of the entities and relationships have a unique identifier. There are many factors to be considered in developing the Ontologies such as:

- Choice of Domain
- Consider Reuse
- Observe main factors
- Defining Classes with the properties and Class Hierarchy
- Create Instances of classes

It is assumed the ontology is available to build the knowledge extraction which is required for annotations *i.e.* Ontology needs to already exist in the proposed system. In the main, a combination of entities and relationships (nodes and edges) can be considered as a directed acyclic graph. We utilize Ontology in the environment of sharing information as a pattern of a conceptualization exacting the domain of interest. Whereas conceptualization may be implicit or explicit and deals with objects, attributes and relationships [16].

Ontology holds a limited list of terms and their meanings together with the relationships between these terms. It describes the meaning of objects and carries information about what types are available in the domain, their properties, and how they are related to one another. The terms stand for classes of objects in the field while the relationships can be ranging from subclass relationships to limitations on values that can be taken. Using these classes and their properties enables the user to create relations among classes and terms by identifying the property value through the Ontology. This will provide guidance on the annotation process using default information and retrieving information in terms of supporting user query [9].

The structure of Ontology is presented as a hierarchy that is established by linking classes with relationships. Every class in the hierarchy of Ontology will be related to at least one other class with one of these relationships. This structure provides general root or top classes (*e.g.* Movie) and some more specific classes that appear further down the hierarchy (*e.g.* Comedy, Thriller). These are arranged hierarchically with rich semantic relationships between terms and attributes which organize information within the database.

Figure 4 shows a simplified model of how the Ontology philosophy used in this research, which include concepts (depicted as square boxes), two exemplary instances (depicted as round boxes) and relations between them (depicted as arrows; continuous arrows are used for hierarchical relations, a dashed line represents a self-defined semantic relation). Classes represent general concepts or knowledge categories and form the basic structure of the Ontology. Instances represent individual concepts and are grouped into super ordinate classes; and properties are used to represent relations between classes as well as to specify their typical attributes. The classes Person and Movie in our example also possess data type properties (like has_runtime ,has_date_of_birth) which do not relate to other classes but specify class attributes. They can be directly filled with values which are not formal parts of the Ontology.

Ontology is the most significant part in our automatic annotation functionality [8]. This is due to the components which makes the extraction and formalization of the semantics feasible.

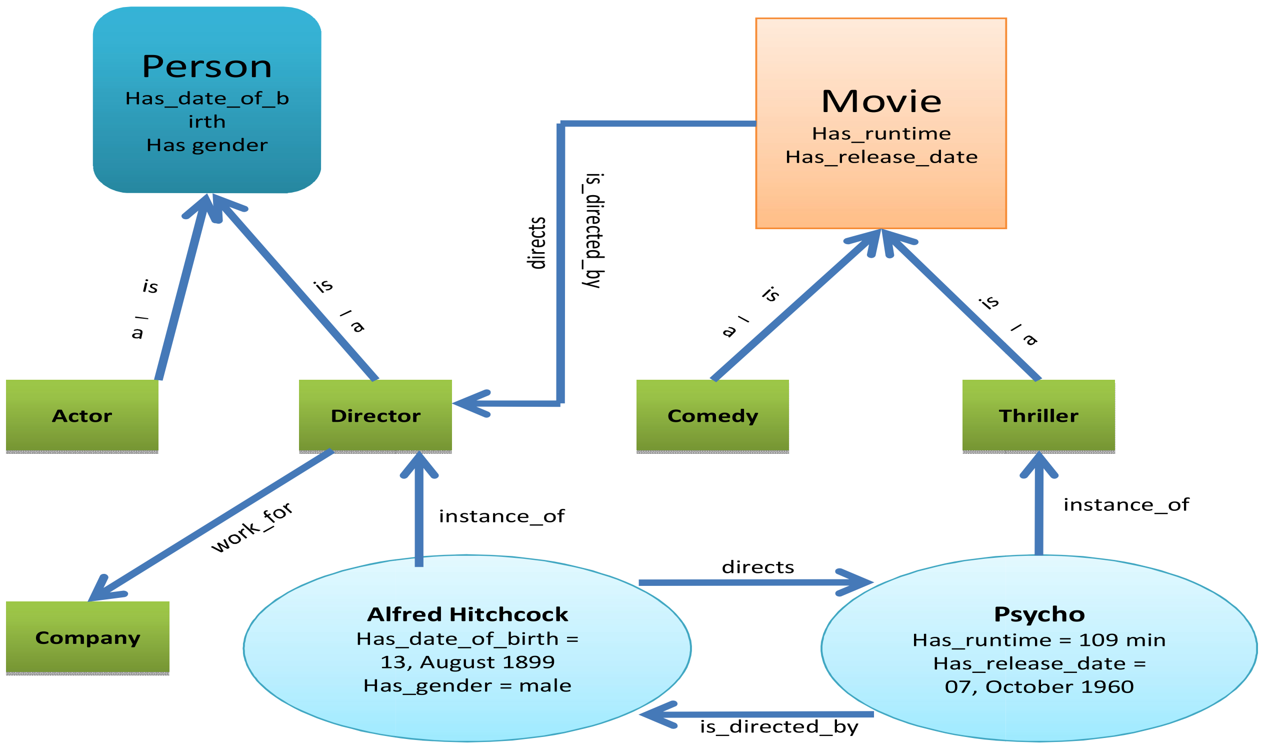

Figure 4 A simplified Model of Basic Ontology Elements

In ontologies, classified relationships are frequently labelled as part_of (for meronymy), and is_a (for hyponymy), but also may called differently, e.g. subset_of, subclass_of, kind_of , etc. This variety of naming relations in ontologies is one of the difficulties in using different ontologies together and in reusing existing ontologies. As we have already mentioned, ontology editors and ontology languages allow one to establish a variety of specific semantic relations via object properties. Yet there are some knowledge models that are built in formal languages but (almost) only use hierarchical structures.

This has already led to some discussion about whether such models should be regarded as 'full' ontologies [19]. One may certainly say that although ontologies provide the methods for exploiting specified semantic relations, there still is a dominance of hierarchical relationships.

Furthermore, hyponymy is the only relation to be typically treated as a first class relation in ontology languages. This means, for example, that in OWL we can find predefined is-a relations, while all other relations have to be established by the user through concept properties.

Consequently, most ontology editors (and the respective ontology visualization tools) separate hyponymy from other relations. Tools for editing ontologies are usually based on a tree-structure establishing the representation of a domain model [10]. Visualizations also often focus on the hierarchical structures within ontology.

Semantic annotation is the generation of specific metadata using terms drawn from an Ontology document in a formal language RDF [15]. RDF is usedfor representing a metadata that allows a machine to easily process that metadata. The basic data model in RDF contains three types of object which are Resources, Properties and Statements.It describes the meaning of terms and their relationships in modeling primitives of RDF called schema (RDFs). The relationships between resources can be described by a mechanism provided by RDF. These include classes, subclasses, sub-properties, range and domain limitations of properties [4]. With these extra primitives, RDFS can be assumed as a primitive Ontology language that can be used to define the semantics of particular domains. Ontology is needed on top of the RDFs to provide a way to express semantics in a machine processable form. In terms of developing the algorithm, it is essential to use different types of Ontology, these are:

- Top- Level Ontology (Upper Ontology): A universal Ontology all over domains (*i.e.* Illustrate the universal concepts)
- Domain Ontology; AnOntology which is significant in a particular domain
- Lexical Ontology: Define special relations between conceptual categories *(i.e.* relationships among synsets).

Lexical Ontology [5] will be used in this work to provide new methods for accessing the information contained in web documents which extend the existing web. The lexical Ontology that will be used in the work is WordNet [3] as this database qualifies as an upper Ontology, containing the most wide-ranging of concepts in addition to more specific concepts which are related to each other.
The above philosophy is the basis of our algorithm which is to match the same topic in the sentence with entity / sub entity in the Ontology document.

2.2.2 WordNet

WordNet (WN) is a large public electronic dictionary, thesaurus and electronic lexical database [3], that has the potential of serving Natural Language Processing systems effectively due to its size and sophistication. The compilers' original idea was to "identify the most important lexical nodes by character strings and to explore the patterns of semantic relations among them".
More than 155,327 words forms are listed in the database in a structure that facilitates lookup on the basis of semantic similarities. In WordNet, a particular word is very often co-listed in a number of synonym groups. In addition to synonymy, which is the main

organizing lexical relation in the database, a host of semantic relations are incorporated. For example, the nominal senses of book: it is co-listed with the synonym volume, and it is possible to locate hypernyms (*e.g.* product), hyponyms (*e.g.* album, journal), holonyms ("book is a part of …"), meronyms (*e.g.* binding, cover) and coordinate terms (*e.g.* inspiration and deliverable) for the synonym set containing this sense. The system of semantic relations stored for verbs are equally elaborate. Adjectives and adverbs are also available.
WN defines a rich set of relationships between words. The main functions roles of WordNet in this system are:

- Combining thesaurus and dictionaries through synsets (set of synonyms).
- Providing general definitions and hierarchical relationships between words among concepts. This essential function is necessary for retrieving information in terms of supporting user query [9].

In this work WordNet will be used as lexical Ontology that contains word and world knowledge that is usable for representing web knowledge; typically called knowledge-oriented information retrieval. A hybrid bottom-up top-down methodology can assist to transform the WordNet to recognized requirements in term of automatic extraction using a set of conceptual relations [17].
In the analyzing phase, the proposed algorithm uses the concept of converting a series of characters into a serious of tokens (*i.e.* split the text into tokens and then the sentences into words using tokenizer) [7]. This process called tokenization. Tokenization will provide additional context information. In the WordNet, there are four divisions which are considered as syntactic "part of speech" categories, namely as:

1. Verbs;
2. Nouns;
3. Adjectives;
4. Adverbs

Where in the verbs, nouns, adverbs and adjectives are structured in logical grouping called synset (or synonym sets), each set has a distinct cognitive concept (*i.e.* each representing one underlying lexical concept). The WordNet contain many English words around 155,327, namely verbs, adverbs, adjectives and nouns. These words are polysemous having different senses or meanings. The word McIntosh, for example has three word-senses: McIntosh#(sweet), McIntosh#2(Computer) and McIntosh#3(family name). WordNet has the ability to distinguish 207,016 word-senses.
The main components of WordNet applied in this thesis as follows:

- **Synset**

The synset, which is the construction block of the WordNet text, consists of all possible vocabulary that expresses specific concepts. Thus synsetis utilized as lexical concept in our algorithm. In WordNet the words are organized in 117597 synsets and around 207016 word-sense pairs [11].

Synset considers a list of synonymous collocations or words (*e.g.* "Red Marker", "place in") while synonymous sense are grouped into one synset for example – the word "board" and "plank" may both refer to "a stout length of sawn timber, with varying sizes and with varying applications". In this sense, the "board" and "plank" are synonymous; the word "board" may be synonymous to another word in a difference sense, and it will be grouped in yet another synset.
The synset also consist of a list of words called a "collocation" – where these words or collocations are grouped so that they can be interchanged, and they can appear in more than one part of speech and in more than one synset. The philosophy and role of the synset are grouped all the senses of the same word together and each form in a different synset, to ensure that no two synsets will have the same meanings.
The synset in the WordNet is organized in a lexicographer files. These files describe the relations between groups of synonyms and relations between one synset and another synset using pointers. Table 1 listed the details about number of synsets, strings, and total word-sense available in Wordnet [11].

Table 1 Number of Synsets, Strings and Word-Senses

Part of Speech	Synsets	Strings	Word senses
Noun (n)	61426	117079	145104
Adjective (a)	18877	22141	31302
Verb (v)	13650	11488	24890
Adverb (r)	3644	4601	5720
Total	117597	155327	207016

- **Pointers**

Pointers [12] represent the relations between the words in one synset and another synset. There are two types of relations organized in this system, these are:

1. Lexical relations – This type represent the relations among words that are related semantically only and may exist include hyponymy, antonymy, meronymy and holonymy.
2. Semantic relations - This type represents the relations between the meanings of the words.

There are various properties related to each part of speech for the potions of Wordnet. This will lead to have special actions *i.e.* the relations holonym/meronym and hypernymy/hyponymy are essential to the group of nouns related to WordNet. The adjectives are grouped mainly related to the similarity and antonymy relations. Table 2 shows the relations counting by the linker type.

Table 2Relations Counting by the Linker Type

Relations	Verb	Adverb	Adjective	Noun
hypernym	13124	-	-	75134
hyponym	13124	-	-	75134
instance hypernym	-	-	-	8515
instance hyponym	-	-	-	8515
part holonym	-	-	-	8874
part meronym	-	-	-	8874
member holonym	-	-	-	12262
member meronym	-	-	-	12262
substance holonym	-	-	-	793
substance meronym	-	-	-	793
attribute	-	-	643	643
domain region	2	2	76	1247
domain member region	-	-	-	1327
domain usage	16	73	227	942
domain member usage	-	-	-	1258
domain category	1237	37	1113	4147
domain member category	-	-	-	6534
Entail	409	-	-	-
cause	219	-	-	-
also	589	-	2683	-
verb group	1748	-	-	-
similar	-	-	22622	-
antonym	1089	718	4080	2142
derivation	23095	1	12911	35901
participle	-	-	124	-
pertainym	-	3213	4852	-
Total	54652	4044	49331	265297

The semantic relations are divided into two categories to the element kinds of the relation. They are semantic linker and lexical linker. The semantic linkers link synsets and lexical linkers used to link specific word senses. This will lead to have three groups for the 26 semantic pointer as shown in Table 3.

Table 3 Classification of Pointers in WordNet

Relation Type	Classification Standard	Semantic Pointers
Lexical linker	Between senses	Antonym, participle, pertainym, derivation
Semantic linker	Between synsets	Hypernym, hyponym, instance hypernym, instance hyponym, part holonym, partmeronym, member holonym, member meronym, substance holonym, substance meronym, substance meronym, entail, cause, similar, attribute, verb group.
both	Between senses or synsets	Also, domain category, domain member category, domain region, domain member region, domain usage, domain member usage

- **Token**

A token is a block of text [12] which has been categorized according to function or meaning. Using a lexical analyzer function will divide the sequence of characters into "tokens" according to the function to have meanings, the process called tokenization. Each individual instance of the sequences of these characters is called lexemes.

A token can be anything from number to arithmetic operator to a character but needs to be part of a structured set- for example to tokenize the following expression: Sub=13 - 8;
lexeme

Sub	Identified as an Identifier in the token type
=	Identified as an assignment operator in the token type
13	Identified as a number in the token type
-	Identified as a subtraction operator in the token type
8	Identified as a number in the token type
;	refer to End of statement

Once the tokenization is completed; parsing will follow where data which has been interpreted will be loaded into the data structures to be used for interpretation, and compiling.

- **Scanner**

Scanner [12] is applying as a function that converts a sequence of character into tokens. The scanner can contain the information on the possible orders of characters that are handled within tokens. For example, a token of type integer may contain numeric digits. In manycases, the longest match rule is applied in which the first character is used to determine what kind of token will follow as well as each succeeding characters are then processed one following another until a new character which may not be acceptable for the token type is encountered. In other cases, the creation of these lexemes are more complicated – even involving backtracking the previous characters.

- **Lexer Generator**

A lexer generator [12] is a way used to perform lexical analysis to the text under consideration – or to convert sequence of characters to tokens. A lexer also known as the tokenizer or scanner that's splits the input characters and converts them into tokens.

2.2.3 Jena Toolkit

To be able to create applications that can read and process semantic metadata represented in RDF, a conformant implementation of the RDF specification needs to be developed. In the past different implementations have been developed that are based on dissimilar interpretations of the RDF specification [6]. This has caused much confusion for developers as they want to create applications that work with RDF correctly but are unsure of the correct interpretation of RDF . Jena was developed to provide the much needed conformant implementation of the RDF specification. Jena is a Java Application Programming Interface (API) for the creation and manipulation of RDF graphs developed by Hewlett Packard Laboratories [6]. It provides programmers with a simple, easy to use collection of Java classes and ways for creating and using OWL, RDF, RDFS, SPARQL. This allows applications to read RDF, write RDF and even query RDF.

Figure5 shows the architecture of the Jena implementation [17]. It has been designed to permit the easy integration of alternative processing modules such as parsers (1), serializes (2), stores and query processors (3). The API has been well-defined in terms of relation interfaces so that application code can work with different implementations without change [McBride B., 2005]. Java interfaces are included for representing all RDF key concepts like resources, models, statements, literals, properties, etc. A common set of classes (4) has been created to implement these interfaces. A model class (5) provides a generic implementation of an RDF graph (collection of statements). A standard interface connects model to classes that implement storage and basic querying of RDF statements (6).

The RDF graph is stored in a great deal simpler abstraction known as graph. Graph model is a quite rich programming interface. This permits Jena to use a wide range of different storage strategies as long as they conform to the graph interface. Jena has the ability of using a tradition disk-based topple index to supply a graph as per in an SQL database, or as a persistent store or as an in-memory store. Other stores can be connected to Jena through a suitable extension point provided by the graph interface. The process of calling graph API to work on that particular store is permitted by writing an adapter.

The semantic regulations of OWL, RDF and RDFS are the key characteristic of SW applications. The information can be inferred by using the semantic regulations which is not clearly specified in the graph model. For example, Z is a sub-class of X if Z is a sub-class of class Y and Y is a sub-class of X. Jena's inference API allowed entailed triples to be added explicitly in order to appear in the store.

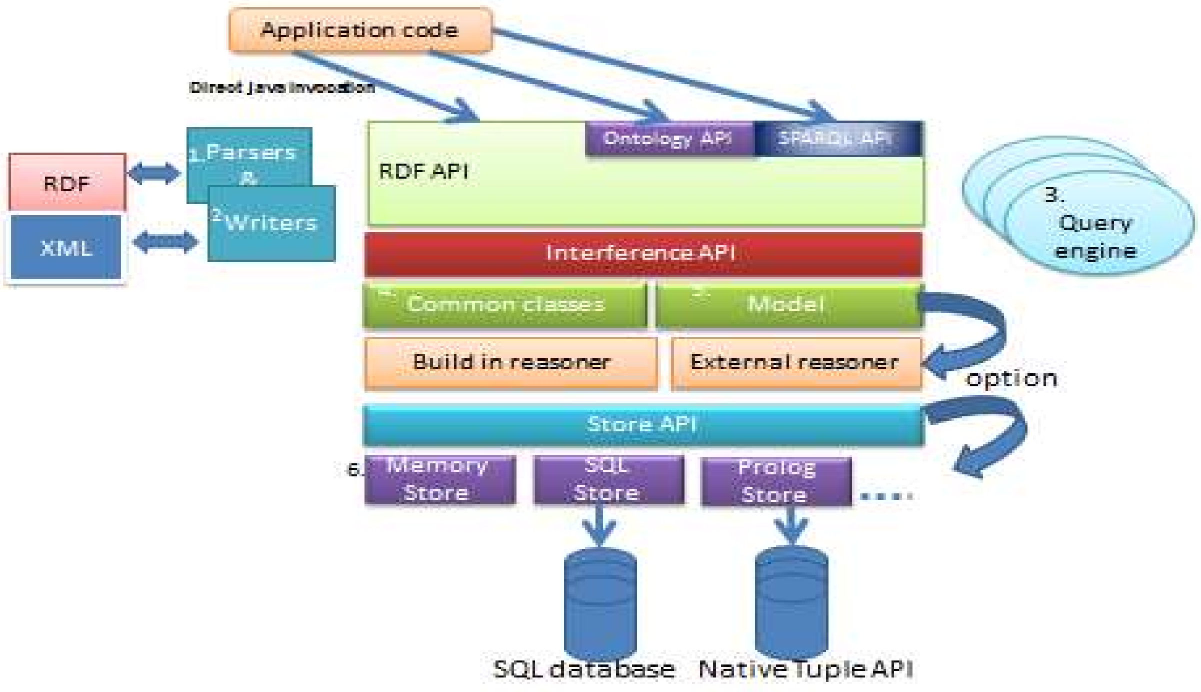

Figure 5 Architecture of the Jena implementation [McBride, B., 2001]

Jena framework will be used as a platform in this research to provide the necessary system requirements. It provides a reading and writing Parser and environment for RDF, RDFS, OWL and SPARQL. Additionally, Jena has dominant and elastic SPARQL method which is used for querying an RDF to express arbitrary queries and to extract data [19]. It is concluded that a number of rule engines provided by Jena assist to accomplish this work**.** This accomplishment can be either using application custom rules or using the built-in rule sets for RDFS and OWL.
Alternatively, the same work with various, specific, reasoning algorithms can be done by connecting the inference API to description logic (DL) engine [Simon Jupp, *et. al.*, 2009]. Furthermore, Jena can be used in an inference engine using the Model Factory methods to make inference about a model based on Ontology.

2.3 Functional Requirements

The Functional requirements are methods or features that must be included in the system [1] to satisfy the original aims of the system. The key functional requirements are listed below:

- **Open an Ontology** – The annotator must be able to open an existing Ontology from a URI or a local document.
- **Open a web page** – The annotator must be able to open a web page by specifying a www.
- **Extract Knowledge from web page** – This will involve scanning a web page to extract relevant information.
- **Automatically generate possible annotations**- The information extracted from the web page will be used to automatically create suggestions for annotations based on the current Ontology. These possible annotations will then be presented to the annotator as a list which is easily understood.
- **Post generated annotations** – The annotations generated by the model will be sent to the annotation server.
- **Reject annotations** – If required, the annotator may reject annotations that they consider unnecessary – these will be removed from the full list of automatic annotations.
- **Save annotations** – The annotations created by the annotator will be saved to an XML/ RDF format on his local system.

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, Rejected or Posted at the same time by repeating this process for each annotation selected. The Use Case diagram in Figure 6 outlines the main tasks that the annotator is able to perform [1].

2.4 Non-Functional Requirements

- **Efficiency –** The system will have to analyze web pages of various sizes using an extracting information technique and so this analysis will be efficient to ensure the annotator does not have to wait too long for a response.

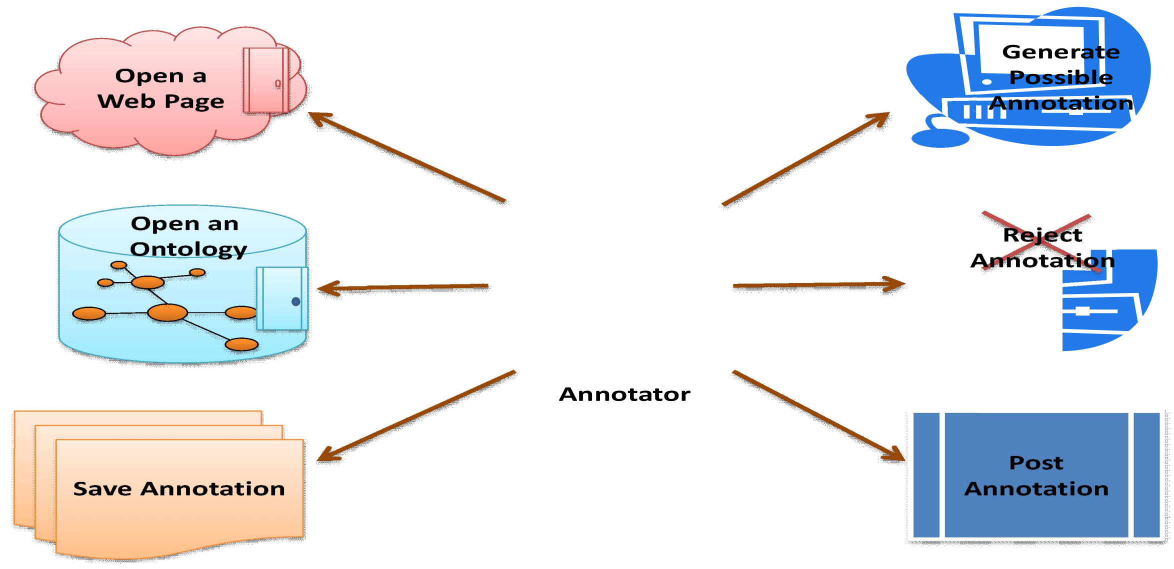

Figure 6 Use Diagram for Role of Annotator

- **Adaptability** – The system will be easily adaptable to work with different Ontologies. As the SW grows a vast amount of semantic metadata and will need to be represented via annotations. This will result in many different types of Ontology. For the system to be useful it must be able to be used to create annotations for these different Ontologies. The system must also be adaptable to work in different domains of knowledge. To help provide this it should be possible to adjust the types of information component extracts.

- **Usability –** Like the Web, the author believes the Semantic Web will be used by an enormous amount of people, each having different abilities. For the Semantic Web to be fruitful, these users need easily be able to annotate web pages. This will lead to a greater amount of semantic metadata that can be then be exploited by search engines, intelligent agents and various other applications in ways described earlier.

- **Reliability** – Like any good service, the system should be reliable. The Semantic Web will depend on users to create their own semantic annotations and so they must have an annotation tool that they can rely on time and time again to aid them in this task. This feature will also indirectly increase the overall usability of the system.

- **Platform Independence** – The system should be compatible with Microsoft Windows platforms. This will further enable a large range of people to use the system.

3. 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.

4. REFERENCES

[1] Ala'a Qasim 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.

[4] Alexander Maedcher and Steffen Staab, (2001); "Ontology Learning for the Semantic Web". IEEE Intelligent Systems Journal, Vol. 16 Issue 2, March 2001.

[3] Amaro, R., R. P. Chaves, P. Marrafa, and S. Mendes, (2006); "Enriching WordNets with new Relations and with Event and Argument Structures",7th International Conference on Intelligent Text Processing and Computational Linguistics , pp. 28 - 40, Mexico City, 2006.

[4] Andreas Harth, Jürgen Umbrich, Stefan Decker, (2006); "MultiCrawler: A Pipelined Architecture for Crawling and Indexing Semantic Web Data",Proceedings of 5thInternational Conference on Semantic Web (ISWC 2006), Athens, GA, USA, (5-9)th November, 2006, pp. 258-271.

[5] Atanas K. Kiryakov and Kiril Iv. Simov and Marin Dimitrov, (2001); "OntoMap: Ontologies for Lexical Semantics", In: Proc. of the RANLP 2001 Conference, Tzigov Chark, Bulgaria, 5-7 September 2001. PP. 142-148.

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

[7] Chwilla, D.J., & Kolk, H.H.J. (2002) ; “Three-Step Priming in Lexical Decision” . *Memory & Cognition*, Vol.30, no.2, pp. 217-225.

[8] Dietmar Jannach, Kostyantyn Shchekotykhin, Gerhard Friedrich, (2009); ”Automated ontology instantiation from tabular web sources-The All Right system, Web Semantics: Science, Services and Agents on the World Wide Web, Vol.7, no.3, p.136-153, September, 2009.

[9] Ding Zhou , Jiang Bian , Shuyi Zheng , Hongyuan Zha , C. Lee Giles, (2008); “Exploring social annotations for information retrieval, Proceeding of the 17th international conference on World Wide Web, April 21-25, 2008, Beijing, China.

[10] Hai H. Wang, Nick Gibbins, Terry R. Payne, Domenico Redavid, (2012); “A Formal Model of the Semantic Web Service Ontology (WSMO)”, Journal of Information Systems, Vol. 37, no. 1, pp. 33-60.

[11] Huang Xiao-xi and Zhou Chang-le, (2007); “An OWL-based WordNet lexical ontology”, Journal of Zhejiang University SCIENCE A, vol.8 ,no.6, pp. 864-870, 2007.

[12] Maedche, Alexander and Staab, Steffen, (2010), “Ontology Learning for the Semantic Web”. http://eprints.kfupm.edu.sa

[13] McBride, B., (2001). “Jena: Implementing 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

[14] Nicola Guarino and Chris Welty, (2000); “A Formal Ontology of Properties”, Proceedings of EKAW-2000, 12th International Conference on Knowledge Engineering & Knowledge Management, Spring-Verlag LNCS, Vol. 1937, PP. 97-112, October, 2000.

[15] Pan, J. Z. and Horrocks, I., (2007); “RDFS(FA): Connecting RDF(S) and OWL DL”, IEEE Transactions on Knowledge and Data Engineering, Vol.19, no.2, Februry 2007, pp.192 – 206.

[16] Sheila Kinsella, John G. Breslin, Alexandre Passant, Stefan Decker, (2008); “Applications of Semantic Web Methodologies and Techniques to Social Networks and Social Websites”, Reasoning Web, PP. 171-199, 2008.

[17] Staab S, & Studer R, “*Handbook on Ontologies”,* Berlin: Springer-Verlag, 2004; Gruber, T., ”Collective Knowledge Systems: Where the Social Web meets the Semantic Web”, Journal of Web Semantics, Vol.6, no.1, pp.4– 13.

[18] 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.

[19] Weller, K., (2007); “Folksonomies and Ontologies: Two New Players in Indexing and Knowledge Representation”, In H. Jezzard (Ed.), Applying Web 2.0: Innovation, Impact and Implementation. Proceedings of the Online Information Conference, London, Great

Britain (pp. 108– 115). London: Learned Information Europe.