EXTRACTION AND INDEXING OF TRIPLET-BASED KNOWLEDGE USING NATURAL LANGUAGE PROCESSING

by

David C. Hooge Jr.

(Under the Direction of Budak Arpinar)

ABSTRACT

A proper understanding of any document relies heavily upon two things: an understanding of the relationships between terms and a grasp of the manner in which language relates one term to another. For example a full comprehension of the sentence “Jane plays basketball” requires the reader to first understand that Jane is related to basketball by her taking part in this activity; second, the reader must have an understanding that of how basketball relates to other terms. Thus, for a full grasp of the sentence the reader must be aware that basketball is a sport among other things. These two understandings are missing from current search and storage methodologies and are instead largely replaced with word distance measures. As such the only relation stored by most modern methods is that the word “Jane” appears near the word “basketball.” Our system remedies these two problems through both relationship recognition as well as a grasp of how concepts relate to one another as in the linking of “sports” to “basketball.” This allows for automated semantic information storage and beyond this enables storage of information in a manner that resembles the structure of language.

INDEX WORDS: Natural Language Semantic Store, Semantic Web, Ontologies, RDF, Indexing, Natural Language Processing

EXTRACTION AND INDEXING OF TRIPLET-BASED KNOWLEDGE USING NATURAL LANGUAGE PROCESSING

by

DAVID CARL HOOGE JR.

B.S., Birmingham-Southern College, 2003

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2007

EXTRACTION AND INDEXING OF TRIPLET-BASED KNOWLEDGE USING NATURAL LANGUAGE PROCESSING

by

DAVID CARL HOOGE JR.

Major Professor:	Budak Arpinar
Committee:	Prashant Doshi John Miller

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
May 2007

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

The most often cited challenge of the Semantic Web movement is the transformation of Natural Language text into Semantic Information [Bern00a]. That is, the alteration of human understandable information into machine readable data. Even HTML content lacks methods for allowing a program to process a document's meaning. There are some attempts to remedy this problem such as allowing the placement of meta-information into HTML and Web sites that allow their users to annotate other sites with terms regarding the subject of a given site or page [Abra98, Hamm05] or snippets of a page (i.e., microformats, RDFa (www.rdfa.info)). But for the most part, the act of placing a document online is largely akin to placing a new book onto a bookshelf with only a vague title to guide a reader to the information.

The natural and intuitive ability of humans to understand language, in both written and spoken form, comes from the knowledge of the terms being discussed and the relations among these terms. For example, English speakers know that a more general term for "table" is "furniture" and that "stuff" and "thing" can be used as a catchall for any concept. Computer programs, on the other hand, have to rely on analysis such as frequency of terms, and structure of links between the terms (i.e., href in HTML). Such methods have proven success as it is evident by top search engines. However, we claim that a better computer processing of text is possible by exploiting relationships among the terms. As the current Web evolves into the Semantic Web, it is expected that relationships will play an increasingly significant role both in the field of research [Shet03] and within the commercial sector [Shet05]. In fact, there are commercial products that make use of natural language parsing within a specific domain, such as MedScan[1].

[1] http://ariadnegenomics.com/products/medscan/

The challenge is, to develop methods that can convert sentences into a form that allows better computer processing (i.e., improvements on retrieval of correct documents).

Search engines have a limited understanding of relationships present in text and are generally limited to the notion that a term is present in a document and appears within a certain distance of other terms. That is, the actual relationships between data are ignored. Without knowledge of the relationships present in data any queries for related entities will be error prone. Our approach attempts to remedy this by injecting a human's understanding of language into document processing for indexing and retrieval. This is accomplished first, through the recognition of both terms and any relationships among them; and secondly by relating the terms and relationships to an ontology of terms. The ontology used herein is structured in such a way that more general or more specific forms of a term are linked. For example, "Sports" and "Golf" are directly linked because "Golf" represents a specific sport. This allows our system a grasp of both the relationships present in the text as well as relationships among terms that may not be directly present in the text. Thus, our system seeks to address two major problems: one, the problem of entity placement and two, the problem of relationship recognition. Here, by the entity placement problem we refer to the lack of understanding of the specificity, generality, or relationship that a term has to other entities. This concept was illustrated in the previous example where it was noted that "Golf" and "Sports" are related by one term being more specific than the other. The relationship recognition problem represents a lack of understanding of the links between entities as stated in the text. For example, the sentence "Joe is a Lawyer," draws a clear relationship between Joe and the profession of lawyer. Typically, this type of relationships is completely ignored in favor of a more simplistic inspection of word distance. Our approach to these problems relies on breaking sentences into three pieces consisting of a subject, a relation,

and an object. For instance, suppose a document contains the text: “viruses are the cause of many diseases.” Most search engines would note that the terms “viruses” and “diseases” appear in the same document somewhat close to each other and therefore some relationships exist between them. However, there is a causal relationship present between the terms that would be ignored. Our approach recognizes the causal relationship present in this sentence and represents this using a subject – predicate – object form; or more specifically as “Viruses” <cause of> “diseases.”

In this paper, we use ontologies and natural language processing to move one step ahead on the transformation of natural language text into a form or representation that facilitates an improvement on the processing of information. The contributions of this paper are two-fold. First, we introduce a novel method of creating subject – predicate – object triplets from natural language. Second, we demonstrate the applicability of this triplet representation for querying and retrieval of documents through an application that we developed.

CHAPTER 2

MOTIVATION

People have a natural and intuitive understanding of the hierarchy inherent to the language they speak. As mentioned before, “table” and “furniture” are related by the first being a more specific form of the second. It is this comprehension of general to specific along with a grasp of the specific relations among terms that allows speakers to quickly move from one topic to another in conversation with little or no explanation of the transition.

Unfortunately, this human understanding of language is rarely recognized by search applications. Search engines simply recognize what terms are contained within a document and their proximity to each other. There is no understanding of the actual relations among terms or where the term stands in the hierarchy of language. Golf is recognized only as a location in memory and not as a sport.

Beyond the hierarchy of general to specific, search engines have an extremely limited understanding of any relationship present in the document. This comprehension is generally limited to a knowledge that a term is present in a document and appears within a certain distance of other terms. This causes the system to be ignorant of the actual relationships between entities in the data. Without knowledge of the relationships present in data any queries for related entities will be error prone. As an example, suppose a user wishes to search for “famous authors not awarded a Nobel prize” or any other negative relationship between two entities. This would return documents that contain “authors” and “Nobel prize” in close proximity which would, in all likelihood, be a list of authors that where awarded a Nobel prize; the exact opposite of what the user was seeking.

This underscores the strength of the Semantic Web model and the weakness of the current Internet. As it currently stands the World Wide Web contains no well-defined or coherent mechanism for storing or processing relationships between pieces of information. Most current Internet search applications are based solely on the relationship of reference proximity and have no understanding of the nature of a relationship or the specificity of a given term.

The Natural Language Semantic Store attempts to remedy this by injecting a human's understanding of language into search and retrieval. Both TAP [Guha02] and WordNet [Fell98] are used to lend the system an understanding of the hierarchy of language while natural language parsing in the form of JavaNLP [Klei02] is used to mine relationships among entities. Discovery of entity relationships is used not only during document storage but also during user search in order to ensure a constant awareness of links between entities.

Thus, the system detailed herein addresses two major problems present in current search methodologies. One, an ignorance of the relationships specified by the author and two, a lack of understanding of how different terms relate to each other through the structure of language. These two problems are detailed in greater depth in the two sections to follow.

2.1 Entity Placement Problem

When an entity is hashed to a location in memory this provides no understanding of the specificity, generality, or relationship the term has to other entities. An understanding of these concepts can not only improve search results but also lend the engine an understanding that emulates a human's own grasp of language. The system and algorithm presented here attempts to address this lack of understanding on the part of other search methods by using developed ontologies in addition to natural language processing techniques. This lends the system a comprehension of language that resembles a human's own understanding.

2.2 Relationship Recognition Problem

Indexing based on term location in a document causes any relationships between entities presented in the text to go unprocessed. The result of this is that all searches simply scan for terms identical to those present in a query and appearing in close proximity to each other. The system detailed herein attempts to repair this problem through natural language processing that converts sentences and queries into their component entities and recognizes any relationships between them.

CHAPTER 3

BACKGROUND

One of the major goals of Natural Language Processing is to lend a computer the same level of language understanding possessed by a fluent speaker. While this goal currently remains unaccomplished there are a multitude of progressions along numerous paths that all attempting to create a viable solution. However, despite the existence of many different paths most people would readily agree that any application that finally achieves this goal would require three general components [Mahe95], [Cull86].

1. *Dictionary:* More specifically the system requires an understanding of the meaning of words and their association with other words. This is best thought of, as the title implies, as lending the program a dictionary it can page through and look up word meanings. This does not, however, denote a true understanding of the word; that portion of the system in detailed in the following component.
2. *Inference Engine:* Where the previous dictionary component lends the system the ability to know the various meanings of a word; an Inference Engine allows the system to choose from amongst them. For example: the word chair has two very distinct and different meanings, namely it can refer to the article of furniture one sits upon or the head of some governing body as in "science department chair." The dictionary component reports these two meanings to the program whereas the inference engine gives the ability to choose the proper definition for the situation.
3. *Language Understanding:* Arguably an understanding of the language naturally stems from the combination of the above two components, however, for the sake of completeness it is listed here as a separate piece. This represents the ability of the system

to fully realize the meaning an author intended in writing a given passage. This ranges from word understanding to a grasp of the general meaning of a sentence.

This is not to say that the ultimate ends of Natural Language Processing is language fluency. This idea is better viewed as a general idealized goal for the field. Many of the systems described in the sections that follow have goals that are quite divergent from that of full language understanding. However, they do all share a common thread in the desire to give a computer application some understanding of human communication. The understanding a system seeks ranges from computer object representation of the sentence [Klei02] to text processing for the purpose of classifying document as being part of a given field [Guth99] to even aiding document retrieval in larger systems [Jone99].

Perhaps then a better statement of the aims of NLP is that it wishes to create a machine understanding of natural language. This understanding is not always analogous with fluency. NLP seeks to give a computer some ability to process human created text in much the same manner a computer utilizes a data object or accesses a database. It wishes to gift the computer with the ability to pull information from text in the same manner data is pulled from a database upon execution of a query.

This desire is extraordinarily similar to that of the Semantic Web movement which, in part, wishes to transform the web from a shelf of books into a database. At their heart both seek to transform something originally meant for human consumption into something that a machine has a ready understanding of. This represents the greatest link between the two subsets of Computer Science in addition to underscoring why the combinations of their methods is both ready and needed.

In this section we will first provide an overview of the background of the Natural Language Processing subset of Artificial Intelligence, then move to overview a few modern systems with a special focus on any ways in which Natural Language uses several of the methodologies also associated with the Semantic Web movement.

3.1 A History of NLP

Natural Language Processing found its beginnings in the mid-1960s with two major systems containing similar underlying functionality: ELIZA [Weiz66] and STUDENT [Bobr66]. These two systems are highly representative of what can be considered the first generation of Natural Language Processing systems. Their understanding of language, as well as the domains in which they were able to function, were extremely limited. This heavy domain dependence stemmed from a programmatic reliance upon grammatical structure and word meaning rules that existed solely within a single domain. This means that the computers lacked any ability to understand the meaning behind language and rather emulated this understanding through the rules that always held true for a given domain. For example, in the domain of law the word "ruled" denotes a decision handed down by a judge, thus the outcome of a given court case can not be far behind. In this way these systems used general heuristics as well as exploiting clichéd phrases within the domain to create the illusion of text understanding. In addition to these limitations the systems were strictly able to process simple declarative or interrogative sentences and nothing more complex.

Running parallel to these systems was what, at the time, seemed like a promising branching of NLP: language translation. While this segment is, for the most part, outside of the scope of this overview it is important to note the effect early translation programs had upon later NLP applications. The first translation applications sought to convert between one language and

another through a simple dictionary understanding of words. This means that terms in one language were simply converted to their counterparts in another [Cull86]. However, this once again points to the problem of choosing which meaning of a word is in current use. This approach ultimately provided unacceptable results leading to the aforementioned conviction that any natural language system required something more than a dictionary understanding of the words being used.

Later generations of NLP applications learned from the previous processing and translations attempts and included an integration of semantics. Of note in this vein was the SHRDLU [Wino72] system which married semantic understanding with a reasoning engine borrowed from Artificial Intelligence. The promising results from this lent further credence to the belief that language understanding required a grasp of both word meaning and word use. This second generation also marked the advent of systems that sought to transform natural language into some machine understandable intermediary form [Scha75] rather than create a direct understanding of language. This process can be thought of as a conversion of the human created text into object representation. This allows for application processing that is much the same as the manner in which a program is able to interact with its own contained objects. This lends a degree of representational understanding of the sentence to the program.

These initial generations of NLP research are outlined here to further emphasis the ready ability to combine Natural Language techniques with those of the Semantic Web. These two subsets of Computer Science seek to, in some sense, lend programs the ability to understand text meant strictly for human consumption.

3.2 NLP and the Semantic Web

As NLP systems have matured over the years they have also realized an increased need for language understanding. As time has passed this need for language understanding has drawn closer and closer to the Semantic Web vision for a machine understandable World Wide Web. It is our belief that the two subsets have of late drawn close enough that a linking of the two is nearly unavoidable.

Perhaps the best example of this is the previously mentioned ontological based NLP engine. This engine, referred to as the Mikrokosmos Project [Mahe95], utilizes a situated ontology in order to allow the program an in-depth understanding of the language of a given domain. In fact more descriptive time is spent, within the framework of the paper, describing the creation of the ontology rather than the formulation of the Natural Language engine. This underlies the project's heavy reliance upon its ontology, a concept that represents one of the major methodologies at the heart of the Semantic Web movement. In particular the Mikrokosmos project outlines several general ways in which an ontology bolsters the process of Natural Language parsing or any other system:

1. *Symbolic Meaning:* A fully formed ontology guarantees that every concept represented within a text will not only be mapped to meaning but to relationship with other concepts. This ensures some baseline understanding of any term used within the text.
2. *Meaning Collapse:* Because of an ontology's ability to store relationships amongst objects in addition to the object's meaning much of the definition of a word can be collapsed and represented through relationships. For example, the concept of a "syndrome" is highly linked to medical science. This is a fact normally explained in the

definition of this concept. An ontology would allow for the explicit omission of this information in favor of simply showing a link between the two.

3. *Constraint Understanding:* Some concepts only make sense within certain bounds. For example: liquid water can only exist at a range between thirty-two and two hundred and twelve degrees Fahrenheit. An ontological representation allows for easy understanding of complex concepts such as this and gives natural language processor a shortcut to understanding.
4. *Faster Learning:* An existing ontology provides a groundwork for the understanding of any new concept. This concept is quite familiar to most people, however few have seen this idea stated in such a way. The idea here is that once one has a baseline understanding of a topic it becomes easier to gain additional understanding regarding this subject. For example, children are taught addition prior to multiplication because once addition has been learned the understanding of multiplication simply requires one's mind to compute several addition functions.

The key difference between this work and our own is the system's reliance upon carefully formatted and correct domain information. Our system has none of these requirements and as such has none of the domain dependences and needs for carefully constructed knowledge bases to represent them.

Another system that attempts automated information extraction from natural language text is the Artequakt project [Alan03]. However, like the previous system Artequakt also requires a previously formed ontology in order to fully extract information. Here the ontology takes the form of a classification structure rather than the more general ontology of terms as was seen in the Mikrokosmos project. This classification ontology is utilized as the system searches

online for documents and information that matches its in-built classification structure. Thus, the system contains no initial corpus of information but rather issues searches to web search engines in order to locate information related to the ontology that it is given. Thus, the system no only locates its own information but is also able to update the information it extracts as new facts are placed on the web.

Like the Mikrokosmos project mentioned before the Artequakt system differs from our own in its reliance upon a correctly constructed ontology. Thus, the user must in large part tell the system what information he or she wishes to locate and then run the system in order to allow it to find this information. Due to our reliance upon natural language processing the program detailed herein is able to extract information without a previously created ontology and beyond this is able to create information from documents related to any domain.

Another example of a Natural Language system that includes a heavy reliance upon ontological information comes from the use of WordNet [Fell98] to create much of the word understanding present in the previous system. It should, however, be noted that WordNet represents something closer to a general language understanding mechanism rather than the ontology present in the previous system. Specifically, Amit Bagga et al. illustrates one use of WordNet with their message understanding system [Bagg97]. This system seeks to allow its user to build an information extraction system based not on text formatting or regular expressions, as is frequently used, but on an understanding of the language given in the text. In much the same way as was mentioned previously with a situated ontology WordNet provides the ability for the program to gain a semantic understanding of the words and phrases being used in addition to a comprehension of a word's relation to other terms. This work diverges from our own by its manner of information extraction. More specifically the message understanding

system extracts information from a Web page into a preexisting template. Thus, the program scans the text looking for specific pieces of information such as phone number or street address rather than seeking to capture all information present in the text.

A similar use of a language ontology is seen in the Semtag and Seeker project [Dill03]. The information created by the Semtag portion of the project is then used by seeker to index the information and allowing for speedy retrieval. Here the TAP ontology is used to allow entity recognition in the three step process of semantic tagging:

1. *Spotting pass*: Documents are retrieved from the store and all references to entities in TAP are located. The system then stores the ten words to either side of the entity reference creating a window of context around the object.
2. *Learning pass*: Once all windows from the document store have been saved they are all scanned in order to determine the corpus wide distribution of terms.
3. *Tagging pass*: The distribution of terms is used in this pass to disambiguate all the terms. This is done by inspection of terms contained within the windows surrounding each entity as well as within the document as a whole. Thus, if a document contains a reference to Tiger Woods in addition to multiple references to video games then the system could reasonably state that the reference to Tiger Woods is not to the person but rather to the video game series baring the person's name. With disambiguation complete the system then tags the entity with the proper TAP reference.

The Semtag and Seeker project differs from our own in its ultimate goal; more specifically Semtag and Seeker wishes solely to disambiguate entities with regards to their TAP reference whereas our project wishes to extract triplets from text. Thus, Semtag and Seeker is

best considered as part of any future work upon our project that would allow for disambiguation of entities.

Entity disambiguation is once again seen in the SKR project to locate terms present in the UMLS metathesaurus [Srin02]. Here natural language parsing is used to locate noun phrases within medical abstracts. These phrases are then matched against the concepts contained within the UMLS knowledge base using flexible matching techniques. This means that the matched concept need not be stated exactly the same as the concept present in the thesaurus but need only match within a certain word range.

The SKR project differs from our word in its focus on simply using natural language processing to location entities rather than extract facts regarding the information present in text. In fact, the a large portion of its functionality is encompassed by a portion of our own project with the only difference being that we have left disambiguation as future work rather than incorporating it as part of our current work.

The practice, mentioned previously, of translating natural language into an intermediately form of programmatic object is used create document summaries [Jure04]. This system, however, includes a reversal of the previous examples; here Natural Language Parsing techniques are used to create an intermediary ontology. Whereas the previously mentioned systems represent NLP utilizing aspects of the Semantic Web movement; thus, the current system is better thought of as the Semantic Web utilizing NLP. This system first parses a given text using a natural language processor that converts the sentence into an annotated object. This object is then mined to create an intermediary graph representation of the information contained in the sentence. Once the full text has been mined the graph object is inspected for patterns that indicate which sentences can be extracted from the text and included in a summary. The creation

of the intermediary graph form is simply a middle ground for the ultimate goal of summarizing the text. Thus, this graph does not represent a complete and correct representation of the information in the text, rather it is simply a means to the ultimate ends of text summarization. While this system's focus is less upon correct formulation of the information in the text into ontological form and more upon creating a representation that can be used for summarization this still represents the use of a technique typically associated with the Semantic Web.

A similar use of an intermediary form of information representation is found in the works of the Attempto project [Kuhn06]. However, unlike the previous summarization project the authors of the text must use the Attempto Controlled English (ACE) language. The project summarizes the major facts of texts pulled from the biomedical domain to allow for more effecting information mining. In this way a large corpus of biomedical information, which is notoriously difficult to cross-reference with other information sets, can be mined and then compared to other ACE language rendered information. The authors note that their language is capable of fully representing 56% of the headlines the system was tested over with support for partial representation of another 23%.

The difference of this project from our own work lies once again in the need for the author to create a representation of the information other than the standard language representation. Our own work mines the existing language and makes no requirement that either the author originally represent the information in a intermediary structure or that the document be later reformulated by another person.

While the previous examples have largely confined themselves to inspecting natural language for information the START [Katz02] system proposes using a syntax that greatly resembles natural language. This allows creates a semi-structured basis from which to create

semantic information. Thus, a person with only a passing knowledge of the Semantic Web would be able to create information for use by various semantic systems. An interesting facet of this concept is its underlying presumption that the semantic method of representing information is not far from the natural language means of information presentation. While this concept deviates greatly from our own work so that little comparison between the two systems can be made it should be noted that this concept greatly underlies our own work. Both share a belief in the relatedness of natural language and the Semantic Web method of information storage.

There also exist systems that attempt triplet extraction from text by all-together different means. Specifically an attempt is made at inspecting the structure and language used in HTML code in order to discover relationships among entities [Svat03]. This technique, however, limits itself to a heavy reliance upon inspecting the structure of HTML code. Any natural language processing takes a backseat to the structured HTML. Thus, while this system does create a map of links among entities it requires extensive additional information to do so in the form of HTML scrubbers and parsers.

Each of the previous systems provide and example of the successful marriage of NLP and Semantic web techniques. They underscore both the ability of the two subsets to be combined in addition to the relatedness of the two areas. While the focus and methodologies of the systems outlined are divergent they do all share a common goal of heightened language understanding. As noted before this understand can take radically different forms, from a speaker's fluency in a language to the ability to represent a sentence as an object that can then be passed and parsed between functions.

It should also be noted that a similar method of document classification to our system is used in the Semantic Enhancement Engine [Hamm02]. This system seeks to annotate natural

language text with information that specifies the proper domain for the terms contained in the document. Ultimately, however, this system diverges from our own in that it seeks document and entity classification whereas our own system is more largely concerned with relationship recognition as stated by the author of a document.

3.3 Marrying NLP and Semantic Web

Natural Language Processing and the Semantic Web movement engender many of the same concepts. Most notably they both seek to lend some form of language understanding to machines. While the goals are by no means completely similar they carry enough similarity that either side would find its objectives greatly aided by methodologies from the other. The modern systems mentioned here show this exceptionally well in the form of a working merge of components dear to both subsets of Computer Science.

Beyond this our system differs from all the system herein on three points in addition to addressing the two previously mentioned problems with the current breed of web searching applications:

1. *No ontology needed:* Because of our use of natural language parsing and the subsequent processing of this information our system does not require an ontology in order to extract data from unstructured text. Rather it relies on an understanding of the structure of language itself.
2. *Domain independence:* Our reliance on an understanding of the structure of language further means that our system can operate over the text drawn from any domain.
3. *Requires only natural language:* We solely require the text of an article for triplet extraction. There is no requirement of metadata regarding the text. Thus, far less

human intervention is required for the conversion of human created text into semantic information.

A heightened level of integration between the Semantic Web and Natural Language processing is both possible and needed. The two fields have goals and methods that are too highly related for them to be considered together.

CHAPTER 4

NATURAL LANGUAGE IMPLEMENTATION

The Natural Language Semantic Store system operates in two distinct phases, as do all other search applications: document store and document retrieval. In addition, the system itself also consists of two distinct parts: the natural language processor, detailed in this section, and the Term Hierarchy Tree, detailed in the section to follow.

4.1 Natural Language Processor

The natural language processing portion of the system detailed here seeks to address the second of the two problems given above, that is our approach for creating triplets from text focuses on the problem of relationship recognition. That is, special attention is paid to relations formed between entities and every effort is made to ensure that the all relationships are captured as precisely as possible.

Natural language processing in our system relies on a parse-tree produced by an existing NLP parse engine. We chose Stanford's JavaNLP parsing engine because it represents an established code base as well as for its log-linear run time [Klei02]. JavaNLP parses all entered text into a tree structure that begins at a root node, denoted as root and containing no information, and progresses downwards to leaf nodes based on phrasal dependence.

While our approach uses the JavaNLP engine to generate a tagged form of the information present in the sentence this does not mean it is solely dependent upon JavaNLP for all functionality. While the methods detailed below often give examples as generated by JavaNLP it should be noted that we anticipate that our approach would work equally well for any natural language processing engine that returns both part-of-speech tags and phrase dependences.

JavaNLP parses all entered text into a Tree structure that begins at a root node, denoted as root containing no information, and progresses downwards to leaf nodes based on phrase dependence.

```
"Tiger Woods donates to a large number of charities."
(ROOT [69.474]
  (S [69.371]
    (NP [20.560] (NNP [8.264] Tiger) (NNP [9.812] Woods))
    (VP [47.672] (VBZ [11.074] donates)
      (PP [31.541] (TO [0.003] to)
        (NP [27.963]
          (NP [15.561] (DT [1.413] a) (JJ [5.475] large) (NN [5.979] number))
          (PP [11.856] (IN [0.669] of)
            (NP [10.784] (NNS [7.814] charities))))))
    (. [0.002] .)))
```

Figure 1 - The Parse Tree Generated by JavaNLP

JavaNLP uses the Penn Treebank set of language tags to annotate a given sentence [Marc93]. This tagging set is broken down into three levels. These represent grammatical usage from most inclusive to least inclusive:

1. *Clause Level*: This allows for full tagging of a coherent clause, the most common of which is a single declarative clause represented as S.
2. *Phrase Level*: This enables phrase recognition within a clause. The most common of these, as well as the most useful to this project, are noun and verb phrases represented as NP and VP respectively.
3. *Word Level*: This level simply provides the part of speech of each word in the sentence. For example: proper noun, noun, plural noun, etc.

Clauses exist only singularly and cannot be contained by any other structure excepting the root tag which is able to contain all other tags; however, this represents a design decision of

the tree data structure rather than an actual tagging of the language. Phrases are generally contained within clauses, however they can appear outside of any clause as in the case of a sentence fragment, and are able to contain other phrases. In fact, it is a phrase's ability for self-containment that forms the basis of a sizable portion of triplet creation. Word level tags associate with a single word and reveal its part of speech.

Triplet creation relies, primarily, on phrase level tags. These tags provide us with indications on precisely how a given natural language sentence can be reformed into the subject - predicate - object triplet associated with the Semantic Web movement and, more specifically, with RDF information.

By itself the tagging done by the JavaNLP engine tells little about the information contained within a given sentence. It is solely concerned with revealing part of speech and phrasal dependencies present in the text. Thus, our program works by only using JavaNLP to generate a parse tree such as that of Figure 1. Besides the generation of parse trees, no other processing is required from JavaNLP. The generation of triplets, indexing and storage are accomplished strictly by our own methods detailed below.

The entire process of initial triplet creation, or more exactly the first five steps given below, requires a single pass through the tree. This generally makes the creation of the tree object representation of a sentence the most computationally intensive portion of the natural language processing segment of the program. This is, however, not always the case since tree creation is heavily affected by the complexity of the sentence.

4.2 Natural Language Algorithm Overview

Once the sentence tree has been created, our program parses it for both entity recognition and triplet formation. Thus, the parse tree given in Figure 1 represents the full contribution of any Natural Language Parser used by our program.

The challenge of converting from the parse tree structure to triplet form lies in the fact that the relationships between entities in text can be extremely complex. This fact is best illustrated by the 27% average error rate among the untrained human subjects (described later in the evaluation section (Section 6)). Our method addresses the problem of entity recognition and relationship formation by first locating entities within a sentence. The sentence is then further inspected to locate all relationships between entities. Conversion from a natural language sentence to triplet format occurs in seven phases, listed here and detailed below:

1. *Entity Recognition:* Noun phrases are extracted from the sentence tree and are scanned for entities based on part-of-speech recognition.
2. *Predicate - Object Recognition:* Verb phrases are extracted from the sentence tree and are used to form initial versions of the predicate, object pairing.
3. *Predicate - Object Augmentation:* Prepositional phrases are used to further alter the predicate - object pairing created in the previous step. This often involves combining the previous predicate - object linking with the prepositional phrase to create a `new predicate. A new object is then formed from what remains of the verb phrase.
4. *Triplet Creation:* The fully augmented predicate - object is recombined with the entity recognized in step one to form a full subject - predicate - object triplet.

5. *Pronoun Resolution:* All pronouns in the triplet are resolved to the closest proper noun phrase. This is the only step that can draw information not only from the current sentence but from any previous sentences as well.
6. *Triplet Filtration:* Triplets that have a high likelihood of being poorly formed or incorrect are discarded. Generally, this is accomplished by detecting redundant information across several triplets.
7. *Secondary Predicate Parsing:* Some of the predicates returned by the previous steps can be condensed to less wordy forms that still maintain much of the same meaning. Here a tree structure is created from the predicate and unneeded statements are eliminated.

In our approach, we use RDF (Resource Description Framework) to represent extracted entities and relations. One of the primary advantages of RDF, the knowledge modeling framework most often used in the Semantic Web, is that it enables easy recognition of complex data sets as a graph structure. In general this takes the form of XML represented statements about entities, or more intuitively statements that give some link between two entities and then further describe the nature of the link. This allows for several facts to be represented as a chain of statements, for example: “John Smith <studied at> Homeland High <founded by> Peter Sinclair.” However, natural language, if divided inexpertly, will form triplets whose understanding is dependent upon information contained in some proceeding set of triplets [Wood75] For example, consider the two triplets regarding a company’s product: “Acme” <makes> “Anvils” <for> “export”. Whereas in the previous example both triplets represent a single fact in this second example only the first of the two triplets represents a complete statement.

In the chain of statements given in the proceeding paragraph each of the two triplets has meaning independent of the meaning in any proceeding or trailing triplets. That is, understanding the fact that Homeland High was founded by Peter Sinclair does not require the reader to understand that John Smith also studied there. In our approach, the generation of each triplet is intended so that it represents a single piece of information.

4.3 Entity Recognition

When a noun phrase is located within the phrasal sentence tree it is understood to have the possibility of containing a single entity. Here the term entity refers to any abstract concept or real world object; for example: “Natural Science” would be recognized as an entity due to the fact that this references an abstract concept; additionally “University of Georgia at Athens” represents a real world place and would also be noted as an entity.

Once the noun phrase has been extracted from the tree it is parsed and nouns in both their singular and plural forms are combined to form a single entity. Specifically this means words fully contained within a noun phrase and tagged with NN - noun, NNS - plural noun, NNP - proper noun, NNPS - plural proper noun, or JJ - adjective. Proper nouns are given a special focus as these will later be used during the pronoun resolution phrase of triplet creation. This filters out all excessively descriptive words and returns the complete entity name. In should be noted that despite this need to remove descriptive words adjectives are still included as part of the proper noun. This is because adjectives often provide modifications to the entity that describe what type or give qualifying information that is essential to a full description.

With the noun phrase extracted, it is parsed and nouns in both their singular and plural forms are combined to form a single entity. For example, Figure 1 includes the noun phrase: (NP [20.560] (NNP [8.264] Tiger) (NNP [9.812] Woods)) this phrase would become the

single entity “Tiger Woods.” Proper nouns are given a special focus as these will later be used during the pronoun resolution phrase of triplet creation. This filters out all excessively descriptive words and returns the complete entity name. However, if the noun phrase is found to contain a proper noun then all words not tagged as a proper noun are removed from inclusion as part of the entity. The noun phrase mentioned previously could also have been phrased as: (NP [20.560] (NN [7.623] golfer) (NNP [8.264] Tiger) (NNP [9.812] Woods)). While this phrasing mixes nouns and proper nouns it would still produce the same outcome, namely “Tiger Woods.” Intuitively this rule is described as: “proper nouns do not mix with others.” This is done to ensure correct recognition of proper nouns.

While noun phrases contain only a single entity it is possible for several noun phrases to be bound to multiple identical predicate – object pairings rather than the more previously reviewed single entity binding. This happens when a conjunction separates two entities, for example consider the sentence: “Jack and Jill ran up the hill.” Here, both Jack and Jill are associated with the exact same predicate - object, thus meaning there are two triplets created from the sentence, namely: “Jack <ran up> hill” and “Jill <ran up> hill.” For the purposes of this program this is referred to as Coordinating Conjunction Splitting and is given a greater focus in the section of the same name to follow.

As for the assignment of URIs to entities or any component of a triplet, there are two ways the identifier can be assigned:

1. <u>The entity is part of the preexisting ontology:</u> In this case the object is simply assigned a URI that is identical to the previously assigned one. For example, Figure 2 shows the hypernym relationship present in WordNet and two entities are present along with their URIs, namely “chair” and “folding chair.” Thus, if either of these terms were located

within a sentence then once converted to triplet form and stored they could be referred to in the same manner as the entity in WordNet.

2. The entity represents a previously unseen term: In the case that the term is not present in WordNet nor TAP then the underlying storage subsystem is used to assign a URI to the object. Generally, this assignment takes a form extremely similar to that given in Figure 2.

```
http://wordnet.princeton.edu/wn#102894344-chair-n
<http://wordnet.princeton.edu/wn#hypernymOf>
http://wordnet.princeton.edu/wn#103253680-folding_chair-n
```

Figure 2. A Hypernym Relationship in WordNet.

It should be noted that TAP is a shallow but broad ontology and its knowledge base containing basic lexical and taxonomic information about a wide range of popular objects [Guha02]. In addition, our system allows any ontology to be loaded and used with the same effect as those detailed above. Therefore instead of TAP and WordNet any other preexisting ontology can be used in our system.

The phase of entity recognition and all other needed phases are explained on an on step-by-step in Table 1. This shows how each of the processes detailed here and in the sections to follow is applied to a sentence to produce the final output of a triplet.

4.4 Predicate - Object Recognition

In a manner similar to the extraction of noun phrases in the previous step verb phrases are likewise pulled from the tree. However, unlike noun phrases, verb phrases do not contain a single entity. In fact, all verb phrases, unless part of a sentence fragment, contain an underlying verb phrase and noun phrase, present in that order. These two included phrases are utilized to form the initial versions of the triplet predicate – object association.

Because of the verb followed by noun ordering of information within a verb phrase the

initial formulation of a predicate - object is fairly straightforward. The beginning verb phrase is

Triplet Creation Step	**Portions of Parse Tree Inspected**	**Product of Parse**
Entity Recognition	(NP [20.560](NNP [8.264] Tiger) (NNP [9.812] Woods))	"Tiger Woods"
Predicate – Object Recognition	(VP [47.672](VBZ [11.074] donates) (PP [31.541] (TO [0.003] to) (NP [27.963] (NP [15.561] (DT [1.413] a) (JJ [5.475] large) (NN [5.979] number))	"Tiger Woods" <donates to> "a large number"
Predicate – Object Augmentation	(PP [11.856] (IN [0.669] of) (NP [10.784] (NNS [7.814] charities))))))	"Tiger Woods" <donates to a large number of> "charities"

Table 1. A Step-by-Step Iteration through the Tree in Figure 1.

recognized and parsed, words tagged with VB - verb, VBD - verb past tense, VBG - verb gerund, VBN - verb past participle, VBP - verb non-third person singular present, VBZ - verb third person singular present, TO - to, DT - determiner, or JJ – Adjective are added to the predicate. Note the presence words not tagged as verbs within the predicate, specifically: to, determiners, and adjectives.

Determiners are included because they often provide limiting modifications on the predicate that can alter its meaning greatly. For example "few" is labeled as a determiner and if dropped from the predicate changes meaning from denoting a small subset of the objects in question to all of a given object. There is, however, one alteration made to the determiner rule: the word "the" while marked as a determiner is never included in the predicate. This is simple because the word "the" contains no useful information and is safe to leave out of the predicate.

Adjectives are included for much the same reason as determiners, these words represent extra descriptive text that if removed from the tends to alter the meaning enough that one finds the predicate referring to an entirely different set of things than was the intention of the author of the sentence.

Finally, “to” is included because this provides a link between predicate and object that if dropped would cause the predicate to be extremely hard to understand. For example, take the fragment of a sentence “ran to police” if rendered as predicate - object without “to” we would have “ran” as predicate and “police” as object; this makes it unclear if the object helped to operate the police department of if he sought aid from police.

The noun phrase is extracted and parsed in an identical manner as was described in the entity recognition step. Beyond verb phrases and noun phrases the other phrase recognized and processed is the prepositional phrase. While this phrase is extremely meaningful later in the algorithm at the current stage it is simply parsed and included as part of the predicate.

4.5 Predicate - Object Augmentation

There are two major types of augmentations that the initial predicate - object can undergo: coordinating conjunction splitting and prepositional phrase combination.

1. Coordinating Conjunction Splitting: A coordinating conjunction split is best understood when first illustrated with an example. Say we are currently interested in parsing the sentence: “Dick kicked and threw the rock.” There are actually two full subject - predicate - object triplets present in this sentence, namely “Dick <kicked> rock” and “Dick <threw> rock.” An identical process occurs in the case of a comma delimited list of items. It should also be noted the conjunction splitting is able to produce two slightly different forms of the same predicate. Consider the verb phrase: “incapable of

recognizing and repeating." When properly processed this produces: "<incapable of recognizing>" and "<incapable of repeating>."

2. Prepositional Phrase Combinations: In its simplest form, a prepositional phrase will cause the initially created predicate - object to be merged into a single new predicate followed by the prepositional phrase. Consider as an example the parse tree presented in Figure 1, as processing begins the initial entity "Tiger Woods" and ends at charities to produce: "Tiger Woods <donates to a number of> charities." This example is shown on a step-by-step basis in Table 1.

4.6 Triplet Creation

Following the predicate - object creation and augmentation phase, the final two portions of a full triplet are recombined with the entity to which they relate. Determining which predicate - object combines with what subject is a trivial task due to the structure of the sentence tree. Predicate - objects are combined with the noun phrase that immediately precedes them in the sentence tree. This noun phrase will always represent the subject of the trailing verb phrase. This process is simply reversed in the case of sentences with leading verb phrases.

4.7 Pronoun Resolution

Pronoun resolution is the process of changing any pronouns present in the triplet into the entities to which they refer. It also marks the only portion of natural language processing which spans across multiple sentences. All other processing is limited solely to the current sentence in question. The guiding principle behind the form of pronoun resolution used herein is that a pronoun appears closest to the proper noun to which it refers, here closest refers strictly to word distance.

Most often a pronoun is used immediately following the proper noun to which it refers for both the sake of simplicity as well as ease of understanding. However, this is not always the cause and will sometimes lead to a pronoun being resolved to the incorrect entity. This problem and the subject of pronoun resolution is a complex and thus we leave both its mention and its application to the future work section.

4.8 Triplet Filtration

When completed, the above methods for triplet generation tend to overproduce. More exactly, they are disposed to return, along with triplets that represent important information stated by the author, triplets that are either simple repetitions of previously stated information or are simply too trivial to be present in triplet form.

The order by which triplets were created becomes important during the filtration stage. As stated before, sentences are parsed and triplets are created from a single pass through the sentence tree. Thus, each successive triplet represents information from a later portion of the sentence. Many of the filtration thus tend to favor keeping any triplet that appears later in the list of triplets produced. This is because the most common filtration is removal due to information repetition.

Most sentences contain anywhere from one to three triplets. This stems from the very nature of the sentence: a sentence is used to convey a coherent thought. This concept is used in triplet filtration, namely one of the requirements for filtration is the existence of a large number of triplets.

If a sentence does produce more than three triplets then all of the triplets are checked by three rules; applied in the order that follows:

1. *Short Subsumption:* Subsumption generally occurs when I trivial fact is stated and later expanded upon. If the sentence produces more than three triplets it has the possibility of triplet removal due to this rule. A triplet is discarded if it is extremely short and fully contained within a triplet that follows it. For example, the first of the two triplets: “Jon” <viewed> “film” and “Jon” <viewed the film> “North” would be removed due to this rule’s influence.
2. *Trivial Similarity:* If after the processing of the above more than five triplets still remain for a sentence some triplets can be removed in accordance with this rule. A triplet is removed from the output if its subject and object are identical and its predicate is nearly identical to a following triplet. Consider the triplets: “Peter” <ran quickly from> “Bloodhound” and “Peter <ran from> “Bloodhound,” here the first of the two would be eliminated due to its simple restatement of the second.
3. *Extreme Verbosity*: Finally, if the previous two rules leave more than five triplets unfiltered this final harsh rule is applied. If a subject appears still in more three triplets then all triplets beginning with this subject are filtered.

4.9 Secondary Predicate Parsing

The final of all natural language parsing steps resolves the problem of wordiness within the predicate. Towards this end the predicate is passed to the JavaNLP engine to form a sentence tree of just this portion of the sentence. This second parsing produces a tree different from a simple subset of the sentence tree produced originally.

Secondary Predicate Parsing takes place in three steps applied in-order; detailed below. It should be noted that if the predicate does not conform to any of the three cases below it is left unaltered.

1. Chain of Information Removal: If triplets are found to form a chain of information then these triplets are not removed. As noted before one of the major design decisions of our method was that each triplet should embody a whole and discrete piece of information. Any secondary processing over these triplets has the chance of reintroducing the triplet dependency problem outlined previously.
2. Leading Verb Phrase Parsing: If the predicate begins with a verb phrase then it is possible to reduce it to the simplest predicate formulation; namely, verb phrase immediately followed by noun phrase. For example, the predicate <ran for mayor during> contains this formulation of verb immediately followed by noun and can be reduced to <ran for mayor> with no loss of meaning.
3. Leading Noun Phrase Parsing: If the predicate begins with a noun phrase then a reduction to single noun phrase is possible. This generally happens when the predicate represents an existential phrase. Consider the predicate <is a longstanding> this can be reduced to the simple existential <is>.

```
(ROOT [19.809]
  (S [11.925]
    (VP [11.519] (VB [0.003]
have)
      (VP [7.801] (VBN [5.431]
built)))))
```

```
(ROOT [43.838]
  (SINV [40.731]
    (VP [12.981] (VBZ [11.074]
straddle))
    (NP [23.604]
      (NP [9.070] (NN [6.411] line))
      (PP [12.593] (IN [4.967]
between)))))
```

Figure 3 - Two Predicate Parse Trees

Figure 3 presents two different sentence trees formed by predicate parsing, both of these would fall under the leading verb phrase rule. The left example represents a tree including multiple verb phrases which form the new predicate. The right example gives a tree that contains a verb phrase followed by a noun phrase used to form the new predicate. The method does not require the ordering of one verb phrase followed by a one noun phrase, rather it is best described as: any number of verb phrases followed by a single noun phrase. Any trailing noun or verb phrases will then be discarded.

The presence of prepositional phrases within the predicate phrase trees should also be noted. These phrases are included as part of the new predicate. Also, should any prepositional phrases trail the final noun phrase they will also be included included, however, no other phrases are given this treatment. This is because, as alluded to previously, prepositional phrases often provide information that modifies and adds to information present in verb and noun phrases. Thus, any attempt to remove or truncate these phrases would be extremely error-prone and would result in triplets that do not accurately represent the data present in the original sentence.

CHAPTER 5

TERM HIERARCHY TREE IMPLEMENTATION

The Term Hierarchy Tree, shown partially in figure 4 can be thought of addressing the first of the two problems which are introduced in Section 1. Namely, it seeks to address the problem of entity placement, or rather: how a specific entity relates to others. Beyond addressing this issue it also serves as an indication of the validity of the information created from triplet processing. More exactly, it shows how the information pulled from natural language can be related to an existing ontology.

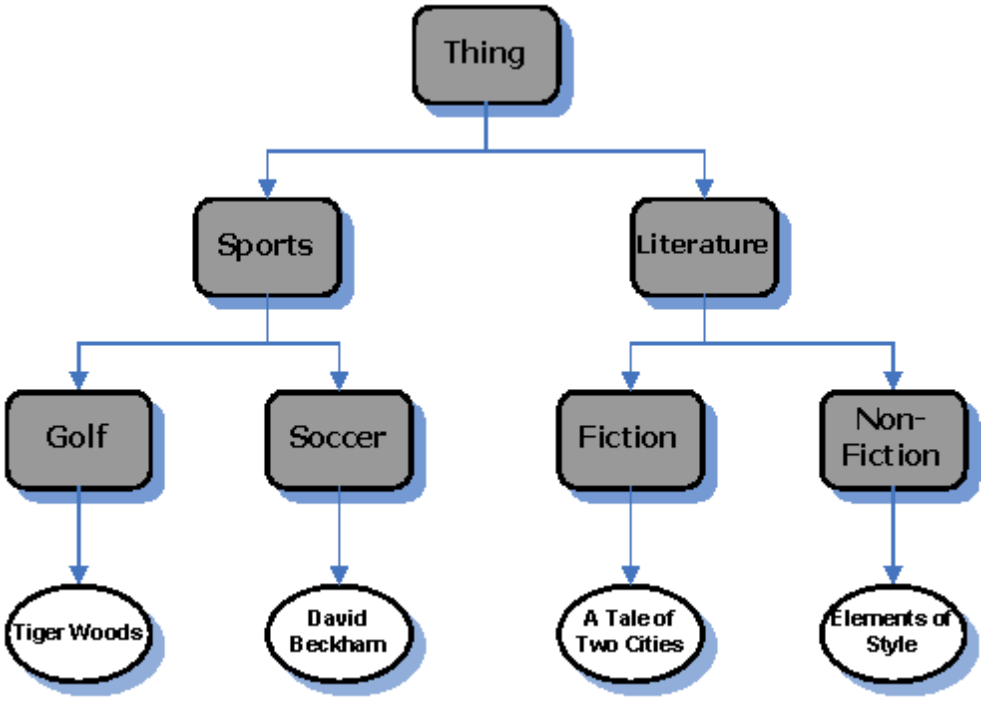

Figure 4. An Example of a Term Hierarchy Tree.

The tree in Figure 4 gives a brief overview of the Term Hierarchy Tree. Here terms donated by WordNet are shown as grey square figures while entities from TAP are displayed as

white circles. Note that as one progresses downward in the tree structure terms become more specific forms of the previous term. For example, “Sports” is related to both “Golf” and “Soccer” as these are more specific sports topics. TAP entities are placed under subjects to which they have a high relation, in this example the book *Elements of Style* is given as a prime example of a nonfiction publication.

Several existing components are integrated to give our system an ability to process the language that emulates some of the facets of the understanding of a fluent speaker. Sesame [Broe02] acts as a backend for information storage and retrieval. WordNet [Fell98] is used for information regarding English language terms, while TAP [Guha02] lends the system an understanding of common entities and helps determine their relation to the information stored in WordNet.

Sesame is an open source semantic database that contains support for both schema inference and querying. It was chosen because of its flexibility in terms of store and access methods in addition to its speed. Jena [McBr02] another leading open source semantic information store solution was also considered during the course of research for this project, however Sesame frequently bettered Jena in our tests.

WordNet is a lexical reference system that closely resembles a human’s own innate understanding of language. It contains nearly all dictionary words well as how these words relate to other similar terms. For the purposes of our project, WordNet is used to form a basic understanding of the generality or specificity of a term in addition to a grasp of synonym information. For example, if one was to inspect the term “chair” one would see that a more general term is “furniture” in addition to being presented with a list of more specific terms

ranging from “desk chair” to “chair and a half.” One would also be presented with synonyms for the word “chair” such as “seat.” This example can be seen as the entities are present in WordNet in figure 4.

TAP is quite similar to WordNet but with a differing focus. Whereas WordNet contains dictionary terms, TAP contains numerous real world entities ranging from people and places to even some of the more recent electronic devices. Combining these two knowledge bases allows the system to have an inbuilt understanding of an entity's “place in the grand scheme of things.”

```
http://wordnet.princeton.edu/wn#102894344-chair-n
<http://wordnet.princeton.edu/wn#hyponymOf>
http://wordnet.princeton.edu/wn#104004316-seat-n

http://wordnet.princeton.edu/wn#102894344-chair-n
<http://wordnet.princeton.edu/wn#hypernymOf>
http://wordnet.princeton.edu/wn#103945550-rocking_chair-n

http://wordnet.princeton.edu/wn#102894344-chair-n
<http://wordnet.princeton.edu/wn#hypernymOf>
http://wordnet.princeton.edu/wn#103497608-ladder-back-n

http://wordnet.princeton.edu/wn#102894344-chair-n
<http://wordnet.princeton.edu/wn#hypernymOf>
http://wordnet.princeton.edu/wn#103253680-folding_chair-n

http://wordnet.princeton.edu/wn#102894344-chair-n
```

Figure 5. Raw Triplet Data Produced by a Search for Chair in WordNet.

The Term Hierarchy Tree has two major components listed below and detailed in greater depth to follow:

1. Dictionary Store: This portion of the program includes all information from the TAP and WordNet knowledge bases in addition to any information added from outside sources. It

is also home to integration methods that allow information stored in differing formats to work together and refer to each other.

2. Indexing: This portion of the program controls indexing and reference for all data stored. This functionality is largely handled through Sesame.

5.1 Dictionary Store

The dictionary store is initially comprised of information from TAP and WordNet, and is later expanded upon as documents are stored. Whereas WordNet represents a dictionary of terms TAP and be thought of as representing an encyclopedia of concepts. It primarily stores the names of real world entities; for example: the names of all people who play for the New York Nicks is included in addition to all products currently produced by Apple. This knowledge base is linked to the information contained in WordNet by the category information given for each entity in WordNet. To illustrate this consider searching WordNet for information regarding “Tiger Woods.” This returns two distinct sets of entities, one regards the person “Tiger Woods” and is marked with the general category “Athlete.” The other set of entities are all video games made utilizing Tiger Wood’s name such as “Tiger Woods PGA Tour 2004” and are marked “Console Game Software.” This category information can then be used to place entities within the Hierarchy of Terms.

This Hierarchy of Terms is created from WordNet’s hypernymy and hyponymy relationships. This provides a relationship of terms from most general to most specific. This is utilized in our program in order to determine a relationship amongst otherwise disparate terms, for example a triplet containing “Golf” would be found, through the relationship of hypernymy to be related to “Sports” and thus the user performing a search could be offered information on a more general form of the subject s/he searched for.

While the preceding gives an example of an ordering of terms which, for the most part, have a direct relationship between each other, this does not accurately represent the majority of terms. What happens in the case of terms that bare little or no resemblance to each other, such as in the case of "sports" and "books?" It is from this fact that the name "Term Hierarchy Tree" is drawn. "Sports" and "books" represent two distinct branches of the tree, so that while these two concepts may have an identical more general ancestor, they represent two different unrelated parts of the tree.

In this way all terms stored within WordNet can be arranged in a tree structure with each level of the tree symbolizing a level of generality. This is, however, not to say that all terms present at the second level of the tree have the same level of specificity, or classify the same number of terms. The various branches of the tree will often have radically disparate levels of generality. However, within the local context each step down in the tree represents a step downward to the most specific terms of that domain. As an example let us revisit the previously mentioned terms "books" and "sports." Say that these two terms are present on the same level of the tree and that the "books" node has two children: "fiction" and "non-fiction" while "sports" also has two children: "golf" and "soccer." An understanding of the meaning of these terms immediately exposes to the reader the fact that golf and soccer refer to very specific sports while fiction and non-fiction represent the two most general classifications of books. Yet within the domain of sports and books each represents the next step downwards towards more specific terms. The uneven tree structure this ultimately forms stems from the fact that some areas of interest can be subdivided to a greater extent than others.

Up to this point the information provided by TAP has been largely ignored in the discussion of the Term Hierarchy Tree. TAP is integrated with the information from WordNet

by means of the relation information associated with each entity in TAP. In terms of the Term Hierarchy Tree this means that each entity contained within TAP can be thought of as a leaf node. Thus, TAP is utilized for the correct indexing of entities to the categories given by WordNet. To return to the example given above, suppose that "Tiger Woods" is found as the subject of a triplet. "Tiger Woods" would then be located as a component of TAP and beyond this linked to the information in WordNet as is stated previously. Thus, "Tiger Woods" would be related to "Athlete" which would be in turn related to "Sports" and so on.

This raises an interesting question regarding the placement of "Tiger Woods" under "athlete" rather than under "golf." This placement represents a design decision made by the makers of TAP. It should be noted that either placement makes sense, the entity "Tiger Woods" is highly concerned with both golf and is properly termed an athlete. This problem of dual placement is addressed by allowing a given entity to appear in more than on location. Thus, it is quite possible for "Tiger Woods" to appear in both categories. It should, however, be noted that the "out of the box" functionality of the system solely includes the placement of "Tiger Woods" as an athlete; his eventual placement under "golf" is, however, detailed below in the section that discusses the storage of information to the Term Hierarchy Tree.

5.1.1 Storage of Documents

When a document is entered into the system, it is first parsed by the natural language processor. This produces a set of triplets and entities that represent the information contained in the document. This data is then aggregated and representative entities and triplets are pulled from the document. The process of determining which triplets and entities can be thought of as representative is fairly simple but slightly different for each of the two document metrics:

1. Determining Entity Representativeness: The total number of entities produced by a document is counted and each is assigned a score based on the number of times it appears either as an entity or as part of a triplet. All entities referenced by at least twenty percent of the triplets created from the document are then taken to be representative and are noted as such along with their score. The score is noted for help in comparing documents that are determined to be largely about a given entity.
2. Determining Triplet Representativeness: The determination of triplet representativeness then relies on the entities found to be representative of the document. To revisit the example mentioned previously: when the program is executed using WordNet and TAP information "Tiger Woods" is initially related to "athlete." However, if a document is entered into the system that includes numerous sentences between "Tiger Woods" and "golf" then the indexes of the system will relate Tiger Wood's to golf.

Once the representative triplets and entities of a document are determined then this text stored with the appropriate relations. Thus, if two of the entities found to be highly related to a document were "golf" and "sports" then a reference to the document would be added under these two categories.

5.1.2 Retrieval of Documents

Retrieval of documents is initiated by a user supplied query. This query can range anywhere from an entity name to a full natural language question complete with punctuation. Queries are first parsed by the natural language processor in the same manner as a document is parsed for storage. This will generate either a set of triplets or, in the most simplistic of searches, a set of entities. These entities and triplets are then recalled through the use of the Sesame storage back end.

The Term Hierarchy Tree adds another benefit to information recall. It allows the user to not only view documents related to the search but also categories to which documents have a high relation.

A final note should be added regarding the ordering of documents. Documents are ordered by the degree of relationship they have with a given term. This means that if there are two documents associated with golf, one which has references to golf in twenty percent of its triplets and entities and the other using golf eighty percent of the time, then the second document will be displayed first. In the case of a search that includes a triplet or several entities the documents are ordered by a simple addition of percentages.

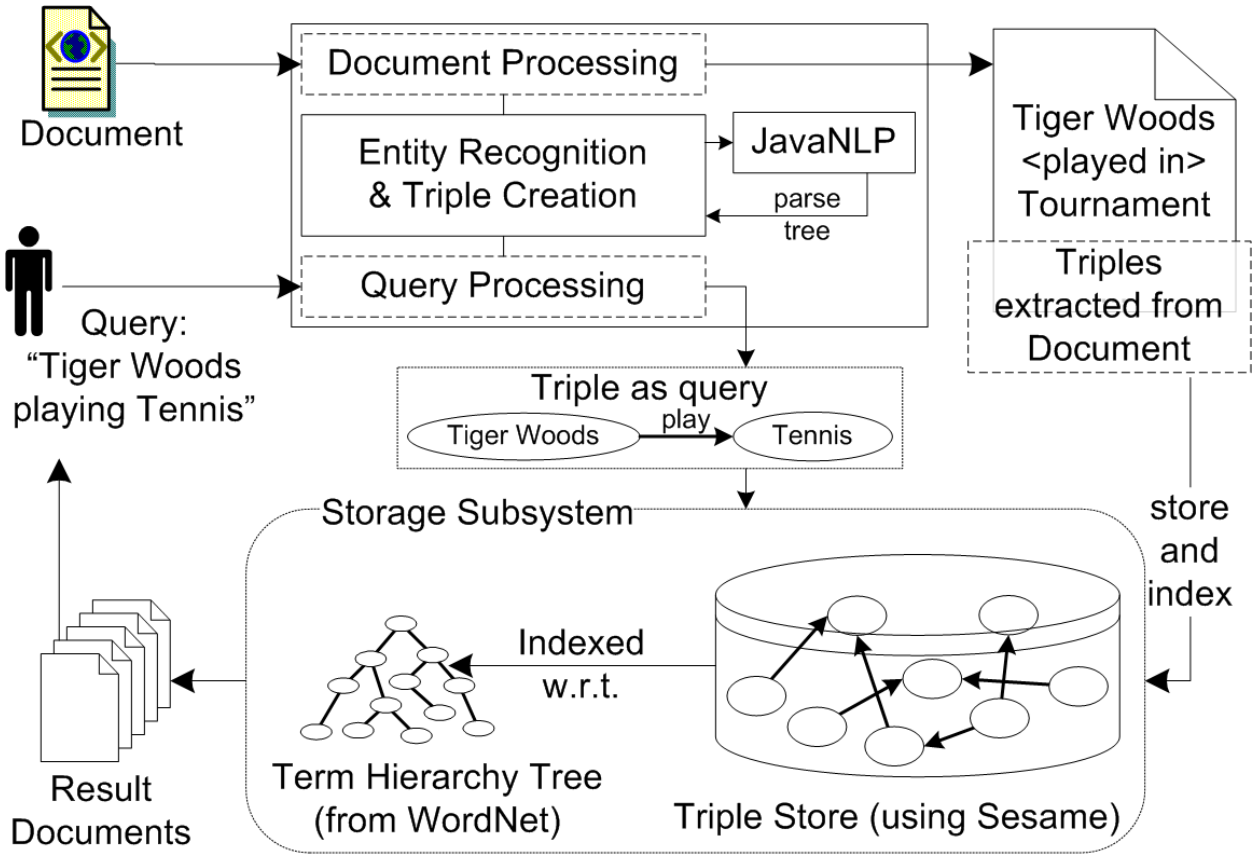

Figure 6. System Architecture.

5.2 Storage Platform

All indexing, storage, and retrieval is handled through the semantic data store engine Sesame [Broe02] as mentioned before. The capability most often used behind the scenes is Sesame's schema inference. This allows us to enter triplet information as it is extracted from the documents. Sesame then creates all needed schema information automatically without any further intervention beyond the initial configuration.

As for indexing, Sesame can be set up to allow indexing on a number of different dimensions, that is to say that it supports indexing in nearly exactly the same way that modern databases support indexing. For our purposes, indexing is done on both entities and the relationships between them. In this way while there are no specific searches which are optimized in general all searches reap some of the benefits of indexing.

Searching is accomplished through the use of Sesame's own query language, SeRQL [Broe04]. While it supports other Semantic Web query languages such as RQL [Karv02], and RDQL [Seab04], we have chosen SeRQL because of its native support within Sesame. This proved to yield slightly faster response times than were found to be the case with other query languages. In addition to query language support Sesame supports all major file formats for semantic information, such as N-Triples [Gran02], and N3 [Bern00], and allows for importing and exporting to and from these formats.

The system architecture is given in Figure 6 and shows the process of both storing a document and then the later retrieval of documents through a user entered query. Thus, the system can be thought of as having two phases that occur in order: first, document storage and second, document retrieval. Also note the system's heavy use of the triplet creation engine; this portion of our system is used for both relating documents to terms in the ontology and for queries.

CHAPTER 6

EXPERIMENTAL RESULTS

Testing focuses on the triplet creation engine of the program. This is due to the fact that the Term Hierarchy Tree was created and utilized largely as a method of lending a schema to the created triplets.

The dataset of *news articles* was chosen for several reasons: one, the heavily fact based nature of the articles they represent an excellent choice for our method of information extraction. Second, the articles are written for human consumption making the job of the human tester easier. In addition, this type of articles reflects a sizable portion of the documents placed on the Internet within a given day. Third, for ease of testing the documents need to be somewhat short, roughly a page in length, the inverted pyramid writing style of news articles allows text to be cut from the ends of articles without losing meaning. Beyond these reasons we chose human testing rather than testing with some preexisting corpus because of the novelty of our approach. While several testing corpuses address a problem that is related to those we wish to address none provide a dataset that provides for a clear translation from sentence to triplet.

The first part of the testing phase was accomplished by presenting a University of Georgia Computer Science Masters student who is not related to this research project with a set of twenty news articles. He then generated all the triplets that he believed were possible from these articles. Once done, he was allowed to review and discuss the triplets in order to determine if any incorrect triplets were present in this human created set. Thus, through initial production followed by review a sort of “gold standard” of triplets was generated for the news articles. In a similar manner the same twenty articles were presented to our system and the triplets it generated were reviewed and scored.

The second part of our testing was accomplished by presenting nine Computer Science graduate students who had just finished a course reviewing Semantic Web methodologies a worksheet that included instructions for triplet generation in addition to five news articles. The instructions were nearly identical to the provisions listed at the beginning of Section 3.1. That is, they told students that each triplet should represent a discrete piece of information as well as giving an accurate fact as stated in the article. The triplets the students generated were then gathered and reviewed to form a set of correct student generated triplets. This set was created by first inspecting all triplets created by each subject and eliminating all incorrect triplets. The set was then further reduced by comparing each subject's triplets against the triplets generated by all other subjects. By doing this, the final set of human created triplets includes only correct unique triplets as given by the nine human subjects.

This set is then compared against the triplets generated by our system to determine the overlap. A triplet overlaps with another if the two are determined to contain identical information. Generally this means that the two triplets were actually identical, containing all of the same words in the same order. Occasionally, however, there is a slight word variation between the triplets that does not result in a difference of meaning. For example, one student gave the relation "works for" while another gave "employed by," these two relations obviously represent identical information.

As mentioned previously testing took place in two stages. The first stage compared the triplets generated by the program against triplets generated by a human expert knowledgeable of the Semantic Web.

Figure 7 shows two important metrics for measuring the abilities of the system. The rightmost of the two comparisons shown for each of the systems gives the percentage of triplets

generated found to be correct on further review. The leftmost comparison gives the percentage of the "gold standard" of triplets that were captured by each system. Notice especially in Figure 4 that while our system's triplet accuracy is at 50% the algorithm manages to capture 81% of the "gold standard" triplets. This means that while the system overproduces it is still able to capture a majority of the correct triplets.

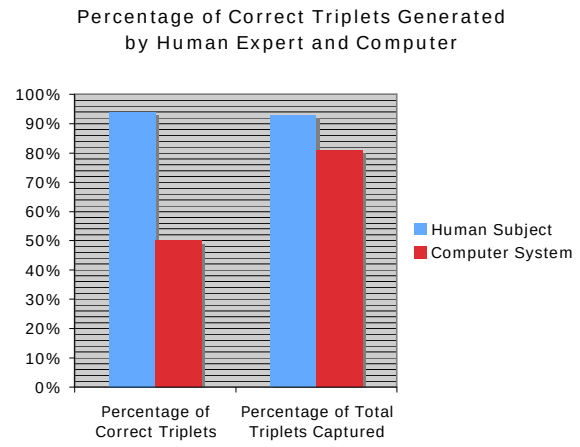

Figure 7. Percentage of Correct Triplets Generated.

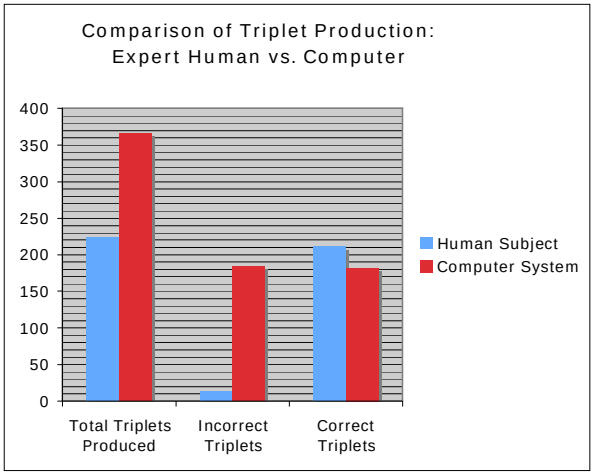

Figure 8. Triplet Generation Numbers.

The Figure 8 gives raw numbers for triplets produced by both the human and the computer system. This serves to once again emphasize that while our system does overproduce triplets the numbers of correctly produced triplets are extremely similar between both the two.

The second stage of testing involved a comparison between nine human subjects and the computer system to determine overlap as given below. In this phase of testing our system was found to overlap with 53% of the triplets created by the untrained subjects. This seeming reduction in the ability of our system to capture triplets stems from two sources:

1. *The computer system captured more triplets than the human subjects*: The human subjects captured a total of 104 correct and unique triplets, 55 of which were found to be identical to those generated by our system. However, our system generated another 23 triplets that were correct and not captured by any human subjects.
2. *Unlike the triplets generated by the expert the triplets generated by untrained humans contained inferences:* Inference triplets are triplets that represent a fact present in the article that requires an understanding of the text to create. For example, one article stated that that Intel sought to overhaul parts of its business in order to increase profit and then listed processors and memory among the portions to be overhauled. One student then created triplets that noted that both processors and memory are key parts of Intel. While this represents a correct triplet it should be noted that the formation of this triplet requires a human understanding of the text.

Both the expert testing and the inexpert testing phases of testing reveal one of the limitations of that system presented here, namely that while we are capable of boasting high recall the precision of the system is a good deal lower. We believe this low precision is acceptable given the system described herein also relies on the previously mentioned Term Hierarchy Tree structure. Thus, while document parsing will yield incorrect triplets the process will also create enough correct triplets that combined with the document metrics given above will yield the proper document recall.

There are, however, several methods that we leave to future work that are available towards the improvement of the precision of our system. These improvements are:

1.) Improvement of the JavaNLP used – While the algorithms described here work with any Natural Language Parsing system the ability to produce triplets can only be as good as the initial parse generated. The training data chosen for JavaNLP in this project represents the fastest backing file available for the project. Because of the high order of all Natural Language processing algorithms speed was a concern for our system. The training corpus used here has 80.1% precision [Klei02] whereas dataset that are slower and parse the sentence structure more deeply are able to produce 86.6% precision.

2.) Lowered triplet filtration limits – As mentioned previously our system was designed with some ability to accept inaccurate triplets due to the Term Hierarchy Tree design. Thus, the filtration limits are set purposefully low to prevent the incorrect removal of accurate triplets. By increasing the threshold for removal a larger number of incorrect triplets would be removed. This would, however, also result in the deletion of correct triplets, something this project wished to avoid as much as possible.

CHAPTER 7

CONCLUSION

The approach presented here addresses several of the problems present in modern search applications. Namely, it seeks to store and recall information based on entities and any relations present between the entities rather than scanning the documents for the existence of a searched for phrase. Towards this end Stanford's JavaNLP tool provides an infrastructure used to transform natural language sentences into the common triplet form used by nearly all Semantic Web applications. This natural language processing engine servers a dual role: first, it is used as a means to store documents and second, it is used in the same manner to process any queries issued to the system by a user. This allows the system a better processing of all relationships and entities present in the text. The end product allows for search and indexing based on this concept. This, we believe, is something largely lacking from modern information retrieval techniques.

CHAPTER 8

FUTURE WORK

There are several branches that we feel are beyond the scope of the current work that would, however, provide an excellent focus for additional work on this project. The primary of these is an improvement upon pronoun resolution. This is the subject of several papers [Tetr99], [Mitk98], and as such is well beyond this work's focus on creating a search and index method based on the relationship between entities. While the limited form of pronoun resolution used currently works well enough that it does not truly hamper the system it does represent a natural next step in development.

Beyond improvements of entity reference that would be accomplished through further work on pronoun resolution the ability to view information mined from a document as a graph would be extremely helpful to a quick understanding of the information. This topic is also the subject of a number of paper within the Semantic Web community [Deli06], [Flui02] and we feel that it would lend the user the ability to review all document relating to his search with a thumbnail overview of all the information. This could allow an at a glance understanding of information gleaned from a multitude of different sources and could possibly preclude the user having to read any part of the actual document in question.

This idea of visualization could also be further extended to allow for merging of all documents placed under a given classification. This would allow an "at a glance" understanding of all information mined related to a given subject. Given that the system detailed herein places a heavy focus on ensuring that each triplet mined from the document in question represents a fact that the author wishes to convey to the reader this would produce a graph of facts regarding a topic. Doing so would allow the user to understanding quickly and easily the various

relationships between differing segments of information regarding a category.

The document ranking methods used herein represent another area for improvement based on recent research in Semantic Web. Specifically retrieval ranking could be improved through the addition of metrics regarding the relations between entities [Alem03]. Thus, the system could gain a greater understanding of the degree to which a given entity is related to another or additionally the level of significance of the linking between the two. The determination of the "importance" of different relations within differing context could additionally be done through semantic association discovery as given in [Alem06].

REFERENCES

[Abra98] David Abrams, Ronald Baecker, and Mark H. Chignell. *Information archiving with bookmarks: Personal web space construction and organization*. In Conference on Human Factors in Computing Systems, 1998. pp 41–48.

[Alan03] H. Alani, S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis, and N. *Shadbot. Automatic ontology-based knowledge extraction from web documents. IEEE Intelligent Systems*, 2003; pp 14-21.

[Alem03] B. Aleman-Meza, C. Halaschek, I. B. Arpinar, and A. Sheth. *Context-Aware Semantic Association Ranking.* First International Workshop on Semantic Wb and Databases, Berlin, Germany, September 7-8, 2003; pp. 33-50

[Alem06] B. Aleman-Meza, A.P. Sheth, D. Palaniswami, M. Eavenson, and I.B. Arpinar. *Semantic Analytics in Intelligence: Applying Semantic Association Discovery to Determine Relevance of Heterogeneous Documents.* In Siau, K.L. ed. Advanced Topics in Database Research, Idea Group Publishing, 2006.

[Anya05] K. Anyanwu, A. Maduko, A. Sheth. *SemRank: Ranking Complex Relationship Search Results on the Semantic Web.* In Proceedings of the Fourteenth International World Wide Web Conference, 2005.

[Bagg97] A. Bagga, J.Y. Chai, and A.W. Bierman. *The role of WordNet in the creation of a trainable message understanding system*. In Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference. 1997.

[Bern00] T. Berners-Lee. *Primer: Getting into RDF & Semantic Web using N3*. http://www.w3.org/2000/10/swap/Primer.html, 2002.

[Bern00a] T. Bernes-Lee, J. Hendler, and O. Lassila. *Semantic Web*. In Scientific American, May 2000.

[Bobr66] D. G. Bobrow. *Natural language input for a computer problem solving system*. In MAC-TR-1, Project Mac,1966.

[Broe02] J. Broekstra, A. Kampan, and F. van Harmelen. *Sesame: A generic architecture for storing and querying RDF and RDF Schema*. In International Semantic Web Conference, 2002. pp. 54-68.

[Broe04] J. Broeskstra and A. Kampman. *SeRQL: A second generation RDF query language*. In Semantic Web Advanced Development Europe: Workshop on Semantic Web Storage and Retrieval, 2004.

[Cull86] R. Cullingford. *Natural Language Processing: A Knowledge-Engineering Approach*. Rowman and Littlefield, Totowa, NJ, 1986.

[Deli06] Leonidas Deligiannidis, Amit P. Sheth, and Boanerges Aleman-Meza. *Semantic Analytics Visualization.* In Proceedings of the IEEE International Conference on Intelligence and Security Informatics 2006, May 23-24, 2006, San Diego, CA, USA.

[Deke03] Ofer Dekel, Christopher D. Manning, Yoram Singer. *Log-Linear Models for Label Ranking*. In Advances in Neural Information Processing Systems 16. Cambridge, MA: MIT Press, 2004. pp. 497-504.

[Dill03] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. *SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation.* World Wide Web Conference Budapest, Hungary (2003)

[Fell98] C. Fellbaum. *Wordnet: An Electronic Lexical Database*. MIT Press, 1998.

[Flui02] C. Fluit, M. Sabou, F. van Harmelen. *Ontology-based Information Visualisation*. In Visualising the Semantic Web, 2002.

[Gran02] J. Grant, D. Beckett, D. *RDF Test Cases*. W3C Working Draft. http://www.w3.org/TR/2002/WD-rdf-testcases-20021112, 2002.

[Gruh04] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. *How to build a webfountain: An architecture for very large-scale text analytics.* In IBM Systems Journal - Utility Computing, 2004.

[Guha02] R. Guha and R. McCool. *Tap: A Semantic Web Platform*. In Computer Networks 42, 2002. pp. 557-577.

[Guth99] L. Guthrie, J. Guthrie, and J. Leistensnider. *Document classification and routing in Natural Language Information Retrieval*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999. pp. 289-310.

[Hamm02] B. Hammond, A. Sheth, and K. Kochut, *Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content*. In Real World Semantic Web Applications, pp. 29-49, 2002

[Hamm05] Tony Hammond, Timo Hannay, Ben Lund, and Joanna Scott. *Social bookmarking tools (I): A general review*. In D-Lib Magazine, 2005.

[Jone99] Karen Jones. *What is the Role of NLP in Text Retrieval in Natural Language Information Retrieval*, Tomek Strzalkowski, Ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999. pp. 1-21.

[Jure04] Jure Leskovec, Marko Grobelnik, and Natasa Milic-Frayling. *Learning sub-structures of Docment Semantic Graphs for Document Summarization*. In Link Analysis and Group

Detection, 2004.

[Karv02] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. *RQL: A Declarative Query Language for RDF*. In Word Wide Web Conference, 2002.

[Katz02] Boris Katz, Jimmy Lin. *Annotating the Semantic Web using natural language*. In Proceedings of the 2nd Workshop on NLP and XML, 2002.

[Klei02] D. Klein, and C. Manning. *Fast Exact Inference with a Factored Model for Natural Language Parsing*, In Proceedings of Neural Information Processing Systems, 2002.

[Kuhn06] Tobias Kuhn, Loic Royer, Norbert E. Fuchs, Michael Schroeder. *Improving Text Mining with Controlled Natural Language: A Case Study for Protein Interactions*. In Third International Workshop on Data Integration in the Life Sciences, Hinxton, UK, 2006.

[Mahe95] K. Mahesh, and S. Nirenburg, *A Situated Ontology for Practical NLP*. In Proceedings Workshop on Basic Ontological Issues in Knowledge Sharing, 1995.

[Marc93] Mitchell Marcus, Beatrice Santorini, and Maryann Marcinkiewicz. *Building a large annotated corpus of English: the Penn Treebank.* In Computational Linguistics,1993.

[McBr02] B. McBride. *Jena: A semantic Web toolkit*. In IEEE Internet Computing, 2002. pp. 55-59.

[Mitk98] R. Mitkov. *Robust pronoun resolution with limited knowledge*. In Proceedings of the Eighteenth Conference on Computational Linguistics, 1998.

[Popo03] B. Popov, A. Kiryakov, A. Kirilov, et al. *KIM - Semantic Annotation Platform.* In Proceedings of the 2nd International Semantic Web Conference, 2003.

[Scha75] R. C. Schank. *Conceptual Information Processing*. Elsevier, New York, 1975.

[Seab04] A. Seaborne. *RDQL—A Query Language for RDF*. W3C, Member Submission. http://www.w3.org/Submission/2004/SUBM-RDQ-20040109, 2004.

[Shet03] A. Sheth, I. B. Arpinar, and V. Kashyap, *Relationships at the Heart of Semantic Web: Modeling, Discovering, and Exploiting Complex Semantic Relationships*, In Enhancing the Power of the Internet Studies in Fuzziness and Soft Computing, 2003.

[Shet05] A. Sheth. *From Semantic Search and Integration to Analytics.* In Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005.

[Srin02] Suresh Srinivasan, Thomas C. Rindflesch, William T. Hole, Alan R. Aronson, and James G. Mork. *Finding UMLS Metathesaurus Concepts in MEDLINE*. Proceedings of the American Medical Infomatics Association, 2002.

[Svat03] V. Svatek, J. Braza, and V. Sklenak. *Towards Triple-Based Information Extraction from Visually-Structured HTML Pages*. In Poster Track of the 12th International World Wide Web Conference, Budapest, 2003.

[Tetr99] Joel R. Tetreault. *Analysis of syntax-based pronoun resolution methods*. In Proceedings of ACL, 1999.

[Weiz66] J. Weizenbaum. *ELIZA - A computer program for the study of natural language communications between men and machines.* In Communications of the Association for Computing Machinery,1966. pp. 36-45.

[Wino72] T. Winograd. *Understanding Natural Language.* Academic Press, New York, 1972.

[Wood75] W. A. Woods. *What's in a Link: Foundations for Semantic Networks*. In

Representation and Understanding: Studies in Cognitive Science. Academic Press, 1975. pp. 35-82.