2010

Ontology-guided extraction of structured information from unstructured text: Identifying and capturing complex relationships

Sushain Pandit
Iowa State University

Ontology-guided extraction of structured information from unstructured text: Identifying and capturing complex relationships

by

Sushain Pandit

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Vasant Honavar, Major Professor
Dimitris Margaritis
Samik Basu

Iowa State University

Ames, Iowa

2010

DEDICATION

Dedicated to Supreme God, my parents and Sachin Tendulkar's world record double-hundred.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to everyone who helped me in conducting the research leading to this thesis. Firstly, I would like to thank my advisor, Dr. Vasant Honavar for his guidance throughout the period of this research work. I would also like to thank my committee members, Dr. Dimitris Margaritis and Dr. Samik Basu for their thoughtful insights and pointers. Further, I would like to acknowledge useful discussions with Neeraj Koul, Heyong Wang and Ganesh Ram Santhanam, members of the Artificial Intelligence Research Laboratory. I would also like to warmly thank my parents for all those intangible bits of reassurance given at key moments. Finally, I would like to express my gratefulness to the Center for Integrated Animal Genomics and the Center for Computational Intelligence, Learning, and Discovery for the research assistantship that provided partial support for me during my graduate studies at Iowa State University.

ABSTRACT

Many applications call for methods to enable automatic extraction of structured information from unstructured natural language text. Due to the inherent challenges of natural language processing, most of the existing methods for information extraction from text tend to be domain specific. This thesis explores a modular ontology-based approach to information extraction that decouples domain-specific knowledge from the rules used for information extraction. Specifically, the thesis describes:

1. A framework for ontology-driven extraction of a subset of nested complex relationships (e.g., Joe reports that Jim is a reliable employee) from free text. The extracted relationships are semantically represented in the form of RDF (resource description framework) graphs, which can be stored in RDF knowledge bases and queried using query languages for RDF.

2. An open source implementation of SEMANTIXS, a system for ontology-guided extraction and semantic representation of structured information from unstructured text.

3. Results of experiments that offer some evidence of the utility of the proposed ontology-based approach to extract complex relationships from text.

CHAPTER 1. OVERVIEW AND MOTIVATION

This chapter provides an introduction to the main topics and the motivation behind the thesis. A brief overview of our contributions and the outline of the overall structure of the thesis is also provided.

1.1 Information Extraction

Information-driven advanced AI problems as well as semantic computing issues dealing with linked-data and semantic web, have long warranted the need of domain-specific, structured and consumable information. Keeping with the philosophical point of view of Stoll and Schubert, "Data is not information, Information is not knowledge, Knowledge is not understanding, Understanding is not wisdom", it is fairly easy to draw contrasts [1] between such *domain specific and consumable* information and raw data that is often openly available and constantly generated at the rate of over a tera-bytes per day. In essence, the constantly growing data becomes rather useless if we are unable to extract meaningful, relevant and consumable information out of it.

This overarching need for extracting structured information from raw data has motivated various systematic processes for information extraction. For a while such processes were largely manual, with domain experts semi-automatically extracting relevant information constructs from data sources (web, text, images, publicly available structured data, etc), validating them against their predefined set of domain-specific rules, and organizing them into a useful formal representation.

Recently, however, there has been a lot of attention on building systems that try to automatically (semi or fully) extract information from free text, validate it against a domain description and make a coherent representation out of it. Our work in this thesis focuses on this particular paradigm and we describe it in more detail in following sections.

1.2 Extracting Domain-specific Semantic Information from Text

The general problem of interpretation of natural language texts is very difficult [12], however there have been significant improvements that directly relate to the feasibility of such interpretations: (i) Natural Language Processing (NLP) to enhance parsing and sentence-specific semantic interpretations; and (ii) formal representations to capture domain knowledge, such as ontology, taxonomies, etc.

Advances in NLP have enabled us to extract relationships and entities, which are essential building blocks of any sort of information that is extracted from text. Further, the growth and acceptance of open World Wide Web Consortium (W3C) [47] standards for encoding and representing knowledge, such as Web Ontology Language (OWL) [43] and resource description framework (RDF) [40], have made it easier to express domain-specific information in a consumable form.

Overall, structured information extraction encompasses three fundamental steps: (i) NLP-based relationship and entity extraction, (ii) semantic mapping and validation of the extracted constructs, and (iii) representation of the extracted constructs in a generic formalism. However, each of these steps has numerous open issues and inherent challenges, and we discuss some of them in the next section, followed by our motivations to address them.

1.3 Open Problems and Challenges

1.3.1 Issues in NLP-based Relationship and Entity Extraction

Generally, information extraction algorithms extract relations as: (i) simple verbs based on speech tagging [7], (ii) complex associations based on dependency parses [2], or (iii) induced relations through term co-occurrence in large text corpora. Similarly, entities are generally extracted as: (i) simple nouns, (ii) modified, complex nouns. These algorithms make use of certain extraction patterns, which may be statically encoded in the form of rules, or dynamically induced by various machine learning approaches on large text corpora.

Some of the challenges encountered by conventional algorithms utilizing static rules, include: (i) over-dependence on parse trees, resulting in inability to extract indirect, implicit and complex relationships, and (ii) the need for named-entities to guide relationship extraction. On the other hand, approaches based on rule-induction require enormous amount of textual resources across a fairly wide-range of text corpora, to be able to induce generic rules.

1.3.2 Issues in Semantic Mapping and Validation

A domain-specific information extraction process needs to make sure that the extracted relationships and entities (information constructs) are semantically mapped and validated against a domain description. This domain description is often described in the form of an ontology [46], capturing the underlying relations and concepts occurring within the domain. This step raises a number of challenges, some of which are: (i) operating in an incomplete description of the domain, viz., the case when the ontology description does not capture all the relations and concepts, and (ii) enriching the ontology with new found knowledge without introducing inconsistencies.

1.3.3 Issues in Generic Representation

Finally, we need to represent the extracted information using a specification that is widely accepted and consumable. This step is critical in the overall process of information extraction since in its absence, the extracted information will exist in an independent world, which is highly undesirable, considering the amount of recent attention that linked data [42] and knowledge integration [10], [11] have received.

One of the most common mechanisms for representing information in machine consumable form, is the W3C family of specifications: RDF, RDF schema, and OWL. Thus, it becomes an interesting issue to be able to represent the extracted information constructs within the existing W3C specifications.

With these issues in mind, we motivate the need for our work in the following section.

1.4 Motivation for Creating a Novel Information Extraction Framework

1. Our main overarching motivation stems from the need to identify, extract, validate and represent complex relationships *(e.g., Joe reports that Jim is a reliable employee)* in order to facilitate comprehensive information extraction from unstructured text.

   Such relationships often occur in a non-standard form where one part of the sentence refers to the other part through an inherent dependency relationship, or the underlying information in the sentence is not suitable for a straightforward extraction task. The issue with such complex relationships is that at least one of the three steps in the process of information extraction becomes problematic. In our experiments, we saw one or more of the following problem patterns: (i) the first step required special rules to be able to identify and extract the relationships, (ii) the validation step required enrichment of the existing ontology since it made sense to capture the information construct corresponding to the extracted relationship, however, the existing definitions in the given domain ontology were not sufficient to do that, and (iii) the representation step required certain finesse to express the complex relationship within the given set of specifications.

   In chapter 3, we describe some interesting examples that motivated the need for capturing these complex relationships in spite of the inherent difficulty of the task, and clearly identify the subset of complex nested relationships that we intend to handle.

   Encapsulated within this main motivation, there are several ideas that we elaborate below:

   - A common approach for extracting new relationship patterns is to utilize rule-induction algorithms [30] that induce rules in a semi-supervised manner to capture newly found relationships. However, we believe that such automatic rule induction is not the best way to deal with complex relationships because of their complexity, uniqueness, the amount of text corpora and time required to figure out whether the induced rule is generic enough to capture the corresponding complex relationship pattern occurring across a significant enough collection of texts. Instead, we try to experimentally arrive at generic rules to capture our subset of complex relationship patterns.

- Similarly, one of the more common approaches to handle information constructs that can not be validated against a given domain ontology, is to simply ignore them. However, we argue that there are simple ontological extensions that can be made without losing consistency, to be able to capture such cases.

- Finally, there are many highly-expressive logics, such as description logic [48] (OWL-DL [43], OWL-Full [43]), F-logic [44], which may be utilized to express complex extracted information. However, we argue that such formalisms become an overkill while trying to express free text and thus, to the extent possible, we try to capture complex information while retaining the simplicity of RDF specification.

2. Our second motivation follows from the first one in that we hope to formulate the set of rules that would work best to extract certain complex relationship types. In order to do this, it is necessary to have a suitable information extraction system that allows us to run experiments and test the performance on large text documents.

3. Our third motivation is to enable complex queries in order to answer certain questions about the information expressed within unstructured text. Since we intend to extract complex relationships using our proposed framework, it becomes interesting to experimentally analyze and determine the kind of questions that can be answered once we have extracted a large set of linked information through these relationships and entities.

1.5 Goals

Against this set of background, challenges and motivations, we formulate various frameworks and algorithms in this thesis, to address some of these concerns as briefly described below:

1. **Complex Relationships**: We identify a subset of complex relations *(e.g., Joe reports that Jim is a reliable employee)* that are especially interesting from the perspective of structured information extraction (out of the many complicated sentential structures that occur in free text) and have an above average likelihood of occurrence in a random sample of free-text. We clearly define their scope w.r.t what we intend to cover within our extraction framework.

2. **Entity and Relationship Extraction**: Having clearly defined the type and nature of complex relationship patterns that we intend to extract, we precisely formulate the sets of extraction rules that can identify and extract them. We also formulate an algorithm that utilizes these rules to extract relationships and entities (information constructs) from unstructured text.

3. **Semantic Validation**: Further, we propose methodology to validate and semantically associate the extracted information constructs against a given domain description captured in an ontology. We also include ways to perform ontology enrichment with newly found relationships, whenever possible.

4. **Representation**: We formulate a methodology to represent these extracted and validated information constructs using existing (non-extended) Resource Description Framework (RDF) specification. We especially introduce mechanism to represent the complex relationships in a way that enables the possibility of complex querying on the extracted information.

5. **Implementation and Empirical Evaluation**: Based on our analysis and algorithms, we provide implementation of an information extraction system, called SEMANTIXS (System for Extraction of doMAin-specific iNformation from Text Including compleX Structures), which can be utilized to extract and semantically represent structured information from free text. We describe the architectural and design details of the system. We also report results of some experiments that offer evidence of the utility of the proposed ontology-based approach to extraction of complex relationships from text. The system

implementation is open-source (under GNU General Public License), and is available at http://sourceforge.net/projects/semantixs/.

1.6 Thesis Outline

The remainder of this thesis is organized as follows:

- In Chapter 2, general terminology related to the steps in structured information extraction is introduced. Preliminary concepts for domain ontologies, knowledge bases, RDF specification, etc are also provided. A brief survey of related work is also given.

- In Chapter 3, we present our information extraction from text framework along with the complex relations that we intend to extract. We also present the corresponding rules to extract those relations. Further, we talk about the frameworks that take care of validation and enrichment, and the RDF representation that we utilize for expressing extracted information. We conclude the chapter with a discussion on the algorithms incorporating these frameworks.

- In Chapter 4, we describe the system architecture, design and other details of *SEMANTIXS*, an information extraction system implemented based on the above analysis and algorithms.

- In Chapter 5, we describe our experiments and results achieved using SEMANTIXS and free text processing.

- We conclude the thesis work in Chapter 6. We summarize the work done in the thesis along with the main contributions. We also provide a number of interesting future threads of investigation that directly relate to this research.

CHAPTER 2. PRELIMINARIES AND RELATED WORK

This chapter provides preliminaries, definitions and problem formulations for this thesis. We also give an overview of the related work in the area of information extraction from text.

2.1 Derivative Structures: Parse Trees and Dependency Graphs

Due to complete lack of semantics in unstructured text, for any task involving information extraction from text, it becomes crucial to work with an intermediate representation of text. For this purpose, any such task employs linguistic parsers to perform syntactic analysis of text to determine its grammatical structure with respect to a formal grammar for the language[1] that it is operating in.

The parsing process results in the generation of parse trees and optionally, dependency graphs that are data structures derived from the text itself, which capture the implicit structure within the tokens in the input text. In the remainder of this section, we give a brief overview of the terminology related to these data structures. We limit our discussion to the scope of this thesis.

2.1.1 Parse Tree

A parse tree is an ordered and rooted tree that represents the syntactic structure of a sentence. In this section, we describe the Penn Treebank Notation utilized by most parsers for tagging the sentence before generating the parse tree. These tags are often utilized in information extraction systems to formulate rules based on which they perform extraction.

[1]In this work, we always assume that the text is from the English language; more specifically, all that is captured by the englishPCFG.ser provided by with the Stanford Parser package

2.1.1.1 Penn Treebank Notation

These are simplified forms of the definitions found in the Penn Treebank manual[2].

- S: Simple declarative clause
- NP: Noun Phrase. Phrasal category that includes all constituents that depend on a head noun.
- VP: Verb Phrase. Phrasal category headed a verb.

2.1.2 Dependency Graph

A dependency graph is a data structure that captures the implicit dependencies (sentential semantics) between tokens (words) in a sentence. In this section, we give a formal definition of dependency graph that is useful for our analysis, and define various dependency relations that we use in our work.

Definition: (*Dependency Graph)* Given an English language sentence T comprising of a word-set W representing the words in the sentence, a dependency graph, G of the sentence T is defined as a directed graph with node-set W and a labeled edge-set connecting the nodes in W s.t. for any two connected nodes, the label on the edge represents the dependency relationship between the words represented by the nodes.

Figure 2.1 shows a dependency graph for the sentence - “Heart attack causes reduced average lifespan”.

For purposes of our work, we describe the following set of dependencies that are necessary for understanding our rule framework in Chapter 3. These are simplified forms of the definitions found in the Stanford typed dependencies manual[3].

- **amod: adjectival modifier** - An adjectival modifier of an NP is any adjectival phrase that serves to modify the meaning of the NP.

[2]Refer - ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual/notation.tex for a complete list
[3]Refer - http://nlp.stanford.edu/software/dependencies_manual.pdf for a complete list

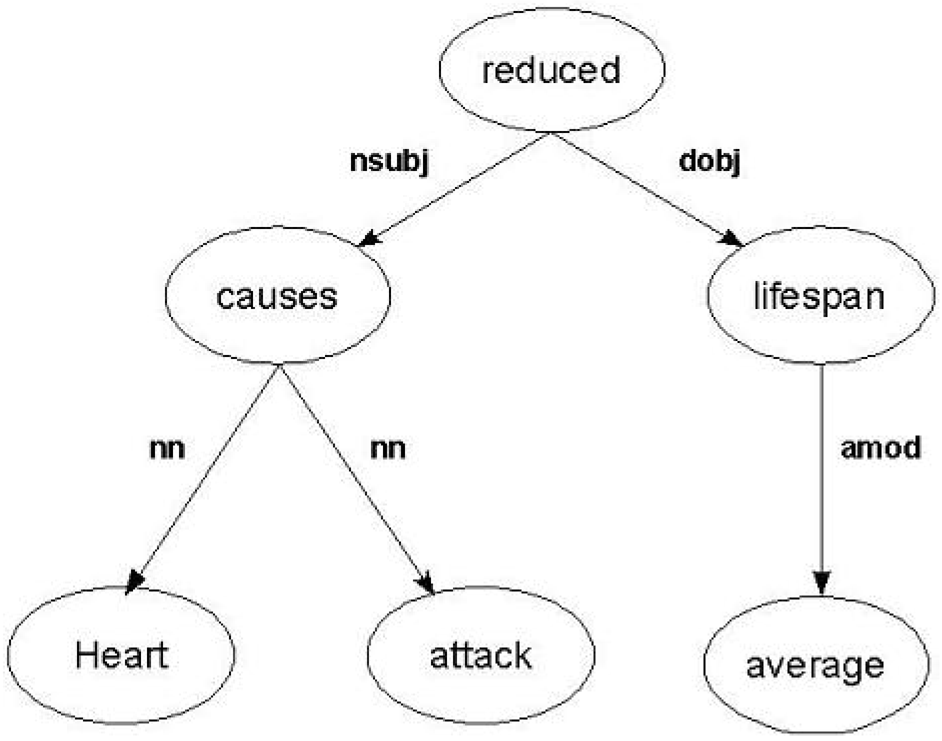


Figure 2.1 A Simple Dependency Graph

- **ccomp: clausal complement** - A clausal complement of a VP or an ADJP is a clause with internal subject which functions like an object of the verb or of the adjective; a clausal complement of a clause is the clausal complement of the VP or of the ADJP which is the predicate of that clause. Such clausal complements are usually finite (though there are occasional remnant English subjunctives).

- **conj: conjunct** - A conjunct is the relation between two elements connected by a coordinating conjunction, such as “and”, “or”, etc. We treat conjunctions asymmetrically: The head of the relation is the first conjunct and other conjunctions depend on it via the conj relation.

- **dobj : direct object** - The direct object of a VP is the noun phrase which is the (accusative) object of the verb; the direct object of a clause is the direct object of the VP which is the predicate of that clause.

- **neg: negation modifier** - The negation modifier is the relation between a negation word and the word it modifies.
- **nn: noun compound modifier** - A noun compound modifier of an NP is any noun that serves to modify the head noun.
- **nsubj : nominal subject** - A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb.
- **parataxis: parataxis** - The parataxis relation (from Greek for "place side by side") is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, a clause after a ":" or a ";".
- **pobj : object of a preposition** - The object of a preposition is the head of a noun phrase following the preposition. (The preposition in turn may be modifying a noun, verb, etc.)
- **prep: prepositional modifier** - A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, or noun.

2.2 Formal Specifications for Validation and Representation

As briefly mentioned in the previous chapter, a structured information extraction task requires a formal specification in order to be able to make sense of the extracted constructs (entities and relationships). There are various representation mechanisms utilized for expressing a domain description, however, one of the most widely used is an ontology specification. We give a brief overview of this next.

2.2.1 Domain Ontology, Instances and Knowledge Base

Definition 1: (*Ontology*) An ontology is a structure $\mathcal{O} = (\mathbb{R}, \mathbb{C})$ such that:

- The sets $\mathbb{R}$ and $\mathbb{C}$ are disjoint and their elements are called relations and concepts respectively.
- The elements of $\mathbb{R}$ induce a strict partial order $\prec$ on the elements in $\mathbb{C}$. This ordering of the form, $c_i \prec c_j$, $c_i, c_j \in \mathbb{C}$, is called the concept hierarchy.

Since the ontology merely defines the concepts and relationships in the domain, for a process intending to extract domain-specific information, it is highly desirable that there be a specification of assertions that may occur in the domain.

As an example, consider a sentence within the *Sports* domain - *Sachin Tendulkar scored 200 runs.* An ontology for *Sports* may have a concept *Cricketer* and a relation *scoredRuns* with domain *Cricketer* and range *Numeric.* However, unless there is another specification that asserts *Sachin Tendulkar* as a *Cricketer*, an information extraction system can not correctly extract this as a domain-specific construct.

With this motivation, we define the notion of a domain ontology with instances (a.k.a knowledge base), which is a combination of a domain description (in the form of an ontology with concepts and relations) and certain concrete assertions (instances of those concepts) about it.

Definition 2: (*Domain Ontology with Instances*) A domain ontology with instances is defined with respect to an ontology $\mathcal{O}$, as a structure $\mathcal{DOI} = (\mathcal{O}, \mathbb{I})$ such that:

- $\mathbb{I}$ is a set, whose elements are called instances.
- There exists a function $h : \mathbb{I} \longrightarrow \mathcal{P}(\mathbb{C})$, where $\mathcal{P}(\mathbb{C})$ is the power-set of the set of concepts for the ontology $\mathcal{O}$.

2.2.2 Validation Process

Given a sentence as input, the process of information extraction extracts certain words from the sentence, which it identifies as possible information candidates. In this thesis, we always refer to them as *candidate information constructs.* These candidates are then validated against the input domain description specified as an ontology. Upon validation, they become *information constructs* ready to be represented using a formalism. We describe the validation process in detail when we present our frameworks in Chapter 3.

2.3 Generic Representation

Information extraction systems utilize various formalisms to represent the information constructs. In this section, we present a brief overview of one of the popular representation mechanisms, specified by W3C.

2.3.1 Resource Description Format(RDF)

RDF utilizes the idea of representing statements in *subject-predicate-object* format. Each such realization is called an RDF *triple*. As an example, lets reconsider the representation of example from section 2.2.1. Once the information extraction process extracts and validates the constructs from the sentence - *Sachin Tendulkar scored 200 runs*, this extracted information can be simply represented in the triple format as - *Sachin Tendulkar-scoredRuns-200.*

RDF also provides advanced capability to make assertions about statements instead of entities. This process is called RDF *Reification* [41]. We make use of this capability to represent our complex relationships described in Chapter 3.

2.4 Problem Definition

With this background, we can now define the problem of structured information extraction from free text in a somewhat concrete manner as follows.

Definition 3: Given a text fragment, Z, consisting of sentences $\{T_i\}$ with word-sets $\{W_i\}$, a domain description captured in an ontology $O(R, C)$, a set of instances Y, and a function $h : Y \longrightarrow \mathcal{P}(C)$, where $\mathcal{P}(C)$ is the power-set of C, structured information extraction on Z is defined as a process that must do the following:

1. Determine a set T_{CTR} of candidate triples using an extraction algorithm.

2. Validate T_{CTR} to find a set $\{K = \{s_i, p_i, o_i\}\}$ of triples with respect to $O(R, C)$, Y and h.

3. Represent triples in K using a suitable representation mechanism.

This process is a hybrid of in-formalism (free-text, NLP, extraction algorithms, etc) and formalism (ontology-based validation, etc). In the next chapter, we discuss our formulation of each of these steps and make this hybrid notion more clear. We now give a brief overview of related work.

2.5 Related Work

There have been varied efforts around the problem of information extraction from text. We move forward to a discussion of different approaches to information extraction task and distinguish our work in context of the present state-of-art. Further, we also present an overview of representative approaches in complex relationship extraction from text since that is our main motivation to formulate a novel framework for information extraction from text.

2.5.1 Different Flavors of Information Extraction from Text

There are two main approaches to information extraction distinguished by whether they rely on manual engineering to discover extraction rules by inspection of a corpus, or use statistical methods to learn rules from annotated corpora. The key advantage of the former approach is that it does not require large amounts of training corpora (which is often expensive to acquire) and that the best performing systems have been known to be hand-crafted [35]. On the flip side, the rule formation process is often laborious and in case of domain-specific rules (for instance, rules that are tightly coupled with the ontology specification), domain adaptation may require significant re-configuration. In case of latter, domain portability (in principle) is relatively straightforward due to automatic rule induction, however training data may be hard to acquire, and changes in domain specification may require re-annotation of the entire training data.

Due to the availability of well-represented descriptions of concepts, relations and instances in certain domains, like Biomedicine [39], [38], there has been significant effort on extracting domain-specific information from text. Recently, there have also been large-scale efforts [37] to extract entity mentions, facts and/or events from text. Although statistical approaches to information extraction have shown decent results for entity mentions, such as identifying genes names in Biomedical literature [refer 14], they are not as effective in case of relationship identification due to lack of annotated test corpora. This problem is further aggravated in case of complex relationships (e.g., Joe reports that Jim is a reliable employee) that occur in implicit and non-standard form as we shall see in the next chapter. As a result, there has been considerable focus on rule-based techniques that, even though being labor-intensive and requiring a lot of manual formulation, prove to be more effective and transparent in capturing the semantic criteria. Further, systems that utilize rule-based approaches, in a domain-independent manner, are easier to extend using a modular design.

As indicated above, rule-based approaches can be broadly divided into two genres based on whether the rule-formulation is tightly coupled with the domain description or not. Author in [18] presented an approach to build a dictionary of extraction patterns, called concepts or conceptnodes. However, the extraction patterns are triggered by domain-specific triggering words. Similarly, [19] and [20] generate multi-slot extraction rules and learn extraction patterns respectively, however both require a specification of at least one domain-specific word. Although [21] allows more expression power than [20], it still relies on exact word matches to some extent. Besides inducing rules coupled with the domain in some manner, all these approaches operate at a different level of granularity than our approach in that we are interested in extracting a subset of nested complex relationships (e.g., Joe reports that Jim is a reliable employee) that are completely independent of a domain and can be entirely identified by looking at a dependency graph or parse tree of a sentence. In some cases, we use trigger words, however, they are language constructs (like propositions, conjunctions, etc) and not specific to a domain.

Authors in [13] proposed an ontology-based approach to extract relationships and compound entities from Biomedical text using rules that operate on parse trees. They further suggested an unsupervised approach [16] to joint extraction of compound entities and relationships using information theoretic measures. Although the structured information extraction framework that we have proposed in our work is similar to [13] in that we also utilize a rule-based algorithm for entity and relationship identification and extraction, followed by validation against an ontology, the purpose of our work is quite different due to (i) the focus on complex relationship structures (e.g., Joe reports that Jim is a reliable employee), and (ii) formulation of rules in an entirely domain-independent manner. Similarly, [15] proposed a rule-based system to extract regulatory mechanisms from Biomedicine literature and [17] presents an ontology-based information extraction tool that extracts data with the aid of context words defined in the ontology. There are many similar efforts, such as [22], [23] that focus on domain-specific information extraction. Although these approaches capture many relationship forms and suggest a general paradigm of relationship extraction using rules, all of them are broadly motivated by extraction purposes specific to their domain of discourse. In contrast, our rule-framework is meant to capture complex relationships of very general form that have correlations with linguistic structures and not with any specific domain of discourse.

2.5.2 Complex Relationship Extraction from Text

In this section, we give an overview of representative approaches within the field of complex relationship extraction from text. A comprehensive survey of relation extraction mechanisms is given in [36]. In line with the approaches described above for the larger problem of information extraction, some of the common supervised approaches to relationship extraction formulate the problem as that of classification in a discriminative framework (naturally requiring positive and negative training examples). Issues with these approaches are that they are difficult to extend to new relationship-types (especially complex ones) due to lack of labeled data.

As an alternative, semi-supervised approaches to address the task of relationship extraction from text, are often based on pattern induction (or slot filling), with the implicit assumption that terms belonging to a common linguistic context would occur in relationships with certain common semantics. For example, [24], [30] make use of this paradigm to learn taxonomic relations and induce patterns respectively. Other popular systems using semi-supervised approach to relationship extraction include [33], [34]. Further, there are rule-based systems that fall within the genre of semi-supervised approaches (as mentioned in the previous subsection) that make use of extraction-rules on intermediate syntactic structures, like dependency graphs and parse trees, to achieve similar goals.

For complex relationships, the general statistical (or machine learning) approach is to factor them into binary relationships, train binary classifiers for capturing relatedness of entities and then form complex relationships using related entities and the binary relationships they participate in. For example, a representative approach described in [31] utilizes this approach to identify binary relationships in news text. [32] utilizes similar notions and a scoring metric on maximal cliques to discover complex relationship instances.

In most of these approaches, the working definition of a complex relationship has been taken as any *n-ary* relation in which some of the arguments may be unspecified. However, as we illustrate in the next chapter, not all candidates in our identified set of *complex* relationships conform to this definition. We intend to capture such cases when, on surface, we may have a case of binary relationship, however its arguments are not simple or compound entities, but another relationship instance. Modeling this problem within the discriminative framework raises intricate issues, such as non-uniform feature spaces in case of an entity being related to a fact (combination of entities). Although, there has been work on designing kernels to capture non-linear feature spaces, we still run into the same issue of manually labeling corpora for appropriate test-set creation. Thus, to handle this flavor of complexity, we resorted to the rule-based approaches based on intermediate syntactic structures.

It may be noted that apart from the differences pointed above, our work can be distinguished from the general task of relationship extraction from text in that we are formulating a structured information extraction framework, and relationship extraction is one of its many parts (entity and relationship extraction, semantic validation and representation).

CHAPTER 3. A NOVEL FRAMEWORK FOR EXTRACTING INFORMATION FROM TEXT

In the previous chapter, we introduced the necessary background to enable us to begin discussion of our proposed framework for information extraction from text. In this chapter, we describe our work in detail. We start with an overview of our approach, and proceed towards an elaboration of each of the individual steps in the overall process.

3.1 Approach Overview

As per definition 2.4, our overall methodology for information extraction from text is divided into three main parts: (i) processing the text sentence-by-sentence, generating the parse tree and dependency graph for each sentence and applying extraction rules to identify and extract candidate relationships and entities, (ii) validating these candidates against the domain ontology model and optionally enriching the model by new concepts and relationships that capture these candidates in a consistent manner, and (iii) representing the validated candidates in RDF notation leveraging the existing W3 syntax specification. We now address each of these parts in detail.

3.2 Text Processing and Generation of Derivative Structures

As indicated in the previous chapter, we intend to utilize both dependency graphs and parse trees in our approach to information extraction. The dependency graphs would help in extraction of complex, nested and implicit relationships, and parse trees can be leveraged for simpler relationships or whenever a finer analysis of individual entities is necessary.

For these purposes, we wanted to utilize a third-party Parsing library, which could provide both these structures out-of-the-box. After a brief evaluation of some of the popular parsing libraries [28], [26], [29], we decided to go ahead with Stanford Parser, because of its flexibility, ease of usage, speed and precision.

As a first step, we utilize the Stanford Parser to get the dependency graph and parse tree representations of a candidate sentence. These structures are then consumed by the extraction algorithm to extract candidate relationships and entities in accordance with the rule framework described in the next section.

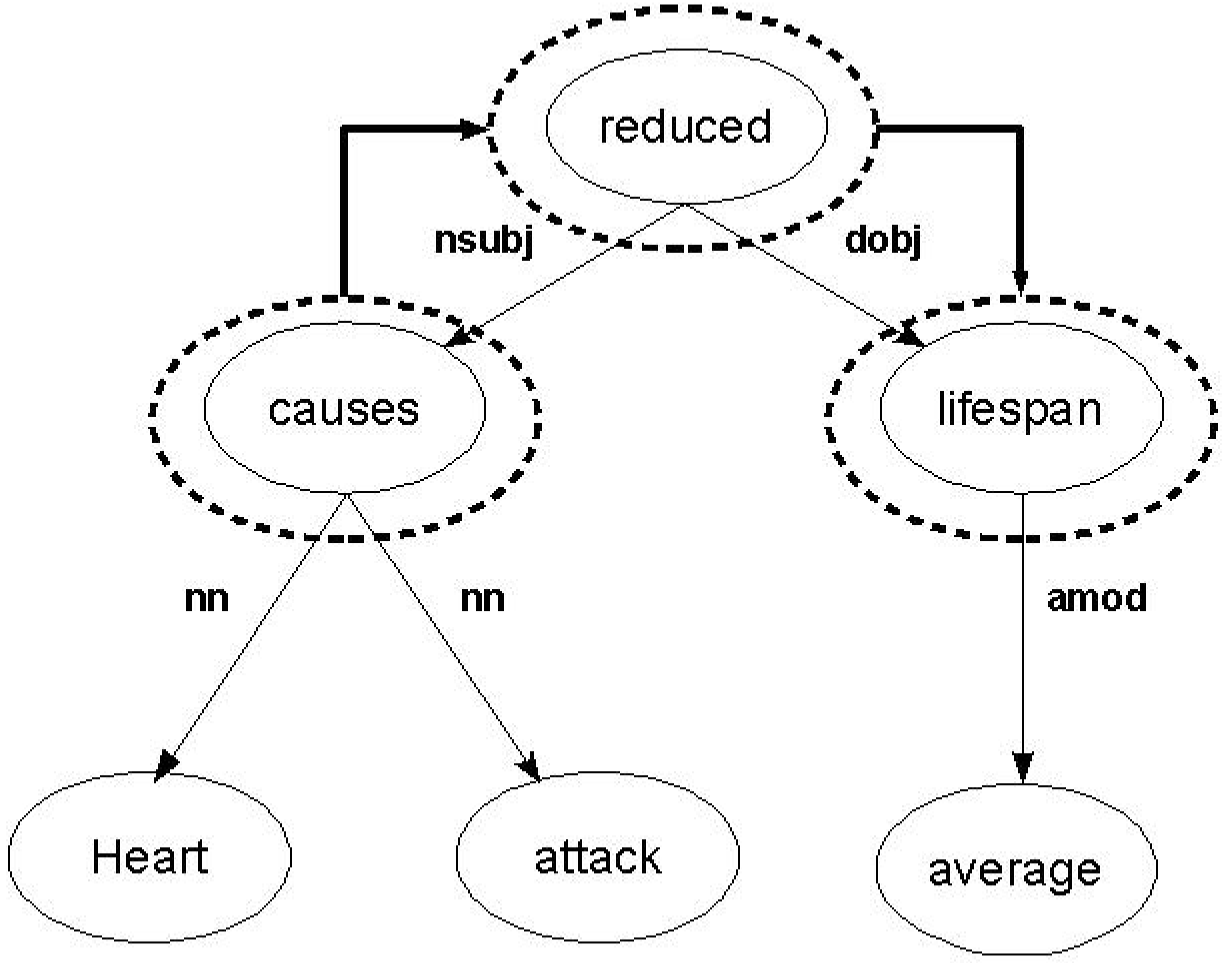


Figure 3.1 Dependency Graph with Application of a Simple Extraction Rule

3.3 Composite Rule Framework for Entity and Relationship Extraction

An extraction rule is a simple statement encapsulating a set of premises and consequents. By definition, a rule warrants the execution of actions defined in the consequents whenever the conditions defined in the premises hold. For simplicity of explanation, we will use action and consequent interchangeably. Similarly, we would use condition and premise interchangeably. For instance, an informally specified extraction rule for the dependency graph in Figure 2.1 can be - *If "labels nsubj, dobj occur along a path in the graph", then "extract that path as an information construct"*. This rule, when utilized by an information extraction algorithm, would result in the extraction of - *"Causes-reduced-lifespan"* as a candidate information construct as illustrated in Figure 3.1. With this intuitive notion in mind, we formally define the terminology that we will be using throughout the chapter.

3.3.1 Terminology

We utilize the following notations to describe the framework and algorithms that we have formulated.

p_i : i^{th} condition or premise for a rule (defined below).

c_j : j^{th} action or consequent for a rule, corresponding to a set of premises $\{p_i\}$.

$G(V, E)$: A dependency graph with vertex-set V and edge-set E.

$G_S(V')$: Subgraph of G induced by the vertex-set V'.

D : A set of labels denoting the typed dependency relations defined for the English language (refer Section 2.1.2)

$l : E \longrightarrow D$: A label function that defines a specific label from the set D, for edges of the graph G.

$P_{\{e_i\}}$: A labeled path in the graph G, comprising of the edge-set $\{e_i\}$.

The information extraction rules, utilized by the algorithms that we will be eventually describing, are of the following form:

<u>Definition 4:</u> (*Extraction Rule*) For a dependency graph G, we define an extraction rule as,

$r_i : \{p_i\} \longrightarrow \{c_i\}$, meaning $If\, \{p_i\}\, holds,\, perform\, \{c_i\}$.

This generic definition of extraction rule can have various realizations based on the usage scenario and complexity of the relationships encountered in the information extraction task being performed, one of which is described below for illustration.

$r_i : \{\exists P_{\{e_i\}} \,|\, \{l(e_i)\} = D', D' \subset D\} \longrightarrow$ {Extract all the vertices associated with the edge-set $\{e_i\}$}

Here, r_i encodes the rule that if there exists a specific sequence of dependency labels without considering the order (defined by the set D) along some path in the dependency graph of the given sentence, then the sequence of nodes (which represent words in the corresponding sentence) forms an information construct and thus, the algorithm is recommended to extract them.

3.3.2 Identifying and Defining the Relationship Types

In this section, we try to clearly identify and define the subset of complex relationships that we intend to capture by our rules. Since relationships can not exist without the entities that they are associating, we also give a description of the entities that we expect to identify and extract.

Given an English language sentence, t, we define the complexity of the relationship(s) expressed between the entities in t, by analyzing the type, nature and number of dependencies that exist within the words of t.

3.3.2.1 Simple Relationships

Intuitively, a simple relationship is expected to have a single verb connecting two entities, which may or may not have a modifier. It is not expected to have internal clauses, implicit dependencies, multiple subjects or objects. Formally, we say that a sentence contains a simple relationship if all of the following conditions hold for the dependency graph G generated from the sentence T:

- It contains no more than one subject and object each. This further implies that it has at most one dependency of type *nsubj* and one from the set $\{dobj, pobj\}$.
- It does not contain any clause-level dependencies, conjunctions, or a clausal subject. Further, it can only contain noun-compound, or adjectival modifiers. In terms of Stanford dependencies, this implies that it does not contain any dependencies from the set $\{ccomp, xcomp, acomp, compl, conj, csubj, csubjpass\}$. Also, it can only have $\{amod$, $quantmod$, $nn\}$ as modifiers.

  It is conceded that due to the extremely diverse nature of English language sentence construction, there may be many sentences that satisfy our intuitive notion of simple relationships and yet, they may not be captured by the conditions stated above. We are indeed cognizant of this fact and thus, do not claim the above conditions to be exhaustive in nature. We arrived at these conditions following an experimental approach in which we manually analyzed the dependency graphs emerging from varied sentences and tried to determine common patterns. However, for the purposes of this thesis, we limit the scope of definition of simple relations to the conditions above.

3.3.2.2 Complex Relationships

Intuitively, the set of complex relationships should be expected to capture every relation that is not simple. However, for purposes of this thesis, we limit the scope of complex relations to the following specific cases. For ease of understanding, we illustrate these cases with descriptive examples.

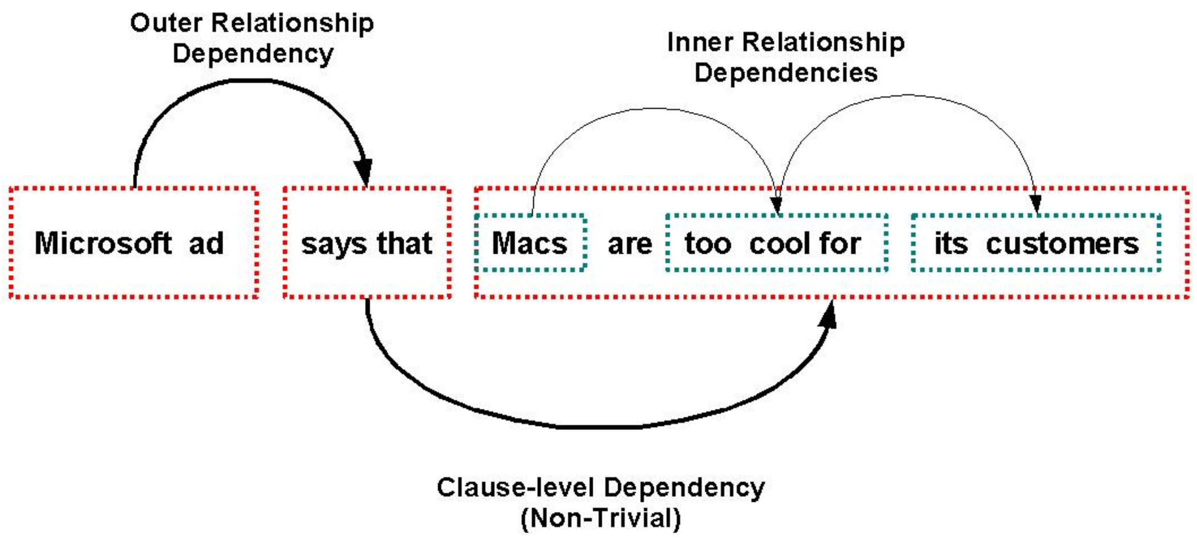

Outer Relationship Dependency
Inner Relationship Dependencies
Microsoft ad
says that
Macs
are
too cool for
its customers
Clause-level Dependency (Non-Trivial)

Figure 3.2 Example for a Relationship with Internal Clauses

1. **Case when the relationship has internal clauses**

   This type of relationship is distinguished by the fact that it has a main subject that refers to an internal clause through a verb. Such an internal clause is often interpreted as the object of the verb that it is dependent upon. These relationships make for very interesting candidates as far as information extraction from text is concerned because they implicitly encapsulate relationships between entities and facts, instead of two simple entities. We illustrate the essence of this notion through the example shown in Figure 3.2. The sentence taken from an online article[1], makes an assertion about *Macintosh PCs (Macs)*. A information extraction system, operating in the PC and Technology domain, may capture this assertion as an information construct about the entity, Macintosh PC. However, it is clear that the assertion is an opinion, expressed by another Technology vendor (Microsoft). For a generic information extraction system, it is important to capture this inter-clausal dependency and thus, we selected this relationship type in our subset of complex relationships.

   Before we go further into our analysis using the example above, it is important to note

[1]http://www.macworld.com/article/139691/2009/03/

that it captures the general structure of this relationship type comprehensively. This is since the verb relating the main clause with the inner clause (in this case, *says*) may as well be any other verb, or modified verb. Similarly, the main clause (in this case the main subject) may as well be any other noun, modified noun, or a pronoun (we perform simple pronoun resolution in our algorithm). Further, we do not constraint the inner clause in any way. Thus, this example, although specific in nature, captures any such generic sentence with an inter-clausal relationship.

One of the most troubling issues in handling natural language texts is its varying nature leading to different representations that all imply the same fact. For instance, here are a few other representations of the sentence in Figure 3.2, in the order of increasing difficulty with respect to information extraction task:

(a) "That Macs are too cool for its customers, says Microsoft ad."

(b) "Microsoft ad says: Macs are too cool for its customers."

(c) "Macs are too cool for its customers –Microsoft ad."

(d) "Hey, listen to what Microsoft ad says.. Macs are too cool for its customers !..."

(e) "Macs are too cool for its customers.. I do not say this, Microsoft ad does.. see for yourself."

These instances made us quickly realize that it would be highly improbable to formulate rules to identify complex relationships out of sentences that contain extraneous words, which are not directly related to the main theme captured in the sentence (*c, d and e* above). This is since, unlike with simple relationships where we can base our analysis on the main (unique) verb, complex relationships can have multiple verbs, subjects and objects with complex and implicit dependencies that can be hard to identify and extract in cases when there are extraneous words. This is since these extraneous words introduce dependencies that are not directly related to the underlying information captured in the sentence.

Thus, we specifically restrict the scope for our rule-coverage to the forms captured by cases *a and b* above, apart from the main sentential form illustrated in Figure 3.2. The other motivation for the restriction of scope is that we were able to identify suitable Stanford dependency relations for each of these cases, which formed the basis of our rule formulation.

2. **Case when modifiers implicitly qualify the meaning of the relationship**

   In this type of relationship, there is a main clause whose meaning is either being qualified by a prepositional modifier, or the presence of the modifier gives an entirely different context to the main clause. These relationships are again interesting candidates for information extraction since they contain qualified information, which may or may not be true on its own at the time of extraction, however their extraction can lead to interesting new discoveries at a later point. This is since there may be a point in future when the previously qualified piece of information becomes true without the qualification and it may be possible to assert certain credibility to the source that had originally claimed the qualified information (the source claiming the qualified information could have been captured using the extraction rules for the relationship type 1).

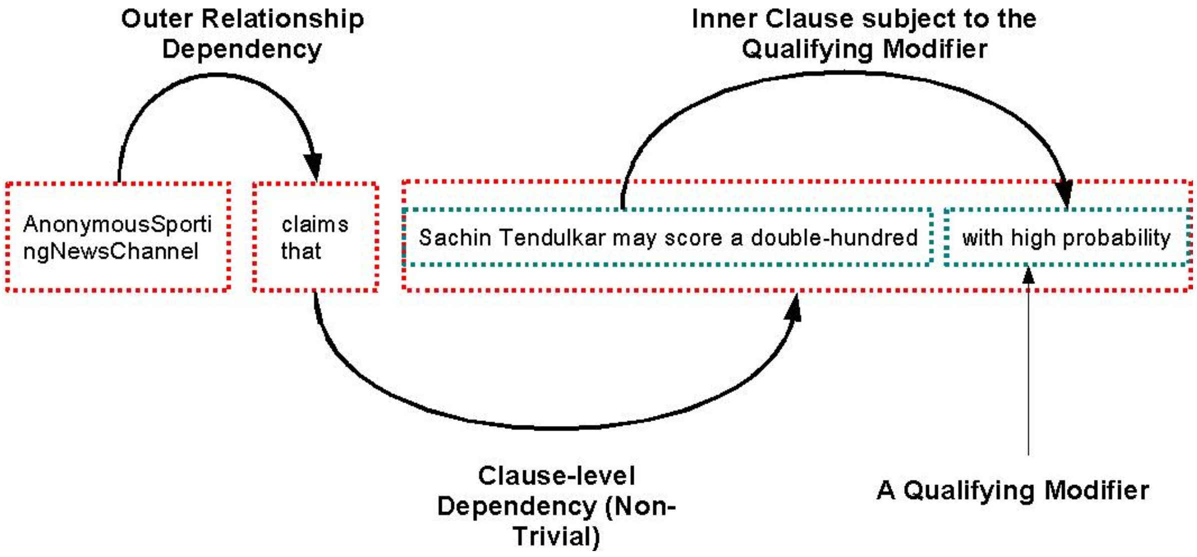

Outer Relationship Dependency
Inner Clause subject to the Qualifying Modifier
AnonymousSportingNewsChannel
claims that
Sachin Tendulkar may score a double-hundred
with high probability
Clause-level Dependency (Non-Trivial)
A Qualifying Modifier

Figure 3.3 Example Illustrating the Intuitive Motivation behind Case 2

As an example, lets consider the sentence - *AnonymousSportingNewsChannel claims that Sachin Tendulkar may score a double-hundred with high probability.* It may be observed that this sentence contains a main clause (*AnonymousSportingNewsChannel claims that*) referring an internal clause. The internal clause can be captured by extraction rules for relationship of case 1. Further, if we are able to identify and extract the internal clause (*Sachin Tendulkar may score a double-hundred*) with the qualification (*high probability*), we would be able to develop interesting insights about the source (*AnonymousSportingNewsChannel* at a later point in time as discussed above. This idea is concisely illustrated in the Figure 3.3.

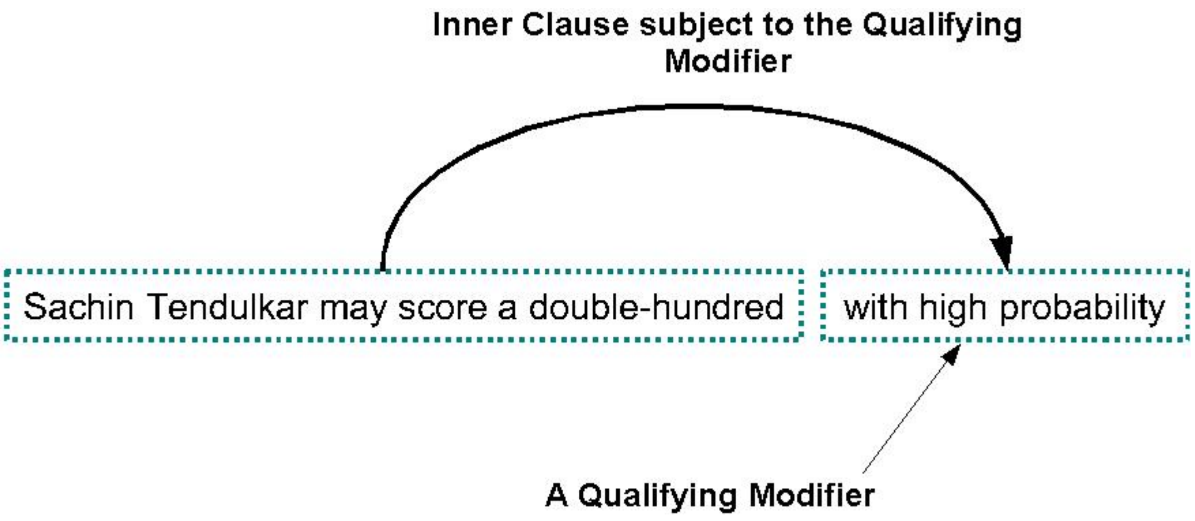

Inner Clause subject to the Qualifying Modifier
Sachin Tendulkar may score a double-hundred
with high probability
A Qualifying Modifier

Figure 3.4 Example for a Relationship with a Qualifying Modifier

With this motivating example in mind, we decided that it is important to capture the qualified structure expressed in such relationships, and thus, we selected this relationship type as the second case for our subset of complex relations. In rest of this section, we will be referring to the sentence shown in Figure 3.4.

Before we go further into our analysis using the example above, we need to make sure that the example captures the general structure of this relationship type. Unfortunately, this case is not as straightforward as the previous one and we need to consider some variations. First of all, we would need to generalize the connector for the modifying qualifier (in this example, *with*). Most obvious choice is to consider the set of all common prepositions. However, we need to only consider those, which can (i) have an associated adjective that can act as the value of the qualification (in this case, *high*, and (ii) be relevant in the sense of capturing a relational modification (iii) be supported by consistent Stanford typed dependency labels. With these observations and for the sake of reduced overall complexity, we restricted our set to the following prepositions - {*with, of, in, for*}.

Further, after performing some experiments to understand the effect of verb-clause (in this example, *may score*) variation, we found that there are subtle variations in the dependency graphs generated from very similar sentences. For instance, all of the following sentences lead to slightly varying dependency graphs when compared with our original sentence of Fig 3.4:

(a) "There is a high probability that Sachin Tendulkar may score a double-hundred."

(b) "Sachin Tendulkar visits USA with his entire family."

(c) "Sachin Tendulkar scores a hundred with absolute masterclass."

In contrast, if we keep the verb-clause unchanged, we get similar dependency graphs, even though we may vary the overall representation of the main sentence. For instance, all of the following sentences lead to similar dependency graphs:

(a) "With high probability, Sachin Tendulkar may score a double-hundred."

(b) "Sachin Tendulkar, in superb form, led the charge."

(c) "Sachin Tendulkar, with high probability, may score a double hundred."

Since we utilize dependency graphs to formulate our rules, the case with same verb-clause presents no real challenge. In fact, it results in better sentential coverage for our rules. However, to handle the case of varying dependency graphs, we had to slightly generalize our rules so that they could accommodate subtle variations in dependency graphs. We elaborate on this in the following subsection when we discuss the rule construction.

3. **Case where multiple relations are formed by coordinating conjunctions**

   This type of relationship represents all the sentences with at least one conjunction *and, but, or, yet, for, nor, so.* For reducing the complexity of the representation step, we only handle the case with the and-conjunction.

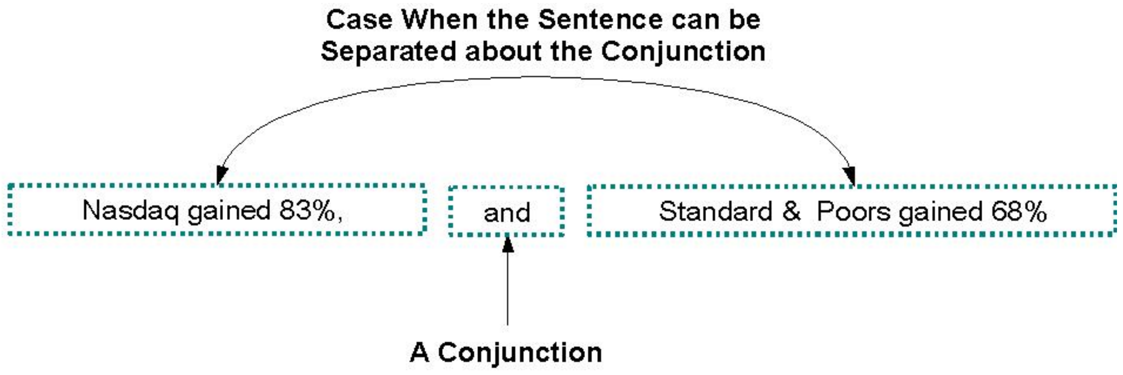

Case When the Sentence can be Separated about the Conjunction
Nasdaq gained 83%,
and
Standard & Poors gained 68%
A Conjunction

Figure 3.5 Example with a Simple Placement of Conjunction

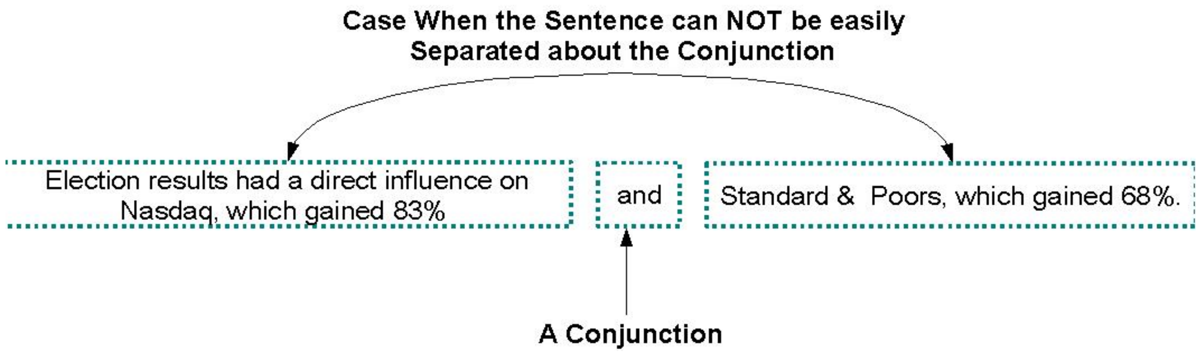

Case When the Sentence can NOT be easily Separated about the Conjunction
Election results had a direct influence on Nasdaq, which gained 83%
and
Standard & Poors, which gained 68%.
A Conjunction

Figure 3.6 Example with a Complex Placement of Conjunction

The simplest case of this relationship occurs when the conjunction separates two simple or complex clauses. In this case, the extraction task simply requires the separation of the sentence about the conjunction, followed by usual processes required for individual simple/complex clauses (refer Figure 3.5). However, in many other cases, the second clause is dependent on the subject or predicate of the first clause. This is illustrated in Figure 3.6. The latter case requires special handing with respect to the information extraction task. Further, these relationships generally contain two or more of the other relationship types described in this section and thus, we thought it to be relevant to include them in our set of complex relationships.

Complex relationship types 1 and 2 are often encountered in complex literature (such as Biomedicine) and an effective approach for such domains is to interpret the internal clause as a single complex object in a modified form. Although this may work for certain highly specialized domains, as we saw through the examples above, it can easily overlook some interesting information constructs that our approach would be able to extract. We make this more concrete in chapter 5, where we do some evaluations and report interesting results achieved from utilizing our approach.

Having clearly defined the type and nature of relations that we intend to extract, we now turn our attention to the rules that can identify and extract them. For ease of analysis and discussion, we logically group the rule formulation by complex and simple relationship types.

3.3.3 Formulating Rules for Complex Relationship and Entity Extraction

In 3.3.2.2, we observed that most of the complex relationships are characterized by implicit and explicit dependencies between parts of the sentence. Since such dependencies are nicely captured by the dependency graph (refer chapter 2 for an overview of dependency graph), we focus on it to formulate our rules. For this, we follow the methodology described below.

1. Refer to the dependencies described in section 2.1.2 and choose the ones that fit the structure for each type of complex relationship described in previous subsection.

2. Form conditions based on whether a dependency label (or a sequence of labels) is found along a set of edges in the dependency graph.

3. Form actions by deciding which nodes to extract as constructs.

4. Express premises and consequents to form the extraction rule.

We now apply these steps for each of the complex relationship types.

3.3.3.1 Rule Formulation for Case 1

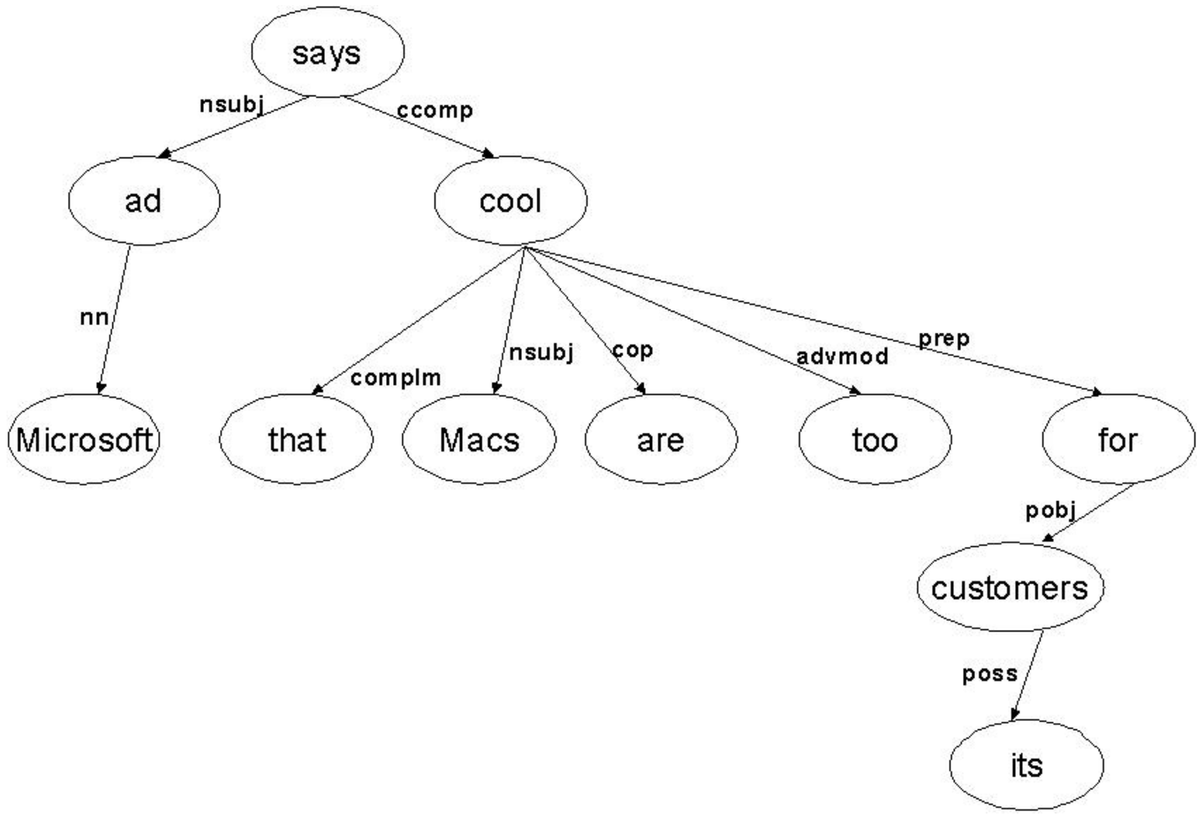


Figure 3.7 Dependency Graph for the Sentence in Figure 3.2

1. For this case, we base our rule on two main clausal dependencies from the overall set of Stanford dependency relations - *Clausal complement (ccomp) and Parataxis (parataxis)* (For a definition of these dependencies, refer Chapter 2, Section 2.1.2). Clausal complement and Parataxis capture the structure of the main relationship form as illustrated in Figure 3.2 (as well as form 1a) and form 1b of case 1 respectively.

2. For understanding the conditions based on which we would extract constructs, lets observe the dependency graph for the sentence in Figure 3.2, as illustrated in Figure 3.7. The main observation to be made is that *ccomp* nicely captures the clause-level dependency between the verbs of the sentence.

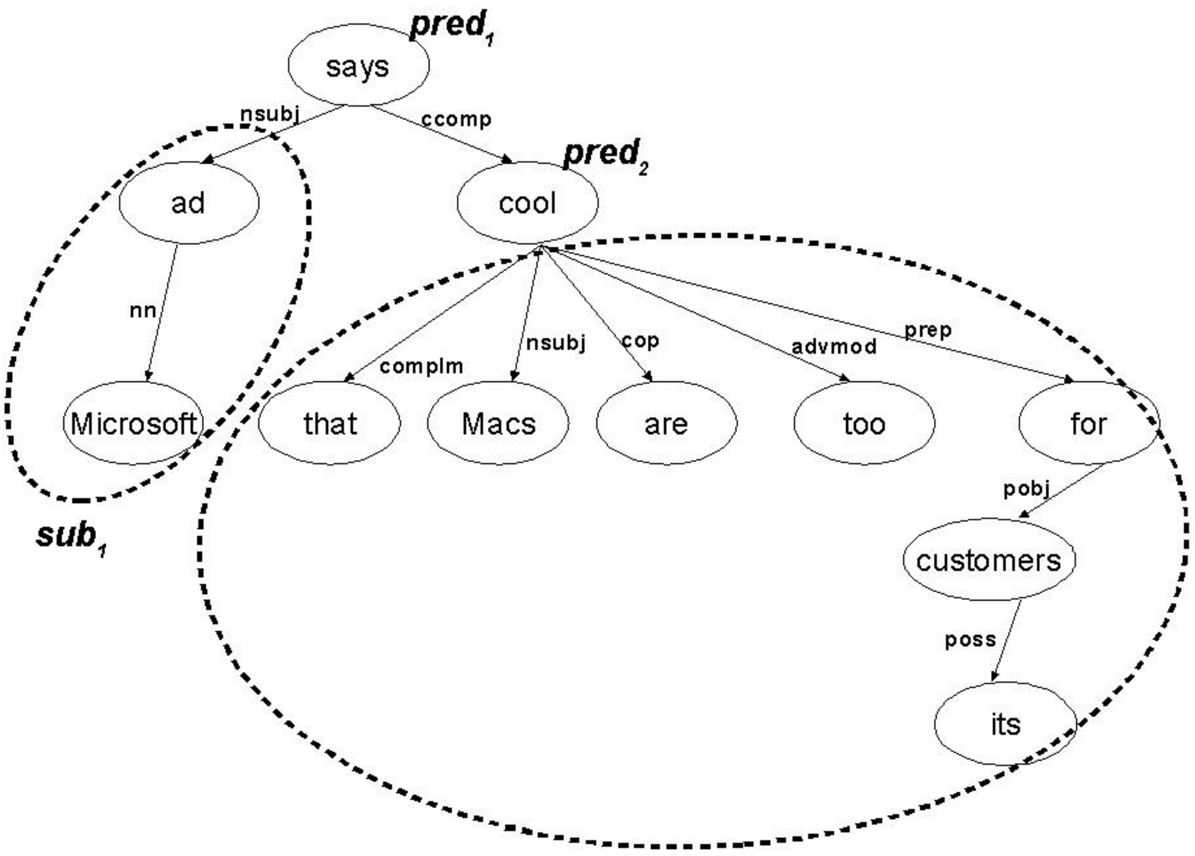


Figure 3.8 Extraction Rule Application for a Relationship with Internal Clauses

3. Further, we want to capture the information that the main subject (Microsoft ad) has a relation (says) with the composite object, which is an inner clause and thus, has its own subject (Macs), relation (coolness), and object (customers). In case of the main subject, we extract the modifiers as well as the main subject. This is easily achieved by looking for the noun compound modifier dependency (nn) or quantitative phrase modifier (quantmod) in our extraction rule.

4. Thus, we would like to extract the following constructs for this case:

   $pred_1$ = {Node with two outgoing edges with labels "nsubj" and "ccomp"}

   sub_1 = {$Node_1$=Node that is connected to the $pred_1$ node by an edge with label "nsubj", Node connected to $Node_1$ by an edge with label "nn" or "quantmod"}

In a similar manner, we would extract the constructs when *parataxis* dependency appears in the graph instead of *ccomp*. We do not illustrate this with an example since it generates very similar dependency graph to the one shown in Figure 3.7, other than the fact that *ccomp* is replaced by *parataxis*.

We leave out the constructs within the inner clause for now since it only comprises a simple relationship, which can be easily extracted using the rules for simple relationships described later in this section.

Expressing this formally using first-order logic notation, the general rule-set for this case is described as follows.

Extraction Rule 1: (*Extraction Rule for Relationships with Internal Clauses*) Given a dependency graph $G(V, E)$ with a label function l, for an English-language sentence T, the information extraction rule-set to identify and extract the complex relationship with internal clauses as described in case 1, is given as,

- $r_{RIC_1} : \{\exists u, v, w \in V, \exists e_1(u, v), e_2(v, w) \in E \,|\, l(e_1) = \text{“}nsubj'' \cap (l(e_2) = \text{“}ccomp'' \cup l(e_2) = \text{“}parataxis'')\} \longrightarrow \{pred_1 = \{v\}, sub_1 = \{u\}\}$

- $r_{RIC_2} : \{\exists u, v, w, t \in V, \exists e_1(u, v), e_2(v, w), e_3(u, t) \in E \,|\, l(e_1) = \text{“}nsubj'' \cap (l(e_2) = \text{“}ccomp'' \cup l(e_2) = \text{“}parataxis'') \cap l(e_3) \in \{\text{“}nn'', \text{“}quantmod''\}\} \longrightarrow \{sub_1 = sub_1 \cup \{t\}\}$

Result of application of this rule to the sentence in Figure 3.2 is shown in Figure 3.8

3.3.3.2 Rule Formulation for Case 2

1. For this case, we base our rule on two modifier dependencies - *Prepositional modifier (prep) and Adjectival modifier (amod)* (For a definition of these dependencies, refer Chapter 2, Section 2.1.2). Prepositional and Adjectival modifiers capture the modifying qualifier and value of qualification respectively.

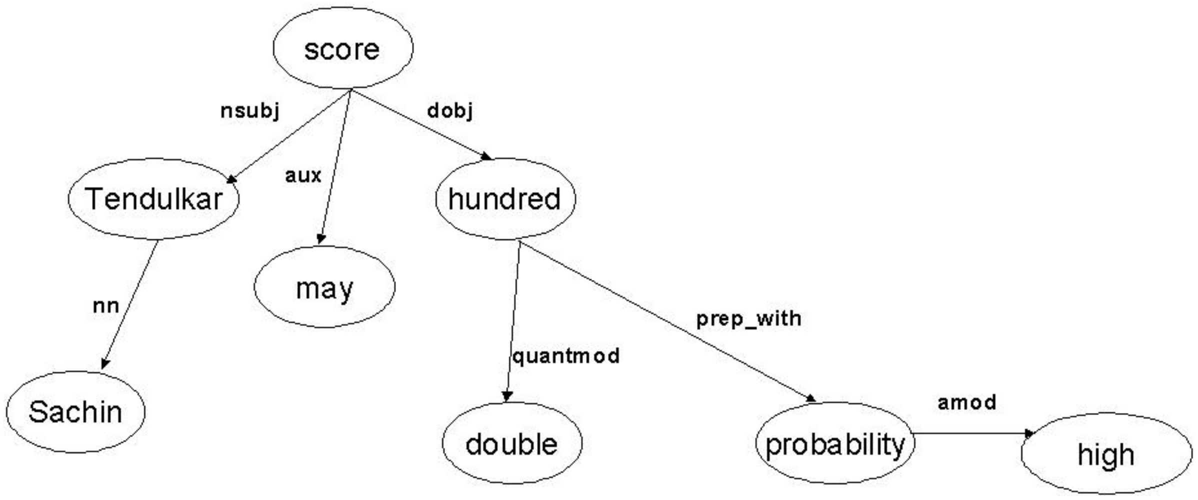


Figure 3.9 Dependency Graph for the Sentence in Figure 3.4

2. For understanding the conditions based on which we would extract the information constructs, we again observe the dependency graph for the sentence in Figure 3.4, as illustrated in Figure 3.9. The main observation to be made is that *prep* identifies a qualification associated with the main clause and *amod* identifies the value (in this case, the degree).

   Recall that we need to take care of subtle differences in dependency graphs with respect to verb variations. This variation is basically around the placement of edge labeled *prep*, viz. it may be connected to either of the three head-nodes in the main-clause (subject, predicate or the object). To account for this, we simply ignore the placement of *prep*. We observe that as long as we have identified a *prep–amod* pattern in the graph, we can get all the qualifying information that we need to be able to perform extraction of the qualified relationship.

3. Now, for reasons that will be clear when we get to the validation and enhancement steps, we intend to capture the following information - "There is a *qualified relationship* with the subject *Sachin Tendulkar*, relationship *may score*, object *double century*, and probability *high*". Similar to the previous case, we extract the modifiers as well as the

main subject/object by looking for any modifier dependency (nn, quantmod, etc) in our extraction rule.

4. In all, we would like to extract the following constructs for this case:

   $pred_1$ = {Node with two outgoing edges with labels "nsubj" and "dobj"}

   sub_1 = {$Node_1$=Node that is connected to the $pred_1$ node by an edge with label "nsubj", Node connected to $Node_1$ by an edge with label "nn" or "quantmod"}

   obj_1 = {$Node_1$=Node that is connected to the $pred_1$ node by an edge with label "dobj", Node connected to $Node_1$ by an edge with label "nn" or "quantmod"}

   $qual_1$ = {Node with two edges with labels "prep" and "amod"}

   val_1 = {Node that is connected to $qual_1$ by an edge with label "amod"}

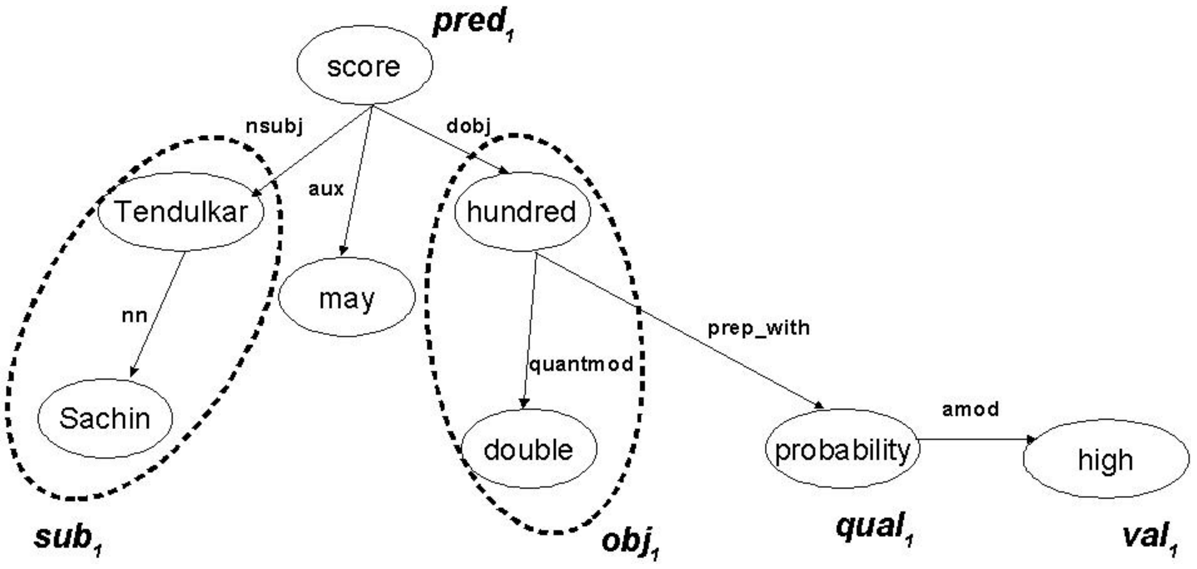


Figure 3.10 Extraction Rule Application for a Relationship with Qualifying Modifiers

<u>Extraction Rule 2:</u> (*Extraction Rule for Relationships with Qualifying Modifiers*) Given a dependency graph $G(V, E)$ with a label function l, for an English-language sentence T, the extraction rule-set to identify and extract the complex relationship with qualifying modifiers as described in case 2, is given as,

- $r_{RIC_1} : \{\exists u, v, w, x, y \in V, \exists e_1(u, v), e_2(v, w), e_3(w, x), e_4(x, y) \in E \,|\, l(e_1) = \text{“}nsubj''$ $\cap\, l(e_2) = \text{“}dobj'' \cap l(e_3) = \text{“}prep'' \cap l(e_4) = \text{“}amod''\} \longrightarrow \{pred_1 = \{v\},\, sub_1 =$ $\{u\},\, obj_1 = \{w\},\, qual_1 = \{x\},\, val_1 = \{y\}\}$

- $r_{RIC_2} : \{\exists u, v, w, x, y, t \in V, \exists e_1(u, v), e_2(v, w), e_3(w, x), e_4(x, y), e_5(u, t) \in E \,|\, l(e_1) =$ $\text{“}nsubj'' \cap l(e_2) = \text{“}dobj'' \cap l(e_3) = \text{“}prep'' \cap l(e_4) = \text{“}amod''$ $\cap\, l(e_5) \in \{\text{“}nn'', \text{“}quantmod''\}\} \longrightarrow \{sub_1 = sub_1 \cup \{t\}\}$

- $r_{RIC_3} : \{\exists u, v, w, x, y, t \in V, \exists e_1(u, v), e_2(v, w), e_3(w, x), e_4(x, y), e_5(w, t) \in E \,|\, l(e_1) =$ $\text{“}nsubj'' \cap l(e_2) = \text{“}dobj'' \cap l(e_3) = \text{“}prep'' \cap l(e_4) = \text{“}amod''$ $\cap\, l(e_5) \in \{\text{“}nn'', \text{“}quantmod''\}\} \longrightarrow \{obj_1 = obj_1 \cup \{t\}\}$

Result of application of this rule to the sentence in Figure 3.4 is shown in Figure 3.10

3.3.3.3 Rule Formulation for Case 3

In the case of relations with conjunctions, we do not formulate extraction rules based on dependency graphs. This is since conjunctions are words that connect parts of a sentence and thus, do not necessarily have an implicit or explicit dependency on any but their immediate successor and predecessor in the sentence. In such a scenario, use of a dependency graph can be avoided.

Instead, we take care of this as an auxiliary case in our extraction algorithm itself by analyzing the structure of the parse tree representation. We basically look at the parse of the part of the sentence to the right of the conjunction, and apply the the rules described below. Note that these are not similar to the *extraction rules* that we have been using since in this case, we are referring to the already extracted constructs from the left clause and utilizing them directly for extraction information from the right part. In this process, if we figure that the right part does not have any references from the left, we simply utilize our pre-existing rule-framework for extracting information from it as if it was a distinct sentence, and thus, we do not require any extraction rules for this case at all.

The rules we follow for analyzing conjunctions are as follows:

- If the parse tree for the right part contains a simple declarative clause (S), treat it as a distinct sentence and utilize the pre-existing rule-framework for extracting information.
- If the parse tree contains a verb phrase (VP) and noun phrase (NP), append the extracted subject of the left part to the right part and treat it as a distinct sentence and utilize the pre-existing rule-framework for extracting information.
- If the parse tree contains only a noun phrase (NP), append the extracted subject and the predicate of the left part to the right part and treat it as a distinct sentence and utilize the pre-existing rule-framework for extracting information.

3.3.4 Formulating Rules for Simple Relationships

In case of simple relationships, we extract the following constructs:

$pred_1$ = {Node with two outgoing edges with labels "nsubj" and "dobj"}

sub_1 = {$Node_1$=Node that is connected to the $pred_1$ node by an edge with label "nsubj", Node connected to $Node_1$ by an edge with label "nn" or "*mod"}

obj_1 = {$Node_1$=Node that is connected to the $pred_1$ node by an edge with label "dobj", Node connected to $Node_1$ by an edge with label "nn" or "*mod"}

Here, *"*mod"* is a shorthand notation, used only for this case, to denote any modifier type dependency.

Expressing these rules formally using first-order logic notation, the general rule-set for this case is described as follows.

Extraction Rule 3: (*Extraction Rule for Simple Relationships*) Given a dependency graph $G(V, E)$ with a label function l, for an English-language sentence T, the extraction rule-set to identify and extract the simple relationship as described in section 3.3.2.1, is given as,

Algorithm 1 Extracting Candidate Information Constructs

```
1: procedure ExtractConstructs($p, G, R, l$)
2: $rawConstructs$ = CALL ExecuteRules($G, R$)
3: if $flag_{clausal}$ is true then
4:    Break $p$ about $pred_1$ into $p_l$ with node-set $V_1$ and $p_r$ with $V_2$
5:    $G_S(V_2, E_2)$ = Subgraph of $G$ induced by $V_2$
6:    $innerConstructs$ = CALL ExtractConstructs($p_r, G_S, R, l$)
7:    for all $ele$ in $innerConstructs$ do
8:       Form an $outerConstruct$ using ($sub_1, pred_1, element$)
9:       Add $outerConstruct$ to the List of $outerConstructs$
10:   end for
11:   Cache first element from $outerConstructs$
12:   return $outerConstructs$
13: else if $flag_{conj}$ is true then
14:   Break $p$ about $conj_{and}$ into $p_f$ with node-set $V_1$ and $p_s$ with $V_2$
15:   $G_{S1}(V_1, E_2)$ = Subgraph of $G$ induced by $V_1$
16:   $G_{S2}(V_2, E_2)$ = Subgraph of $G$ induced by $V_2$
17:   $outerConstructs_1$ = CALL ExtractConstructs($p_f, G_{S1}, R, l$)
18:   Cache first element from $outerConstructs_1$
19:   if $p_s$ contains ‘S’ then
20:      $outerConstructs_2$ = CALL ExtractConstructs($p_s, G_{S2}, R, l$)
21:   else if $p_s$ contains ‘VP’ and ‘NP’ then
22:      Append $sub$ from the Cache to $p_s$
23:      $outerConstruct_2$ = CALL ExtractConstructs($p_s, G_{S2}, R, l$)
24:   else if $p_s$ contains ‘NP’ then
25:      Append $sub$ and $pred$ from the Cache to $p_s$
26:      $outerConstruct_2$ = CALL ExtractConstructs($p_s, G_{S2}, R, l$)
27:   end if
28:   Add $outerConstructs_1$, $outerConstructs_2$ and $outerConstruct_2$ to $outerConstructs$
29:   return $outerConstructs$
30: else if $flag_{enrich}$ is true then
31:   $outerConstructs$ = CALL PerformEnrichments($rawConstructs$, ‘qualify’, $l$)
32:   return $outerConstructs$
33: else
34:   Create a construct Using ($sub_1, pred_1, obj_1$) and Cache to resolve any pronouns
35:   Add the construct to $outerConstructs$
36:   return $outerConstructs$
37: end if
```

- r_{RIC_1} : $\{\exists u, v, w \in V, \exists e_1(u,v), e_2(v,w) \in E \,|\, l(e_1) = \text{``}nsubj'' \cap l(e_2) = \text{``}dobj''\} \longrightarrow \{pred_1 = \{v\},\, sub_1 = \{u\},\, obj_1 = \{w\}\}$

- r_{RIC_2} : $\{\exists u, v, w, t \in V, \exists e_1(u,v), e_2(v,w), e_3(u,t) \in E \,|\, l(e_1) = \text{``}nsubj'' \cap l(e_2) = \text{``}dobj'' \cap l(e_3) \in \{\text{``}nn'', \text{``}*mod''\}\} \longrightarrow \{sub_1 = sub_1 \cup \{t\}\}$

- r_{RIC_3} : $\{\exists u, v, w, t \in V, \exists e_1(u,v), e_2(v,w), e_3(w,t) \in E \,|\, l(e_1) = \text{``}nsubj'' \cap l(e_2) = \text{``}dobj'' \cap l(e_3) \in \{\text{``}nn'', \text{``}*mod''\}\} \longrightarrow \{obj_1 = obj_1 \cup \{t\}\}$

An example of the application of only r_{RIC_1} is shown in Figure 3.1.
For sake of simplicity in validation and representation framework, we do not extract and represent relationships with negation modifiers in this work. We simply perform a negation check by looking for the presence of *neg* dependency, and if found, we skip the sentence from further processing.

The rule framework described in this section is used by algorithm 1 when it calls ExecuteRules. Next, we discuss our validation framework.

3.4 Semantic Validation Framework

For this section, we split our discussion into two subsections. In the first one, we detail the steps taken to perform validation of the extracted information constructs using the domain ontology model. In the next section, we describe the special case where we handle enrichments for information constructs that were extracted from sentences comprising certain special relationship cases (such as, *qualified*).

3.4.1 Performing Validation against the Domain Ontology Model

Our basic approach (for advanced approach, refer 3.4.3) for validating candidate information constructs *(subject, predicate, object)* against a domain description captured in an ontology model, and a set of instances is as follows:

1. Find an instance match for the subject and the object. For determining these matches, we perform simple syntactic comparisons sequentially on the entire set of subject (similarly object) candidates starting with the extracted head sub_1 or obj_1.

2. If a match for subject and object is found, find a matching property for the *predicate* in the domain ontology model. For this, we perform syntactic comparisons on domain ontology relationships and the predicate.

3. If we are able to find these matches, we check if the class concepts to which the instances for subject and predicate are asserted lie respectively in domain and range of the property matched.

If all these conditions hold, we add the construct *(instance for subject, property, instance for object)* to the set of validated constructs. The above conditions are captured formally in the validation criteria specified below.

<u>Validation Rule:</u> For a text fragment Z consisting of sentences $\{T_i\}$ with word-sets $\{W_i\}$, a set T_{CTR} of candidate constructs extracted by an extraction algorithm, a domain description captured in an ontology $O(R, C)$, a set of instances Y, a function $h : Y \longrightarrow \mathcal{P}(C)$, and a mapping F from the set W to $R \cup Y$ that is able to type the words in the sentence to an instance in Y or a relationship in R (whenever such a mapping is intuitive based on the domain of discourse), the validation process must result in a set $\{K = \{s_i, p_i, o_i\}\}$ of 3-tuples s_i, p_i, o_i (validated constructs) s.t. the following holds:

$\{\exists y_{1i}, y_{2i} \in Y, \exists r_i \in R |\, (y_{1i}, r_i, y_{2i}) \in K \Longleftrightarrow (\exists w_{1i}, w_{2i}, w_{3i} \in W_i, c_{1i}, c_{2i} \in C\, |\{w_{1i}, w_{3i}, w_{2i}\} \in$ $T_{CTR} \cap (w_{1i}, y_{1i}) \in F \cap (w_{2i}, y_{2i}) \in F \cap (w_{3i}, r_i) \in F \cap c_{1i} \in h(y_{1i}) \cap c_{2i} \in h(y_{2i}) \cap c_{1i} \in$

$Domain(r_i) \cap c_{2i} \in Range(r_i)\}$

3.4.2 Discussion on Basic Validation

In our experiments, we found that condition in step 3 causes rejection of a lot of triples, even in cases where the extracted construct actually made sense as per the domain. The reason for this is an incomplete description of the domain as captured by the ontology. While in most cases, it is best to reject potential constructs based on this condition, there may be a need to behave opportunistically and have some flexibility in these criteria, especially in cases when we find a matching property, but do not find a match for the subject or the object. Similarly, there may be cases when we find matching instances for the subject and the object, but not the predicate. We consider these cases along with some other enrichments in the next subsection.

3.4.3 Performing Enrichments

In this section, we describe the process[2] that we undertake for handling the constructs extracted from relationships with qualifying modifiers as well as other enrichments that help improve the performance of our extraction algorithm.

1. In the case of qualifications, we create new definitions in the ontology model to account for the qualifying relationships. In this case, extraction algorithm 1 invokes the enrichment module before the validation is performed on the constructs.

2. We also handle a few selective cases where one of the matches in the basic validation approach described in 3.4.1 does not work. For these cases, the enrichment module is invoked after the validation process determines that an enrichment is required.

Thus, overall, the enrichment process needs to ensure that it (i) invokes the validation module to validate the constructs passed to it depending on the case it is handling (ii) enriches

[2]This part of the validation framework is yet to be included within our SEMANTIXS system (refer Chapter 4) at the time of writing this thesis

Algorithm 2 Performing Validation and Enrichments

```
1: procedure PerformValidation($construct, level$)
2: Perform basic validation in 3.4.1 and store result in $returnValue$
3: if $returnValue$ is not $null$ then
4:     return $returnValue$
5: end if
6: if $level$ is '2' or '3' then
7:     Perform the enrichments in 3.4.3.2 and store result in $returnValue$
8:     if $returnValue$ is not $null$ then
9:         return $returnValue$
10:    end if
11: end if
12: if $level$ is '3' then
13:    Special case
14: end if
15: return $null$
```

```
1: procedure PerformEnrichments($construct, action, level$)
2: if $level$ is '1' then
3:     return
4: end if
5: if $action$ is 'qualify' then
6:     Perform the steps given in 3.4.3.1
7: else if $action$ is 'generalize' then
8:     Perform the steps given in 1
9: else if $action$ is 'add' then
10:    Perform the Steps given in 2
11: end if
```

the ontology so that it is able to capture the qualified relationship construct, and (iii) returns a list of validated and enriched constructs.

We discuss each of these steps below for the two cases described above.

3.4.3.1 Handling Qualifications

Recall that the rule for this relationship type extracted the following constructs $\{sub$, $pred$, obj, $\{qual_i$, $val_i\}\}$ (refer section 4).

1. Validating the constructs being qualified

   (a) We utilize the validation rule given in the basic approach 3.4.1 to validate $\{sub$, $pred$, $obj\}$.

   (b) Further, we find a match for *val* within the set of instances. For this, we utilize syntactic comparisons as before.

   If a match for the *val* is found, we proceed to the next step of enrichment. In case, no match is found, we do not perform any enrichment.

2. Enriching the ontology

   For capturing the extra qualifications, we make the following additions to the domain ontology.

   (a) As a one-time enhancement, it creates a new concept, *QualifiedRelationship*. We also create three new relationships *hasQualifiedSubject, hasQualifiedPredicate, hasQualifiedObject*, all with the domain *QualifiedRelationship* and range as the most general concept in the given ontology hierarchy.

   (b) Further, each time the enrichment module is asked to perform enrichment for qualifications, it creates a new instance (say *QualifiedRelationship_1*) of type *QualifiedRelationship*, creates a new property named *qualifier* with domain *QualifiedRelationship* and range as the concept of which the instance (that we found for *value*) is asserted. It creates such a property for each {*qualifier, value*} pair.

3. Returning enriched constructs

   Finally, it returns the following constructs in {*subject, predicate, object*} notation.

   (a) {*QualifiedRelationship_1, hasQualifiedSubject, sub*}, {*QualifiedRelationship_1, hasQualifiedpPredicate, pred*}, {*QualifiedRelationship_1, hasQualifiedObject, obj*}.

   (b) Further, for each {*qualifier, value*} pair, it returns the triple, {*QualifiedRelationship_1, qualifier, value*}

3.4.3.2 Handling Mismatches in Validation

We try to address two cases of mismatches in the basic validation process.

1. **Mismatch resolved by property generalization**

   This type of mismatch is caused when the validation framework finds a match for all the three constructs, however the domain/range check in 3 fails.

   (a) Enriching the ontology

   We handle this case by creating a generalization of the matched property with extended domain and range containing the concepts for matched subject and object respectively. The naming scheme used is - *matched property name_Generalized.*

   (b) Returning enriched constructs

   With the above addition, we can now capture the information constructs in our model and thus, we return the tuple {*subject, matched property name*_Generalized, *object*}

2. **Mismatch resolved by property addition**

   This type of mismatch is caused when the validation framework finds find a match for the subject and object, however the predicate match in 2 fails.

   (a) Enriching the ontology

We handle this case by creating a new property with domain and range containing the concepts for the matched subject and object respectively. The naming scheme used is - *matched property name_Created.*

(b) Returning enriched constructs

With the above addition, we can now capture the information constructs in our model and thus, we return the tuple {*subject, matched property name_*Created*, object*}

This validation framework is used by algorithm 2.

Algorithm 3 Representing Validated/Enriched Information Constructs

```
1: procedure RecursiveValidateAndRepresent($construct, level$)
2: if $obj$ from $construct$ is a list then
3:    $validatedConstruct$ = CALL RecursiveValidateAndRepresent($obj, level$)
4:    if $validatedConstruct$ is not $null$ then
5:       Reify $validatedConstruct$ as a RDF Statement
6:       Create RDF triple using $sub$, $pred$ and the Reified Statement
7:    end if
8: else
9:    $validatedConstruct$ = CALL PerformValidation($construct, level$)
10:   if $validatedConstruct$ is not $null$ then
11:      Create an RDF $triple$ using $validatedConstruct$
12:      return $triple$
13:   end if
14: end if
```

3.5 Representation Framework

In this section, we describe the process to represent the extracted and validated constructs in Resource Description Framework.

3.5.1 Transformation of Information Constructs into Graph Formalism

For representation purposes, we seek a set of transformations from the set of extracted and validated information constructs, $\{K = \{s_i, p_i, o_i\}\}$ (determined as per our extraction, r_{RIC_i} and validation (refer 1) rules described in previous sections) to an RDF graph G_{RDF}. We provide a description of the actual transformation process that we employed in our framework, followed by formalization of the notion.

3.5.2 Primitive Transformation for Representing Simple Information

Simple information is extracted from sentences comprising simple relationships, with or without the enrichment that handles mismatches in validation. Since enrichment that handles mismatches in validation, replaces the extracted predicate by a new one (refer section 3.4.3.2), it does not make a difference to the representation of the information constructs.

We represent simple information using RDF triple notation. An example of this is shown in Figure 3.11 (refer appendix A for serialization in xml format). This figure shows the RDF representation of the information extracted from the sentence in the example discussed in section 2.2.1, using a suitable domain ontology with instances.

We now formalize this transformation.

Primitive Transformation: For a set $\{K = \{s_i, p_i, o_i\}\}$ of 3-tuples s_i, p_i, o_i (validated information constructs), a primitive transformation is defined on the elements of
$K_{simple} = \{\{s_i, p_i, o_i\} | \{s_i, p_i, o_i\} \in K \cap |o_i| = 1\}$ as follows:

$TransformPrimitive(\{s_i, p_i, o_i\}) \longrightarrow G_{RDF}$, which creates a graph G_{RDF} consisting of node-set $\{s_i, o_i\}$ and edge-set $\{p_i\}$ such that the only edge connects the two nodes.

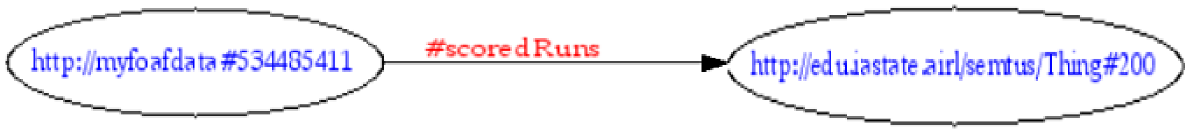

http://myfoafdata#534485411
#scoredRuns
http://edu.iastate.airl/semtus/Thing#200

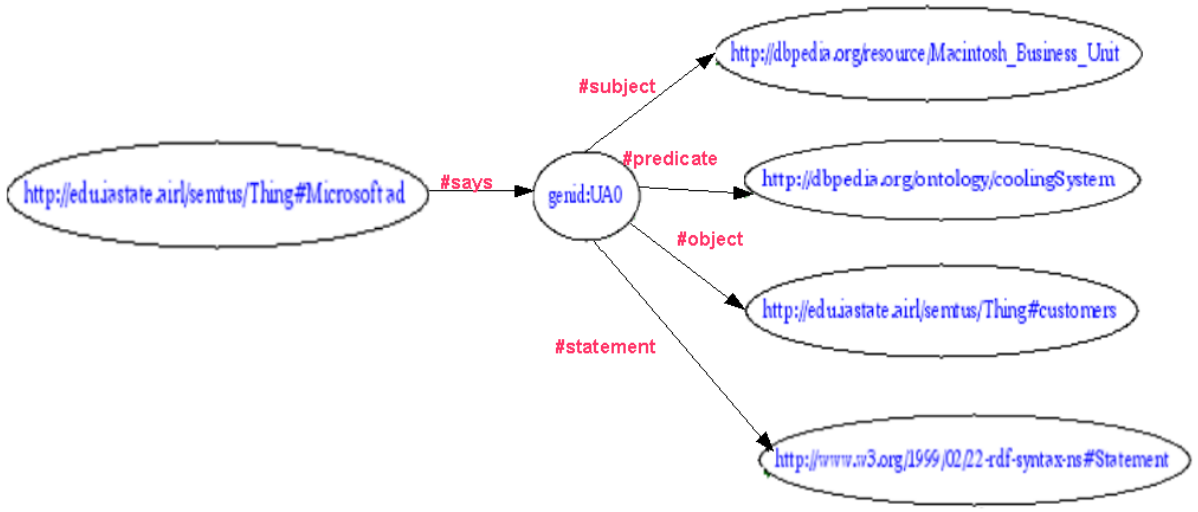
Figure 3.11 RDF representation of Simple Information

3.5.3 Transformations for Representing Complex Information

Structurally complex information is extracted from sentences comprising complex relationships (with or without enrichments that handle mismatches in validation). In this case, we also cover the enrichments that handle qualifications since they comprise of complex relationship.

We discuss these cases individually below.

1. **Representation of information extracted from relationships with internal clauses (refer 1)**



http://dbpedia.org/resource/Macintosh_Business_Unit
#subject
#predicate
http://dbpedia.org/ontology/coolingSystem
http://edu.iastate.airl/semtus/Thing#Microsoft ad
#says
genid:UA0
#object
http://edu.iastate.airl/semtus/Thing#customers
#statement
http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement

Figure 3.12 RDF representation of Reified Internal Clauses

Recall from 3.3.3.1 that the information constructs extracted from these relationship types are: $\{pred_1, sub_1\}$. Further, there is an internal clause that is dependent on $pred_1$. For purposes of explaining our representation, we can assume (without loss of generality) that the internal clause does *not* consist of a complex relationship form other than a qualified relationship. In other words, it consists of a simple relationship, a qualified relationship, or some other relationship form that we did not cover in our framework. This assumption holds because we can recursively utilize our extraction rule-framework until we get to a clause with one of those relationship types. We use this idea in Algorithm 1.

Thus, we need to represent a subject associated with another fact (extracted from the inner clause), which, in turn, may be associated with some other fact and so on. We model this dependence using *reified* statements in RDF as done in Algorithm 3. We reify the internal fact as a statement in RDF and then create a triple using the subject and the predicate with this statement as the object.

Refer figure 3.12 for an example (refer A for serialization in xml format). This figure shows the RDF representation of the information extracted from the sentence in Fig 3.8.

We now formalize this transformation.

<u>**Composite Transformation:**</u> For a set $\{K = \{s_i, p_i, o_i\}\}$ of 3-tuples s_i, p_i, o_i (validated information constructs), a composite transformation is defined on the elements of $K_{complex} = \{\{s_i, p_i, o_i\} | \{s_i, p_i, o_i\} \in K \cap |o_i| > 1\}$ as follows:

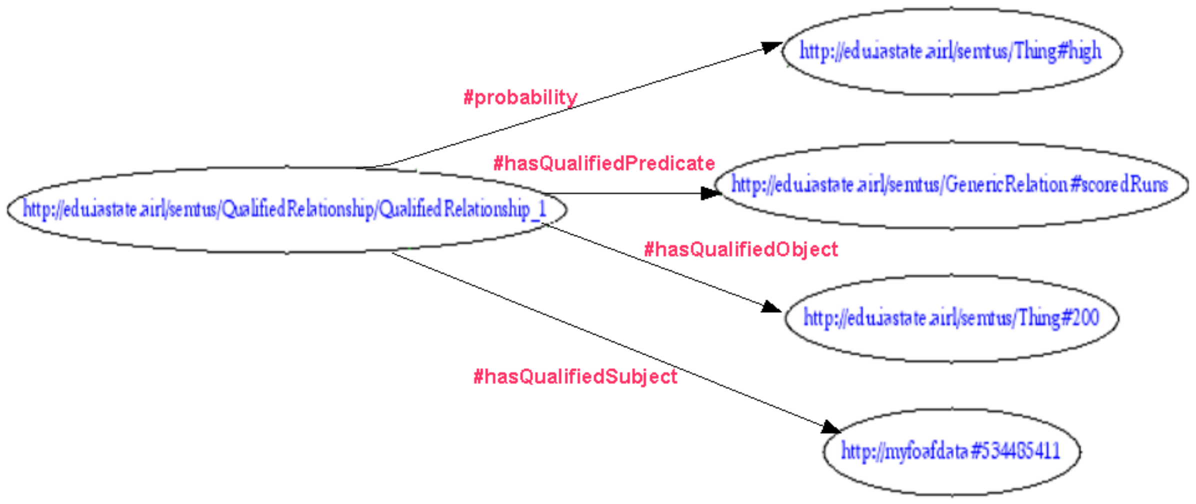
$TransformComposite(\{s_i, p_i, o_i\}) \longrightarrow TransformPrimitive(\{s_i, p_i, t\})$
$\cap TransformPrimitive(\{t, obj, o_{o_i}\}) \cap TransformPrimitive(\{t, pred, p_{o_i}\})$
$\cap TransformPrimitive(\{t, sub, s_{o_i}\}) \cap TransformPrimitive(\{t, stmt, id\})$, where t is a special node, called *blank node*, *sub*, *obj* and *pred* are special edges, p_{o_i}, s_{o_i} and o_{o_i} are internal elements of o_i.

This composite transformation triggers five primitive transformations creating subgraphs for each.

2. **Representation of information extracted from relationships with qualifications (refer 2)**

   Recall from 3.3.3.2 that the information constructs extracted from these relationship types are: $\{$*pred*, *sub*, *obj*, *qual*, *val*$\}$. Recall also from 3 that the enrichment module returns the following enriched constructs:

   - $\{$*QualifiedRelationship_1*, *hasQualifiedSubject*, *sub*$\}$, $\{$*QualifiedRelationship_1*,



http://edu.iastate.airl/semtus/QualifiedRelationship/QualifiedRelationship_1
#probability
http://edu.iastate.airl/semtus/Thing#high
#hasQualifiedPredicate
http://edu.iastate.airl/semtus/GenericRelation#scoredRuns
#hasQualifiedObject
http://edu.iastate.airl/semtus/Thing#200
#hasQualifiedSubject
http://myfoafdata#534485411

Figure 3.13 RDF representation of Qualified Relationships

hasQualifiedpPredicate, pred}, {*QualifiedRelationship_1, hasQualifiedObject, obj*}.

- Further, for each {*qual, val*} pair, it returns {*QualifiedRelationship_1, qual, val*}

Since all of these are simple information pieces, we can easily represent them using the primitive transformations described in 3.5.2.

An example of this is shown in Figure 3.13 (refer A for serialization in xml format). This figure shows the RDF representation of the information extracted from the sentence in Fig 3.10.

This representation framework is used by algorithm 3.

3.5.4 Existential Claims based on our Information Extraction Framework

We now make the following claims and justify them against the analysis done so far.

- **Claim 1**: The resulting graphs from *TransformPrimitive* and *TransformComposite* are valid RDF fragments.

  The correctness of this claim follows from the definitions of *TransformPrimitive* and *TransformComposite* and their one-to-one correlation with triples and reified statements in RDF.

- **Claim 2**: There always exists a transformation from a natural language sentence containing at least one of the relationship types identified by us, to a graph formalism such that the underlying information expressed in the relationship is captured in a query-able form in the graph.

  Proof Sketch: For proving the correctness of this claim, we show the existence of such a transformation for an arbitrary natural language sentence. Consider a sentence T with word-set $\{W_i\}$. Assuming the sentence falls within the scope of the English language grammar[3], there exists a transformation from $\{W_i\}$ to (p, G), where p is a parse tree and G is a dependency graph. Using (p, G), we can use ExtractConstructs (refer 1) to get a set of extracted information constructs, T_{CTR}. Using this set and applying the validation rule (refer 1 and algorithm 2), we can get a set of extracted and validated information constructs, K. Once we get this set, we can use *TransformPrimitive* and *TransformComposite* to get a set of graphs and from validity of claim 1, it then follows that those graphs are valid RDF fragments, which can be queried using query-languages for RDF (like SPARQL [45]).

3.6 Discussion on Algorithms

In this section, we briefly describe our algorithms that make use of the frameworks described above to extract information from domain-specific text.

The main algorithm 4 controls the overall information extraction task by invoking the extraction, validation and representation procedures. It takes a document as input, tokenizes it into individual sentences, generates the derivative structures for analysis and invokes the entity and relationship extraction module 1. It also takes an argument *level* as input to give its caller a better control of the overall information extraction process. Based on the *level*, it optionally turns off the the enhancement module and behaves opportunistically in terms of validation. We elaborate on this in the next Chapter when we discuss implementation details

[3]In this work, we always assume that the text is from the English language; more specifically, all that is captured by the englishPCFG.ser provided by with the Stanford Parser package

Algorithm 4 Information Extraction from a Text Source

```
1: procedure ExtractInformation($D, level$)
2: Tokenize document about sentence boundaries to get a Set, $Sentences$
3: for all $T$ in $Sentences$ do
4:    $(p, G)$ = Get Parse Tree and Dependency Graphs from $T$
5:    $outerConstructs$ = CALL ExtractConstructs($p, G, R, level$)
6:    for all $construct$ in $outerConstructs$ do
7:       CALL RecursiveValidateAndRepresent($construct, level$)
8:    end for
9: end for
```

of *SEMANTIXS*.

The extraction module 1 uses our rule framework to analyze the sentence for potential information constructs. It also invokes the procedure to enrich 2 if recommended by the rule-execution. Finally, it returns the list of candidate information constructs (*outerConstructs* in 4). These constructs are then passed on to the composite validation and representation module 3 that recursively analyzes the structure of the information construct to invoke the validation/enrichment module and decide the representation method to use. The result of this process is an RDF graphs (or a set of graphs depending on the size and nature of the document) of the form illustrated in Figure 3.11, 3.12 and 3.13.

3.6.1 Pronoun Resolution in Algorithm 1

The extraction procedure ExtractConstructs is formulated in a way such that it is able to perform simple cases of pronoun resolution (refer line 33 for the candidate information constructs extracted by the rule framework. This extra functionality greatly improves the recall of the overall algorithm on generic texts, which usually contain a lot of pronoun references. This will become more clear when we present our results in Chapter 5. We will see that this feature, together with our rule framework for complex dependencies, enable our algorithms to resolve and extract fairly complicated cross-sentential dependency situations.

CHAPTER 4. SEMANTIXS: SYSTEM ARCHITECTURE AND OVERVIEW

In this chapter, we describe the technical details of SEMANTIXS (System for Extraction of doMAin-specific iNformation from Text Including compleX Structures)[1], which has been implemented based on the analysis and algorithms presented in the previous chapter. In the first section, we describe the architectural overview of SEMANTIXS, followed by the design and implementation details in the next section.

4.1 SEMANTIXS Architectural Overview

SEMANTIXS is a system for ontology-guided extraction and semantic representation of structured information from unstructured text, implemented as a web application, that can extract, represent and visualize domain-specific information from free-text in the form of complex (and simple) relationships. It does this by applying rule-based extraction algorithms to free-text and identifying key entities and relations, referring to certain background knowledge that is given as input, validating the extracted constructs against this background knowledge in order to achieve coherent results as per the given domain, and finally, representing the results in the form of RDF graph(s), which can be analyzed using an in-built visualizer.

Besides incorporating our novel framework for extracting, validating and representing complex relationships, there are several salient features that make SEMANTIXS unique. Some of these are described below.

- SEMANTIXS provides multiple operating modes out-of-the-box, based on the desired

[1]An open-source, ready-to-deploy implementation of the system as a web application as well as the corresponding source code is available at http://sourceforge.net/projects/semantixs/

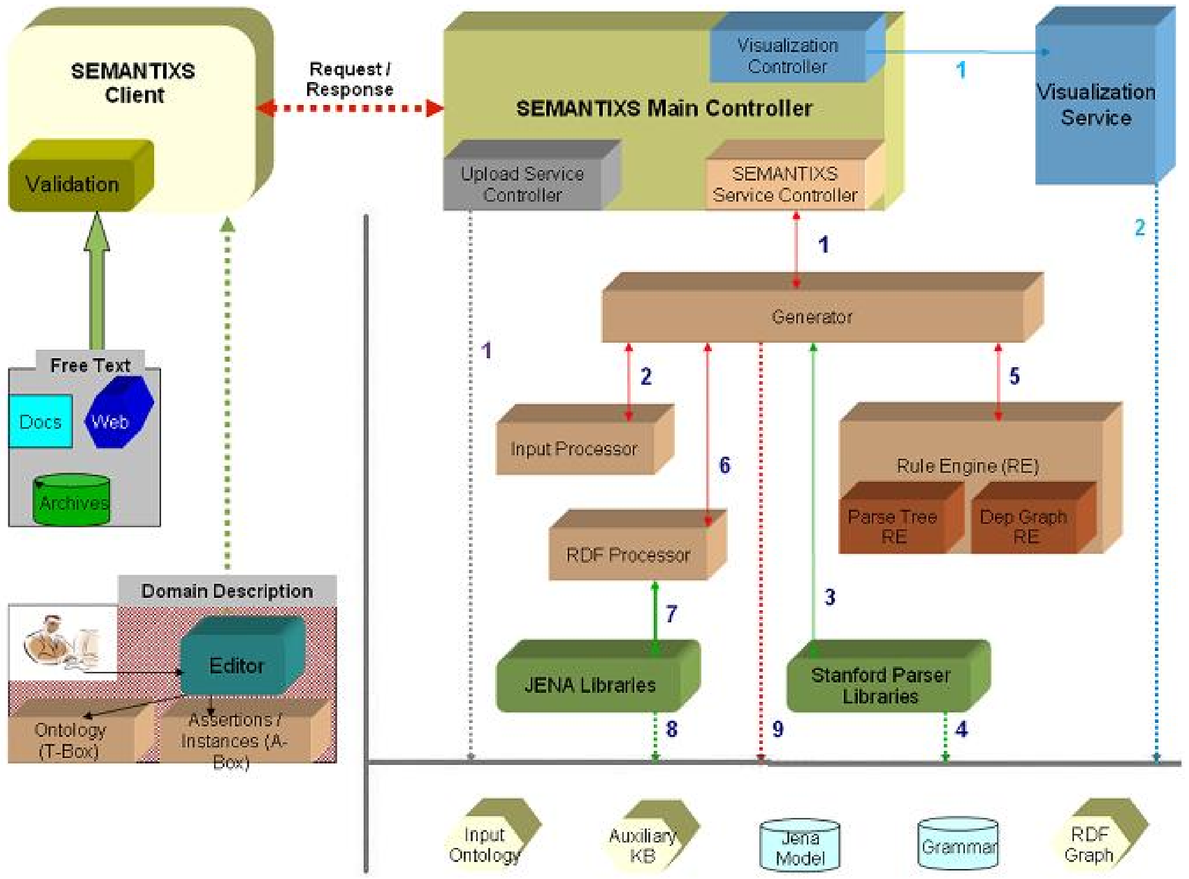


Figure 4.1 SEMANTIXS Architectural Diagram

complexity of analysis and exhaustiveness of domain description.

Since our original intent has been to create a ontology-guided information extraction system, in the default setting, SEMANTIXS expects a file in .rdf or .owl format as input, which is assumed to define the domain of discourse completely. In most cases, this would mean that the file is expected to contain the domain ontology (class concepts + relationships) as well as instance assertions that one expects in text. With this assumption, SEMANTIXS works within the given domain description and it does not try to extract any information that is not relevant to the description as defined in the uploaded file. This assumption works fairly well in case of expert users with a detailed knowledge of their domains or for well-defined domains that have a fairly exhaustive ontology (such as

UMLS [38]) and glossary of pre-defined terms or instances (such as MESH [39]). In this default setting SEMANTIXS works as per algorithm 4 without utilizing the module for enrichments, defined in 2.

However, it would be often unreasonable to assume such exhaustive domain descriptions in case of a naive user, given the inherent difficulty in modeling a domain correctly. Thus, we decided to slightly abuse our concrete definition of information extraction from text (refer 2.4) and provide some extra flexibility as follows.

For the partially-specified domain description option, SEMANTIXS works as per algorithm 4 utilizing the module for performing enrichments as defined in 2. For the option when there is no domain description assumption, SEMANTIXS works as per algorithm 4, again utilizing the module for performing enrichments 2. However, this time, it behaves opportunistically by creating new relations and concepts from unknown constructs.

- SEMANTIXS expresses the extracted information constructs in the form of a set of graphs.

  Since the output of SEMANTIXS always conforms to the W3C guidelines for RDF, it automatically becomes a graph specification that can be consumed by any algorithm that expects a set of graphs as an input. This makes SEMANTIXS an invaluable platform for numerous research query-based investigations on semantic graphs such as knowledge discovery, question answering, opinion mining, etc. We elaborate on some of these in chapter 6.

- SEMANTIXS provides extensive functionality to visualize and analyze the extracted RDF graphs.

  SEMANTIXS provides a reference to visualize the extracted semantic graph(s) using W3C's open visualization service. In addition, SEMANTIXS also provides an extensive analysis tool that can be used to visualize parts of the graph(s). This allows a user to analyze subgraphs associated with certain key entity or relationship, while ignoring the rest. Such a feature becomes invaluable when the resulting semantic graph is very large in size.

Figure 4.1 shows the architectural diagram of SEMANTIXS, which can be divided into three key components: (i) service request/response framework to handle service requests from the client and return the appropriate response (handled by the SEMANTIXS Main Controller), (ii) the core text processing and information extraction framework (handled by the SEMANTIXS Service Controller) (iii) the visualization and analysis framework including part of the UI (handled by the SEMANTIXS Visualization Controller). We describe the design and implementation-level details of these frameworks in the next section. In the remaining part of this section, we go through the high-level architecture and explain the interaction between components.

4.1.1 SEMANTIXS Component Interaction

An end user interacts with SEMANTIXS client by uploading an ontology, setting the desired *Analysis Complexity Level* and using a text fragment for processing. In all, within SEMANTIXS environment, the user can (i) upload the ontology file to the server, (ii) view the response in terms of the extracted information, (iii) visualize the response in the form of RDF graphs, and (iv) Analyze the response in the form of RDF subgraphs. We discuss each of these component interactions separately below.

- **Upload interaction**:

    1. When the user tries to upload the ontology file, the SEMANTIXS client immediately sends an HTTP request to the SEMANTIXS server. This request is intercepted by the servlet container, which redirects it to the appropriate controller (in this case, upload service). The controller performs the file upload and sends a response back to the server depending on whether the operation was a success or failure.
    2. This ontology stays on the server until the user exits the workspace. Upon exit, the SEMANTIXS client sends another request to the server to delete the file, which is

again handled in the same way as the previous request.

- **Viewing the response in terms of the extracted information**:

  1. When the user sends a text fragment for processing SEMANTIXS client application invokes the validation module to certify that the text is good enough for further processing. This comprises basic validation on the size of text as well as the character encoding. Once done, the client sends a request to the server with the text and the *Analysis Complexity Level*.
  2. This request is again intercepted by the servlet container within which the SEMANTIXS application resides and the control is delegated to the SEMANTIXS service controller.
  3. The service controller delegates the request to the generator, which is the module implementing algorithm 4. It invokes the input processor to process the input text, calls Stanford parsing libraries to generate parse trees and dependency graphs, and passes those structures to the rule-engine.
  4. The rule-engine has two separate modules that handle the dependency graph and parse tree respectively. The rule engine implements algorithm 1 to extract the information constructs for a particular sentence and passes these back to the generator.
  5. Next, the generator invokes the rdf processor which handles the validation and representation part as per the algorithms in 2. The processor interacts with the Jena libraries and generates a new RDF model based on the extracted information constructs.
  6. Finally, the generator serialized the RDF model in XML format and persists it on the file system. It also returns the serialization back to the user through an HTTP response.

- **Visualizing the response in the form of RDF graphs**:

1. When the user selects this option from the UI menu, the SEMANTIXS client sends a request to an open service from W3C. This service allows anyone to visualize a well-formed RDF document as a graph. Since the RDF generated by SEMANTIXS conforms to the W3C specifications, a user can directly utilize the service to visualize the complete RDF graph(s). The client opens up a new window with the service and instructs the user to paste the generated semantic metadata into appropriate textbox within the service page. Following this the user can select the option to visualize the graphs.

- **Analyzing the response in the form of RDF sub-graphs**:

    1. When the user selects this option from the UI menu, the SEMANTIXS client sends another request to the server. This time it request the visualization service and thus, the servlet container delegates the request to the visualization controller. This controller creates a visualization of RDF graph that is persisted on the filesystem in the previous step 4.1.1.
    2. With different interactions from the user on the UI, the visualization controller responds with the appropriate response that user sees back on the UI.

4.2 Design and Implementation Details

SEMANTIXS is developed as a Java-based web application deployed in a Apache Tomcat[2] server. This ensures ease of portability in different environments. It incorporates a number of open-source toolkits and services to provide a distinctive experience to the user. Its semantic processing capability comes from Jena Semantic Web Toolkit[27], parsing capability from Stanford Parser[26], visualization capability from a SVG visualizer from HP Labs and it utilizes Google Web Toolkit(GWT)[3] to provide an impressive look-and-feel to the user-interface.

[2]http://tomcat.apache.org/
[3]http://code.google.com/webtoolkit/

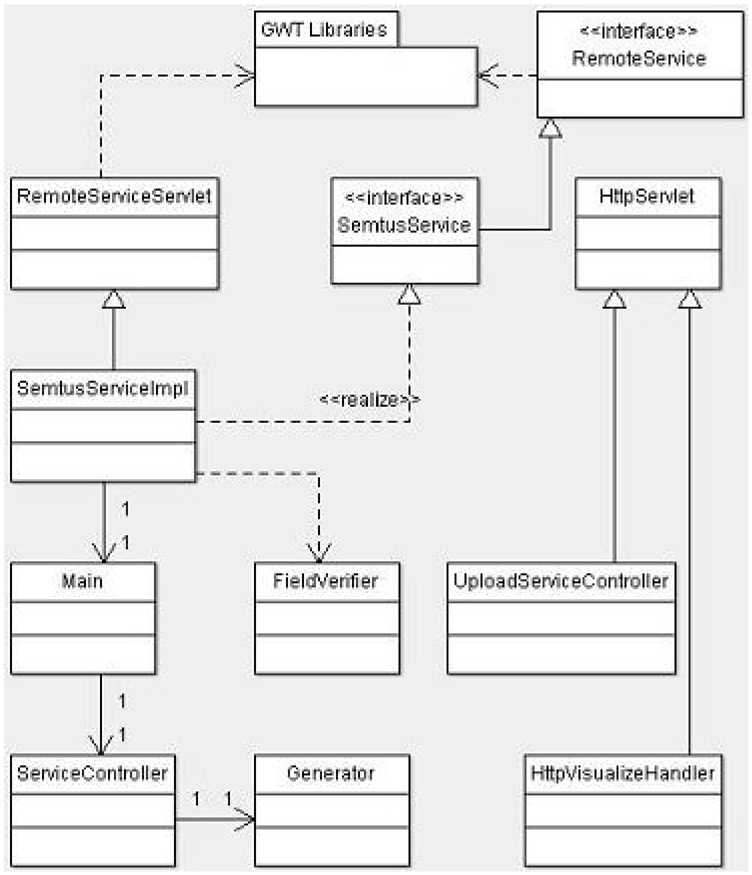


Figure 4.2 UML Representation of Service Request Delegation Framework

In this section, we delve deeper into the implementation of SEMANTIXS and describe various aspects of the design and implementation approaches employed during its development.

4.2.1 Service Request/Response Framework

The Request-response framework comprises of the following key classes:

- *SemtusServiceImpl* that implements the *SemtusService* interface. This is the class that handles the service requests from the client. It has dependencies on a few GWT framework libraries since the client UI was developed using GWT.

- *Main*, *ServiceController* and *Generator*, which handle the *chain-of-responsibility* in the delegation process emerging from SemtusServiceImpl.

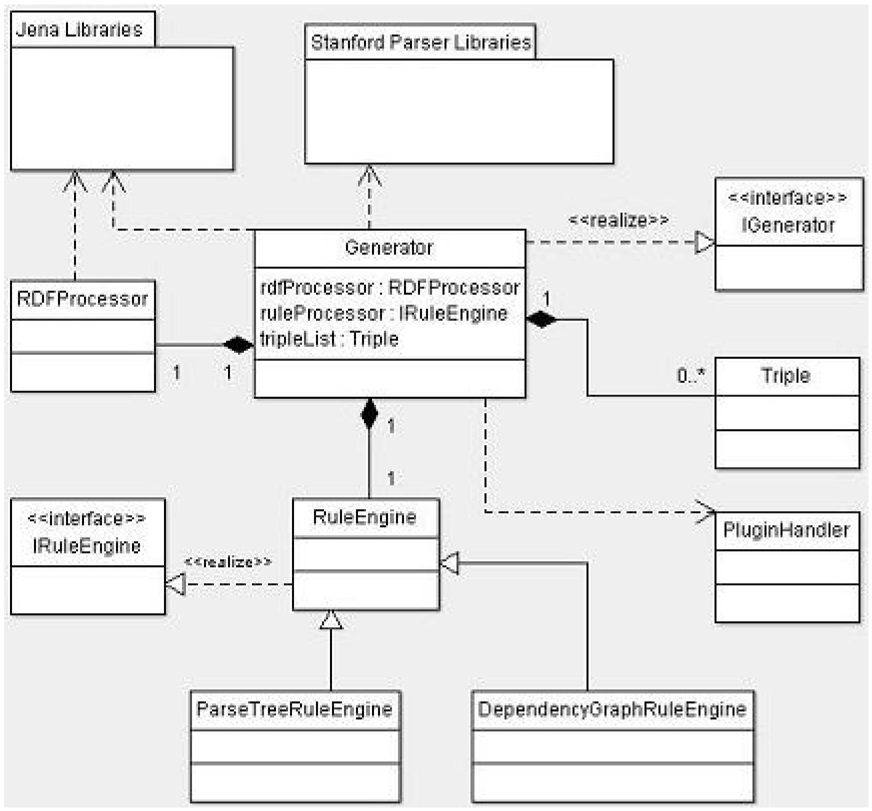


Figure 4.3 UML Representation of Text Processing and Information Extraction Framework

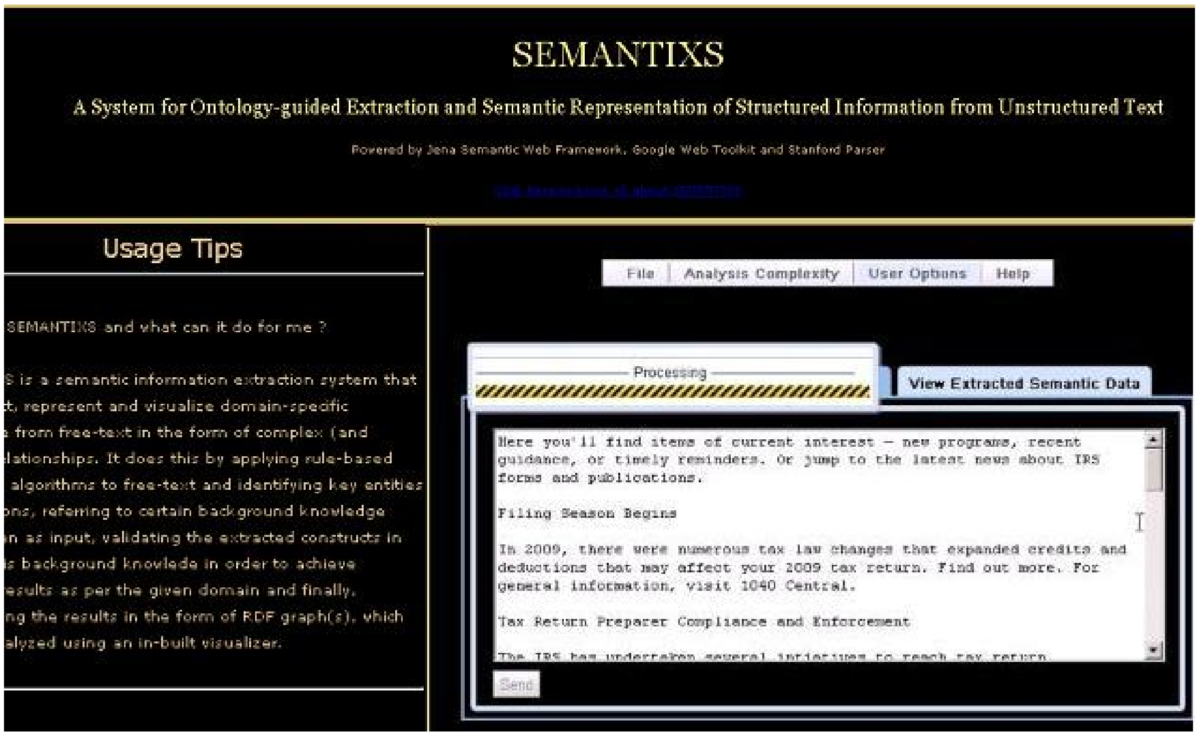
- *UploadServiceController* and *HttpVisualizeHandler*, which are the servlets handling the upload and visualization capabilities respectively.

The interaction between these classes is captured in the UML class diagram in Figure 4.2.

4.2.2 Core Text Processing and Information Extraction Framework

This framework is responsible for the bulk of the work in SEMANTIXS. It comprises of the following classes:

- *RuleEngine* that generalizes the specialized functionalities provided by the *ParseTree RuleEngine* and *DependencyGraphRuleEngine* classes, and conforms to the contract specified by the *IRuleEngine* interface.



SEMANTIXS

A System for Ontology-guided Extraction and Semantic Representation of Structured Information from Unstructured Text

Powered by Jena Semantic Web Framework, Google Web Toolkit and Stanford Parser

Usage Tips

SEMANTIXS and what can it do for me ?

S is a semantic information extraction system that
t, represent and visualize domain-specific
from free-text in the form of complex (and
lationships. It does this by applying rule-based
algorithms to free-text and identifying key entities
ons, referring to certain background knowledge
n as input, validating the extracted constructs in
s background knowlede in order to achieve
esults as per the given domain and finally,
ng the results in the form of RDF graph(s), which
alyzed using an in-built visualizer.

```
Here you'll find items of current interest — new programs, recent
guidance, or timely reminders. Or jump to the latest news about IRS
forms and publications.

Filing Season Begins

In 2009, there were numerous tax law changes that expanded credits and
deductions that may affect your 2009 tax return. Find out more. For
general information, visit 1040 Central.

Tax Return Preparer Compliance and Enforcement
```

Figure 4.4 Text Processing View

- *RDFProcessor* that utilizes the *Jena Libraries* to validate and represent the extracted information constructs.

- *Triple* that is a class representing an extracted information construct with a *subject, predicate, object*.

- *Generator* that has a composition relation with *RuleEngine*, *RDFProcessor* and *Triple*, and brings together the functionality offered by all these classes to implement the algorithm 4.

The interaction between these classes is captured in the UML class diagram in Figure 4.3.

4.2.3 User Interface, Visualization and Analysis Framework

The UI is created using the rich GWT libraries. GWT allowed us to write all our code in Java, while it took care of conversion of that code to UI widgets, Javascript, HTML, etc. The

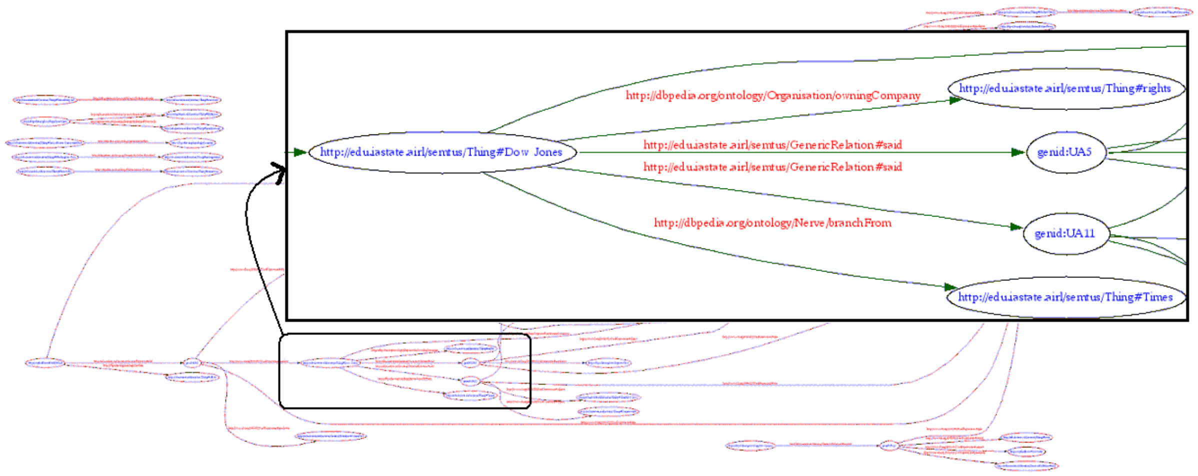

http://edu.iastate.airl/semtus/Thing#Dow_Jones
http://dbpedia.org/ontology/Organisation/owningCompany
http://edu.iastate.airl/semtus/Thing#rights
http://edu.iastate.airl/semtus/GenericRelation#said
http://edu.iastate.airl/semtus/GenericRelation#said
genid:UA5
http://dbpedia.org/ontology/Nerve/branchFrom
genid:UA11
http://edu.iastate.airl/semtus/Thing#Times

Figure 4.5 Semantic Graph View

UI comprises of the following main parts:

- **Workspace**: A snapshot of SEMANTIXS workspace while processing the text is shown in Figure 4.4. It consists of a tabbed-view pane containing text-areas for text input and displaying the response. The usage is quite intuitive.

- **Menu Options**: SEMANTIXS UI provides options for the selection of *Analysis Complexity Levels*, based on which it requests the information extraction operation. There are three different complexity levels based on whether the user has (i) an exhaustive ontology description and does not want to sacrifice correctness at any cost (level 1), (ii) partially-specified ontology and wants to have enrichments, whenever possible (level 2), and (iii) does not have a good ontology specification and thus, would be fine with all the information that SEMANTIXS can extract (level 3).

The visualization and analysis module is responsible for all of the visualization ability provided by SEMANTIXS. It mainly provided the following two features:

- **Visualizing the extracted RDF Graphs**: As mentioned previously, the visualization service invokes an openly-available servlet from W3C, which can directly consume the output from SEMANTIXS. This results in descriptive (and possibly very large) graphs as illustrated in Figure 4.5.

- **Analyzing the extracted RDF Sub-graphs**: For analyzing parts of the graph, instead of the entire graph (which can turn out to be very large), SEMANTIXS provides another visualization service that focuses only on subgraphs formed out of the entire graph by focusing on certain entities or relationships.

CHAPTER 5. EMPIRICAL EVALUATION AND ANALYSIS USING SEMANTIXS

In previous chapters, we presented our frameworks for information extraction from text and associated system implementation. We also elaborated on how our algorithms are based on extraction rules to identify and extract candidate information constructs from a sentence. The performance of our algorithms critically depends on the ability of those rules to extract the information as a fraction of the total information present in the text input. In this chapter, we discuss some of our experiments and present the results in a coherent manner that clearly highlight the effectiveness of our approach towards complex relationship extraction.

5.1 Evaluation Scenario

We planned our evaluation in such a way that the relevance of our complex relationship subset, extraction rules as well as representation framework is demonstrated in a clear way. We randomly selected a few text fragments from web articles, each with high levels of complexity with respect to the sentential forms. The reason for selecting random online articles (and not a pre-annotated, well-defined text corpora) was that we wanted to test the relationship extraction capability of our system on free text and did not want to have any anticipation of what to expect in the input. We processed these articles with SEMANTIXS operating in level 3 (4.1) so that it would extract as much information as it can and perform enrichments. We chose the widely used DBpedia knowledge-base for domain specification. We elaborate more on our experimental setup in the next section.

5.2 Experimental Setup: Text, Ontology and Instances

5.2.1 Text

In order to test our system on some of the most complex sentences that can naturally occur in text, we decided to run it on text extracted from online news articles. Further, we also wanted to demonstrate the feasibility of our framework to perform complex queries in order to answer certain questions about the information expressed within unstructured text. We queried[1] *CBS News.com* with the keyword "Dow Jones". At the time of our experiment, this query resulted in about 5228 articles, videos and other related material. We randomly selected a few text articles from the results of the query, totaling over 80 long and complex sentences, and used them for our experiments. The idea was to extract information focused around a few entities ("Dow Jones" would always feature with some other key entities) and then query the resulting RDF graph to discover interesting facts about them.

Table 5.1 Overview of Experimental Texts: Counts of Positive and Negative Instances

Text #	**Simple**	**Type 1**	**Type 2**	**Type 3**	**References**
1	7/6	7/1	1/2	2/1	4/2
2	13/0	6/0	3/0	3/0	2/2
3	23/5	12/2	2/0	2/1	6/2
4	18/14	5/0	1/0	2/0	1/2
Total	61/25	30/3	7/2	9/2	13/8

Before running our experiments, we manually analyzed the text to ensure that it indeed comprised of a significant number of complex sentential forms. We found that the experimental texts were not rich enough in relationship types 2 and 3, and thus, we augmented some of the texts with those relationships. Table 5.1 shows an overview of the counts of relationships manually discovered in the texts after augmentation. The counts indicate the number of positive/negative instances of relationships found in text. Positive instances comprise relationships that we expect to be identified and extracted as per the domain description, and negative instances comprise those which we do not.

[1]http://www.cbsnews.com/1770-5_162-0-4.html?query=Dow+Jones&searchtype=cbsSearch&tag=mncol;pageb

5.2.2 Ontology and Instance Data

For capturing a good-enough domain-specification, we chose DBpedia since it has one of the most comprehensive set of ontology and types available online. We utilized the latest DBpedia ontology[2] and a subset of instances[3] to perform our experiments. We also used a custom-made FOAF dataset for capturing some extraneous names that were expected to occur in the texts, but were not captured in the DBpedia type definitions. This can be thought of as a logical extension to the DBpedia types without references in the DBPedia ontology.

Further, since we operate SEMANTIXS in level 3, we get quite a few *GenericRelations* that SEMANTIXS extracts and adds on its own irrespective of whether there is a corresponding definition in the DBpedia ontology or not. Our intention in doing this is to be able to extract a decent-sized RDF graph, so that we can demonstrate complex querying on it. Since the complexity lies in the *structure* of the captured information and not the *content*, we chose to go ahead with this approach.

Table 5.2 Correctly Classified Information: Counts of Positive and Negative Instances

Text #	**Simple**	**Type 1**	**Type 2**	**Type 3**	**References**
1	7/3	5/1	0/2	2/1	3/0
2	13/0	5/0	2/0	3/0	2/0
3	22/3	10/1	1/0	2/1	5/1
4	16/10	4/0	1/0	2/0	1/1
Total	58/16	24/2	4/2	9/2	11/2

5.3 Results and Interpretation

The results from our information extraction are presented in Table 5.2. This table reveals the counts for correctly classified information (extracted, validated and represented correctly, or rejected at some stage as appropriate). For a listing of the texts utilized for these experiments, refer appendix A.

[2]http://wiki.dbpedia.org/Downloads34#dbpediaontology
[3]http://wiki.dbpedia.org/Downloads34#ontologytypes

	p	**n**
p'	58	9
n'	3	16

Table 5.3 C.M. for Simple

	p	**n**
p'	24	1
n'	6	2

Table 5.4 C.M. for Type 1

	p	**n**
p'	4	0
n'	3	2

Table 5.5 C.M. for Type 2

	p	**n**
p'	11	6
n'	2	2

Table 5.6 C.M. for References

As pointed out in Chapter 3, the sentences comprising complex relationship often contain one or more simple relationships. Thus, in coming up with Table 5.2, we used the following methodology on a per-sentence basis:

- If the information extracted comprised a complex relationship of type 1 or 2, then the correctness would be judged based on whether (i) the correct structural representation was extracted for complex relationship (ii) the correct semantic representation was extracted for the simple relationship within.

- If the information extracted was correct and complete (in the sense of extracting all the relationships present in the sentence), it would contribute to all the corresponding counts for the extracted relationships.

- If the information extracted was partially-correct and not complete (in the sense of extracting all the relationships present in the sentence), it would still contribute to the corresponding counts for the correct relationship(s) that have been extracted by SEMANTIXS.

For consistency, we used analogous methodology while creating the reference Table 5.1.

	p	**n**
p'	9	0
n'	0	2

Table 5.7 C.M. for Type 3

Table 5.8 Results: Precision, Recall and F-measure

Text #	**Simple**	**Type 1**	**Type 2**	**Type 3**	**References**
Precision	0.86	0.96	1.0	1.0	0.65
Recall	0.95	0.80	0.57	1.0	0.85
F-measure	0.90	0.87	0.73	1.0	0.74

5.3.1 Interpretation

Confusion matrices for individual relationship types are reported in Tables 5.3, 5.4, 5.7 and 5.6, and the precision, recall and f-measures are reported in Table 5.8.

From the results, we observe that the algorithm achieves 86% precision in extraction of simple relationships. Its high recall is explained by the fact that we operated in the advanced (level 3) mode for our experiments (created new relationships). Further, it had 80% recall in identifying and extracting complex relationships of type 1. In case of complex relationships, precision value does not provide much insight into the performance since we base our correctness measure on the structure extracted. In case of type 2, the recall degraded due to extra complexity leading to false positives (refer next section). In case of type 3, the correctness is based entirely on whether the steps in rule 3.3.3.3 are carried out correctly, and not on the analysis of right and left fragments. Thus, as expected, the precision and recall are 100% for this type.

We now give an analysis of the errors and probable causes.

5.3.2 Discussion on Errors

In case of simple relationships, most false positives and false negatives were caused due to shallow syntactic comparisons to determine matches between the candidate information constructs and ontology concept, relationships or instances. For instance, not interpreting *General Motors* as a related words and thus, failing to match with the instance definition for it. In case of relationship type 1, most false negatives were caused due to multi-level dependency structures and cross-sentential references. For example, *But she acknowledged* ***that*** *with* ***that*** *gain came pain, on the backs of taxpayers.* Although our extraction routine is designed to handle recursive structures, it found hard to correlate implicit referrants (*that gain*) with their correct references. This resulted in a performance hit in terms of the recall. The algorithm was, however, able to handle these cases when they occurred alone, correctly resolving about 65% of pronoun references with 85% recall. Further, in case of type 1, algorithm encountered problems similar to the case when it was handling simple relationships. This happened when it tried to find matches for the outer subject and predicate. This resulted in some false positives leading to less than 100% precision that we had originally expected for complex relationships.

In case of type 2, most false negatives occurred again due shallow syntactic comparisons to determine a match for the extracted qualification and value. Further, there were a few outliers that contained relevant information, which should have been extracted, however, the generated dependency graphs did not have the pattern that the algorithm was expecting. In case of references, the algorithm achieves high recall, but low precision since we use rather naive pronoun resolution methodology. Due to this, it resolves pronouns aggressively, leading to many false positives and resulting in a decrease in the precision.

Apart, many of the failing cases were co-references (example - *Jill Schlesinger* and *Mrs. Schlesinger*) and our algorithm is not designed to handle them. About 1% of the sentences contained negations, conjunctions other than *and*, or dependency structures that we had not originally identified while defining our rules.

5.4 Querying the Graph

We now turn our attention to the extracted RDF metadata, which forms a semantic graph and demonstrate how our information extraction framework can enable complex querying in order to answer certain questions about the information expressed within unstructured text. In Figure 5.1, we have only shown that fragment of the generated graph(s), which is related to the entity *Dow Jones* in some way. Refer appendix A for some more examples of the extracted information (in RDF).

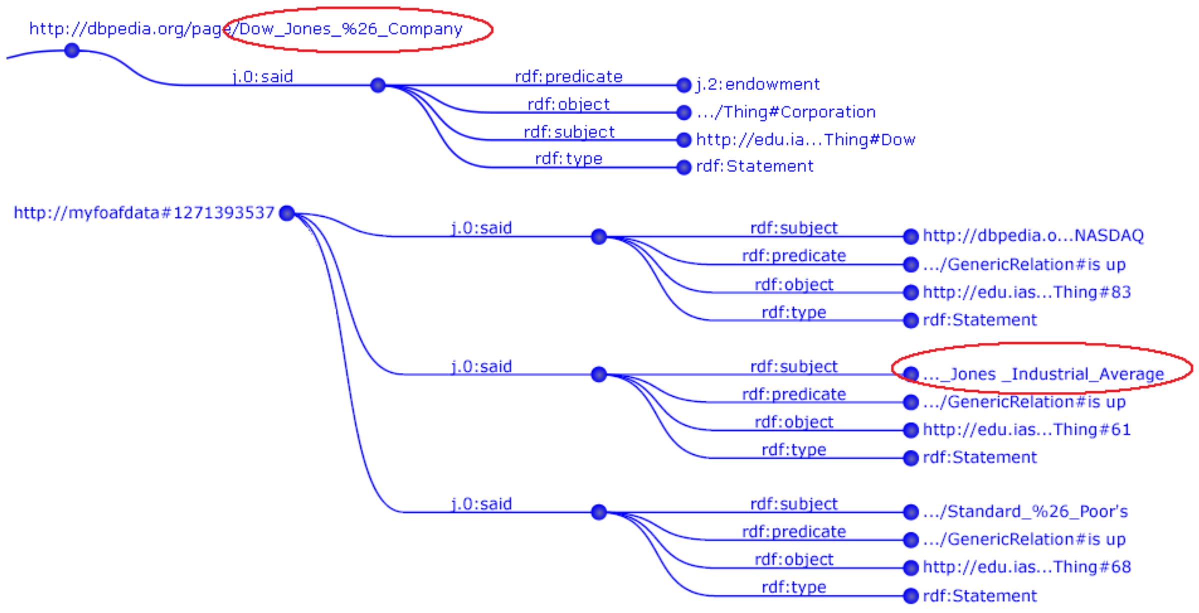

http://dbpedia.org/page/Dow_Jones_%26_Company
j.0:said
rdf:predicate
rdf:object
rdf:subject
rdf:type
j.2:endowment
.../Thing#Corporation
http://edu.ia...Thing#Dow
rdf:Statement
http://myfoafdata#1271393537
j.0:said
rdf:subject
rdf:predicate
rdf:object
rdf:type
http://dbpedia.o...NASDAQ
.../GenericRelation#is up
http://edu.ias...Thing#83
rdf:Statement
j.0:said
..._Jones _Industrial_Average
.../GenericRelation#is up
http://edu.ias...Thing#61
rdf:Statement
j.0:said
.../Standard_%26_Poor's
.../GenericRelation#is up
http://edu.ias...Thing#68
rdf:Statement

Figure 5.1 RDF Sub-graph for the Entity "Dow Jones"

5.4.1 Formulating Complex Questions

We now give an outline of a generic question-asking paradigm and a query-plan for the graphs extracted by our framework. For illustrative purposes, we also give concrete examples of these questions that are fairly interesting from the perspective of our extracted information.

1. **Finding the subjects of assertions that were made about an entity:** *Who made any assertions about Dow Jones ?*

2. **Finding entities based on complex criteria:** *What are the entities that Dow Jones made qualified statements about ?*

3. **Finding entities based on relationship participation:** *Which entity appears in most facts ?*

4. **Finding entities based on participation in reified statements:** *Which entity is most talked about ?*

5.4.2 Formulating a Query-plan

We now give an outline of the approach that can be taken to answer these questions by forming SPARQL queries. These queries can be fired on a single Jena model, loaded with all the generated RDF graphs.

Table 5.9 Answers to Complex Questions

Question #	**Answer #**
1	Bancrofts, Schlesinger
2	General Motors, Citi
3	(Murdoch's) News Corp
4	Dow Jones

1. For answering 1, we look for reified statements with the entity *(Dow Jones)* as the subject and find all the triples that have this statement as its object. The subjects of such triples give us an answer to the question. We can also find what was actually asserted by simply returning the statement (object) of the triples that we found.

2. For answering 2, we look for triples with *Dow Jones* as the subject and from those, we find all such triples that have a reified statement as their object. From these statements, we find all those that have a *QualifiedRelationship* instance as their subject and *hasQualifiedSubject* as predicate. Then, we return the object of these statements as our answer.

3. For answering 3, we simply return the resource that appears most number of times as the object of simple triples (not statements).

4. For answering 4, we group all the reified statements by their subjects. Then, we find the group whose statements appear as the object in triples most number of times. We return the common subject of the statements of that group as our answer.

Some of the answers to the questions in 5.4.1 from querying on the graphs generated from our experiments are enlisted in Table 5.9.

It is clear from the above analysis that our information extraction framework can be quite useful in extracting complex information that can eventually be used to ask fairly complex questions that relate entities to statements and facts. This can have numerous applications from question answering to domain-specific expert systems. We discuss some of these in the next chapter.

CHAPTER 6. CONCLUSION

6.1 Summary

Many applications call for methods for automatic extraction of structured information from unstructured natural language text. Due to the inherent challenges of natural language processing, most of the existing methods for information extraction from text tend to be domain specific. In this thesis, we explore a modular ontology-based approach to information extraction that decouples domain-specific knowledge from the rules used for information extraction. Specifically, we describe an ontology-driven extraction of a subset of nested complex relationships (e.g., Joe reports that Jim is a reliable employee) from free text. The extracted relationships are represented in the form of RDF (resource description framework) graphs which can be stored in RDF knowledge bases and queried using query languages for RDF. We have also designed and performed analytical experiments that offer some evidence of the utility of the proposed ontology-based approach to extraction of complex relationships from text. For being able to do this, we have implemented our algorithms and rule-engine in the form of SEMANTIXS (System for Extraction of doMAin-specific iNformation from Text Including compleX Structures).

6.2 Contributions

The main contributions of this thesis include:

1. We have identified a subset of nested complex relationship patterns and formulated extraction rules to identify them.

2. Next, we have formulated the following set of algorithms:

(a) An algorithm that utilizes the sets of extraction rules to extract the relationships and entities (information constructs) from unstructured text.

(b) An algorithm that validates and semantically associates the extracted constructs with the given domain ontology, and perform ontology enrichment with newly found relationships, whenever possible.

(c) An algorithm to represent these constructs using existing (non-extended) RDF specification.

(d) An algorithm that utilizes the above three to perform the task of information extraction from textual sources.

3. Based on the above analysis and algorithms, we have implemented SEMANTIXS, a system for ontology-guided extraction and semantic representation of structured information from unstructured text. The system is available as an open-source software under GNU General Public License at SEMANTIXS home. We have also designed and performed certain analytical experiments and reported the results in order to clearly demonstrate the significance of our overall framework.

6.3 Further Work

Besides being a significant extension of the current state of the art for complex relationship extraction, the work presented in this thesis provides an extensible meta-level framework on top of which numerous applications and research threads can be based. Some of these are outlined below:

- **Enhancements to Improve Precision and Recall**: As we mentioned in section 3.4.1, we perform simple syntactic comparisons to determine matches between extracted candidate information constructs and ontology elements (concepts, relationships, instances, etc). This approach is limited in many ways and can be improved upon by considering synonyms, or referring external resources (like [8]), performing indirect resolutions, etc. The precision can be further improved by incorporating better algorithms for pronoun resolution and adding a module that resolves co-references. All of these and other similar linguistic enhancements are separate research tracks in themselves and can be used to improve any system for information extraction from text.

- **Complex Knowledge Discovery and Question Answering**: In information retrieval, question answering refers to the general task of automatically answering a question posed in natural language. There have been recent efforts [25] that utilize conceptual graph formalism to address this task. As briefly illustrated in 5.4.1, our framework can be utilized to capture relationships between entities and facts (with qualifications) in an RDF graph. Once such a linked information structure has been extracted for a specific domain, it can then be queried upon using SPARQL. Alternatively, knowledge discovery algorithms can be applied to it, in order to extract paths between entities as newly discover knowledge. A few examples of such efforts are [3] and [4].

- **Opinion Mining and Recommendation Systems**: Since our framework is able to extract relationships between entities and facts, it can be easily extended to build information graphs comprising entirely of opinions or recommendations (*said, feels, claims, recommends, etc*). Once such an RDF graph has been extracted, methodologies similar to those mentioned under the second point (above) can be employed to mine opinions.

- **Ontology-building and Domain Analysis**: The module used to handle mismatches (3.4.3.2) in validation enriches the ontology by generalizing or adding new properties. This can be extended further to build the domain ontology from scratch, or analyze the domain in general. A few examples of methodologies for ontology building from text that utilize similar techniques are [5] and [6].

APPENDIX A. EXPERIMENTAL TEXTS AND EXTRACTED INFORMATION

Text Fragments Used in Experiments

Article 1 - http://www.cbsnews.com/sections/earlyshow/living/money/main500173.shtml

Article 2 - http://www.cbsnews.com/8301-503983_162-5054220-503983.html

Article 3 - http://www.cbsnews.com/stories/2007/06/01/business/main2873658.shtml

Article 4 - http://www.cbsnews.com/stories/2006/10/05/business/main2069040.shtml

Sample Extracted Information in the Form of RDF Metadata

A.0.1 Sample Text Fragment

With Wall Street closing yesterday at 10,552, CBSMoneywatch.com editor-at-large Jill Schlesinger said the Dow Jones Industrial Average is up over 61% over the past year. "That is the best one-year since the Depression, the best 12 months," she said on CBS' "The Early Show" this morning, also noting that the Nasdaq is up 83%, and Standard & Poors is up 68%. "I really want to point out one thing that's a little scary: The S&P 500 is still at the same level as 1998 - we still are 27% below where we were in 2007," Schlesinger said. Schlesinger said the sectors which recovered most in the last year were business services (publishing, advertising, consulting) up +210%; media business ("thankfully for us!") up 138%; and a 155% jump in financial services - "all those taxpayer-bailed out companies, the banks, the mutual fund companies, the big insurance companies," she said. Schlesinger also said the numbers bode well for the recovery of 401(k)s.

A.0.2 Sample Generated RDF Sub-graph

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:j.0="http://edu.iastate.airl/semtus/GenericRelation#"
xmlns:j.2="http://dbpedia.org/ontology/" >
<rdf:Description rdf:about="http://myfoafdata#1271393537">
<j.0:want rdf:resource="http://edu.iastate.airl/semtus/Thing#scary"/>
 <j.0:said rdf:nodeID="A0"/>
 <j.0:said rdf:nodeID="A1"/>
 <j.0:said rdf:nodeID="A2"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A1">
 <rdf:subject rdf:resource="http://dbpedia.org/page/Dow_Jones
 _Industrial_Average"/>
 <rdf:predicate rdf:resource="http://edu.iastate.airl/semtus/GenericRelation
 #is up"/>
 <rdf:object rdf:resource="http://edu.iastate.airl/semtus/Thing#61"/>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns
 #Statement"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A2">
 <rdf:subject rdf:resource="http://dbpedia.org/page/NASDAQ"/>
 <rdf:predicate rdf:resource="http://edu.iastate.airl/semtus/GenericRelation
 #is up"/>
 <rdf:object rdf:resource="http://edu.iastate.airl/semtus/Thing#83"/>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns
 #Statement"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
 <rdf:subject rdf:resource="http://dbpedia.org/page/Standard_%26_Poor's"/>
 <rdf:predicate rdf:resource="http://edu.iastate.airl/semtus/GenericRelation
 #is up"/>
 <rdf:object rdf:resource="http://edu.iastate.airl/semtus/Thing#68"/>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns
 #Statement"/>
 </rdf:Description></rdf:RDF>
```

A.0.3 RDF Serialization for Graph 3.11

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:j.0="http://edu.iastate.airl/semtus/GenericRelation#">
<rdf:Description rdf:about="http://myfoafdata#534485411">
<j.0:scoredRuns rdf:resource="http://edu.iastate.airl/semtus/Thing#200"/>
 </rdf:Description>
 </rdf:RDF>
```

A.0.4 RDF Serialization for Graph 3.12

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:j.0="http://edu.iastate.airl/semtus/GenericRelation#">
<rdf:Description rdf:about="http://edu.iastate.airl/semtus/Thing
#Microsoft ad">
 <j.0:says rdf:nodeID="A0"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
 <rdf:subject rdf:resource="http://dbpedia.org/resource
 /Macintosh_Business_Unit"/>
 <rdf:predicate rdf:resource="http://dbpedia.org//ontology/coolingSystem"/>
 <rdf:object rdf:resource="http://edu.iastate.airl/semtus/Thing#customers"/>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns
 #Statement"/>
 </rdf:Description>
 </rdf:RDF>
```

A.0.5 RDF Serialization for Graph 3.13

```
<rdf:RDF    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:j.0="http://edu.iastate.airl/semtus/GenericRelation#"
xmlns:j.1="http://edu.iastate.airl/semtus/QualifiedRelationship#" >
<rdf:Description rdf:about="http://edu.iastate.airl/semtus/
QualifiedRelationship#QualifiedRelationship_1">
<j.0:probability rdf:resource="http://edu.iastate.airl/semtus/Thing#high"/>
 <j.1:hasQualifiedPredicate rdf:resource="http://edu.iastate.airl/semtus/
 GenericRelation#scoredRuns"/>
 <j.1:hasQualifiedObject rdf:resource="http://edu.iastate.airl/semtus/
 Thing#200"/>
 <j.1:hasQualifiedSubject rdf:resource="http://myfoafdata#534485411"/>
 </rdf:Description>
 </rdf:RDF>
```

BIBLIOGRAPHY

[1] Data, Information, Knowledge, and Wisdom, http://www.systems-thinking.org/dikw/dikw.htm. URL http://www.systems-thinking.org/dikw/dikw.htm.

[2] M. C. Marneffe, B. MacCartney and C. D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. *In LREC*, 2006.

[3] K. Anyanwu, A. P. Sheth. Rho-Queries: enabling querying for semantic associations on the semantic web. *WWW*: 690-699, 2003.

[4] C. Ramakrishnan, W. H. Milnor, M. Perry and A. P. Sheth. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explorations 7(2)*: 56-63, 2005.

[5] P. Gawrysiak, et al. Text Onto Miner A Semi Automated Ontology Building System. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, Vol. 4994/2008, 2008.

[6] R. Valencia-Garcia, et al. An Approach for Ontology Building from Text Supported by NLP Techniques. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, Vol. 3040/2004, 2004.

[7] Part-of-speech tagging, http://en.wikipedia.org/wiki/Part-of-speech_tagging. URL http://en.wikipedia.org/wiki/Part-of-speech_tagging.

[8] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, Vol. 38, No. 11: 39-41, 1995.

[9] Learning by Reading, http://userweb.cs.utexas.edu/users/mfkb/RKF/projects/lbr.html/. URL http://userweb.cs.utexas.edu/users/mfkb/RKF/projects/lbr.html/.

[10] D. S. Kim, K. Barker and B. Porter. Knowledge integration across multiple texts. *Proceedings of the fifth international conference on Knowledge capture*: 49-56, 2009.

[11] D. S. Kim and B. Porter. Integrating declarative knowledge: Issues, algorithms and future work. *Proceedings of the Spring AAAI Symposium Series*, 2008.

[12] M. Bates, R. M. Weischedel. Challenges in natural language processing. *Cambridge University Press.*

[13] C. Ramakrishnan, K. J. Kochut and A.P. Sheth. A Framework for Schema-Driven Relationship Discovery from Unstructured Text. *International Semantic Web Conference*: 583-596, 2006.

[14] Critical Assessment of Information Extraction Systems in Biolog, http://www.mitre.org/public/biocreative/. URL http://www.mitre.org/public/biocreative/.

[15] J. Saric, L. J. Jensen, R. Ouzounova, I. Rojas and P. Bork. Extraction of regulatory gene/protein networks from Medline. *Bioinformatics*, Vol. 22 no. 6: 645650, 2006

[16] C. Ramakrishnan, P. N. Mendes, S. Wang and A. P. Sheth. Unsupervised Discovery of Compound Entities for Relationship Extraction. *Lecture Notes in Computer Science, Springer Berlin / Heidelberg*, Vol. 5268/2008: 146-155, 2008.

[17] Q. N. Rajput, S. Haider, N. Touheed. Information Extraction from Unstructured and Ungrammatical Data Sources for Semantic Annotation. *World Academy of Science, Engineering and Technology*, 2009.

[18] E. Riloff. Automatically constructing a dictionary for information extraction tasks. *Proceedings of the 11th National Conference on Artificial Intelligence*, AAAI-93: 811816, 1993.

[19] S. Huffman. Learning information extraction patterns from examples. *Workshop on new approaches to learning for natural language processing*, IJCAI-95: 127142, 1995.

[20] J. Kim and D. Moldovan. Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactiops on Knowledge and Data Engineering*, 7(5): 713724, 1995.

[21] S. Soderland, et al. Crystal: Inducing a conceptual dictionary. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, IJCAI-95: 13141319, 1995.

[22] K. Fundel, R. Kuffner and R. Zimmer. RelExRelation extraction using dependency parse trees. *Bioinformatics*, Vol. 23 no. 3: 365371, 2007

[23] C. Friedman, et al. GENIES: a natural-language processing system for the extraction of molecular pathways from journal articles. *Bioinformatics*, Vol. 17 Suppl. 1: 1367-4803, 2001

[24] P. Cimiano, A. Pivk, L. Schmidt and S. Staab. Learning taxonomic relations from heterogeneous evidence. *Ontology Learning from Text: Methods, evaluation and applications*, IOS Press, 2005

[25] W. Salloum. A Question Answering System based on Conceptual Graph Formalism. *KAM*, 2009.

[26] The Stanford Parser: A statistical parser, http://nlp.stanford.edu/software/lex-parser.shtml. URL http://nlp.stanford.edu/software/lex-parser.shtml.

[27] An Introduction to Jena RDF API, http://jena.sourceforge.net/.
URL http://jena.sourceforge.net/tutorial/RDF_API/index.html.

[28] R. McDonald, K. Lerman, and F. Pereira. Multilingual Dependency Parsing with a Two-Stage Discriminative Parser. *Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, 2006.

[29] Natural Language Toolkit (NLTK), http://www.nltk.org/. URL http://www.nltk.org/.

[30] S. Blohm and P. Cimiano. Scaling up pattern induction for web relation extraction through frequent itemset mining. *Proc. of the KI 2008 Workshop on Ontology-Based Information Extraction Systems*, 2008

[31] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *JMLR*, 2003

[32] R. McDonald, et al. Simple algorithms for complex relation extraction with applications to biomedical IE. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*: 491-498, 2005

[33] A. Yates, et al. TextRunner: open information extraction on the web. *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*: 25-26, 2007

[34] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. *Proceedings of the fifth ACM conference on Digital libraries*: 85-94, 2000

[35] G. Neumann and F. Xu. Intelligent Information Extraction. *LT-lab, DFKI, Germany*, 2004

[36] N. Bach and S. Badaskar. A survey on relation extraction. *Language Technologies Institute, Carnegie Mellon University*, 2007

[37] Calais, http://en.wikipedia.org/wiki/Calais_(Reuters_Product).
URL http://en.wikipedia.org/wiki/Calais_(Reuters_Product).

[38] Unified Medical Language System, http://www.nlm.nih.gov/research/umls.
URL http://www.nlm.nih.gov/research/umls.

[39] Medical Subject Headings, http://www.nlm.nih.gov/mesh.
URL http://www.nlm.nih.gov/mesh.

[40] Resource Description Framework (RDF), http://www.w3.org/RDF.
URL http://www.w3.org/RDF.

[41] RDF Semantics, http://www.w3.org/TR/rdf-mt/#Reif.
URL http://www.w3.org/TR/rdf-mt/#Reif.

[42] Linked Data, http://linkeddata.org.
URL http://linkeddata.org.

[43] Web Ontology Language (OWL), http://www.w3.org/TR/owl-features.
URL http://www.w3.org/TR/owl-features.

[44] F-logic, http://en.wikipedia.org/wiki/F-logic.
URL http://en.wikipedia.org/wiki/F-logic.

[45] SPARQL, http://www.w3.org/TR/rdf-sparql-query.
URL http://www.w3.org/TR/rdf-sparql-query.

[46] Domain and upper ontologies.
URL http://en.wikipedia.org/wiki/Ontology_(information_science).

[47] World Wide Web Consortium (W3C), http://www.w3.org/standards.
URL http://www.w3.org/standards.

[48] Description Logics, http://dl.kr.org. URL http://dl.kr.org.