LOGICAL FORM IDENTIFICATION FOR MEDICAL CLINICAL TRIALS

by

Clint A. Tustison

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Arts

Department of Linguistics and English Language

Brigham Young University

December 2004

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Clint A. Tustison

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Deryle W. Lonsdale, Chair
Date	David W. Embley
Date	Alan K. Melby

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Clint A. Tustison in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Deryle W. Lonsdale Chair, Graduate Committee

Accepted for the Department

Lynn Henrichsen
Department Chair

Accepted for the College

Van C. Gessel
Dean, College of Humanities

ABSTRACT

LOGICAL FORM IDENTIFICATION FOR MEDICAL CLINICAL TRIALS

Clint A. Tustison

Department of Linguistics and English Language

Master of Arts

Programming a computer to understand natural language has become increasingly more important as the amount of natural language in electronic format has increased. One of the areas where text understanding is valuable is medical literature. Most of the research on information extraction and text understanding in medical literature has focused on medical abstracts and researchers have used various tools to get interesting results. While medical abstracts have proved very fruitful for this type of research, very limited research has been done on extracting information and understanding text from medical clinical trials. Clinical trials are very important to doctors and medical organizations and programming a computer to automatically understand the data would be valuable. This thesis presents LG-Soar, a system capable of parsing eligibility criteria in clinical trials and outputting a semantic representation using predicate logic. This approach is different than other information extraction approaches to natural language in that it uses a cognitive modeling engine to convert the parsed sentences into corresponding predicate logic forms. Initial results reveal that LG-Soar is a viable system for doing natural language text extraction and understanding.

ACKNOWLEDGMENTS

I would like to thank the National Science Foundation for supporting this research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Google currently retrieves 4,285,199,774 web pages millions of times a day.[1] More and more people are accessing the huge amounts of data available electronically and are becoming increasingly dependent on understanding this information. With an increase in computing power, textual analysis and understanding has become an important area of research in the fields of information extraction and natural language processing (NLP).

As electronic texts become more available to researchers (and humans in general), an interesting dichotomy has emerged. On one hand, electronic text posted on the Internet caters to users' ability to read and analyze that information. Those interested in putting information on the Internet design the data's structure to be easy for humans to digest. Data designed for human consumption must follow conventional syntactic and semantic constraints of the users' natural language.

On the other hand, humans have very limited computational capacity for analyzing in any totality the vast amount of electronic information available. Recently, researchers have turned to computing power to help humans access and process large quantities of data on the Internet. This work has involved devising new strategies and algorithms to convert electronic natural language text into various formats that can then be processed by a computer. One of the ways this can happen is by using information extraction to manipulate the data. Some information extraction strategies use natural language techniques, while others rely more heavily on statistical methods. Many times, the type of data analyzed determines the method used.

[1]http://www.google.com - Accessed July 2004.

Researchers design information extraction systems to perform various tasks, and these tasks require various levels of linguistic processing. Some systems are only concerned with parsing out the extracted information and therefore only require the use of a syntactic parser. Other systems need more in-depth processing and include a semantic component that can give some meaning to the extracted information. Yet other systems are dependent on real-world knowledge and require a pragmatic component to relate the data gathered from the system to outside information.

The Internet gives users information on nearly every subject known to man. While the majority of these subjects are not interesting to researchers, a few subjects have caught their attention. One area recently identified as being interesting is the medical domain. Much of the NLP research done with medical literature has involved developing systems that extract different types of relationships from text.

This thesis describes a general-purpose NLP tool designed to identify and extract logical forms from medical clinical trials.[2] Because of the various components which are used in the design of this tool, LG-Soar is not only able to quickly and robustly parse this type of natural language, but is also able to take that parsed output and derive a higher level of semantic meaning than what would be gained by the parse itself. The corresponding semantic information can then inform further processing that requires a more structured meaningful input.

This thesis comprises five chapters. Chapter 2 is a review of different tools used to extract and analyze information from electronic texts. In order to accomplish the task of understanding the medical trials used in the current research, it is important that a syntactic component and a syntax-to-semantic component work together. Chapter 3 will explain the choice of components used in the LG-Soar system. Chapter 4 contains the technical details of the LG-Soar system and outlines how the components work together to create a robust engine. A discussion of the results and some concluding remarks form the basis for chapters 5, 6, and 7.

[2]The term *logical form* in this work is defined as a predicate logic form representing the shallow semantics of a given utterance. It is not to be confused with the standard usage of the term in current formalist syntax and semantics, where the term applies to a post-syntactic level of semantic representation.

Chapter 2

Literature Review

Information extraction research has increased over the last few years. Traditionally viewed as a tool used mainly by computer scientists, information extraction has emerged as an engineering method useful in a variety of disciplines. Its versatility has helped the field grow tremendously, and more research is continually being done to determine how to improve the various methods and techniques used.

2.1 Information Extraction Domains

Information extraction systems need input, and this input can take many forms. Since the invention of written language, text has become an important medium of information exchange. The advent of the computer age and digital technology has lead to the creation of media that can now be recorded and stored such as videos and DVDs. The information extraction field has attempted to keep pace by focusing on novel ways to extract different types of information from various media, including web video clips (Rosenfeld et al., 2003), and sports highlights (Radhakrishan et al., 2004).

2.1.1 Text

Text can take different forms, and usually the way the text is formatted will determine what method to use when attempting to extract information. Text can be classified into one of three categories: unstructured (or free), structured, and semi-structured (Magnani and Montesi, 2004).

Unstructured Text

Unstructured text follows natural language rules and grammar and is (for the most part) understandable by a human. It is free in the sense that it does not fit into any type of organizational structure such as would be found in a database or table. Newspaper articles and medical abstracts are examples of free text.

Because free text consists mainly of natural language prose, natural language processing techniques are often used when extracting information from this type of text. These types of techniques include syntactic analysis, semantic tagging, domain recognition, etc. While no unstructured text information extraction engine is able to perform at the same level as a human, unstructured text techniques do provide useful results that can be used to analyze various types of text.

Structured Text

Structured text refers to information that is contained in a database or information that follows a very rigid format. When the structure of the data is known, information extraction on this type of data is usually very straightforward. Regular expression and pattern matching techniques are often very useful when attempting to extract the required information.

Semi-Structured Text

Semi-structured text falls somewhere in between structured and unstructured text. It is usually more difficult to extract information from this type of text because it is not sufficiently structured to allow for full regular expression matching and, at the same time, it does not contain text free enough to use only traditional NLP techniques such as syntactic parsing and semantic tagging. Usually a combination of these techniques is used to extract information from semi-structured text.

2.1.2 Web-Formatted Data

Information extraction initially dealt with text documents, usually newspaper articles or abstracts. However, with the recent Internet explosion, researchers have focused more heavily on web-formatted data.

Text information on the web can take any of the forms mentioned earlier. Web text often contains some combination of itemized or bulleted lists, hyperlinks, tables, or tags. All of this information can be used as input for an information extraction engine, but care must be taken when designing the system so that the correct structures can be identified and extracted.

Additional information available on the web that is not easily identifiable is referred to as the hidden web. The hidden web contains information that usually can only be accessed by filling out a form and submitting information. Since many web-based applications dynamically provide data to those searching for information, the web pages are often generated dynamically each time a user fills out a specific query. Programs are being developed (Liddle et al., 2001; Raghavan and Garcia-Molina, 2001) that attempt to extract information from the hidden web.

2.2 Information Extraction Applications

Because information extraction is a widely researched field, researchers have used various techniques to extract data from different text domains. Each domain is unique and presents specific challenges to overcome. Some of these domains are briefly discussed below.

2.2.1 Book Reviews

One domain where information extraction approaches are used are recommender systems. These systems analyze users' likes and dislikes and recommend products to them. One such system that uses information extraction techniques along with machine-learning ones is LIBRA, a system developed at the University of Austin (Mooney and Roy, 2000). This system recommends book titles to users based on prior training information given to the system by the users themselves. A simple pattern-matching information extraction algorithm is used to extract data about each book title from Amazon.com, which is then presented to the user.

2.2.2 Genealogy

An extremely popular domain on the Internet is family history. Genealogy and genealogy-related searches now account for a very large portion of all Internet search traffic. Millions of people are accessing and posting genealogical information online. Because so many people are doing genealogical research, few standards exist which the majority of genealogy enthusiasts use to structure their information online. This makes it difficult for researchers to design systems to adequately extract relevant data from genealogical sites on the Internet.

As difficult as this can be, researchers have been attempting to do just that: extracting information such as names, birthplaces, deathplaces, etc. in order to make better sense of the huge amount of data available. Information extraction on this type of data has proven to be doable and has produced some very good initial results (Walker and Embley, 2004).

2.2.3 Medline Abstracts

Another domain of information which has recently received a lot of attention from information extraction researchers is the medical domain, specifically Medline abstracts. These abstracts are sponsored by the U.S. government, as well as the National Institutes of Health, and can be found online.[1] The Medline database contains a wealth of information about nearly every health topic imaginable, all from the National Library of Medicine. Medline abstracts contain useful information about various disease- and health-related issues. Because the information located in these abstracts is extremely useful, and because of the sheer amount of information, Medline abstracts have become an important resource for medical information extraction. Many researchers have turned their focus to extracting various types of relationships found in these abstracts. These types of relationships include gene relations (Stephens et al., 2001), protein inhibit relationships (Pustejovsky

[1]http://www.medlineplus.gov

et al., 2002), acronym-meaning pairs (Pustejovsky et al., 2001), abbreviation definitions (Schwartz and Hearst, 2003), and molecular binding relationships (Rindflesch et al., 2000).

2.3 Information Extraction Methods

Multiple methods are used by researchers when developing information extraction systems. Wrappers, ontologies, and parsers are some of the more popular tools used in information extraction.

2.3.1 Wrappers

Wrapper generation is a relatively new information extraction method used in many different systems. Wrappers are specialized programs that identify the interesting data within a document using pattern-matching approaches.

While wrapper technology has traditionally focused on semi-structured and structured text, wrappers using NLP-based approaches are more appropriate for free-language text documents. These NLP-approaches use traditional NLP techniques such as part-of-speech tagging and syntactic and semantic parsing and analysis. Some of the more well-known tools that have been developed using NLP approaches along with wrapper technology include RAPIER (Califf and Mooney, 1999), SRV (Freitag, 1998), and WHISK (Soderland, 1999).

2.3.2 Ontologies

Another method for doing information extraction relies on the construction of light-weight ontologies. These ontologies are built to recognize and organize the desired information to be extracted. Significant work in ontology-based approaches has been done by Brigham Young University's DEG (Data Extraction Group) (Embley et al., 1999). Their work has focused on extracting data from a variety of sources such as job announcements, digital camera information, and automobile classified advertisements.[2]

[2]http://www.deg.byu.edu

Table 2.1: Differences between dependency and link grammars

Dependency Grammar	Link Grammar
Notion of a root word	No notion of a root word
Links are not labeled	Links are labeled
A dependency exists between heads and dependents	Links are undirected
Cycles are not allowed in the structure	Links may form cycles
Grammar rules are dependency rules	Grammar rules are lexical rules

2.3.3 Parsers

In order to do any type of structured analysis on free text, a component that can parse the sentence into a syntactic representation is critical. This structure is necessary to understand even minimally how the individual words of a text are related and how meaning (at a very basic level) is derived. Theories and methodologies of syntactic parsing abound and systems have been developed using particular theories as the basis for the way they do the syntactic analysis. Due to the fact that a number of ways to derive the syntactic form from a particular piece of textual information exist, the goal a particular system has, as well as the type of text it will process, is extremely important when determining what type of parser to use. Parsing strategies range from statistical to linguistic and the next section will discuss a few of the different parsing approaches used by NLP researchers, including those who do information extraction.

Lexical Dependency Parsing

One of the most popular parsers used in information extraction today is the Collin's parser (Collins, 1996), a parser which has proved to be successful in parsing large amounts of data. This parser is based on lexical dependency theory, a theory which combines statistics and probabilities of lexical dependencies occurring within an utterance to determine the best parse. Lexical dependency parsing is similar to other approaches in that one of its tenets is the importance of modeling the head-modifier relationship between pairs of words.

Dependency and Link Grammars

Dependency grammars are based on the idea that a sentence can be analyzed by connecting the words in a sentence using links. The link grammar parser uses some of the tenets outlined by dependency grammar theory, but there are differences that exist between the two. Table 2.1 outlines the major differences between dependency and link grammar formalisms as found in (Sleator and Temperley, 1991). An elaboration of these differences can be found in Schneider's work on constituency, dependency, and link grammars (Schneider, 1998).

Exemplar-Based Parsing

Another parsing approach used among researchers is exemplar-based parsing. This approach uses approaches similar to statistical techniques. The main difference between these exemplar-based approaches and other statistical approaches is that the former are lazy learners (all the training data is stored), while the latter are greedy learners (once the knowledge has been abstracted from the training data, the training instances are discarded). Two examples of exemplar-based systems that have proven successful in a variety of NLP tasks are TiMBL (Daelemans et al., 1999) and Analogical Modeling (Jones, 1996; Skousen, 1989; Skousen et al., 2002).

Morpho-Semantic Parsing

One parsing strategy that has been used in the medical language field is morpho-semantic parsing (Baud et al., 1998). Most linguistic-based parsing systems tend to focus their efforts on parsing text at the word level. While this has proved successful in a number of applications, it is not always the best choice. In morpho-semantic parsing, the division for the parse is actually at the morpheme level, rather than the word level. This method excels when parsing data where rich semantic information is encoded in the morphemes of a given word. Medical vocabulary is well-known for its use of morphemes and these morphemes encode rich semantic data about the meaning of a particular word. By parsing out this information, information extraction systems can better understand the data.

Table 2.2: Ambiguities in English

Lexical Ambiguity	
1. I got a bat for Christmas.	a. bat = animal b. bat = baseball equipment
Structural Ambiguity	
2. . . . good coaches and players.	a. [NP good coaches and players] b. [NP good coaches] and [NP players]
Quantificational Ambiguity	
3. Everyone loves someone.	a. $\forall x[\exists y[loves(x, y)]]$ Everyone loves some person or other. b. $\exists y[\forall x[loves(x, y)]]$ There is one person everyone loves.
Morphological Ambiguity	
4. Clint's. . .	a. Clint's a linguist. [is] b. Clint's always worked hard. [has] c. Clint's thesis is done. [possession]

2.4 Problems in Information Extraction

Information extraction researchers want to create robust systems that can rapidly process incoming text. Despite these goals, researchers tend to program their systems to focus more heavily on either speed or robustness. This is often determined by the needs and desires of the end user. In order to develop a system that rapidly processes text in a robust fashion, many challenges must be overcome. One of these challenges is dealing with the text itself.

2.4.1 Ambiguity

Natural language is notorious for its ambiguity. Ambiguity can arise for a variety of reasons, and each of these can be combined with others to create very difficult-to-understand sentences for even native speakers of the language in question.

Four types of ambiguities which occur in English text are presented in Table 2.2, and it is easy to see how multiple readings or parses can be interpreted from a single sentence. Even though the examples only show two different readings for each sentence, typical English sentences are generally much longer, and therefore much more ambiguous.

Table 2.3: Prepositional phrase ambiguity

No. of PPs	No. of Parses
2	2
3	5
4	14
5	132
6	469
7	1430
8	4867

Sometimes they can result in hundreds of different parses, some which are grammatically correct and some which are not.

A variety of factors contribute to the number of parses a single sentence can contain. Quantifier clauses, relative clauses, conjunctions, and prepositions are just some of the devices used in English which are notorious for increasing the ambiguity of a particular sentence. Table 2.3 shows how prepositional phrases in a sentence can drastically increase the number of possible parses.

With just four prepositional phrases, a sentence such as *The man on the sand with the dog on the leash by the ocean fell* results in fourteen different parses. Prepositions alone have the ability to quickly make a sentence highly ambiguous. Add to the sentence any type of lexical, scope, or morphological ambiguity and the number of parses of a single sentence drastically increases.

Humans are generally able to make sense out of ambiguous phrases and sentences when they are kept to a minimal. In order to be effective (as long as the goal of the parser is not to specifically model human behavior), information extraction systems must be able to determine not only the number of different parses a sentence can have, but which parses are grammatically correct and which are not. They must also be able to determine, based on a number of internal and external clues, which one of the many possible parses is the most likely or most correct in a given context.

2.4.2 Text Difficulty

Another problem often faced by information extraction systems is determining the level of text that will be analyzed. Textual information varies in terms of difficulty. Even newspaper text varies, for example, and one can quickly understand this problem by comparing the text difficulty of a small local newspaper and the Wall Street Journal. Add to that the variation of difficulty that occurs across domains, and the text register can become quite problematic.

2.4.3 Pragmatics

Another challenge for information extraction systems is trying to deal with contextualized meaning, or pragmatics. In order for a computer to understand natural language text and convert it to a machine-readable format, it must understand at minimum the syntactic information encoded within the utterance. In order to understand the deeper meanings within the electronic texts, though, a semantic component must also be included, which can help to resolve any ambiguities that may arise after the syntactic analysis has been performed. Additional real-world pragmatic information can be used in conjunction with syntactic and semantic information to provide additional meaning.

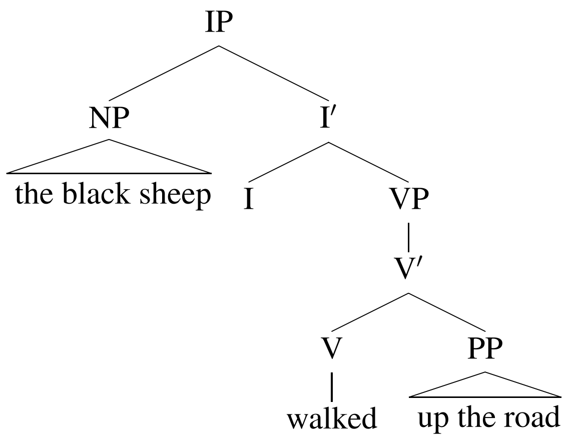


Figure 2.1: Example syntactic constituency parse

Figure 2.1, for example, shows a syntactic parse of the sentence *The black sheep walked up the road* in a typical constituent tree structure. However, the output of the parse only gives limited meaning to the utterance. It is not until semantic rules are applied that a meaning becomes more apparent. Pragmatic information, however, helps to develop the meaning even further by looking at

> "language from the point of view of the users, especially of the choices they make, the constraints they encounter in using language in social interaction, and the effects their use of language has on the other participants in an act of communication" (Crystal, 1997).

It is this type of contextualization that informs the reader or hearer of this utterance that *black sheep* might not necessarily mean a four-legged animal with black wool, but rather a wayward person. The idea that words can and do have meanings relating to concepts in the real world is something on which most information extraction systems do not deal with.

Conclusion

As we have just outlined, the information extraction research field is very broad and involves many researchers doing different types of projects with different goals. While information extraction tools and systems have been developed for a wide variety of domains, we have seen that a few tools have been created to extract information from medical literature. These tools, however, have not been used to process literature from medical clinical trials.

While the various components used in the system described in this thesis have already been developed independently by other researchers, there has been no collaboration to integrate the components together to create a robust predicate logic extraction system. This research focuses on how these tools have been integrated together, as well as the changes that had to be made to the system components in order to have a viable extraction tool.

The next chapter discusses the components used in the system and also describes why these components were chosen for this particular project.

Chapter 3

Method

As outlined in the previous chapter, information extraction systems are usually domain specific. The medical text domain is a very rich area that has not been fully explored by information extraction researchers. Most of the current work in medical information extraction has focused on extracting information from Medline abstracts. This thesis presents LG-Soar, a logical form identification and extraction tool that is capable of extracting predicate logic structures from another type of medical text, namely medical clinical trials. The process is outlined graphically in Figure 3.1. The rest of this chapter describes the steps the system goes through in order to accomplish the predicate logic extraction. This chapter also explains why the components used in this system were chosen and how they work together.

3.1 Clinical Trials

Clinical trials are used by medical professionals as a tool for recruiting patients to undergo new treatments or receive experimental medications in order to improve patient health. In 1997, the Food and Drug Administration (FDA) realized the importance of providing a systematic registry of clinical trials that could be accessed both by patients and providers and called for the development of such a system. The National Library of Medicine (NLM) and the National Institutes of Health (NIH) undertook the project and two years later unveiled an online repository of clinical trials (McCray, 2000). This repository of trials currently contains about 8,800 studies which are sponsored by various organizations including the NIH, other federal agencies, and private industries.[1] This repository

[1]http://www.clinicaltrials.gov

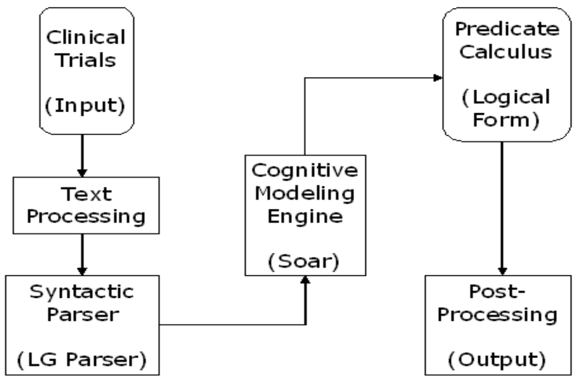

Clinical Trials (Input)
Text Processing
Syntactic Parser (LG Parser)
Cognitive Modeling Engine (Soar)
Predicate Calculus (Logical Form)
Post-Processing (Output)

Figure 3.1: Logical form identification process

of clinical trials receives about 3,000,000 page views per month.[2] With so much attention being given to ways to increase patient survival rates and decreasing the time it takes for experimental new drugs to reach patients, the need for improving and automating access to the information in clinical trials repositories is very important.

When providers have clinical trials to post on the website, they can log into a personal account where they have access to the forms used for uploading the clinical trial information to the repository. For the eligibility section, there is a text box provided which the providers use to enter in the criteria for their particular trial. No format restrictions are placed on how the information is entered. In addition to the text box for the eligibility criteria, there is a dropdown menu where providers can choose which gender the trial is intended for. Also, there are dropdown boxes and text boxes for information regarding the age of the patient for the clinical trial. The user interface a provider is presented with when entering eligibility information for a clinical trial can be seen at *http://prsinfo.clinicaltrials.gov/elig.html.*

Clinical trials contain a wealth of information about specific treatments or experimental drugs being tested to aid in the comfort and recovery of patients. Each trial located in the online repository is divided into a series of sections that contain specific information

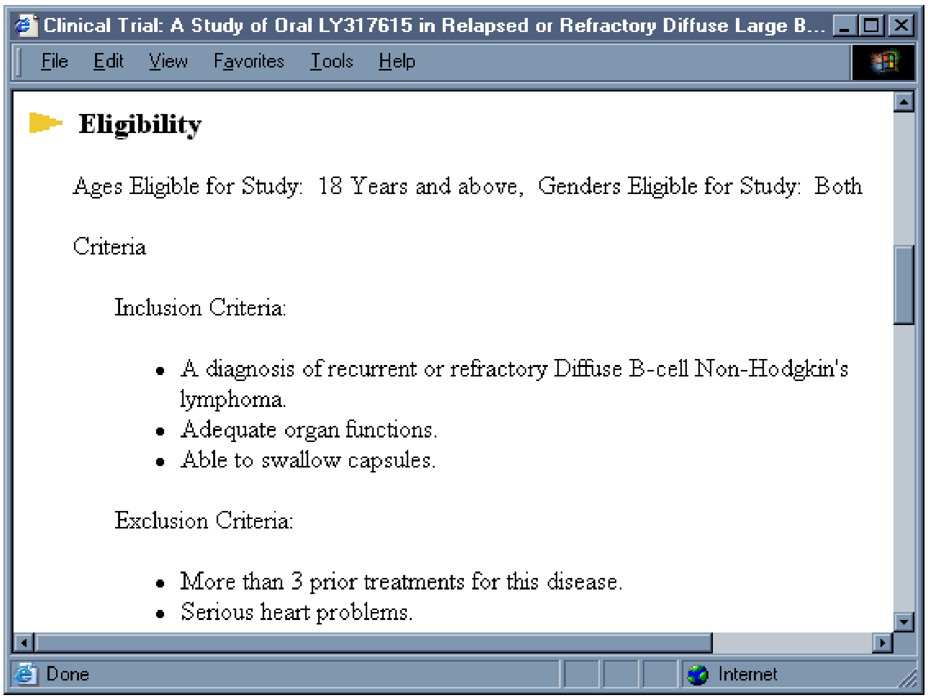
[2]http://www.clinicaltrials.gov/ct/info/about - Accessed July 2004.

```
Recruitment Status
Sponsor
Purpose
Description of the purpose of the trial
  Condition, intervention, phase (in table format)
  MEDLINEplus related topics
  Study type
  Official title
  Further study details
Eligibility
  Ages and sexes eligible for study
  Description of inclusion and exclusion criteria
  Location and contact information
  Names, addresses, telephone numbers, e-mail addresses
  Recruitment status at specified trial locations
More information
  Links to more information (e.g., related Web sites)
  Publications relevant to the study (if available)
  Study identification numbers (submitted by data providers)
  National Library of Medicine identifier (e.g., NCT00001789)
  Date study started
  Date recruitment status verified
  Date last updated
```

Figure 3.2: Information included in clinical trials

regarding the trial that is useful to providers and patients. Figure 3.2 shows a hierarchy of the different components mentioned in each individual clinical trial.

This thesis is concerned with the information located specifically in the Eligibility section of a clinical trial. As indicated by the name, and as shown in Figure 3.2, this section contains a listing of the requirements that a person must satisfy in order to participate in the trial. Depending on the trial, this section can contain eligibility criteria, ineligibility criteria or both. The eligibility criteria section lists requirements the patient must have or follow, and the ineligibility criteria section lists requirements that the patient must not have in order to be eligible for the trial. In addition to the eligibility and ineligibility criteria located in each trial, nearly every eligibility section includes information outlining both the age(s) a patient should have and also the gender necessary in order for the patient to be eligible. An example section of a web page containing clinical trial eligibility requirements is shown in Figure 3.3.


Clinical Trial: A Study of Oral LY317615 in Relapsed or Refractory Diffuse Large B...
File Edit View Favorites Tools Help
Done
Internet

Eligibility

Ages Eligible for Study: 18 Years and above, Genders Eligible for Study: Both

Criteria

Inclusion Criteria:

- A diagnosis of recurrent or refractory Diffuse B-cell Non-Hodgkin's lymphoma.
- Adequate organ functions.
- Able to swallow capsules.

Exclusion Criteria:

- More than 3 prior treatments for this disease.
- Serious heart problems.

Figure 3.3: a of clinical trial NCT00042666

A text preprocessing stage locates and extracts the criteria from the eligibility section of trial web pages and converts the criteria to XML. XML is used because it can be easily defined to represent the information in which we are interested. The criteria from the newly-formatted XML document is then used as input to the next phase of the process, the syntactic parser.

3.2 Syntactic Parser

The next step in the process involves using a syntactic parser to take the natural language criteria and produce a corresponding syntactic representation. In order to accomplish this step, we use the link grammar parser (Sleator and Temperley, 1991).

3.2.1 Link Grammar

We chose to use the link grammar parser (LG parser) for a variety of reasons. The off-the-shelf version of the parser comes already bundled with detailed information regarding English and the relationship of English words to each other. A dictionary file and a grammar file are both included in the distribution, which make it possible to modify the way the system runs natively.

The raw ability of the parser to robustly parse text is its biggest advantage. The parser can parse a wide variety of syntactic constructions, including many rare and idiomatic ones. The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and then assign some structure to the rest of the sentence. It is able to handle unknown vocabulary and make reasonable guesses from context and spelling about the syntactic categories of unknown words. This is important because many specialized words found in clinical trials are not in the parser's dictionary. Misspellings can also be processed. It also has knowledge of capitalization, numerical expressions, and a variety of punctuation symbols and can parse these as having relationships to other words in a sentence.

Another advantage of the LG parser is its speed. It is written in the C programming language and can run through high volumes of text without substantial delay. The parser also comes packaged with an API so it can be easily integrated with other applications and be freely downloaded for academic and research purposes.[3]

Finally, the LG parser has been used by a number of researchers to perform various natural language tasks where parsing incoming information is paramount. Some of these projects have included parsing news video subtitling content (Nakamura and Kanade, 1997), processing ambiguities in an answer extraction system (Molla and Hess, 2002), and understanding speech using prosody techniques (Hunt, 1994).

3.3 Syntax-to-Semantics Conversion

The ability to identify and output logic forms from text is a complicated task that involves more than the ability to parse a sentence. The information being parsed must also

[3]http://bobo.link.cs.cmu.edu/link

be understood. Some systems do this by filling in predetermined templates of information about the domain in which they are working. In essence, they do regular expression pattern matching on the text to find information they know already exists within the data. In a similar manner, our system needed a way to convert the syntactic representation generated from the syntactic parser to a more meaningful semantic representation.

The way this system does this is by using a syntax-to-semantics conversion tool. It uses a cognitive modeling architecture called Soar to translate the parsed sentence into logical form. This section will describe Soar in general and discuss some of features of the architecture that make it a desirable tool for this particular system.

3.3.1 Soar

The Soar architecture plays a vital role in this work. Soar is a model and theory of cognition which has the ability to model human cognitive processing (Newell, 1994). As an attempt to represent a unified theory of cognition, Soar goes beyond normal programming languages by having...

> ...embedded in it a specific theory of the appropriate primitives underlying symbolic reasoning, learning, planning, and other capabilities...necessary for intelligent behavior (Laird, 2003).

The goal of Soar researchers is to develop and model intelligent agents, and various applications have resulted. Projects include NL-Soar (Lewis, 1993) (a system to model how humans use natural language), Instructo-Soar (Huffman and Laird, 1995) (a system to model learning in humans), and Tac-Air-Soar (Jones et al., 1999) (an intelligent system for simulating military airborne missions). A wealth of information about Soar programs and related research, along with information on how to download the architecture free of charge is located at *http://sitemaker.umich.edu/soar*.

Because the goal of Soar is to model human cognition, certain characteristics must be available in Soar to model behavior. According to Newell, cognitive behavior, and therefore cognitive architectures, must exhibit the following six characteristics (Newell, 1994):

1. Be goal-directed
2. Reflect a rich detailed environment
3. Reflect a large amount of knowledge
4. Require the use of symbols and abstractions
5. Be flexible, and a function of the environment
6. Require learning from the environment and experience

The LG parser and the Soar cognitive modeling architecture have been integrated to form Link-Grammar Soar, or LG-Soar. These six characteristics are therefore present in LG-Soar and make it a system capable of translating natural language input to a semantic representation. A more detailed explanation of how these six characteristic fit within the Soar architecture and theory will not be discussed here. In the next chapter, we will discuss in detail how Soar is able to perform the syntax-to-semantics conversion for clinical trials.

3.4 Output Formats

Two different output formats were chosen for this particular project because of their ability to represent logical forms of text. One of the output formats is a Discourse Representation Structure and the other is first-order predicate calculus (FOPC) expressions. Discourse Representation Theory (DRT) is a formal linguistic theory for describing semantic and pragmatic relationships within single utterances or across utterances (Kamp and Reyle, 1993). FOPC is also an ideal formalism for representing semantic information. Both of these formalisms are state-of-the-art tools for representing semantic information. They can also be interchanged and their outputs can be translated between the two formalisms. The difference between the formalisms used in this project is their roles in terms of their output. Discourse representation structures (DRSs) are ideal for visually representing logical output while output represented as FOPC is better when additional processing needs to be done on the text.

Predicate calculus is used in many intelligent systems in order to provide a meaningful representation of language being fed into the system. In fact, the predicate logic produced by LG-Soar is actually used in another project that uses the logic form produced by LG-Soar to match clinical trial eligibility criteria with patients' medical records (Parker, 2003). This is just an example of a way that predicate logic generated from medical clinical trials could be useful in NLP-based biomedical applications.

Having sketched the major components of the system, we are now ready to describe these components along with the technical details of how they are all integrated with each other to produce a system capable of extracting predicate logic from text.

Chapter 4

System Description

The previous chapter outlined the method used for extracting predicate logic from clinical trials. This section provides a technical description of LG-Soar and the various components that comprise the system.

4.1 Clinical Trials Corpus

Since clinical trials were used as the input for this project, we developed a way to automatically crawl the clinical trial repository and download a number of trials to use as a corpus. The clinical trials repository uses a search page which anyone can use to access any of the more than 8,000 trials online. There are different ways that searching can be done. Trials can be located by trial number, disease/condition, trial location, or age group, for example. In order to get a broad sampling of trial criteria, we programmed our search to download all of the trials that have a Utah trial location. This totaled 246 different trials.

After we downloaded all of the trials, we generated a file containing all of the criteria from the 246 trials we downloaded previously. The number of criteria we pulled from the 246 trials totaled 4707. However, some of the criteria were duplicates. After removing duplicates, the total number of criteria in the corpus was 4566.

4.2 Pre-Processing

In order to run the clinical trials through the LG-Soar system, some preprocessing on the text must first take place. Three main steps need to happen before the text extracted from the HTML page of a clinical trial is ready to be sent through LG-Soar. First, the HTML file is converted to its corresponding XML. Next, the relevant information in each trial is tagged. Finally, some additional processing takes care of some problematic linguistic issues in regards to the criteria themselves.

4.2.1 HTML to XML Conversion

Because this thesis is only concerned with processing the information in the eligibility criteria section, the first step is to identify this specific section of each HTML document and separate it from the rest of the information on the page. In order to accomplish this, we used a pre-existing Python script which takes the HTML page, finds the relevant part of the document, and then converts it to XML.

The new XML file containing the clinical trial eligibility criteria has only three unique tags: `<criteria>`, `<criterion>`, and `<text>`. These tags identify the specific information that LG-Soar will be processing. This project is only concerned with part of the information located between the `<text>` and `</text>` tags. The important information is the natural language describing the actual criteria.

4.2.2 Tagging Relevant Information

Once the XML file has been generated from the original HTML, the next step consists of tagging each criterion. This is a necessary step because of the fact that some of the information contained between the `<text>` and `</text>` tags is not criteria. This non-criterion information gives additional information regarding the criterion, but it will not be sent on to be processed by the rest of the system. For example, this information might mention whether the criterion is an inclusion or exclusion criterion or whether it relates to the patient or the disease. In order to determine which information is necessary to process and send on, a Perl script is called that looks for patterns to avoid. Since most of the patterns are somewhat consistent across trials, only a handful of patterns are needed to look for the correct structures. A numerical tag is then appended onto the tag containing the criteria as shown in Figure 4.1.

```
<criteria trial="http://www.clinicaltrials.gov/ct/show/NCT00042666">
 <criterion>
  <text>Eligibility</text>
  <text val="1">Ages Eligible for Study: 18 Years and above,</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text val="2">Genders Eligible for Study: Both</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Inclusion Criteria:</text>
  <text val="3">A diagnosis of recurrent or refractory Diffuse
              B-cell Non-Hodgkin’s lymphoma.</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Inclusion Criteria:</text>
  <text val="4">Adequate organ functions.</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Inclusion Criteria:</text>
  <text val="5">Able to swallow capsules.</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Exclusion Criteria:</text>
  <text val="6">More than 3 prior treatments for this disease.</text>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Exclusion Criteria:</text>
  <text val="7">Serious heart problems.</text>
 </criterion>
</criteria>
```

Figure 4.1: Tagged XML file of clinical trial NCT00042666

4.2.3 Dummy Subject Addition

After converting the trial to XML and tagging the criteria, the actual natural language processing starts and is what this thesis is most concerned with. This final stage of preprocessing manipulates the natural language text of the criterion so that it can be processed by the rest of the system. Another Perl script performs this processing, as well as some additional formatting which will be described shortly.

Because of the components used in LG-Soar, the text input into the system must follow certain conventions. The most important one is the requirement that the text form a complete sentence, where the term *complete* can for now be loosely defined as a subject followed by a verb, with other constituents.

The reason this constraint must be satisfied is because the LG parser, which will be discussed in the following section, requires it. While it is possible to change this requirement by manipulating the grammar of the parser, for this project it proved to be impractical and unnecessary. Most of the clinical trials in the registry contain criteria that are complex noun phrases. Out of the more than 8,000 trials currently online, only a small percentage have criteria structured as complete sentences.

To correctly handle these complex noun phrases and convert them into complete sentences, a *dummy* subject and verb were prepended to each criterion. *A criterion equals* was the dummy material we chose to add.

Not all of the criteria in the clinical trials are complex noun phrases; some are actual sentences or other structures. Automatically prepending a subject and verb to each criterion makes these structures ungrammatical for the LG parser. As such, not all of the criteria can be processed; however, most of the criteria can.

Two situations arise where criteria do not consist of complex noun phrases. These are the gender and age criteria. As was mentioned previously, gender and age information in clinical trials is entered separately from other criteria. For this reason, age and gender data is usually formatted similarly across clinical trials. We did some additional Perl processing to convert these structures to sentences that could be read into the LG parser. The results of prepending the dummy subject and verb and processing the gender and age information can be seen below:

1. *A criterion equals* `an age greater than 18 years.`
2. *A criterion equals* `both genders.`
3. *A criterion equals* `a diagnosis of recurrent or refractory Diffuse B-Cell Non-Hodgkin's lymphoma.`
4. *A criterion equals* `adequate organ functions.`
5. *A criterion equals* `an ability to swallow capsules.`
6. *A criterion equals* `more than 3 prior treatments for this disease.`
7. *A criterion equals* `serious heart problems.`

Additionally, the capitalization of the initial word in each criterion was converted to lower case while keeping acronyms intact.

Adding a dummy subject and verb were not the only things we did to manipulate the text before being input into the next phase of the system. Another area where we decided to change the data were instances where the first word in the criterion could be nominalized in order to produce a grammatical reading from the parse. So for instance, the criterion *able to swallow capsules* would be rendered as *an ability to swallow capsules* so that when the dummy subject and verb are added, the sentence could be parsed correctly. While this is an ad hoc approach, there are significant instances where this happens and it increases the performance of the system.

4.3 Link Grammar Parser

Once the HTML of the clinical trial has been converted to individual sentences, they are ready to be processed by the link grammar parser. The system reads in a .txt file containing each criterion on a separate line in the file and parses each sentence individually. Then the sentence is analyzed by the syntax-to-semantics conversion engine (Soar) before going on to the next sentence.

The parser's role in this is to produce a syntactic representation for each of the sentences. Appropriate links are attached to each of the constituents in the sentence and the links are drawn, followed by the syntactic output. Because English is highly ambiguous, a single input sentence has the potential of having multiple parses or readings. However, in this project, we do not consider additional readings and instead use the first parse provided by the parser which usually (but not always) reflects the correct syntactic structure of the

```
    +----------------------------Xp---------------------------+
    |                       +-------------Op------------+     |
    +-----Wd-----+          |        +---------A--------+     |
    |     +--Ds--+----Ss----+        |        +----AN---+     |
    |     |      |          |        |        |         |     |
LEFT-WALL a criterion.n equals.v serious.a heart.n problems.n .

       LEFT-WALL      Xp      <---Xp---->  Xp         .
(m)    LEFT-WALL      Wd      <---Wd---->  Wd         criterion.n
(m)    a              Ds      <---Ds---->  Ds         criterion.n
(m)    criterion.n    Ss      <---Ss---->  Ss         equals.v
(m)    equals.v       O       <---Op---->  Op         problems.n
(m)    serious.a      A       <---A----->  A          problems.n
(m)    heart.n        AN      <---AN---->  AN         problems.n
       .              RW      <---RW---->  RW         RIGHT-WALL
```

Figure 4.2: Link grammar output for *A criterion equals serious heart problems*

natural language input. The link grammar parser does come with the ability to change how parses are scored, but such changes were not implemented in the work described in this thesis.

The link grammar parser has a dictionary of about 60,000 word forms each divided up into one of 50 different files based on four part-of-speech categories (adjectives, adverbs, nouns, and verbs). Each part of speech is further broken down into different categories relating to that part of speech. For example, there are three different adjective files, one containing *-est* adjectives, one containing *-er* adjectives, and the other containing miscellaneous types.

In order for the parser to determine how to parse an utterance, it must first determine the grammaticality of the input sentence. A sequence of words is considered a sentence if three conditions are met. These constraints are summarized below:

1. PLANARITY: Links cannot cross.
2. CONNECTIVITY: Links must indirectly connect all the words together.
3. SATISFACTION: Correct links must be used to connect the words together.

A parse of a grammatical sentence is shown in Figure 4.2.

The first noticeable difference between the grammatical parse in Figure 4.2 (which represents the link grammar formalism) and the grammatical parse in Figure 2.1 (which

Table 4.1: Major link types and sublinkages

Link	**Major link role**	**Sublinkage role**
Xp	connects punctuation to words	connects the period at the end of sentence
Ds	connects determiners to nouns	number agreement on determiner (singular)
Ss	connects subject-nouns to finite verbs	noun agreement (singular)
Os	connects transitive verbs with their objects	verb agreement (singular)
Wd	connects main clauses to the left wall	used specifically in declarative sentences
A	connects attributive adjectives to nouns	none in this instance
AN	connects noun modifiers with nouns	none in this instance

represents a more traditional linguistic tree parse), is the structure. The link grammar parse is a more linear or flat structure, while the traditional linguistic parse depicts relationships in a more hierarchal fashion.

The other difference is the actual nodes or links that connect all of the words in the sentence together. In the traditional approach shown in Figure 2.1, only a handful of nodes are used. However, in the link grammar formalism, there are a total of 107 major linkage categories, and each of these categories can have related sublinkages. These linkages provide the information for understanding how words relate to one another.

When we undertook this project, LG-Soar could only derive a semantic representation from a few of the links provided by the LG parser. In order to increase the coverage of the system, we programmed LG-Soar to interpret the logical form for a wide range of different links, such as those corresponding to prepositional phrases, relative clauses, adjectives, adverbs, negation, modals, nominal compounds, imperatives, and conjunctions.

In Figure 4.2, there are a total of seven major link types: X, W, D, S, O, A, AN. Each of these links contains additional information in the form of sublinkages (represented by the lowercase letters). Table 4.1 shows how the links encode information about the sentence.

Figure 4.2 shows how a parse of *A criterion equals serious heart problems* would be represented syntactically by the LG parser. Different links connect the words in the sentence. These links are the key to extracting the semantic meaning from the syntactic output. LG-Soar is able to look at the listing for each word link pair and its label to gather the semantic information. The next stage consists of using a syntax-to-shallow-semantics engine to determine the semantic meaning.

4.4 Syntax-to-Semantics Engine

Soar is a production-driven system which uses productions to tell the program about the state of the world and the knowledge it has about the way that world functions. Productions are a series of if-then rules used to determine the condition of the world and whether or not actions can take place in relation to those conditions. The conditions on the left-hand side (LHS) of the rule represent the *if* part of the rule, while the actions on the right-hand side (RHS) represent the *then* part. Conditions test for the presence or absence of data stored in the knowledge base of the system. If all the conditions are properly met, the rule fires and the actions on the RHS of the production execute. Actions can add or remove structures from working memory, perform miscellaneous functions like write output to the screen, or call additional procedures written in Tcl (Tool Command Language) to perform other actions.

4.4.1 Knowledge Representation in Soar

While Soar productions perform actions that change the state of the world, another element encodes knowledge about the world that the productions must access in order to understand how the world is put together. This component is called working memory and encodes necessary information about the world, such as the present state or conditions of the world and objects within that world, as well as future goals the program might have. The way Soar structures this information is by using states. States contain relevant knowledge about the current condition of the objects within the world in which Soar is operating. By having this knowledge accessible, working memory can allow productions to change the state of the world in accordance with goals and subgoals the system might have.

4.4.2 Soar Processing in LG-Soar

This next section will describe how Soar is used to convert the output produced by the LG parser to a logical representation of the semantics of the input criteria. Soar goes through two stages when accomplishing this task. The first stage consists of defining the semantic relationships that exist between the different words, and the next stage consists of looking for specific relationships that exist and producing the appropriate logical form for those relationships. While most of the work done for this thesis focused on developing the second phase, both phases will be explained.

Defining Relationships Using Soar

As mentioned above, the first step in the Soar process is to identify the relationships that exist from the output given by the LG parser and was already in place for this project (Lonsdale et al., 2001). One of the differences mentioned earlier between dependency grammar and link grammar is that the latter does not identify a root node or word within the utterance. This has implications for doing semantic analysis because in order to derive a semantic meaning, the root of the utterance needs to be identified. Once this word is identified, relationships of other words to that word can be identified and semantic meaning can be determined. Since the LG parser does not do this, it was necessary to encode the information in Soar to be able to accomplish this task. The way that Soar does this is by looking at the links generated by the LG parser. Knowing what those links stand for, and which constituents can combine together, Soar can then use productions that encode the knowledge about those links and determine the semantic relationships.

The Soar phase that takes care of this step consists of nearly 150 Soar productions that identify link/constituent positions and convert them to semantic equivalents.

Table 4.2 outlines the first 10 steps LG-Soar goes through for the sentence *A criterion equals serious heart problems* once the link grammar output has been defined and passed on. This step is the beginning of defining the semantic representation of the sentence. After Phase 1 is complete, then comes Phase 2, which will be described next.

Table 4.2: Initial trace of syntax-to-semantics conversion by LG-Soar

Operator Type	Description
1: O2 (find-root)	This production finds a root word for the sentence In this first case, the root is set to LEFT-WALL.
2: O4 (find-root)	The same operator fires but this time another root is found, this time the word *x*.
3: O5 (give-root-ref)	This gives the root word a reference, which is the word *equals*.
4: O6 (add-center)	This recognizes that the word *equals* needs to be treated as a verb type.
5: O8 (add-arg)	This recognizes the *S* link from the link grammar parser and proposes that it treat the word attached to it *(criterion)* as the external argument of the sentence.
6: O10 (add-center)	This is similar to the previous add-center operator except that instead of recognizing *equals* as a verb type, it recognizes the word *criterion* as being a third person type.
7: O12 (add-arg)	This recognizes that the word *x* has an argument which can be described as being a definite description *(an)*.
8: O9 (add-arg)	This recognizes the *O* link from the link grammar output and proposes that the word attached to this link (in this case the word is *problems*) be treated as the internal argument for the sentence.
9: O13 (add-center)	This is the same as the previous add-center operator except that it recognizes the object of the sentence, *problems*, as being a third person type.
10: O15 (add-arg)	This recognizes the *A* link from the output generated by the link grammar parser and proposes that it be treated as an adjective modifying a noun, which in this case is *serious* (the adjective) modifying *problems* (the noun).

Outputting Logical Form

As pointed out previously, two phases are necessary when converting link grammar syntactic output to a semantic representation. The first phase was outlined briefly above, and it consists of translating the links into a semantic equivalent. This includes such actions as finding appropriate roots in the sentence and identifying various constituents such as the internal and external arguments.

The next phase in the process is the phase that actually outputs the appropriate logical form for the input sentence, or criterion in this case. Soar productions look for ways sentences are structured in the Soar world or model, and decide if the appropriate conditions are met so that actions which output the right constituents are executed. The Phase 1 output (which is also the starting point for Phase 2) is a state that represents how the world is structured. In order for this information to be processed, however, additional Soar productions are proposed and fired that take the information existing in the state and decide how to best output it logically. Phase 2 consists of over 150 Soar productions. These productions test to see whether certain structures are present in the state or knowledge of the system and then translate these to logical output. Most of the information that is relevant to the structure of the text is located in the `^model` attribute of the identifier S1. The system has a model that outlines the various components that make up the core knowledge of the sentence. The model contains several individual ideas that are each assigned variable names. In this case, the variable names are N2, N3, N4, N5, and N6. These ideas are the main ideas of the sentence and do not include constituents such as determiners or prepositions.

Each idea contains information important to the knowledge of the system. For example, the predicate *problems* is assigned the variable name *N6*, which is then converted to the predicate statement *problems(N6)*. In another example, the predicate *equals* is given the variable name *N3*. Also, based on the information given to the system by the model, we can see that *equals* has two arguments. An external argument and an internal argument are on the model and provide a reference to *equals*. This information is then converted to a predicate argument structure, or *equals(N2,N6)*, where *N2* is the variable name for the predicate *criterion*.

Table 4.3: Operators for *A criterion equals serious heart problems*

Number	Production Type	Output
1	adj-noun-noun*seq	No output
2	adj-noun-noun*arg	No output
3	feature-feature	indef(N2)
4	compound-adj-noun-noun	serious(N6) & heart_problems(N6); criterion(N2)
5	ext-nuc-int	equals(N2,N6)
6	processing-complete	No output

Table 4.4: Productions used for *A criterion equals serious heart problems*

Number	Description
1	This finds each of the ideas on the model and puts an augmentation on the idea stating that it has been recognized so further processing can take place.
2	This is similar to the previous one except that it organizes the ideas that have been recognized by the previous production.
3	This outputs the predicate-argument structure of indefinite and definite modifiers attached to nouns.
4	This outputs the predicate-argument structure corresponding to sentence structures consisting of an adjective followed by two nouns, as in *serious heart problems*. It also outputs the predicate-argument structure for the external argument *x*.
5	This production outputs the predicate-argument structure for the external and internal arguments.
6	This production verifies that all the ideas in the model have been recognized. It then calls the correct output mechanism which displays the output accordingly (either as a DRS in CLIG or text in an .xml file.

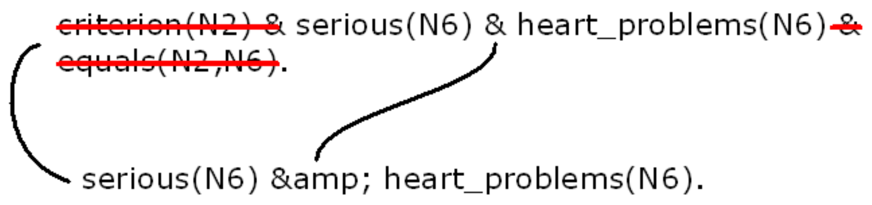

~~criterion(N2) &~~ serious(N6) & heart_problems(N6) ~~&~~
~~equals(N2,N6)~~.
serious(N6) &amp; heart_problems(N6).

Figure 4.3: LG-Soar post-processing

Once the state has been created and the model contains the information describing the relationships between the ideas in a sentence, the next step consists of calling Soar productions which take that information and output it in a logical form.

In *A criterion equals serious heart problems*, each of the productions in Table 4.4 fires because the state has conditions that match against what the productions are looking for. Once the processing-complete operator is reached, the logical form that is output is ready to be put through the post-processor.

4.4.3 Post-Processor

The main purpose of the post-processor in the LG-Soar system is to clean up the output that has been generated by the Soar engine. As was outlined earlier, some pre-processing takes place when a criterion is read into the system. The pre-processing adds a dummy subject and verb to each criteria which makes it possible for the criterion to be parsed by the link grammar parser. However, because the dummy subject and verb are not actually part of the criterion extracted from the clinical trial, it is extraneous information that needs to be taken out. The post-processor removes the extraneous information.

In order to do the necessary cleanup, we used Prolog. Prolog is a logical and declarative programming language that is widely used in artificial intelligence programs and contains facts and rules about a system's knowledge. We programmed a series of Prolog axioms to do some natural language processing on the output generated by LG-Soar up to this point. These axioms remove the dummy subject and verb from each criterion, eliminate any redundancies that occur, and filter out any irrelevant data as shown in 4.3.

The subject and verb are removed along with their corresponding arguments because they do not provide any additional information about the criteria and were in fact not even in the original criteria from the clinical trial. Other changes, such as the fact that the `&` is changed to `&amp;` occur in order to produce a valid XML document.

4.5 Output Formats

Once the post-processor has been called, the criteria are finally ready to be output. The logical form output can be either output visually or to a file and in order to accomplish this, we had to do a few things with the system. We wrote additional Soar productions which take the internal structure of the Soar model and output the corresponding information to a display type of our choosing. We also integrated the CLIG system with LG-Soar to be able to visualize the output graphically.

4.5.1 Discourse Representation Structures

One possible output format for the system is Discourse Representation Structures (DRSs). When DRSs need to be output, we use CLIG, the Computational Linguistics Interactive Grapher. This is a program that was designed to represent various types of linguistic structures (Konrad, 1995). It can be readily integrated into other programs where different linguistic structures can be represented and viewed. The grapher can display X-bar trees, DRSs, feature-value structures, or any combination of these. Users can also add interactive hyperlinks and buttons to the output. We had to integrate CLIG with the rest of the system in order to use it for graphically outputting DRSs.

While CLIG is not ideally formatted for all computational applications, it does have its benefits and uses. First, it is easy to see the representation of the parsed sentence, and hence is beneficial for testing and debugging. When a sentence does not parse correctly, it is easy to see where the parse went wrong. The incorrect parse can then be tracked back to where the error occurs in the LG-Soar code. When newer syntactic structures are programmed in LG-Soar, CLIG is useful to see how the current system treats them, thereby testing what needs to be done to correct the representation. The DRS representing *A criterion equals serious heart problems* can be seen in Figure 4.4.

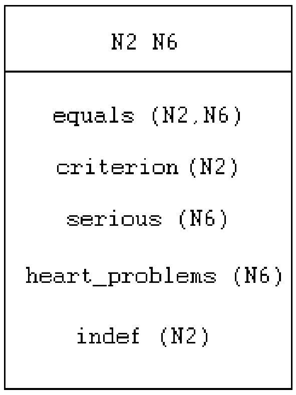


Figure 4.4: DRS for *A criterion equals serious heart problems*

4.5.2 XML

The next type of output used in this project is XML. The goal of this project is to convert eligibility criteria to a predicate logic form. This output can then be used as input to other systems that want to access the information found in the clinical trial. In order to use the output, a medium of exchange must be used that will work for everybody needing to access the data. XML is this medium of exchange.

When the XML file that is generated from the HTML is created, it puts the necessary criteria in between `<text>` tags. This information is then sent to a .txt file which can be read into the LG-Soar system. After LG-Soar creates the predicates for each of the incoming criteria, the system uses another Perl file to merge the newly-created predicates back into the XML file surrounded by `<pred>` tags which are located just under the corresponding criteria. Figure 4.5 shows part of a final XML file. This file contains both the natural language of the criteria as well as the predicate output. It is then ready to be used as input for any other system needing that information.

4.6 Summary

Because we used many different components in this system, considerable integration was necessary to process the incoming text. We used various programming languages such as Perl, Prolog, Tcl, and Python to get the components to work together.

```
<criteria trial="http://www.clinicaltrials.gov/ct/show/NCT00042666">
 <criterion>
  <text>Eligibility</text>
  <text val="1">Ages Eligible for Study: 18 Years and above,</text>
  <pred val="1">age(N4) &amp; quantification(N5,greater_than)
                      &amp; measurement(N4,N5) &amp; units(N5,years)
                      &amp; magnitude(N5,18)</pred>
 </criterion>
 <criterion>
  <text>Eligibility</text>
  <text val="2">Genders Eligible for Study: Both</text>
  <pred val="2">both_genders(N4)</pred>
 </criterion>
... (ADDITIONAL CRITERIA) ...
 <criterion>
  <text>Eligibility</text>
  <text>Criteria</text>
  <text>Exclusion Criteria:</text>
  <text val="7">Serious heart problems.</text>
  <pred val="7">serious(N6) &amp; heart_problems(N6)</pred>
 </criterion>
</criteria>
```

Figure 4.5: Final XML output

The work described is significant for a variety of reasons. First of all, we improved the Soar/Link-Grammar parser interface by allowing a file containing one sentence per line to be read in with a single command. However, no more than one sentence per line is allowed. The first version of the system only allowed one sentence at a time to be input at the command line. We also created various Perl scripts which did some text manipulation on the clinical trials. One script extracts the actual criteria from an XML file so they can be sent to the LG parser while another script numbers the criteria in the XML file. Yet another script merges the criteria with corresponding predicate logic structures into an XML file. Another significant contribution was that we changed the grammar file for the LG parser so that more syntactic constructions could be parsed correctly. This improved the final results that we obtained. We also wrote numerous Tcl procedures which send Soar information to CLIG so that predicate argument structures can be formatted correctly and we programmed various Prolog axioms to filter out unimportant information from the structures generated by LG-Soar. In order to tie the entire system together, we automated the predicate-extraction process so that the XML file corresponding to a clinical trial can be input into the system and the corresponding predicate argument structures of each criterion in the trial can be output. This whole process can be accomplished with a single command. Finally, we evaluated the system with a new evaluation program written by Vasile Rus. This evaluation program is specifically intended for evaluating precision and recall of predicates and arguments output by a logical form extraction system and we directly applied it to medical clinical trial information for the first time ever with this project.

Chapter 5

Results

This chapter reports on the metrics used to evaluate LG-Soar as an IE tool. We explain the metrics used to evaluate the output generated by the system and report on the actual results obtained from running the system.

5.1 Evaluation Metrics

Information extraction is a broad field where researchers are developing different applications to perform various activities. Even though the applications vary, most IE systems are evaluated using three measurements: precision, recall, and F-measure.

5.1.1 Precision and Recall

Precision and recall measurements are best determined by using a counting technique referred to as a confusion matrix. Table 5.1 shows a confusion matrix.

When an entity is extracted, we have to determine whether it is correct. The extracted entities that are correct are the true positives; the incorrectly extracted entities are the false positives; the entities that are not extracted but should have been are the false

Table 5.1: Confusion Matrix

Predication		
	+	-
answer `[+]`	true positive	false negative
answer `[-]`	false positive	true negative

negatives. True negatives are the entities that are not extracted and should not have been. Generally, true negatives are not used in information extraction system evaluations.

From this matrix, precision and recall can be measured. In short, precision measures the correctness of the output, while recall measures the completeness of the output. According to Hobbs,

> When you promise to tell the whole truth, you are promising 100% recall. When you promise to tell nothing but the truth, you are promising 100% precision (Hobbs, 2002).

When using these evaluation metrics for logic form identification and extraction, they are measured on two separate levels: the predicate level and the argument level. The way these are determined are outlined below.

Argument precision:

$$\frac{No.\ Correctly\ Identified\ Arguments}{No.\ All\ Identified\ Arguments} \tag{5.1}$$

Predicate precision:

$$\frac{No.\ Fully\ Identified\ Predicates}{No.\ All\ Identified\ Predicates} \tag{5.2}$$

Argument recall:

$$\frac{No.\ Fully\ Identified\ Arguments}{No.\ Arguments\ To\ Be\ Identified} \tag{5.3}$$

Argument precision:

$$\frac{No.\ Correctly\ Identified\ Predicates}{No.\ All\ Predicates\ To\ Be\ Identified} \tag{5.4}$$

5.1.2 F-Measure

Another measure that has gained popularity among researchers evaluating information extraction systems is the F-measure. This metric takes both precision and recall into

Table 5.2: Sample LG-Soar output

Original Source	**LG-Soar Output**
Adenocarcinoma of the pancreas	adenocarcinoma(x) & of(x,y) & pancreas(y)
Brain metastasis	brain_metastasis(x)
Femoral neck osteoporosis	femoral(x) & neck_osteoporosis(x)
Pregnancy	pregnancy(x)
High risk of VTE	VTE(x) & risk(y) & of(y,x) & high(y)
Controlled COPD	controlled(x) & COPD(x)
Genders eligible for Study: Female	female_gender(x)

consideration to determine an overall score of a system by calculating the harmonic mean of the two measurements (van Rijsbergen, 1979).

$$F = \frac{2 * (recall * precision)}{recall + precision} \tag{5.5}$$

5.2 Qualitative Results

The LG-Soar system outputs predicate logic structures. Table 5.2 shows some interesting output results from the clinical trial corpus. Appendix A shows three examples of criteria and the process they go through in order to be output correctly, while Appendix B shows some results of criteria that for various reasons were not output correctly by the system.

5.3 Quantitative Results

In order to compute the results of the LG-Soar system, we used a logical form identification evaluation system developed by organizers of the Senseval '03 logic form identification task. Senseval is an organization committed to furthering research in language and computation by organizing a series of tasks available to researchers worldwide. The purpose of these competitions is to foster more interest and eventually more success in dealing with natural language problems computationally. This past year was the first year a logical form identification event has been scheduled. The organizers developed an

Table 5.3: Initial LG-Soar quantitative results

	Precision	Recall	F-Measure
Argument	70%	61%	65%
Predicate	65%	63%	64%

evaluation program written in C that anyone can use who is developing logical form identification tools.[1] The evaluation tool takes as input a gold form file (which contains the correct handcoded predicate/argument structures) and a logic form file (which contains the output generated by a particular system). The results which are returned consist of both predicate and argument precision and recall.

In order to test our system, we randomly selected twelve different clinical trials from the government-sponsored online repository. From those twelve trials, 102 criteria were extracted and after exact duplicates were removed, a total of 77 criteria remained.

Out of the 77 criteria available to run through the system, only about half of them were able to be parsed by the LG parser. For reasons of ungrammaticality, the other half of the critieria were not parsed and thus not sent on to the rest of the system. After running these 34 criteria through the system, we achieved the results outlined in Table 5.3. As can be seen, precision and recall measures for both the predicates and arguments hover between 60-70%. To relate those numbers to the definitions given above for precision and recall, we can see that 70% of the arguments and 65% of the predicates were accounted for exactly, while 65% of the arguments and 64% of the predicates had at least some of the right information.

In the next chapter we will discuss how these initial results compare to other researchers attempting to extract different relationships from biomedical literature.

[1]http://www.cs.iusb.edu/vasile/logic/indexLF.html

Chapter 6

Discussion

In this section we will discuss some of the implications for the results that were obtained when running LG-Soar on medical clinical trials.

As was mentioned earlier, information extraction is a widely researched field and nearly every domain of information is unique in terms of what the extracted information consists of. These differences often make it difficult to compare different tools across domains even though similar evaluation metrics are used (i.e. precision and recall). So for example, the success rate of name extraction from newspaper articles currently hovers around 90%. However, relation extraction is lower around 80%, and event extraction hovers around 60% (Hirschman, 2002). While the current LG-Soar system is not at the level that name extraction is, Table 6.1, which is adapted from Pustejovksy (Pustejovsky et al., 2002), shows that it is very comparable to other relational or event extraction systems.

Another interesting finding that occurred when we compared our results with other research in the biological domain was that typical medical extraction systems tend to have high precision values but low recall values. The reason this is so is because most other

Table 6.1: Comparison of relation capture for biology

Source	Relation	DB	Prec	Recall
Craven '99	location	Yeast	92%	21%
Rindflesch '00	binding	MEDLINE	73%	51%
Proux '00	interact	Flybase	81%	44%
Pustejovsky '02	inhibit	MEDLINE	90%	57%
Tustison '04	**argument**	**Clinical Trials**	**70**%	**61**%
Tustison '04	**predicate**	**Clinical Trials**	**65**%	**63**%

researchers doing extraction on biomedical literature are looking for specific relationships. When the relationship is found, they can usually find the arguments associated with that relation. However, if they do not find that relation initially, they have a more difficult time finding the arguments associated with it. For this reason these other systems tend to have higher precision values but lower recall values. With LG-Soar, the relationships we are looking for are more general. We are looking for all of the predicate and argument relationships that exist in a trial. Precision tends to be lower because there are more relationships to find; however, recall is higher because of the fact that LG-Soar can usually find at least part of the relationship, whereas other approaches are more hit and miss in what they do.

6.1 Benefits of LG-Soar

While many improvements could still be made to the system to make it more robust, there are many benefits the system has that make it a worthwhile tool when doing predicate logic identification and extraction. First, the components which make up the system are all freely downloadable for research purposes, and they can be integrated together in ways to create interesting applications. Also, the system is not domain specific. While this thesis has focused on its use in the medical domain, we have tested it with other text information such as newspaper headlines and genealogical data (Tustison, 2004). In order to use other types of text, relatively minor changes would be needed to the overall system and, depending on the types of structures prevalent in the new input, a few new productions would have to be written. This makes LG-Soar a good tool to use when extracting predicates from text.

6.2 Future Work

Initial results obtained from running LG-Soar on medical clinical trials show that it is a promising approach to the emerging field of logical form identification. However, more research on the project could invariably produce better results and provide for some more interesting discussion. There are a few areas of research on the LG-Soar system that would be beneficial to the system overall.

Early on in the discussion we mentioned that in order to send most of the criteria to the LG parser, a dummy subject and verb had to be added to each one. Well, because eligibility criteria can be formatted any way the provider entering the data wants, there are times that the criteria are complete sentences and not noun phrases. As can be imagined, this will cause problems for the parser and the sentence will not parse because the dummy subject and verb will be added to a sentence that already has a subject and verb. By adding an additional pre-processing step that would do some analysis on the incoming criteria to determine if it is already a complete sentence, the number of criteria that could be parsed would increase, thereby increasing the performance of the system.

Another area that could use improvement is the LG parser itself. While we did make some changes to the parser when implementing it in our system, it remains virtually untouched in how it performs when downloaded from the Internet. One way that the parser could be improved is by improving the grammar file of the parser. The best approach to doing this would be to conduct a more in-depth corpus analysis on the criteria in the medical trials to determine the types of structures that are currently not able to be handled by the parser or are handled incorrectly. A far-reaching example is units. Units are an understudied area of linguistics and many times the parser labels a unit or a unit value pair with an incorrect link. This has further implications when the parse is passed to the Soar engine because the wrong link will produce an incorrect predicate argument relationship.

Two minor changes that could be added to increase the robustness of the parser would be to look at the scoring algorithm used by the parser and determine if it could be improved, and the other change would be to add medical vocabulary to the parser's dictionary. Both of these changes, however, would have a small impact on improving performance of the system because of the parser's ability to already score parses fairly correctly and also because the parser is able to robustly handle unknown words that might be fed in.

Along with improving the grammar file of the parser, the Soar engine could be improved by increasing the number of syntactic structures it can correctly handle. Also, it would be interesting to turn on learning in Soar to see how well it can recognize similar structures coming in to the system. A final area of future work would be to integrate the system with semantic web agents to see if they could work in conjunction to crawl for information about clinical trials and eligibility criteria.

Chapter 7

Conclusions

This thesis has described LG-Soar, a system capable of extracting predicate logic structures from medical clinical trials. While the importance of predicate logic in intelligent systems has been known for quite some time, only recently have researchers started to combine many of the information extraction and natural language processing tools and resources together in order to develop systems to automatically do this.

By automating the process, new programs can be developed that will have intelligent capabilities. In this thesis we have described a system we have created that that can take semi-structured clinical trials and produce a corresponding predicate logic representation. This output in particular is currently being used as input for an intelligent system that automatically matches patients with clinical trials (Parker, 2003). The predicate logic outputted by LG-Soar gets matched up with similar concepts in a medical database. These concepts can then be compared to concepts found in a particular patient's medical record. The process will help speed up the time it takes to recruit patients for clinical trials because the process can be done automatically. Doctors will no longer have to rely solely on face-to-face meetings with a patient to find out whether or not the patient is eligible for a certain clinical trial. Using predicate logic as input as described above is just one way it can be used to create systems that can make intelligent decisions about different situations.

While LG-Soar is by no means the final answer concerning the question of how to best go about extracting logical structures, it does present an interesting solution to the problem. Due to the fact that predicate logic identification research is relatively new, there are no traditional approaches. In order to understand how to best go about the process of automatically extracting predicates, researchers need to present different ideas and methods to the information extraction community as a whole in order to stimulate further research and comparisons with other approaches. As more and more researchers become interested in this problem, and as more conferences and discussion take place which are devoted to uncovering solutions, not only will overall results improve, but standards will develop that will help determine how these systems can be better evaluated. If this thesis has helped to stimulate further discussion on different approaches to the problem of logical form identification and extraction, it will have been a success.

Bibliography

Baud, R., Lovis, C., Rassinoux, A.-M., and Scherrer, J.-R. (1998). Morpho-semantic parsing of medical expressions. In *Proceedings of the AMIA Annual Symposium*, pages 760–764, Orlando, FL, USA. Hanley & Belfus.

Califf, M. E. and Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, Menlo Park, CA, USA. AAAI.

Collins, M. (1996). A new statistical based parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Crystal, D. (1997). *A Dictionary of Linguistics and Phonetics*. Blackwell, Oxford, UK.

Daelemans, W., Zavrel, J., van der Sloot, K., and van den Bosch, A. (1999). TiMBL: Tilburg Memory Based Learner, version 2.0, reference manual. Technical Report ILK-9901, Tilburg University, Netherlands: ILK Research Group.

Embley, D., Campbell, E., Jiang, Y., Liddle, S., Lonsdale, D., Ng, Y.-K., and Smith, R. (1999). Conceptual-model-based data extraction from multiple-record web documents. In *Data and Knowledge Engineering*, pages 227–251.

Freitag, D. (1998). Information extraction from HTML: application of a general machine learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 517–523, Menlo Park, CA, USA. AAAI.

Hirschman, L. (2002). *Natural language processing for biology*. MITRE Corporation.

Hobbs, J. R. (2002). Information extraction from biomedical text. *Journal of Biomedical Informatics*, 35:260–264.

Huffman, S. B. and Laird, J. E. (1995). Flexibly instructable agents. *Journal of Artificial Intelligence Research*, 3:271–324.

Hunt, A. J. (1994). Improving speech understanding through integration of prosody and syntax. In *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*, pages 442–449.

Jones, D. (1996). *Analogical Natural Language Processing. Studies in Computational Linguistics*. University College London Press.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P. G., and Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1):27–41.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Konrad, K. (1995). The CLIG grapher for linguistic data structures. http://www.ags.uni-sb.de/ konrad/clig.html.

Laird, J. (2003). Soar 8 tutorial. http://www.eecs.umich.edu/ soar/tutorial.html.

Lewis, R. (1993). *An Architecturally-based theory of Human Sentence Comprehension*. PhD thesis, Carnegie Mellon University.

Liddle, S., Yau, S., and Embley, D. (2001). On the automatic extraction of data from the hidden web. In *Proceedings of the International Workshop on Data Semantics in Web Information Systems*, pages 27–30.

Lonsdale, D., Hutchison, M., Richards, T., and Taysom, W. (2001). An integrated system for processing information from genealogical text. In *Workshop on Technology for Family History and Genealogical Research*.

Magnani, M. and Montesi, D. (2004). A unified approach to structured, semistructured and unstructured data. Technical Report UBLCS-2004-9, University of Bologna.

McCray, A. T. (2000). Better access to information about clinical trials. *Annals of Internal Medicine*, 133(8):609–614.

Molla, D. and Hess, M. (2002). Dealing with ambiguities in an answer extraction system. In *Proceedings of ATALA Workshop on Representation and Treatment of Ambiguity in Natural Language Processing*, pages 21–24.

Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries*, pages 195–204, New York, NY, USA. ACM Press.

Nakamura, Y. and Kanade, T. (1997). Semantic analysis for video contents extraction: Spotting by association in news video. In *Proceedings of the Fifth ACM International Conference on Multimedia*, pages 393–401.

Newell, A. (1994). *Unified Theories of Cognition*. Harvard University Press.

Parker, C. (2003). Generating medical logic modules for clinical trial eligibility. BYU Computer Science Master's Thesis Proposal.

Pustejovsky, J., Castano, J., Cochran, B., Kotecki, M., and Morrell, M. (2001). Automatic extraction of acronym-meaning pairs from Medline databases. In *Proceedings of Medinfo*, pages 371–375.

Pustejovsky, J., Castao, J., Zhang, J., Kotecki, M., and Cochran, B. (2002). Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Proceedings of the 7th Pacific Symposium on Biocomputing*, pages 362–373.

Radhakrishan, R., Xiong, Z., Divakaran, A., and Ishikawa, Y. (2004). Generation of sports highlights using a combination of supervised and unsupervised learning in audio domain. Technical Report TR2003-144, Mitsubishi Electric Research Laboratories.

Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Databases*, pages 129–138, San Francisco, CA, USA. Morgan Kaufmann.

Rindflesch, T., Rajan, J., and Hunter, L. (2000). Extracting molecular binding relationships from biomedical text. In *Proceedings of ANLP-NAACL*, pages 188–195, San Francisco, CA, USA. Morgan Kaufmann.

Rosenfeld, A., Doremann, D., and DeMenthon, D., editors (2003). *Video Mining*, chapter Unsupervised Mining of Statistical Temporal Structures in Video. Kluwer.

Schneider, G. (1998). A linguistic comparison constituency, dependency, and link grammar. Master's thesis, University of Zurich.

Schwartz, A. and Hearst, M. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, pages 451–462.

Skousen, R. (1989). *Analogical Modeling of Language*. Kluwer Academic Publishers.

Skousen, R., Lonsdale, D., and Parkinson, D. B. (2002). *Analogical Modeling: An exemplar-based approach to language*. John Benjamins.

Sleator, D. and Temperley, D. (1991). Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. In *Machine Learning*, pages 233–272, Hingham, MA, USA. Kluwer Academic Publishers.

Stephens, M., Palakal, M., Mukhopadhyay, S., Raje, R., , and Mostafa, J. (2001). Detecting gene relations from Medline abstracts. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 483–496.

Tustison, C. (2004). Automatically extracting predicate argument structures from natural language text. *Lacus Forum XXX*.

van Rijsbergen, C. (1979). *Information Retrieval*. Butterworths.

Walker, T. and Embley, D. (2004). Automating the extraction of genealogical information from the Web. In *Fourth Annual Workshop on Technology for Family History and Genealogical Research*. Brigham Young University.

Appendix A

Examples Showing LG-Soar Process

This appendix shows just three examples of criteria that are input into the LG-Soar system. The table outlines the the corresponding parse output by the LG parser and then shows the final output produced by the system.

Table A.1: LG-Soar processing examples

Original Input	Syntactic Parse	Final Output

History of cancer	`    +------------------------Xp-----------------------+`
	`    +-----Wd----+                                      +`
	`    +--Ds--+----Ss----+---Os---+--Mp--+--Jp-+          +`
	`    +     +           +        +      +     +          +`
	`LEFT-WALL a criterion.n equals.v history.n of cancer.n.+`
		cancer(N5) `&amp;` history(N4)
		`&amp;` of(N4,N5)

Femoral neck osteoporosis	`    +----------------------------------Xp----------------------------+`
	`    +                      +---------------Os--------------+        +`
	`    +-----Wd----+          +          +----------A---------+        +`
	`    +     +--Ds--+----Ss----+         +         +-----AN----+        +`
	`    +     +      +          +          +         +            +        +`
	`LEFT-WALL a criterion.n equals.v femoral[?].a neck.n osteoporosis[?].n.`
		femoral(N6) `&amp;`
		neck_osteoporosis(N6)

Age eligible for study: 18 years and above	`     +----------------------------Xp---------------------------+`
	`     +-----Wd-----+                                +---Opc--+   +`
	`     +     +--Ds--+----Ss----+--Os--+--Mam--+--MVt--+   +Dmcn+   +`
	`     +     +      +          +      +       +       +   +    +   +`
	` LEFT-WALL a criterion.n equals.v age.n greater.a than 18 years.n.`
		age(N4) `&amp;`
		greater_than(N4,N5) `&amp;`
		measurement(N5) `&amp;`
		units(N5,N6) `&amp;` years(N6)
		`&amp;` magnitude(N5,18)

Appendix B

Examples of Incorrect Output

In this appendix, we present 15 different examples of criteria found in the clinical trials repository. These examples are criteria that are currently not parsed out correctly in the LG-Soar system. By studying these examples, future researchers can have a snapshot of specific ways to improve the functionality of the system.

Input	**Output**	**Reason for Incorrect or Incomplete Output**
Hemoglobin greater than 9 g/dL	No output	Does not parse because the units *g/dL* confuse the parser.
No prior tyrosinase:368-376 (370D), gp100:209-217 (210M), or ART-1:26-35 (27L) peptides	neg(N5) `&amp;` prior_tyrosinase(N5)	Parser does not know how to treat the number combinations. In this example, the parser treats *prior* as a noun.
Karnofsky 60-100`%`	No output	Does not parse because of the `%` sign.
HLA-A2 positive	No output	Does not parse because *positive* in the link grammar dictionary is an adjective and therefore the input is not a complete sentence.
Prior chemotherapy allowed	prior(N4) `&amp;` chemotherapy(N4)	The word *allowed* is not in the output. Also, *prior* is here treated as an adjective.
Any age	age(N4)	No contextual information is given about what *any* refers to semantically.
An estimated 5,000 obese diabetic subjects from the Look AHEAD clinical trial. There will be 2,500 women (50`%`) and 30`%` minorities.	No output	There is no output because there are two sentences per line fed into the system, which is not allowed at this time.
Patient has used concomitant medications that may suppress the immune system.	No output	More work needs to be done with modals in LG-Soar.

Are likely to have bleeding disorders.	No output	The input already contains a verb. This verb conflicts with the dummy verb *equals* which is added during pre-processing.
Ascites (abdominal fluid)	ascites(N4)	Soar is not programmed to handle links identifying extra symbols, e.g. parentheses.
Pregnancy/breastfeeding.	No output	The parser does not know how to treat the / symbol.
4. Women known to be pregnant	No output	Some criteria are numbered, which causes problems for the parser.
Physical examination within normal limits;	No output	Does not parse because of the semicolon. The LG parser requires a sentence to terminate in a period or nothing.

Table B.1: Examples of incorrect output