NATURAL LANGUAGE PARSING AND REPRESENTATION IN XML

By

EUGENIO JAROSIEWICZ

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2003

To all those who think differently.

ACKNOWLEDGMENTS

I would like to thank my family, friends, teachers, and employer.

TABLE OF CONTENTS

page

LIST OF FIGURES

Figure page

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

NATURAL LANGUAGE PARSING AND REPRESENTATION IN XML

By

Eugenio Jarosiewicz

May 2003

Chair: Douglas D. Dankel II
Major Department: Computer and Information Science and Engineering

This thesis presents an approach for translating natural language into an intermediate form represented in XML that is suitable for use in computation. It demonstrates an application of this work used in processing human natural language input as part of a web-based question-answering system.

This thesis provides a brief overview of natural language processing. It describes the difficulties faced in the field of computational linguistics, some of the most commonly used solutions, and how they relate with the current state of the art in enabling a semantic web. This is followed by the practical and philosophical motivations for this work. Finally, the implementation is explained, along with a discussion of the limitations of the system and potential for future work.

This research was part of the WebNL project at the University of Florida, the goal of which was to develop a system that can provide intelligent answers in natural language to user-entered natural language questions. The system is composed of four parts–a

natural language parser to XML translator, a query engine, a knowledge base, and a user interface. The focus of this thesis is the first component and its related issues.

CHAPTER 1
INTRODUCTION

Information and language are the cornerstones of our civilization. Without some form of language, we would not be able to share information with each other and pass it down to our progeny. Without continual accumulation of information, the world as it is today would not be possible. In light of this, the development of language can be regarded as a revolutionary step in our history.

Over time, some of the most important discoveries, inventions and creations have allowed us to more effectively communicate and distribute knowledge and information– the development of writing, the printing press, the telegram/telephone. All of these technologies have the following advantage in common–they have allowed people to exchange information faster and more reliably than before. However, these same technologies share a similar drawback–they do not allow for efficient searching and location of a specific piece of information.

Many people believe we have been and are presently witnessing another pivotal point in our evolution. With the advent of computers and the Internet, it is now possible for people to not only communicate and share information at increasingly amazing speeds, but to also search for information just as rapidly. The capability to search enormous amounts of information with relative easy is what separates this new form of communication from all previous ones.

However, speed and quantity are not the final solution, but just a step in right direction. There still remain many challenges and unresolved issues with access to all

this information. Most of the data on the Internet is highly unorganized. Furthermore, most of this information is in the same format as those that have preceded the Internet– namely, written in natural language. While this seems very straightforward and does not seem to be inherently problematic, the difficulty is that so far we have not been able to program computers to use and truly understand natural language the way we as humans do.

Humans seem to have an innate capability to learn, understand and use language. Looking at language as another technology, it is one of the oldest and most evolved of those we possess. It is also one of the seemingly most complicated ones, because we do not fully comprehend how we use it. Ever since the beginnings of computer science people have tried to duplicate the process of language understanding in machines, and have had little success.

Currently, searching on the Internet is not the ideal situation one would imagine. Because computers do not fully understand natural language, they are limited in their capacity to make sense of an entered query or of all the information that can be retrieved. This limits current search technologies to keyword matching. For simple queries, this can be a quick and effective method, but for advanced queries, this is not sufficient. A related problem that is especially relevant to simple common keyword queries is that the volume of information presently available is very large, so specific results are difficult to achieve. Some search engines try to use advanced heuristics or even techniques from natural language processing and understanding to improve searches, however the results leave something to be desired.

Beyond the issue of keyword searches at a syntactic level of language lies a much more fundamental and challenging problem–that of semantics. Semantics is the meaning of a given subject. While computers can very quickly scan large volumes of data for a given string or pattern, they still do not understand what they are searching for or the data they are searching through. Most of the data on the World Wide Web (or web, which is the most visible part of the Internet), is stored in the HTML format. HTML was designed to be a simple format for interchanging information between people. Since its origins it has expanded to encompass many things (often inconsistently), but its primarily function is a presentation and layout language. To make it easier for machines to understand the content of the data they are processing, various other markup languages have been created. Of these, the Extensible Markup Language, or XML, has become the most widely adopted format. However, XML only describes the structure of resources, and this by itself does not attribute any meaning to the data. More complicated concepts are necessary and have been created–the Resource Description Framework (RDF), Schemas, and various ontologies. While many of these seem to hold some promises for various aspects of enabling machines and computer programs to understand the information they process, it is still unclear how effective they will be.

The rest of this thesis covers related topics in more detail. Chapter 2 provides more background to many of the technologies involved. Chapter 3 provides an overview of the implemented system. Chapter 4 explains the details of the parser/translation part of the system. Chapter 5 concludes with a discussion of the limitations of the system and considers possible future solutions and directions.

CHAPTER 2
BACKGROUND

Language and the Brain

Even as you read this, your brain is processing enormous amounts of information. Visual inputs of colors and patterns from your eyes (or sensory information from your other senses such as sounds or feelings from your sense of touch when reading Braille) are clumped together and are recognized as distinct symbols such as letters, numbers, punctuation, or other patterns. Characters unite to form words, which accumulate into phrases, sentences, and paragraphs.

Unfortunately, even though it is simple to describe abstractly, the exact mechanisms by which all of these things happen are very complicated, and still not fully understood by anyone. No one is yet able to describe in complete detail how the atoms on a sheet of paper or electrons from a monitor are processed by the brain, represented as knowledge, and stored as information therein. This is the neurological equivalent of what is known as the Knowledge Representation Problem, which we cover again later.

All of the combinations of symbols and words compose the system that we commonly know as language. The Merriam-Webster dictionary defines language as "1 a : the words, their pronunciation, and the methods of combining them used and understood by a community" and "2 : a systematic means of communicating ideas or feelings by the use of conventionalized signs, sounds, gestures, or marks having understood meanings" [Merriam-Webster 2002]. Both of these definitions touch upon aspects that most people

would generally associate with the concept of language. However, the definitions are far from an exact description of how language is used or understood.

It is not known whether people are born without any inherent knowledge of language. There are two distinct views on this. The empiricist view assumes "that a baby's brain begins with general operations for association, pattern recognition, and generalization, and that these can be applied to the rich sensory input available to the child to learn the detailed structure of language" while the rationalist view postulates "that the key parts of language are innate–hardwired into the brain at birth as part of the human genetic inheritance" [Manning & Schutze 1999].

It is not clear whether language evolved by necessity to work with brains, or whether our brains evolved to be capable to processing language; however while this is slightly beyond the scope of this thesis, it is the author's opinion that quite likely both influenced each other. What is evident is that during youth most children are taught a language, at which point they may become capable of acquiring more knowledge for themselves, either from other people or from other sources such as books, magazines, journals, etc. This can be seen as a process similar to that of bootstrapping a computer [Whatis 2002].

Today, language is a medium for the transmission of information. It has evolved (alongside our brains) over thousands of years into a system for the communication and representation of ideas and knowledge. Without language, society would not be possible today. Language is the primary method how humans prosper–by accumulating and passing down knowledge from one generation to the next. In the next section we consider some of the principles of knowledge and information.

Information and Knowledge

Information is a very precious resource. It affects the world and peoples lives in increasingly complex ways. This might be because information itself is very complicated and diverse. Information can be both tangible or intangible, important or meaningless, necessary or redundant, decisive and clear-cut or vague and fuzzy.

Perhaps the most important reason why information is so important is because accurate and correct information is not always attainable. From tomorrow's weather to the outcome of a simple program, some processes are not completely determinable or computable to us in the present as far as we know it. The reasons for this inability vary from a lack of information to the presence of contradictory information, a lack of other resources, or simply inherent in the laws of math (a mathematician named Gödel proved that one can not state or prove all true facts in a first order model–this is known as Gödel's Incompleteness Theorem [Vamos 1990]) or physics (some people speculate on the possibility of time travel and perhaps too this is just another bit of information outside our current knowledge, but this is not a thesis on theoretical physics).

Nevertheless, everyone uses information every day, in some form or another, and always without knowing all the facts. Over time humans have evolved into rational beings that can think and reason even in the presence of partial, ambiguous, or contradictory information, or sometimes even in the total absence of information! Just as humans invented and constructed machines to help them work in the physical world, they invented and created computers to help in the abstract world of knowledge and information.

The Internet

There is an enormous amount of information contained in written form in books. However, it is not always easy or fast to transmit the contents of books, particularly between large distances. One of the original goals of the Internet and similarly the World Wide Web were to facilitate sharing of information ubiquitously [Moschovits et al. 2002, Berners-Lee 2002]. With the advent of computers and the Internet, it has become easy for people with the proper resources to access and share information.

There is a significant amount of information available on the Internet and the quantity is increasing at a rapid rate. Until recently it would have been nearly impossible to have a system that had a corpus of information large enough to encompass the arbitrary topics that people know of and discuss. The expansive growth of the Internet has created such a corpus–the Internet itself. Much of the material on the Internet is written for people, by people. As one would expect, this means that a majority of it is written in natural language. The size and content are both a blessing and a burden. While containing large quantities of information, the relatively large size of the Internet makes it difficult to search efficiently, and since people are not perfect, there is often contradictory information that makes it difficult to obtain valid and correct data, necessitating a more intelligent search. There is a large body of research done on this, with a yearly conference devoted to the subject of text retrieval [TREC 2002].

Parsing and Syntax

Going back to our definition of language, we stated that at a base level it is simply a combination of words and other symbols. Parsing is the process of taking input, analyzing it, and assigning it a structure. The input data generally comes from the domain of languages, be it artificial (such as a computer programs written in C), or

natural (such as English) [Wall et al. 1996]. The difficulty with natural languages as compared to artificial ones is that they are much less structured. Artificial languages are typically specified with a precise syntactic structure, where the meaning of sentences is derived from this structure, and an invalidly arranged sentence is not defined. Natural languages often have a certain structure, but are much more lenient and allow for irregularities. Actually, it should be said that human beings are able to interpret and understand natural language sentences containing inconsistencies, while computers are not. Since we do not fully understand the process of natural language, we have not been able to program computers to use it as fully as we do, or otherwise we would be speaking to and programming computers by language!

Semantics

Beyond the simple structural construction of words into sentences, the purpose of language is to convey meaningful information. Semantics is involved with the meaning of a particular subject in question. Natural language processing and understanding has been an active area of research in the computer science and other communities since very early in the history of computers. Major research projects existed in the 1950s for translating sentences in one language into another. While it was originally thought to be a straightforward task to write a program that would allow a computer to interact with a person using natural language, it has so far been shown to be quite the opposite. A large part of this difficulty comes from the originally overlooked complexity of semantics.

The dynamics of natural language and unbounded scope of its content makes automatic natural language question answering very difficult. Conversations and other forms of discourse generally involve a particular context that provides necessary background for meaning and understanding of the statements. A sentence typically

contains more information and meaning than the sum of its words. Furthermore, understanding a sentence depends on not just the words that compose it, but also on the context in which it is presented. Without a specific context and even sometimes with one, a question or other utterance can have many interpreted meanings. Once attributed a specific context, meanings can be applied to each individual word. Being able to understand a given input and rationally process it is one of the "holy grails" of AI research. To do this would be to solve the Knowledge Representation problem, at least in part.

A start to a possible solution to the problem would be to be able to represent a string and the meanings of its constituent words, followed by its phrases and larger structures. Ideally, one would want to be able to describe the structure and content of natural language in an unambiguous way. After this it would be possible to continue to the more difficult problem of representing and understanding meaning.

XML

The Extensible Markup Language (XML) is the self-proclaimed universal format for structured documents and data on the Web [World Wide Web 2002]. XML is actually a *meta-language*–a language for describing other languages–that lets you design your own customized markup languages for limitless different types of documents [Flynn 2002]. It is highly likely (and desirable) that XML will establish itself as the main high-level protocol for information transfer on the Web [Brew 2002]. XML allows users to define custom tags and elements for the tags, which allows it to represent any content. All of these features make XML an ideal choice for representing the structure of natural language.

Summary

Language is a medium for the transmission of information. Information is infinite in quantity and forms of representation. Furthermore, the same piece of information can have multiple methods of interpretation, understanding, and use. The Internet and the World Wide Web are public abstract spaces for storage and distribution of information. XML has established itself as the primary method for representing structured data. The following chapter describes how these technologies are combined and used in the WebNL system to create a natural language web-based question answering system.

CHAPTER 3
WEBNL

In our endeavor to address some of the problems of intelligently searching the Internet for information, we created the WebNL system. The WebNL system attempts to use internal knowledge of natural language to more accurately understand the user's query and the contents of the data it is searching, as well as providing an intelligent interface to the search with responses in natural language. The system is composed of four modules. The modules are a natural language parser and XML translator, a query generator and information retriever, an XML knowledge base, and an intelligent interface and natural language answer generator.

Using WebNL

As can be guessed from the name, WebNL is accessed via the web. All of WebNL is operated via the intelligent interface, which is a web-based application written with Flash. The interface takes the user's question and passes it to the parser. The parser receives the query, generates a parse tree from it, and converts it into an XML representation of a natural language parse tree. The XML is read by the query generator, which creates a query in XML Query Language (XQL), invokes the query on the XML Knowledge Base (XMLKB), and returns part of the XMLKB structure to the intelligent interface. The interface then is responsible for processing the returned structure and generating a natural language response to the user's question. Figure 3.1 illustrates the operation of the system.

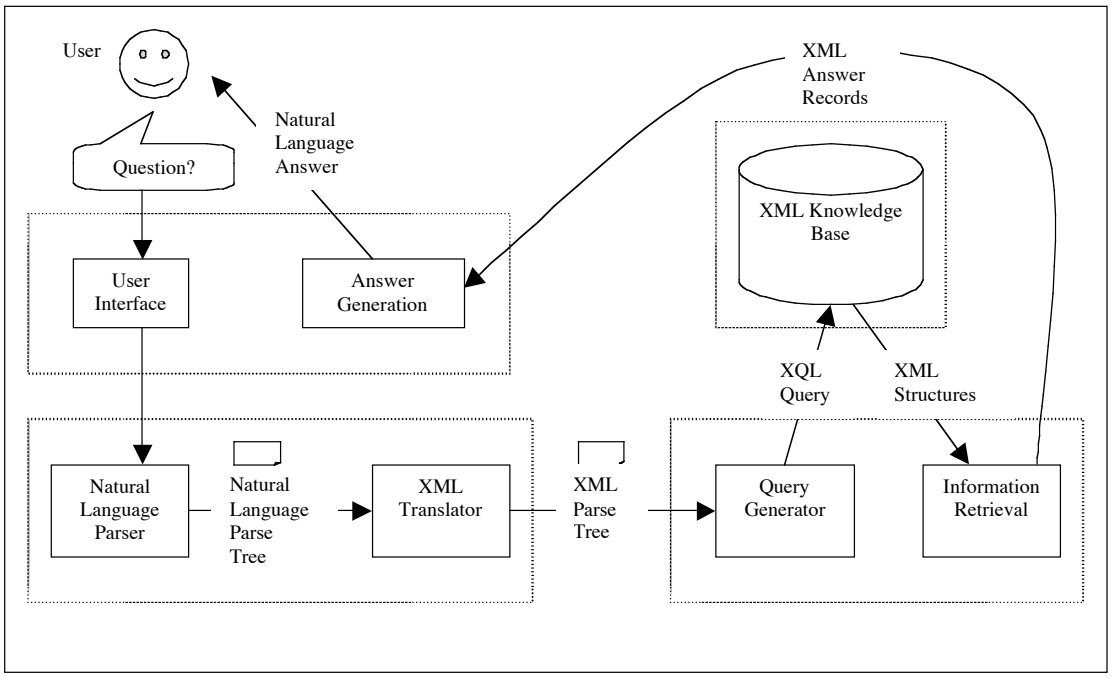

Figure 3.1 – Overview of WebNL

The Parser Module

The first part of the system that processes the user input is the parser module. It receives the user input entered through the interface (e.g., "What is the description of COP5555?") and passes it to a natural language parser. The natural language parser parses the sentence to create a constituent tree. The results from this are shown in Figure 3.2. It then takes the constituent tree and post-processes it in a second parsing stage, where it emits the parse tree in XML format. This is passed to the query module. Figures 3.3 shows the final output from this module. Since the topic of this chapter is this parser module, it is discussed in depth in Chapter 4.

```
++++Time                                                0.01
seconds (1401.64 total)
Found 4 linkages (2 with no P.P. violations)
  Linkage 1, cost vector = (UNUSED=0 DIS=0 AND=0 LEN=7)

    +----------------------Xp---------------------+
    |             +-----Ost----+                  |
    +---Ws--+Ss*w+   +---Ds---+---Mp---+-Js-+    |
    |       |    |   |        |        |    |    |
LEFT-WALL what is.v the description.n of COP5555 ?

Constituent tree:

(S What
   (S (VP is
          (NP (NP the description)
              (PP of
                  (NP COP5555)))))
   ?)
```

Figure 3.2 – The constituent tree output from the natural language parser.

The Query Module

The second stage of processing occurs in the query module. It accepts the parsed natural language question in XML format and constructs a query in XQL format. To improve the quality of the results and identify the most accurate answers, the query engine uses various heuristics, scoring methods, and search algorithms such as Question Analyzing, Keyword Matching, Element Indexing, Directory Searching, and Tag Element Searching. To increase the likelihood of finding a match, the query engine also expands the query with synonyms of the question terms by using an electric online dictionary called WordNet [Miller & Fellbaum 1998].

Once the query engine has identified the document within the XMLKB that contains the best answer, it executes the query on the knowledge base, returning the

```
<?xml version="1.0" encoding="UTF-8"?>

<QUERY>
    <SENTENCE>
        <NOUNPHRASE>
            <PRONOUN string="What">
                <ROOT>What</ROOT>
                <NUMBER>indeterminate</NUMBER>
            </PRONOUN>
        </NOUNPHRASE>
        <SENTENCE>
            <VERBPHRASE>
                <VERB string="is">
                    <ROOT>be</ROOT>
                    <TENSE>present</TENSE>
                    <NUMBER>singular</NUMBER>
                </VERB>
                <NOUNPHRASE>
                    <ARTICLE string="the">
                        <ROOT>the</ROOT>
                        <TYPE>definite</TYPE>
                    </ARTICLE>
                    <NOUN string="description">
                        <ROOT>description</ROOT>
                        <NUMBER>singular</NUMBER>
                    </NOUN>
                    <PREPOSITIONPHRASE>
                        <PREPOSITION string="of">
                            <ROOT>of</ROOT>
                        </PREPOSITION>
                        <NOUNPHRASE>
                            <NOUN string="COP5555">
                                <ROOT>COP5555</ROOT>
                                <NUMBER>singular</NUMBER>
                            </NOUN>
                        </NOUNPHRASE>
                    </PREPOSITIONPHRASE>
                </NOUNPHRASE>
            </VERBPHRASE>
        </SENTENCE>
    </SENTENCE>
</QUERY>
```

Figure 3.3 – The final output of the natural language to XML parser module

matching records in XML form. The query engine then post-processes the results, removing unnecessary tags and adding some additional meta-information about the resulting answers such as whether the answers returned were exact or partial matches, and writes this out to a file. Figure 3.4 shows an example of output from the query module. More examples can be found in [Pridaphattharakun 2001].

```
<?xml version="1.0" ?>
<RESULT number="1">
  <QUERY string="WHAT IS THE DESCRIPTION OF COP5555?" />
  <ANSWER type="E">
    <COURSE>
      <CONTENT>Graduate course: Programming Language Principles</CONTENT>
      <TEXT>Programming Language Principles</TEXT>
      <NUMBER>
        <CONTENT>Programming Language Principles course number</CONTENT>
        <TEXT>COP 5555</TEXT>
      </NUMBER>
      <DESCRIPTION>
        <CONTENT>Description of Programming Language Principles</CONTENT>
        <TEXT>History of programming languauges, formal models for specifying languages,
        design goals, run-time structures, and implementation techniques, along with survey
        of principal programming languauge paradigms.</TEXT>
      </DESCRIPTION>
      <PREREQ>
        <CONTENT>Prerequisites for Programming Language Principles</CONTENT>
        <TEXT>COP 3530</TEXT>
        <LINK>
          <TEXT>COP 3530</TEXT>
          <TARGET>http://www.cise.ufl.edu/~ddd/grad/undergrad_pre.html#COP3530</TARGET>
        </LINK>
      </PREREQ>
    </COURSE>
  </ANSWER>
</RESULT>
```

Figure 3.4 – The output from the query module

The XML Knowledge Base Module

Once the query generation module creates a query, it executes the query on the XML Knowledge Base. The XMLKB is the central database containing all the information that the system effectively knows and can provide data about. In the current implementation,

the XMLKB is a static database of documents in XML format is manually created before the system is used.

Since the structure of the knowledge base is known a priori, this allows the use of a Document Type Definition (DTD) for the rest of the documents. The DTD is the initial file of the knowledge base. It provides definitions of the XML structures used in the rest of the files. The next part of the database, the directory file, is the most important. It is the meta document for the database and contains a list of the remainder of the documents along with a summary of their contents. The rest of the documents are sub-area files that contain specific information about each individual section of the knowledge base. Examples of each of these files can be found in [Nadeau 2002].

The Interface Module

The final component of the WebNL system is the interface module. Its purpose is twofold, serving both as an intelligent interface and as a natural language answer generator. As the intelligent interface, it is the only component of the system the user sees and interacts with, so it is the only part of which they are aware. Part of a good question answering system is that it makes it easy for users to search for information and presents that information in an easy to read format.

To achieve this, the interface module is implemented as a FLASH web application. It provides an intuitive interface for entering questions and providing answers. It is also the driver for the rest of the system and is the one that invokes the other components sequentially. When the user enters a query, it passes this to the parser module, the results of which are used as input for the query module. Once the query module searches the knowledge base, the interface module reads the file created and parses it, filtering information based on the amount and types of results. It then presents answers to the

users questions via hyperlinks, or as natural language content generated using a template-based approach, depending on the results. Examples of the results provided by the intelligent interface can be found in [Nantonio 2001].

Summary

The WebNL system is composed of four modules. The modules are a natural language parser and XML translator, a query generator and information retriever, an XML knowledge base, and an intelligent interface and natural language answer generator. Data is passed between each module in the form of XML files. The following chapter covers the natural language parser and XML translator module in detail.

CHAPTER 4
TRANSLATING NATURAL LANGUAGE TO XML

The essential problem and motivation for this project is that humans and computers do not speak the same language. Humans and language have evolved over millennia, and human's capacity for expression is seemingly only limited by their imagination. While human languages tend to have a general structure, this appears to be solely for the purpose of facilitating people to communicate in similar terms, but this does not limit the complexity or variety of uses of natural languages.

Computers on the other hand fundamentally only understand and reason in a binary language–yes or no, on or off. However, from this base foundation, it is possible to implement logic and formulate advanced modes of computation. Over the relatively short period of the history of computers, people have succeeded in conceiving increasingly abstract methods of communicating with computers. However, almost all high-level computer languages and user-interfaces are still based in the strict world of rules that computers are programmed by, and do not know how to deal with the unrestricted domain of human language.

An ideal solution to the problem would be for computers to be able to communicate with humans in natural language. In an attempt to develop a solution to this problem requires a method of having computers process and understand natural language. As was previously discussed, the act of understanding is complex and not fully understood, and as such is a topic outside the scope of this thesis. To be able to process natural language

is a worthy goal, and merits investigation. Such a system might be used as a basis for reasoning with language at a more abstract level.

Parsing

The first step in processing a statement is to recognize it as such. This may sound straightforward, but in the general case with computers is not so. There are many issues involved with locating the boundaries of sentences when dealing with arbitrary input. For the simplified case of the WebNL system, the boundary is clearly defined as the statements are expected to be in the form of a user's question. It should be noted therefore that the parsing strategies discussed here deal primarily with intra-sentence parsing, and not inter-sentence parsing.

Once a sentence or statement is identified, the next goal is to analyze the sentence and reduce it into its constituent members. As mentioned, language is actually very complex, with a full description of all the subtleties spanning many books, so it is not discussed in detail here. However, since it is relevant, we provide some basic examples for reference.

Sentence Structure

An English sentence is generally composed of three parts–a subject, an action, and an object. A very simple example is the following–

Alice buys an apple.

The subject of the sentence is the noun "Alice", which is also a proper name. The action is the verb "buys", and the object is "an apple". The word "an" is an article, and in this case is used to connect the action verb with the object. This is a very simple sentence; there are many other kinds of words in the English language, and a combinatorial explosion of ways to put them together.

Ambiguity

Ambiguity in grammars refers to the situation when two or more distinct paths are available for a parser to follow, and it is not immediately known which, if any, is the correct or best choice. Sometimes, certain words can be used in different ways. For example, the word "saw" can be either the past tense of the verb "see" or the physical instrument for cutting. This ambiguity leads to the major problems of parsing sentences. In many simple to moderate cases, this ambiguity can be resolved by other contextual information, such as location of the word in the sentence and how it is used, or in cases of names, capitalization, etc. However, in very ambiguous cases, there is no way to resolve the ambiguity from just one sentence, and more contextual information is needed–such as the circumstance in which the sentence is presented. This ambiguity is at the syntactic level, but can also occur at the semantic level. A well-known example is the following–

Time flies like an arrow.

Without knowing the context that this was made in, there are multiple ways to parse this (with "Time," "flies," or "like" being the verb), and even more ways to interpret this. One should easily see the difficulties that can arise from ambiguity.

Grammars

A grammar is simply a formal specification of the structures allowable in the language. There are a variety of different grammars. Work in language theory began with Chomsky (1956), who […] defined context-sensitive, context-free, and regular grammars [Allen 1987]. Grammars can be defined by a set of rewrite rules that describe what structures are allowed, and how symbols may be expanded into other sequences. There are two kinds of symbols, terminal and non-terminal symbols, which vary only in

whether they can or cannot be expanded into other sequences of symbols, respectively. Grammars may also be visualized as transition networks consisting of nodes and labeled arcs. Regular grammars are equivalent to simple transition networks in their generative capacity to construct sentences, while context-free grammars are less restrictive; these are equivalent to recursive transition networks [Allen 1987].

Parsing Techniques

Parsing techniques are methods of analyzing a sentence to determine if its structure is accordance with a grammar. Many distinct parsing techniques have been invented. It is not our goal to cover them all here. Most of the parsing techniques work based on similar principles, with two of the most common parsing techniques being top-down and bottom-up parsing [Allen 1987].

In top-down parsing you begin from the representation of a sentence as a single symbol, and you try to decompose the representations into sub-constituents until you derive the final words, or terminal symbols. One of the problems with top-down parsing is that with ambiguous grammars it is possible to attempt to decompose a symbol incorrectly, by decomposing it into further non-terminal symbols which later do not match and decompose into terminal symbols. To solve this, it is necessary to keep a list of backup states to search in case of failure. Likewise, some grammars cannot be parsed when using a depth-first search, because the search results in an infinite left-recursion loop matching a particular symbol and decomposing it into another series of symbols beginning with itself.

Bottom-up parsing works in the opposite direction, trying to match individual words and replacing them with their syntactic categories. Bottom-up parsing has the advantage

of not being susceptible to left-recursion, but has similar disadvantages. Shift-reduce parsing is an example of bottom-up parsing.

Many other parsing techniques and algorithms exist. Mixed-mode parsing uses a combination of the top-down and bottom-up strategies. Parsers can also use look-ahead to improve the efficiency of the algorithms and avoid some potential pitfalls of ambiguity. Look-ahead can help resolve local ambiguities, but no amount of look-ahead will help if the whole sentence is ambiguous. In such situations some type of heuristic is necessary–often called an "oracle" or "omniscient parser function".

One of the main issues of the parsers discussed above is the inefficiency in backtracking when ambiguities arise and when there are incomplete structures in the input. Chart parsing is a method which helps overcome these problems. A chart is a form well-formed sub-string table (wfst). It helps avoid duplication of work by saving each successfully parsed sub-string into a chart structure that can be consulted anytime afterwards, making it unnecessary to re-parse a string in the same way twice when backtracking.

For more detailed discussions on parsing, please refer to [Allen 1987, Weingarten 1973, Sebesta 1997], or any good reference on natural or programming languages.

Link Parser

Many parsers have already been written. Most of the existing parsers use one of the techniques described above, a variation of one of the techniques, or combination of multiple techniques. The output produced by the parsers varies, usually some form of tagged structure or constituent tree. Since XML was chosen as the language for communication between the intermediary stages of WebNL, we required a parser that could produce output in XML format. After evaluating several existing parsers, the

"Link Grammar Parser" from Carnegie Mellon University was selected as the most fitting for our use.

The Link Parser claims to be a unique parser in that it is "based on an original theory of English syntax" [Temperley et al. 2002]. The parser takes a sentence and assigns a syntactic structure that consists of a set of labeled links connecting pairs of words. On initial examination, this concept is not dissimilar from other parsing techniques. Most parsing techniques have a left hand part of the sentence that has already been parsed and the corresponding state, along with the right hand remainder of the input that is being searched for possible syntactical matches in the grammar. The differentiating feature of the link parser is "rather than thinking in terms of syntactic functions (like subject or object) or constituents (like verb phrase), one must think in terms of relationships between pairs of words" [Temperley et al. 2002]. Thus, the grammar is not composed of constituent parts of sentences, but rather words "blocks" that accept connections on both the left and right sides to similar blocks. Words have rules about how their connectors can be matched, that is, rules about what would constitute a valid use of that word. A valid sentence is one in which all the words present are used in a way which is valid according to their rules, and which also satisfies certain global rules [Temperley et al. 2002]. Parts of speech, syntactic structures, and constituents can be recovered by tracing these linkages as paths on a graph.

To increase the parser's performance and accuracy, many improvements have been made which can be generally applied to all parsers. For example, the parser is able to handle unknown vocabulary by making intelligent inferences about syntactic categories of words based on context and spelling, including use of suffixes, capitalization,

numerical expressions, and punctuation. It uses a moderately sized dictionary of 60,000 word forms and can skip unknown portions of sentences.

The parser produces two forms of output, a unique linkage structure showing the internal representation of the connections between words, and also a constituent tree labeling the major syntactical parts. It does not fully tag all parts of speech and does not provide other useful relevant meta-information such as word root, tense and number.

Translating to XML

Our goal was to have the parser module emit a parse tree of English language in XML form. To achieve this, we took a two-pronged approach. First, we slightly modified the Link parser's output to be more usable for our purposes. Then, we wrote a second program that post-processes the output of the parser and transforms it into XML, adding additional information where available. During development, a basic web interface was also created to test the process.

The Link parser has many extra features with which we were not directly interested in but that are outputted along with the results of the parse. These included the CPU usage time, total number of complete and incomplete parses or linkages (including ones with null links), the associated computed cost vector (the link parser computes a cost for each linkage with the highest scoring parse chosen as the best), and a connection diagram of the linkages between words. Our parser module does not use this information and it was not essential for the user of the system, so its output is suppressed.

The second stage of the parser module takes the constituent tree outputted by the Link parser and post-processes it, transforming it into a well-formed XML parse tree. As previously explained, for XML to be well-formed it must be internally consistent, with all elements containing opening and matching closing tags. To achieve this, the post-

processor was designed as a recursive parser, processing each enclosed word of the parse tree successively in-order, emitting the opening element tags, adding whatever possible extra information in form of attributes or sub-elements, until it reaches an element that is not a child branch of the current tree structure. The call stack is then unwound, emitting the required closing element tags in the reverse order. This simple technique guarantees that all element tags match correctly, therefore insuring the well-formed structure of the XML parse tree.

The second stage of the parser module also attempts to add extra information to the elements of the parse tree that can be used later. The parser attempts to add attributes to the basic word forms–nouns and verbs, but also adds limited information where available, such as to articles and pronouns. These extra tags are somewhat comparable to features in Augmented Grammars in that they provide meta information about each node, but in this implementation they are not used for other purposes such as subject-verb agreement. For nouns and verbs, the parser attempts to determine the root form of the word, by comparing against a dictionary and searching the substring of the word for well-known prefixes and suffixes. During this process, words are marked as being of a certain number or tense depending if they match certain suffixes. Word types such as articles and pronouns are specially handled with custom cases, and are marked as certain kinds, such as definite or indefinite, depending on rules written into the parser. Finally, most common forms of punctuation are also marked and tagged appropriately.

Summary

People and computers communicate using different languages. A system that can accept input from a user in natural language (English) and transform it into an intermediate form that can be processed on an abstract level is a very useful tool. The

natural language parsing and XML translating module described in this chapter utilizes the Link Parser from CMU to transform English language questions in an XML representation. However, there are inherent difficulties in such a transformation, perhaps best described in a quote by the linguist Edward Sapir : “All grammars leak.” The next chapter discusses some of the limitations as well as accomplishments of this system and gives ideas for future research.

.

CHAPTER5
CONCLUSION

Language is a complex system. Computer technology (along with other fields such as linguistics, psychology, and neurology) has made substantial inroads into the topic of natural language processing and understanding, but the initial rapid progress and successes have not scaled to match the intricacies of language as even a young child's brain is capable of understanding. It is still unclear whether the capacity for language or learning and intelligence at a human level is limited to advanced biological organisms or if someday machines will be able to think and communicate on the level of human natural language.

Accomplishments and Limitations

The goal of the WebNL system was to create an intelligent web-based question-answering system. The key components required by such a system would include an intelligent interface, a natural language parser, a query engine and a knowledge base. XML was used as the intermediary language for communication between the separate modules in the system, as well as the structure of the knowledge base. Creating a system that has the intelligence to answer questions from a human is a formidable task that has not been completely successfully accomplished to this day. This is due to some of the inherent difficulties presented in this thesis, which are discussed further below.

Since many parsers have already been written, we decided to avoid reinventing the wheel by writing a new parser from scratch, but instead to modify an already existing parser for our purposes. The benefit of this approach was that it saved significant time in

developing the fundamental framework for the parser that every parser writer would have to do. It also provided a stable and exceptionally well-performing parser, since it has had years of use and development, which we would not have been able to accomplish in a time-constrained schedule. However, there were some drawbacks to this as well. From an engineering standpoint, by using an existing parser there is loss in flexibility that one would gain by designing and creating a new parser. The existing parser is large and complex, and is not suitable for simple modification. From a natural language standpoint, the Link Grammar is similarly complex and not suitable for simple modification. Many parts of the Link Grammar have been refined over the years to special cases (certain rare and idiomatic syntactic constructions) and just like a puzzle that is pieced together, it is very much dependent on this fragile structure.

Since the Link parser was not well suited for modification, the choice of adding a second post-processing stage resulted in increased flexibility. However, by having two unique and disconnected parts, much information from the first stage that could be useful is not available in the second stage.

As was previously mentioned, one of the largest problems with all the data on the Internet is that it is highly unorganized. Searching against an unorganized database makes it difficult to effectively find items and with current technology is almost impossible to know what the contents of the data truly designate. To overcome this, it was decided to create a well-ordered database in advance for a specific domain. The benefit of this is that it enables the system to easily search the structure of the knowledge base. It also provides a template for what an automated knowledge base generator would create. The obvious downside is that the knowledge base does not automatically gain

information. It is incapable of learning new material and thus is limited in scope to the domain for which it was created, and not able to provide information about other topics.

Finally, all the distinct modules of the system have some inherent knowledge of language. However, they do not share any specific understanding of language between them. Ideally, it would be possible to specify a Document Type Definition (DTD) for the shared information, but the complexity of language is not easily representable in a DTD.

Future Research

An ideal question answering system would be able to handle any coherent form of input, search and find information about its constituent parts, and provide an intelligent answer based on the context of the data. While our system is able to do these things on a constrained scale, applications of the future will not have these limitations.

The obvious potentials for future research are the limitations of our current system. Ideally, the system would be able to use the Internet as a continuously growing knowledge base and extract information from it, dealing with inaccurate and contradictory information. The user interface could be improved to be more dynamic, learning from the user's previous questions to better narrow down the search and results.

The parser itself could improve in a variety of ways, ranging from simple additions to more complicated restructuring. The parser could use specific Internet dictionary sites as additional references to its internal corpus of words. It could also use the Internet as a source for statistical reasoning and as a source of pragmatic and general world knowledge. Like the interface, the parser could use discourse knowledge from previous queries to be able to deal with pronoun resolution in subsequent queries. As was previously mentioned, the parser is composed of two distinct stages, which gives it more

flexibility at a loss of some information along the way. Ideally, the parser would be able to utilize all of the information available to it at every stage of parsing.

Likewise, there is a disconnect between the parser and the rest of the modules in the system, in that there is no “main” function which oversees the results from each stage and properly handles unique situations–if for example the parsing stage found two or more distinct parsers (a often occurring scenario), the interface could present this fact to the user and prompt for clarification.

LIST OF REFERENCES

Allen, J. Natural Language Understanding. Benjamin/Cummings, Menlo Park, CA, 1987.

Berners-Lee, T. Tim Berners-Lee. Available at http://www.w3.org/People/Berners-Lee/ , last accessed on 11/30/2002.

Brew, C. Editor. Helpdesk FAQ. Available at http://www.ltg.ed.ac.uk/helpdesk/faq/Tools-html/0015.html , last accessed on 11/30/2002.

Flynn, P. The XML FAQ. Available at http://www.ucc.ie/xml/ , last accessed on 11/30/2002.

Jurafsky, D. and Martin, J. H. Speech and Language Processing. Prentice Hall, Upper Saddle River, NJ, 2000.

Manning, C. D. and Schütze, H. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, 1999. pp. 4-5.

Merriam Webster Dictionary. Available at http://m-w.com/cgi-bin/dictionary?language , last accessed on 11/30/2002.

Miller, G. A. and Fellbaum, C. WordNet: A Lexical Database for the English Language. Available at http://www.cogsci.princeton.edu/~wn/, 1998. Last accessed on 11/30/2002.

Moschovitis, C. J. P. , Poole, H. , Schuyler, T. , and Senft, T. M. History of the Internet. Available at http://www.historyoftheinternet.com/chap2.html , last accessed on 11/30/2002.

Nadeau, N. An XML Information Base and Explorer for a Natural Language Question Answering System. Master's Thesis, University of Florida, Gainesville, FL. 2002.

Nantonio, N. Intelligent Interface Design For a Question Answering System. Master's Thesis, University of Florida, Gainesville, FL. 2001.

Pridaphattharakun, W. Information Retrieval and Answer Extraction for an XML Knowledge Base in WebNL. Master's Thesis, University of Florida, Gainesville, FL. 2001.

Sebesta, R. Concepts of Programming Languages. Addison-Wesley, Menlo Park, CA, 1997.

TechTarget Network. Bootstrapping – a Whatis Definition. Available at http://whatis.techtarget.com/definition/0,,sid9_gci214479,00.html , last accessed on 11/30/2002.

Temperley, Davy, Sleator, Daniel and Lafferty, John. The Link Grammar Parser. Carnegie Mellon University. Available at http://www.link.cs.cmu.edu/ , last accessed on 11/30/2002.

Text Retrieval Conference Homepage. Available at http://trec.nist.gov/ , last accessed on 11/30/2002.

Vamos, T. Computer Epistomology: A Treatise on the Feasibility of the Unfeasible or Old Ideas Brewed New. World Scientific, Singapore, 1990. pp. 25, 86.

Wall, L. , Christiansen, T. and Schwartz, R. L. Programming Perl, Second Edition. O'Reilly & Associates, Sebastopol, CA, 1996. p. 2-9.

Weingarten, F. W. Translation of Computer Languages. Holden Day, San Francisco, CA, 1973.

World Wide Web Consortium. Extensible Markup Language. Available at http://www.w3.org/XML/ , last accessed on 11/30/2002.

BIOGRAPHICAL SKETCH

Eugenio Jarosiewicz was born on May 8, 1978, in Buenos Aires, Argentina. At the age of six his family moved to the United States, Miami, FL. While growing up he attended Argentine and Ukrainian schools in addition to regular public schools. On May 1st of 1999 he graduated from the University of Florida with a Bachelor of Science degree in computer and information science and engineering. He is currently continuing his education pursuing a Master of Science degree while working full-time at Apple Computer, Inc. He has many interests, including computers and the universe at large.