Extraction of Basic Noun Phrases from Natural Language Using Statistical Context-Free Grammar

By
Taniza Afrin
tafrin@vt.edu

Thesis submitted to the faculty of Virginia Polytechnic Institute and State University in the partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical Engineering

Dr. W. R. Cyre, Chairman
Dr. H. F. VanLandingham
Dr. Tim Pratt

May 25, 2001
Blacksburg, Virginia

Keywords:
Noun Phrase, Probabilistic Parser, Information Extraction, Stochastic Grammar

Extraction of Basic Noun Phrases from Natural Language Using Statistical Context-Free Grammar

Taniza Afrin

Abstract

The objective of this research was to extract simple noun phrases from natural language texts using two different grammars: stochastic context-free grammar (SCFG) and non-statistical context free grammar (CFG). Precision and recall were calculated to determine how many precise and correct noun phrases were extracted using these two grammars. Several text files containing sentences from English natural language specifications were analyzed manually to obtain the test-set of simple noun-phrases. To obtain precision and recall, this test-set of manually extracted noun phrases was compared with the extracted-sets of noun phrases obtained using the both grammars SCFG and CFG. A probabilistic chart parser was developed by modifying a deterministic parallel chart parser. Extraction of simple noun-phrases with the SCFG was accomplished using this probabilistic chart parser, a dictionary containing word probabilities along with the meaning, context-free grammar rules associated with rule probabilities and finally an algorithm to extract most likely parses of a sentence. The probabilistic parsing algorithm and the algorithm to determine figures of merit were implemented using C++ programming language.

Acknowledgment

I thank my academic and research advisor Dr. Walling R. Cyre, for his guidance, support, and assistance throughout my project duration. I am grateful for his prudent advice and invaluable information that helped me to solve many technical problems and finally, to complete this project. I would also like to thank Dr. Hugh F. VanLandingham and Dr. Tim Pratt for serving on my committee. I would like to thank Michael Alley, my course instructor of technical writing, for his wise and constructive review of my thesis write-up. I also would like to thank National Science Foundation (NSF) for financing this research.

Finally, I would like to thank my parents, my husband, all my family members and friends for their continuous support and encouragement.

Table of Contents

Abstract ii
Acknowledgment iii
List of Figures vii
List of Tables ix
Chapter 1: Introduction 2
1.1 Motivation 2
1.2 Research Approach 3
1.3 Research Contributions 4
Chapter 2: Review of Relevant Work 7
2.1 Statistical Parsing with a Context free Grammar 7
2.2 Statistical Techniques for Natural Language 9
2.3 Three New Probabilistic Model for Dependency Parsing 10
2.4 A Probabilistic Parser and its Application 11
Chapter 3: Background 14
3.1 English Language Terminology 14
3.1.1 Lexical Categories 14
3.1.2 Simple Noun Phrases 16
3.2 English Language Grammar 17
3.2.1 Context-Free Grammar 17
3.2.2 Stochastic Context-Free Grammar 20
3.3 Grammatical Tagging 21
3.2.1 ASPIN Tagset 21
3.2.2 Penn Tree-Bank Tagset 22
3.4 Grammar Rules for Simple Noun Phrases 25
3.5 Chart Parsing 27
3.5.1 Probabilistic Chart Parsing 28
3.6 Theory of a Probabilistic Chart Parser 30
3.6.1 Finding the Observation-probability in Probabilistic Chart Parsing 30

3.6.2 Calculating the Probabilities of Words 34
3.7 Performance Measures 35
3.7.1 Positive, Negative, False Positive and False Negative Noun-Phrases 35
3.7.2 Recall 36
3.7.3 Precision 37
3.7.4 F-Measure 37
3.7.5 Trade-Off between Precision and Recall 38
Chapter 4: Implementation 39
4.1 The Stochastic Parser Implementation 39
4.1.1 Samples of Dictionary, Multi-Word Dictionary and Grammar files 39
4.1.2 Samples of the Stochastic Phrases and Chart lists 42
4.2 Main classes of the ProbChunker Program 43
4.2.1 Stochastic Grammar Classes 45
4.3 ProbCompare Program 48
Chapter 5: Results 50
5.1 Recall, Precision and F Factor 50
5.1.1 Performance Metrics for Input1.txt 51
5.1.2 Performance Metrics for Input2.txt 56
5.1.3 Performance Metrics for Input3.txt 62
5.1.4 Performance Metrics for Input4.txt 65
5.1.5 Performance Metrics for combined.txt 68
5.2 Comparing with Others Work 71
Chapter 6: Conclusions 75
6.1 System Capabilities 75
6.2 System Limitations 76
6.3 Future Work 77
Appendix A: List of Acronyms 78
Appendix B: List of Variables 79
Appendix C: Notations of Penn Tree-Bank Tagset 80
Appendix D: A Sample Test-Set of Simple Noun-Phrases 89
References 97

Vita 100

LIST OF FIGURES

Figure 3.1 Simple Context-Free Grammar 18
Figure 3.2 Nonterminal Symbols 18
Figure 3.3 Sub-Tree for the Prepositional Phrase "in the DMA controller" 19
Figure 3.4 A Possible Parse-Tree for "a host machine" 19
Figure 3.5 Another Possible Parse-Tree for "a host machine" 20
Figure 3.6 Stochastic Context-Free Grammar 21
Figure 3.7 A Penn Tree-Bank Tree 23
Figure 3.8 Noun Phrase Grammar Rules 26
Figure 3.9 Stochastic Noun Phrase Grammar Rules 27
Figure 3.10 Chart Parsing for "The DMA controller" 28
Figure 3.11 Stochastic Chart Parsing for "The DMA controller" 29
Figure 3.12 First Parsing Approach 31
Figure 3.13 Second Parsing Approach 32
Figure 3.14 Third Parsing Approach 32
Figure 3.15 POS-Probabilities of the noun-phrase 34
Figure 3.16 Measures of Precision and Recall 35
Figure 4.1 Stochastic Context-free Grammar 40
Figure 4.2 Main Dictionary 41
Figure 4.3 Multi-Word Dictionary 41
Figure 4.4 The Possible Most-Probable Parses for the Fragment 42
Figure 4.5 The Longest Most-Probable Phrases for the Fragment 43
Figure 4.6 CParser Class Diagram 45
Figure 4.7 Cgramlist Class Diagram 46
Figure 4.8 CgramNode Class Diagram 47
Figure 4.9 Cconstitute Class Diagram 47
Figure 5.1 Percentage Recall vs. Sentence No. for SCF Parser for Input1.txt 53
Figure 5.2 Percentage Recall vs. Sentence No. for CF Parser for Input1.txt 53
Figure 5.3 Percentage Precision vs. Sentence No. for SCF Parser for Input1.txt 54

Figure 5.4 Percentage Precision vs. Sentence No. for CF Parser for Input1.txt.....54
Figure 5.5 Percentage F-factor vs. Sentence No. for SCF Parser.....55
Figure 5.6 Percentage F-factor vs. Sentence No. for CF Parser.....55
Figure 5.7 Percentage Recall vs. Sentence No. for SCF Parser for Input2.txt.....58
Figure 5.8 Percentage Recall vs. Sentence No. for CF Parser for Input2.txt.....58
Figure 5.9 Percentage Precision vs. Sentence No. for SCF Parser for Input2.txt.....59
Figure 5.10 Percentage Precision vs. Sentence No. for CF Parser for Input2.txt.....60
Figure 5.11 F-factor vs. Sentence No. for SCF Parser.....61
Figure 5.12 F-factor vs. Sentence No. for CF Parser.....61
Figure 5.13 Percentage Recall vs. Sentence No. for SCF Parser for Input3.txt.....63
Figure 5.14 Percentage Recall vs. Sentence No. for CF Parser for Input3.txt.....63
Figure 5.15 Percentage Precision vs. Sentence No. for SCF Parser for Input3.txt.....64
Figure 5.16 Percentage Precision vs. Sentence No. for CF Parser for Input3.txt.....64
Figure 5.17 Percentage Recall vs. Sentence No. for SCF Parser for Input4.txt.....66
Figure 5.18 Percentage Recall vs. Sentence No. for CF Parser for Input4.txt.....66
Figure 5.19 Percentage Precision vs. Sentence No. for SCF Parser for Input4.txt.....67
Figure 5.20 Percentage Precision vs. Sentence No. for CF Parser for Input4.txt.....67
Figure 5.21 Percentage Recall vs. Sentence No. for SCF Parser for combined.txt.....69
Figure 5.22 Percentage Recall vs. Sentence No. for CF Parser for combined.txt.....69
Figure 5.23 Percentage Precision vs. Sentence No. (SCF Parser) for combined.txt.....70
Figure 5.24 Percentage Precision vs. Sentence No. for CF Parser for combined.txt.....70
Figure 5.25 Reported Percentage Recall.....74
Figure 5.26 Reported Percentage Precision.....74

LIST OF TABLES

Table 3.1 Listing of 8 Parts-Of-Speech of English Language 15
Table 3.2 Simple Noun Phrases 16
Table 3.3 Changes of Penn Tree-Bank POS Tags to 20 ASPIN POS-Tags 24
Table 3.4 Changes of Penn Tree-Bank Syntactic Tags to ASPIN Tags 25
Table 3.5 Stochastic Grammar Rules for "The DMA controller" 31
Table 3.6 Contingency Matrix 36
Table 4.1 C++ Header and Source Files 44
Table 4.2 C++ Header and Source Files 48
Table 4.3 Classes of the "ProbCompare" Program 49
Table 5.1 Sets of Input Text Files 51
Table 5.2 Cumulative Performance Metrics for Input Set 1 52
Table 5.3 Cumulative Performance Metrics for Input Set 2 57
Table 5.4 Cumulative Performance Metrics for Input Set 3 62
Table 5.5 Cumulative Performance Metrics for Input Set 4 65
Table 5.6 Cumulative Performance Metrics for Combined.txt 68
Table 5.7 A Comparison between Church Work and This Research Work 72
Table 5.8 Performance Metrics of Some Statistical Parsing Systems 73
Table C.1 Listing of 12 Punctuation Tags of Penn Tree-Bank 80
Table C.2 Listing of 36 POS Tags of Penn Tree-Bank 81
Table C.3 Listing of Syntactic Tags of Penn Tree-Bank 82

Chapter 1: Introduction

1.1 Motivation

Over the last several decades, researchers of the artificial intelligence and speech recognition communities are trying to develop an algorithm to interpret English natural language using computers. The main objective is to allow computers or machines understand the natural language. However, the research in knowledge-representation using traditional natural language (NL) techniques has been stalled for a long time [8]. Researchers in these areas are now considering various statistical approaches to overcome the limitations to represent natural language automatically. Currently, the statistical approach to represent natural language is an area of hot research. The statistical techniques to extract information from English documents have shown promising results mainly due to its inherent probabilistic approach towards knowledge-representation. This approach allows uncertainty as well as imperfection that comes along with natural language.

My thesis work deals with the detection of simple noun phrases (one of the most important phrase types in English NL) using stochastic context-free grammar (SCFG) as well as using a probabilistic parser. Parsing means breaking a sentence into its constituent nonterminals. For instance, the sentence “Sally drinks.” gives the parsed nonterminal for “Sally” as noun and for “drinks” as verb. Extracting simple noun-phrases is useful in the study of artificial intelligence for various reasons, such as, for an index-term generation in an information retrieval; for the extraction of collocational knowledge from large corpora; and for the development of computational tools for language analysis [10]. Moreover, a noun-phrase parser can be used as a “preprocessor” for the development of a more complex and ambitious parsing system. Noun phrases can also serve as a more appropriate translation unit than any other words.

1.2 Research Approach

To accomplish my objective to compare the extracted noun phrases using both stochastic context free (SCFG) and context free (CFG) grammars, the Penn Tree-Bank tagset, which is used in SCFG, has been changed into the tagset used by the ASPIN grammar. In addition, the number of rules of this probabilistic grammar was reduced eliminating the complex rules, (which detects complex noun phrase) to detect only the simple noun phrases. Chunker program, which was originally used as parser to detect simple noun phrases using CFG, is also modified to use the stochastic context free grammar (SCFG). Two functions were written in C++ to set "word" and "multi-word" probabilities in the modified prob_chunker program. Parsing algorithm was also modified to detect the most probable parses depending on the grammar rule probability and word probability. Three different classes (CgramNode, Cgramlist, Cconstitute) written in C++ programming language have been developed to load the SCFG. Ten functions of these classes were to provide abstraction for the probabilistic grammar such as updating probabilities, deleting duplicate rules, reducing the number of grammar rules (to include rules concerning only noun phrases) and providing a dynamic link to each grammar rule.

To compare extracted noun phrases with exact manually parsed noun phrases, a comparison-algorithm was implemented using a program called ProbCompare. This program was written in C++ language and was developed to calculate both precision and recall of the extracted noun phrases using both the grammars SCFG and CFG. US patents regarding "DMA Controllers" were selected as the input natural language text documents. Ten different US patents were used as input specification documents. Both the dictionary and multi-word dictionary were also updated to identify more noun phrases and thereby increase recall and precision of the extracted noun phrases. All the written C++ programs run on a Windows 95/NT platform. A MATLAB program was written to plot "precision vs. sentence number" and "recall vs. sentence number". For these validation tests, input specification contains six paragraphs from six different US patents. 229 sentences were also manually parsed for this purpose to calculate recall, precision and figure-of-merits for the extracted noun phrases.

1.3 Research Contributions

The author of this thesis made the following contributions.

Dictionary and Multi-Word Dictionary

The main dictionary originally used for non-statistical parser was expanded to include additional meanings for 30 existing entries and to correct meanings for 10 entries. In addition, the dictionary was updated from 5020 entries to 5200 entries to increase the percentages of precision and recall. Both semantic and lexical categories for these entries have been included. These new 180 words were found from the twelve technical US-Patent documents.

The multi-word dictionary used by the non-statistical parser was also updated to contain additional 10 multi-word entries, which occur very frequently in English documents. These multi-words were also found in the above mentioned US patents.

Modification of ASPIN (CFG) Grammar

In order to extract "simple" noun phrases from English documents and compare these extracted phrases with the manually parsed simple noun phrases, three rules have been excluded from the ASPIN grammar. These rules were initially designed to extract first-level composite noun phrases, which contain parenthetical details. In addition, two grammar-rules were added to ensure the same platform for Context-Free Grammar (CFG) as was for Stochastic Context-Free Grammar (SCFG).

Modification of Charniak's Probabilistic Grammar

Charniak's probabilistic grammar was modified to obtain the Stochastic Context-Free Grammar (SCFG). Afterwards, the stochastic parser of my research used

this SCFG to collect the statistical data regarding a particular grammar rule. In order to use the same dictionary by these two grammars (SCFG and CFG), I needed to satisfy two requirements. These requirements were: (1) using the same "tag-sets" for both grammars, and (2) using the same decision metric to detect as well as distinguish the simple noun phrases from composite ones.

To meet the first requirement, the 17 POS (Parts-Of-Speech) tags of ASPIN grammar replaced the 36 POS tag-sets of Charniak's probabilistic grammar. This replacement ensures the compatibility with the dictionary used in both parsers. To satisfy the second requirement, the number of rules used in the SCFG was reduced from 14792 to 11060 by manually eliminating noun-phrase rules that would have otherwise generated the complex noun-phrases. Among these 11060 grammar rules, there were 1624 noun-phrase rules.

ProbChunker

Three classes (CgramNode, Cgramlist, and Cconstitute) containing ten function modules were written in C++ programming language to use the SCFG along with the modified probabilistic parser program ProbChunker. These function modules were used to load 11060 probabilistic grammar rules, to delete duplicate rules, to update the probabilities of the grammar rules, to provide the dynamic links to each rule by the rule's "id" number as well as by the rule's non-terminals, and also to extract 1624 noun-phrase rules. To implement the probabilistic parsing algorithm, three additional functions were written to assign probabilities to parses of the sentence and then to determine the most probable parses for simple noun phrases. Two functions were written to assign probabilities to the words and multi-words of a sentence.

ProbCompare

A program called “ProbCompare” was written in the C++ programming language to determine the performance metric of the stochastic and non-stochastic parsers. Over 200 sentences were manually parsed to use as the "correct" simple noun-phrases for this program. Extracted simple noun phrases were compared against the correct noun phrases in this program. After determining the number of correct, incorrect or partially correct noun phrases, that were detected by the parsers, three performance metrics, Recall, Precision, and Figure-Of-Merit, were calculated.

Plotting Performance Metrics vs. Number of Sentences

To illustrate the variation of the performance metrics over various sentences, a Matlab program was written to plot “recall vs. sentence” and “precision vs. sentence”. These graphs were used to study the degree of relationships between the decision metrics and the parsing strategies(statistical and non-statistical) .

Chapter 2: Review of Relevant Work

In recent years, the artificial intelligence community has studied various stochastic behaviors of natural language (NL) to carry out successful information extraction processes. Many researchers have suggested that the statistical approach will be the next step to make computers understand NL. At AT&T, Jones and Eisner are trying to automate 250,000 English sentences, that specify the operational tests for a typical telephone switching system [13]. In automating the testing process, the main challenge is to extract at least the surface content of the naturally occurring English text using parts-of-speech parsers.

The objective of my research is to present a parsing algorithm that uses stochastic context-free grammar to extract simple noun phrases automatically from the English language. For comparison, the ASPIN grammar is used to extract simple noun phrases. In my research, U.S. patents of the technical documents regarding DMA controllers are used as natural language text-files in the probabilistic parsing algorithm to detect simple noun phrases. Charniak [2], Church [6], Jones and Eisner [13] and many others have used this approach previously to detect parts-of-speech using statistical data. This chapter reviews the work that has been carried out by Charniak [2& 3], Jason M. Eisner [11], and by Jones-Eisner [13] with various statistical parsing techniques to extract information from natural English language.

2.1 Statistical Parsing with a Context Free Grammar and Word Statistics

E. Charniak [3] investigated a parsing system for natural language using a statistical probabilistic grammar. The method that Charniak described in this paper [3] presents an attractive method for determining the most likely parses of a complete sentence considering the probabilities of the constituents or words forming the rule and probability of the rule itself.

Eugene Charniak presented a parser in which the grammars and probabilistic parameters were induced from a Penn Wall Street Journal tree-bank [17]. He described that the best parsing of an English text file is achieved assigning probabilities to possible parses of a sentence. In this model, a sentence is parsed using the traditional technology of context-free chart parsing. This model does not find all the possible parses for a sentence. Instead, this model finds the most probable parsing of the sentence.

The goal of Charniak's experiment was to determine the precision, which is how precisely this model can parse a sentence, and also to determine the recall, which gives a percentage of correctly detected parses for the sentence using a statistical grammar. This experiment collected the data for both precision and recall using different probabilities assigned differently. The results were then compared to determine which one is the better strategy.

The experimental results described by Charniak [3] showed that the superiority of an algorithm in parsing English sentences depends upon the extensive collection of statistics that it uses. The results also showed that the major determinants of performance in different parsing algorithms are the probabilities of the rules and words used by those systems. The results showed that including probabilities for the rule for a given head and the nonterminal word, the SimCollins [7] model can achieve precision/recall of 86.1/86, while the Magerman [14] model, which does not include this probability, achieves 84.9/84.6. Charniak's paper also confirms that a statistical model is superior to non-statistical models to interpret language.

In my thesis, I have used the same strategy that Charniak used for his statistical parsing with the probabilistic context-free-grammar (PCFG). My thesis uses the probabilities also generated by the same Penn Wall Street Treebank for context-free-grammar rules to determine noun phrases over English text documents. The initial PCFG used here was obtained from Charniak in a private communication. However, my thesis compares the recall and precision for extracted simple noun phrases (not all the parts-of-

speech tags) that are obtained using two different grammars (probabilistic context-free-grammar (PCFG) and traditional context-free-grammar (CFG).

2.2 Statistical Techniques for Natural Language Parsing

Eugene Charniak [2] in this paper has reviewed the current statistical work on syntactic parsing and part-of-speech tagging. The traditional parsing technique is compared with the statistical parsing technique in this paper. Though the probabilistic context-free grammar (PCFG) is an easier extension of the traditional context-free grammar (CFG), Charniak showed that the PCFG is relatively "flat" (constituents contain fewer substructures) and is created using the rules' probabilities of the manually parsed sentences. For example, the Penn Tree-bank grammar [4], which achieves an average of about 75% precision and recall [3], is formed by 50,000 hand parsed sentences. Therefore, this tree-bank grammar holds all the possible grammar rules that form those 50,000 sentences. As a result, the possible parses that this tree-bank generates can never represent the accurate parsing of a sentence that has a different rule structure than any of those 50,000 sentences.

Charniak [2] concludes in his paper that an alternative to the misparsing sentences that arises from the scarcity of the right rule in the tree-bank is to invent new grammar rules. Generated using the Hidden Markov Model [8 & 14], these new grammar rules are known as "Markov grammars" [2]. Charniak provided the formula, shown in Equation 2.1, to calculate the probability of the new rule based on the probabilities of the previous rule constituents.

$$p(r \mid l) = \prod p(t_i \mid l , t_{i-1}) \; ; for \; t_i \in r \qquad (2.1)$$

where t_{i-1} represents the previous constituent of this rule, l represents the constituent type, and r represents the rule itself. In other words, Equation 2.1 states that the probability of a rule r, given that the constituent type is l, can be calculated by

multiplying the joint probabilities of the current constituent and the previous constituents. Though this paper proposes a mathematical way to calculate the probabilities of new rules that are not included in tree-bank grammars, no formal studies on how well the Markov grammars perform in comparison to the tree-bank grammars are mentioned. Instead, Charniak [2] claims, " For non-lexicalized parsers, tree-bank grammars work slightly better."

My thesis relies on the probabilistic context-free-grammar (PCFG) to extract simple noun phrases from English natural language specifications. However, my algorithm for automatic extraction of noun phrases is used for the US patents of the technical documents regarding DMA Controllers. Therefore, the number of ambiguous noun-phrases in these US patents is negligible, not as in English literature. My thesis work also calculates the recall and precision for the extracted noun phrases for these documents.

2.3 Three New Probabilistic Models for Dependency Parsing

Jason M. Eisner [11] has presented a probabilistic parser that not only assigns tags to the parts-of-speech, but also assigns a bare-bones dependency structure.

Eisner proposed three distinct, lexical hypotheses about the probability space underlying a sentence structure. The experimental results comparing three probabilistic models of parsing found in Eisner's research were obtained analyzing sentences from the Wall Street Journal. The key attraction of his work is that, his proposed probabilistic parsing models do not require any hand-written grammar; the parser is trained on an annotated corpus.

The first model, called "Bigram Lexical Affinities", generates a sequence of tags according to Markov process, with the random choice of each tag conditioned on the previous two tags. Church [6], Jelinek [12], and Merialdo [18] have also considered this

approach to tagging sentences. However, the Eisner [11] model provides one more step in analyzing each sentence. This model assumes that the probability of the word *j* gets linked to word *i* is lexically sensitive and is dependent on the <tags, words> pairs at both *i* and *j*. The probability of drawing a given parsed sentence from the population is selected in this model using the Equation 2.2. In this equation, "link presences and absences" is a random variable and can be either 1 or 0.

$$Pr(words, tags, links) = Pr(words, tags) * Pr(link\ presences\ and\ absences \mid words, tags) \quad (2.2)$$

The second model called "Selectional preferences" assumes that in a legal dependency parsing, every word except for the head of the sentence has exactly one parent. Unlike the first model, this model does not generate all the possible parses to discard later. Instead, this system generates a sequence of tagged words to specify a parent for each word. The third model, known as "Recursive Generation", generates a Markov sequence of <tags, words> pairs to serve as a left children of the newly added word. This is a sort of lexicalized context-free model.

All three models are compatible with a "Bottom-Up" parser. Using these three models, Eisner reported the percentage of nouns correctly tagged to be 90.1%, 89.8% and 90.2%. My research is also based on a bottom-up parser, which selects the parses based on the probabilities of grammar-rule and words. Though the probabilities of words are not selected from any statistical corpora, the approach I used in selecting the best parse is similar to the first model of Eisner.

2.4 A Probabilistic Parser and Its Application

Mark A. Jones and Jason M. Eisner [13] described a general approach to the probabilistic parsing of context-free grammars. Jones and Eisner presented a parser that constructed accurate syntactic and semantic analysis for the sentences of a given

language. Adopting the "bootstrapping" approach, Jones and Eisner started with a small manually-parsed set of sentences. Then their parser provided the analyses for further sentences based on the previous manually parsed sentences. In this paper, Jones and Eisner claimed that their parser finds and identifies the correct parse in 96% of the sentences for which the parser finds any parse using joint syntactic and semantic statistics [13]. However, without the knowledge of the semantic statistics, this success rate is 79%.

A research objective of Jones and Eisner was to generate at least one parse for a sentence and to identify the correct parse. In addition, the time required by the parser to accumulate the knowledge necessary to get at least one parse (correct or incorrect) was also taken into consideration. The system described by Jones and Eisner uses the modified most probable (MP) criterion to select the best parse. The MP criterion calculates the probability of the parse tree considering the left context of the tree. On the other hand, the modified MP not only adds the most likely tree in each set, but it also adds all trees within some fraction ε ($0<\varepsilon<1$) of the most likely one. Jones-Eisner parser calculates the probability of a new incomplete parsing tree depending on the a-priori probabilities of previous child trees.

The key feature of their parser is the effective caching of all the left-context probabilities for trees that are already in the chart. Once the priori-probabilities are cached, the probability of a next dotted tree can be computed with the help of well-known Bayes' Theorem. For example, Equation 2.3 gives the probability of the incomplete dotted tree $h_{j,l}$, given that all the previous sub-string w_l has already occurred.

$$\begin{aligned} Pr[h_{j,l} \mid w_l] &= Pr[\, h_{i,j} \,\&\, h_{j,l} \mid w_l \,] \\ &= Pr[h_{i,j} \mid w_l] \,.Pr[h_{j,l} \mid h_{i,j} \,\&\, wj] \\ &= X_1 * X_2 \end{aligned} \tag{2.3}$$

In this equation,

$h_{j,l}$ = the next incomplete dotted tree;

w_l = the sub-string of length l in the input sequence;

$h_{i,j}$ = the previous incomplete dotted tree; and

X_1 & X_2 = the conditional probabilities to left-context trees.

A new figure of merit called "corpus perplexity" is also introduced in this paper. The corpus perplexity, *b(C)* measures the ambiguity of any set of sentences *C* under a given grammar, where *S* is a sentence of the set *C*:

$$\log b(C) = \frac{\sum_{s \in C} \log (\text{number of Parses for } S)}{\sum_{s \in C} (\text{number of words in } S)} \qquad (2.4)$$

Jones-Eisner parsing algorithm found the corpus perplexity *b(C)* to be 1.313. That means, their parser introduces 20 parses for a typical 11-word sentence, only one of which is correct.

Finally, in this paper, Jones and Eisner conclude that both the distributional knowledge and the general linguistic knowledge, that are available to humans, will help their bootstrapping approach to enable the parser to learn more directly from an unbracketed corpus quickly and accurately. The parsing algorithm presented in this paper used a bootstrapping approach unlike the parsing approach that I used in my research. My research selects the most likely parses of noun-phrases based on the probabilities associated with the parses. To calculate the probability of a parse, my research considers the grammar rule-probability and does not take account the syntactic probability. Jones and Eisner reported 79% success rate [6] in parsing sentences correctly without the semantic statistics; my research shows 97% to 99% average success rate in recalling the correct parses for simple noun phrases. In my research, my probabilistic parsing algorithm selects only 1 parse per sentence eliminating all other parses that have lower probability and smaller span. Whereas, Jones-Eisner parser finds 1.03 parses per sentence based on the probability of the syntactic and semantic knowledge [13].

Chapter 3: Background

This chapter focuses on the theory of the probabilistic chart parsing, which has been my research objective. In addition, this chapter also discusses the concepts and terminology related to this research. Probabilistic chart parsing is an extended version of non-stochastic parsing. Since, my research extracts simple noun phrases from English documents using a statistical grammar, this chapter also includes a discussion on statistical English language grammar and on parts-of-speech of the English language.

3.1 Terminology

This section includes a discussion of terminology used in the English language and in parsing sentences. In order to extract information from a language, understanding the basic linguistic concepts of that language is necessary.

3.1.1 Lexical Categories

A basic unit of every language is its sentence. Sentences in English language consist of grammatically structured words. Linguists have grouped these English words into classes that show similar syntactic behavior and often a typical semantic type [21]. These word classes are known as parts-of-speech (POS). There are eight basic parts-of-speech in the English language. The two most important POS are *noun* and *verb*. Table 3.1 provides a brief description of these parts-of-speech of English language.

Table 3.1 Listing of 8 Parts-Of-Speech of English Language

Part of Speech	Description [13]	Examples
Noun	Words that refer to the names of persons, places, things or concepts.	man, tree, courage, controller, dignity, specification, thesis.
Pronoun	Words that can replace nouns or can be used instead of nouns.	he, someone, they, them,
Verb	Words that express the actions or states in a sentence.	appear, become, send, sent, leave.
Adjective	Words that describe the properties of nouns.	large, best, principal, distinguished
Adverb	Words that modify verbs, adjectives and other adverbs.	quickly, clearly, however, very, not
Preposition	Words that indicate the relationships between the nouns or pronouns and some other parts of speech [13].	at, in, with, on, under.
Conjunction	Conjunctions join clauses, parallel nouns, and adjectives.	and, or, until, when, because.
Determiners	Words that specify the referents of the phrases.	a, an, the this, that

3.1.2 Simple Noun Phrases

A noun phrase is a syntactic unit of the sentence in which information about the noun is gathered [16]. The embedded noun is the *head* of the noun phrase and is the central constituent that determines the syntactic character of the phrase. This research concentrated on extraction of simple noun phrases from English documents. A **simple noun phrase** is defined in this thesis as a phrase that can have pre-modifiers, post-modifiers and a head noun but can not contain conjunctions or prepositional phrases. The pre-modifiers of a simple noun phrase can be determiners, adjectives, and adjectival phrases. The post-modifiers of the simple noun phrase can be adverbs, adjectival phrase, or some relative clauses. Table 3.2 shows some examples of simple noun phrases.

Table 3.2 Simple Noun Phrases

Simple Noun Phrase	Modifiers & Head-Noun
The DMA controller	Pre-modifier (Determiner and Adjective): "The DMA" Head-noun: "controller"
The microprocessor 8086	Pre-modifier (Determiner): "The" Head-noun: "microprocessor" Post-modifier (identifier): "8086"
a data processing apparatus	Pre-modifier (Determiner and Adjectival-phrase) : "a data processing" Head-noun: "apparatus"
The controller sending the data	Pre-modifier(Determiner): "The" Head-noun: "controller" Post-modifier (Relative clause): "sending the data"
US Patent Number 65456	Pre-modifier (Adjectival-phrase): "US Patent" Head-noun: "Number" Post-modifier (identifier): "65456"

3.2 English Language Grammar

A language consists of the set of strings (or utterances) that the language users accept. This description of language is not useful for either automatic processing or for teaching. A language can be approximately described by a grammar. Actually, a grammar defines formal language. A formal language is defined over an alphabet as a subset of the set of all strings obtained by concatenating one or more symbols from the alphabet [10]. The "alphabet" of a language is the set of all possible indivisible symbols of that language.

A formal grammar **G** is defined as a quadruple **G = <V, T, S, P >.** In this quadruple, **V** represents the set of non-terminal symbols, **T** represents the set of terminal symbols (the alphabet), and **S** represents the "sentence" or top-level non-terminal construct. **P** represents the finite set of grammar rules. In this thesis, the terminal symbols consist of English words and punctuation marks. The top-level nonterminal is **np,** which stands for "noun-phrase". In this research, I have used two English grammars to parse noun-phrases. These are a Context-Free Grammar (CFG) and a Stochastic Context-Free Grammar (SCFG). This section includes a brief discussion on the syntactical structures of these grammars.

3.2.1 Context-Free Grammar (CFG)

A context free grammar consists of grammar rules of the form:

$$\mathbf{X} \longrightarrow \omega \qquad (3.1)$$

where **X** is a nonterminal and **ω** is a sequence of terminals, **T** and nonterminals, **V**. Figure 3.1 shows several context-free grammar rules.

np	→	det	np
np	→	noun	noun
pps	→	prep	np
prep	→	*in*	
det	→	*the*	
noun	→	*DMA*	
verb	→	*controller*	

Figure 3.1 Simple Context-Free Grammar

For example, some nonterminal symbols and constituents used in the prepositional phrase "in the DMA controller" are shown in Figure 3.2.

Nonterminals		Examples
Determiner	det	"the"
Preposition	prep	"in"
Prepositional-Phrase	pps	"in the DMA controller"
Noun-Phrase	np	"DMA controller"

Figure 3.2 Nonterminal Symbols

The terminal words of this sentence-fragment are "in", "the", "DMA" and "controller". Using the grammar of Figure 3.1, this fragment can be represented by the subtree. This subtree is shown in Figure 3.3.

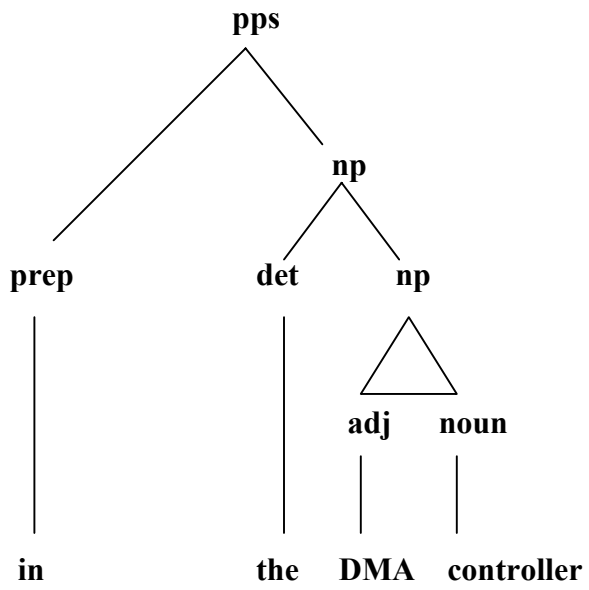

Figure 3.3 Parse-Tree for the Words "in the DMA controller"

Grammars approximating natural languages tend to be ambiguous and assign multiple parses to a sentence. This can be seen in the Figures 3.4 and 3.5, which show two possible parses of the noun-phrase "a host machine". The computers using only the knowledge of CFG may not be able to extract the correct parse in this case.

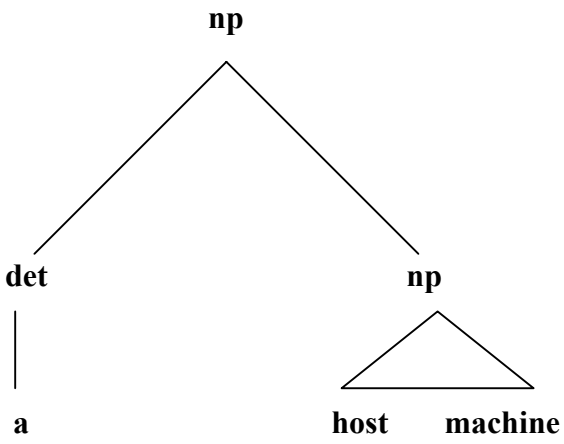

Figure 3.4 A Possible Parse-Tree for "a host machine"

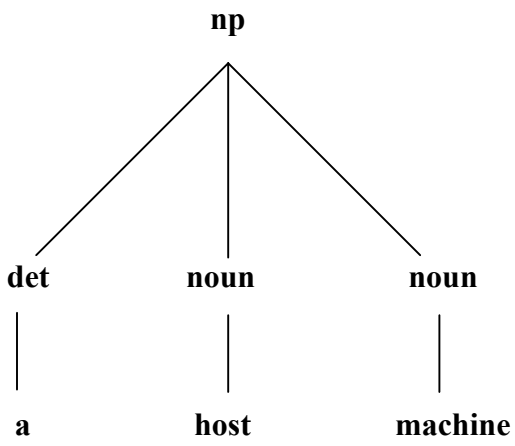

Figure 3.5 Another Possible Parse-Tree for "a host machine"

3.2.2 Stochastic Context-Free Grammar (SCFG)

Each rule in a SCFG has an associated probability. The rule probability signifies the likelihood that the rule is used in parsing a sentence relative to all rules with the same left-side non-terminal. Equation 3.2 shows the mathematical expression for an SCFG rule.

$$(\quad X \longrightarrow \omega \quad , \quad p \ ) \qquad (3.2)$$

where **X** is a nonterminal, **ω** is a sequence of terminal symbols **T** and nonterminal symbols **V** of the rule and **p** is the rule's probability.

English language itself is ambiguous; the same word can appear to be different part-of-speech in sentences. For example, the terminal symbol "control" can be an adjective (in "the control signal"), a noun (in "the control of the DMA") and a verb (in "to control the signal") depending on its placement in a sentence. In other words,

uncertainty or probability is an inherent characteristic of English language. Therefore, providing the statistical data related to each rule, the SCFG helps the stochastic parser analyze all the possible parse trees for a single sentence with greater accuracy and reliability. Figure 3.6 shows several typical rules for a SCFG derived from the Penn Tree-Bank [17] that consists of 4.5 million words of successfully parsed English sentences.

S	→	np	vp	.	0.193518
adjs	→	adv	adv	adj	0.0036083
np	→	noun	adj		3.20769e-05
np	→	noun	nounp		0.0256375
pps	→	prep	np	noun	1.05608e-05

Figure 3.6 Stochastic Context-Free Grammar

In the above-mentioned SCFG rules, the second entry signifies that a nonterminal "adjs" (adjective phrase) found by this rule in a sentence, has a probability of 0.0036083 relative to the other adjs rules.

3.3 Grammatical Tagging

The first step in automatic extraction of information is to perform an automatic grammatical part-of-speech tagging. This research uses two grammars: a SCFG and a CFG. The original stochastic grammar used the Penn Tree-Bank tagset [17]. On the other hand, the CFG used the ASPIN [9] tagset. Both of these tag-sets were designed for English, and are described briefly in this section.

3.3.1 ASPIN (CFG) Tagset

The ASPIN Tagset was originally generated to use along with a simple context-free (CF) parser. This tagset is simpler and smaller than the Penn Tree–Bank tagset. The ASPIN tagset does not include detailed categories of words. For instance, this tagset does

not have any tag to distinguish between the comparative and superlative forms of adjectives. Excluding the second order complexity in words' categories and including only traditional parts-of-speech, common phrases and punctuation, the ASPIN tagset provides an efficient way to parse and to extract simple noun-phrases. The ASPIN tagset contains 23 POS tags and 5 syntactic tags.

3.3.2 Penn Tree-Bank (SCFG) Tagset

The Penn Tree-Bank Tagset is a simplified version of the Brown tagset (The Brown University Standard Corpus). The Brown tagset has 87 simple tags. The key strategy of building the Penn Tree-Bank tagset was to eliminate redundancy by taking into account both the lexical and syntactic information [17]. Thus, many POS tags used in the Brown Corpus tagset were eliminated from the Penn Tree-Bank tagset. For example, the Penn Tree-Bank tagset does not distinguish between the subject pronouns from object pronouns, but the Brown corpus does. The Penn Tree-Bank tagset contains 12 punctuation and currency symbols which is included in the Appendix C. In addition to these 12 tags, the Penn Tree-Bank also contains 36 POS (Parts-Of-Speech) tags. Table C.2 of the Appendix C shows these 36 parts-of-speech tags. The Penn Tree-Bank uses several syntactical tags also to denote the phrases and clauses of English documents. In addition, a brief description of the phrasal notations of the Penn Tree-Bank syntactic tags is also included in the Table C.3 of the Appendix C.

A tagset should successfully encode not only features of the classification, telling the user the useful information about the grammatical class of word, but also the predicative features to predict the behavior of other words in the context. In this regard, the Penn Tree-Bank is an attractive tagset. Since parts-of-speech (POS) can be motivated on semantic, syntactic or morphological details, there is an on-going debate on POS tags. The Penn Tree-Bank POS tags provide very distinguished performances in NLP (Natural Language Processing) applications. Charniak's probabilistic grammar [4] uses the Penn Tree-Bank tags with several modifications. Assuming that the auxiliary verbs have a

very distinctive distribution, this grammar uses an additional "AUX" tag with the auxiliary verbs. Figure 3.6 shows an example of Penn Tree-Bank. This example illustrates most of the major features of trees in the Penn Tree Bank data and is taken from Penn Tree-Bank illustration [17]. The lower levels of the parse-tree such as POS tags are not shown.

```
( ( S  (NP-SBJ   The move)
         (VP   followed
             (NP (NP  a round )
                 (PP of
                    (NP (NP similar increases)
                            (PP by
                               (NP other lenders))
                                 (PP against
                                    (NP Arizona real estate loans)))))
             ,
             (S-ADV (NP-SBJ *)
                       (VP reflecting
                            (NP (NP a continuing decline)
                                 (PP-LOC in
                                        (NP that market))))))
.) )
```

Figure 3.7 A Penn Tree-Bank Tree.

This figure shows the parse tree for a sentence "The move followed a round of similar increases by other lenders against Arizona real estate loans, reflecting a continuing decline in the market." using Penn Tree-Bank tagset. As is observed from the above figure, this tree-bank attempts to explain grammatical and semantic details.

To compare results between the ASPIN and Charniak's grammars, the Penn Tree-Bank tags were converted to the ASPIN tagset. Table 3.3 lists the changes of the 36 POS tags of the Penn Tree-Bank into 20 POS ASPIN grammar tags. In addition, a brief description with examples of these tags is included in this table. Table 3.4 lists the 5 syntactic tags of the ASPIN grammar and corresponding 5 syntactic tags of the Penn Tree-Bank.

Table 3.3 Changes of Penn Tree-Bank POS Tags to 20 ASPIN POS-Tags

Tag No.	Penn Tree-Bank tags	ASPIN POS tags	Description	Examples
1	CD	#	Cardinal Number	one, 3, fifteen
2	JJ, JJR, JJS	adj	Adjective, Comparative, Superlative	good, better, best
3	RB,RBR,RBS	adv	Adverb, Comparative, Superlative	however, faster, fastest
4	DT	det	Determiner	the, a, this
5	CC	conj	Conjunction	and, but, or
6	FW, SYM, X	id	Identifier, Unknown Words, Foreign Words, Symbols	—
7	MD	mod	Modal	can, could, will
8	NN,NNP	noun	Simple Singular Noun, Proper Singular Noun	data, processor
9	NNS, NNPS	nounp	Plural Noun, Proper or Simple	controllers, buffers, books
10	CD	ord	Ordinal Number	first, second, fifth. sixth
11	PDT	pdet	Pre-Determiner	only, each of, one of
12	IN	prep	Preposition	in, of, with
13	PRP, PRS, WP, WPS	pron	Pronoun, Personal and Possessive; Wh-pronoun, Possessive Wh-pronoun	he, these, some, whom, which
14	SBAR SBARQ	scon	Subordinate Conjunction	when, whenever, where, whereas, whereby, while, wherein, whilst whereupon
15	EX	there	There	there
16	TO	to	To	to
17	VBP, VBZ	verb	Verb-Singular form, Non-3rd Person, 3rd Person	works, contains
18	VB	verbp	Verb-Plural form or Base form	work, contain
19	VBD,VBN	ven	Verb, past or past participle	worked, contained
20	VBG	ving	Verb, Gerund	working, containing

Table 3.4 Changes of Penn Tree-Bank Syntactic Tags to ASPIN Tags

Tags No.	Penn Tree-Bank Tags	ASPIN Tags	Description
1	ADJP	adjs	Adjective Phrase
2	ADVP	advp	Adverbial Phrase
3	NP	np	Noun Phrase
4	PP	pps	Prepositional Phrase
5	VP	vp	Verb Phrase

3.4 Grammar Rules for Simple Noun Phrases

This section includes the grammar rules required by the probabilistic SCF parser and CF parser to extract simple noun-phrases. As was mentioned earlier, the Penn Tree-Bank tagset of the stochastic context-free grammar has been replaced by the ASPIN tagset.

The ASPIN Grammar

The ASPIN grammar [9] used in this thesis to extract simple noun phrases is shown in Figure 3.8.

The Probabilistic Context-Free Grammar

The probabilistic context-free grammar used in this thesis has a total of 1624 rules only to extract simple noun phrases. As was mentioned earlier, the Penn Tree-bank tags featuring the original probabilistic grammar were replaced by the ASPIN tags. Figure 3.9 lists some of these probabilistic grammar-rules.

np	→	pdet	det	adjs	head	
np	→	pdet	det	head		
np	→	pdet	adjs	head		
np	→	pdet	head			
np	→	det	ord	#	adjs	head
np	→	det	ord	#	head	
np	→	det	ord	adjs	head	
np	→	det	ord	head		
np	→	det	#	adjs	head	
np	→	det	#	head		
np	→	det	adjs	head		
np	→	det	head			
np	→	ord	#	adjs	head	
np	→	ord	#	head		
np	→	ord	adjs	head		
np	→	ord	head			
np	→	#	adjs	head		
np	→	#	head			
np	→	adjs	head			
np	→	head				
np	→	head	range			
np	→	np	#			
head	→	noun				
head	→	id				
head	→	noun	head			
head	→	id	head			
adjs	→	adj				
adjs	→	adj	adjs			
adjs	→	adj	,	adjs		

Figure 3.8 Noun Phrase Grammar Rules

Results		Constituents			Probability
np	→	det	noun		0.148146
np	→	noun			0.123495
np	→	nounp			0.064068
np	→	noun	noun		0.066792
np	→	det	adj	noun	0.049849
np	→	adj	nounp		0.039799
np	→	det	nounp		0.025704
np	→	adj	noun		0.025916
np	→	noun	nounp		0.025638
np	→	det	noun	noun	0.032025
np	→	#			0.017428

Figure 3.9 Probabilistic Noun Phrase Grammar Rules

3.5 Chart Parsing

Chart Parsing is an algorithm to find the structure of a sentence of a language according to a context-free grammar (CFG). A chart parser parses each sentence of English language accepting one token (or word) at a time, and then creates a chart. A chart is a labeled graph, which consists of edges and nodes. An example of a chart parsing is shown in Figure 3.10. As is seen in this figure, a chart can be thought as a graph of nodes representing points between words in a sentence, linked by edges representing words and constituents.

From this figure, it is evident that a chart parser will create as many edges as the parser can satisfy the grammar rules to fit the structure of the sentence. For example, the phrase “The DMA controller” has been parsed with five sets of grammar rules.

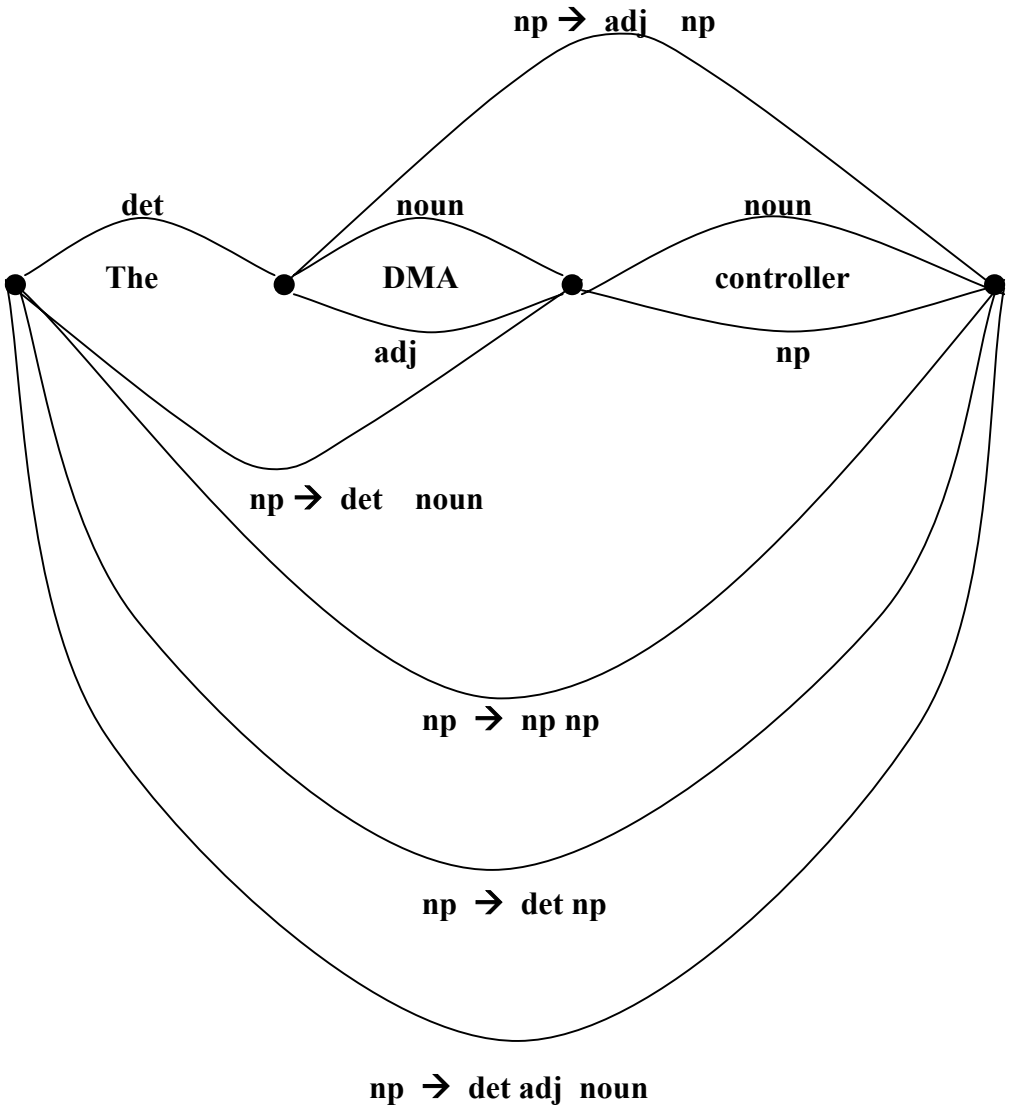

Figure 3.10 Chart Parsing for "The DMA controller"

3.5.1 Stochastic Chart Parsing

A stochastic chart parser also uses the same principles to generate parse-trees for words as well as sentences. However, a stochastic chart parser does not need to store all the possible parses of a particular phrase (or words). Rather, only the most likely parses are stored. Depending on the available statistical knowledge, a stochastic parser can

reduce the number of edges created considerably. The statistical approach to find the most likely parses with increasing precision and decreasing number of edges has been considered so far to be the most promising approach in information extraction techniques. Looking at the previous example of traditional chart parsing, it can be realized that the size of a chart can be very large depending on the number of words and the number of grammar rules. Even for a single word like "DMA", the parser created three chart entries: "adj", "noun" and "np" (shown in Figure 3.10). However, using the statistical parser, the number of chart entries can be reduced to two. Figure 3.11 shows chart parsing using the statistical parser. The probability associated with each parse is shown within the parenthesis. Since each chart entry takes part in the next level of parsing, so the reduction in chart entries using statistical knowledge is not only required for precision but also for the speed of the computer.

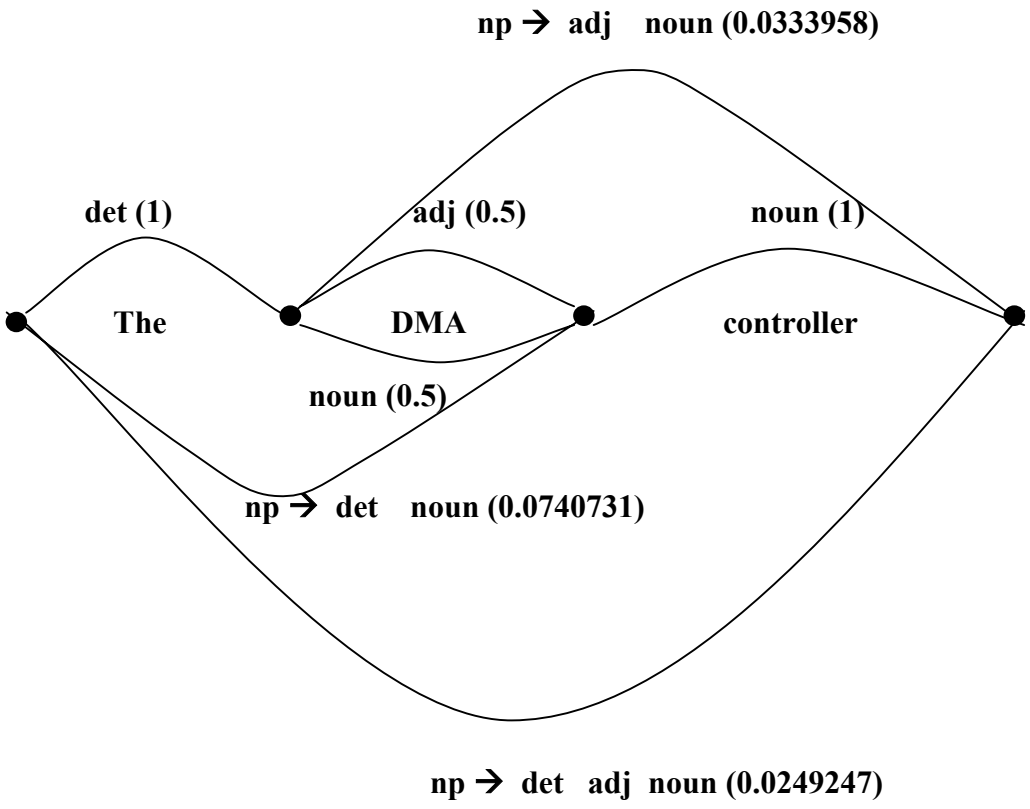

Figure 3.11 Stochastic Chart Parsing for "The DMA controller"

3.6 Theory of a Stochastic Chart Parser

This section focuses on the theory of a stochastic chart parser. The parsing algorithm to extract the most likely noun phrases is also discussed. As was mentioned earlier, probabilistic chart parsing is an algorithm to find the most probable structures for a sentence of a language according to a stochastic context-free grammar (SCFG). A stochastic parser finds the most probable parses using the available statistical knowledge. This section describes the general theory behind a stochastic parser and points out the limitations in adopting such an approach.

3.6.1 Finding the Observation-probability in Stochastic Chart Parsing

This thesis concentrates on the extraction of simple noun phrases from the English language specifications using a stochastic parser. In this section, an example is presented to find the probability of a simple noun phrase with the help of the priori-probabilities of states and words. Equation 3.3 shows the probability of finding a simple noun-phrase given a rule **R**, terminals $\mathbf{T_i}$ and non-terminals $\mathbf{V_i}$ by conventional calculation

$$P(\text{np Rule Application}) = P(\text{np-Rule}) * \prod_i P(V_i) * \prod_i P(T_i) \qquad (3.3)$$

To avoid the situation where a sub-tree of a parse tree having much higher probability than that of the parse-tree is selected instead of the parse-tree, the parse implemented here does not include the grammar rule-probabilities of the parse trees in evaluating the probability of the parent tree. After creating a chart of most likely parses for simple noun-phrases, the parser selects the longest possible parse for a phrase in a bottom-up fashion. Equation 3.4 shows the probability of finding a simple noun-phrase.

$$P(\text{np Rule Application}) = P(\text{np-Rule}) * \prod_i P(T_i) \qquad (3.4)$$

Let us consider a simple noun-phrase "the DMA controller". For this noun-phrase, we have a sequence of three terminals ***{"the", "DMA", "controller"}***. Equation 3.4 can be simplified for this noun-phrase as follows:

$$P(np \mid the\ DMA\ controller) = P(np\text{-}Rule) * P(det \mid the) * P(adj \mid DMA) * P(noun \mid controller)$$

This equation can be evaluated for all the possible parses. Table 3.5 lists three possible stochastic grammar rules used by the program to parse this noun-phrase.

Table 3.5 Stochastic Grammar Rules for "The DMA controller"

Rule No.	Tags	Probability	No. Of Terms	Constituents
6	np	0.0498494	3	det adj noun
17	np	0.0123538	2	np noun
231	np	6.13676e-005	2	det np

Figures 3.12, 3.13 and 3.14 show three possible-parses obtained for this simple noun phrase using these three stochastic context-free grammar rules by the program. The rule probabilities are also shown in these parse-trees within parenthesis.

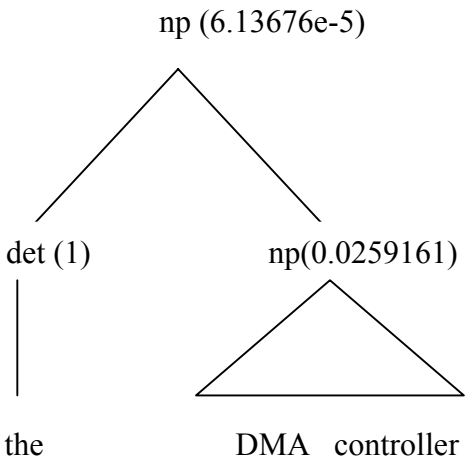

Figure 3.12 First Parsing Approach

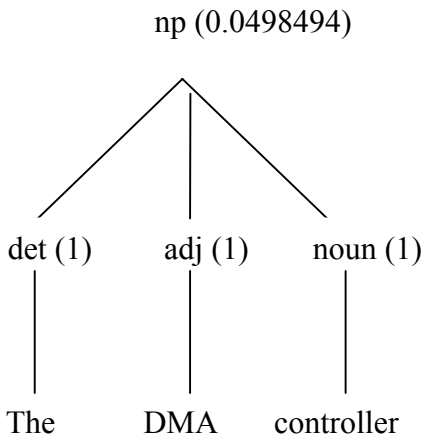

Figure 3.13 Second Parsing Approach

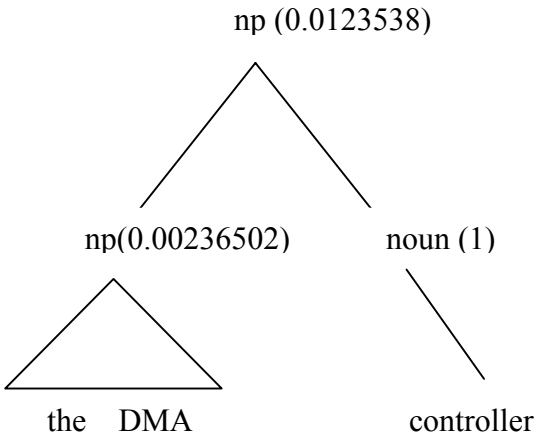

Figure 3.14 Third Parsing Approach

So, for the first case with the parse shown in Figure 3.12, the probability of the observation was found to be:

P (np =The DMA controller)

= P (np→ det np) ∗ P (det|the) ∗ P (adj|DMA) ∗ P (noun|controller)

= 6.13676e-005∗ 0.5∗1∗1 = 0.0000306838

So, ***P (np =The DMA controller)= 0.0000306838***…….....(3.5)

Similarly, for the second case with the parse shown in Figure 3.13, the probability of the noun-phrase was found to be:

P (np =The DMA controller)

= P (np→ det adj noun) ∗ P (det|the) ∗ P (adj|DMA) ∗ P (noun|controller)

= 0.0498494∗ 1.0 ∗ 0.5 ∗1.0

= 0.0249247

So, ***P (np =The DMA controller)= 0.0249247***……(3.6)

Finally, for the last case with the parsing shown in Figure 3.14, the probability of the observation is found to be:

P (np =The DMA controller)

= P (np→ np noun)∗[P (det|the) ∗ P (adj|DMA) ∗ P (noun|controller)]

= 0.0123538 ∗ 1.0∗0.5 ∗ 1.0 = 0.0061769

So, ***P (np =The DMA controller)= 0.0061769***…….…… (3.7)

Comparing the Equations 3.5, 3.6, and 3.7 which describe the obtained probabilities for the observed simple noun-phrase, it can be concluded that the second

parse tree with the highest probability of 0.0249247 is most likely to represent the mentioned simple noun-phrase and so was selected by the probabilistic parser.

3.6.2 Calculating the Probabilities of Parts-of-Speech

As was mentioned earlier, to calculate the probability of a parse tree, the knowledge of probabilities of parts-of-speech are necessary. This section describes the algorithm used to assign probability to the words of English language. Due to the unavailability of the statistical corpora of word-probabilities, this project assigned probabilities to the parts-of-speech using Equation 3.8:

$$P(X \mid w) = \frac{1.0}{\text{No. of Parts-of-Speech that can describe the word in the dictionary}} \quad \text{.....(3.8)}$$

where the $P(X|w)$ indicates the probability of part-of-speech X given a word w. This equation was used to assign probabilities to not only the words of the dictionary file, but also to the multi-words of the multi-word dictionary file. Let us consider the noun-phrase "a direct memory access system." The probabilities of the parts-of-speech that can describe the non-terminals of this phrase are listed in the Figure 3.15.

P(det | a) = 1;
P(adj | direct)= 0.5;
P(noun |memory) = 1.0;
P(noun | access)=0.5;
P(noun | system) = 1.0 .

Figure 3.15 POS-Probabilities of the Noun-Phrase

3.7 Performance Measures

In order to measure the performances of information-extraction strategies proposed by the researchers of the artificial intelligence community, several standard performance metrics have been developed. Three performance metrics are precision, recall and figure-of-merit. In this thesis, the results of the stochastic parser to extract simple noun phrases are compared with that of others' work in terms of these metrics. This section provides the mathematical expressions as well as descriptions for the performance metrics used in this research.

3.7.1 Positive, Negative, False Positive and False Negative Noun-Phrases

Figure 3.16 shows a Venn diagram of possible parsing outcomes. The right circle represents the "actual" set of simple noun-phrases that are assumed to be the "correct" noun phrases determined by human parsing. The left circle represents the extracted set of simple noun-phrases by an experiment. The rectangular area represents a set of all possible outcomes in parsing.

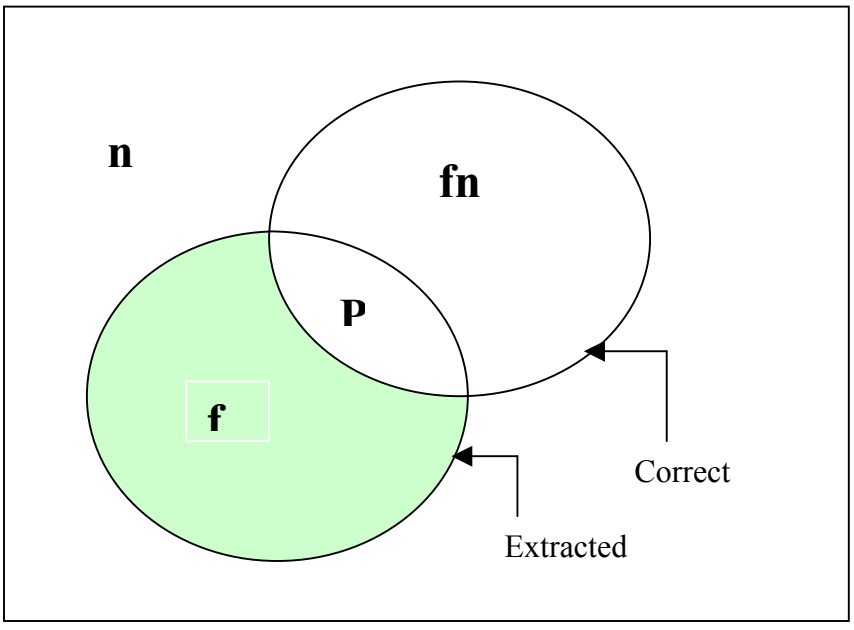

Figure 3.16 Measures of Precision and Recall

In the figure 3.16,

n represents the "true negative" word sequences that are not considered to be noun-phrases by humans,

p represents "true positive" noun phrases that are "correct" noun phrases by human parsing,

fp represents the "false positive" noun-phrases that are found by a parser but are not correct,

fn represents the "false negative" noun-phrases that are not detected by the parser but are correct.

The "Correct" simple noun-phrases and the "Extracted" ones can be thought of as the indicator random variables, where the "Extracted" phrases are the ones found by the parser. Then the joint probabilistic distribution of these two variables can be expressed as a 2∗2 contingency matrix [16]. This contingency matrix is shown in the Table 3.6.

Table 3.6 Contingency Matrix

SYSTEM	**Correct**	
	Included	Not Included
Extracted	**p**	**fp**
Not Extracted	**fn**	**n**

3.7.2 Recall

Recall is defined as the proportion of the actual noun phrases that the parsing system extracted from the English documents [16]. In simple words, what percentage of the total "Correct" noun phrases is correctly detected by the system is expressed by this metric. Mathematically, this metric is defined as in equation 3.9.

$$\textbf{Recall} = \frac{\textbf{p}}{\textbf{p} + \textbf{fn}} \qquad \textbf{(3.9)}$$

where, **p** and **fp** are the same symbols described above.

3.7.3 Precision

Precision is defined as the proportion of the correct noun phrases to the total number of simple noun phrases that the parsing system extracted from the English document [16]. In simple words, what percentage of the total noun phrases is correctly detected by the system is expressed by this metric. This metric determines the precision for the parsing system to extract simple noun phrases. Mathematically, this performance metric can be expressed as in the Equation 3.10:

$$\textbf{Precision} = \frac{\textbf{p}}{\textbf{p} + \textbf{fp}} \qquad \textbf{(3.10)}$$

where **p** and **fp** follow the same definitions as described in the previous section.

3.7.4 F-Measure

Figure-of-Merit, **F** is used to combine the recall and precision. If precision is defined as **P** and recall as **R** for a system, then this metric F-measure is defined as in the Equation 3.11 mathematically in terms of **P** and **R**.

$$F = \frac{P * R}{\alpha R + (1-\alpha) P} \qquad (3.11)$$

where, α is an weighting factor and is often assumed to be 0.5 to provide equal weightings to precision **P** and recall **R**. This performance metric is widely used to compare performances of parsing strategies.

3.7.5 Trade-Off between Precision and Recall

In the applications of information retrieval, there is generally a trade-off between the precision and recall. Usually, precision goes down while recall increases for a system. In addition, there is also a trade off between the **F-measure** and Precision **P** or Recall **R** for a system. The bias of **F-measure** is towards maximizing the true positive **p**, whereas the accuracy is sensitive only to the number of classification errors [16].

Chapter 4: Implementation

This chapter focuses on the implementation of the stochastic chart parser. The main classes of the programs "ProbChunker" and "ProbCompare", which were used to extract the most likely noun-phrases and to generate the performance metrics, are briefly discussed.

4.1 The Stochastic Parser Implementation

The stochastic parser is an extension of the chart parser, which was a part of a project called the Automated Specifications Interpreter (ASPIN) system [14]. Only the extensions to that parser are discussed here. The stochastic parser takes an unrestricted English document as its input. After loading the stochastic grammar, multi-word dictionary and dictionary, the parser creates a chart containing the selected parses along with their associated probabilities for each input sentence. Once the probabilistic chart is created, the parser selects the longest noun phrases found and outputs these phrases in a text file "nphrases.txt". In this section, a sample of the stochastic chart generated for an input text has been included. Also included has been an example showing how the probability of a parse was calculated for this project. Some samples of the input files such as dict.txt, mdict.txt and probabilistic grammar "probgramn.txt" are provided.

4.1.1 Samples of Dictionary, Multi-Word Dictionary and Grammar files

The first step in initializing the stochastic parser is to load a dictionary, a multi-word dictionary and stochastic grammar. The parser uses two types of dictionaries: a main dictionary and a multiword dictionary. The main dictionary defines 5200 words with their possible parts-of-speech. The multiword dictionary defines groups of two or

more words forming a lexical unit and provides their POS. The stochastic grammar contains 11060 grammar rules. After initializing the grammar and deleting the duplicate rules, probabilities are updated for the remaining 9509 grammar rules. A sample of these three input text files regarding the word-meanings, multi-word's meanings, and probabilistic grammar-rules are used along with the probabilistic parser to extract simple noun-phrases from English language. The formats of these three input-files are shown in three figures. Figure 4.1 shows the format of the stochastic context-free grammar.

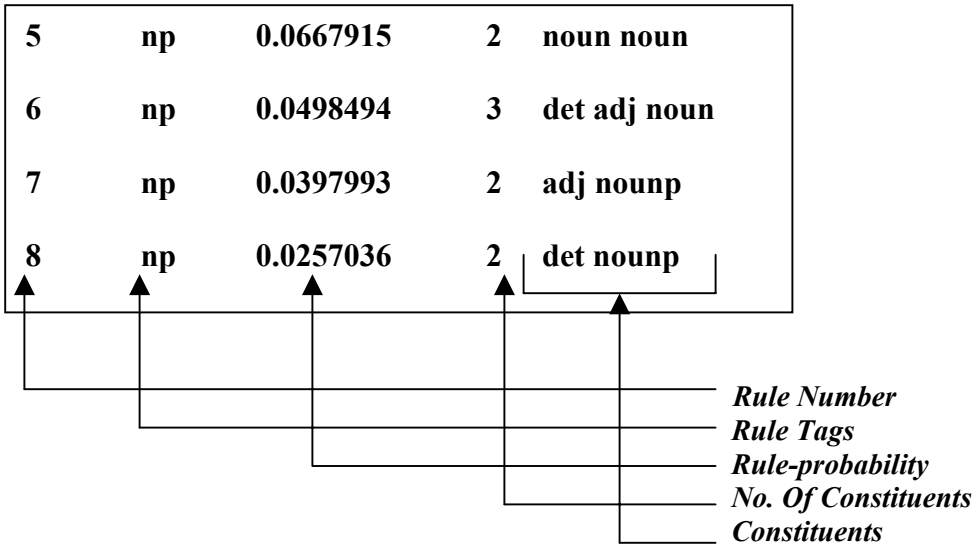

Figure 4.1 Stochastic Context-free Grammar

Figure 4.2 shows the format of the main dictionary containing the words and their meanings. The dictionary bears both the lexical and semantic categories for the words. Figure 4.3 shows the format of the multi-word dictionary, which contains the meanings for frequently used multi-words of English language. After the dictionaries and the grammar rules are loaded, an empty chart is initialized.

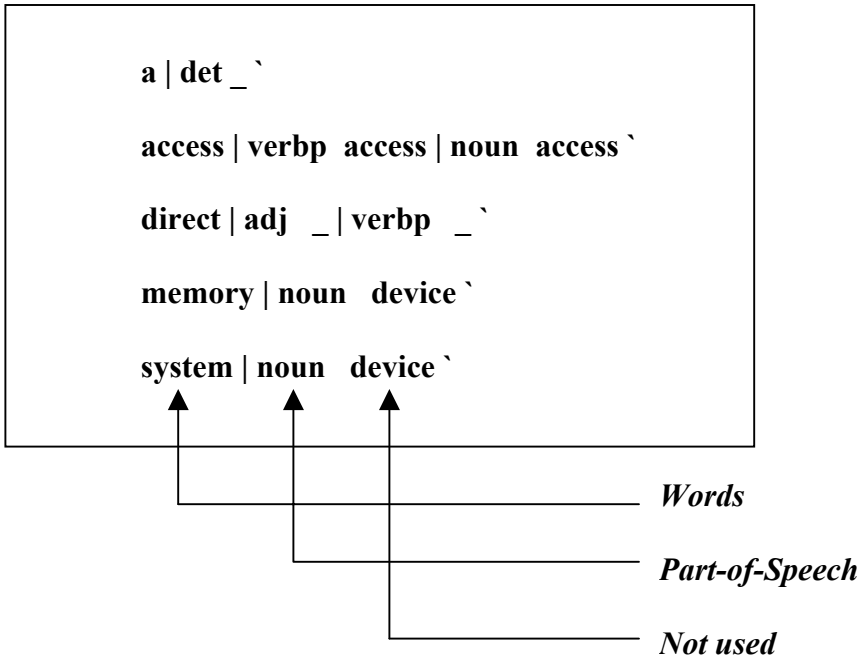

Figure 4.2 Main Dictionary

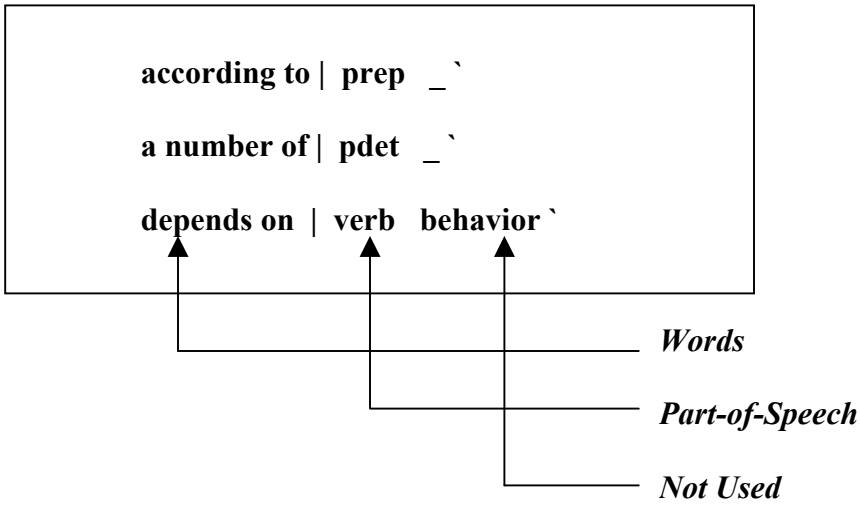

Figure 4.3 Multi-Word Dictionary

4.1.2 Samples of the Stochastic Phrases and Chart Lists

In this section, a partial chart and the partial chunks found by the parser for a sentence fragment "In a controller for a host machine" have been included. Looking at this sentence, we can easily deduce that there are two simple noun-phrases embedded in it. One is "a controller" and the other one is "a host machine". Figure 4.4 shows us the probabilistic chart that contains the most probable parses found for this fragment using this program.

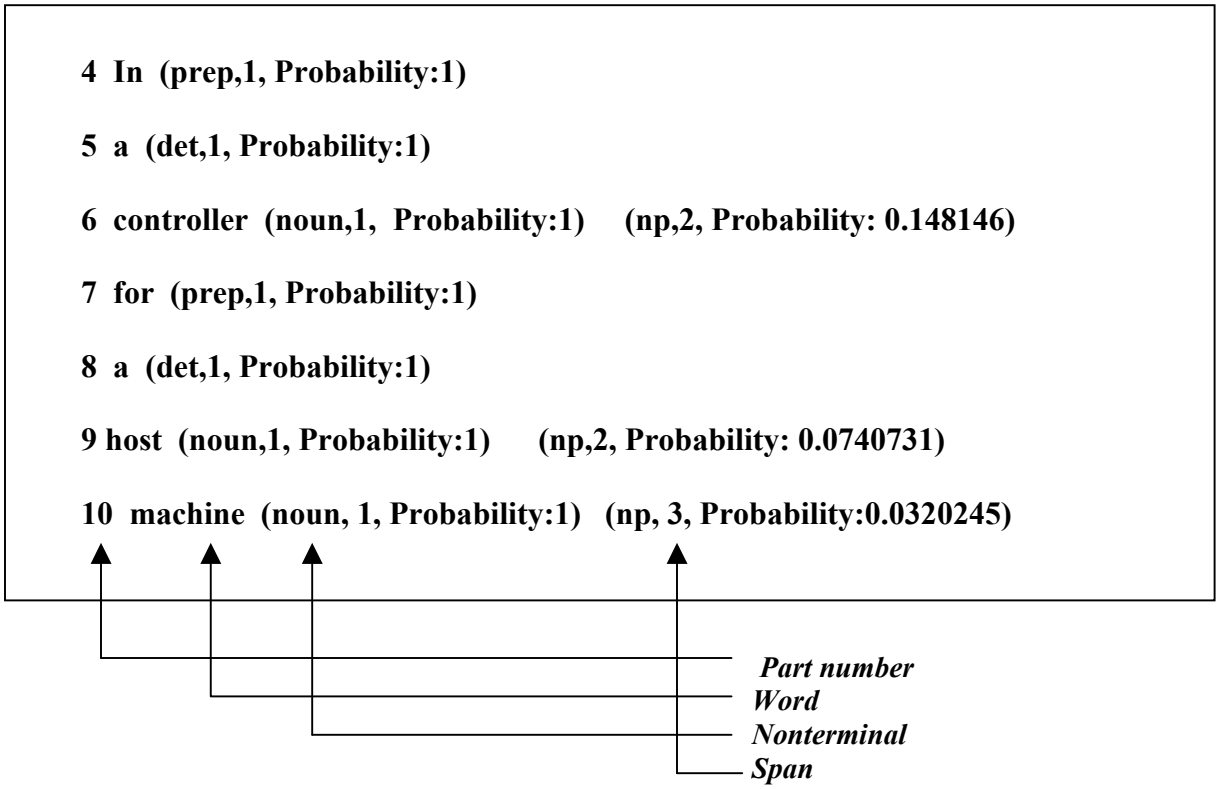

Figure 4.4 The Possible Most-Probable Parses for the Fragment

As seen in this figure, each part has a unique identity number starting from one for each new sentence. The span of each resultant nonterminal is noted within the parenthesis. For example, part 10 has an entry **(np, 3, Probability: 0.0320245)**, which indicates that the resultant nonterminal is a noun-phrase with a span of three and the last

node or word of this noun-phrase is “machine”. In other words, **(np, 3, Probability: 0.0320245)** entry confirms that the words “a host machine” create a simple noun phrase with a probability of 0.0320245. Since, the chart shows all the possible and most-likely phrases with different spans, from these phrases, the longest phrase representing a simple noun-phrase can be extracted. The most probable and the longest possible noun phrases found for the example-fragment are shown in Figure 4.5.

In	***prep(In) Probability 1.0 Rule –***
a controller	***np(det(a)noun(controller)) Probability 0.148146 Rule 1***
for	***prep(for) Probability 1 Rule –***
a host machine	***np(det(a)noun(host)noun(machine)) Probability 0.0320245 Rule 11***

Figure 4.5 The Longest Most-Probable Phrases for the Fragment

4.2 Main Classes of the ProbChunker Program

The primary class of ProbChunker is the CParser class. The CParser class uses the other main classes: CChart, CChunk, CDictionary, Cgramlist, CgramNode, Cconstitute, CLexer and CMWDictionary. Table 4.1 shows the C++ header and source files for this program that were used to implement the parsing algorithm described in Chapter 3.

Table 4.1 C++ Header and Source Files

The Parser	
Header Files	**Source Files**
Chart.h	Chart.cpp
Dictionary.h	Dictionary.cpp
Grammar.h	Grammar.cpp
Lexer.h	Lexer.cpp
MWDictionary.h	MWDictionary.cpp
CgramNode.h	CgramNode.cpp
Cconstitute.h	main.cpp
Parser.h	Parser.cpp

Figure 4.6 shows the class diagram of the stochastic parser. Booch class diagrams are used to show the class interactions for the stochastic parser. In these class diagrams, a class is represented by a dotted irregular outline. A line from one class to another, with a dot on the first, indicates that the first class uses objects of the second class [19]. An integer on the line shows the number of objects used.

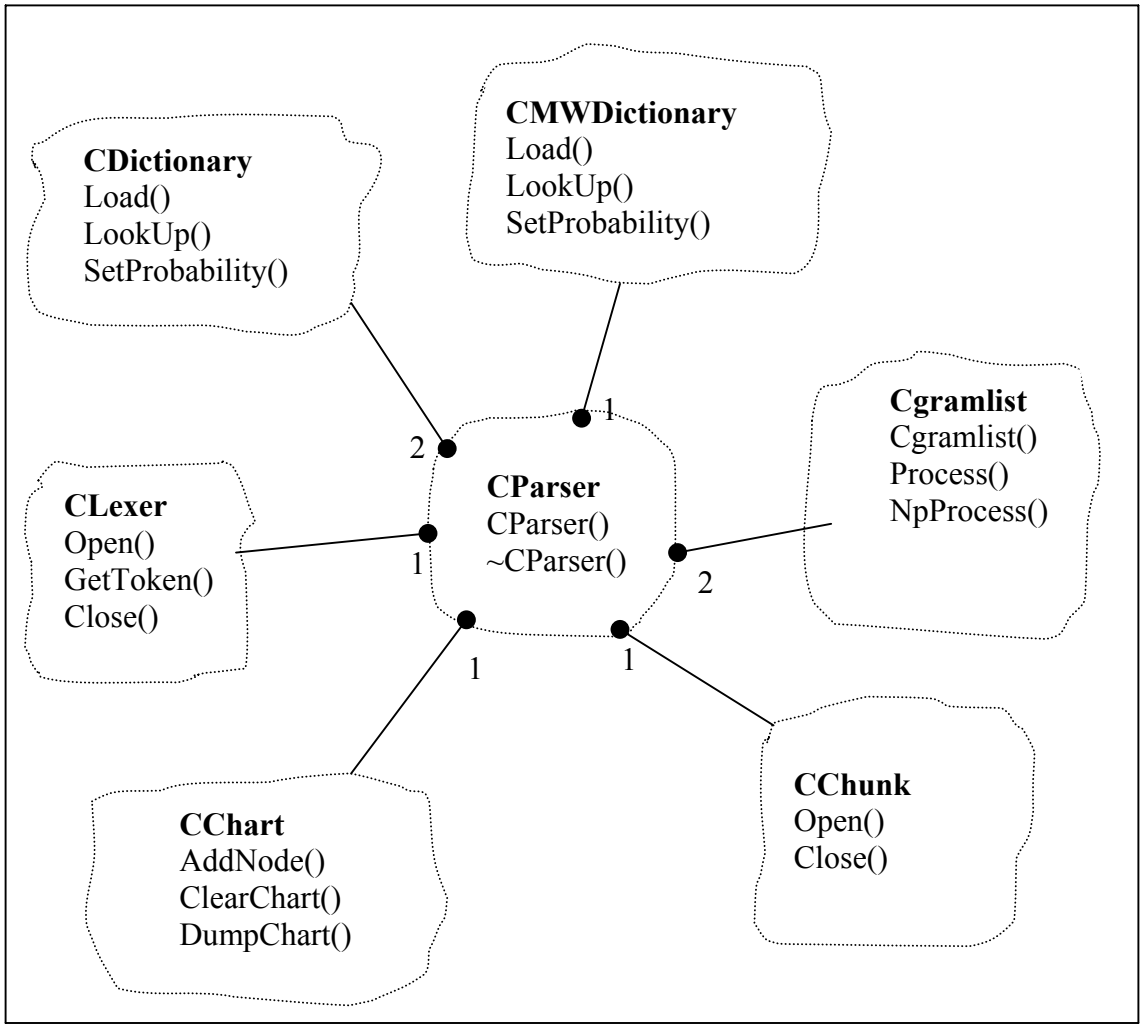

Figure 4.6 CParser Class Diagram

4.2.1 Stochastic Grammar Classes

Three classes Cgramlist, CgramNode and Cconstitute written in C++ programming language are used to load the stochastic context-free grammar, to update the probability, and to provide a dynamic link to the main class CParser. This section includes a brief discussion of these classes. In addition, Booch class-diagrams are provided to show the class interactions.

Cgramlist

The constructor of the CParser class instantiates a Cgramlist object, which uses a linked list to hold the grammar rules. The Cgramlist object creates CgramNode objects to store these rules. The Cgramlist object uses the Process() member function to compare each rule with every other rule to exclude the duplicate rules, to update probability, and to provide a dynamic link. The NpProcess() member function of the Cgramlist object deletes all other dynamic links for the grammar rules except the noun phrase rules. The class diagram of Cgramlist is shown in Figure 4.7.

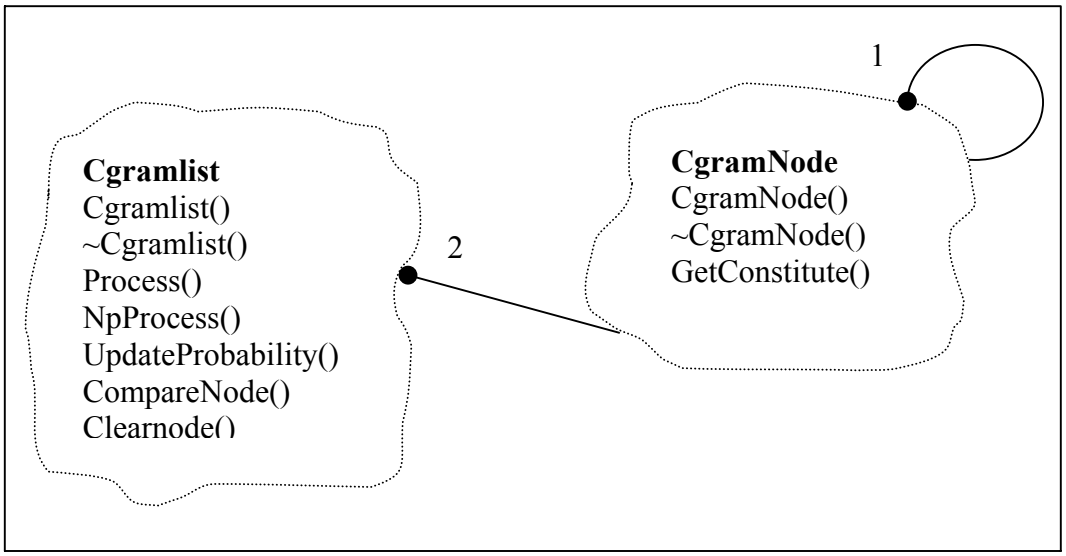

Figure 4.7 Cgramlist Class Diagram

CgramNode

This class is used to store each grammar rule in a linked list format. Each CgramNode class bears a dynamic link to the next CgramNode class as well as next grammar rule. An object of this class instantiates two objects of Cconstitute class. Each Cconstitute class holds the constituents or non-terminals forming the grammar rule.

The member function GetConstitute() of the CgramNode class is used to obtain the constituent non-terminal and its position in each grammar rule. The class diagram of this class is shown in Figure 4.8.

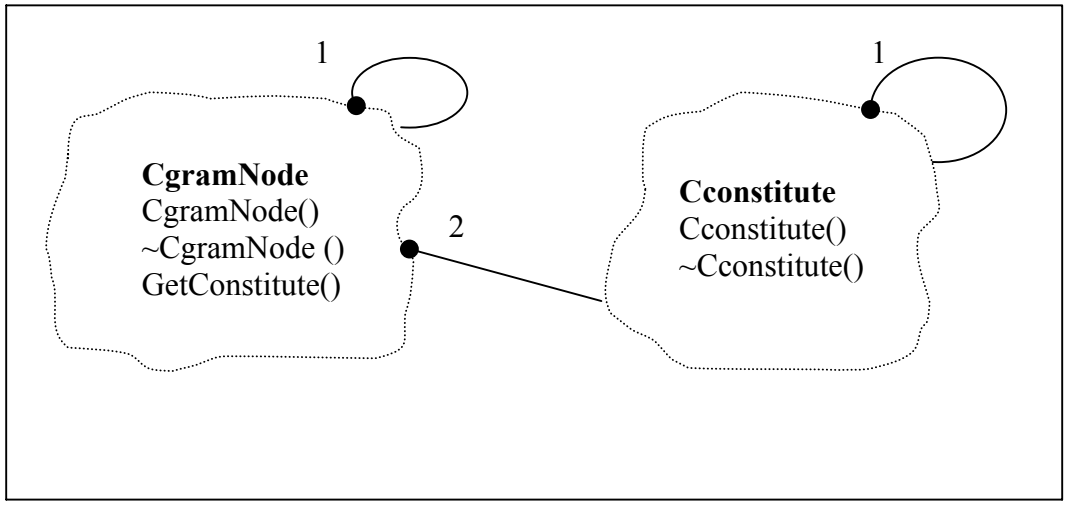

Figure 4.8 CgramNode Class Diagram

Cconstitute

This class is used to store the constituent nonterminals that form each grammar rule in a linked list format. Each CgramNode class instantiates Cconstitute class. Figure 4.9 shows the class diagram for this class.

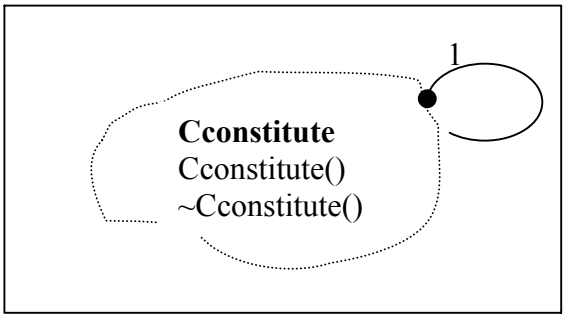

Figure 4.9 Cconstitute Class Diagram

4.3 ProbCompare Program

A C++ program "ProbCompare" was written to obtain the cumulative performance metrics for the probabilistic parser after each sentence. The metrics, as were described in section 3.5, show how precisely and accurately the probabilistic parser can extract simple noun phrases. To evaluate the performance metrics, about 229 English sentences were manually parsed to generate the simple noun-phrases. This input file, containing these manually parsed noun-phrases, was called "npmanual.txt". Table 4.2 lists the C++ header and source files that were used to generate the performance metrics. Table 4.3 shows the classes developed for this program. A brief description of each class is also included in this table.

Table 4.2 C++ Header and Source Files

The ProbCompare	
Header File	Npcall.h
Source Files	list.cpp

Table 4.3 Classes of the "ProbCompare" Program

Brief Descriptions of the Classes	
CNpcall	This CNpcall class is instantiated by the Clist class. This class represents each noun-phrase of the input files.
Clist	This class, instantiated by the CNounlist class, generates the dynamic list for the input files regarding simple noun-phrases.
CNounlist	This class does the following work: (1) holds the dynamic link to the extracted simple noun-phrases; (2) loads the text file "npmanual.txt" which contains the list of manually parsed noun-phrases; (3) defines the functions to calculate the performance metrics such as recall, precision, and f-factor; (4) defines three functions to determine the false positive, false negative, positive and negative noun-phrases; and (5) outputs the cumulative performance metrics into three text-files.

Chapter 5: Results

This chapter discusses the results of the performance metrics obtained by the stochastic context-free parser and non-stochastic context-free parser in extracting simple noun-phrases from English natural language. In addition, the performance of my SCFG has been compared with the observed or reported performance by Charniak [4], Magerman [14], Collins [8], Charniak [3], and Collins [7]. Four sets of input documents containing total 229 sentences were manually parsed to generate the "test-set" of simple noun-phrases. Afterwards, the extracted noun-phrases were compared with this test-set to calculate the performance metrics for the Stochastic Context-Free (SCF) parser and the Non-stochastic Context-Free (CF) parser as well as the deterministic parser. Ten US patents regarding the “DMA Controller” were considered as the input English text files to extract simple noun-phrases using the stochastic parser and the non-stochastic parser.

5.1 Recall, Precision & F-Factor

Three metrics (Recall, Precision and F factor) are considered as the standard evaluation metrics to compare various techniques used in NLP. Recall is defined as the proportion of the "correct" noun phrases extracted by the parsing system. Precision is defined as the proportion of the "correct" noun phrases to the total number of extracted noun phrases. This section includes my obtained performance metrics of the simple noun phrases extracted from the 10 US patent documents containing 229 sentences. To calculate Recall, Precision and F Factor, equations noted in section 3.7 were implemented in a C++ program “ProbCompare”. This program generated the cumulative performance metrics after each sentence of the input documents. Table 5.1 shows the distribution of four input sets and corresponding US patents used to generate the input sets. The table also includes the total number of noun-phrases found in each input document and the average number of noun phrases per sentence throughout the entire document.

Table 5.1 Sets of Input Text Files

Input Set	US Patents	Average No. of Noun Phrases Per Sentence	Total No. of Noun-Phrases
Input1.txt	US Patent 4137565 US Patent 4180855 US Patent 5175818	**6**	**171**
Input2.txt	US Patent 4180855 US Patent 4723223 US Patent 5067075 US Patent 5038218 US Patent 5590377	**5**	**370**
Input3.txt	US Patent 4404650 US Patent 4455620	**7**	**105**
Input4.txt	US Patent 4417304	**7**	**745**
Combined.txt	All of the Above US Patents.	**6**	**1391**

The following subsections present the results obtained in extracting simple noun-phrases form these input documents using CF parser and SCF parser.

5.1.1 Performance Metrics for Input1.txt

The ASCII text file "Input1.txt" contains 28 sentences of three US patents as listed in table 5.1. The performance metrics were calculated for the extracted simple noun phrases for these US patents using both CFG and SCFG cumulatively after each sentence. These results are listed below in table 5.2. As is seen from the table, 95.7% recall and 95.7% precision have been observed for the extracted simple noun-phrases by the SCF parser; 77.1% recall and 72.8% precision have been observed for the extracted noun-phrases by the non-stochastic parser. These tabulated results indicate that the statistical data can improve the performance of the parser to extract simple noun-phrases. Since recall and precision were calculated cumulatively for this input text file containing three

different US patents, to observe the variations of these performance metrics over each of these documents, the performance metrics were also independently evaluated for these documents. Clearly, the use of the stochastic context-free grammar in extracting simple noun-phrases shows promising results.

Table 5.2 Cumulative Performance Metrics for Input Set 1

Input Set 1	**Number Of Sentence**	**Stochastic Context-Free (SCF) Parser**			**Context-Free (CF) Parser**		
US Patents		**% Recall**	**% Precision**	**F Factor**	**% Recall**	**% Precision**	**F Factor**
4137565	5	100	95.6	42.05	80.6	66	18.1
5175818	13	96.6	91.9	53.75	76.3	64.6	20.3
4180855	10	98.4	93.8	57.6	85.1	87	34.4
Input1.txt	**28**	**95.7**	**93.4**	**147.41**	**77.1**	**72.8**	**68.1**

The cumulative variations of these three metrics have been plotted using a simple Matlab program and are shown in the following figures for both statistical context-free (SCF) and non-statistical context-free (CF) parsers. In these figures, the solid blue-line represents the characteristics of the cumulative recall or precision for the extracted simple noun phrases, whereas the line represented by the "o" sign is the average cumulative performance metric. In the Figures 5.1 and 5.2, cumulative variation in percentage recall has been plotted respectively for probabilistic parser and non-stochastic parser. As is seen from these figures, the stochastic parser can recall a higher

percentage of simple noun phrases than the traditional parser. An improvement of 12% in recalling simple noun phrases has been realized by the SCFG over the CFG.

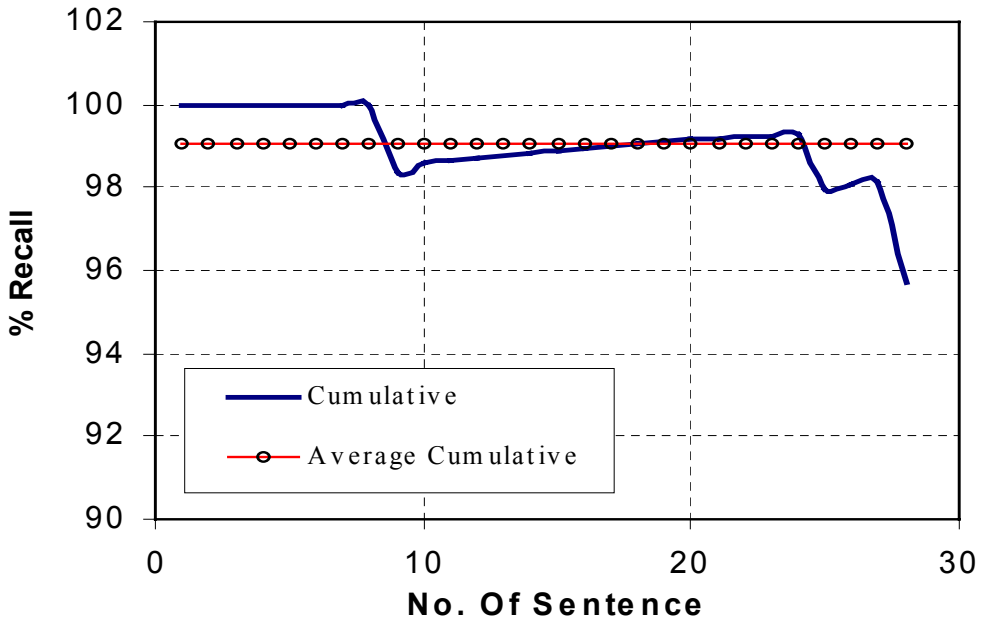

Figure 5.1 Percentage Recall vs. Sentence Number for SCF Parser

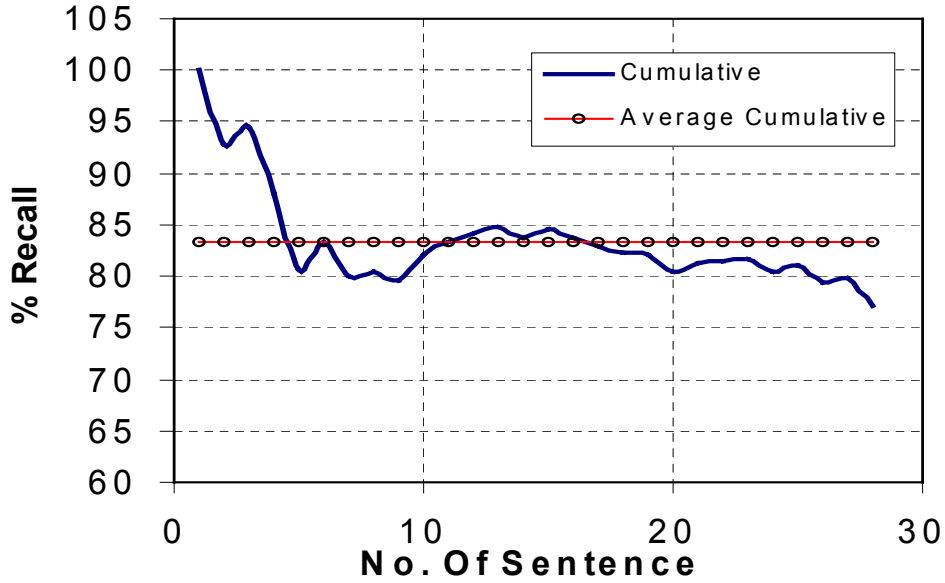

Figure 5.2 Percentage Recall vs. Sentence Number for CF Parser

In addition to the higher recalling rate, stochastic parser shows better precision rate also. Figures 5.3 and 5.4 verify this statement.

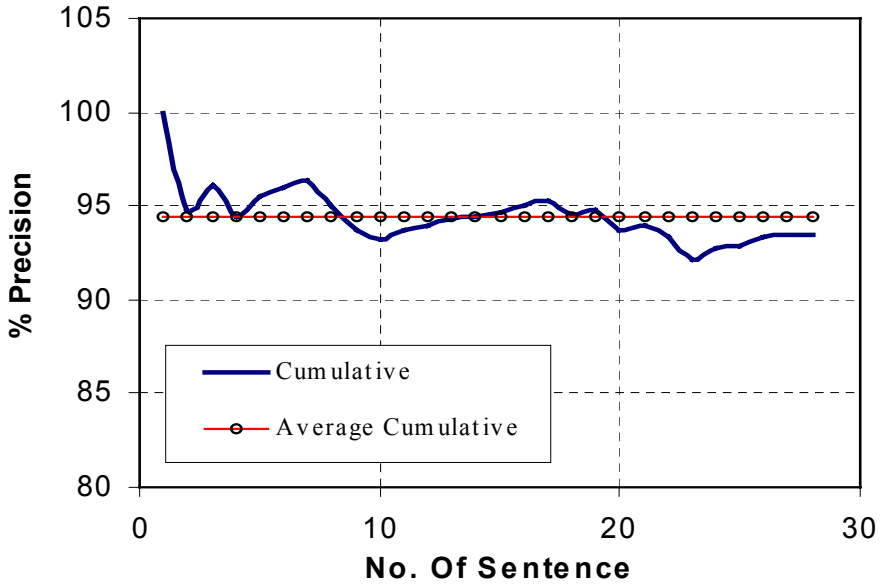

Figure 5.3 Percentage Precision vs. Sentence Number for SCF Parser

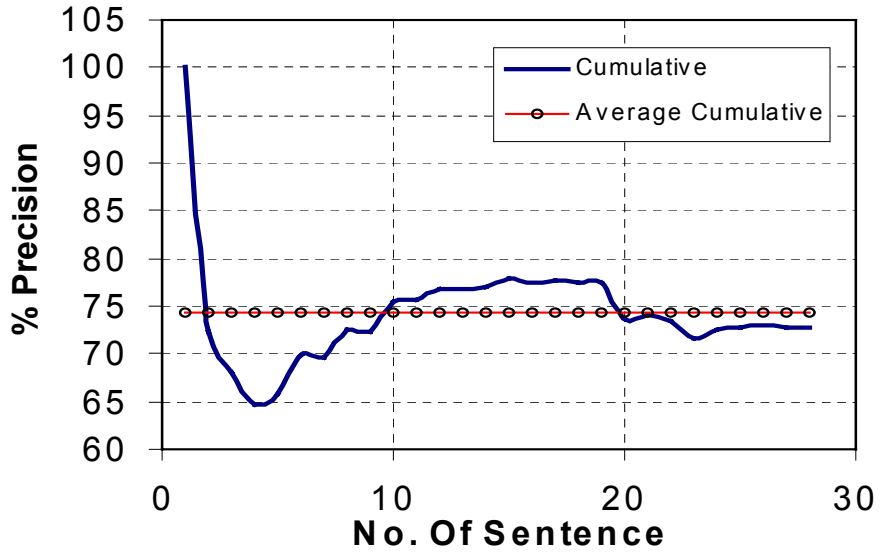

Figure 5.4 Percentage Precision vs. Sentence Number for CF Parser

These figures show the cumulative variation of the precision for the extracted simple noun-phrases using the stochastic context-free grammar and the non-statistical context-free grammar respectively. The precision rate was improved by an average of 20%, when the stochastic parser was used instead of traditional chart parser for the very same input text file "Input1.txt". Figures 5.5 and 5.6 show the variation of cumulative **F factor** (**=0.5∗Recall∗Precision/(Recall+Precision)**) over sentences for the stochastic and non-stochastic parser respectively.

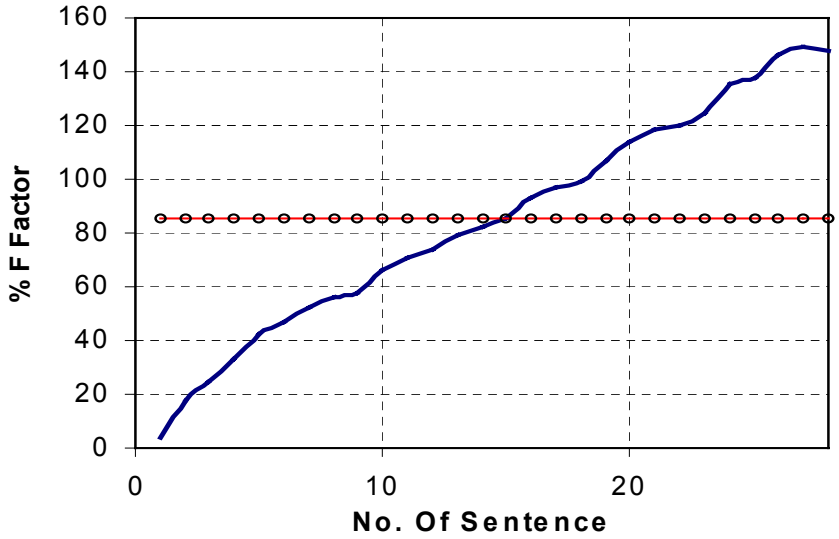

Figure 5.5 Percentage F-factor vs. Sentence No for SCF Parser

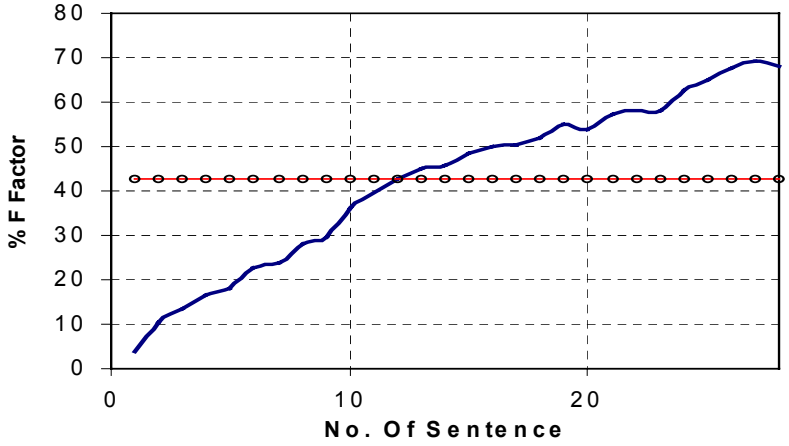

Figure 5.6 Percentage F-factor vs. Sentence Number for CF Parser

Both these figures show increasing F factor with an increase in sentence number. However, the slope of the variation-curve as well as the rate of change of F factor is higher for SCF parser than that of the CF parser, indicating that the stochastic parser extracts simple noun phrases more efficiently and precisely than CF parser for these 28 sentences.

5.1.2 Performance Metrics for Input2.txt

The ASCII text file "Input2.txt" contains 73 sentences of five US patents as listed in Table 5.1. Five different paragraphs were taken from the middle section of each US patent documents to ensure varieties in input sentences. The contents of each paragraph are distinct in the sense that each describes a different topic independent of each other. It was found that the stochastic parser provided not only the better and reliable extraction of simple noun phrases from the English documents, but also provided approximately the same rate of precision and recall throughout the whole input documents. For the "Input2.txt" file, the performance metric was also calculated for the extracted simple noun phrases using both CFG and SCFG cumulatively after each sentence. The result obtained for each US patent as well as for the combined one (Input2.txt) is shown in the Table 5.3.

Table 5.3 Cumulative Performance Metrics for Input Set 2

Input Set 2	Number Of Sentence	Stochastic Context-Free (SCF) Parser			Context-Free (CF) Parser		
US Patents		**% Recall**	**% Precision**	**F Factor**	**% Recall**	**% Precision**	**F Factor**
4180855	12	100	98.1	52.5	100	94.4	50.1
4723223	11	100	100	46	92.9	95.12	36.65
5067075	11	96.7	93.7	56.2	76.5	81.8	28.5
5038218	26	100	99	118.5	92.5	85.1	76.2
5590377	13	95.3	88.4	60	87.1	85.72	46.7
Input2.txt	**73**	**98.6**	**96.0**	**331.75**	**90.6**	**87.6**	**239.6**

The percentage recall and the percentage precision of the extracted simple noun phrases by the stochastic parser has been found to be 98% and 96% respectively; whereas 90.6% recall and 87.6% precision were noted for the CF parser. These tabulated results indicate that the inclusion of the statistical knowledge about the grammatical structures of English language improve the performances of the parser to recall simple noun-phrases by 8% with an increasing precision.

Figures 5.7 and 5.8 show the variations of percentage recall over number of sentences for both stochastic parser and non-stochastic parser.

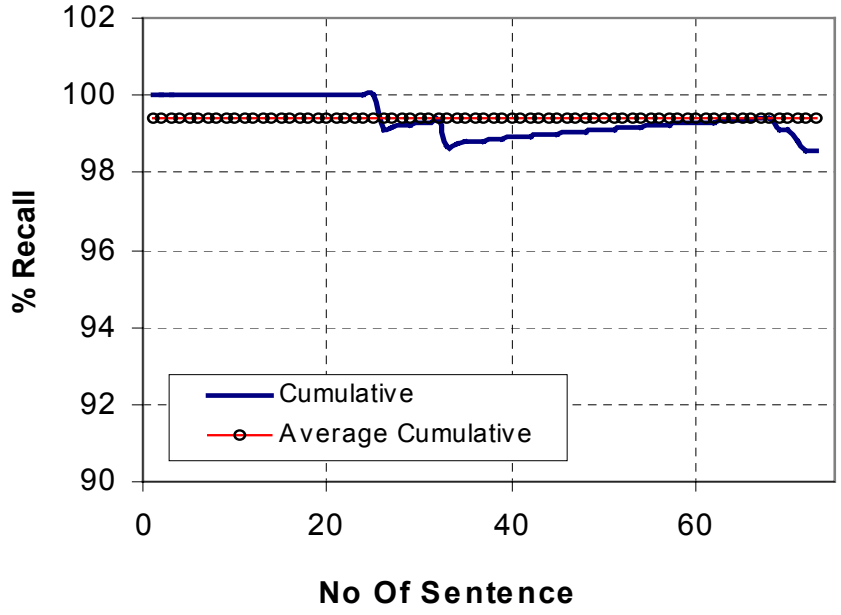

Figure 5.7 Percentage Recall vs. Sentence Number for SCF Parser

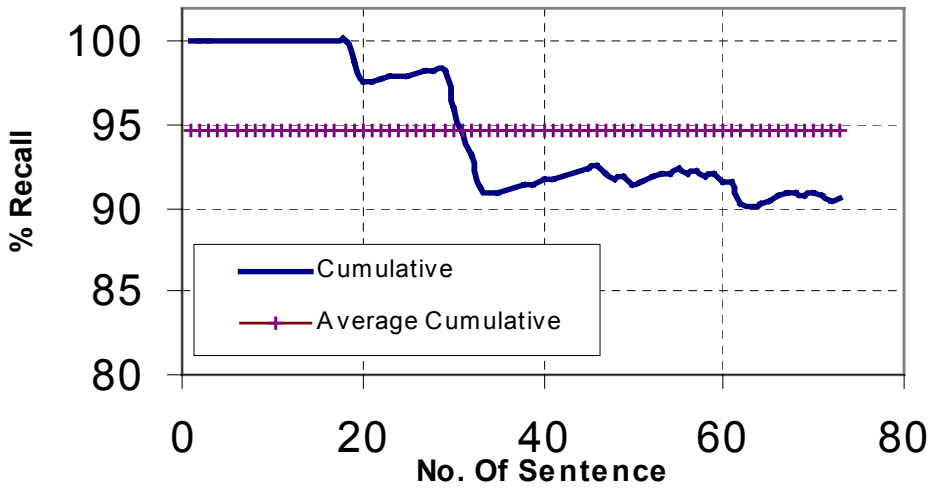

Figure 5.8 Percentage Recall vs. Sentence Number for CF Parser

As is seen from these figures, the probabilistic parser provides not only the higher rate of recall in extracting simple noun phrases, but it also offers a steady rate of recall throughout the input text file. The non-probabilistic parser extracts noun-phrases with a wide variation in percentage recall. It is mentioned before that the input file "Input2.txt" contains sentences from five different US patents. Non-statistical parser extracts simple noun phrases with a sudden variation in percentage recall from these sentences. Therefore, the performance metric for the CF parser is more dependent on the contents of the input-text files than the SCF parser.

To determine which parser extracts simple noun-phrases with greater precision, the variations in precision over these wide varieties of sentences have been plotted. Figures 5.9 and 5.10 show the cumulative precision rate that I obtained using the probabilistic parser and non-probabilistic parser over these 73 sentences.

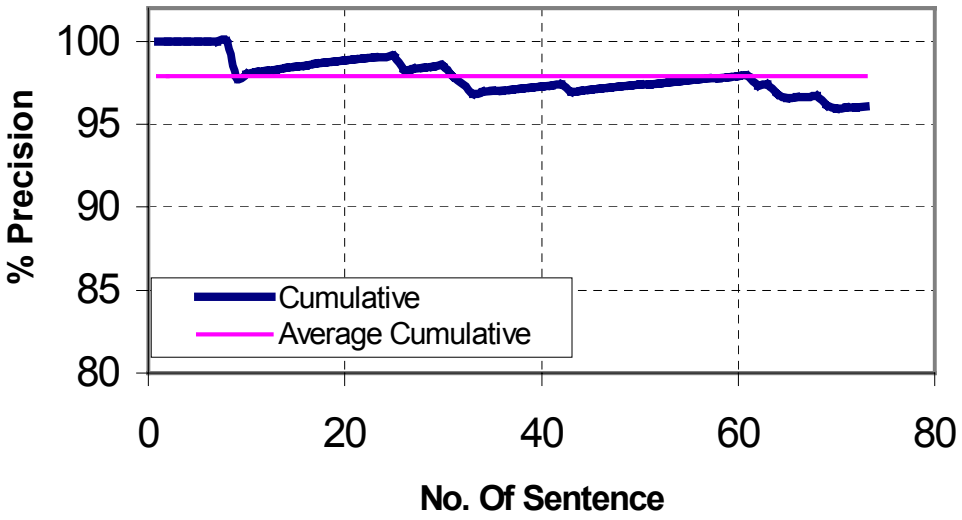

Figure 5.9 Percentage Precision vs. Sentence Number for SCF Parser

From these figures, it is seen that the stochastic parser provides a higher precision rate and a steady precision rate than the non-stochastic parser. If we look at the Figure 5.10 which represents the precision rate obtained for the non-stochastic context-

free parser, we can assume that even though the overall trend of the precision rate is decreasing throughout the input text file, precision rate becomes more steady within the sentence numbers of 32 to 73 indicating that the precision in extracting simple noun phrases using the CF parser is more likely to depend on the input-contents.

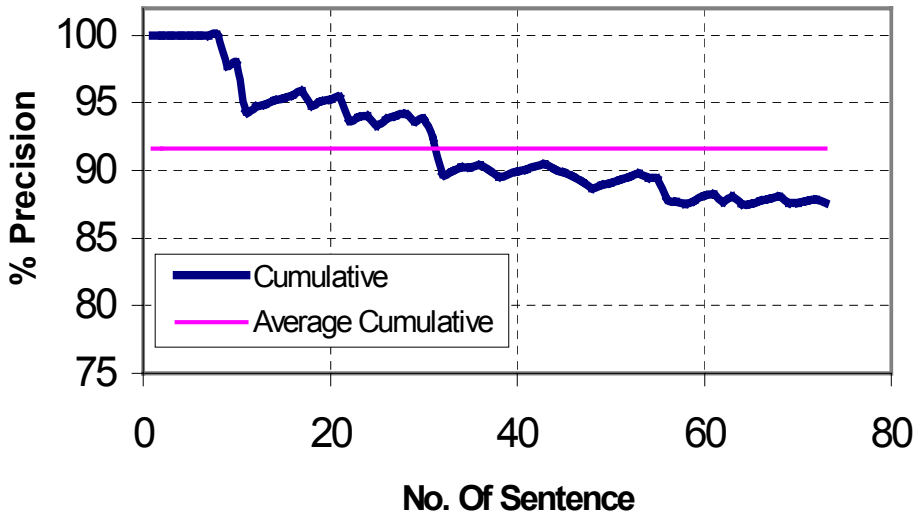

Figure 5.10 Percentage Precision vs. Sentence Number for CF Parser

Figures 5.11 and 5.12 show the F-factor variations obtained by the respective parsers in extracting simple noun-phrases. Both plots show an increasing F-factor with an increase in sentence number. As is expected, Stochastic parser gives a higher rate of change of F-factor.

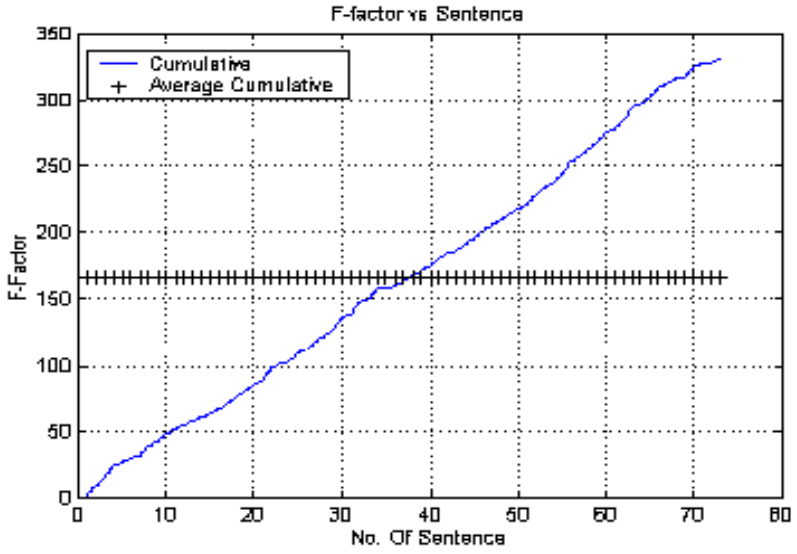

Figure 5.11 F-factor vs. Sentence Number for SCF Parser

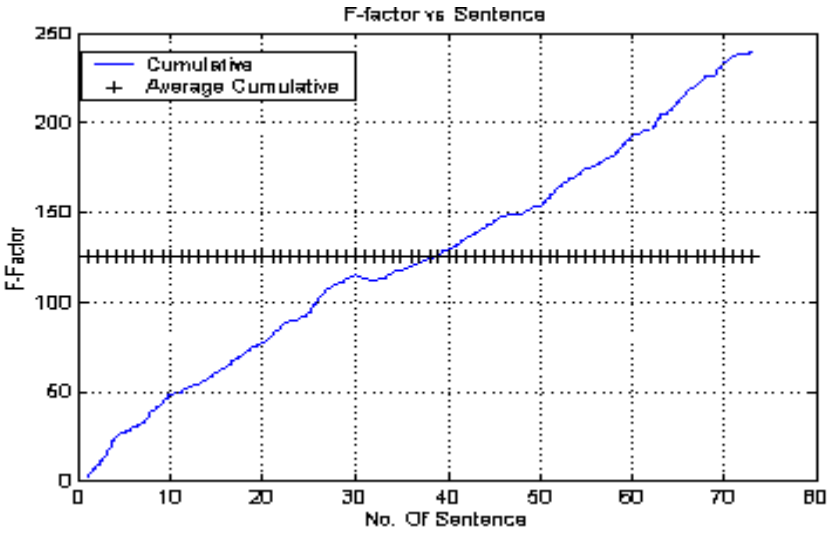

Figure 5.12 F-factor vs. Sentence Number for CF Parser

5.1.3 Performance Metrics for Input3.txt

The ASCII text file "Input3.txt" contains 14 sentences of two different US patents as listed in Table 5.1. For this input document, the context-free parser shows a much lower rate in precision (73.8%) and recall (64%) in extracting simple noun-phrases. The stochastic parser shows 100% recall and 92% precision rate for this same document. It was found that the stochastic parser extracted the "correct" parse for a sequence of words, having multiple meanings, more efficiently than the non-stochastic parser. For example, a "correct" noun-phrase "receiving states" of this input document was successfully extracted by the stochastic parser, but the non-stochastic parser only extracted the word "states" as a simple noun-phrase. Another "correct" noun-phrase "The first address bus switch circuit" of this input document was also successfully extracted by the stochastic parser, but the non-stochastic parser extracted two noun-phrases ("The first address bus" and "circuit") from this one simple phrase. The "correct" noun-phrase "the same" was not extracted at all by the non-stochastic parser. Table 5.4 shows the observed performance metrics for these parsers.

Table 5.4 Cumulative Performance Metrics for Input Set 3

Input Set 3	Number Of Sentences	Stochastic Context-Free (SCF) Parser			Context-Free (CF) Parser		
US Patents		% Recall	% Precision	F Factor	% Recall	% Precision	F Factor
Input3.txt	14	100	92.6	84.6	69.6	73.8	34.4

Figures 5.13 & 5.14 show the variations in percentage recall in extracting simple noun phrases for the stochastic parser and non-stochastic parser respectively. The variations in percentage precision for these parsers are shown in Figures 5.15 and 5.16. These figures are self-explanatory. However, it is worthy to note that the stochastic parser

provides much reliable extraction of information than the non-stochastic parser. Depending on the input texts, where the performance metric goes down for the CF parser, SCF parser preserves its performance with high degree of accuracy.

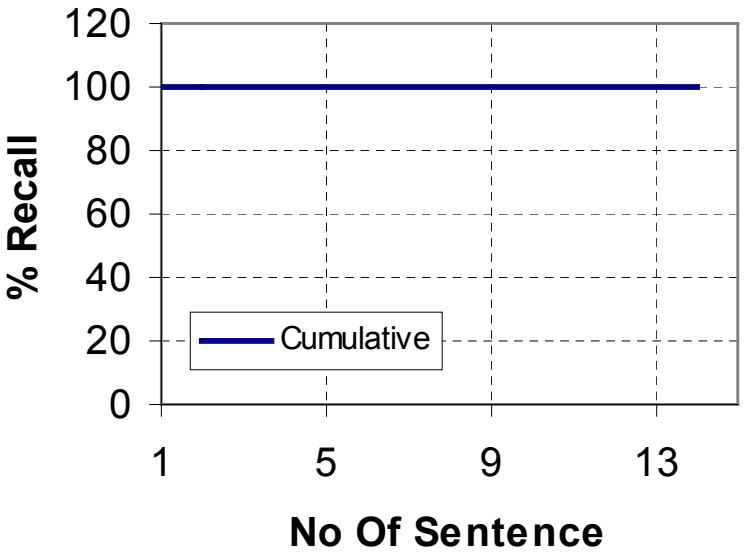

Figure 5.13 Percentage Recall vs. Sentence Number for SCF Parser

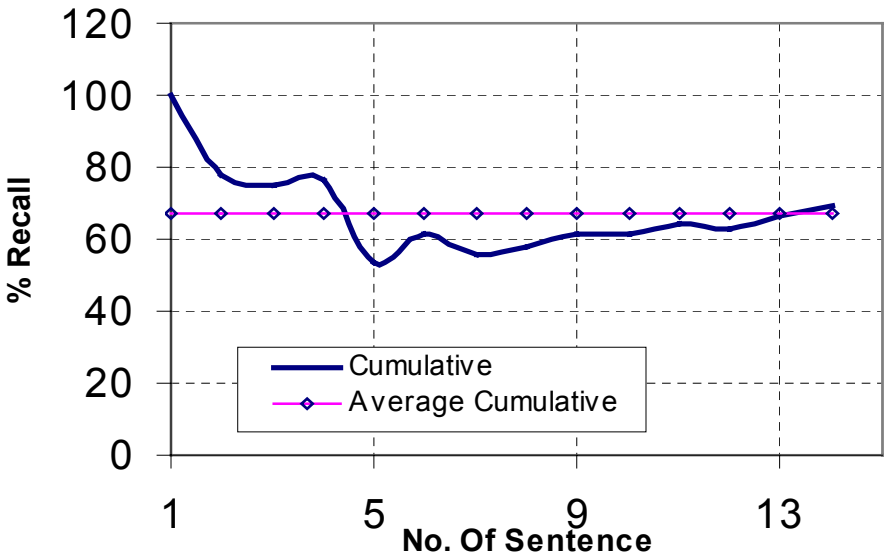

Figure 5.14 Percentage Recall vs. Sentence Number for CF Parser

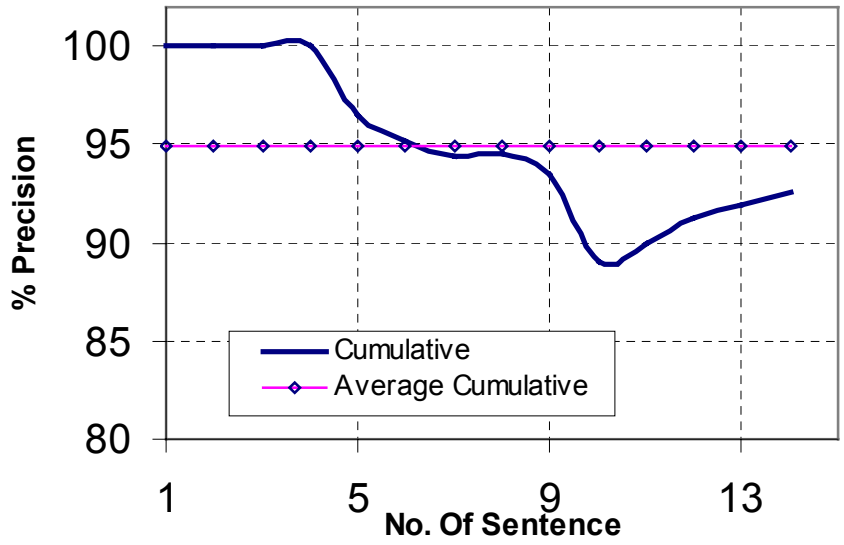

Figure 5.15 Percentage Precision vs. Sentence Number for SCF Parser

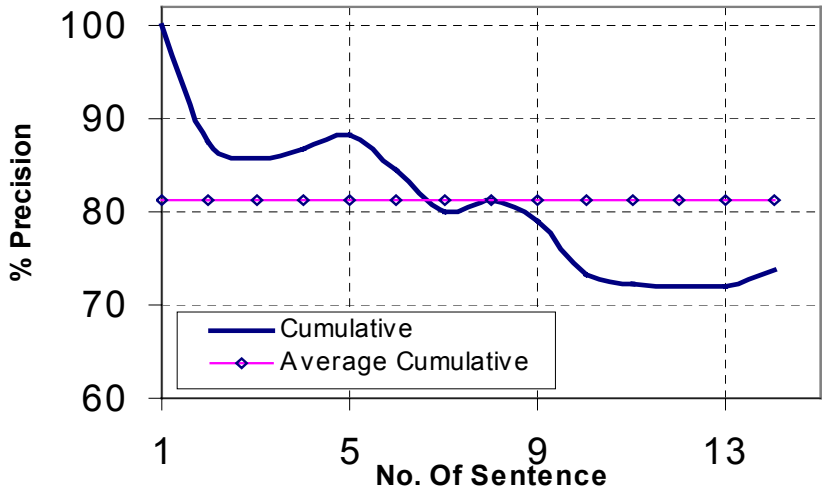

Figure 5.16 Percentage Precision vs. Sentence Number for CF Parser

5.1.4 Performance Metrics for Input4.txt

The ASCII text file "Input4.txt" contains 114 sentences of the US patent 4417304. Table 5.5 shows the cumulative performance metrics for both the CF and SC parsers. This tabulated data reconfirms that the stochastic parser provides reliable extraction of simple noun phrases from the English documents with greater precision than the CF parser.

Table 5.5 Cumulative Performance Metrics for Input Set 4

Input Set 4	Number Of Sentences	Stochastic Context-Free (SCF) Parser			Context-Free (CF) Parser		
US Patent 4417304		% Recall	% Precision	F Factor	% Recall	% Precision	F Factor
Input4.txt	114	99.0	92.5	646.8	87.4	81.0	384.3

The tabulated data was collected from a single document of 114 sentences, so that the results reflect the performance metrics for these parsers over a uniform input document with a large number of sentences. As is noted in the table, the cumulative recall is 99% for the SCF parser and that for CF parser is 87.4%. The non-stochastic parser shows an improved performance for a uniform input document; whereas, the stochastic parser provides almost a steady rate in performance metrics for both the uniform input and non-uniform input (containing an wide varieties of sentences).

Figures 5.17 & 5.18 show the variation in recall and Figures 5.19 & 5.20 show the variations in precision in extracting simple noun-phrases by the SCF parser and the CF parser respectively.

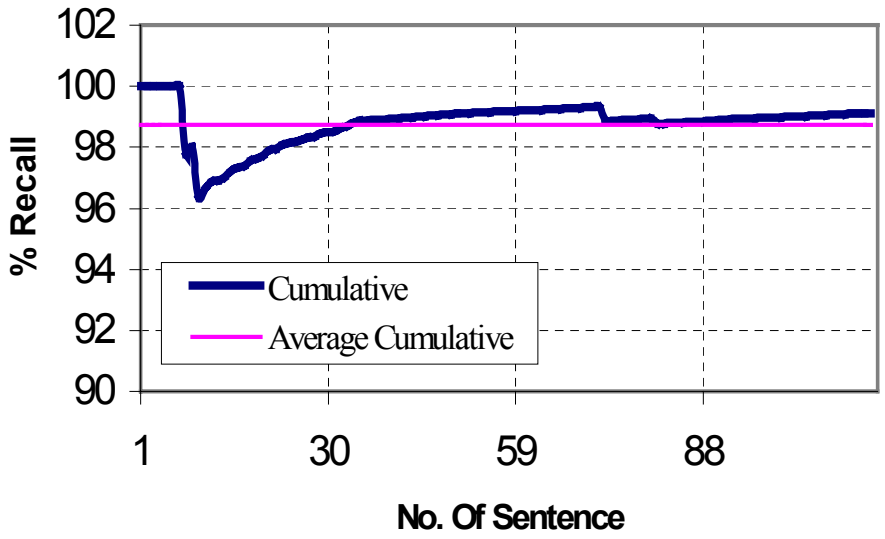

Figure 5.17 Percentage Recall vs. Sentence Number for SCF Parser

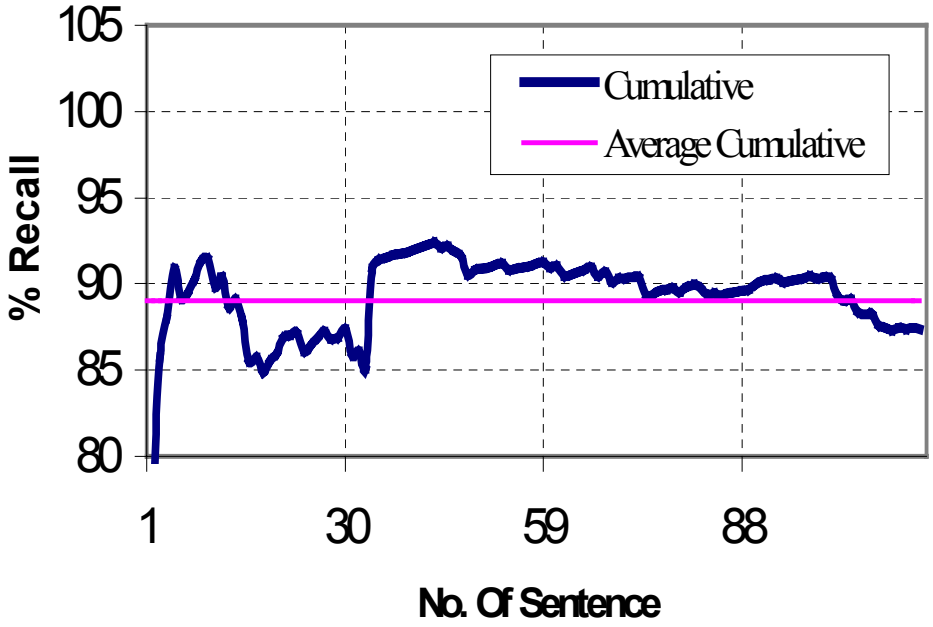

Figure 5.18 Percentage Recall vs. Sentence Number for CF Parser

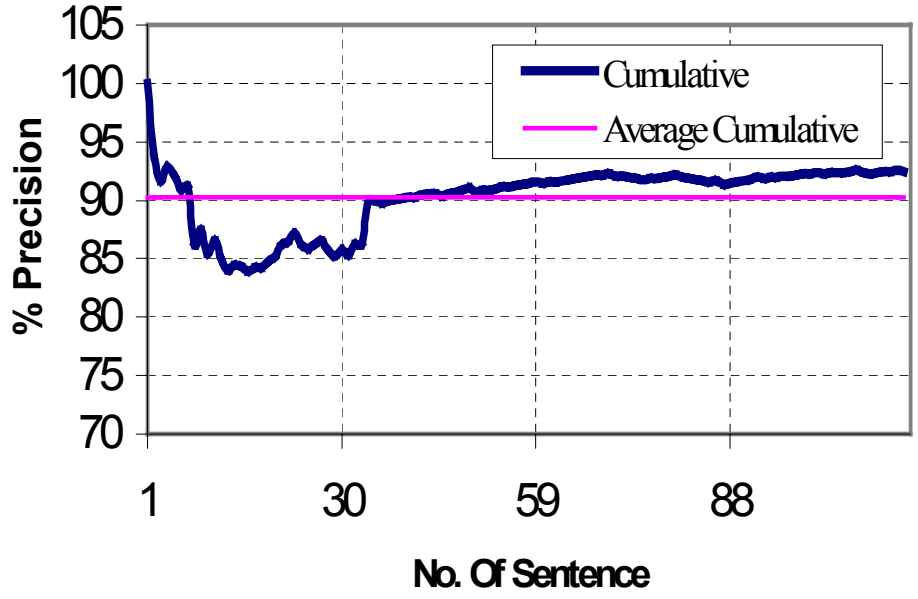

Figure 5.19 Percentage Precision vs. Sentence Number for SCF Parser

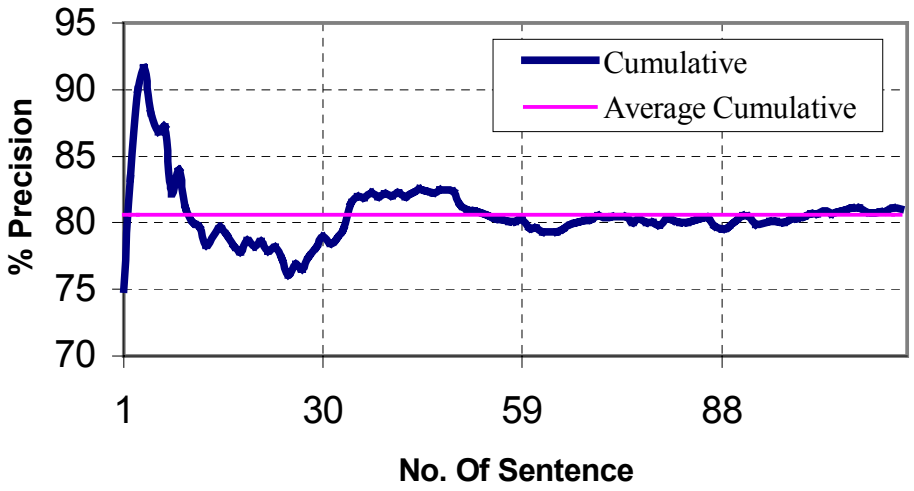

Figure 5.20 Percentage Precision vs. Sentence Number for CF Parser

5.1.5 Performance Metrics for Combined.txt

The ASCII text file "Combined.txt" was obtained by combining all four previously mentioned input text-files. This combined input document was used to generate the performance metrics for SCF and CF parsers as shown in Table 5.6. This combined document, containing 229 wide varieties of sentences, provides a better test of the performance that can be obtained by the SCF and CF parsers in extracting simple noun-phrases. As was mentioned earlier, all the performance metrics were calculated for the extracted simple noun phrases cumulatively after each sentence of the input document.

Table 5.6 Cumulative Performance Metrics for Combined.txt

Input Set	Number Of Sentence	Stochastic Context-Free (SCF) Parser			Context-Free (CF) Parser		
10 US Patents.		% Recall	% Precision	F Factor	% Recall	% Precision	F Factor
Combined .txt	229	98.9	93.5	1216.4	86.5	81.5	727.6

These tabulated results show the percentage recall to be 89.9% for the SCF parser and 86.5% for the CF parser. Besides the higher recall, the stochastic context-free parser provides us with the higher rate in precision than the non-stochastic parser. Figures 5.21 & 5.22 show the variations in percentage recall over these sentences in extracting simple noun-phrases by the stochastic parser and the non-stochastic parser respectively. On an average 12% improvement in recalling the simple noun phrases has been observed for SCF parser than the CF parser.

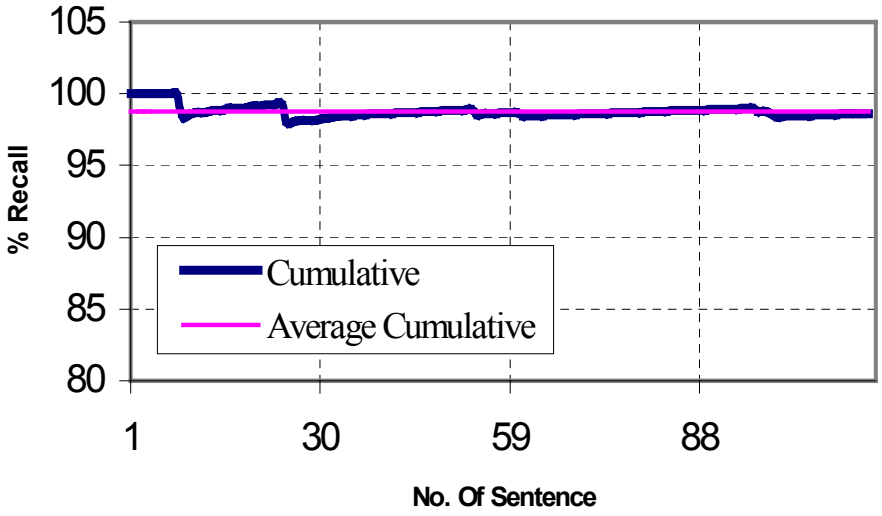

Figure 5.21 Percentage Recall vs. Sentence Number for SCF Parser

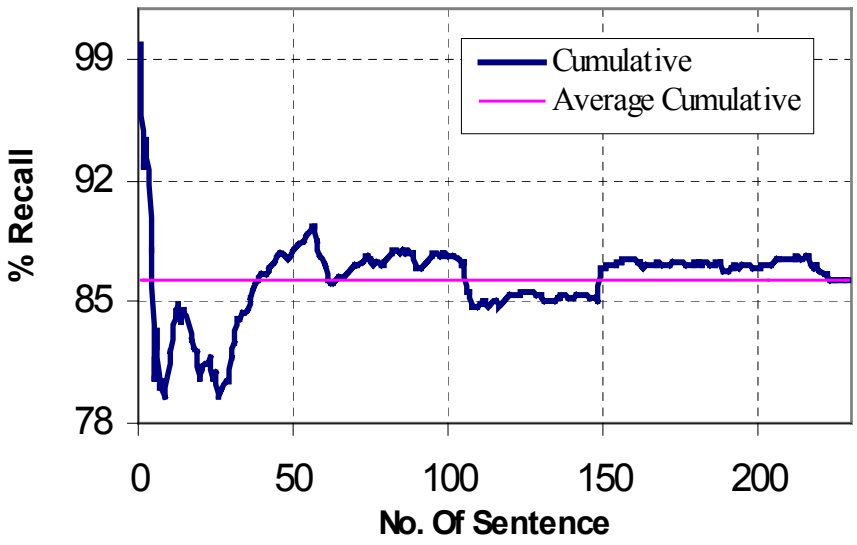

Figure 5.22 Percentage Recall vs. Sentence Number for CF Parser

Figures 5.23 & 5.24 show the variation in the cumulative precision over these 229 sentences for SCF parser and CF parser respectively.

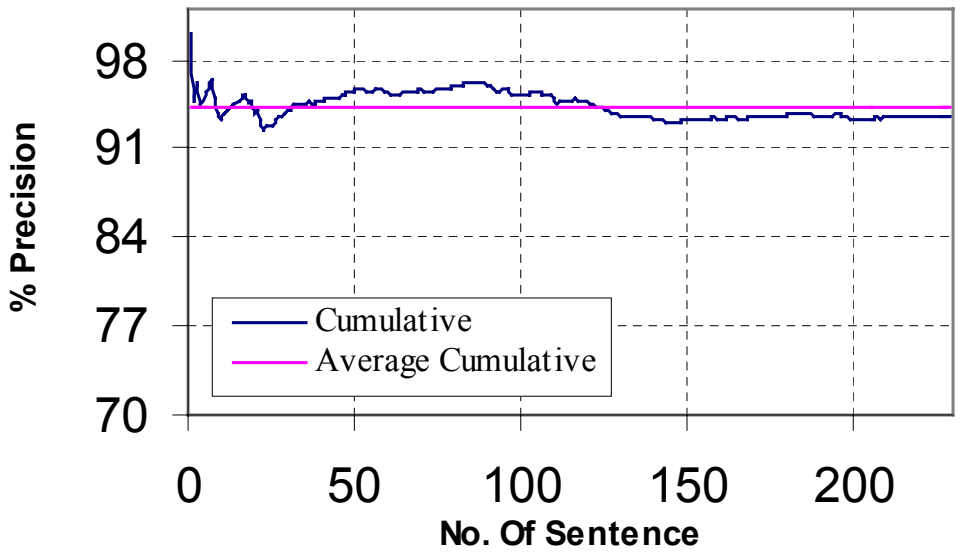

Figure 5.23 Percentage Precision vs. Sentence Number for SCF Parser

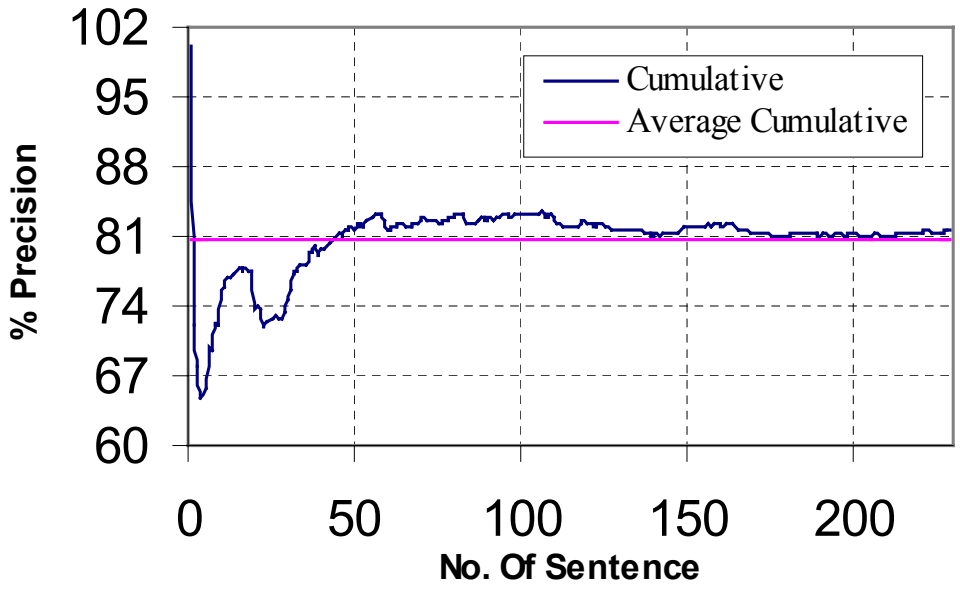

Figure 5.24 Percentage Precision vs. Sentence Number for CF Parser

From the above figures and tables, several conclusions can be drawn on the performance metrics observed in extracting simple noun phrases by the stochastic parser and non-stochastic parser. These are listed below:

(1) an improvement of about 12% has been observed in percentage recall for the stochastic parser in extracting simple noun-phrases;
(2) an improvement of about 11% has been observed in percentage precision for the SCF parser as compared to that of the non-stochastic parser in extracting simple noun-phrases;
(3) the stochastic parser provides a steady rate in precision and recall for extracting simple noun-phrases irrespective of the contents of the input document, as opposed to the non-stochastic parser whose performance varies largely depending on the input documents.

5.2 Comparing Results with Others' Work

This section compares the results obtained in this research using a probabilistic parser with that obtained or reported by various researchers having the same goal of information extraction from English language. So far, Kenneth W Church [1988] has presented the most successful work in parsing and extracting simple noun-phrases from English language. Table 5.7 shows the results reported by Church. A comparison between Church's work and this research is also included in this table.

Table 5.7 A Comparison between Church Work and This Research Work

System	Church [1988]	This Research [2001
Parsing Algorithm	Dynamic Stochastic Parsing Algorithm	Most Probable Longest Phrase Parsing Algorithm
Probability Considered	Lexical Probabilities & Contextual Probabilities	Contextual Grammatical Probabilities
Parsing Approach	Bottom-Up	Bottom-Up
Probability Estimates Obtained From	Tagged Brown Corpus	Tagged Penn Tree-Bank
Reported Performance Metrics	**95%-99% "correct" Tagging. Among 243 noun-phrases of sample sentences only 5 omitted.**	**Recall rate 98.9% Precision Rate 96%**

Table 5.8 shows the performance metrics reported by Charniak [4], Magerman [14], Collins [8], Charniak [3], and Collins [7] in extracting information from English documents. The researchers obtained the results included in this table not just for the simple noun-phrases but for extracting the best parse of the entire sentence. Since, it is more difficult to parse entire sentences than noun-phrases, this table does not present a fair comparison. The comparison is included here to provide some perspectives.

Table 5.8 Performance Metrics of Some Statistical Parsing Systems

Reported System	% Recall	% Precision
Charniak , 1996 [4]	80.4	78.8
Magerman , 1995 [14]	84.6	84.9
Collins, 1996 [8]	85.8	86.3
Charniak, 1997 [3]	87.5	87.4
Collins, 1997[7]	88.1	88.6
This Research (noun-phrase only)	98.9	96

Figures 5.25 & 5.26 show the percentage recall rate and the percentage precision rate obtained for various stochastic parsers including my research.

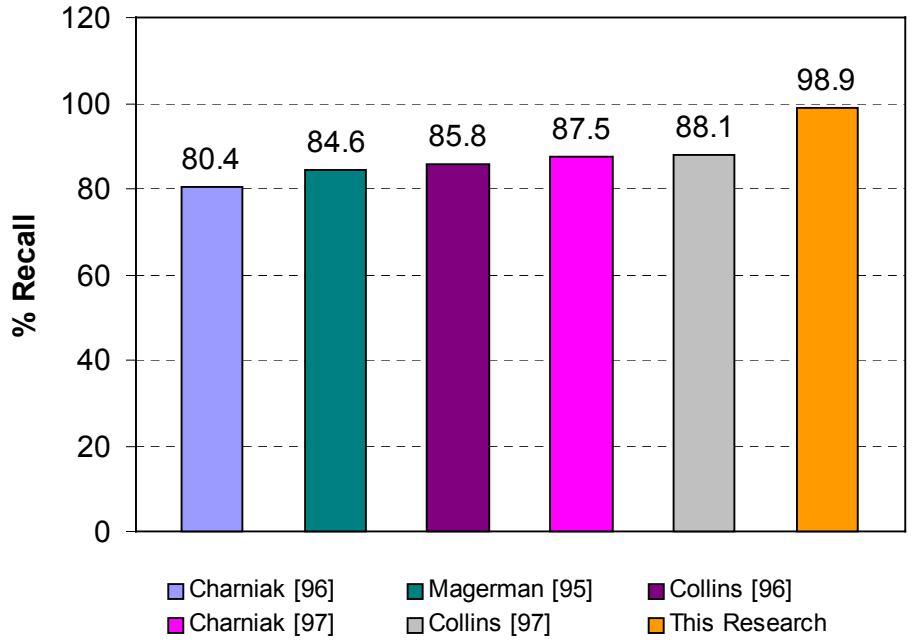

Figure 5.25 Reported Percentage Recall

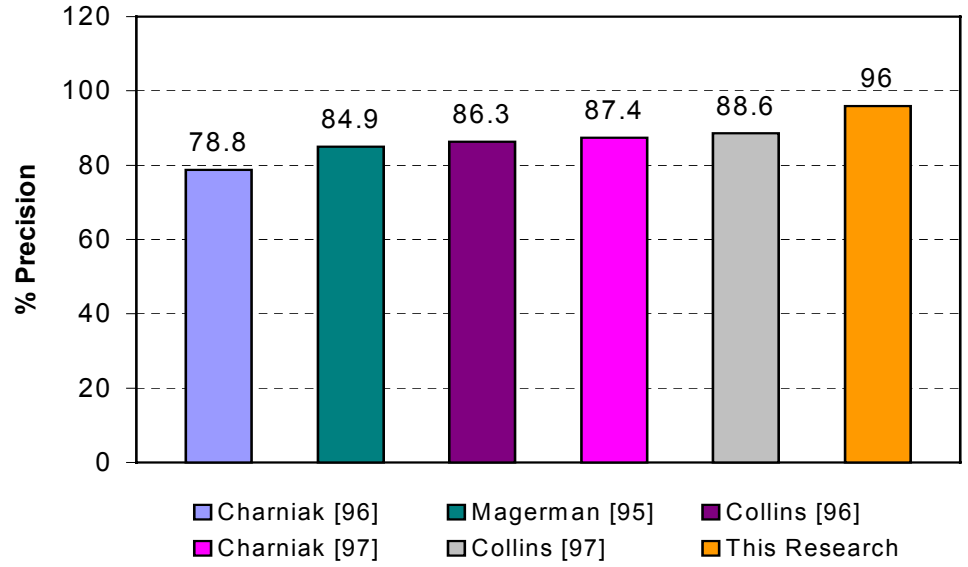

Figure 5.26 Reported Percentage Precision

Chapter 6: Conclusion

This chapter focuses on the capabilities and the limitations of the probabilistic parser presented in this thesis. In addition, some directions for future work have been suggested. The results of this research, presented in Chapter 5, indicate that the statistical knowledge regarding the grammatical rules of English language can improve extraction of simple noun-phrases to a large extent. These results leave us with an optimistic approach towards the understanding of natural language

6.1 System Capabilities

The probabilistic parser ProbChunker can successfully analyze and extract simple noun phrases from English with the probabilistic knowledge of grammatical structures of English sentences. This parser was used to parse approximately 4000 sentences taken from US patent-technical specifications. The dictionary contains 5200 words and the probabilistic grammar contains 1624 noun-phrase rules to identify simple noun-phrases in the English document. The cumulative percentage recall and precision are found to be 98.9% and 96% for simple noun phrases.

Since English sentences can be of wide varieties, to generate reasonably unbiased results for the probabilistic parser, a total of 229 sentences taken from 10 different US patent documents was analyzed to produce the graphs of performance metrics. These graphs provide a vivid description of the variations in precision and recall rates of extracting simple noun phrases throughout an English document. This program has also successfully extracted a maximum of 104 noun phrases from a single "sentence". This capability indicates that this algorithm, unlike some other contemporary stochastic parsers [3 & 7], is not limited to the number of words in a sentence.

All the functions and classes of this program used to implement the probabilistic parsing algorithm are written in an object-oriented fashion. This ensures that this program can always be integrated with the statistical data as well as probabilities of the lexical contents of the English words (if available in future) to extract information more efficiently. Though this project uses a total of 1624 grammar-rules regarding only the syntactic structures of simple noun phrases in English document, this program uploads and updates the probabilities for all the 11060 grammar rules. Therefore, the program can be used to extract other phrases of English natural language without much modification.

6.2 System Limitations

This system uses only the probabilities of grammar rules to detect simple noun phrases. As was pointed out in Chapter 3, to implement the probabilistic model in extracting information, we also need to include the conditional probabilities of words and the transitional probabilities of words. Due to the unavailability of such statistical collection of data, this project does not use the exact word probabilities. Instead, this project assigned word probabilities depending on the number of word-meanings in the dictionary. Even though the 98.9% cumulative percentage recall that was achieved in this project is very promising, this result was obtained extracting only one of the seven parts-of-speech of English language. Nevertheless, the noun-phrase is the most important part-of-speech in English documents.

The parsing algorithm assumes that a sentence is always ended by the full-stop (.) punctuation mark and therefore outputs a parsing chart only when a full-stop is encountered. However, in English documents, a sentence is often ended by successive white spaces (for example entries in a table) and not by a full stop mark. These sorts of situations in English sentences often cause the probabilistic parser (with a larger number of grammar rules) to generate a large number of parses with a dead end.

Finally, like other statistical parsers, the speed of the probabilistic parsing to extract information is an issue for this project. Considering the probabilities of all parses after each token, this program reduces the number of parses to at most two to three, which in turn reduces the execution time of the parser. Still, the problem regarding the speed of the computer is to be addressed for efficient use of this probabilistic parser.

6.3 Directions for Future Work

In future, the following extensions should be incorporated with the probabilistic parser to increase the precision and recall in retrieving information from English documents:

(1) finding the conditional and transitional word-probabilities, the probabilistic parser can be modified to include the statistical knowledge of words;
(2) finding the frequencies of occurrence of the grammar rules in this probabilistic parser, the size of the grammar can be reduced to increase the speed of the parser by deleting the rules that were never or infrequently used; and
(3) considering all the grammar rules, phrases other than the noun-phrase can also be extracted to observe the accuracy and precision of this parsing algorithm.

APPENDIX A: List of Acronyms

AI	Artificial Intelligence
CF	Context-Free
CFG	Context-Free Grammar
NL	Natural Language
NLP	Natural Language Processing
PCFG	Probabilistic Context-Free Grammar
POS	Parts-Of-Speech
SCFG	Stochastic Context-Free Grammar
adj	Adjective
adjs	Adjective phrase
adv	Adverb
advp	Adverbial Phrase
pps	Prepositional Phrase
noun	Noun Singular
nounp	Noun plural
np	Noun Phrase
verb	Verb Singular form
verbp	Nonterminal verb plural form
vp	Verb Phrase

Appendix B: List of Variables

G	Probabilistic Context-Free Grammar
L	Language (generated and accepted by a grammar).
P	A set grammar rules
S	Sentence.
T	Terminal Symbols
V	Non-Terminal Symbols
fp	False Positive
fn	False Negative
n	True Negative
p	True Positive
t	Parse Tree.
$\{w^1, \ldots\ldots w^n\}$	Terminal Vocabulary

Appendix C: Notations of the Penn Tree-Bank Tagset

C.1 Punctuation Tags

Table C.1 Listing of 12 Punctuation Tags of Penn Tree-Bank

Tags No.	Symbol	Description
1	#	Pound Sign
2	$	Dollar Sign
3	.	Sentence-final Punctuation
4	,	Comma
5	;	Colon, Semi-Colon
6	(	Left Bracket Character
7	)	Right Bracket Character
8	==	Straight Double Quote’
9	‘	Left Open Single Quote
10	“	Left Open Double Quote
11	’	Right Close Single Quote
12	”	Right Close Double Quote

C.2 POS Tags of the Penn Tree-Bank

Table C.2 Listing of 36 POS Tags of Penn Tree-Bank

Tags No.	Non-terminal	Description
1	CC	Coordinating Conjunction
2	CD	Cardinal Number
3	DT	Determiner
4	EX	Existential ("there")
5	FW	Foreign Word
6	IN	Preposition/Subordinate Conjunction
7	JJ	Adjective
8	JJR	Adjective, Comparative
9	JJS	Adjective, Superlative
10	LS	List Item Marker
11	MD	Modal
12	NN	Noun, Singular or Mass
13	NNS	Noun, Plural
14	NNP	Proper Noun, Singular
15	NNPS	Proper Noun, Plural
16	PDT	Pre-determiner
17	POS	Possessive Ending (i.e. Taniza's Thesis)
18	PRP	Personal Pronoun
19	PPS	Possessive Pronoun
20	RB	Adverb
21	RBR	Adverb, Comparative
22	RBS	Adverb, Superlative
23	RP	Particle
24	SYM	Symbol, Mathematical or Scientific
25	TO	To
26	UH	Interjection
27	VB	Verb, Base form
28	VBD	Verb, Past Tense
29	VBG	Verb, Gerund/Present Participle
30	VBN	Verb, Past Participle
31	VBP	Verb, Non-3rd person, Singular, Present.
32	VBZ	Verb, 3rd person, Singular, Present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WPS	Possessive Wh-pronoun
36	WRB	Wh-adverb

C.3 Syntactic Tags

Table C.3 Listing of Syntactic Tags of Penn Tree-Bank

Tags No.	Syntactic Tags	Description
1	ADJP	Adjective Phrase
2	ADVP	Adverb Phrase
3	NP	Noun Phrase
4	PP	Prepositional Phrase
5	S	Simple Declarative Clause
6	SBAR	Clause introduced by Subordinate Conjunction
7	SBARQ	Direct Question introduced by WH-word or Phrase
8	SINV	Declarative Sentence, Subject-Auxiliary Inversion
9	SQ	Sub-constituent of SBARQ excluding Wh-word
10	VP	Verb Phrase
11	WHADVP	Wh-Adverbial Phrase
12	WHNP	Wh-Noun Phrase
13	WHPP	Wh-Prepositional Phrase
14	X	Constituent of Unknown Category

C.1 Clause Level Notations

S (Simple Declarative Clause): This clause is not introduced by a subordinating conjunction or wh-word. This clause also does not show the subject verb inversion.

SBAR (Subordinate Clause): Clause that is introduced by a subordinating conjunction.

SBARQ: Direct question introduced by a wh-word or wh-phrase.

SINV: Inverted declarative sentence, subject is inverted. In other words, subject follows the tensed verb or modal.

SQ: Sub-constituent of SBARQ, which does not include wh-question.

C.2 Phrase level Notation

ADJP: Adjective Phrase. The phrase is headed by an adjective.

ADVP: Adverb Phrase. This phrase acts in the position of an adverb.

CONJP: Conjunction Phrase. This phrase is used to indicate several multi-word conjunctions.(For example, 'as well as', 'instead of'.)

FRAG: Fragment.

INTJ: Interjection, used instead of the POS tag UH.

LST: List marker is used to include the surrounding punctuation.

NAC: Not a constituent. This is used to show the scope of certain pre-nominal modifiers within a noun phrase.

NP: Noun phrase.

NX: To mark the head of the noun phrase, this is used with the complex noun phrase.

PP: Prepositional Phrase.

PRN: Parenthetical

PRT: Particle, same as the 'RP' tag in the POS tagset.

QP: Quantifier Phrase used within the noun phrase.

RRC: Reduced relative clause.

UCP: Unlike Coordinated Phrase.

VP: Verb Phrase.

WHADJP: Wh- adjective phrase, an adjectival phrase containing a wh-adverb.

WHADVP: Wh-adverb phrase contains a wh-adverb such as *how*.

WHNP: Wh-noun phrase contains some wh-words such as, "who", "which book".

WHPP: Wh-prepositional phrase. This is a prepositional phrase containing a wh-noun phrase.

X: Unknown constituent.

C.3 Function Tags

_ADV: Adverbial tag marks a constituent other than ADVP or PP when it is used adverbially. Constituents that themselves are modifying an ADVP generally are not tagged _ADV. Sometimes more specific adverbial tag is also used, like _TMP tag can be used to imply _ADV tag for the word "yesterday".

_NOM: Nominal tag is used to mark free relatives and gerunds when they act nominally.

C.4 Grammatical Role

_DTV: Dative tag is used to mark the dative object in the unshifted form of the double object construction. If the preposition "for" is used to introduce the "dative" object, then it is tagged as _BNF (benefactive). _DTV or _BNF tag can only be used after verbs that can undergo dative shift.

_LGS: Logical subject is tagged with the _LGS. It is attached to the NP object and not to the PP node itself in passives.

_PRD: This tag is used to mark any predicate that is not a verb phrase VP.

_PUT: This tag marks the locative complement of the word "put".

_SBJ: Surface subject tag _SBJ marks the structural surface subject of every clause including those with the null subject.

_TPC: Topicalized tag _TPC marks elements that appear before the subject in a declarative sentence for restricted cases.

_VOC: The vocative tag _VOC marks the nouns of address, regardless of their position in the sentence.

C.5 Adverbial Tag

_BNF: The benefactive tag marks the beneficiary of an action. It is usually attached to NP or PP. This tag is used only when the verb exhibits dative shift or the prepositional variant.

_DIR: The direction tag is used to mark adverbials that answer the questions "from where" and "to where". This tag is used mostly with verbs of motion, and financial verb.

_EXT: This extent tag _EXT marks the adverbial phrases that describe the spatial extent of an activity for example, the noun phrase "five-miles". However obligatory complements don't receive _EXT tag. Words like "fully" or "completely" are absolute and are not tagged with _EXT.

_LOC: This locative tag marks adverbials that indicate place or setting of the event. In cases of apposition involving SBAR, the SBAR can not be labeled as _LOC.

_MNR: The manner tag _MNR marks adverbials that indicate the manner.

_PRP: The purpose tag _PRP is used to mark the purpose or reason clauses and PPs.

_TMP: The temporal tag marks temporal or aspect adverbials that answer the questions *when, how often*. It is also used to tag the NP that indicates dates or time. The tag _TMP is not used with the possessive phrases.

C.6 Miscellaneous Phrase-Tags

_CLR: The tag _CLR (closely related) marks constituents that occupy the intermediate ground between argument and adjunct of the verb phrase. In a broad sense, the tag corresponds to the "prediction adjuncts", prepositional ditransitives, and some phrasal verbs. The precise meaning of this tag depends on the category of its phrase.

_CLF: The cleft tag _CLF marks it-clefts, as well as true clefts. It can be added to the labels S, SINV or SQ.

_HLN: The Tag _HLN marks headlines and datelines. The headlines and datelines always constitute a unit of text, and are structurally independent.

_TTL: While the title comes inside of a text, the title tag is attached to the top node of a title.

C.7 Null Elements

***T*:** Trace of A movement.

(NP*): Trace of A movement, or arbitrary PRO.

0: The null complementizer.

***U*:** unit.

***?*:** placeholder for ellipsoid material.

***NOT*:** anti placeholder in template gapping.

Identity Index: Identity index is a number following a bracket tag and is used as an identity number for that constituent. It usually appears when there is a null element.

The Reference Index: Reference index is that number which follows the null element. It corresponds to the identity index of the constituent with which the null is associated.

Pseudo-attach: Pseudo attach is used to show that non-adjacent constituents are related. Four different types of pseudo-attach are used to show the relations. These four types of pseudo-attach are listed below:

EXP: Expletive tag for extraposition.
ICH: Interpret Constituent Here tag to denote the discontinuous dependency.
PPA: Permanent Predictable Ambiguity tags (ambiguity).
RNR: Right node raising tag for shared complements.

Parenthetical: The nodes labeled as PRN dominates the parenthetical elements. PRN node contains the punctuation marks such as commas, dashes, and parentheses.

Appendix D: A Sample Test-Set of Simple Noun-Phrases

This section includes the test-set containing manually extracted simple noun-phrases for the input document "Input3.txt." Input3.txt contains 14 sentences from two US patent documents and is given below.

"US Patent Number: 4404650 .
Title: Method and circuit arrangement for transmitting binary signals between peripheral units which are connected to one another via a central bus line system. [ABSTRACT]
A method and a circuit arrangement are disclosed for transmitting binary signals between peripheral units which are connected to one another via a central bus line system. The signal transmission is to take place without the necessity of including a central processor, which is likewise connected to the central bus system. For this purpose, it is provided that signal transmission requests from the individual peripheral units are transmitted to a memory access device which is also connected to the central bus line system and which, in response to the receipt of such transmission request, de-activates the central processor in respect of the transmission of signals to the central bus line system. The memory access device supplies the peripheral units which are to participate in a signal transmission operation with control signals which transform the latter from a starting state into a transmitting state or into a receiving state and then, in alternation with control signals which serve for the output and receipt of signal groups. Following the transmission of the last signal of the number of signals reported to the memory access device in respect of a signal transmission operation, the memory access device supplies the peripheral units with a reset signal by which the same are returned from their transmitting and receiving states into their starting states.

US Patent number: 4455620 .
In the direct memory access mode, the data bus separating circuit is controlled, such that, the data bus being connected to the input/output controller and the memory is separated from the central processing unit. The first address bus switch circuit is controlled, so that,

the address bus connected to the direct memory access controller is switched to the memory and the second address bus switch circuit is controlled such that the address bus connected to the memory is switched to the direct memory access controller. Therefore, according to the embodiment, the data transfer line in the central processing unit mode and the data transfer line in the direct memory access mode can be selectively switched with a relatively simple structure. Accordingly, a principle object of the present invention is to provide a direct memory access control apparatus in which a central processing unit can perform such an operation as interrupt processing independently, even during direct memory access transfer of the data between a high speed input/output control device and a memory. One aspect of the present invention resides to selectively switch a data transfer line in a central processing unit mode and a data transfer line in a direct memory access mode with a relatively simple structure. These objects and other objects, features, aspects and advantages of the present invention will become more apparent from the following detailed description of the present invention when taken in conjunction with the accompanying drawings."

Listed below are the manually extracted simple noun phrases for each sentence following immediately after that particular sentence. The number of each sentence (starting from "0") precedes the noun-phrases which were manually extracted from that sentence. The word "end" follows the simple noun-phrases to indicate that the entire sentence has been analyzed and a full-stop (.) sign has been encountered in the sentence. The simple noun-phrases taken from the test-set for this input document are included below.

US Patent Number: 4404650.
| 0 |
~ 1 US Patent Number ~
~ 2 4404650 . ~
end

Title: Method and circuit arrangement for transmitting binary signals between peripheral units which are connected to one another via a central bus line system.

| 1 |
~ 1 Title ~
~ 2 Method ~
~ 3 circuit arrangement ~
~ 4 transmitting binary signals ~
~ 5 peripheral units ~
~ 6 one another ~
~ 7 a central bus line system . ~
end

[ABSTRACT]

A method and a circuit arrangement are disclosed for transmitting binary signals between peripheral units which are connected to one another via a central bus line system.

| 2 |
~ 1 ABSTRACT ~
~ 2 A method ~
~ 3 a circuit arrangement ~
~ 4 transmitting binary signals ~
~ 5 peripheral units ~
~ 6 one another ~
~ 7 a central bus line system . ~
end

The signal transmission is to take place without the necessity of including a central processor, which is likewise connected to the central bus system.

| 3 |
~ 1 The signal transmission ~
~ 2 the necessity ~

~ 3 including a central processor ~
~ 4 the central bus system . ~
end

For this purpose, it is provided that signal transmission requests from the individual peripheral units are transmitted to a memory access device which is also connected to the central bus line system and which, in response to the receipt of such transmission request, de-activates the central processor in respect of the transmission of signals to the central bus line system.

| 4 |
~ 1 this purpose ~
~ 2 signal transmission requests ~
~ 3 the individual peripheral units ~
~ 4 a memory access device ~
~ 5 the central bus line system ~
~ 6 the receipt ~
~ 7 such transmission request ~
~ 8 the central processor ~
~ 9 the transmission ~
~ 10 signals ~
~ 11 the central bus line system . ~
end

The memory access device supplies the peripheral units which are to participate in a signal transmission operation with control signals which transform the latter from a starting state into a transmitting state or into a receiving state and then, in alternation with control signals which serve for the output and receipt of signal groups.

| 5 |
~ 1 The memory access device ~
~ 2 the peripheral units ~
~ 3 a signal transmission operation ~

~ 4 control signals ~
~ 5 the latter ~
~ 6 a starting state ~
~ 7 a transmitting state ~
~ 8 a receiving state ~
~ 9 alternation ~
~ 10 control signals ~
~ 11 the output ~
~ 12 receipt ~
~ 13 signal groups . ~
end

Following the transmission of the last signal of the number of signals reported to the memory access device in respect of a signal transmission operation, the memory access device supplies the peripheral units with a reset signal by which the same are returned from their transmitting and receiving states into their starting states.

| 6 |
~ 1 Following the transmission ~
~ 2 the last signal ~
~ 3 the number of signals ~
~ 4 the memory access device ~
~ 5 a signal transmission operation ~
~ 6 the memory access device ~
~ 7 the peripheral units ~
~ 8 a reset signal ~
~ 9 the same ~
~ 10 their transmitting ~
~ 11 receiving states ~
~ 12 their starting states . ~
end

US Patent number: 4455620 .

| 7 |

~ 1 US Patent number ~

~ 2 4455620 . ~

end

In the direct memory access mode, the data bus separating circuit is controlled, such that, the data bus being connected to the input/output controller and the memory is separated from the central processing unit.

| 8 |

~ 1 the direct memory access mode ~

~ 2 the data bus separating circuit ~

~ 3 the data bus being connected ~

~ 4 the input/output controller ~

~ 5 the memory ~

~ 6 the central processing unit . ~

end

The first address bus switch circuit is controlled , so that, the address bus connected to the direct memory access controller is switched to the memory and the second address bus switch circuit is controlled such that the address bus connected to the memory is switched to the direct memory access controller.

| 9 |

~ 1 The first address bus switch circuit ~

~ 2 the address bus ~

~ 3 the direct memory access controller ~

~ 4 the memory ~

~ 5 the second address bus switch circuit ~

~ 6 the address bus ~

~ 7 the memory ~

~ 8 the direct memory access controller . ~
end

Therefore, according to the embodiment, the data transfer line in the central processing unit mode and the data transfer line in the direct memory access mode can be selectively switched with a relatively simple structure.
| 10 |
~ 1 the embodiment ~
~ 2 the data transfer line ~
~ 3 the central processing unit mode ~
~ 4 the data transfer line ~
~ 5 the direct memory access mode ~
~ 6 a relatively simple structure . ~
end

Accordingly, a principle object of the present invention is to provide a direct memory access control apparatus in which a central processing unit can perform such an operation as interrupt processing independently, even during direct memory access transfer of the data between a high speed input/output control device and a memory.
| 11 |
~ 1 a principle object ~
~ 2 the present invention ~
~ 3 a direct memory access control apparatus ~
~ 4 a central processing unit ~
~ 5 such an operation ~
~ 6 interrupt processing independently ~
~ 7 direct memory access transfer ~
~ 8 the data ~
~ 9 a high speed input/output control device ~
~ 10 a memory . ~

end

One aspect of the present invention resides to selectively switch a data transfer line in a central processing unit mode and a data transfer line in a direct memory access mode with a relatively simple structure.

| 12 |
~ 1 One aspect ~
~ 2 the present invention ~
~ 3 a data transfer line ~
~ 4 a central processing unit mode ~
~ 5 a data transfer line ~
~ 6 a direct memory access mode ~
~ 7 a relatively simple structure . ~
end

These objects and other objects, features, aspects and advantages of the present invention will become more apparent from the following detailed description of the present invention when taken in conjunction with the accompanying drawings .

| 13 |
~ 1 These objects ~
~ 2 other objects ~
~ 3 features ~
~ 4 aspects ~
~ 5 advantages ~
~ 6 the present invention ~
~ 7 the following detailed description ~
~ 8 the present invention ~
~ 10 the accompanying drawings . ~
end

References

[1] Caraballo, A., S., and Charniak, E., "New Figures of Merit for Best-First Probabilistic Chart Parsing," proceedings of the Conference on Empirical Methods in Natural Language Processing, 1996, pp. 127-132.

[2] Charniak, E., " Statistical Techniques for Natural Language Parsing," Proceedings of the 14th National Conference on Artificial Intelligence, AAAI Press/MIT Press, 1997.

[3] Charniak, E., "Statistical parsing with a context free grammar and word statistics," Proceedings of the 14th National Conference on Artificial Intelligence, Menlo Park, AAAI Press/MIT Press, 1997.

[4] Charniak, E., "Tree Bank Grammars," proceedings of the 13th National Conference on Artificial Intelligence, Menlo Park, AAAI Press/MIT Press, 1996, pp. 1031-1036.

[5] Charniak, E., "Statistical Language Learning," MIT Press, 1997.

[6] Church, K., W., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," Proceedings of the 2nd Conference on Applied Natural Language Processing, ACL, 1988, pp. 136-143.

[7] Collins, M., J., "Three generative lexicalized models for statistical parsing," Proceedings of the 35th Annual Meeting of the ACL, 1997, pp. 16-23.

[8] Collins, M., J., "A new statistical parser based on bigram lexical dependencies," Proceedings of the 34th Annual Meeting of the ACL, 1996, pp. 184-191.

[9] Cyre, W., R., "Extracting Design Models form Natural Language Descriptions," Tenth International Conference on Industrial Engineering Applications of Artificial Intelligence Expert Systems, Atlanta, Georgia, 1997.

[10] Cyre, W. R., "Design of Restricted Sublanguage", Proceedings of Theoretical Approaches to Natural Languages Understanding Workshop, Halifax, Novia Scotia, 1985.

[11] Jason, M., Eisner, "Three New Probabilistic Models for Dependency Parsing: An Exploration," Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, August 1996, pp. 340-345.

[12] Jelinek, Fred, "Markov source modeling of text generation," journal of Impact of Processing Techniques on Communication, Dordrecht, 1985.

[13] Jones, A., Mark, and Eisner, M., Jason, "A Probabilistic Parser and Its Application," AAAI workshop on Statistically Based NLP Techniques, Menlo Park, San Jose, CA, 1992, pp. 20-27.

[14] Maggerman, D., M., "Statistical Decision-tree Models for Parsing," Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, 1995, pp. 276-283.

[15] Maggerman, D., M., "Natural Language Parsing as Statistical Pattern Recognition," Ph.D. Thesis, Stanford University, 1994.

[16] Manning, C., D., and Schutze, H., "Foundations of Statistical Natural Language Processing," the MIT press, 1999.

[17] Marcus, P., M., Santorini, B., and Marcinkiewicz, A., M., "Building a Large Annotated Corpus of English: the Penn Treebank," Department of Computer and Information Science, University of Pennsylvania.

[18] Merialdo, B., "Tagging text with a probabilistic model," proceedings of the IBM Natural Language ITL, Paris, France, 1990, pp. 161-172.

[19] Ritesh, D., S., "Phrasal Document Analysis for Modeling," MS Thesis, Virginia-Tech University, 1998.

[20] Stark, H., and Woods, J., W., "Probability, Random Processes, and Estimation Theory for Engineers," Prentice Hall, Upper Saddle River, NJ 07458, 2nd Edition, 1986.

[21] Voutilainen, A., Helsinki, "A Noun Phrase Parser of English," Ohio State University, Ohio-USA, 1992.

[22] Wardhaugh, R., "Understanding English Grammar: A Linguistic Approach," Blackwell Publishers, Oxford UK & Cambridge USA, 1995.

Vita

Taniza Afrin was born in Dhaka, Bangladesh. She completed her undergraduate study from Bangladesh University of Engineering and Technology in September 1997. Then, she worked as a faculty member in the Electrical and Computer Engineering Department of the same university for a year and half. In January 1999, she enrolled at Virginia Polytechnic Institute and State University to pursue her Masters in Electrical Engineering. Taniza will be working for Hughes Network System, Maryland.