Automated Acquisition of Technical Concepts From Unrestricted English Text Using Noun Phrase Classification

Christine Laurendeau

Department of Computer Science
University of Ottawa

A Thesis Submitted to
The School of Graduate Studies and Research and
The Ottawa-Carleton Institute for Computer Science
In Partial Fulfilment of the Requirements
For the Degree of
Master of Computer Science

ISBN 0-315-85812-5

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Acknowledgements

The author is indebted to Dr. Stan Szpakowicz for his supervision and helpful suggestions during the research and writing of this thesis.
The author also appreciates the invaluable suggestions she received from Mr. Sylvain Delisle.
Finally, she would also like to acknowledge the financial assistance provided under a contract with Cognos Incorporated and grants from the Ontario Technology Fund and the Natural Science and Engineering Research Council.

Abstract

This thesis describes an approach to acquire technical concepts from an English language free text without use of knowledge specific to the domain of expertise described in the text. Only syntactic knowledge and text statistics are used to classify each Noun Phrase in the text into one of five categories of technicality, from **Technical** to **Not Technical**. The algorithms devised and their performance are discussed.

A secondary topic addressed in this thesis is syntactic category disambiguation. Because the Noun Phrase Classification module requires a Sentence Parser to extract the syntactic structure of each sentence in the text, the syntactic category (noun, verb, preposition, and so on) of each word must appear in the Sentence Parser's Word Dictionary. A syntactic category disambiguation module was designed so that whenever an unknown word (a word which is not defined in the Word Dictionary) is encountered in the text, the disambiguation module attempts to determine its syntactic category automatically using the categories of the neighbouring words with a bottom-up chart parser and text statistics.

Table of Contents

Table of Figures

Table of Tables

Introduction

In this thesis, I will describe the AZTEC system (AnalyZer for TEchnical Concepts), which automatically acquires a set of technical concepts from an English language text on any technical domain of expertise. Given that AZTEC must be able to process a source text on any domain, no semantic or pragmatic knowledge can be used in the concept acquisition process. AZTEC is therefore restricted to using general syntactic knowledge and text statistics in order to classify each concept of the text into the most likely of several categories of technicality. An automatically acquired set of technical concepts can assist a database administrator in creating a conceptual model of the domain described in the text. The more technical concepts will denote the entities to be represented in the domain model. Although the acquisition of relationships between entities mentioned in a text is feasible, it is outside the scope of this thesis and may be considered as a future enhancement.

A secondary function of the AZTEC system is to simplify, from the user's point of view, the problem of unknown words. In order to extract concepts from the technical text, AZTEC uses a Sentence Parser which requires a lexicon of all the words in the text along with their syntactic category (for example, *noun*, *verb*, *preposition*, and so on). However, the existing Word Dictionary associated with the Sentence Parser is far from complete, especially in terms of open-class words (*nouns*, *verbs*, *adjectives* and *adverbs*). In an effort to minimize user interaction with AZTEC when the Sentence Parser encounters an unknown word (i.e. a word which does not appear in the Word Dictionary), the Syntactic Category ANalyzer (SCAN) module was designed to automatically determine the syntactic category of the unknown word and add it to the Word Dictionary.

Both AZTEC and the Sentence Parser were developed at the Knowledge Acquisition Laboratory of the University of Ottawa's

Department of Computer Science as part of the TANKA (formerly KATE) project, which is intended for the construction of a conceptual model of a domain described in an un-edited technical text.[1]

The general organization of this thesis is the following. Chapter 1 will provide an overview of the AZTEC system, explaining the main assumptions that guide its design, and describing its four main components, the Text Base Builder, the Classification Module, the Sentence Parser and the SCAN module.

Chapter 2 delves into the details of AZTEC's two concept classification algorithms: Classification by Sentence (section 2.1) and Classification by Block (section 2.2). The interface module between the Classification algorithms and the Sentence Parser is described in section 2.3. Section 2.4 chronicles the evolution of the Classification algorithms in order to better document and justify the decisions made. Section 2.5 discusses the Classification results obtained and the conclusions reached with the help of those results. Section 2.6 explains the limitations of AZTEC's classification process, section 2.7 provides a framework of related research, and section 2.8 discusses future enhancements to improve the Classification algorithms and their results.

Chapter 3 describes the SCAN module: the assumptions associated with it are given in section 3.1; the two main approaches used to automatically determine an unknown word's syntactic category are described in sections 3.2 (the Chart Parser approach) and 3.3 (the Collocational Probability Matrix approach); section 3.4 gives the step-by-step SCAN algorithm, combining the methods described in sections 3.2 and 3.3; section 3.5 provides an overview of the results obtained with SCAN; section 3.6 discusses the limitations of the SCAN algorithm; and section 3.7 gives an outline of previous research in the area of syntactic category disambiguation.

---

[1]Stan Szpakowicz, "Semi-Automatic Acquisition of Conceptual Structure from Technical Texts," International Journal of Man-Machine Studies 33 (1990): 385-397.

Finally, the conclusion discusses the overall performance of the Classification algorithms and the SCAN module. Appendices 1 through 5 contain the results of various versions of the Classification algorithms executed on sentences taken from chapters 1 and 2 of the PowerHouse QUIZ manual.[1] Appendices 6 and 7 provide a manually compiled list of the concepts found in chapters 1 through 6 of the QUIZ manual and Sylvain Delisle's 1990 paper describing the Sentence Parser,[2] along with the classification categories to which I believe those concepts belong. These appendices are used in section 2.5 for comparison with the actual results computed by the Classification algorithms so that a success rate can be approximated. Appendices 8 and 9 list the concept classification results using QUIZ chapters 1 through 6 and the Parser text.

---

[1]Cognos Incorporated, PowerHouse QUIZ (1985).

[2]Sylvain Delisle, A Parser for Processing Technical Texts With a Large Coverage of English TR-90-25. University of Ottawa, Department of Computer Science (1990).

Chapter 1
Overview of the AZTEC System

The **AZTEC** system automatically acquires from an English language technical text a set of its most technical concepts. To do it, AZTEC tentatively classifies all of the text's concepts into one of five categories of technicality: **Technical, Rather Technical, Possibly Technical, Not Technical,** or technicality **Unknown.**

In this thesis, I will demonstrate to what extent such a concept classification is possible using only the syntactic and statistical information directly available from the text, namely the frequency of occurrence of Noun Phrases (NPs), and the result of a comparison between the words of a NP and an existing list of about 2000 of the most common English words.

This chapter will provide an overview of the main assumptions in developing the AZTEC system, and it will describe its four main components. These are the Text Base Builder which converts un-edited text into an organized file of sentences, the Sentence Parser which parses a text sentence and returns its syntactic structure to the Classification Module, the Classification Module which extracts concepts from those syntactic structures and classifies them in order of technicality, and the Syntactic Category ANalyzer (SCAN) module which attempts to determine the syntactic categories of unknown words in the sentence.

1.1 Main Assumptions

The principal assumption in building AZTEC was that it should be applicable to technical texts on any subject. Such a restriction of domain-independence meant that no semantic or pragmatic domain

knowledge would be available when processing the text and classifying its concepts. Hence the only type of knowledge which could be used in the Classification process had to be obtained either directly from the text's syntax or from statistics which could be gathered on the text or its syntactic structures.

Given this restriction, it was assumed that the text's Noun Phrases (NPs) would be the best syntactic structures to analyze in order to extract the text's concepts. Nouns generally denote the entities in sentences, whereas verbs are used to indicate the relationships between these entities. All NPs constructed of a head noun and zero or more modifiers (e.g. *the dog*, *the brown dog*, *the big guard dog*) are considered for concept classification, as are NPs built around gerunds (e.g. "*The writing on the wall* said it all"), regardless of the syntactic structure (e.g. NP, Verb Phrase, Prepositional Phrase, embedded sentence, and so on) in which they occur. Pronouns (e.g. *you*, *us*, etc.) and embedded sentences used as NPs (e.g. "*Writing on the wall* was an exercise in futility") are not considered, because tools are not currently available for the Sentence Parser to conduct anaphora resolution on the pronouns, and embedded sentences used as NPs generally do not describe simple concepts, but rather actions or situations: "while noun phrases are used to refer to things, sentences . . . are used to assert, query, or bring about some partial description of the world."[1]

Another major assumption in AZTEC is the use of frequencies of occurrence to drive the concept classification. For AZTEC's purposes, NPs which occur most frequently in the text are assumed to be key concepts in the text's domain. They are classified as more technical concepts than NPs which do not appear as much in the text. The rationale behind proceeding with the assumption that technical words tend to occur frequently is that a system such as AZTEC, which uses strictly syntactic knowledge to process a text, can have no sense of what it means for a concept to be technical. The best it can hope to

---

[1]James Allen, Natural Language Understanding (Menlo Park, CA: The Benjamin/Cummings Publishing Company, 1987), 30.

do is to narrow down what it is that the text is *about*, i.e. which concepts are used most frequently to describe the text's domain. It is logical to suggest that whatever domain an author describes, she will use that particular domain's technical terms repeatedly in definitions and descriptions. This assumption is in keeping with even very early work on text statistics:

> . . . whatever the topic, the closer certain words are associated, the more specifically an aspect of the subject is being treated. Therefore, wherever the greatest number of frequently occurring different words are found in greatest physical proximity to each other, the probability is very high that the information being conveyed is most representative of the article.[1]

Inevitably, along with those key concepts describing the text's domain, some general, common usage, multi-domain concepts, such as *object* or *item*, are bound to occur very frequently. Some of the work in automatic book indexing shows that in addition to term frequency, an inverse document frequency is considered in ranking concepts.[2] The inverse document frequency takes into account the number of documents in which the term appears, so that multi-domain concepts which appear in several documents can be weeded out. Unfortunately, AZTEC processes only one document at a time and cannot make use of this measure. So in order to minimize the number of non-technical generic concepts being classified as technical, AZTEC uses Longman's list of 2158 common English words as a reference point. This list was originally put together to provide the smallest subset of words from which every definition in Longman's Dictionary of Contemporary English could be built. Therefore, if a user of Longman's Dictionary is familiar with all 2158 common words, she will possess the tools to understand every definition in Longman's Dictionary.

---

[1]H. P. Luhn, "The Automatic Creation of Literature Abstracts," IBM Journal of Research and Development 2 (April 1958): 160-161.

[2]Martin Dillon and Peggy Federhart, "The Use of Discriminant Analysis to Select Content-Bearing Words," Journal of the American Society for Information Science 33 (1982): 245.

The head noun and modifiers of a NP to be classified are compared against Longman's list. If a word does appear in the list, it is considered less likely to be technical, and thus a frequently occurring word which appears in Longman's list will be bumped down on the scale of technicality toward the **Not Technical** category. Similarly, a less frequently occurring word which does not appear in Longman's list has a good chance of being technical, and in this case the associated NP will be moved up on the scale of technicality toward the **Technical** category. The rationale behind this assumption is that the common words in Longman's list are generally common-usage words, used in texts of any domain, and not usually used as technical terms in technical texts. They can therefore be invaluable in weeding out the frequently occurring non-technical terms (for example, the common-usage words) and highlighting the infrequent terms which may be technical.

Another assumption in developing AZTEC, and one that is used mainly to discriminate between seemingly equal courses of action, is that a concept is better over-classified (classified in a more technical category) than under-classified. This is mainly so that the user can safely ignore the lower, less technical classifications without missing any technical NPs. As a result of this assumption, users will have to sift through a greater number of less technical NPs in the higher categories, but at least they will be able to ignore the **Not Technical** category, knowing that no technical NPs have been mis-classified there. This assumption came into play at the stage when I was refining AZTEC's classification algorithms by examining their results and determining which changes to make to improve them. The evolution of the classification algorithms will be chronicled in Chapter 2, section 2.4, so that the assumptions and decisions made throughout the development of AZTEC can be documented and justified.

A final assumption relates to the importance of minimizing AZTEC's interaction with the users, prior to their acceptance of AZTEC's classification of each NP. In fact, this is the assumption which

led to the implementation of the SCAN module, a syntactic category disambiguator for the unknown words encountered in the text by the Sentence Parser. The SCAN module will be described in detail in Chapter 3. Since all the classified NPs are displayed to the user at once, one of the future enhancements to AZTEC could involve a screening out process, where each NP would in turn be presented to the user in the context of the sentences in which it appears. The user would then either approve of the associated concept as a technical one, or reject it.

1.2 Major Components

The first step in acquiring any concept from the technical text consists in structuring the text in such a manner that it becomes easy for the Classification Module to access it automatically. To this end, the AZTEC system uses a Text Base Builder, implemented by Denis Fauconnier and Jean-Baptiste Poline at the Knowledge Acquisition Laboratory of the University of Ottawa's Computer Science Department. Once the Text Base Builder has organized the text into a Text Base, AZTEC can invoke its Classification Module to extract the most likely technical concepts from the Text Base, using a Sentence Parser to extract the syntactic structure of each Text Base sentence, and the SCAN module to ensure that each of the words in the sentence is defined in the Sentence Parser's Word Dictionary (Figure 1).

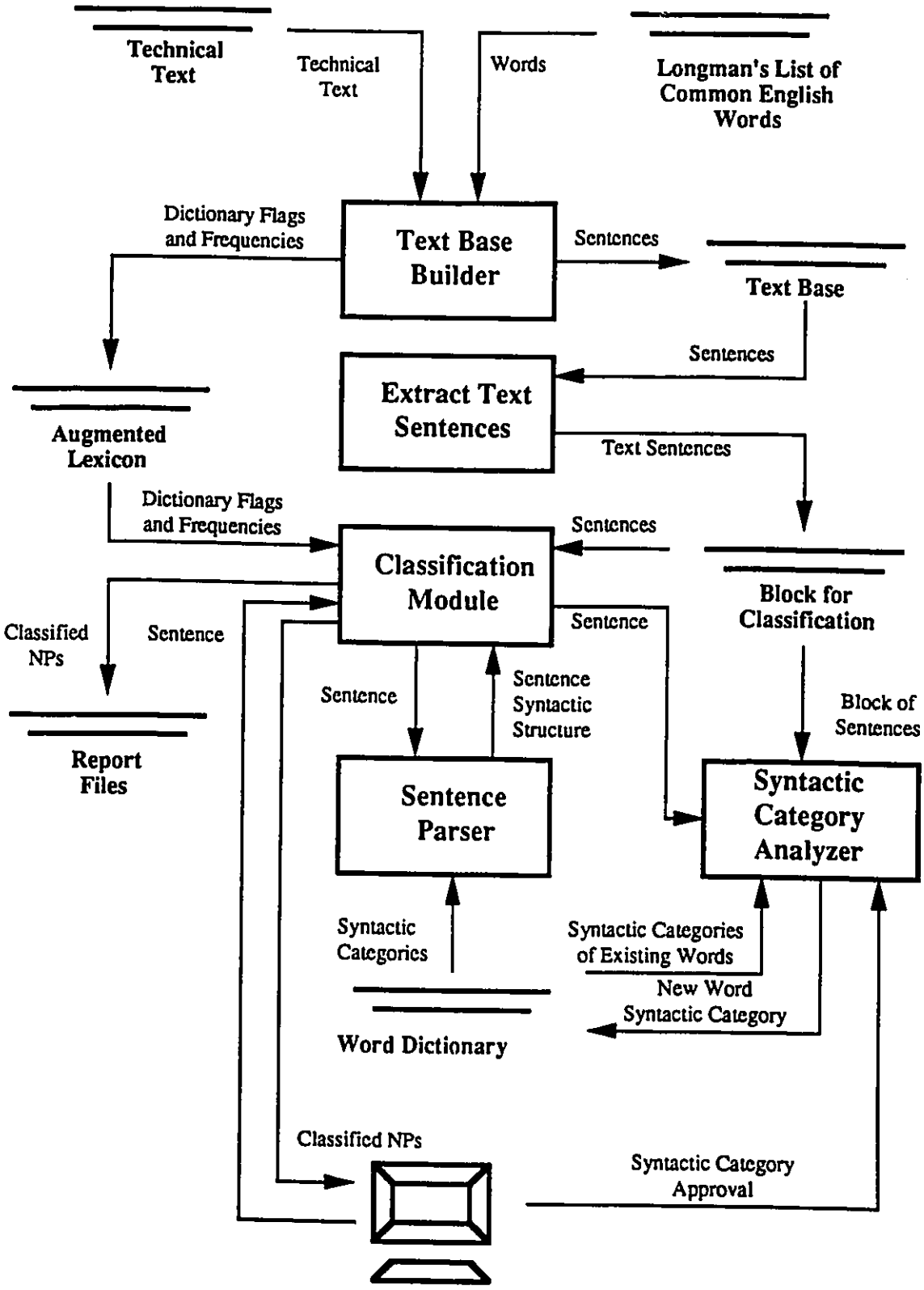


Figure 1 - AZTEC

1.2.1 The Text Base Builder

The Text Base Builder (TBB) reads an un-edited English text from a file, and builds a text hierarchy, with chapters at the highest level of the hierarchy, and with the text's individual sentences at the lowest level. Each level of the text hierarchy is associated with one file of the Text Base. For its purposes, AZTEC only needs the *sentence* file, which contains all of the text's individual sentences. However, before the Classification Module can use the Text Base, a utility module called *Extract Text Sentences* must be invoked in order to remove from the Text Base the sentence fragments which the TBB extracted from the tables and figures in the text. Because only complete sentences are considered for concept classification, the Extract Text Sentences module creates a new file, *Block for Classification*, containing only the complete sentences from the Text Base. All subsequent processing uses the Block for Classification as its text base.

In order to provide AZTEC's Classification Module with the statistical information it needs to determine the level of technicality of each concept, the TBB had to be modified to create an *Augmented Lexicon* of all the words found in the text, along with the needed information.

The Augmented Lexicon, so named because it contains more than the lexical information typical of conventional lexicons, is organized with one entry for each word found in the text, along with the word's frequency of occurrence in the text, and its *dictionary flag*. The dictionary flag, consisting of a simple "y" or "n", tells us whether or not the word appears in Longman's list of 2158 common English words. The purpose of this flag is to help refine the Classification process by discriminating between frequent technical and non-technical NPs, and between infrequent ones.

1.2.2 The Classification Module

Once the TBB has set up the Text Base and created the Augmented Lexicon, the Classification Module is ready to begin its tentative classification of the concepts found in the sentences of the Block for Classification.

Since Noun Phrases (NPs) represent the objects, and thus the concepts, of a text, they are the structures to be analyzed and subsequently classified in the Classification Module. As will be described in detail in Chapter 2, two types of Classification were implemented in the AZTEC system: Classification by Sentence and Classification by Block. The reason for both types is that Classification by Sentence was implemented first, and it reflects the logic behind the first generation of AZTEC's classification algorithms. When it evolved into Classification by Block, I decided to make both types of Classification available to the user. The essential difference between the two algorithms is that in Classification by Sentence the user is asked to provide the sentence to be analyzed, and the NPs of this sentence are classified using the single word frequencies found in the Augmented Lexicon. With Classification by Block, however, the user opts to process the entire Block for Classification at once, meaning that it is possible, and far more interesting, to use *NP* frequencies as the NPs occur in the block of text, rather than the single *word* frequencies found in the Augmented Lexicon. The subtle differences between both types of frequencies, and the inherent superiority of the second over the first, will become more apparent in Chapter 2 when both types of Classification are described.

For each sentence that is processed, either obtained from the user or from the Block for Classification, the Classification Module will need to invoke the Sentence Parser to obtain the syntactic structure of that sentence. But first, it must ensure that each word in the sentence, along with its syntactic category (e.g. *noun*, *verb*, *adjective*, etc.) appears in the Sentence Parser's Word Dictionary. This is where the SCAN module comes in.

Before invoking the Sentence Parser, the Classification Module passes the sentence to the SCAN module, which examines each word and checks whether it appears in the Word Dictionary. If it does not, then SCAN will attempt to automatically determine the word's syntactic category, given the syntactic categories of the words preceding and following the unknown word. When invoked from Classification by Block, the SCAN module uses statistical information to determine the syntactic category of the unknown word, so the entire block is used by SCAN early in the Classification by Block process to accumulate the needed statistical information. Once each word of a sentence is known to have at least one entry in the Word Dictionary, the Sentence Parser is invoked and returns the sentence's syntactic structure to the Classification Module.

All of the NPs in the sentence are then extracted from its syntactic structure, and classified into one of the five concept categories, using the Augmented Lexicon's dictionary flags, and either its word frequencies as well or the NP frequencies, depending on the type of Classification selected by the user. The five concept categories of technicality are the following: **Technical, Rather Technical, Possibly Technical, Not Technical,** and technicality **Unknown.**

Once computed, the results of Classification by Sentence are displayed on the screen, and a report file is created for Classification by Block. In both cases, each NP found in the sentence or the block of text appears in the resulting NP Classifications, under the category in which AZTEC believes it to belong.

NP Classifications are generally linear, in the sense that the NPs are assigned to categories where the most technical concepts are usually the most frequent ones whose constituents do not appear in Longman's list, and where the least technical concepts are the least frequent ones whose constituents do appear in Longman's list. As a result, the category names only serve as labels to help the users determine a cut-off point below which they find the NPs too non-technical to be useful to the construction of a conceptual model of the

text's domain. In general, the **Technical** category will contain the NPs believed to be technical terms in the text's domain; the **Rather Technical** category will contain NPs slightly less technical than those found in the **Technical** category; the **Possibly Technical** category will contain the NPs believed to be perhaps remotely technical; the **Not Technical** category will contain the NPs believed to be non-technical; and the **Unknown** category will contain the NPs which are too infrequent in the text to be assigned to a category.

1.2.3 The Sentence Parser

The Sentence Parser[1,2] was implemented by Sylvain Delisle at the Knowledge Acquisition Laboratory of the University of Ottawa's Computer Science Department as part of the TANKA project. Since TANKA is intended for the construction of a conceptual model of a domain described in an un-edited technical text, the parser was designed to process unrestricted domain-independent text.

Of the five main modules comprised in the Sentence Parser, only two are used by AZTEC: the Word Dictionary, and the syntactic analyzer. Each entry of the Word Dictionary is associated with a word, along with its part of speech (i.e. its syntactic category) and other information related to the part of speech, although one word may have multiple entries in the Word Dictionary. *Light*, for example, will have one entry as a transitive verb (e.g. "She will *light* a candle"), one entry as a count noun (e.g. "There is only one stop *light* between here and there"), one entry as a mass noun (e.g. "*Light* was shining through"), and one entry as an adjective (e.g. "The package felt *light*"). The parser's syntactic analyzer uses "phrase structure rules expressed in the definite clause grammar formalism (DCG) of Prolog"[3] to verify a sentence's grammatical accuracy and to return a

---

[1]Delisle, Parser for Technical Texts.

[2]Sylvain Delisle, "A Broad-Coverage Parser for Knowledge Acquisition From Technical Texts," Proceedings of the Fifth International Conference on Symbolic and Logical Computing (1991): 169-183.

[3]Delisle, Parser for Technical Texts, 8.

structure indicating the syntactic units (e.g. NPs, Verb Phrases, Prepositional Phrases, etc.) associated with the sentence.

However, in order to decompose a sentence into its syntactic units, each of the words in the sentence must have one or more entries in the Word Dictionary. Therefore, when an unknown word (i.e. a word for which there is no Word Dictionary entry) is encountered by the Sentence Parser, it queries the user for the syntactic category associated with the word. Unfortunately, the Sentence Parser offers the user a menu of 24 possible syntactic categories, which can be understood if the user has some knowledge of linguistics. Since AZTEC makes no such assumption of its users, the SCAN module was devised in order to simplify the unknown word problem by acquiring the syntactic category of the unknown word and adding it to the Word Dictionary. As a result, since SCAN is always invoked before the parser, the Sentence Parser never encounters an unknown word.

1.2.4 The Syntactic Category ANalyzer (SCAN) Module

The SCAN module was devised to minimize user interaction in the event that a word for which there is no entry in the Sentence Parser's Word Dictionary is encountered. As will be described in Chapter 3, SCAN applies a chart parser to a segment of the sentence containing the unknown word. Since it is assumed that all the function words (prepositions, conjunctions, auxiliaries, etc.) appear in the Word Dictionary already, the chart parser will use a simple grammar of phrases (NPs, VPs, PPs, Adjective Phrases, etc.) to attempt to determine to which of the four content categories (*noun*, *verb*, *adjective*, or *adverb*) the unknown word is most likely to belong. Unfortunately, the chart parser approach has its limitations. In fact, because many words can be assigned to more than one syntactic category, it is often impossible to narrow down the four possible syntactic categories to less than two or three. For this

reason, a modified version of Steven J. DeRose's algorithm,[1] which requires the creation of a database with statistics on collocations, was implemented to take over where the chart parser approach leaves off.

In the end, the user is asked to approve the syntactic category found to be the most likely candidate for the unknown word. If the user does not approve, the next likely candidate is displayed, and so on. Once one of the four possible syntactic categories is approved, SCAN adds the appropriate entry to the Word Dictionary.

---

[1]Steven J. DeRose, "Grammatical Category Disambiguation by Statistical Optimization," Computational Linguistics 14 (Winter 1988): 31-39.

Chapter 2
Noun Phrase Classification

This chapter describes in detail the best algorithms, in terms of the quality of the results they yield, for Classification by Sentence and Classification by Block, along with the NP Seeker module they both use as an interface to the Sentence Parser. The evolution of classification algorithms, from Classification by Sentence tc Classification by Block, will be described in section 2.4, and the performance of the final algorithms will be discussed in section 2.5. This will lead to a discussion of the limitations inherent in AZTEC's approach (section 2.6), an overview of previous work related to other attempts at Knowledge Acquisition from text syntax and statistics (section 2.7), and directions for future work with AZTEC (section 2.8).

2.1 NP Classification by Sentence

In Classification by Sentence, one sentence from the technical text is provided by the user and processed (see Figures 2 and 3). Since both types of classification require the use of the Augmented Lexicon, it is assumed that the entire text has been pre-processed by the Text Base Builder (TBB), and therefore that Classification by Sentence will not process arbitrarily chosen sentences not appearing in the technical text. In the end, when each NP in the sentence has been processed and classified, each concept category is displayed on the screen, along with the NPs which were classified in that category.

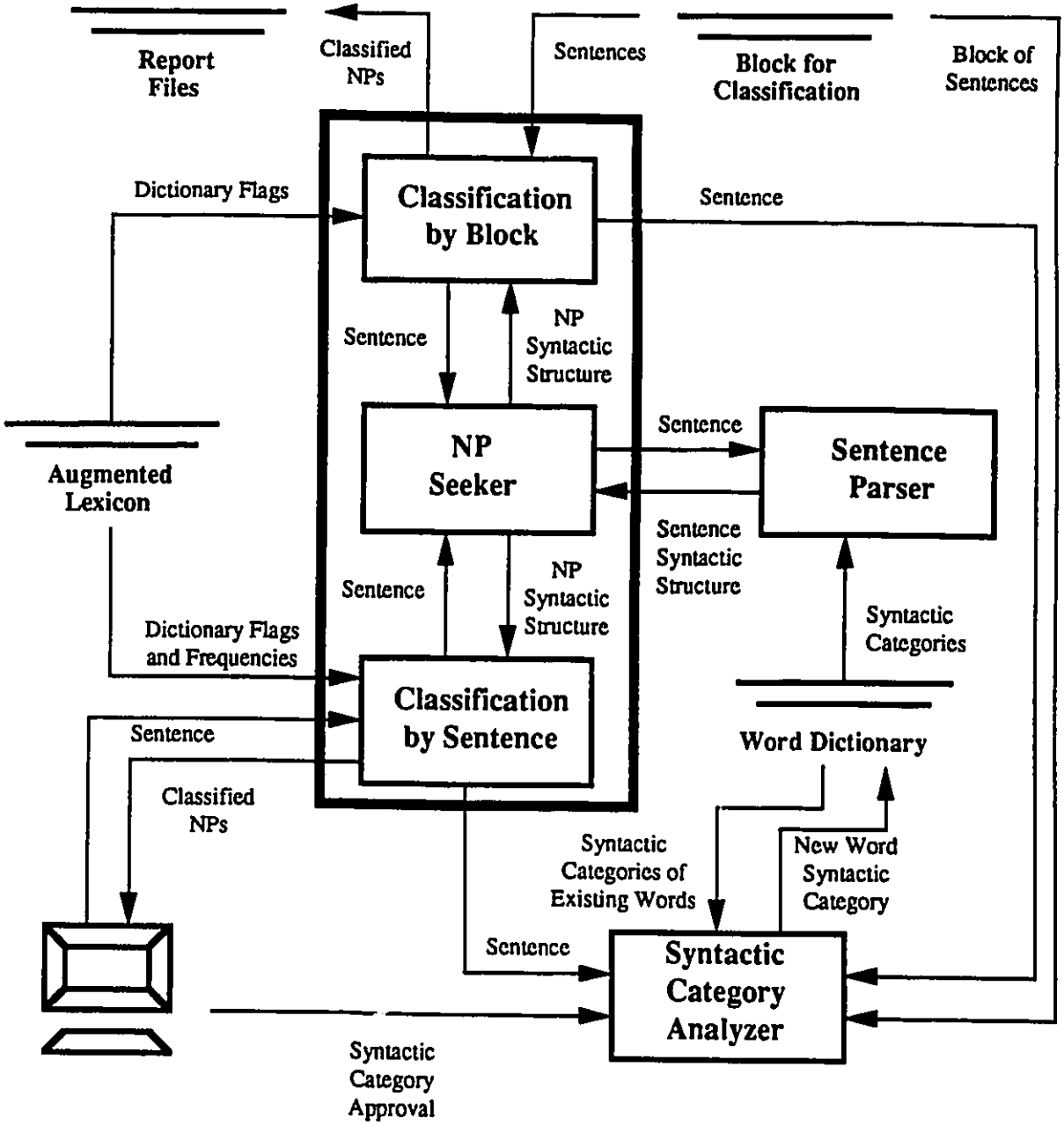


Figure 2 - Classification Module

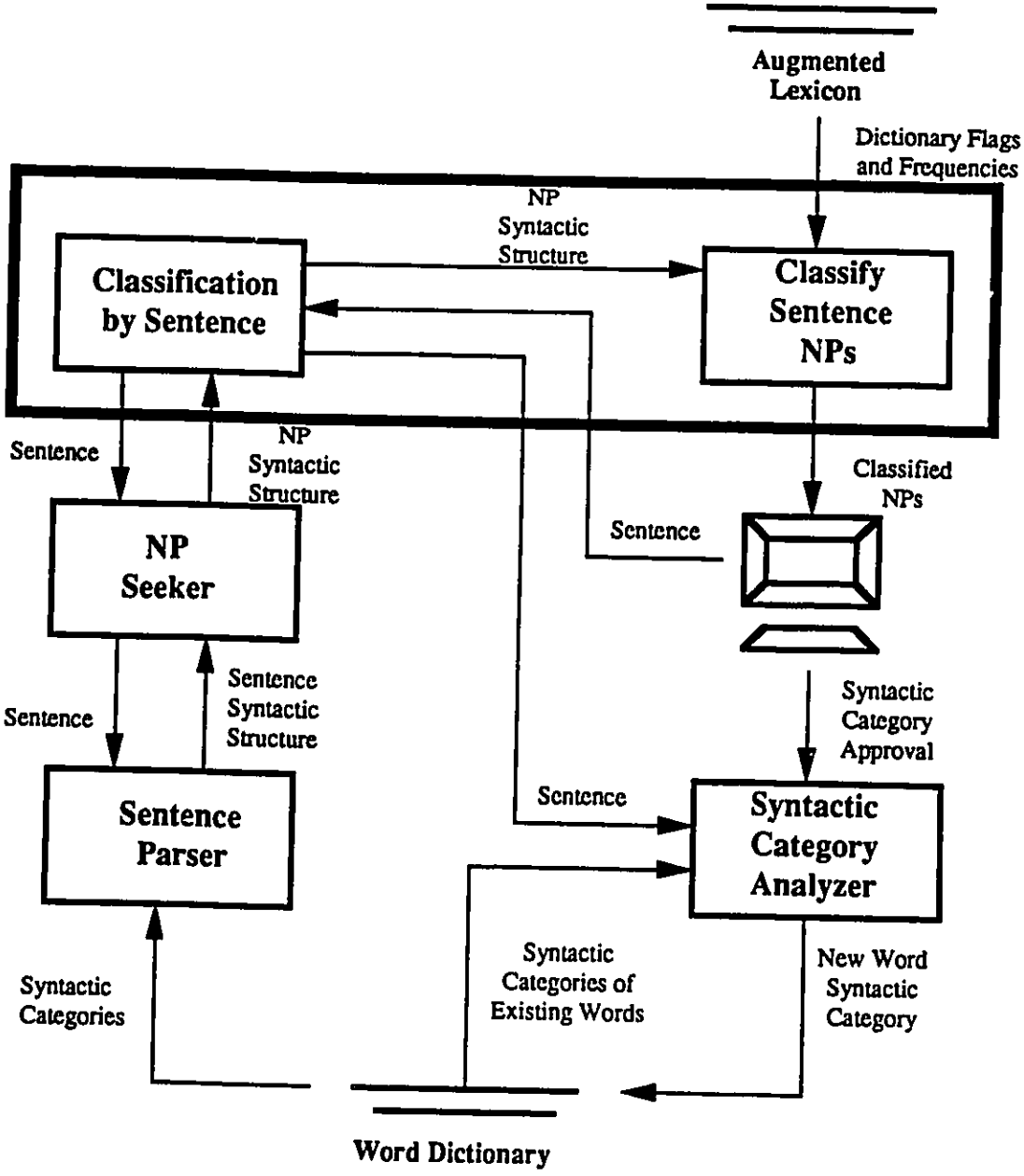


Figure 3 - Classification by Sentence Module

2.1.1 Extracting the NPs

In both types of classification, the Sentence Parser is used to parse the sentence and to obtain the syntactic structure associated with it. However, before the Sentence Parser is invoked, the SCAN module ensures that each word in the sentence appears with its syntactic category in the Sentence Parser's Word Dictionary. Once the Sentence Parser has returned the syntactic structure to the *NP Seeker* module, this module extracts the NP structures found in the sentence structure, which are then passed on to the *Classify Sentence NPs* module for classification.

The nature of the NPs retained for concept classification will be clarified in section 2.3, where the NP Seeker interface between the Classification Module and the Sentence Parser will be described. In general, only the NPs with a head noun and zero or more modifiers are considered. The head noun can be a noun (e.g. "*house*") or the present participle form of a verb (e.g. "the *writing* on the wall"), and the modifiers can be noun-noun modifiers (e.g. "a *brick* house"), adjectives (e.g. "a *big* house"), the present (e.g. "a *growing* household") or past (e.g. "a *flooded* basement") participle form of a verb, or any grammatically valid combination thereof (e.g. "a *large flooded* basement").

2.1.2 Classifying the NPs

The *Classify Sentence NPs* module (see Figure 3) processes and classifies each of the NPs returned by the NP Seeker, one at a time. It retrieves from the Augmented Lexicon the dictionary flag and the frequency of the NP's head noun and of each of its modifiers. In both types of classification, it is always assumed that the head noun carries more weight than its modifiers when a classification category is being determined. This is because the head noun is used to describe a concept, and its modifiers serve to specialize that concept.

2.1.2.1 *Determining the Dictionary Flags*

In Classification by Sentence, only one dictionary flag is used for all the modifiers. Therefore, if the head noun has more than one modifier, each with its own dictionary flag, the flag used for the group of modifiers is the one associated with the majority of the modifiers in the group, with the "n" flag (indicating that none of the modifiers appears in Longman's list) chosen in case of a tie. Assuming that there are seldom more than three modifiers associated with a head noun, this strategy becomes more intuitive. If there is only one modifier, then its dictionary flag is used. If there are two modifiers, and they both have the same dictionary flag, it is used, otherwise the "n" dictionary flag is used. This is in keeping with our assumption that a concept is better slightly over-classified than under-classified.

2.1.2.2 *Determining the Frequencies*

Before a NP is classified, its Frequency Area (FA) will be determined, based on the frequencies of the words in the NP. The FA is simply an indicator of how frequently the NP's constituents (its head noun and each of its modifiers) occur in the text, with respect to the constituents of other NPs. This comparison between the constituents of different NPs is based on the number of words in the text for which the constituents of a NP account. For example, a NP made up of two modifiers and a head noun (e.g. the *lazy brown fox*), where each modifier occurs twice and where the head noun occurs three times in the entire text, will account for 2 + 2 + 3 = 7 occurrences in the text. Similarly, a NP with no modifiers, whose head noun occurs five times in the text, will account for five words in the text. This scheme has proven to be an effective way of comparing the relative frequencies of NPs in the text, based on the individual word frequencies found in the Augmented Lexicon.

2.1.2.3 *Computing the Frequency Area*

Frequency Areas are used to determine how the NPs rank with respect to each other in terms of word frequencies. Since the FA of a NP is used to classify it, it is computed before the NP's concept classification.

In order to compare the NPs with each other and assign them to a FA, three bounds are used to segregate the NPs into four FAs. Because FAs and concept classifications can be computed for any length of text, the first two bounds are percentages of the total number of words in the text, and the third is a constant designed to segregate NPs whose constituents account for only one word in the text. In Classification by Sentence, MinBound1 (the highest bound) is set to 0.5% of the number of words in the text, and MinBound2 (the middle bound) is set to 0.25%. MinBound3 (the lowest bound) is set to a constant just above 1 (i.e. 1.1), so that NPs which account for only one word in the text can be placed in a separate FA and a distinct concept category. These Minimum Bound percentages were determined through trial and error tests which will be described in section 2.4. The percentages yielding the best NP classifications were retained for the final algorithm.

An important observation is that the precise boundaries between FAs, and therefore the Minimum Bounds, are merely labels inserted at various points throughout a linear set of concepts. The percentages used for the Minimum Bounds were obtained by examining the classification results they yielded and then deciding on the improvements which could be made. This process was entirely subjective, since I was deciding to which category each concept should belong. However, the general nature of the final algorithm and its adequate performance with different technical texts as input will demonstrate how well such a subjective approach worked.

The graph in Figure 4 illustrates how the frequency area of a NP is computed. For Classification by Sentence, *Term Percentage* is not used and is set to 100%. Its purpose will be explained in detail in the section on Classification by Block. The frequency area of a NP is

determined solely by how the frequency of its words ranks with respect to the three bounds. For example, a NP is considered to be relatively frequent (FA1) if its constituents account for more than 0.5% of the words in the text, and it is considered to be relatively infrequent (FA3) if its constituents account for less than 0.25% of the words in the text.

2.1.2.4 *Determining the Classification Category*

Based on the frequency area of a NP and the dictionary flags of its head noun and that of its modifiers, a classification of technicality is obtained, as illustrated in Figure 5. In this table, special consideration is given to NPs with no modifiers. Such NPs are represented in Figure 5 with "0" as the dictionary flag of their modifiers. As an example, a NP whose head noun does not appear in Longman's list and whose FA is FA1 will be ranked as a **Technical** concept, regardless of the dictionary flag of its modifiers.

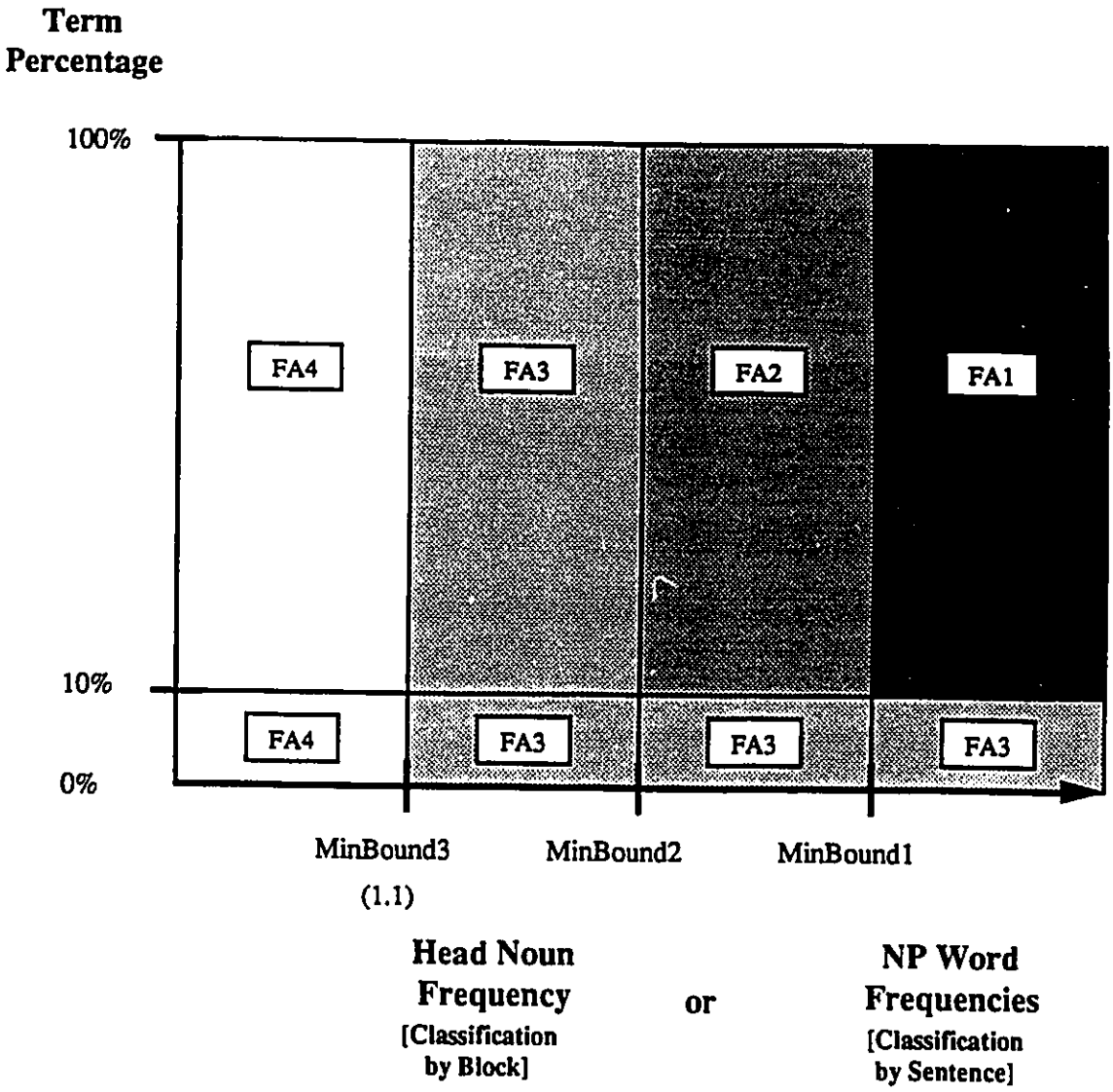


Figure 4 - Frequency Area Graph

Head Noun Flag	Modifier Flag \ Frequency Area	FA1	FA2	FA3	FA4
"n"	"n"	TECH	TECH	RATHER	UNKNOWN
	0	TECH	RATHER	POSS	UNKNOWN
	"y"	TECH	RATHER	POSS	UNKNOWN
"y"	"n"	TECH	RATHER	POSS	UNKNOWN
	0	TECH	POSS	NOT TECH	UNKNOWN
	"y"	RATHER	POSS	NOT TECH	UNKNOWN

Legend:

TECH:	Technical
RATHER:	Rather Technical
POSS:	Possibly Technical
NOT TECH:	Not Technical
UNKNOWN:	Technicality Unknown

Figure 5 - NP Classification Table

2.1.3 An Example

Suppose we had the sentence: "The *key* to the *simplicity* of *QUIZ* lies in its *relationship* with the *data dictionary*." This sentence can be found in Chapter 1 of the PowerHouse QUIZ manual, describing the 4th generation language QUIZ.[1]

Assuming that the technical text consists of the first QUIZ chapter, the following information will be available from the Augmented Lexicon once the TBB has processed the chapter. The word *key* occurs 20 times in Chapter 1 and has a "y" dictionary flag; *simplicity* occurs once and has a "n" dictionary flag; *QUIZ* occurs 46 times and has a "n" flag; *relationship* occurs once and has a "n" flag; *data* occurs 10 times and has a "n" flag; and *dictionary* occurs 6 times and has a "y" flag. Since the total number of words in Chapter 1 is 1830, MinBound1 is set to 0.5% of 1830, which is 9.2, MinBound2 is set to 0.25% of 1830, which is 4.6, and MinBound3 is set to the constant 1.1.

Given those word frequencies, and the fact that Term Percentage is not used in Classification by Sentence and therefore set to 100%, the five NPs in the sentence will be assigned to the following FAs, according to Figure 4. *Key* has a frequency of 20 > MinBound1 and will be assigned to FA1; *simplicity* has a frequency of 1 < MinBound3 and will be assigned to FA4; *QUIZ* has a frequency of 46 > MinBound1 and will be assigned to FA1; *relationship* has a frequency of 1 < MinBound3 and will be assigned to FA4; and *data dictionary* has a frequency of 10 + 6 = 16 > MinBound1 and will be assigned to FA1.

Then, according to the Classification table of Figure 5, *key*, with a "y" dictionary flag, no modifiers and a frequency area of FA1, will be classified as **Technical**; *simplicity*, with a frequency area of FA4, will be classified as technicality **Unknown**; *QUIZ*, with a "n" dictionary flag, no modifiers and a frequency area of FA1, will be classified as **Technical**; *relationship*, with a frequency area of FA4,

---

[1] Cognos Incorporated, PowerHouse QUIZ (1985).

will be classified as technicality **Unknown**; and *data dictionary*, with a "y" head noun dictionary flag, a "n" modifier dictionary flag and a frequency area of FA1, will be classified as **Technical**. These results are summarized below.

NP	Mod. Freq.	Head Noun Freq.	Total Freq.	FA	Head Noun Flag	Mod. Flag	Class'n
key	0	20	20	FA1	y	0	TECH
simplicity	0	1	1	FA4	n	0	UNKNOWN
QUIZ	0	46	46	FA1	n	0	TECH
relationship	0	1	1	FA4	n	0	UNKNOWN
data dictionary	10	6	16	FA1	y	n	TECH

with MinBound1 = 9.2, MinBound2 = 4.6, and MinBound3 = 1.1

Table 1 - Example of Classification by Sentence

2.1.4 The Classification by Sentence Algorithm

The following is the Classification by Sentence algorithm described in this section.

- Read Sentence from the keyboard
- Invoke SCAN with Sentence as argument
- Invoke NP Seeker with Sentence as argument and retrieve List of NP Syntactic Structures
- For every NP in the List of NP Syntactic Structures:
  - Extract the Head Noun and the Modifiers from the current NP
  - Retrieve the Head Noun and the Modifiers' Frequencies and Dictionary Flags from the Augmented Lexicon

- Compute the Modifier Total Frequency and consolidated Dictionary Flag
- Compute MinBound1 and MinBound2 using the total word count found in the Augmented Lexicon
- Compute the FA of the NP, using the Head Noun Frequency + Modifier Total Frequency
- Compute the NP Classification, using FA, Head Noun Dictionary Flag and Modifier consolidated Dictionary Flag; keep NP Classification in memory

- Display Classification results from memory on the screen

2.2 NP Classification by Block

The main differences between Classification by Sentence and Classification by Block are the type of input and the frequencies used for NP Classification. Whereas Classification by Sentence requires the user to provide one of the sentences of the technical text, Classification by Block processes the entire text found in the Block for Classification. Furthermore, because it is known that the entire text will need to be parsed in order for its NPs to be classified, it is possible to accumulate the number of times that each NP occurs in the text. This frequency can then be used to compute the Frequency Area of the NP, instead of using the word frequencies of each of its constituents. For example, the FA of the NP *conjoined sentence* will be based on the number of times that *conjoined sentence* occurs as a NP and the number of times that its head noun *sentence* occurs as a head noun in the text, rather than the number of times that each word occurs as any part of speech. Because of this, Classification by Block yields more linguistically accurate results than Classification by Sentence.

The *NP Count Block* module (see Figure 6) takes each sentence in the Block for Classification, calls SCAN to ensure that there are no unknown words in the sentence, invokes the NP Seeker to obtain the

NPs of the sentence, and stores each NP in a NP database, along with a current frequency counter which is updated each time the same NP is encountered in the text. Once all the sentences in the Block for Classification have been processed, the *Classify NP Block* module uses the NPs and frequencies in the NP database and the dictionary flags found in the Augmented Lexicon to classify each of the NPs into a concept category. A report file is then created with the results of Classification by Block.

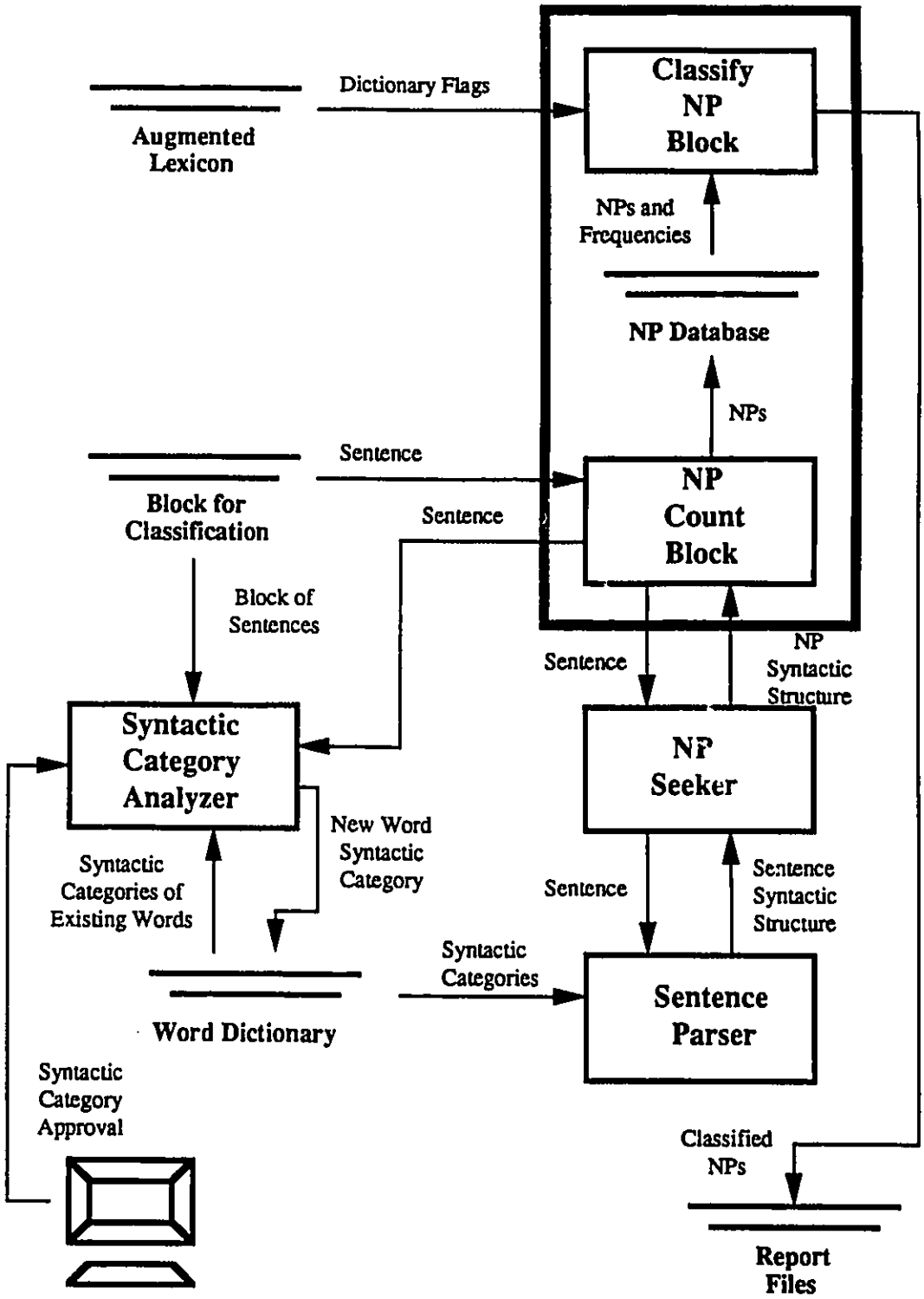


Figure 6 - Classification by Block Module

2.2.1 Extracting and Accumulating the NPs

The NP Count Block module processes each of the sentences in the Block for Classification by invoking the NP Seeker, which has the sentence parsed by the Sentence Parser and then extracts its NPs from the resulting sentence structure, just as in Classification by Sentence. The similarities, however, end there. Whereas at this point, Classification by Sentence invokes its Classification algorithm, the NP Count Block module simply stores the NPs in a temporary NP database in memory, along with each NP's accumulated number of occurrences in the text. Also, whenever the number of occurrences of a NP is incremented in the NP database, so is the number of occurrences of its head noun.

These decisions were based on observations which led to some common-sense discoveries. In the classification results, when the count of the head noun of a NP was not incremented along with the NP's frequency, the head noun itself would end up under-classified. This can be better illustrated with an example. If one of the technical concepts of a text is *computer*, then a reference to *new computer* or *mainframe computer* is still a reference to the head noun concept of *computer*. Similarly, *programming languages* and *natural languages* are both *languages*, despite their representing two very different concepts. Therefore, regardless of the number or the nature of the modifiers, the concept to which the head noun refers remains the same, and its relative frequency should reflect that fact. The only exceptions to this rule are polysemous head nouns, which are used to convey more than one distinct concept (e.g. *National Bank* and *river bank*). In such cases, the multiple meanings of the head noun are by definition so un-related that they would not both be used as technical concepts in the same domain. As a result, no serious mis-classifications will occur if the less technical concept is mistaken for the more technical one.

Furthermore, in order to simplify the classification process, I decided that only a portion of each NP, what I call a *term*, would be

stored in the NP database for subsequent processing. For AZTEC, a term consists of a NP's head noun, preceded by its last modifier (if there is one). The decision to ignore all but the last modifier was made in order to avoid the classification of concepts that are too specific to be of use to the user. The NP frequency of a concept with two or more modifiers will be less than or equal to the frequency of its head noun and last modifier (i.e. its term), and it may be too low for AZTEC to recognize its technicality and to classify it properly. Therefore, rather than using a consolidated dictionary flag for all of the modifiers, which would mean using the more specialized concept described in the NP, I decided to only consider the last modifier, since I had observed that in many cases it is the one which conveys the most specific information about the head noun concept. The reasoning for this is in keeping with the rationale for incrementing the head noun frequency for every NP that is accumulated. Whatever concept is described in a head noun with any number of modifiers is often also described, but to a lesser extent, with the head noun alone, and with the head noun and its last modifier.

For example, a *QUIZ data file* is basically a *file*, more specifically a *data file*, and even more specifically a *QUIZ data file*. Although some specialized information is lost when the *QUIZ* modifier is dropped, the essential nature of the concept *data file* is retained for concept classification. Furthermore, if a NP such as *DBase data file* was also encountered in the text, even though the two concepts refer to completely different types of files, both describe *data files*, and if this concept appears frequently enough in the text, it should be classified in a more technical category. However, if *QUIZ data file* and *DBase data file* were classified as separate concepts, each with its own frequency, the *data file* concept would end up under-classified, because its frequency would be decreased in direct proportion with the number of occurrences of *QUIZ data file* and *DBase data file*.

There are, however, cases in which this approach may seem questionable. For example, in some domains, many technical terms

are made up of a head noun and two modifiers (e.g. *enteric coated aspirin*). In this case, classifying the term *coated aspirin* would work just as well, since regardless of how many times the NP *enteric coated aspirin* appears in the text, the term *coated aspirin* will be extracted from it at least as many times. The worst that can happen is that *coated aspirin* ends up slightly over-classified if it also appears in the text as the term of another NP.

Another possible source of mis-classifications resulting from the reduction of a NP to the head noun and its last modifier is the case where the last modifier is not the one providing the most specialized information about the head noun in a given context. For example, in a text on animal ailments, the NP *rabid black dog* would be better reduced to *rabid dog* rather than *black dog*, but in another domain, *black dog* may be the more appropriate term. Unfortunately, without any domain knowledge, AZTEC is unable to choose the "best" modifier over the others. However, since a classified list of terms is of greater interest to the user than a simple list of head nouns, and having observed that the last modifier is often the one which most specializes the head noun, I decided that for the purposes of concept classification in terms of technicality, the use of a head noun and its last modifier would be sufficient to obtain a valid approximation of concept classifications.

Only when every sentence has been parsed and each NP in the text accumulated and stored in the NP database is the *Classify NP Block* module invoked.

2.2.2 Classifying the NPs

Since Classification by Block only considers terms with either one modifier or none, there is no question of which dictionary flags to use for the modifier and the head noun. However, as will be explained below, two frequencies are used to determine the frequency area of a term: the term's accumulated frequency found in the NP database, and the head noun's frequency.

2.2.2.1 *Using the Head Noun Frequency*

In keeping with the Classification by Sentence approach, the first Classification by Block algorithms only used term frequency to compute the frequency area of a NP. Unfortunately, this led to some inappropriate classifications in the middle categories of **Rather Technical** and **Possibly Technical.**

For example, in the earlier text describing Delisle's Sentence Parser (1990), the terms *conjoined sentence* and *case analyzer* occur 6 and 5 times respectively in the text, and should both be classified as **Technical** or **Rather Technical.** However, the terms *empty clause* and *time constraint*, which would be better classified as **Possibly Technical**, occur 3 and 4 times respectively in the text, although when examining the head noun frequencies of these four terms, *sentence* was found to occur 42 times as a head noun in the text, *analyzer* 15 times, *clause* 9 times and *constraint* 10 times.

With an algorithm using only the accumulated term frequency to compute the Frequency Area of these four concepts, it is more than likely that they would be assigned to the same FA and possibly the same concept category, despite the differences in their head noun frequencies and their optimal classifications. In an attempt to solve this problem, AZTEC was modified to consider the frequency of occurrence of the head noun, rather than the term frequency, to calculate the FA of each concept.

However, when practical experiments were carried out on Classification by Block algorithms which *only* used the head noun frequency to determine the FA of a term, one problem in particular surfaced, requiring an additional measurement to be considered in the final algorithm.

2.2.2.2 *Using Term Percentages*

An interesting problem occurred when the Classification by Block algorithm came to process a term which was relatively not very technical, but whose head noun was highly technical. Because no such term can ever be known, without a doubt, not to be

technical, I found it preferable to have it classified in the **Possibly Technical** category, rather than in the **Not Technical** category. However, when head noun frequencies were the sole factor in approximating the technicality of a NP, a less technical term with a highly technical head noun would end up over-classified. Conversely, when only the term frequency of a NP was used, it would be under-classified. For example, the head noun *computer*, in a computer-related text, is by itself a highly technical term, and so is the term *mainframe computer*. However, the term *new computer* is not very technical at all, even though *computer* is, and the problem occurred when less technical terms with technical head nouns were classified on the same scale as a truly technical term, such as *mainframe computer*, or when it would be under-classified as **Not Technical**.

This problem was solved by incorporating into the Classification by Block algorithm a scheme to further discriminate between two such terms, so that the less technical term may end up in a lower FA (but not too low) even though it has the same head noun as the technical term. This scheme consisted in using the head noun frequency instead of the term frequency, and introducing *Term Percentages* in the FA computation algorithm.

A Term Percentage is the ratio of a term's frequency with respect to the frequency of its head noun. This ratio is compared with one boundary, 10%, which was determined through trial and error tests. The three other boundaries used in Classification by Block to determine the Frequency Area of a NP, namely MinBound1, MinBound2 and MinBound3, are set to different values than in Classification by Sentence. In Classification by Block, MinBound1 is set to 1.5% of the total number of NPs in the NP database, MinBound2 is set to 0.75%, and MinBound3 is again set to the constant 1.1 so that only the terms whose head noun occurs once in the entire block of text fall below this lower bound. This last bound is necessary, because observation of practical experiments showed that there is a significant number of NPs in a text whose head nouns occur only once, and these must be considered apart from the rest of the NPs,

since it is impossible without extra measures to approximate the classification of such NPs.

Another concern of the Classification by Block algorithm was that the introduction of Term Percentages should not undermine the results previously obtained, which were based on head noun frequencies. As a result, I decided to retain the head noun frequency as the basis of comparison against the three Minimum Bounds on the X-axis of Figure 4, and to use the Term Percentage to reflect the individual term's frequency with respect to the head noun along the Y-axis. Therefore, in the final algorithm, whenever a frequency is compared against any of the three Minimum Bounds, the head noun frequency is used, and the frequency of a term is only used for the computation of the Term Percentage.

2.2.2.3 *Computing the Frequency Area and the Classification Category*

Just as in Classification by Sentence, once the FA of a term has been determined according to Figure 4, using its head noun frequency on the X-axis and its term percentage on the Y-axis, its NP classification can be obtained according to Figure 5, using the dictionary flags of the head noun and the last modifier, along with the Frequency Area of the term.

2.2.3 An Example

In order to illustrate the process of FA computation and NP Classification, let us use an example. Suppose that we had Delisle's 1990 parser text, containing 2030 NPs in total. MinBound1 would be set to 1.5% of 2030, namely 30.4, MinBound2 would be set to 0.75% of 2030, which is 15.2, and MinBound3 would be constant at 1.1. The word *sentence* occurs a total of 42 times as a head noun in the text, and among those 42 occurrences, *conjoined sentence* occurs six times, *parsable sentence* occurs once, *long sentence* occurs once, and *complete sentence* occurs three times.

The role of Term Percentage, in this as in any case, is to widen the gap between terms of different technicality, but with similar frequencies. In this instance, the Term Percentages associated with the five NPs are the following.

sentence:	(42 × 100) / 42 = 100%
conjoined sentence:	(6 × 100) / 42 = 14%
parsable sentence:	(1 × 100) / 42 = 2%
long sentence:	(1 × 100) / 42 = 2%
complete sentence:	(3 × 100) / 42 = 7%

Since the same head noun frequency (42) is used with Figure 4 along the X-axis, the Term Percentage reflecting the individual term frequencies will separate the five terms into different Frequency Areas. According to Figure 4, the Frequency Areas of all five terms will be computed to the right-hand side of MinBound1, since 42 > 30.4. Because the Term Percentages of *sentence* and *conjoined sentence* are greater than 10%, both terms will be assigned a Frequency Area of FA1. *Parsable sentence* and *long sentence* will be assigned a Frequency Area of FA3 (since 2% < 10%), as will *complete sentence* (since 7% < 10%).

As a result, since *sentence* has a "y" dictionary flag, as do *long* and *complete*, and since *conjoined* and *parsable* have "n" dictionary flags, the five terms will be classified as follows: *sentence* and *conjoined sentence* will be ranked as **Technical**, *parsable sentence* will be ranked as **Possibly Technical**, and *long sentence* and *complete sentence* will be ranked as **Not Technical**. These results are summarized in the table below.

NP	Term Freq.	Head Noun Freq.	Term Perc.	FA	Head Noun Flag	Mod. Flag	Class'n
sentence	42	42	100	FA1	y	0	TECH
conjoined sentence	6	42	14	FA1	y	n	TECH
parsable sentence	1	42	2	FA3	y	n	POSS
long sentence	1	42	2	FA3	y	y	NOT TECH
complete sentence	3	42	7	FA3	y	y	NOT TECH

with MinBound1 = 30.4, MinBound2 = 15.2 and MinBound3 = 1.1

Table 2 - Example of Classification by Block

2.2.4 The Classification by Block Algorithm

The following is the Classification by Block algorithm described in this section.

- Read Sentence from Block for Classification
- For every Sentence read:
  - Invoke SCAN with Sentence as argument
  - Invoke NP Seeker with Sentence as argument and retrieve List of NP Syntactic Structures
  - For every NP in the List of NP Syntactic Structures:
    - Extract the Term from the current NP
    - Store in NP Database and increment Term Frequency by 1
    - Increment Head Noun Frequency by 1 in the NP Database
  - Read next Sentence from the Block for Classification
- Compute MinBound1 and MinBound2, using the total

number of NPs found in the NP Database

- For every NP in the NP Database:
  - Extract Term Frequency from NP Database
  - Extract Head Noun Frequency from NP Database
  - Get Head Noun Dictionary Flag and Modifier Dictionary Flag from Augmented Lexicon
  - Compute Term Percentage, using Term Frequency and Head Noun Frequency
  - Compute the FA of the NP, using Head Noun Frequency and Term Percentage
  - Compute NP Classification, using FA and Head Noun and Modifier Dictionary Flags; keep NP Classification in memory
- Print Classification results from memory to a report file

2.2.5 Clustering

When it became apparent that there were limits to manipulating frequencies of occurrence to obtain better classification results, I investigated the possibility of using a clustering algorithm to obtain noun clusters which could be of use in improving the classification results. The clustering algorithm I implemented is based on the one put forth by Hirschman, Grishman and Sager[1] as part of the Linguistic String Project at New York University. The authors describe the clusters they obtain in the following way:

> Clusters are made up by grouping together "similar" lexical items. Two lexical items are similar if either the two words appear in a certain argument position under the same operator, e.g. both as subject of a certain verb; or both words operate on the same operand in a certain argument position, e.g. both have an occurrence with the same word as object.[2]

---

[1]Lynette Hirschman, Ralph Grishman, and Naomi Sager, "Grammatically-Based Automatic Word Class Formation," Information Processing & Management 11 (1975): 39-57.

[2]Ibid., 40-41.

By using this algorithm and by augmenting the number of syntactic relations between words that indicate them to be "similar," I hoped to obtain clusters of **Technical** concepts (or concepts which should be classified as **Technical**), clusters of **Rather Technical** concepts, and so on. The intent was that these clusters would serve to verify AZTEC's concept classifications and would be used to correct mis-classifications by computing the average concept category of each cluster and updating the category of the non-conforming terms in the cluster to the average.

Because the algorithm requires a database of all the syntactic relations found in the technical text (e.g. *subject-verb*, *verb-subject*, *verb-object*, *object-verb*, to name a few) and therefore requires that the entire text be parsed, the clustering algorithm is only used for Classification by Block and not for Classification by Sentence. Also, in order for the clustering algorithm to extract these syntactic relations all at once, the NP Seeker was slightly modified so that each time it is invoked by the Classification by Block module during the NP accumulation process, it saves in a file the syntactic structure of each sentence, as it is returned by the Sentence Parser.

Once the entire Block for Classification has been processed by the classification algorithm and the NPs classified, the clustering algorithm can be executed, and its resulting clusters are saved in a file to be used in conjunction with the classification results so that the latter can be improved (see Figure 7). The clustering algorithm used by AZTEC is described below. For a more thorough description, the reader is referred to Hirschman et al. (1975).

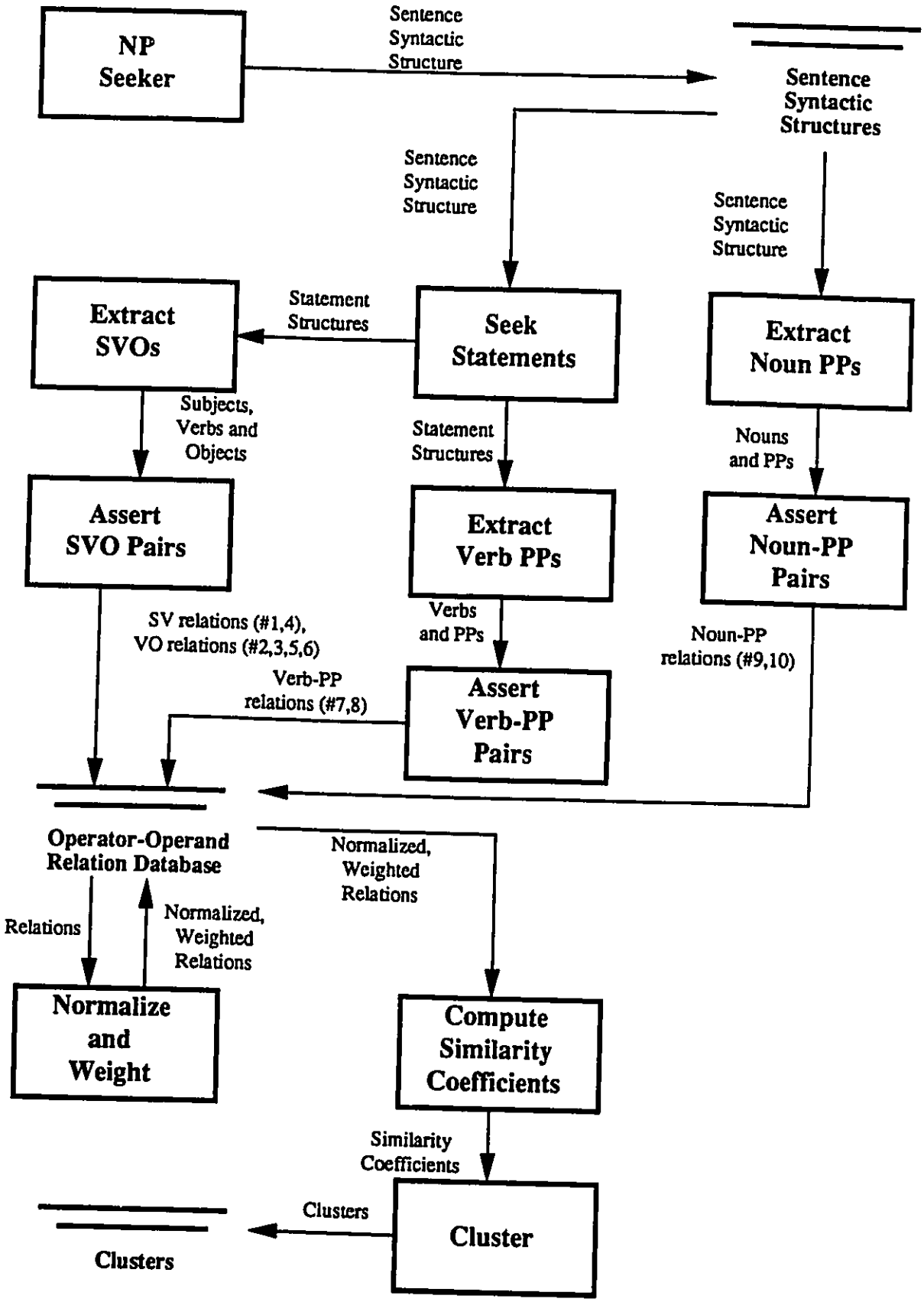


Figure 7 - Clustering Module

2.2.5.1 *Extracting the Syntactic Relations*

The first task of AZTEC's clustering algorithm consists in extracting from the syntactic structure of each sentence the information needed to establish the syntactic relations between pairs of words. Although Hirschman et al.'s algorithm used six syntactic relations, I added another four for a total of ten. Hirschman et al. use the following relations.

Given two words W_i and W_j, and given that operators consist of verbs and that operands represent the operator's arguments, namely the subject and the objects of the sentence,

- relation #1 represents the case where W_j is an operator (e.g. a verb) and W_i is its first argument (the subject)
- relation #2 is the case where W_j is an operator and W_i is its second argument (the direct object)
- relation #3 is the case where W_j is an operator and W_i is its third argument (the indirect object)
- relation #4 is the case where W_i is an operator and W_j is its first argument
- relation #5 is the case where W_i is an operator and W_j is its second argument
- relation #6 is the case where W_i is an operator and W_j is its third argument

In reality, there are only three distinct syntactic relations among these six, but their converse are also needed for the formation of word clusters.

The two syntactic relations I added link together the verb of a sentence with a Prepositional Phrase (PP) modifying it (e.g. "He *called from home*"; "he *called from* the *office*"), and the head noun of a NP with a PP modifying it (e.g. "the *intent of* a *speech*", "the *intent of* a *gesture*"). The other two syntactic relations I added are simply the converse of the above relations, linking a PP with the verb it modifies, and a PP with the head noun it modifies. The addition of syntactic relations of this type was briefly mentioned in Hindle

(1990),[1] which also makes use of Hirschman et al.'s algorithm: "it will be useful to extend the analysis presented here to other kinds of relationships, including more complex kinds of verb complementation, noun complementation, and modification both preceding and following the head noun."[2]

As a result, in my adaptation of Hirschman et al.'s clustering algorithm, relations #1 through #6 are the same as above, and the following were added:

- relation #7 is the case where W_i is an operator (a verb) and W_j is the head noun of a PP modifying it
- relation #8 is the case where W_j is a verb and W_i is the head noun of a PP modifying it
- relation #9 is the case where W_i is a noun and W_j is the head noun of a PP modifying it
- relation #10 is the case where W_j is a noun and W_i is the head noun of a PP modifying it

In relations #7 through #10, the preposition linking the verb or the noun to the head noun of the PP modifying it is also included in the syntactic relation.

Once a relation and its converse has been established between two words, they are added to a temporary Operator-Operand Relation database in memory. For example, if the current sentence is *the girl from Toronto drove her friend to work*, the number of occurrences of the following relations will be incremented by 1:

relation #1 between *girl* and *drove*
relation #2 between *friend* and *drove*
relation #4 between *drove* and *girl*
relation #5 between *drove* and *friend*
relation #7 between *drove*, *work* and *to*
relation #8 between *work*, *drove* and *to*

---

[1]Donald Hindle, "Noun Classification from Predicate-Argument Structures," Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (1990): 268-275.
[2]Ibid., 275.

relation #9 between *girl*, *Toronto* and *from*
relation #10 between *Toronto*, *girl* and *from*

2.2.5.2 *Normalizing and Weighting Word Vectors*

Each word W_i involved in at least one syntactic relation has a vector V_i associated with it. Each V_i will contain $10n$ components, where n is the total number of words in the text involved in syntactic relations, excluding prepositions. Each k-th group of 10 components in V_i contains the number of occurrences in the text of each syntactic relation between W_i and W_k (one of the other words in the text), so a non-zero entry in the m-th position of the k-th group would represent the number of occurrences of relation #m between W_i and W_k. As a simple example involving only one sentence, *the girl from Toronto drove her friend to work*, and assuming that W_1 = *girl*, W_2 = *Toronto*, W_3 = *drove*, W_4 = *friend*, and W_5 = *work*, W_1 will have the following vector associated with it:

```
[ 0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,1,0,
  1,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0  ]
```

where the first non-zero component represents the relation between W_1 and W_2, and thus the second group of 10 components, and the type of the relation (#9), and so the ninth component of the second group. The number in the component (1) represents the number of occurrences in the text of relation #9 between W_1 and W_2. The second non-zero component represents one occurrence of a relation #1 between W_1 and W_3, hence the first component of the third group in V_1 is a 1.

V_3 would be the following:

```
[ 0,0,0 1,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,1,0,0,0,0,0,
  0,0,0,0,0,1,0,0,0  ]
```

where the first non-zero component represents one occurrence of a syntactic relation #4 between W_3 and W_1, the second represents one occurrence of a relation #5 between W_3 and W_4, and the third represents one occurrence of a relation #7 between W_3 and W_5.

In order to save memory, AZTEC only keeps the non-zero vector components in the Operator-Operand Relation database, along with the following information for each component: the number of the first word in the relation (i for W_i), the number of the second word (j for W_j), the preposition linking them if applicable (used for relations #7 through #10 and *nil* otherwise), the type of the relation between W_i and W_j (one of relations #1 through #10), and the number of occurrences in the text of this relation between W_i and W_j.

Before computing a Similarity Coefficient (SC) between two words, which will be used as the basis for the creation of clusters, Hirschman et al.'s algorithm normalizes and weights the number of occurrences of each relation between every pair of words.

> Each vector is divided by a normalization factor to produce a vector of unit length. (The normalization factor for a vector is the square root of the sum of the squares of its components.) . . . The normalized vector for each word Wi is multiplied by a weighting factor which gives less weight to infrequently occurring words: Weighting factor for word Wi = $1 - (0.99 / \sqrt{n})$ where n = the number of occurrences of Wi in operator-operand pairs.[1]

2.2.5.3 *Computing the Similarity Coefficient*

The Similarity Ccefficient (SC) between two words (e.g. two subjects) indicates their relative frequency of occurrence in the same syntactic relations (e.g. subject-verb) with the same words (e.g. the same verb). It is computed as follows: "the similarity coefficient between two words W_i and W_j is the inner product of the normalized, weighted characteristic vectors of the two words:"[2]

$$SC_{ij} = V_i * V_j = \sum_{k=1}^{10n} (V_i)_k \times (V_j)_k$$

---

[1]Hirschman, Grishman and Sager, 44-45.
[2]Ibid., 44.

Just as AZTEC only stores the non-zero components of each vector, only the non-zero inner products between two vector components are considered. AZTEC takes every pair of vectors, V_i and V_j, and for each relation between W_i and any other word W_k, the Operator-Operand Relation database is searched for a similar relation between W_j and W_k. For example, let us assume that a relation #5 exists between W_3 = *drove* and W_4 = *friend*, and that the same relation exists between W_3 and a different word in another sentence, say W_{10} = *kids*. When it comes to computing the SC between W_4 and W_{10}, AZTEC would realize that both words are involved in relation #5 with W_3. The normalized and weighted number of occurrences of W_4 with W_3 in relation #5 would be multiplied by the number of occurrences of W_{10} with W_3 in relation #5, and this product would be added to the products of the number of occurrences of other common relations between W_4 and W_{10} with the same words (if there are any). The resulting sum would be retained as the Similarity Coefficient of W_4 and W_{10}.

2.2.5.4 *Creating the Clusters*

Clusters are formed one word at a time. A new word is only added to an existing cluster if the average SC between the new word and each existing member of the cluster exceeds a certain threshold. The thresholds used by Hirschman et al. in their examples vary between 0.2 and 0.3. A word may be added to any number of clusters, but if no existing cluster can be found where the word's average SC with the other members exceeds the threshold, a new cluster is created with the new word as its sole member. As a result, many one-word clusters remain at the end of the clustering process, and since they can be of no use in improving the classification results, they are deleted.

2.2.5.5 *An Example*

Let us assume that we want to process a text made up of the following sentences: *The girl from Toronto drove her friend to work. The girl from Montreal had been delayed by a few hours. The man drove his kids to school.* The words of this text involved in syntactic relations are numbered as follows: W_1 = *girl*, W_2 = *Toronto*, W_3 = *drove*, W_4 = *friend*, W_5 = *work*, W_6 = *Montreal*, W_7 = *delayed*, W_8 = *hours*, W_9 = *man*, W_{10} = *kids* and W_{11} = *school*. Once all the syntactic relations have been extracted from the sentences, the following information will be available in the Operator-Operand Relation database:

[1, 3, nil, 1, 1]
[3, 1, nil, 4, 1]
[1, 2, from, 9, 1]
[2, 1, from, 10, 1]
[3, 4, nil, 5, 1]
[4, 3, nil, 2, 1]
[3, 5, to, 7, 1]
[5, 3, to, 8, 1]
[1, 6, from, 9, 1]
[6, 1, from, 10, 1]
[7, 1, nil, 5, 1]
[1, 7, nil, 2, 1]
[7, 8, by, 7, 1]
[8, 7, by, 8, 1]
[9, 3, nil, 1, 1,]
[3, 9, nil, 4, 1]
[3, 10, nil, 5, 1]
[10, 3, nil, 2, 1]
[3, 11, to, 7, 1]
[11, 3, to, 8, 1]

where the first argument indicates the word W_i, the second argument indicates W_j, the third argument specifies the preposition for relations #7 through #10 and *nil* otherwise, the fourth argument specifies the type of relation (#1 through #10) and the fifth argument is a counter of the number of occurrences of this relation between W_i and W_j in the text.

Next, the normalization factor (NF) and weighting factor (WF) for each word W_i must be calculated, and the number of occurrences in each component of V_i is normalized and weighted. For example,

since W_1 appears 4 times as W_i, $NF_1 = \sqrt{(4 \times (1 \times 1))} = 2$, $WF_1 = 1 - (0.99 / \sqrt{(4)}) = 0.5$, and $1 \times 0.5 / 2 = 0.25$. As a result:

[1, 3, nil, 1, 1] becomes [1, 3, nil, 1, 0.25]
[1, 2, from, 9, 1] becomes [1, 2, from, 9, 0.25]
[1, 6, from, 9, 1] becomes [1, 6, from, 9, 0.25]
[1, 7, nil, 2, 1] becomes [1, 7, nil, 2, 0.25]

The remaining Operator-Operand Relations are normalized and weighted in the same manner, and the results are the following:

[2, 1, from, 10, 1] becomes [2, 1, from, 10, 0.01]
[3, 1, nil, 4, 1] becomes [3, 1, nil, 4, 0.24]
[3, 4, nil, 5, 1] becomes [3, 4, nil, 5, 0.24]
[3, 5, to, 7, 1] becomes [3, 5, to, 7, 0.24]
[3, 9, nil, 4, 1] becomes [3, 9, nil, 4, 0.24]
[3, 10, nil, 5, 1] becomes [3, 10, nil, 5, 0.24]
[3, 11, to, 7, 1] becomes [3, 11, to, 7, 0.24]
[4, 3, nil, 2, 1] becomes [4, 3, nil, 2, 0.01]
[5, 3, to, 8, 1] becomes [5, 3, to, 8, 0.01]
[6, 1, from, 10, 1] becomes [6, 1, from, 10, 0.01]
[7, 1, nil, 5, 1] becomes [7, 1, nil, 5, 0.21]
[7, 8, by, 7, 1] becomes [7, 8, by, 7, 0.21]
[8, 7, by, 8, 1] becomes [8, 7, by, 8, 0.01]
[9, 3, nil, 1, 1] becomes [9, 3, nil, 1, 0.01]
[10, 3, nil, 2, 1] becomes [10, 3, nil, 2, 0.01]
[11, 3, to, 8, 1] becomes [11, 3, to, 8, 0.01]

The Similarity Coefficient (SC) between each pair of words W_i and W_j is calculated as follows. For each relation in which W_i is involved with another word W_k, it is determined whether or not the same relation exists between W_j and W_k. Using every common relation between W_i and W_j with other W_ks, the SC between W_i and W_j is calculated using the formula:

$$\Sigma\ V_i \times V_j$$

where the number of occurrences in the text of each relation between W_i and W_k and the number of occurrences of the same relation between W_j and W_k are multiplied together and added to the

sum of occurrences of other relations between W_i and W_j. For example, W_1 has a relation #1 with W_3 in common with W_9, so the SC for W_1 and W_9 is calculated as the inner product of $V_1 \times V_9$, i.e. 0.25 × 0.01 = 0.0025, since W_1 and W_9 have no other relation in common. Similarly, W_2 and W_6 both have a relation #10 using the preposition *from* with W_1, and their SC is 0.01 × 0.01 = 0.0001; W_4 and W_{10} have a relation #2 with W_3, and their SC is 0.01 × 0.01 = 0.0001; and W_5 and W_{11} have a relation #8 using the preposition *to* with W_3, and their SC is 0.01 × 0.01 = 0.0001. In summary:

SC (1, 9) = 0.0025
SC (2, 6) = 0.0001
SC (4, 10) = 0.0001
SC (5, 11) = 0.0001

Once the SCs have been computed, the formation of clusters is entirely dependent on the value of the threshold. In this example, since the SCs are very low, we want to set a low threshold, e.g. $t = 0$, if we want to have any clusters at all.

We begin with W_1. Since there are no existing clusters to which W_1 can be added, a new cluster (C_1) is created: C_1 = [*girl*]. Next, we see that W_2 cannot be added to C_1 because the SC between W_1 and W_2 is 0, which does not exceed the threshold, so a new cluster is created: C_2 = [*Toronto*]. Since W_3 has a 0 SC with both W_1 and W_2, it cannot be added to either C_1 or C_2, so it is placed in C_3 = [*drove*]. Similarly, W_4 and W_5 are added to C_4 = [*friend*] and C_5 = [*work*]. W_6, however, has a SC of 0.0001 with W_2, so it can be added to C_2 = [*Toronto, Montreal*]. Neither W_7 nor W_8 has a SC above the threshold with any other word, so each is placed in a single word cluster: C_6 = [*delayed*] and C_7 = [*hours*]. W_9 has a SC of 0.0025 with W_1, so it is added to C_1 = [*girl, man*]. W_{10} has a SC of 0.0001 with W_4 and is added to C_4 = [*friend, kids*], and W_{11} has a SC of 0.0001 with W_5 and is added to C_5 = [*work, school*]. Once the single-word clusters (C_3, C_6 and C_7) have been deleted, we are left with the following four clusters:

C1 = [*girl, man*]
C2 = [*Toronto, Montreal*]
C4 = [*friend, kids*]
C5 = [*work, school*]

2.2.5.6 *Results Obtained*

As the reader can observe from the results obtained in the previous example, given low SCs, the threshold must also be quite low in order to obtain any clusters at all. Hirschman et al. executed their algorithm on a 400 sentence text, and their threshold of 0.25 gave them very good clusters.

I ran AZTEC's augmented version of the clustering algorithm on two texts, the 553 sentences of chapters 1 through 6 of the PowerHouse QUIZ manual, and 486 sentences of Delisle's 1990 parser text.

I initially ran the algorithm on the QUIZ chapters using a clustering threshold of 0.25, but no multi-word clusters (i.e. clusters with more than one member) were formed. I then started experimenting with lower thresholds (0.1, 0.04, 0.03 and 0.01). I found that a threshold of 0.03 yielded the best results with the QUIZ text. I obtained 10 multi-word clusters, but only five of those 10 clusters were noun clusters. The remainder were verb clusters. The five noun clusters, along with their concept category, can be found below. The concept categories associated with the NPs below were computed by AZTEC's Classification algorithm (see Appendix 8) and do not represent the optimal classifications. **T** stands for **Technical**, **R** for **Rather Technical**, **P** for **Possibly Technical**, **N** for **Not Technical** and **U** for **Unknown**.

1 [computer (P), report (T), line (T), production report (N), use (P)]

2 [page (P), contents (N)]

3 [dictionary (N), statement (T)]

4 [employee file (R), file (T)]

5 [good example (N), example (T)]

Even though there are some mis-classifications (e.g. *line* and *production report* would be better classified as **Rather Technical**, and *example* and *dictionary* as **Possibly Technical**), these results were not very enlightening. Two-word clusters provide very little information, especially if both terms in the cluster are formed around the same head noun, as is the case for clusters #4 and #5. AZTEC does not need a clustering algorithm to deduce a relationship between two such terms.

Only the first three clusters prove to be of some interest, but the first cluster contains some common-usage words (*line, use*) among more technical ones (*computer, report, production report*). Experiments on this text have shown that regardless of the threshold used, this problem re-occurs. It seems impossible to infer a single concept category for an entire cluster, since most clusters contain a mixture of technical and non-technical terms.

Using a threshold $t = 0.25$ on Delisle's 1990 parser text, I obtained five noun clusters, all of which were made up of two terms, a head noun and a term with the same head noun. A threshold of $t = 0.1$ produced similar results, yielding 27 noun clusters, 18 of which had two members build around the same head noun. I obtained more interesting clusters by using a threshold of 0.04. Thirty-three noun clusters were formed and are listed below, along with the classification category of each concept, as computed by AZTEC's Classification algorithm (see Appendix 9).

#1 [sentence (T), grammar (T), syntactic phenomenon (T), phenomenon (R), implementation (P), core (P), parse tree (R), tree (P)]

#2 [model (N), entry (P), part (P), form (N), interpretation (P), description (N), certain form (N)]

#3 [design (P), base manager (P), manager (P)]

#4 [case analyzer (P), analyzer (P), input (T), full integration (P), integration (P)]

#5 [syntactic structure (R), structure (P), technical text (T), text (T), parse tree (R), tree (P), substring (P), joining-point (P)]

#6 [example (P), document (P), description (N), core (P), parse tree (R), tree (P), detail (N), difficulty (P)]

#7 [element (P), construction (P), conjoined element (R), transitive construction (R), gerundive construction (R)]

#8 [module (R), syntactic module (T)]

#9 [subsection (P), word (P), dictionary (P), part (P), input (T), reference (P)]

#10 [helper (P), mechanism (R), process (P), look-head mechanism (T)]

#11 [purpose (N), conjoined sentence (T)]

#12 [category (P), way (N), parse tree (R), tree (P), substring (P), joining-point (P), conjunction (P)]

#13 [information (P), user (R), indication (P), sequence (P)]

#14 [string (N), form (N), description (N), constituent (P), situation (P), certain form (N), discontinuous constituent (R)]

#15 [unknown word (R), parse tree (R), tree (P), reference (P), phrase (T), joining-point (P), conjunction (P), verb (N)]

#16 [session (R), parse tree (R), tree (P), parse session (T), joining-point (P), conjunction (P)]

#17 [rule (T), reference (P)]

#18 [detailed description (N), description (N)]

#19 [type (N), approach (R), frequent type (N)]

#20 [reader (P), proper noun (N), noun (N)]

#21 [name (N), presentation (P)]

#22 [predicate (P), order (P)]

#23 [section (R), constraint (P)]

#24 [menu (P), detail (N)]

#25 [note (N), fragment (R), joining-point (P), conjunction (P), parse (R)]

#26 [assertion (P), assumption (P)]

#27 [issue (P), parsing (R), implicit subject (P), subject (N), implicit complement (R), complement (P), verb sequence (P), sequence (P), auxiliary (P)]

#28 [title (N), orthographic bit (U), bit (U)]

#29 [verb phrase (R), phrase (T), conjoined sentence (T)]

#30 [time (P), constraint (P)]

#31 [maximum amount (P), amount (N)]

#32 [pervasive problem (R), problem (P)]

#33 [natural language (R), language (P)]

Although they are far more informative than the clusters extracted from the QUIZ text, possibly because the entire Parser text was written by a single author, the above clusters are of limited use to concept classification. As with the QUIZ text, most of the clusters contain concepts which do not belong to a single concept category.

For example, cluster #1 contains the concepts *sentence*, *grammar* and *syntactic phenomenon*, which are **Technical** concepts, *phenomenon* and *parse tree*, which are **Rather Technical** concepts, and *implementation*, *core* and *tree* which are **Possibly Technical** concepts. No single concept category could be applied to all the concepts in this cluster.

In cluster #15, since the average concept category is somewhat technical, we could have inferred a more technical category for *verb*, which is mis-classified as **Not Technical**. Unfortunately, clusters #12 and #25 show that there are cases in which this practice would lead to the mis-classification of the term whose category is updated.

As a result, since noun clusters can be of no practical use for classification verification and improvements, the clustering algorithm was removed from the Classification by Block process.

2.3 The NP Seeker

The NP Seeker serves as the interface between AZTEC and the Sentence Parser, and it is invoked with the sentence currently being processed by either the Classification by Sentence or the Classification by Block algorithm. In turn, the NP Seeker invokes the Sentence Parser with the sentence and analyzes the syntactic structure returned in order to extract from it only those syntactic sub-structures associated with NPs (see Figure 8).

2.3.1 Using the Sentence Parser

The main draw-back in using the current version of the Sentence Parser to analyze a complex sentence is the amount of time that it takes. In some cases, the parser can take over five minutes for one sentence, which is quite long if the technical text contains over 500 sentences. To ensure that a text of that length takes no longer than approximately 12 hours to be processed using Quintus Prolog 2.5.1 on a Sparc Station 1, the timer associated with the Sentence Parser was set to time-out after five seconds. Longer intervals were initially attempted (15 and 20 seconds), but they caused the Classification by Block algorithm to run for more than 18 hours on a 500 sentence text, so the interval was reduced to five seconds.

If the Sentence Parser can successfully parse the sentence in five seconds or less, the resulting syntactic structure is examined and its NP structures are extracted and further examined for embedded NP structures.

If the Sentence Parser is unable to parse the sentence in five seconds or if it cannot recognize the sentence to be grammatically correct, an alternative approach is attempted. Each word of the sentence is examined, one at a time, to determine whether or not it could possibly mark the beginning of a NP. Among the syntactic categories of words recognized by the Sentence Parser, the following categories can begin NPs: articles, adjectives, adverbs, cardinals,

count nouns, demonstrative or pronominal adjectives, mass nouns, nominal pronouns, ordinals, proper nouns, possessive adjectives, possessive pronouns, quantifiers, question words and relative pronouns.

When such a *start word* is encountered, the rest of the sentence beginning at that word is sent to the NP module of the Sentence Parser. If a NP is found and parsed starting at the current word, its syntactic structure is returned by the parser, as well as the un-parsed portion of the sentence (i.e. the portion starting at the end of the newly parsed NP). The words of the un-parsed portion will also be examined for start words and subsequent NPs. If the current word is not a start word or if the parser cannot recognize a NP at the current start word, the current word is skipped, and the next word is processed in the same way. A time limit of five seconds is associated with each invocation of the NP module of the Sentence Parser.

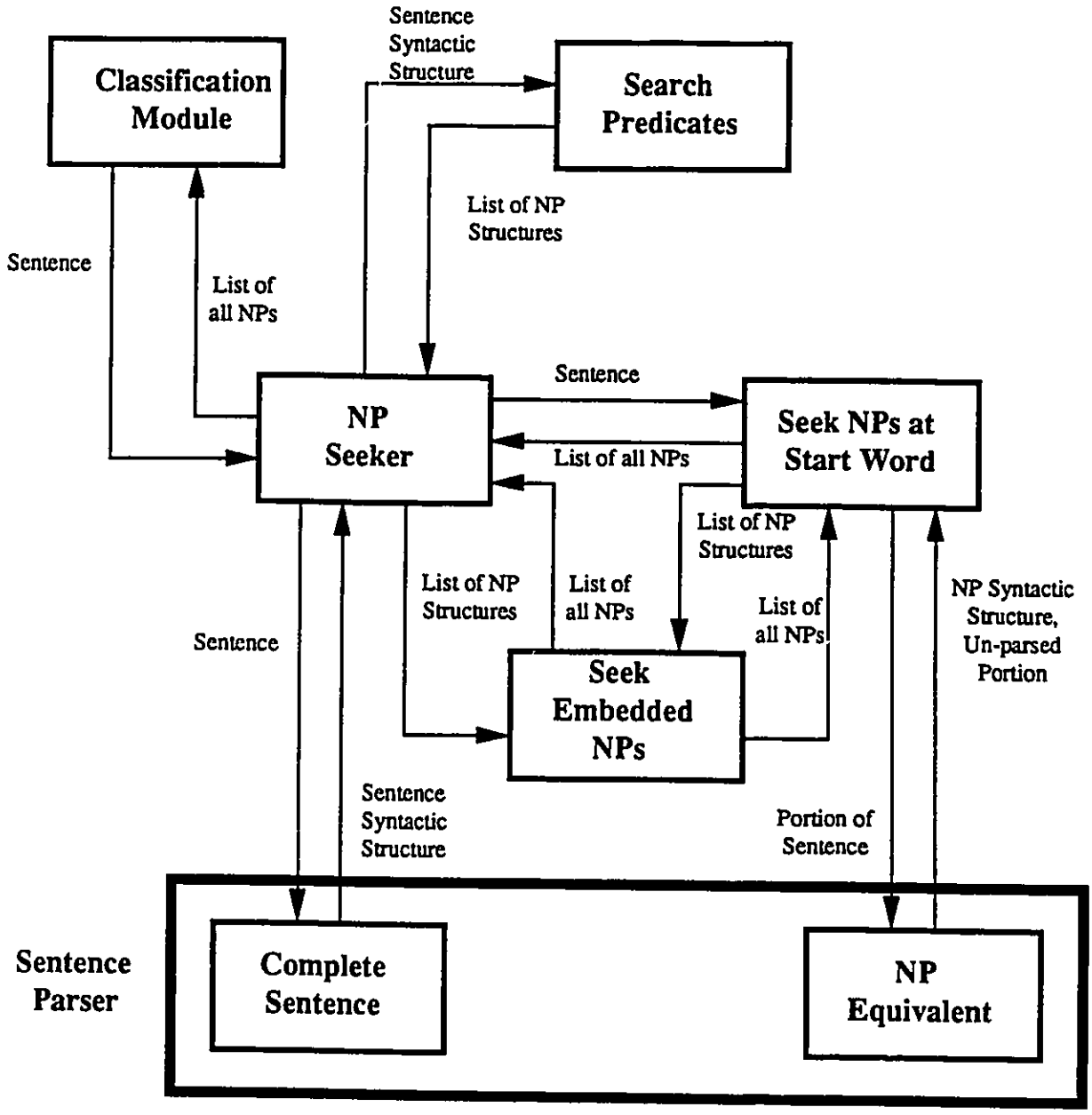


Figure 8 - NP Seeker Module

2.3.2 Extracting the NP Structures

There are two steps involved in extracting all the NPs from the sentence syntactic structure returned by the Sentence Parser: first, the structure is searched for NPs and then each NP found is examined for embedded NP structures.

The following is an example of the syntactic structure of the sentence: *Are there any new features that I should use?*[1]

```
be_question
  wh_word  nil
 be  pl  pres
  exist_ref
   exist  there
 advp  []
 obj1
   entity
    noun  feature
     classifiers
      pre_class  []
      post_class  nil
    number  pl
    det_seq  any
    attributes
    attr  new   pos
     advp  []
    appendages
    prep  nil
     rel_clause
        rel_marker  that
       statement
       subj
        ref  i
           number  sg
        predicate
         regular
          verb  use     conditional_present_simple
            particle  nil
            negation  yes
```

[1]Delisle, Parser for Technical Texts, 42-43.

```
        adverbs []
      voice active
      complement
       entity
         noun feature
          classifiers
           pre_class []
           post_class nil
         number pl
         det_seq any
         attributes
         attr new pos
          advp []
       verb_clause_modifier []
      sentence_head_modifier []
obj2_or_action nil
```

Since the NPs associated with concepts are found in the *entity* structures, the NP Seeker only considers those sub-structures of the sentence structure. Therefore, in its first step, the NP Seeker will extract the following *entity* structure:

```
entity
  noun feature
    classifiers
     pre_class []
     post_class nil
  number pl
  det_seq any
  attributes
  attr new pos
    advp []
  appendages
  prep nil
    rel_clause
      rel_marker that
       statement
       subj
        ref i
          number sg
```

```
predicate
 regular
  verb use    conditional_present_simple
    particle nil
    negation yes
    adverbs []
 voice active
  complement
   entity
     noun feature
      classifiers
       pre_class []
       post_class nil
      number pl
      det_seq any
      attributes
       attr new  pos
        advp []
    verb_clause_modifier []
   sentence_head_modifier []
```

In its second step, the NP Seeker will take the *entity* structure and search it for embedded *entity* structures. Because the sentence contains a relative clause, *that I should use*, the parser will replace the relative pronoun *that* by the NP it modifies, in this case *any new features*, and the relative clause will be re-constructed as the complete sentence *I should use any new features*. As a result, the NP Seeker will extract two occurrences of the NP *any new features* even though it only appears once in the sentence. This feature of the Sentence Parser has proven to be very useful for AZTEC, since *any new features* actually occurs once in the sentence (*are there any new features?*) and once in the embedded sentence (*I should use any new features*). In the end, the following list of two *entity* structures will be returned to the Classification Module:

```
entity
  noun features
    classifiers
     pre_class []
```

```
    post_class  nil
  number  pl
  det_seq  any
   attributes
    attr  new  pos
     advp  []

entity
  noun  features
   classifiers
    pre_class  []
    post_class  nil
  number  pl
  det_seq  any
   attributes
    attr  new  pos
     advp  []
```

2.3.3 An Example

Since section 2.3.2 has provided an example of the case where the entire sentence is successfully parsed, this section will use the sentence *You and I are printing our files and we are reading our messages and letters*[1] to give an example of the case where the sentence is not parsed in its entirety.

Given that none of the first five words of the sentence, *you and I are printing*, can be start words, they are skipped. However, since the sixth word, *our*, is a possessive adjective and therefore a potential start word, the entire sentence beginning at *our* is sent to the NP Equivalent module of the Sentence Parser. A valid NP (*our files*) will be recognized, and the remaining un-parsed portion of the sentence will be: *and we are reading our messages and letters.*

The next potential start word is *our*. The portion *our messages and letters* is passed on to the NP Equivalent module of the Sentence Parser and recognized as a NP.

The two NPs obtained in this manner will be the following:

---

[1]Ibid., 41-42.

```
entity
  noun file
    classifiers
     pre_class []
     post_class nil
  number pl
  det_seq poss/1
  attributes []

conj_nps and 2 pl
    entity
     noun message
       classifiers
        pre_class []
        post_class nil
     number pl
     det_seq poss/1
     attributes []
    entity
     noun letter
      classifiers
       pre_class []
       post_class nil
     number pl
    det_seq []
    attributes []
```

Each NP will then be examined for embedded *entity* structures, and the final list of all the NPs found in the sentence will be the following:

```
entity
  noun file
    classifiers
     pre_class []
     post_class nil
  number pl
  det_seq poss/1
  attributes []
```

```
entity
  noun  message
    classifiers
     pre_class  []
     post_class  nil
  number  pl
  det_seq  poss/1
  attributes  []

entity
  noun  letter
    classifiers
     pre_class  []
     post_class  nil
  number  pl
  det_seq  []
  attributes  []
```

2.4 The Evolution of the Classification Algorithms

As mentioned previously in this chapter, the evolution of the Classification algorithms began with Classification by Sentence. The two measures used in the Classification by Sentence process were known from the outset: the frequencies of occurrence of the words of a NP compared against a Minimum Bound, and the dictionary flags which would indicate whether or not a head noun or a modifier appears in Longman's list of common words. Once I decided to use one consolidated dictionary flag for all the modifiers and the total frequency of the words of a given NP (see section 2.1.2), a three-dimensional Classification Table was needed to compute the category of each concept, with one dimension representing the dictionary flag of the head noun, one dimension the dictionary flag of the modifier(s), and one dimension the frequency of occurrence of the words of the NP.

When I realized that classifying the NPs of an entire block of text would mean that a specific set of sentences would need to be

parsed, and that it would be possible to extract their NPs and accumulate them throughout the text, I decided to implement this new approach as Classification by Block. This approach initially classified NPs just as Classification by Sentence had by using the same Classification Table. The only differences were the term frequency, rather than the total word frequency, compared against the Minimum Bound, and the use of only the last of the head noun's modifiers and its dictionary flag, rather than the consolidated dictionary flag of all the modifiers (see section 2.2.1).

Ever since the earliest Classification by Sentence algorithms, improvements to the classification results were made by examining the results, noticing problematic trends, and finding ways of curbing these trends so that their impact would be minimized. Completely neutralizing problematic trends proved to be impossible, since the measures used to compute the classification category of each NP (frequencies and dictionary flags) would essentially remain the same.

The Classification Table itself has somewhat changed over the evolution of the Classification by Sentence and Classification by Block algorithms. Figure 9 gives an idea of the very first Classification Table, using only one Minimum Bound with either the total word frequency of the NP (Classification by Sentence) or the term frequency (Classification by Block). The optimized table can be found in Figure 5. As the reader can see, the optimized table deals with more cases and thus smaller frequency intervals than the original table, but the classification categories for infrequent NPs are very similar in both tables. Essentially, some changes have been made to the original table to take into account the new types of frequency considered, but the nature of the Classification Table has changed very little. The only greatly varying measures from one version of the Classification algorithm to the next have been the types of frequencies used and the Minimum Bounds against which they are compared.

Head Noun Flag	Modifier Flag \ Term Freq.	Term Freq >= MinBound	Term Freq < MinBound
"n"	"n"	TECH	POSS
	0	TECH	POSS
	"y"	PROB	POSS
"y"	"n"	PROB	POSS
	0	PROB	NOT TECH
	"y"	PROB	NOT TECH

Legend:

TECH:	Technical
PROB:	Probably Technical
POSS:	Possibly Technical
NOT TECH:	Not Technical

Figure 9 - Original NP Classification Table

2.4.1 The Evolution of the Minimum Bounds

The earliest Minimum Bounds were measured in terms of the number of occurrences of the words of a NP with respect to the number of lines in the text. I soon realized that using percentages of the total number of words in the text yielded more reliable results.

Before the Classification by Sentence algorithm was optimized, I started exploring the idea of Classification by Block and set Classification by Sentence aside for a while. In the beginning, the Classification by Block algorithm was very similar to the Classification by Sentence algorithm. AZTEC used the head noun and the last modifier's dictionary flags and the term frequency compared against one Minimum Bound (a percentage of the total number of NPs in the text, rather than words) to assign the NP to one of the four following concept categories: **Technical, Probably Technical, Possibly Technical,** and **Not Technical.**

Tests were carried out with a Minimum Bound of 10% of the total NPs in the text, then 5%, 4%, 3.5% and 3%. The percentages used began at 10% and were steadily decreased, because there were very few terms (only 2 or 3 terms on average for a text with over 150 NPs in total) which occurred more than 10% of the time. The results were examined for the average number of occurrences of the more technical NPs, and I determined that many of them occurred more than 3% of the time, while many of the less technical terms occurred less than 3% of the time, so this number (3%) was retained as the Minimum Bound.

This is a typical example of the approach I used to refine the Minimum Bounds and the Classification process in general. I examined the classification results of the current version and determined which NPs could be better classified. I then determined whether or not all the other NPs fitting the same criteria as the mis-classified NPs (i.e. the NPs with the same dictionary flags and similar frequencies) were also mis-classified, for example whether or not the majority of NPs with a "y" dictionary flag for the head noun, a "n" dictionary flag for the modifier and a frequency below the Minimum

Bound should be classified as **Probably Technical** rather than **Possibly Technical**. If the majority of NPs fitting a particular criteria were mis-classified in the same way, it meant that their classifications could all be improved simultaneously with one change to the Classification algorithm. The same applied to the testing of Minimum Bounds. If a greater number of proper classifications than mis-classifications could be achieved with one change, then that change was made, and I expected that further tests using different texts would reveal in time whether or not these changes were warranted.

One cause for mis-classifications involved terms made up of only head nouns with no modifiers, which have similar frequencies but should be classified in much different categories. For example, in chapters 1 and 2 of the QUIZ manual, *report* occurs five times by itself, and *key* occur four times, although *report* should be classified as **Technical** and *key* as **Possibly Technical** (see Appendix 2). This problem was solved by incrementing the frequency of the head noun each time a term built around the head noun was encountered in the text (see section 2.2.1). Subsequent classifications of *report*, for example, would use its overall number of occurrences as a head noun, rather than the number of times it occurs in the text by itself. As a result, since *report* appears an additional three times as the head noun of a term, and no terms are built around the head noun *key*, the head noun frequency used for *report* is 8, and this number better reflects the technicality of *report* with respect to that of *key* (see Appendix 3). Eventually, because the head noun frequency is a better measure of the technicality of any term, it was adopted instead of term frequency to drive the concept classification of all terms, both those made up of a modifier and a head noun and those consisting of only a head noun (see section 2.2.2.1). The use of Term Percentages (described below) ensured that the term frequency still played a role in the Classification process, so that non-technical terms with a technical head noun would not be over-classified.

I then noticed that the **Possibly Technical** category was getting cluttered with terms of varying technicality, whereas the **Probably Technical** category was consistently empty. In order to curb this trend, I introduced a second Minimum Bound, set to 1.5% of the total number of NPs in the text. This measure was obtained by observing from the results that NPs occurring less than 1.5% of the time were usually less technical than NPs occurring more often (see Appendix 4). Given this additional Minimum Bound, the Classification Table was updated with new classifications.

A third Minimum Bound, set to the constant 1.1 was later introduced when the technicality **Unknown** category was added. It seemed logical to classify in the technicality **Unknown** category NPs whose head noun only occurs once in the text. At first, NPs whose *term* frequency was 1 were classified as technicality **Unknown**, but this resulted in a very large number of NPs in this category, most of which could be properly classified because their head noun occurred more than once in the text, and it could provide a good approximation of their technicality. The use of head noun frequencies ensured that only the NPs which truly could not be classified based on frequency were assigned to the **Unknown** category.

With all the changes made to the Classification Table and the use of texts of increasing lengths (1900-2100 NPs in total) to test AZTEC, I noticed that the highest Minimum Bound (3%) was no longer being used in the newest Classification Table, because in texts with higher numbers of NPs, very few NPs occur as often as 3% of the time. The second Minimum Bound (1.5%), however, was still very useful in separating the more technical terms from the less technical ones, so it was retained as the highest Minimum Bound, and another Minimum Bound (0.75%) was introduced, again based on observation of the results and of what range of boundary would better segregate the NPs occurring less than 1.5% of the time.

Once the Classification by Block algorithm had been optimized, I returned to the optimization of Classification by Sentence to find the best Minimum Bounds as percentages of the total number of

words in the text. I ran tests with 2%, 1.5%, 1% and 0.5% for the highest Minimum Bound, and 1%, 0.75%, 0.5% and 0.25% for the second bound. The best results were obtained using 0.5% as the highest bound and 0.25% as the second bound. Appendix 1 has been organized to show the results of Classification by Sentence using some of the sentences of chapters 1 and 2 of the QUIZ manual and the word frequencies accumulated for chapters 1 through 6 of the QUIZ manual. This provides the reader with an idea of the type of classification obtained for chapters 1 through 6 of QUIZ, although instead of presenting the results individually for each sentence, I have grouped together the NPs of each classification category.

2.4.2 The Evolution of Term Percentages

Unfortunately, with the use of head noun frequencies rather than term frequencies in Classification by Block, there were still some mis-classified terms. Some technical terms made up of a technical head noun and a modifier only occurred once or twice in a short text (i.e. a text with around 150 NPs in total) and would be under-classified, and some non-technical terms built around technical head nouns would be over-classified. As a result, Term Percentages were introduced, where the Term Percentage of a term is the ratio of its term frequency to the frequency of its head noun. Based on empirical observations of the results and the NPs whose classifications were to be improved, a boundary of 20% to be compared against the Term Percentage seemed to be an appropriate choice.

Later results showed that better classifications could be obtained by using a different Term Percentage boundary. Less technical terms either occur very infrequently, in which case their Term Percentage does not matter for the computation of the Frequency Area (see Figure 4), or they do not occur frequently with respect to their head noun, i.e. they account for less than 10% of the total occurrences of their head noun. For example, in chapters 1 through 6 of the QUIZ manual, the head noun *option* occurs 31 times,

but *available option* only occurs once, which means that its Term Percentage is 1 × 100 / 31 = 3.2%. For this reason, a Term Percentage boundary of 10% was introduced to ensure that the very infrequent terms build around more technical head nouns would not be over-classified. It became clear from the results that as the texts used to test AZTEC became longer, fewer terms whose head noun frequency is above the highest Minimum Bound account for more than 20% of the occurrences of their head nouns. The reason for this is that frequently occurring head nouns are either often referred to using only the head noun with no modifiers, or they occur with several different modifiers, meaning that no one term uses the head noun exclusively (i.e. no term has a relatively high Term Percentage). As a result, the Term Percentage boundary was changed from 20% to 10%.

With the introduction of Term Percentages, the Classification Table became a four-dimensional table, so I decided to introduce an intermediate step in the Classification process, where the Frequency Area (FA) of a NP would be computed using the head noun frequency and the Term Percentage (see Figure 4). Once a NP is assigned to one of four FAs, this FA and the dictionary flags of the head noun and the modifier(s) are used in conjunction with the Classification Table to obtain a concept category for the NP (see Figure 5).

2.5 The Classification Results

2.5.1 The Classification by Block Results

I ran the optimized Classification by Block algorithm on two large texts, chapters 1 through 6 of the QUIZ manual containing 1978 NPs in all, and the 1990 Parser text containing 2030 NPs. I then ran the same algorithm using one smaller text, chapters 1 and 2 of QUIZ, which contain a total of 163 NPs. Tables 3 and 4 reflect the success and failure rates of the AZTEC algorithm described in section 2.2 on the two large texts.

The tables in this section were created by comparing the actual AZTEC results against the expected results found in Appendices 6 (for the QUIZ text) and 7 (for the Parser text), which I determined manually according to the categories in which I believed each NP should be classified. The tables can be interpreted in the following manner. Each row of the table is associated with one of the classification categories, except the last row which contains the totals of each column. The *-x* and *+x* columns reflect the number of NPs which appear in the current classification category but which have been mis-classified. The *-x* columns indicate the number of NPs which should have been classified *x* categories below the current category, and the *+x* columns indicate the number of NPs belonging to the category *x* categories above the current one. The *OK* column indicates the number of correctly classified NPs in the current category, the *Total Mis-Classified* column contains the number of mis-classified NPs in the current category, and the *Total NPs* column indicates the total number of NPs that AZTEC has assigned to the current category. In Table 3, for example, AZTEC successfully classified a total of 486 distinct NPs, 19 of which were classified in the **Technical** category, 77 in the **Rather Technical** category, 243 in the **Possibly Technical** category, and 147 in the **Not Technical** category. Of the 243 NPs classified in the **Possibly Technical** category, 35 should have been classified in the **Not Technical** category (one category below **Possibly Technical**), 192 NPs were properly classified, and 16 should have been classified in the **Rather Technical** category (one category above **Possibly Technical**). Therefore, of the 243 NPs in the **Possibly Technical** category, 79% were properly classified and only 21% were mis-classified.

The best results achieved by AZTEC (see Appendix 8) with any of the texts are found in Table 3, which shows a 60% success rate and a 40% mis-classification rate using chapters 1 through 6 of QUIZ, with only 1% of the NPs mis-classified by two categories (columns *-2* and *+2*) and 39% of the NPs mis-classified by one category (columns *-1* and *+1*). Table 4, which contains the results of AZTEC using Delisle's

parser text (see Appendix 9), shows a success rate of 51% and a 48% mis-classification rate. Although these results leave something to be desired, they nonetheless reflect the best results achieved with the Parser text.

Experiments were conducted with both the large QUIZ text and the Parser text to ensure that the two Minimum Bounds and the Term Percentage boundary used to classify the NPs were set to their optimal values for each text. The QUIZ text was used to test Term Percentage boundaries of 20% (Table 5) which is twice the optimal value of 10%, 30% (Table 6) and 5% (Table 7) which is half of the optimal value. The Minimum Bounds were also tested using the same text with values of 3% and 1.5% (Table 8) which are twice the optimal values of 1.5% and 0.75%, and with values of 0.75% and 0.375% (Table 9) which are half of the optimal values.

The results of Table 5 show that a Term Percentage boundary of 20% yields the same mis-classification rate (40%) as the optimal 10% Term Percentage boundary, and that boundaries of 30% and 40% (Table 6) yield a 41% mis-classification rate, which is only 1% higher than the optimal. The reason for this has been stated in section 2.4: as the total number of NPs increases with the length of a text, fewer NPs account for more than 10% of the number of occurrences of their head noun. Table 7, however, shows that a Term Percentage boundary of 5% yields a more significant mis-classification rate (44%).

Variations in the Minimum Bounds also proved to have a negative impact on the results. Table 8, which shows the results using Minimum Bounds of 3% and 1.5% of the total NPs in the text, has a mis-classification rate of 42%, which is 2% higher than with the optimal values. Table 9 shows that the results obtained with Minimum Bounds of 0.75% and 0.375% are even worse: they result in a 47% mis-classification rate.

Similar experiments were carried out using the Parser text, which contains 663 distinct NPs, with Term Percentage boundaries of 20% (Table 10), 5% (Table 11) and 2.5% (Table 12). The Minimum

Bounds were also tested with values of 3% and 1.5% (Table 13) and with values of 0.75% and 0.375% (Table 14).

Although the algorithm using the optimal bounds yields a mis-classification rate of 48% (Table 4), Term Percentage boundaries of 20% and 30% (Table 10) both yield a 50% mis-classification rate, and a 5% boundary (Table 11) yields a 49% mis-classification rate. An additional test was carried out with a Term Percentage boundary of 2.5% (Table 12), and it yielded a 52% mis-classification rate.

This suggests that just as the QUIZ text has an optimal Term Percentage boundary very close to the maximum that it could be, i.e. the optimal boundary of 10% yielded results similar to but slightly better than the results obtained with boundaries of 30% and 40%, the ideal Term Percentage boundary using the Parser text may be slightly below 10%, since a boundary of 5% produces results similar to those obtained with the 10% boundary.

In terms of Minimum Bounds, however, the results were in keeping with those obtained with the QUIZ text, namely that Minimum Bounds of 1.5% and 0.75% are the optimal bounds. Tests with Minimum Bounds of 3% and 1.5% yielded a 52% mis-classification rate (Table 13), and bounds of 0.75% and 0.375% showed a 53% mis-classification rate (Table 14).

These experiments showed that for two texts of 485 and 663 distinct NPs respectively, the Minimum Bounds were optimized at 1.5% and 0.75% of the total number of NPs in the text, but that the Term Percentage boundary may be different for each text, even though the value used in the "optimal" algorithm (10%) yielded the best results for both texts. In general, the Term Percentage boundary seems to yield similar results when set to higher values, such as 30% and 20%. Slightly better results are obtained when the boundary is at its optimal value, and lower values increase the mis-classification rate. Although a Term Percentage boundary between 10% and 20% seems optimal for the QUIZ text, the best value for the Parser text seems to be between 10% and 5%. It was only in tabulating AZTEC's

results with the shorter text (chapters 1 and 2 of the QUIZ manual) that the conclusion from these observations became obvious.

The results in Appendix 4, obtained using Minimum Bounds of 3% and 1.5% and a Term Percentage boundary of 20%, were compared against the expected classifications of each NP (Appendix 6), and the results of this comparison were tabulated in Table 15. The mis-classification rate was found to be 45%, with 38% of the NPs mis-classified by one category and 7% by two categories. However, when the so-called "optimal" values were used with this text (the results can be found in Appendix 5), a mis-classification rate of 75%, with 45% of the NPs mis-classified by one category and 30% by two categories was obtained (Table 16).

This observation, coupled with the results of the experiments on the two longer texts, revealed a trend: that the longer the text, the lower the optimal percentages used as Minimum Bounds and the lower the optimal Term Percentage boundary. The first two QUIZ chapters contain a total of 163 NPs, and the optimal results (with a 55% success rate in Table 15) are obtained using the two Minimum Bounds set to 3% and 1.5% of the total NPs in the text and a Term Percentage boundary of 20%. The text in chapters 1 through 6, however, contains a total of 1978 NPs, and the optimal results for this text (with a 60% success rate in Table 3) are obtained using the two Minimum Bounds set to 1.5% and 0.75% of the total NPs in the text and a Term Percentage boundary of 10% or 20%. A 51% success rate (see Table 4) is achieved with the Parser text containing a total of 2030 NPs using the same Minimum Bounds and a 10% Term Percentage boundary, although the use of a 5% Term Percentage boundary yields the same results.

This leads to the conclusion that the percentages used for the Minimum Bounds and the Term Percentage boundary should be inversely proportional functions of either the total occurrences of NPs or the number of distinct NPs in the text. Although the initial release of AZTEC has not been implemented with the capability of dynamically determining the Minimum Bound percentages and the

Term Percentage boundary used for concept classification, this should be considered in future enhancements. By running AZTEC on texts of various lengths, it may be possible to approximate the functions which could compute the Minimum Bound percentages and the Term Percentage boundary using the number of NPs in the text.

Upon closer examination of Tables 3 and 4, it is possible to see the main reason why there is a limit to the improvements which can be made to the classification results. Most of the mis-classifications in Tables 3 and 4 occur in the **Not Technical** category, which has the lowest success rate for both large texts (38% for the QUIZ text and 43% for the Parser text). With the QUIZ text, 61% of the NPs ranked as **Not Technical** really belong to the **Possibly Technical** category, and 43% of the NPs in the Parser text are mis-classified in the same way. Unfortunately, **Not Technical** concepts and those which belong to the **Possibly Technical** category are virtually indistinguishable in terms of frequencies and dictionary flags. Two terms can easily have the same dictionary flags and very similar term and head noun frequencies, and yet belong to different categories. As a result, it is impossible, using the same measures, to improve the success rate in the **Not Technical** category, short of merging it with the **Possibly Technical** category.

This may be considered in future enhancements, but the decision to merge both categories should be based on how each will be used. The NPs which belong to the **Possibly Technical** category are usually relatively technical concepts in a more general domain related to the one described in the technical text. The QUIZ manual, for example, contains many computer-related terms which are optimally ranked as **Possibly Technical** in the context of the QUIZ text but which are actually technical computer terms. If the application for which AZTEC is to be used does not need to consider such terms, then no effort needs to be made to extract the true **Possibly Technical** terms from the **Not Technical** category. If, however, these related terms are found to be indispensable, then

additional measures will need to be devised to further segregate the terms classified in the **Not Technical** category.

2.5.2 The Classification by Sentence Results

As expected, the Classification by Sentence results are not as good as the Classification by Block results. Table 17 shows the success and failure rates of the optimal Classification by Sentence algorithm using the word frequencies of chapters 1 through 6 of the QUIZ manual, but using only some of the sentences from chapters 1 and 2. A mis-classification rate of 47% was obtained, with 33% of the NPs mis-classified by one category, 13% by two categories and 1% by three categories. Compared to the results of Table 3, where Classification by Block is used to classify the NPs of chapters 1 through 6, and where the mis-classification rate is 40%, Classification by Sentence is found to be far less reliable.

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	4	7	8	0	0	0	7	4	0	11	19
	0% of 19	21% of 19	37% of 19	42% of 19	0% of 19	0% of 19	0% of 19	37% of 19	21% of 19	0% of 19	58% of 19	
RATHER	0	2	35	33	7	0	0	42	2	0	44	77
	0% of 77	3% of 77	45% of 77	43% of 77	9% of 77	0% of 77	0% of 77	54% of 77	3% of 77	0% of 77	57% of 77	
POSS	0	0	35	192	16	0	0	51	0	0	51	243
	0% of 243	0% of 243	14% of 243	79% of 243	6% of 243	0% of 243	0% of 243	21% of 243	0% of 243	0% of 243	21% of 243	
NOT TECH	0	0	0	56	90	1	0	90	1	0	91	147
	0% of 147	0% of 147	0% of 147	38% of 147	61% of 147	1% of 147	0% of 147	61% of 147	1% of 147	0% of 147	62% of 147	
Totals	0	6	77	289	113	1	0	190	7	0	197	486
	0% of 486	1% of 486	16% of 486	60% of 486	23% of 486	0% of 486	0% of 486	39% of 486	1% of 486	0% of 486	40% of 486	

Table 3 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	0	2	15	0	0	0	2	0	0	2	17
	0% of 17	0% of 17	12% of 17	88% of 17	0% of 17	0% of 17	0% of 17	12% of 17	0% of 17	0% of 17	12% of 17	
RATHER	0	10	76	91	21	0	0	97	10	0	107	198
	0% of 198	5% of 198	38% of 198	46% of 198	11% of 198	0% of 198	0% of 198	49% of 198	5% of 198	0% of 198	54% of 198	
POSS	0	0	59	182	77	7	0	136	7	0	143	325
	0% of 325	0% of 325	18% of 325	56% of 325	24% of 325	2% of 325	0% of 325	42% of 325	2% of 325	0% of 325	44% of 325	
NOT TECH	0	0	0	53	53	10	7	53	10	7	70	123
	0% of 123	0% of 123	0% of 123	43% of 123	43% of 123	8% of 123	6% of 123	43% of 123	8% of 123	6% of 123	57% of 123	
Totals	0	10	137	341	151	17	7	288	27	7	322	663
	0% of 663	2% of 663	21% of 663	51% of 663	23% of 663	2% of 663	1% of 663	43% of 663	4% of 663	1% of 663	48% of 663	

Table 4 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	2	4	6	0	0	0	4	2	0	6	12
	0% of 12	17% of 12	33% of 12	50% of 12	0% of 12	0% of 12	0% of 12	33% of 12	17% of 12	0% of 12	50% of 12	
RATHER	0	2	34	34	8	0	0	42	2	0	44	78
	0% of 78	2% of 78	44% of 78	44% of 78	10% of 78	0% of 78	0% of 78	54% of 78	2% of 78	0% of 78	56% of 78	
POSS	0	0	35	193	18	1	0	53	1	0	54	247
	0% of 247	0% of 247	14% of 247	78% of 247	7% of 247	0% of 247	0% of 247	21% of 247	0% of 247	0% of 247	22% of 247	
NOT TECH	0	0	0	56	91	1	0	91	1	0	92	148
	0% of 148	0% of 148	0% of 148	38% of 148	61% of 148	1% of 148	0% of 148	61% of 148	1% of 148	0% of 148	62% of 148	
Totals	0	4	73	289	117	2	0	190	6	0	196	485
	0% of 485	1% of 485	15% of 485	60% of 485	24% of 485	0% of 485	0% of 485	39% of 485	1% of 485	0% of 485	40% of 485	

Table 5 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 20%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	2	4	5	0	0	0	4	2	0	6	11
	0% of 11	18% of 11	36% of 11	45% of 11	0% of 11	0% of 11	0% of 11	36% of 11	18% of 11	0% of 11	55% of 11	
RATHER	0	2	34	34	9	0	0	43	2	0	45	79
	0% of 79	3% of 79	43% of 79	43% of 79	11% of 79	0% of 79	0% of 79	54% of 79	3% of 79	0% of 79	57% of 79	
POSS	0	0	35	192	18	1	0	53	1	0	54	246
	0% of 246	0% of 246	14% of 246	78% of 246	7% of 246	0% of 246	0% of 246	22% of 246	0% of 246	0% of 246	22% of 246	
NOT TECH	0	0	0	56	92	1	0	92	1	0	93	149
	0% of 149	0% of 149	0% of 149	38% of 149	62% of 149	0% of 149	0% of 149	62% of 149	0% of 149	0% of 149	62% of 149	
Totals	0	4	73	287	119	2	0	192	6	0	198	485
	0% of 485	1% of 485	15% of 485	59% of 485	25% of 485	0% of 485	0% of 485	40% of 485	1% of 485	0% of 485	41% of 485	

Table 6 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 30% and 40%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	8	15	8	0	0	0	15	8	0	23	31
	0% of 31	26% of 31	48% of 31	26% of 31	0% of 31	0% of 31	0% of 31	48% of 31	26% of 31	0% of 31	74% of 31	
RATHER	0	5	47	28	7	0	0	54	5	0	59	87
	0% of 87	6% of 87	54% of 87	32% of 87	8% of 87	0% of 87	0% of 87	62% of 87	6% of 87	0% of 87	68% of 87	
POSS	0	0	37	185	14	0	0	51	0	0	51	236
	0% of 236	0% of 236	16% of 236	78% of 236	6% of 236	0% of 236	0% of 236	22% of 236	0% of 236	0% of 236	22% of 236	
NOT TECH	0	0	0	51	80	0	0	80	0	0	80	131
	0% of 131	0% of 131	0% of 131	39% of 131	61% of 131	0% of 131	0% of 131	61% of 131	0% of 131	0% of 131	61% of 131	
Totals	0	13	99	272	101	0	0	200	13	0	213	485
	0% of 485	3% of 485	20% of 485	56% of 485	21% of 485	0% of 485	0% of 485	41% of 485	3% of 485	0% of 485	44% of 485	

Table 7 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 5%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	1	1	5	0	0	0	1	1	0	2	7
	0% of 7	14% of 7	14% of 7	71% of 7	0% of 7	0% of 7	0% of 7	14% of 7	14% of 7	0% of 7	29% of 7	
RATHER	0	2	35	33	6	0	0	41	2	0	43	76
	0% of 76	3% of 76	46% of 76	43% of 76	8% of 76	0% of 76	0% of 76	54% of 76	3% of 76	0% of 76	57% of 76	
POSS	0	0	32	183	21	4	0	53	4	0	57	240
	0% of 240	0% of 240	13% of 240	76% of 240	9% of 240	2% of 240	0% of 240	22% of 240	2% of 240	0% of 240	24% of 240	
NOT TECH	0	0	0	59	101	2	0	101	2	0	103	162
	0% of 162	0% of 162	0% of 162	36% of 162	62% of 162	1% of 162	0% of 162	62% of 162	1% of 162	0% of 162	64% of 162	
Totals	0	3	68	280	128	6	0	196	9	0	205	485
	0% of 485	1% of 485	14% of 485	58% of 485	26% of 485	1% of 485	0% of 485	40% of 485	2% of 485	0% of 485	42% of 485	

Table 8 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 3% and 1.5%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	3	17	15	12	0	0	0	15	17	3	35	47
	6% of 47	36% of 47	32% of 47	26% of 47	0% of 47	0% of 47	0% of 47	32% of 47	36% of 47	6% of 47	74% of 47	
RATHER	0	4	58	28	3	0	0	61	4	0	65	93
	0% of 93	4% of 93	62% of 93	30% of 93	3% of 93	0% of 93	0% of 93	66% of 93	4% of 93	0% of 93	70% of 93	
POSS	0	0	36	168	13	0	0	49	0	0	49	217
	0% of 217	0% of 217	16% of 217	77% of 217	6% of 217	0% of 217	0% of 217	22% of 217	0% of 217	0% of 217	22% of 217	
NOT TECH	0	0	0	50	77	1	0	77	1	0	78	128
	0% of 128	0% of 128	0% of 128	39% of 128	60% of 128	1% of 128	0% of 128	60% of 128	1% of 128	0% of 128	61% of 128	
Totals	3	21	109	258	93	1	0	202	22	3	227	485
	1% of 485	4% of 485	22% of 485	53% of 485	19% of 485	0% of 485	0% of 485	42% of 485	4% of 485	1% of 485	47% of 485	

Table 9 - Results Analysis of QUIZ Chapters 1-6

Minimum Bounds = 0.75% and 0.375%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	0	1	9	0	0	0	1	0	0	1	10
	0% of 10	0% of 10	10% of 10	90% of 10	0% of 10	0% of 10	0% of 10	10% of 10	0% of 10	0% of 10	10% of 10	
RATHER	0	10	74	88	24	0	0	98	10	0	108	196
	0% of 196	5% of 196	38% of 196	45% of 196	12% of 196	0% of 196	0% of 196	50% of 196	5% of 196	0% of 196	55% of 196	
POSS	0	0	59	184	80	10	0	139	10	0	149	333
	0% of 333	0% of 333	18% of 333	55% of 333	24% of 333	3% of 333	0% of 333	42% of 333	3% of 333	0% of 333	45% of 333	
NOT TECH	0	0	0	53	53	11	7	53	11	7	71	124
	0% of 124	0% of 124	0% of 124	43% of 124	43% of 124	9% of 124	6% of 124	43% of 124	9% of 124	6% of 124	57% of 124	
Totals	0	10	134	334	157	21	7	291	31	7	329	663
	0% of 663	2% of 663	20% of 663	50% of 663	24% of 663	3% of 663	1% of 663	44% of 663	5% of 663	1% of 663	50% of 663	

Table 10 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 20% and 30%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	4	8	16	0	0	0	8	4	0	12	28
	0% of 28	14% of 28	28% of 28	57% of 28	0% of 28	0% of 28	0% of 28	28% of 28	14% of 28	0% of 28	43% of 28	
RATHER	0	12	83	95	20	0	0	103	12	0	115	210
	0% of 210	6% of 210	40% of 210	45% of 210	10% of 210	0% of 210	0% of 210	49% of 210	6% of 210	0% of 210	55% of 210	
POSS	0	0	58	174	69	7	0	127	7	0	134	308
	0% of 308	0% of 308	19% of 308	56% of 308	22% of 308	2% of 308	0% of 308	41% of 308	2% of 308	0% of 308	44% of 308	
NOT TECH	0	0	0	53	49	8	7	49	8	7	64	117
	0% of 117	0% of 117	0% of 117	45% of 117	42% of 117	7% of 117	6% of 117	42% of 117	7% of 117	6% of 117	55% of 117	
Totals	0	16	149	338	138	15	7	287	31	7	325	663
	0% of 663	2% of 663	22% of 663	51% of 663	21% of 663	2% of 663	1% of 663	43% of 663	5% of 663	1% of 663	49% of 663	

Table 11 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 5%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	12	14	16	0	0	0	14	12	0	26	42
	0% of 42	28% of 42	33% of 42	38% of 42	0% of 42	0% of 42	0% of 42	33% of 42	28% of 42	0% of 42	62% of 42	
RATHER	0	15	98	91	21	0	0	119	15	0	134	225
	0% of 225	7% of 225	44% of 225	40% of 225	9% of 225	0% of 225	0% of 225	53% of 225	7% of 225	0% of 225	60% of 225	
POSS	0	0	58	157	67	7	0	125	7	0	132	289
	0% of 289	0% of 289	20% of 289	54% of 289	23% of 289	2% of 289	0% of 289	43% of 289	2% of 289	0% of 289	46% of 289	
NOT TECH	0	0	0	50	42	7	7	42	7	7	56	106
	0% of 106	0% of 106	0% of 106	47% of 106	40% of 106	7% of 106	7% of 106	40% of 106	7% of 106	7% of 106	53% of 106	
Totals	0	27	170	314	130	14	7	300	41	7	348	662
	0% of 662	4% of 662	26% of 662	47% of 662	20% of 662	2% of 662	1% of 662	45% of 662	6% of 662	1% of 662	52% of 662	

Table 12 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 2.5%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	0	1	2	0	0	0	1	0	0	1	3
	0% of 3	0% of 3	33% of 3	67% of 3	0% of 3	0% of 3	0% of 3	33% of 3	0% of 3	0% of 3	33% of 3	
RATHER	0	10	69	81	23	0	0	92	10	0	102	183
	0% of 183	5% of 183	38% of 183	44% of 183	12% of 183	0% of 183	0% of 183	50% of 183	5% of 183	0% of 183	56% of 183	
POSS	0	0	59	182	84	17	0	143	17	0	160	342
	0% of 342	0% of 342	17% of 342	53% of 342	24% of 342	5% of 342	0% of 342	42% of 342	5% of 342	0% of 342	47% of 342	
NOT TECH	0	0	0	56	58	13	7	58	13	7	78	134
	0% of 134	0% of 134	0% of 134	42% of 134	43% of 134	10% of 134	5% of 134	43% of 134	10% of 134	5% of 134	58% of 134	
Totals	0	10	129	321	165	30	7	294	40	7	341	662
	0% of 662	2% of 662	19% of 662	48% of 662	25% of 662	5% of 662	1% of 662	44% of 662	6% of 662	1% of 662	52% of 662	

Table 13 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 3% and 1.5%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	2	26	32	29	0	0	0	32	26	2	60	89
	2% of 89	29% of 89	36% of 89	32% of 89	0% of 89	0% of 89	0% of 89	36% of 89	29% of 89	2% of 89	67% of 89	
RATHER	0	18	85	79	7	0	0	92	18	0	110	189
	0% of 189	10% of 189	45% of 189	42% of 189	4% of 189	0% of 189	0% of 189	49% of 189	10% of 189	0% of 189	58% of 189	
POSS	0	0	57	155	63	11	0	120	11	0	131	286
	0% of 286	0% of 286	20% of 286	54% of 286	22% of 286	4% of 286	0% of 286	42% of 286	4% of 286	0% of 286	46% of 286	
NOT TECH	0	0	0	46	44	8	1	44	8	1	53	99
	0% of 99	0% of 99	0% of 99	46% of 99	44% of 99	8% of 99	1% of 99	44% of 99	8% of 99	1% of 99	54% of 99	
Totals	2	44	174	309	114	19	1	288	63	3	354	663
	0% of 663	7% of 663	26% of 663	47% of 663	17% of 663	3% of 663	0% of 663	43% of 663	10% of 663	0% of 663	53% of 663	

Table 14 - Results Analysis of Delisle's Parser Text

Minimum Bounds = 0.75% and 0.375%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	5	0	7	0	0	0	0	5	0	5	12
	0% of 12	42% of 12	0% of 12	58% of 12	0% of 12	0% of 12	0% of 12	0% of 12	42% of 12	0% of 12	42% of 12	
PROB (RATHER)	0	0	8	1	1	0	0	9	0	0	9	10
	0% of 10	0% of 10	80% of 10	10% of 10	10% of 10	0% of 10	0% of 10	90% of 10	0% of 10	0% of 10	90% of 10	
POSS	0	0	6	22	5	0	0	11	0	0	11	33
	0% of 33	0% of 33	18% of 33	67% of 33	15% of 33	0% of 33	0% of 33	33% of 33	0% of 33	0% of 33	33% of 33	
NOT TECH	0	0	0	18	13	1	0	13	1	0	14	32
	0% of 32	0% of 32	0% of 32	56% of 32	41% of 32	3% of 32	0% of 32	41% of 32	3% of 32	0% of 32	44% of 32	
Totals	0	5	14	48	19	1	0	33	6	0	39	87
	0% of 87	6% of 87	16% of 87	55% of 87	22% of 87	1% of 87	0% of 87	38% of 87	7% of 87	0% of 87	45% of 87	

Table 15 - Results Analysis of QUIZ Chapters 1-2

Minimum Bounds = 3% and 1.5%
Term Percentage boundary = 20%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	0	13	3	6	0	0	0	3	13	0	16	22
	0% of 22	59% of 22	14% of 22	27% of 22	0% of 22	0% of 22	0% of 22	14% of 22	59% of 22	0% of 22	73% of 22	
RATHER	0	0	9	1	2	0	0	11	0	0	11	12
	0% of 12	0% of 12	75% of 12	8% of 12	17% of 12	0% of 12	0% of 12	92% of 12	0% of 12	0% of 12	92% of 12	
POSS	0	0	4	4	2	0	0	6	0	0	6	10
	0% of 10	0% of 10	40% of 10	40% of 10	20% of 10	0% of 10	0% of 10	60% of 10	0% of 10	0% of 10	60% of 10	
NOT TECH	0	0	0	0	0	0	0	0	0	0	0	0
	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	0% of 0	
Totals	0	13	16	11	4	0	0	20	13	0	33	44
	0% of 44	30% of 44	36% of 44	25% of 44	9% of 44	0% of 44	0% of 44	45% of 44	30% of 44	0% of 44	75% of 44	

Table 16 - Results Analysis of QUIZ Chapters 1-2

Minimum Bounds = 1.5% and 0.75%
Term Percentage boundary = 10%

Class'n Category	-3	-2	-1	OK	+1	+2	+3	Total +1 or -1	Total +2 or -2	Total +3 or -3	Total Mis-Class	Total NPs
TECH	1	12	3	7	0	0	0	3	12	1	16	23
	4% of 23	52% of 23	13% of 23	30% of 23	0% of 23	0% of 23	0% of 23	13% of 23	52% of 23	4% of 23	70% of 23	
RATHER	0	1	9	1	0	0	0	9	1	0	10	11
	0% of 11	9% of 11	82% of 11	9% of 11	0% of 11	0% of 11	0% of 11	82% of 11	9% of 11	0% of 11	91% of 11	
POSS	0	0	9	26	3	0	0	12	0	0	12	38
	0% of 38	0% of 38	24% of 38	68% of 38	8% of 38	0% of 38	0% of 38	32% of 38	0% of 38	0% of 38	32% of 38	
NOT TECH	0	0	0	18	8	0	0	8	0	0	8	26
	0% of 26	0% of 26	0% of 26	69% of 26	31% of 26	0% of 26	0% of 26	31% of 26	0% of 26	0% of 26	31% of 26	
Totals	1	13	21	52	11	0	0	32	13	1	46	98
	1% of 98	13% of 98	21% of 98	53% of 98	11% of 98	0% of 98	0% of 98	33% of 98	13% of 98	1% of 98	47% of 98	

Table 17 - Results Analysis of QUIZ Chapters 1-2 using QUIZ Chapters 1-6 Word Frequencies Classification by Sentence

Minimum Bounds = 0.5% and 0.25%
Term Percentage boundary = 100%

2.6 Limitations

There is no doubt that the most limiting assumption about AZTEC is its domain independence. Since this means that no semantic information can be used in the concept classification process, AZTEC is forced to rely on the frequencies of occurrence of words or NPs in the text, and I have demonstrated in this thesis that the optimal success rate of concept classification based on statistics does not exceed the 50% to 60% range.

Other limitations faced by AZTEC involve the two external systems on which AZTEC relies, namely the Text Base Builder (TBB) and the Sentence Parser. The TBB has at least one limitation which affects the ease with which a new text can be submitted to AZTEC: it is not able to recognize a paragraph of text if one of its lines has one or more blank spaces between the last character of the line and the carriage return. I have spent a considerable amount of time in an attempt to correct this problem, but without success. Therefore, the texts I used to test AZTEC had to be manually edited in order to remove any spaces at the end of each line, and whenever the user wants to use AZTEC on a new text, she will have to do the same, otherwise the Text Base will be missing a good portion of the text.

The Sentence Parser imposes limitations on AZTEC as well, due to the time it takes to parse most complex sentences. The introduction of a timer associated with each invocation of the parser (see section 2.3) has resulted in the exclusion of many NPs which could not be recognized by the parser in less than five seconds and were therefore over-looked in the Classification process. This limitation not only applies to each invocation of the parser but also to the alternative approach of examining each word of the sentence for NP start words, since for every start word found the NP Equivalent portion of the parser is invoked and has a timer associated with it as well. This may very well be one of the main sources of mis-classifications in Classification by Block, since term frequencies and

head noun frequencies are only accumulated for NPs which were recognized by the parser. However, efficiency improvements from version 2.5.1 of Quintus Prolog to version 3.1 have had a positive impact on this problem, in the sense that more of the parser rules can be tested before the five-second timer interrupts the parsing process. For example, Table 16, which depicts the success and failure rates of the optimal Classification by Block algorithm using chapters 1 and 2 of QUIZ, reflects the results obtained with Quintus Prolog 2.5.1, and only 44 distinct NPs (excluding the ones classified as technicality **Unknown**) were recognized. However, when the Classification by Sentence algorithm was more recently executed on the same text using Quintus Prolog 3.1, 98 distinct NPs were parsed and recognized (Table 17). It is therefore expected that with advances in the technology used by AZTEC, the Sentence Parser will prove to be less of a limitation.

2.7 Related Work

From what I could ascertain by reviewing the work related to knowledge acquisition from text using syntactic information and statistics, no previous attempts at classification by degree of technicality have been made, although some fields of research have been devoted to extracting semantic knowledge from text using only syntactic information and text statistics.

The clustering algorithm described in Hirschman et al. (1975) is an example of an attempt to extract semantic classes using syntactic information and text statistics, since its goal is to cluster together semantically related nouns and verbs using the rates of occurrence of parse tree operands used in similar lexical relations with the same operator (see section 2.5).

The method described by Hindle (1990) makes use of the same principles and attempts to obtain word clusters using a parser, rather than the manually produced parse trees of Hirschman et al., and

using an estimate of the *mutual information* measure (put forth in Fano (1961)) between two words to cluster together related words. The results obtained with this method are very promising, but they would not be of use to AZTEC any more than the results obtained using Hirschman et al.'s algorithm. Table 4 in Hindle (1990)[1] shows one cluster obtained using his method, and it links together the words *boat*, *ship*, *plane*, *bus*, *jet*, *vessel*, *truck*, *car*, *helicopter*, *ferry* and *man*. Although there may be domains in which all these terms belong to the same concept category of technicality, it is my belief that there are even more domains in which they do not. In addition, although Hindle's method is one of the few in the literature which use a parser to extract syntactic relations, there is no mention of the amount of time needed to parse the six million word sample of news stories from the Associated Press.

Church and Hanks (1990) also use the measure of mutual information between two words to quantify word associations. Their goal is to automatically create a word association table, listing for example all the objects occurring with a particular verb. The purpose of this table is to assist a user in organizing a concordance.

There has been similar work carried out in an attempt to acquire other types of semantic relations from text using syntactic information, although some of these methods use pre-existing semantic knowledge. Grishman et al. (1986), for example, present a scheme for acquiring *selectional constraints* from a sublanguage text. The selectional constraints acquired indicate with which verb, for example, the nouns of a semantic class can be used as object or subject. The goal of these constraints is to assist a parser in selecting among multiple parses when another text of the same sublanguage is parsed. The method used is based on Hirschman et al.'s clustering algorithm.

Zernik and Jacobs (1990) attempt to acquire thematic roles from text. The authors use an existing corpus, where each word is tagged with its possible syntactic categories, as a training set to be

---

[1]Hindle, "Noun Classification," 272.

used in tagging subsequent texts. Once the words of a new text have been tagged, lexical relations (e.g. subject-verb, verb-object) are inferred, based on the relative positions of the words. Then, based on pre-existing knowledge of the thematic roles required by each verb, it is possible to determine which object of the verb belongs to what thematic role, for example which object is the *recipient* of the verb *pay*.

Anick and Pustejovsky (1990) make use of the *qualia structures* described in Pustejovsky (1989), which consist of structured representations of nouns containing knowledge about their constituents, their structures and their functions. The goal of this research is to augment the knowledge found in the hand-coded qualia structures with knowledge extracted from the text, by detecting relations between the words of the text, such as ISA relations.

The research of Hayes et al. (1988) also involves classification, but their goal is quite different. They aim to classify news stories into one or more of six broad categories, such as acquisition/mergers, metals, war, and so on, by using the pattern-matching rules in their Knowledge Base to detect key words in the text of a story and correspondingly assigning the story to a category.

Another field of research which seems related to AZTEC is the area of Information Retrieval, more specifically the work achieved in automated book indexing, where the concepts found in texts are examined so that an index can be created for that text. One early attempt, found in Borko (1970), attempted to extract terms from a book by considering pairs of content words. Unfortunately, the system could not automatically eliminate many of the non-meaningful terms, and user intervention was required.

Dillon and Federhart (1982) first attempted to use discriminant analysis, which uses a text-specific discriminant function, to separate topic-related terms from general ones. Later, Dillon and McDonald (1983) were more successful at grouping together related terms based on a Measure of Association (MA) between pairs of multi-word

concepts. This MA is based on the number of occurrences of each word stem in different concepts and is used to group together concepts containing at least one common stem.

The Information Retrieval paper which is the most relevant to AZTEC, Salton (1988), aims to build an index using the NPs extracted from a parsed text. Each NP is assigned a weight, based on its *term frequency* in the text and the inverse of its *document frequency* which is a measure of the number of documents in which the NP occurs. "The best constructs for indexing purposes are those exhibiting a high term frequency, and a relatively low overall document frequency. Such constructs will distinguish the documents, or document sections, to which they are assigned from the remainder of the collection."[1] Salton retains all NPs whose weight is above a certain threshold as part of the index. Given that AZTEC only considers one document at a time, Salton's scheme is consistent with AZTEC's use of the term frequency to assign a weight to each NP.

Another related paper, Luhn (1958), describes an interesting attempt at automatically creating text abstracts, where each sentence of a text is assigned a weight which is based on its number of *significant words*. Significant words are those which do not occur too frequently or too infrequently, excluding "common words," such as prepositions, pronouns and articles. The sentences whose weights exceeds a certain threshold are retained for the abstract.

Maarek et al. (1991) use similar Information Retrieval techniques to create an index which will allow them to browse through and retrieve components from software libraries. The authors use the *quantity of information* measure between two open-class words, which is based on the probability of occurrence of the two words in the text, to compute their *resolving power*, which is based on the number of occurrences of the two words together and

---

[1]Gerard Salton, "Syntactic Approaches to Automatic Book Indexing," Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (1988): 206.

their quantity of information. The pairs of words whose resolving power exceeds a certain threshold are retained for the index.

Of all the articles found, only one is related to the same ultimate goal as AZTEC, namely to assist in the creation of a database schema, though not using free text. Lanka (1985) uses the typical natural language queries to be input to a database to automatically infer the Entity-Relationship diagram of the database. The author relies on the fact that the nouns in the queries refer to entities and that the verbs depict relationships between these entities.

2.8 Future Directions

One of the future directions of AZTEC, and perhaps the most crucial, should be its disassociation from the two external systems which impose some limitations upon it. Unfortunately, there is at present no practical alternative to the use of the Text Base Builder (TBB), which is needed to convert a mostly un-edited text into a file of individual sentences. It should, however, be possible to find an alternative to parsing each sentence in the text in order to reduce the computation time and to increase the number of NPs considered in the classification process. Salton (1986) outlines three methods of extracting concepts from text.

The first method consists in using statistical methods to find the heads of phrases (for example, the head noun of a NP) and combining them with each phrase component (for example, the modifiers of the head noun), where there is less than a stated number of words between the phrase head and the modifier. This approach is similar to the collocational approach used by Church and Hanks (1990), Zernik and Jacobs (1990), and Maarek et al. (1991). The second method proposed by Salton consists in using "a simple syntactic pattern matching method where . . . phrases are then recognized as sequences of words exhibiting certain previously established patterns of syntactic indications (e.g. adjective-noun-

noun, or preposition-adjective-noun)."[1] Dillon and McDonald (1983) use these sequences of syntactic categories, called *concept rules*, and match their tagged text against them to extract the sequences of words they need. The third method consists in using a parser to extract the syntactic structure of each sentence. Parsers were used by Salton (1988) and Hindle (1990), although neither of them indicated the processing time required.

Unfortunately, AZTEC has no access to a large tagged corpus to be used as a training set, so tagging a new text and expecting to simply extract all sequences of nouns preceded by an adjective equivalent or a noun equivalent is not realistic. Given that each word of a text is likely to have at least two possible syntactic categories, the high level of ambiguity introduced by using the two approaches based on extracting collocational words of given syntactic categories would certainly result in the selection of many concepts which are not really NPs in the structure of the sentence. However, since parsing has proven to be very costly, the acquisition of a tagged corpus should be given further consideration.

Another major enhancement to AZTEC involves designing a means to extract the relationships between the concepts of a technical text and possibly rank them by degree of technicality. This could be accomplished by examining the verbs in the text and approximating a relationship category based on frequencies of occurrence and dictionary flags.

Another enhancement to be considered should be the approximation of functions to dynamically compute the percentages to be used to calculate the two Minimum Bounds and the percentage used as the Term Percentage boundary. As outlined in section 2.5, all three boundaries seem to be inversely proportional to the number of NPs in the text. It may be possible to approximate these functions by running AZTEC using several more texts of various lengths,

---

[1]Gerard Salton, "On the Use of Term Associations in Automatic Information Retrieval," Proceedings of the 11th International Conference on Computational Linguistics COLING-86 (1986): 383.

examining the classification results obtained, comparing them against the expected, and determining the optimal Minimum Bounds and Term Percentage boundary. Along the same lines, future enhancements could also include further refining the classification process to minimize the mis-classification rates, especially in the **Not Technical** category.

In Salton (1988), the author uses a parser to extract NPs from a text from which an index is to be automatically built. However, in addition to considering NPs constructed of a head noun and its last modifier, Salton constructs more NPs by combining the head noun with each of its modifiers (e.g. the NP *rabid black dog* would become the two NPs *rabid dog* and *black dog*), and by combining the head noun with the head of each Prepositional Phrase (PP) modifying it (e.g. *methods for computation* becomes *computation methods*, and *subset of medical literature* becomes *literature subset* and *medical literature*).

The creation of additional NPs of this kind could easily be incorporated into AZTEC, although there are cases in which the resulting NPs may not be valid. For example, the NPs of the sentence *There are a number of efficient parsing techniques with known complexity for programming languages*[1] would be converted into the following NPs: *techniques number*, *efficient techniques*, *parsing techniques*, *complexity techniques*, *known complexity*, *languages complexity*, and *programming languages*. Although combining the head nouns with each of their modifiers results in valid NPs, that is not always the case with NPs constructed with a head noun and the head of each PP modifying it. The NPs *techniques number* and *complexity techniques* do not adequately reflect the meaning intended in the sentence. This enhancement to AZTEC should therefore be carefully studied before being considered.

Another enhancement which could be made to AZTEC involves the presentation of the concept classifications once they have been computed. The user should have the option of traversing the list of classified concepts from the most technical category to the least

[1]Delisle, Parser for Technical Texts, 2.

technical and indicating whether or not each concept should be retained to assist in the modeling of the domain described in the text. AZTEC could save the approved concepts in a separate list for the user to refer to at a later time or for use in processing another text from a similar domain.

Chapter 3
Syntactic Category Disambiguation

The goal of the Syntactic Category ANalyzer (SCAN) module is to process a sentence before the Sentence Parser has been invoked, in order to verify that each word of the sentence is defined in the Word Dictionary used by the Parser. If a word is not defined, i.e. if it is an *unknown word*, SCAN will attempt to approximate its syntactic category. It will use the syntactic categories of the preceding and following words and match them and the unknown word against a context-free grammar of phrases.

This chapter will describe the SCAN module in detail, from the assumptions guiding its design and development to the three methods used to find the syntactic category of an unknown word: the Chart Parser approach and two methods based on the Collocational Probability Matrix (CPM) algorithm described in an article by Steven J. DeRose (1988). The results obtained with SCAN will be discussed, as will its limitations, and an overview of the related work in the area of syntactic category disambiguation will be given.

3.1 Main Assumptions

SCAN is subject to the same basic assumptions as AZTEC, namely that no domain or semantic knowledge can be used due to the domain independent nature of AZTEC, and that user interaction should be kept to a minimum.

I have observed in the literature that existing syntactic category disambiguation methodologies which make use of text statistics, like the algorithms described by DeRose, generally rely on statistics gathered on a very large tagged corpus, which are then

used to process all subsequent texts. Unfortunately, AZTEC does not have access to such a large corpus, let alone one which is tagged with the syntactic category of each word. Therefore, any text statistics used by SCAN must be accumulated using the source text whose NPs are to be classified.

Another assumption involves the completeness of the Word Dictionary. It is assumed that all the function words (prepositions, conjunctions, pronouns, and so on) already appear in the Word Dictionary, so that only the four open-class categories (noun, verb, adjective and adverb) are considered as the possible syntactic categories of an unknown word. Noun equivalents include count nouns, mass nouns, proper nouns and the present participle form of a verb, and adjective equivalents include adjectives, the past participle and the present participle forms of a verb.

3.2 The Chart Parser

Because text statistics were not available, I first decided to use a Chart Parser to parse a sentence segment containing the unknown word, using a grammar of phrases with 64 rules. The rationale for the use of parsing is that other methodologies such as morphological analysis (for example, examining the suffix of a word to determine if it is an adverb ending with *-ly*) often only guess at an unknown word's syntactic category, whereas parsing will match a sequence of categories to an accurate grammar of English and so will be more likely to produce the correct syntactic category. The decision to use a Chart Parser was based on the fact that it is not necessary to parse an entire sentence or a syntactic unit (such as a Noun Phrase or a Verb Phrase). Rather, we want to parse a sentence segment which may or may not contain a complete syntactic unit. For this reason, I adopted a Chart Parser[1] using bottom-up rule invocation with top-

[1]Allen, 60-65.

down filtering,[1] rather than an exclusively top-down approach. The combination of both types of rule-invocation strategies is intended to minimize the disadvantages of each method. Similar techniques have been used by other researchers.[2]

3.2.1 Extracting a Context

The *Create CPM* module is invoked from the Classification by Block algorithm before the first call to SCAN, in order to set up a database of text statistics, which is known as the Collocational Probability Matrix (see Figure 10). Once that has been done, the *Chart Parser* module is invoked with each sentence processed by Classification by Block, in order to verify that all of its words appear in the Sentence Parser's Word Dictionary. Unfortunately, the two CPM methods cannot be used in conjunction with Classification by Sentence, since no block of text is available for the creation of the CPM, and it would be pointless to gather statistics on a single sentence. As a result, the Classification by Sentence algorithm does not invoke the Create CPM module.

When a sentence is traversed, each unknown word is processed individually (see Figure 11). For efficiency purposes, it was important that only a segment of the sentence, rather than the entire sentence, be parsed using the Chart Parser. For that reason, each time an unknown word is encountered, a *context* is created, consisting of some words to the left of the unknown word, the unknown word itself, and one word to the right of the unknown word. Since Classification by Sentence cannot resort to the CPM approach, if no syntactic category can be found for the unknown word and approved by the user, subsequent attempts at chart parsing are made by increasing, in a loop, the number of words to the left of the unknown

---

[1]Mats Wiren, "A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing," Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics (1987): 226-233.

[2]Chris S. Mellish, "Some Chart-Based Techniques for Parsing Ill-Formed Input," Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (1989): 102-109.

word in the context, which begins at two words and ends at a phrase boundary, such as a preposition, an article, the word "that", a question word, a possessive adjective, a conjunction, punctuation, or another unknown word. The context to the right-hand side of the unknown word is limited to one word for efficiency purposes, because tests using a variable number of words to the right of the unknown word often took more than 60 seconds for each unknown word. The only exception to the use of a single word in the right-hand side context is when the unknown word is the first word of the sentence and can therefore have no left-hand side context. In this case, the limit on the number of words normally applied to a left-hand side context is applied to the right of the unknown word.

For example, suppose we had the sentence *The umpah pressed grapefruits*, where *the* is an article, *umpah* is the unknown word, *pressed* can be a verb or an adjective equivalent, and *grapefruits* is a noun. Since the *Create Context* module cannot extract a two-word context to the left of the unknown word, it only retains one word, *the*. The right-hand side context will consist of one word following the unknown word, *pressed*. As a result, the complete context would represent the sentence segment *the umpah pressed*, and it would consist of the following syntactic categories: [[art], [no_syncats], [verb, adj]], where art symbolizes the article *the*, no_syncats represents the unknown word *umpah*, and the categories verb and adjective equivalent are associated with *pressed*.

Once the context has been determined, the *Search Grammar* module is invoked to scan the grammar rules in order to find one or more syntactic category matches for the unknown word in the context. If the user opted for Classification by Sentence, this list of syntactic category matches is presented to the user. In Classification by Block, if Search Grammar returns more than one syntactic category, the CPM algorithm is invoked to further rank the best syntactic categories.

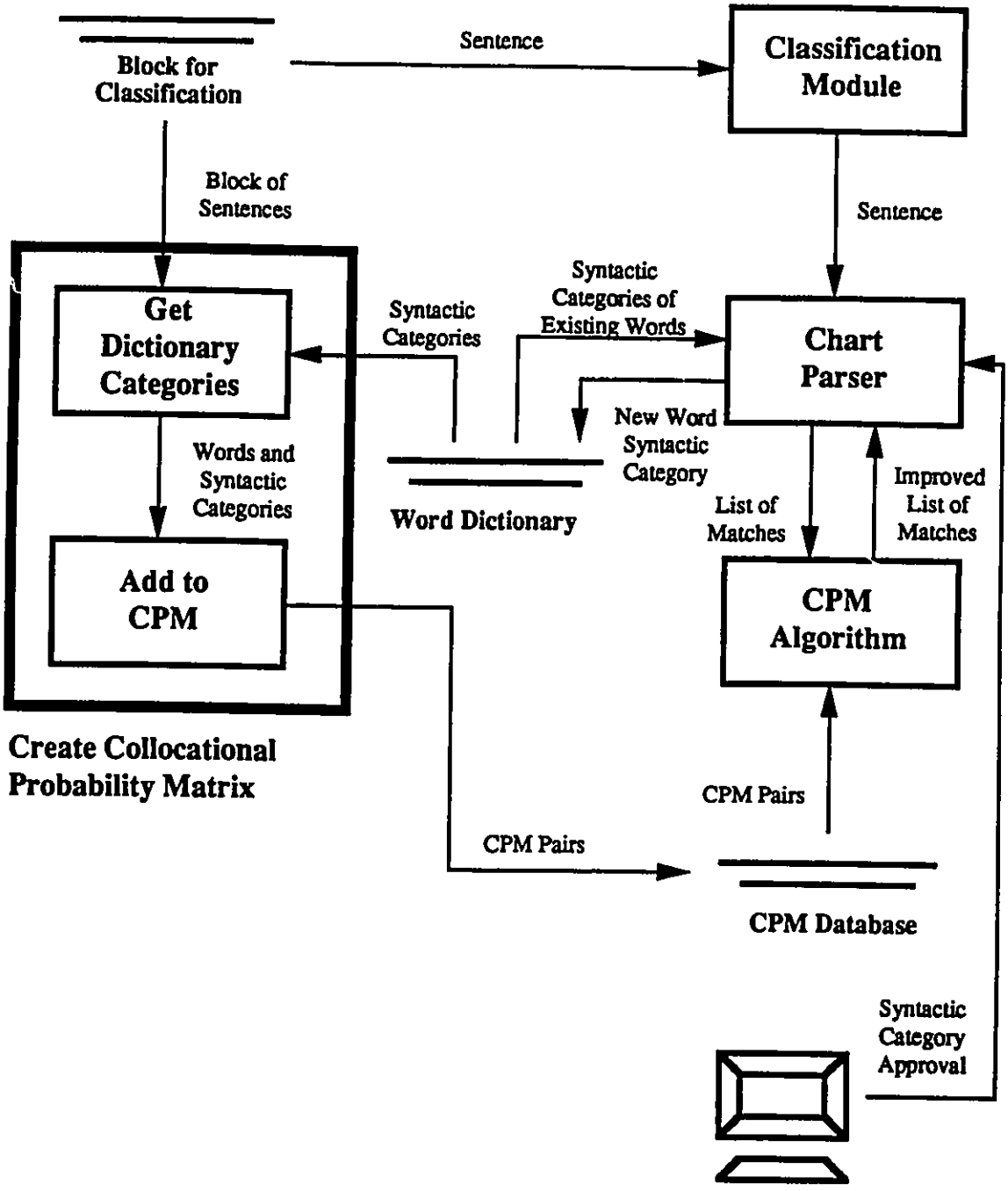


Figure 10 - SCAN Module

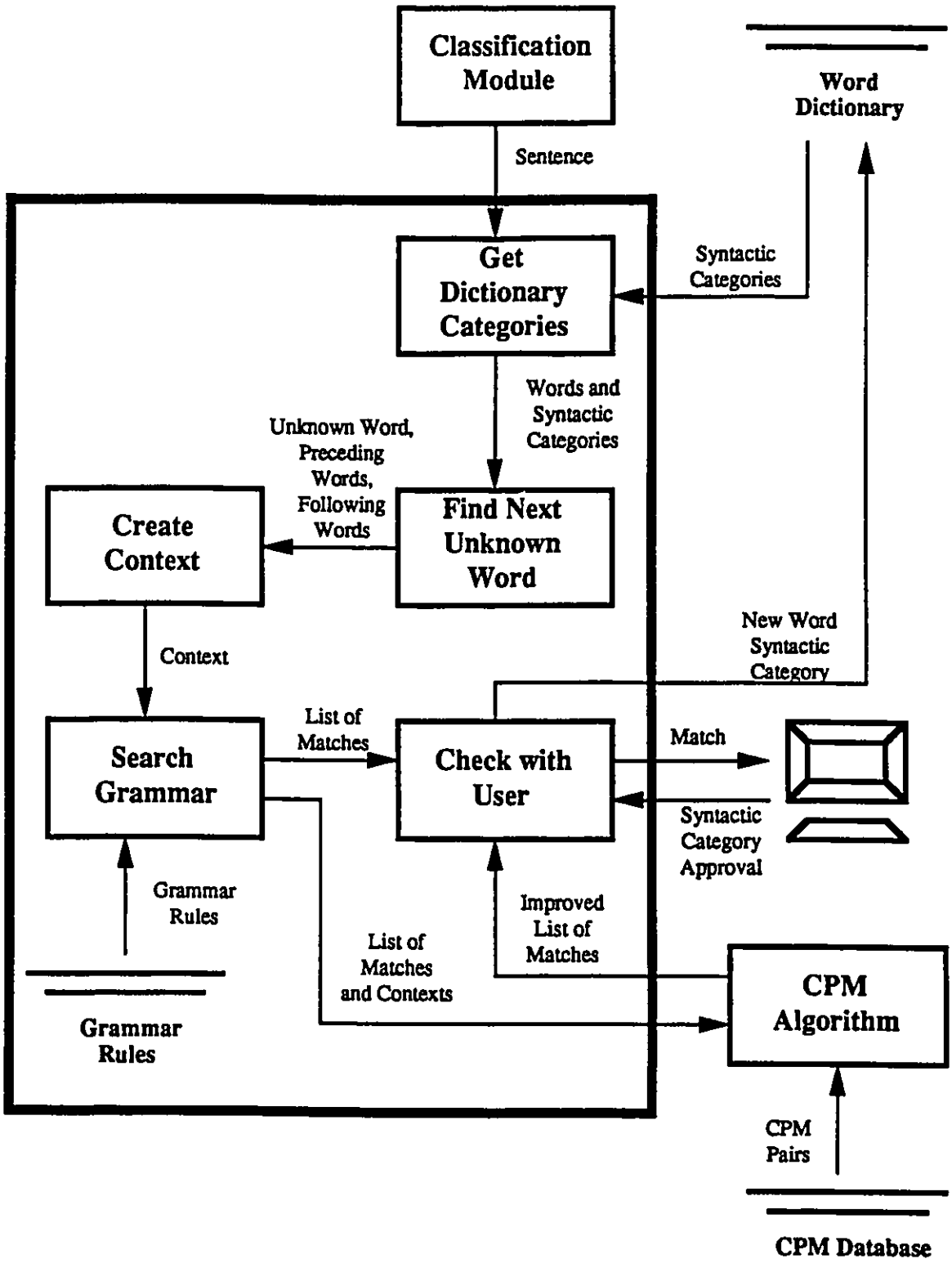


Figure 11 - Chart Parser Module

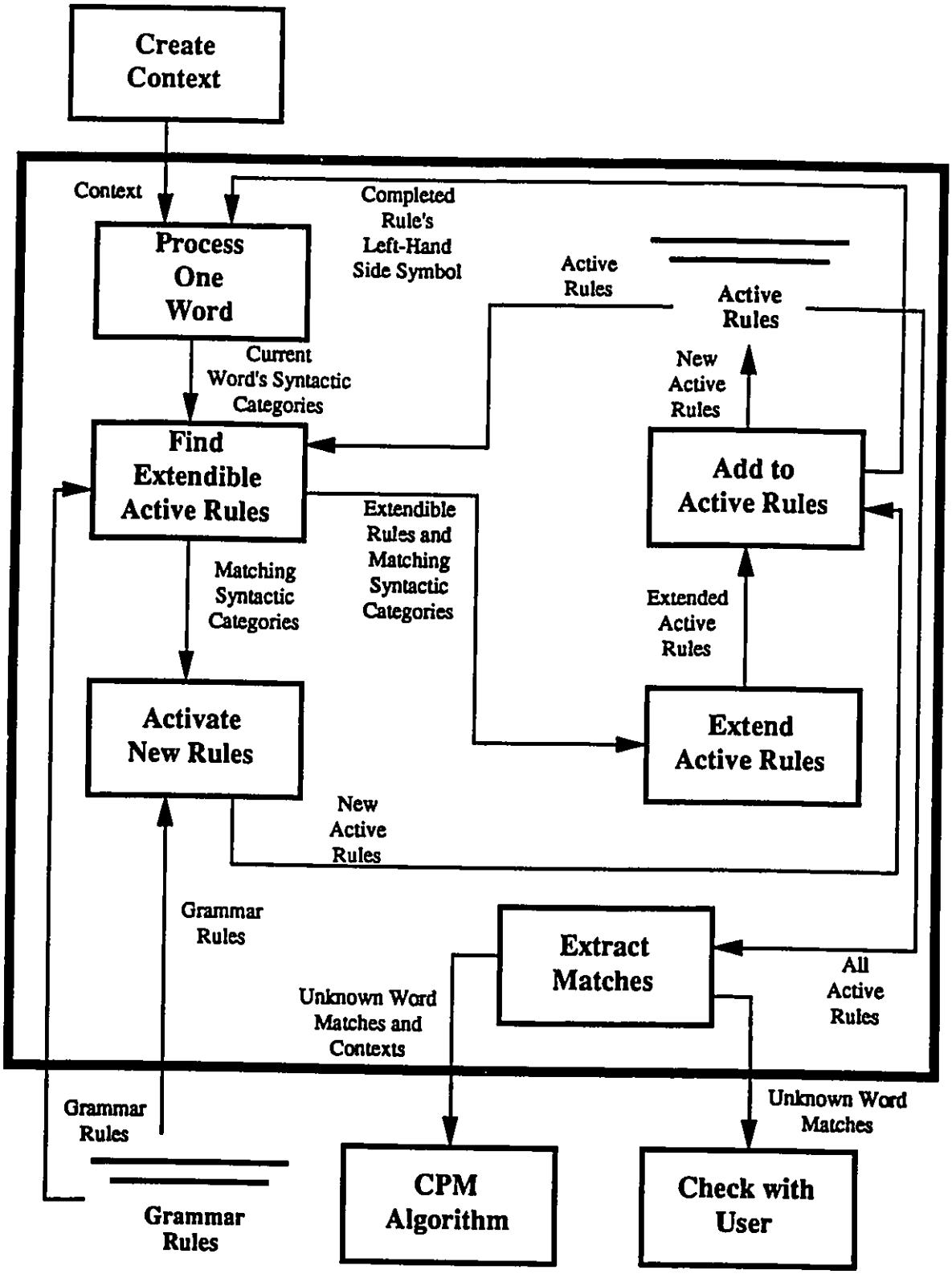


Figure 12 - Search Grammar Module

3.2.2 The Grammar Rules and the Active Rules

In order to match an unknown word to one or more syntactic categories, the entire context is matched against the rules of the grammar. The following grammar rules will be used in this section as an example:

(1) S <-- NP VP
(2) NP <-- art AP
(3) NP <-- art AP noun
(4) NP <-- art noun
(5) VP <-- verb
(6) AP <-- adj
(7) AP <-- adj AP

The information stored by SCAN for each grammar rule consists of a unique rule number, the left-hand side symbol of the rule (for example S, NP, VP and AP), and a list of the right-hand side symbols of the rule.

Throughout the syntactic category disambiguation process, the grammar rules must also be kept in memory in another form, as *active rules*, to reflect the current status of the parsing of the context. Each active rule is associated with one grammar rule, although several active rules could reflect the status of the same grammar rule applied to different portions of the context. An active rule has the following information stored with it in memory:

- the rule number of the grammar rule with which it is associated (for example, if we parsed the first word of the context *the umpah pressed* using the grammar in this section, grammar rules #2, #3 and #4 would be added to the active rules, along with the information below)
- the position of the word in the context at which the coverage of the rule begins (in our example, the coverage of rules #2, #3 and #4 begins at the first context word, *the*)

- the position of the word in the context at which the coverage of the rule currently ends (the coverage of rules #2, #3 and #4 currently ends between the first and the second context words)
- the position within the grammar rule up to which the context has been parsed (since *art* is the only symbol on the right-hand side of rules #2, #3 and #4 matched to the context, the current positions within the grammar rules follow this symbol)
- a list of the terminal symbols with which each word of the context has been associated according to the current rule (in this case, only the first right-hand side symbol of rules #2, #3 and #4, *art*, has been matched to the context).

The position of each word in the context and the position of each right-hand side symbol in a grammar rule are interpreted as follows. Position 0 is located just before the first word or right-hand side symbol, position 1 between the first and the second word or right-hand side symbol, position 2 between the second and third word or right-hand side symbol, and so on. The context used in the above example could therefore be re-written as [0 [art] 1 [no_syncats] 2 [verb, adj] 3], and the grammar rules as:

(1) S <-- 0 NP 1 VP 2
(2) NP <-- 0 art 1 AP 2
(3) NP <-- 0 art 1 AP 2 noun 3
(4) NP <-- 0 art 1 noun 2
(5) VP <-- 0 verb 1
(6) AP <-- 0 adj 1
(7) AP <-- 0 adj 1 AP 2

For example, since *the* is the first word in the context, its syntactic category is located between positions 0 and 1 of the context.

Similarly, an AP is expected between positions 1 and 2 of grammar rules #2 and #3.

3.2.3 Searching the Grammar

For each word in the context, the module *Find Extendible Active Rules* (Figure 12) finds active rules whose coverage of the context ends just before the current context word. It then checks whether any of those rules can be extended to include a syntactic category of the current word. Next, for each of those syntactic categories, two steps are followed. First, the module *Activate New Rules* activates (i.e. adds to the list of active rules) the grammar rules whose first right-hand side symbol matches the current syntactic category of the current context word. In the second step, the module *Extend Active Rules* extends the extendible active rules to include the current word. Top-down filtering reduces the number of active rules produced in both steps, because only the categories to which at least one active rule can be extended are considered in the module Activate New Rules, and only the active rules which expect one of the syntactic categories of the current word as their next right-hand side symbol are extended in the module Extend Active Rules.

Let us use the example of the previous sections, where we have the context [0 [art] 1 [no_syncats] 2 [verb, adj] 3] and the grammar defined above. Since no active rules exist yet, the first context word's syntactic category (*art*) is considered. In the first step, grammar rules #2, #3 and #4 are activated, since their first right-hand side symbol is an article. No active rules can be extended, so the second step does not result in any additions to the Active Rule Database. In summary:

Current Context Word: *the*
Syntactic Category: art
Context Position: (0, 1)
Extendible Rules: -

New Rules: 2, 0, 1, 1, [art]
3, 0, 1, 1, [art]
4, 0, 1, 1, [art]
Extended Rules: -

As mentioned above, the first argument of each active rule is the number of the grammar rule with which it is associated, in this case rules #2, #3 and #4. The second and third arguments represent the start and end positions within the context to which the rule applies; in this case, only the first word *the* has been parsed, and it appears between context positions 0 and 1. The fourth argument represents the current position within the grammar rule up to which the context has been parsed; in the example above, only the first right-hand side symbol, *art*, of rules #2, #3 and #4 has been parsed, so the current position within each of these grammar rules is located just after the first right-hand side symbol at position 1. The fifth argument of the active rules indicates the syntactic categories matched by the rule.

In the example above, since both steps have been carried out using the first word of the context, the second word *umpah* is processed next. Because it is the unknown word, it is assigned all four possible syntactic categories. The parse which covers the greatest number of words in the context once the entire context has been processed will reveal the most likely syntactic category of the unknown word. This way, the unknown word problem becomes a syntactic category disambiguation problem, and the methods described by DeRose can be used. The order in which the categories are processed does not affect the final Active Rule Database, so the order used is the following: adjective equivalent, noun equivalent, verb and adverb.

The module Find Extendible Active Rules determines that active rules #2 and #3 can indirectly be extended to an adjective equivalent and active rule #4 to a noun equivalent. As a result, the verb and adverb categories are not considered as possible matches. Beginning with the adjective equivalent category, two new rules (#6 and #7) are activated in the first step, and in the second step no existing rule can be extended, since active rules #2 and #3 do not directly expect an adjective equivalent as their next right-hand side symbol after the article.

Current Context Word: *umpah*
Syntactic Category: adj
Context Position: (1, 2)
Extendible Rules: 2, 0, 1, 1, [art]
3, 0, 1, 1, [art]

New Rules: 6, 1, 2, 1, [adj] * AP
7, 1, 2, 1, [adj]
Extended Rules: -

At this point, rule #6, which only expects one adjective equivalent, has been completed. This means that not only do we have an adjective equivalent between context positions 1 and 2, but according to rule #6 we also have an AP. The asterisk used in this section will indicate that a rule has been completed, and the symbol next to it will indicate the non-terminal symbol which has been parsed.

Next, rather than processing the remaining possible syntactic categories of *umpah*, the left-hand side symbol of the newly completed rule (AP) is processed. Only active rules #2 and #3 can be extended to an AP, since they both currently end their coverage at context position 1, which is the word at which the new AP begins, and both rules expect an AP as their next right-hand side symbol. In the first step, since no grammar rules have an AP as their first right-

hand side symbol, no new rules are activated. In the second step, however, both active rules #2 and #3 are extended.

Current Context Word: *umpah*
Symbol: AP
Context Position: (1, 2)
Extendible Rules: 2, 0, 1, 1, [art]
3, 0, 1, 1, [art]

New Rules: -
Extended Rules: 2, 0, 2, 2, [art, adj] * NP
3, 0, 2, 2, [art, adj]

We have now completed rule #2, and we have parsed a NP between positions 0 and 2 of the context. This NP is the next symbol which will be processed. In the first step, only rule #1 is added to the Active Rule Database, and in the second step no active rules can be extended to include a NP at context position 0.

Current Context Words: *the umpah*
Symbol: NP
Context Position: (0, 2)
Extendible Rules: -

New Rules: 1, 0, 2, 1, [art, adj]
Extended Rules: -

Now the next possible syntactic category for *umpah*, noun equivalent, is processed. Since no grammar rules begin with a noun, no new rules can be activated, but rule #4, which parsed an article between context positions 0 and 1, expects a noun equivalent as its next right-hand side symbol. Rule #4 is therefore extended.

Current Context Word: *umpah*
Syntactic Category: noun
Context Position: (1, 2)
Extendible Rules: 4, 0, 1, 1, [art]

New Rules: -
Extended Rules: 4, 0, 2, 2, [art, noun] * NP

Since rule #4 has been completed and a NP has been parsed, this symbol is processed next. As a result, rule #1 is added to the Active Rule Database.

Current Context Words: *the umpah*
Symbol: NP
Context Position: (0, 2)
Extendible Rules: -

New Rules: 1, 0, 2, 1, [art, noun]
Extended Rules: -

When each of the current context word's syntactic categories has been processed, the next word, in this case *pressed*, is considered. Since *pressed* has two possible syntactic categories associated with it, verb and adjective equivalent, and since there exist active rules which can be extended to both syntactic categories, each will be examined individually, beginning with verb. The Active Rule Database will be modified as follows.

Current Context Word: *pressed*
Syntactic Category: verb
Context Position: (2, 3)
Extendible Rules: 1, 0, 2, 1, [art, adj]
1, 0, 2, 1, [art, noun]

New Rules: 5, 2, 3, 1, [verb] * VP
Extended Rules: -

Current Context Word: *pressed*
Symbol: VP
Context Position: (2, 3)
Extendible Rules: 1, 0, 2, 1, [art, adj]
1, 0, 2, 1, [art, noun]

New Rules: -
Extended Rules: 1, 0, 3, 2, [art, adj, verb] *S
1, 0, 3, 2, [art, noun, verb] *S

Because S is the start symbol of the grammar, i.e. the top-level symbol, it does not appear among the right-hand side symbols of any grammar rule, so no further processing of S is required.

By treating *pressed* as an adjective equivalent, the following active rules will be added to the Active Rule Database.

Current Context Word: *pressed*
Syntactic Category: adj
Context Position: (2, 3)
Extendible Rules: 7, 1, 2, 1, [adj]

New Rules: 6, 2, 3, 1, [adj] * AP
7, 2, 3, 1, [adj]
Extended Rules: -

Current Context Word: *pressed*
Symbol: AP
Context Position: (2, 3)
Extendible Rules: 7, 1, 2, 1, [adj]

New Rules: -
Extended Rules: 7, 1, 3, 2, [adj, adj] *AP

Current Context Words: *umpah pressed*
Symbol: AP
Context Position: (1, 3)
Extendible Rules: 2, 0, 1, 1, [art]
3, 0, 1, 1, [art]

New Rules: -
Extended Rules: 2, 0, 3, 2, [art, adj, adj] *NP
3, 0, 3, 2, [art, adj, adj]

Current Context Words: *the umpah pressed*
Symbol: NP
Context Position: (0, 3)
Extendible Rules: -

New Rules: 1, 0, 3, 1, [art, adj, adj]
Extended Rules: -

Next, because every syntactic category of each context word has been considered and matched against the grammar rules, the final Active Rule Database is searched for possible matches for the unknown word.

3.2.4 Extracting the Matches

Once every word of the context has been processed, every possible parse of the context can be found in the active rules. The possible syntactic category matches for the unknown word can be extracted from the active rules, along with the number of context words matched by each rule. This number will be used to rate the possible syntactic categories with respect to each other, because it is assumed that a rule matching all the words of the context is more likely to yield the correct syntactic category of the unknown word than a rule which only matches one or two of the context words.

As a result, every active rule which contains the unknown word, and therefore every rule which covers context positions 1 and 2, is examined. The following matches are extracted from the rules, along with the number of context words matched:

adjective equivalent, 1	from rules #6, #7
adjective equivalent, 2	from rules #2, #3, #1, #7
noun equivalent, 2	from rules #4, #1
adjective equivalent, 3	from rules #1, #2, #3
noun equivalent, 3	from rule #1

Only the longest matches are retained for each possible syntactic category, so in the decreasing order of the length of the context segment matched, we obtain the following matches for *umpah*, along with their rating (the number of context words matched), using SCAN's Chart Parser method:

adjective equivalent	3
noun equivalent	3
verb	0
adverb	0

3.3 The Collocational Probability Matrix (CPM) Algorithm

Although the Chart Parser approach has yielded some good results for the sentence *The umpah pressed grapefruits* by narrowing down the possible syntactic categories of *umpah* to adjective equivalent or noun equivalent, it is unable to select the better of the two. The previous example was somewhat simple, in the sense that the Chart Parser narrowed down the possibilities to two syntactic categories, but there are many cases where none of the four syntactic categories can be eliminated. The string *the unfortunate umpah pressed grapefruits*, for example, if parsed using a full grammar of English, would result in the four possible syntactic categories being chosen as matches with equal likelihood. That is because *umpah* could be a noun equivalent, as in *the unfortunate* workers *pressed grapefruits*, an adjective equivalent, as in *the unfortunate* poor *pressed grapefruits*, a verb, as in *the unfortunate* ate *pressed grapefruits*, or an adverb, as in *the unfortunate* eagerly *pressed grapefruits*.

Because of such ambiguities, an additional method was required to further discriminate between the best syntactic categories chosen by the Chart Parser. In his paper, Steven J. DeRose describes algorithms which disambiguate the syntactic categories of words in large texts, in the sense that they automatically determine the best syntactic category of each ambiguous word in the text. An ambiguous word is one which can have more than one syntactic category, such as *pressed* in the example above, which may be a verb in the past tense or the past participle form of a verb used as an adjective equivalent. The systems described by DeRose are not meant to tag unknown words. However, one of our assumptions dictates that an unknown word is restricted to one of the four open-classes of words, so that it is possible to assign the four syntactic categories to each unknown word and have a syntactic category disambiguator choose the best one. Using the results of the Chart Parser, SCAN's

implementation of the CPM algorithm first uses values found in the Collocational Probability Matrix to rate each of the longest Chart Parser matches. If this scheme still does not select one syntactic category above the others, a longer sentence segment is considered, and a *span network* such as the one described in Marshall (1983) and DeRose (1988) is constructed in order to compute the syntactic category with the maximum probability of occurrence, using the values found in the CPM.

3.3.1 Creating the Collocational Probability Matrix

Both strategies mentioned above require the existence of the Collocation Probability Matrix (CPM), which was first put forth by Marshall (1983): "the main innovation of CLAWS is the use of a matrix of **collocational probabilities**, indicating the relative likelihood of co-occurrence of all ordered pairs of tags. This matrix can be mechanically derived from any pre-tagged corpus."[1]

In order to obtain the most accurate results possible, and since no pre-tagged corpus was readily available, I chose to use the source text to collect the statistics for the CPM, and I decided that SCAN's CPM should only contain statistics on pairs of unambiguous collocational words. This means that if one or both of the words being considered as a collocational pair have more than one possible syntactic category associated with them, the pair is not added to the CPM. The rationale behind this is that pairs made up of one or two ambiguous words will increase the number of occurrences of erroneous pairs, which will render the statistics in the CPM less accurate for the subsequent disambiguation of words.

The *Create CPM* module (Figure 10) takes each sentence of the Block for Classification, examines each pair of adjacent words, and if both words are unambiguously defined in the Word Dictionary, the number of occurrences of their collocational syntactic categories is incremented in the CPM. For example, if we had the sentence *The unfortunate workers pressed grapefruits*, where the possible

[1]DeRose, 33.

syntactic categories of each word are the following [[art], [adj], [noun], [verb, adj], [noun]], one art-adj pair and one adj-noun pair would be added to the CPM. However, because *pressed* is ambiguous, it does not feature in any pair. I also added two artificial syntactic categories, *start* and *end*, which are paired with the first and the last word respectively of the sentence if these are unambiguous. These symbols will serve to further assist in the disambiguation process, since the same syntactic categories tend to occur as the first and last words of a sentence. As a result, in the above example, the number of occurrences of the pairs start-art and noun-end are also incremented in the CPM.

3.3.2 Computing the CPM Rating

Both in Marshall's CLAWS algorithm and DeRose's VOLSUNGA algorithm, the number of composite occurrences of a sequence of unambiguous words is calculated as the product of the number of occurrences of each collocation in the sequence. So for example, if we had the same sequence as above, *the unfortunate workers pressed grapefruits*, and we wanted to disambiguate *pressed*, the number of composite occurrences of each of the two possible sequences [art, adj, noun, verb, noun] and [art, adj, noun, adj, noun] would be computed, and the one with the higher number of composite occurrences would be retained as the correct sequence of unambiguous syntactic categories. Using the CPM associated with chapters 1 through 6 of the QUIZ manual, the following pairs would be used, along with their number of occurrences:

start	art	49
art	adj	23
adj	noun	21
noun	verb	20
verb	noun	13
noun	end	66
noun	adj	0.01

In order to rank sequences with a pair which does not appear in the CPM, the value 0 is replaced with 0.01 so that such a pair greatly reduces the number of composite occurrences of the sequence without eliminating it altogether. Since the use of a value between 0 and 1 for such pairs will only result in their number of occurrences being decreased (e.g. 0.01 × 0.01 = 0.0001), the actual value used to replace 0 is un-important, as long as it produces a very low weight. As a result, the number of composite occurrences of *pressed* as a verb would be 49 × 23 × 21 × 20 × 13 × 66 ≅ 4 × 10e8, while the number of composite occurrences of it as an adjective equivalent would be 49 × 23 × 21 × 0.01 × 21 × 66 ≅ 3 × 10e5.

Once the Chart Parser has determined the longest matches for the unknown word in its context, if there is more than one possible match, a CPM rating is calculated using the sequences of grammar rule symbols matched. Using the example of section 3.2.3, the longest sequences matched were the following: [art, adj, adj], [art, adj, verb] and [art, noun, verb], where the unknown word is the second word in each sequence.

Since it cannot be assumed that a context begins or ends a sentence, pairs involving the *start* and *end* symbols are not considered in computing the CPM rating. They are only used in computing Maximum Paths (section 3.3.3). Assuming we had the following CPM pairs and number of occurrences, which can be found in the QUIZ CPM:

art	adj	23
adj	adj	1
adj	verb	0.01
art	noun	102
noun	verb	20

the number of composite occurrences of *umpah* as an adjective equivalent would be 23 × 1 = 23 or 23 × 0.01 = 0.23 according to the

above sequences, and the number of composite occurrences of it as a noun equivalent would be 102 × 20 = 2040. It is clear in this example that, using the QUIZ statistics, *umpah* is far more likely to be a noun equivalent than an adjective equivalent.

In most cases, computing the CPM rating for the longest matches obtained by the Chart Parser will result in one syntactic category emerging as the best match for the unknown word. However, there are cases in which even the CPM rating is the same for more than one of the matches. This situation usually occurs when the context extracted for chart parsing only contains the unknown word. This happens when the unknown word is preceded and followed by punctuation or another unknown word. In such cases, DeRose's Maximum Path algorithm is used.

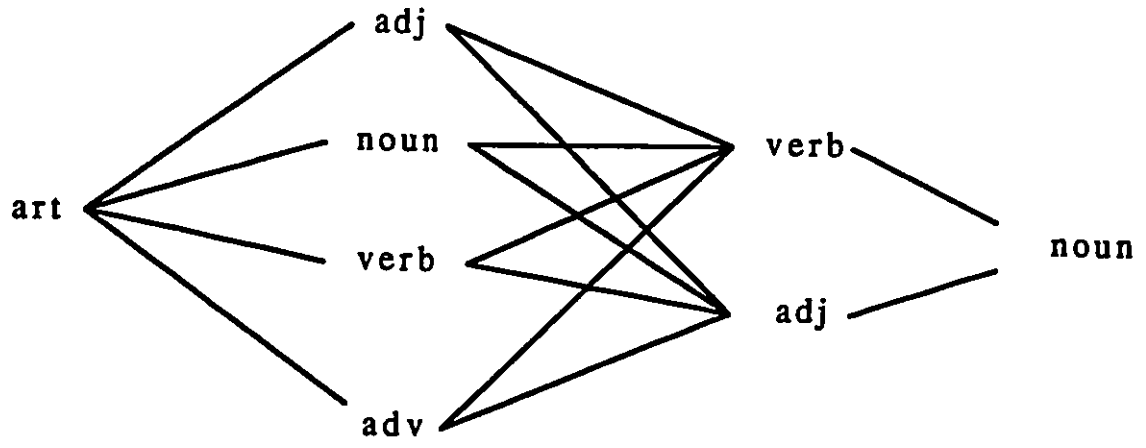
3.3.3 Computing the Maximum Path

If neither the Chart Parser nor the CPM ratings were able to narrow down the four possible syntactic categories to one, a *span network* is constructed, with syntactic categories as nodes and with edges representing collocations. The weight of each path in the network is used to assign yet another rating to the possible syntactic categories of the unknown word.

First, however, a span must be extracted from the sentence, because neither Marshall nor DeRose use contexts. A span built around the unknown word will be the shortest possible sentence segment which contains the unknown word, and begins and ends at an unambiguous word. It will therefore begin at the first unambiguous word preceding the unknown word and end at the first unambiguous word following it. Let us assume that the CPM ratings obtained in our example in section 3.3.2 were identical for adjective equivalent and noun equivalent. A span built around the unknown word of the sentence *The umpah pressed grapefruits* would begin at *the*, which is definitely an article, and it would end at *grapefruits*, which is unambiguously a noun. As a result, the span in this example would be [[art], [adj, noun, verb, adv], [verb, adj], [noun]]. If

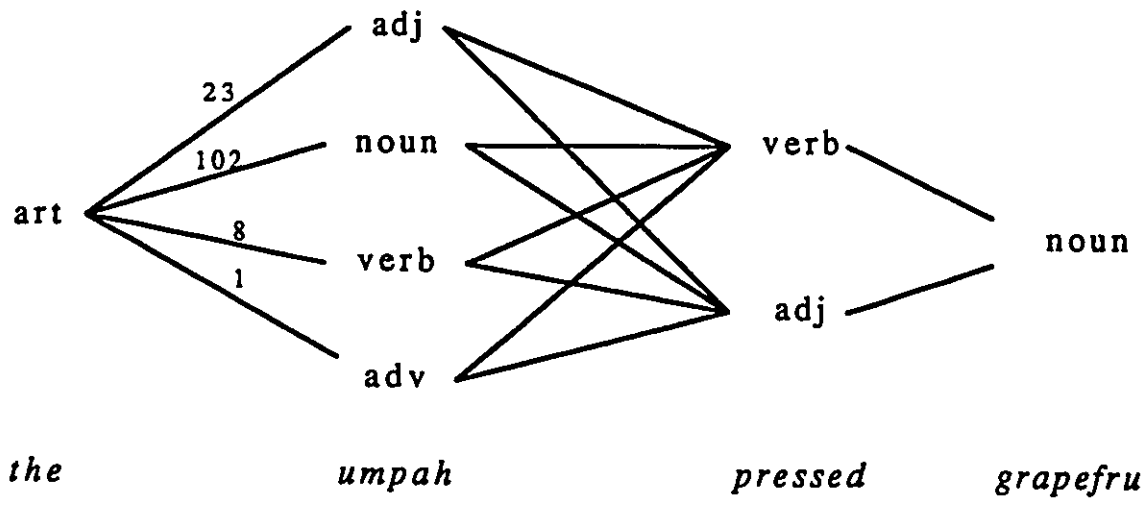
grapefruits had been ambiguous, the *end* symbol would have been used as the next unambiguous word, since *grapefruits* is the last word of the sentence. The same applies to the beginning of a sentence with the *start* symbol.

Next, a span network is constructed, using the syntactic categories of each word as the nodes and placing edges between collocations. The syntactic categories used for the unknown word are the four possible syntactic categories, so the following network is constructed with our example.

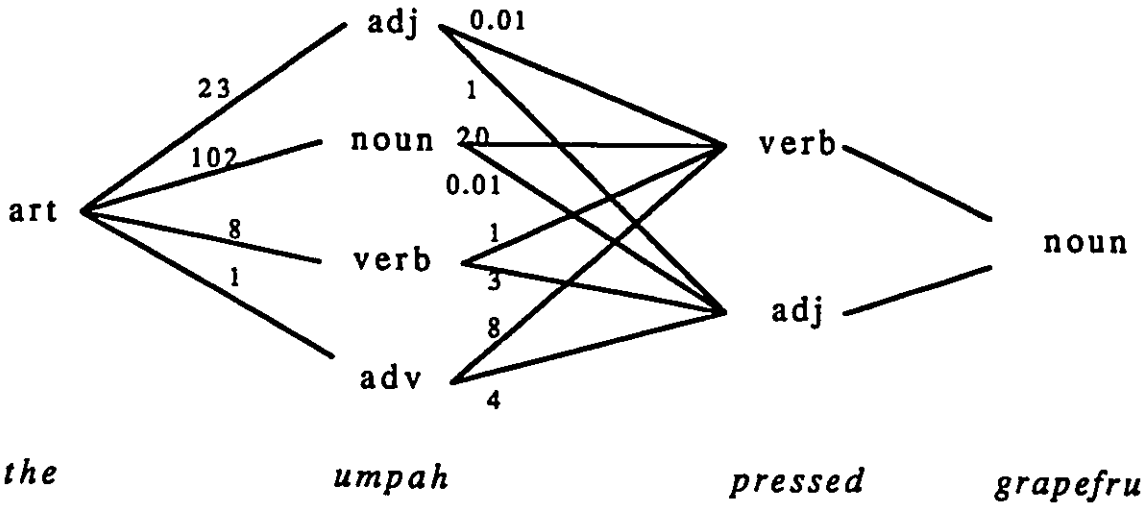


In keeping with DeRose's algorithm, the maximum path from one node to another is the path with the maximum weight of all the paths between those two nodes, where the weight of an edge is its CPM value and the weight of a path is the product of the weight of its edges. For example, the maximum path from the node representing *the* as an article (the *the*/art node) to the node representing *pressed* as a verb (the *pressed*/verb node) would traverse the syntactic category for *umpah* yielding the greatest weight.

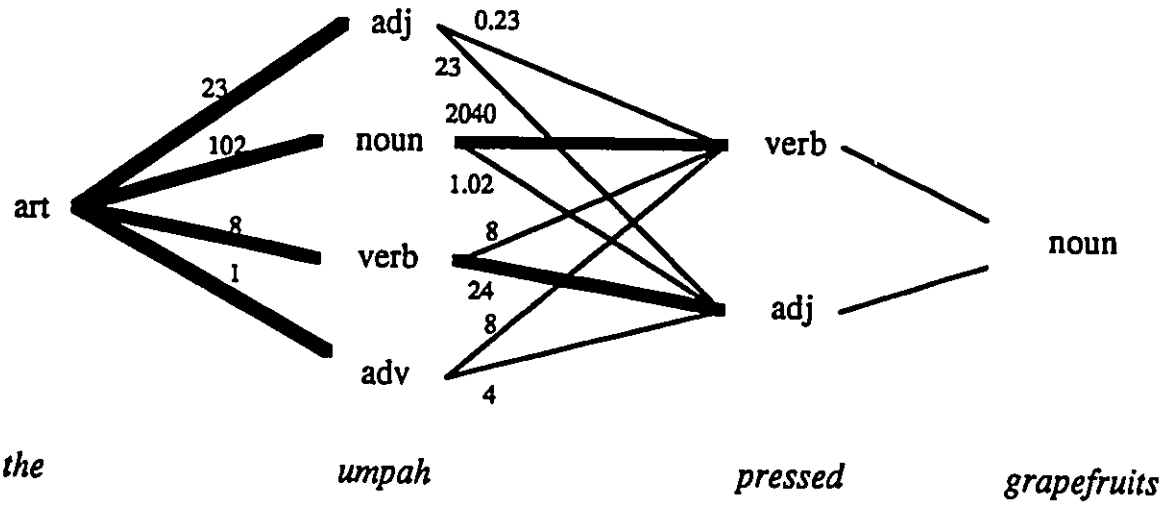
We begin by labeling each edge between *the* and *umpah* with its weight, as it is found in the CPM.



Next, each syntactic category of *pressed* is processed, and the maximum path to each one is computed. Given the following edge weights:

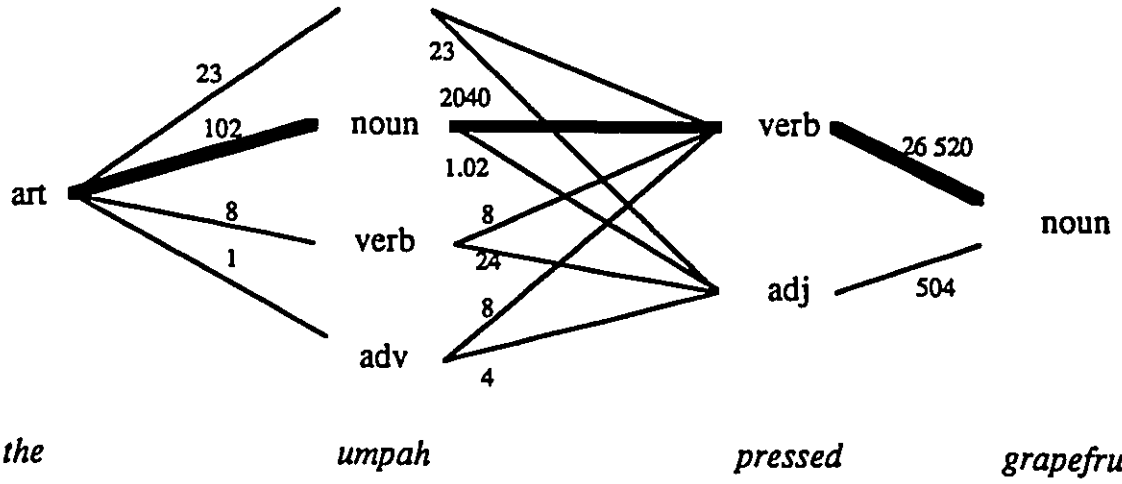


the paths to the syntactic categories of *pressed* are weighted as follows, where the bold edges represent the maximum paths to each syntactic category.



Given these paths to *pressed*/verb and *pressed*/adj, the maximum path to *pressed*/verb is the *the*/art - *umpah*/noun - *pressed*/verb path, which has the highest value (2040) of all four paths to *pressed*/verb, and the maximum path to *pressed*/adj is *the*/art - *umpah*/verb - *pressed*/adj, which has a weight of 24.

The paths to *grapefruits*/noun are computed by multiplying the value of the maximum path to *pressed*/verb (2040) by the weight of the *pressed*/verb - *grapefruits*/noun edge (its CPM value is 13), and by multiplying the weight of the maximum path to *pressed*/adj (24) by the weight of the *pressed*/adj - *grapefruits*/noun edge (its value is 21). The final span network can be found below, with the maximum path highlighted.



The maximum path through the network is therefore *the*/art - *umpah*/noun - *pressed*/verb - *grapefruits*/noun, meaning that the best match for the unknown word, using the CPM values for chapters 1 through 6 of the QUIZ manual, is noun equivalent.

Alternative matches are also ranked so that they can be used as back-ups should the user not approve the best match. The likelihood of each of the remaining syntactic categories is calculated as the weight of the maximum path through that syntactic category. As a result, just as the weight of the maximum path through *umpah*/noun is 26 520, the maximum path through *umpah*/verb has a weight of 504 through *the*/art - *umpah*/verb - *pressed*/adj - *grapefruits*/noun, *umpah*/adj has a maximum path weight of 483 through *the*/art - *umpah*/adj - *pressed*/adj - *grapefruits*/noun, and *umpah*/adv has a maximum path weight of 104 through *the*/art - *umpah*/adv - *pressed*/verb - *grapefruits*/noun. Using these rankings, each of the possible syntactic categories can be displayed to the user one by one by decreasing order of likelihood, with noun equivalent as the most likely, then verb, adjective equivalent, and adverb.

In the end, whichever method first computes a unique best match will display the matches to the user for approval. In Classification by Block, if the user approves none of the matches, and these were obtained using the Chart Parser or the CPM ratings, the Maximum Path method is invoked, otherwise the Sentence Parser's

menu of 24 possible syntactic categories is offered. In Classification by Sentence, the context size is increased by one word, and the Chart Parser is invoked again, until the context can no longer be expanded. Then, if none of the matches have been approved, the 24-option menu is presented to the user.

3.4 The SCAN Algorithm

The SCAN algorithm described in this chapter is summarized below.

- For Classification by Block, create the Collocational Probability Matrix, using the Block for Classification, before Classification by Block begins.
- Obtain Sentence from the Classification module, and the possible syntactic categories of all its words from the Word Dictionary.
- For each unknown word in the Sentence:
    - Create a context of x words preceding the unknown word, the unknown word, and one word following it.
    - For each word in the context:
        - Determine which existing active rules can be extended to which of the syntactic categories of the current word.
        - For each of those syntactic categories:
            - Activate new rules beginning with the current category.
            - Extend existing active rules which expect the current category as their next right-hand side symbol.
            - If a grammar rule has been completed, repeat the three previous steps with the left-hand side symbol of

the completed rule as the current word's category.

- Examine each active rule covering the unknown word and extract its possible category and the number of context words matched by the rule.
- IF a unique match exists and a category is approved,

  add it to the Word Dictionary

  ELSE IF Classification by Sentence

  IF increasing the size of the context yields a different context,

  Increase x by 1, create a new context, and invoke the Chart Parser with the new context

  ELSE use the 24-option menu

  ELSE • Compute CPM ratings of longest Chart Parser matches,

  • IF a unique match exists and a category is approved,

  • add it to Word Dictionary

  ELSE

  • Extract a sentence span built around the unknown word

  • Compute Maximum Path through each possible category of the unknown word

  • IF a unique match exists and a category is approved,

  add to Word Dictionary

  ELSE

  use the 24-option menu

3.5 The SCAN Results

Once SCAN had created the Collocational Probability Matrix (CPM) based on chapters 1 through 6 of the QUIZ manual, I ran the disambiguation algorithm on two texts, QUIZ chapter 5, which contained 55 unknown words, and chapter 6, which contained 87, for a total of 142 unknown words.

The results are tabulated in Table 18, and they are very good. The correct syntactic category was SCAN's first choice 62% of the time, its second choice 22% of the time, its third choice 10% and its fourth choice 6% of the time. The user can therefore expect the correct syntactic category to be in the first two choices in 84% of the cases, which is quite good despite DeRose's reported success rate of 96%. The differences between the two success rates can be directly attributed to the quality of the CPM, since SCAN cannot use a large pre-tagged corpus and must therefore rely on the occasional pairs of unambiguous collocational words in the source text.

Nevertheless, Table 18 shows that although the Chart Parser only arrives at a unique match 13% of the time, that match is *always* the correct syntactic category of the unknown word, meaning that the Chart Parser has a success rate of 100%. This was to be expected, since the Chart Parser relies on an accurate though partial grammar of English.

Table 18 also shows that the CPM ratings method is generally the one which reduces the four possible syntactic categories to one in 85% of the cases. These unique matches are correct 58% of the time, and the correct match is SCAN's second choice 26% of the time, meaning that the correct CPM ratings match will be presented to the user as the first or second choice in 84% of the cases.

Unfortunately, out of 142 unknown words, the Maximum Path approach was only invoked three times, and the correct syntactic category was its third choice twice and its last choice once. This

performance is very poor, considering that DeRose obtained a 96% success rate using the same algorithm.

However, the combination of all three methods have yielded good results, where the user can expect the correct syntactic category to be displayed for approval as either SCAN's first or second choice 84% of the time.

Methods / Rankings	Chart Parser	CPM Ratings	Maximum Paths	Totals
1st Choice	18	70	0	88
	100% of 18	58% of 121	0% of 3	62% of 142
2nd Choice	0	31	0	31
	0% of 18	26% of 121	0% of 3	22% of 142
3rd Choice	0	12	2	14
	0% of 18	10% of 121	66% of 3	10% of 142
4th Choice	0	8	1	9
	0% of 18	7% of 121	33% of 3	6% of 142
Totals	18	121	3	142
	13% of 142	85% of 142	2% of 142	

Table 18 - Results Analysis of SCAN using QUIZ Chapters 5 and 6

3.6 Limitations

The four main limitations on the performance of SCAN are the inefficiency of the Chart Parser, its inability to deal with ambiguity, the lack of a tagged corpus for gathering text statistics, and the addition of new words rather than new syntactic categories for existing words in the Word Dictionary.

If the Chart Parser could find a unique match in a reasonable time-frame (for example, in less than 40 seconds), the other methods, and hence a tagged corpus, would not be needed. Unfortunately, it would seem that bottom-up parsing methods, just like top-down ones, are often inefficient, even if they use grammars which have been reduced as much as possible. For example, in his article on comparing rule invocation strategies in Chart Parsing, Mats Wiren[1] executes eight types of Chart Parsers on two texts, one with 40 sentences and the other with seven sentences, using three different grammars of 8, 43 and 224 rules respectively. Using a bottom-up method with top-down filtering similar to SCAN's Chart Parser, Wiren's Table 2[2] shows that the text with seven sentences was parsed in 74 seconds (≅ 10 seconds/sentence) by the grammar with 43 rules, and Table 4[3] shows that the text with 40 sentences was parsed in 3898 seconds (≅ 100 seconds/sentence) by the grammar with 224 rules. The conclusion to be drawn here is that SCAN's Chart Parser, which on average parses a sentence segment in around 20 seconds with a grammar of 64 rules, is neither more nor significantly less efficient than other Chart Parsers.

Unfortunately, the Chart Parser's inefficiency is only one of its problems. A more significant one is its inability in 87% of the cases to

---

[1]Mats Wiren, "A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing," Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics (1987): 226-233.
[2]Ibid., 231.
[3]Ibid.

narrow down the possible syntactic categories of the unknown word to a single one. This problem is inherent to the English language, though, because of the high number of ambiguous words. For example, because adjectives can be used as heads of NPs and nouns can modify other nouns, it is virtually impossible to discriminate between a noun equivalent and an adjective equivalent. Another source of ambiguity is when an unknown word immediately follows a form of the verb *to be*, which can be used as a verb or an auxiliary. In those instances, the unknown word can belong to a NP following the verb and can therefore be a noun or an adjective (e.g. He is *lucky*, These are *brick* houses), it can be an adverb (e.g. He is *often* right, she is *anxiously* waiting) or a verb (e.g. she is *waiting*, he was *told*).

Because of the inherent inefficiency and ambiguity problems of the Chart Parser, a better method of dealing with unknown words consists in using statistics and the probability of occurrence of each possible syntactic category of the unknown word with the categories of the neighboring words. Since the CPM ratings determine a unique match for the unknown word in 85% of the cases once the Chart Parser has failed to do so, we see that the combination of the Chart Parser (which succeeds in 13% of the cases) and the CPM ratings achieves its objectives 98% of the time.

If, however, the Chart Parser were dropped altogether, the CPM ratings could not be computed on context matches, and SCAN would have to use the Maximum Path approach which also makes use of the CPM ratings. The success rate of this latter method is rather difficult to judge at this point since it was only needed three times in the tests conducted, and each time resulted in the correct syntactic category being poorly ranked. Should a tagged corpus be acquired in the near future, this syntactic category disambiguation method should be re-evaluated. According to the outstanding results obtained by DeRose, we should be able to rely solely on the Maximum Path algorithm once a CPM has been set up using a large tagged corpus.

Another limitation of SCAN is its inability to recognize a new syntactic category of a word already appearing in the Word Dictionary with a different syntactic category. Although there is no doubt that this limitation introduces some degree of error to the disambiguation process, an enhancement to resolve this problem should be considered as part of future work.

3.7 Related Work

In addition to DeRose (1988) and Marshall (1983), a few researchers have explored the area of syntactic category disambiguation using text statistics.

Beale (1985), for example, uses "a transition probability matrix of tag pairs to compute the most probable tag for an ambiguous form given the immediately preceding and following tags,"[1] and achieves a reported success rate of 94.3%. This is far better than earlier attempts, such as those by Greene and Rubin (1971), who obtained a 77% success rate using *context frame rules* applied to a context which includes two words preceding and following the ambiguous word.

Atwell and Drakos (1987) describe a few syntactic category disambiguation methodologies, including CLAWS and RUNNEWTAGSET. The latter clusters ambiguous words whose immediately preceding and following words belong to the same syntactic categories, in the hope that the words of each cluster would belong to the same syntactic category.

Church (1988) uses *lexical probabilities* to calculate the likelihood of a word belonging to a given syntactic category. This measure is calculated as the word's frequency of occurrence as a member of that category divided by its total frequency of occurrence

---

[1] Andrew D. Beale, "Grammatical Analysis by Computer of the Lancaster Oslo/Bergen (LOB) Corpus of British English Texts," Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (1985): 294-295.

in the text. Contexts can also be used for disambiguation purposes using the same principles, for example the probability of an article appearing between a verb and a noun is calculated as the frequency of occurrence of such a context in the tagged text divided by the frequency of an article followed by a noun in the text.

One of the main requirements of all the above methods is for a tagged corpus in order to gather the needed statistics. However, because SCAN is forced to rely on the source text, which is not unambiguously tagged with the appropriate syntactic category of each word, neither the statistics used nor the results obtained can be as accurate as those obtained using the methodologies discussed above.

Other methods of syntactic category disambiguation exist which do not depend on text statistics. Duffy (1986) and Hindle (1989), for example, make use of *disambiguation rules* for this purpose. Duffy's algorithm begins with a set of hand-coded disambiguation rules and has the capability of sending mail messages to the developers whenever a rule fails, so that it can be corrected or additional rules can be introduced in the rule base. The method put forth by Hindle, however, automatically infers new disambiguation rules, beginning with a few hand-coded rules and using a tagged corpus as a training set. Once the training set has been processed and additional rules have been inferred, the reported error rate on the test set, a different text, is 10%.

Another method of syntactic category disambiguation consists in using principles from the area of Example-Based Learning. Rayner et al. (1988) use a set of example sentences and a grammar in order to infer a lexicon, which will contain every word found in the text along with its syntactic category. The lexicon evolves as a set of hypotheses of each word's category as each example sentence is processed. Although the goal of this algorithm is to learn a lexicon and not to disambiguate between the syntactic categories of a word, each word is found to be an unknown word because no lexicon exists at the beginning of the learning process. Unfortunately, this approach

only works well if each word occurs frequently enough in the example sentences, otherwise the syntactic category assigned to each word may not be correct. Also, this method requires a manual pre-ordering of the example sentences for optimal results, "so as to minimize the number of new words introduced at each stage."[1]

Other methods include the use of Lambek calculus ("a categorial reduction system which is analogous to the implicational fragment of propositional logic"[2]) and a theorem prover for syntactic category disambiguation (van der Wouden and Heylen (1988)), and the use of morphological knowledge and rules based on syntactic and semantic knowledge to predict the syntactic category of an unknown word (Fournier et al. (1988)).

---

[1]Manny Rayner, Asa Hugosson, and Goran Hagert, "Using a Logic Grammar to Learn a Lexicon," Proceedings of the 12th International Conference on Computational Linguistics (1988): 525.

[2]T. van der Wouden and D. Heylen, "Massive Disambiguation of Large Text Corpora With Flexible Categorial Grammar," Proceedings of the 12th International Conference on Computational Linguistics COLING 1988 (1988): 695.

Conclusion

The AZTEC system automatically acquires from a technical text a set of its most technical concepts and assigns each concept to one of the five concept categories: **Technical, Rather Technical, Possibly Technical, Not Technical** and Technicality **Unknown.** Chapter 1 provided an overview of AZTEC, the assumptions guiding its design and its four main components: the Text Base Builder, the Classification algorithm, the Sentence Parser and the SCAN module. Chapter 2 described in detail the NP classification process and both the Classification by Sentence and the Classification by Block algorithms. Finally, chapter 3 gave an overview of the SCAN module which attempts to automatically determine an unknown word's syntactic category before the Sentence Parser processes each sentence.

The main purpose of this research was to acquire technical concepts from an un-edited technical text using no domain knowledge, and in the process to examine to what extent syntactic knowledge and text statistics alone could be used for this purpose. My experiments have shown that the success rate associated with this method of concept classification is in the 50%-60% range (see section 2.5), excluding the concepts assigned to the **Unknown** category which is only used for very infrequent concepts. The success rate achieved is significantly better than the random assignation of each concept to one of four categories, which would most likely yield a 25% success rate.

Another conclusion drawn from the development of AZTEC was related to the values of the Minimum Bounds against which each concept's head noun frequency is compared and the percentage used as the Term Percentage boundary (section 2.2). It was known that the value of each Minimum Bound should increase with the total number of NPs in the text, but I discovered that the percentages

used to compute the Minimum Bound values and the percentage used as the Term Percentage boundary should decrease as the total number of NPs increases (see section 2.5). This discovery is likely to render further enhancements to the Classification by Block algorithm far more complicated than expected if the goal of these enhancements is to improve the Classification results. This is especially true if AZTEC is to be executed using a text whose size greatly differs from the texts used in the experiments of section 2.5, since new Minimum Bound percentages and a new Term Percentage boundary will have to somehow be devised for those texts. It should be possible to locally optimize the Minimum Bound percentages and the Term Percentage boundary for several texts of different sizes in order to approximate a function to compute these values automatically for new texts. Such an approximation promises to entail a significant amount of work with no guarantee of success, and should not be undertaken as a minor project.

The SCAN algorithm, on the other hand, yielded far more promising results. Although SCAN's success rate at determining the unknown word's correct syntactic category as its first choice is only 62%, its success rate at ranking the correct one as either its first or second choice is 84%.

In conclusion, this thesis has shown that the acquisition of technical concepts from text using no domain knowledge can yield somewhat mediocre results, which may be why it is not a very common research endeavour. It remains to be seen to what extent AZTEC's methods and results can be improved, given the amount of effort required to enhance the present algorithm.

References

Allen, J. 1987. *Natural Language Understanding*. Menlo Park, CA: The Benjamin/Cummings Publishing Company, Inc.

Anick, P., and J. Pustejovsky. 1990. An Application of Lexical Semantics to Knowledge Acquisition from Corpora. *Proceedings of the 13th International Conference on Computational Linguistics COLING 1990*, vol. 2, Helsinki, pp. 7-12.

Atwell, E. S., and N. F. Drakos. 1987. Pattern Recognition Applied to the Acquisition of a Grammatical Classification System from Unrestricted English Text. *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, pp. 56-62.

Beale, A. D. 1985. Grammatical Analysis by Computer of the Lancaster Oslo/Bergen (LOB) Corpus of British English Texts. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, pp. 293-298.

Borko, H. 1970. Experiments in Book Indexing by Computer. *Information Storage and Retrieval*, vol. 6, no. 1, pp. 5-16.

Church, K. W. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Second Conference on Applied Natural Language Processing*, Austin, pp. 136-143.

Church, K. W., and P. Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics*, vol. 16, no. 1, pp. 22-29.

Cognos Incorporated. 1985. *PowerHouse QUIZ.*

Copeck, T., S. Delisle, and S. Szpakowicz. 1992. Parsing and Case Analysis in TANKA. *Proceedings of the 15th International Conference on Computational Linguistics COLING 1992*, Helsinki, pp. 1008-1012.

Delisle, S. 1990. *A Parser for Processing Technical Texts With a Large Coverage of English.* University of Ottawa, Department of Computer Science. Technical Report TR-90-25.

Delisle, S., and S. Szpakowicz. 1991. A Broad-Coverage Parser for Knowledge Acquisition from Technical Texts. In *Proceedings of the Fifth International Conference on Symbolic and Logical Computing*, ed. E. Johnson, Dakota State University, pp. 169-183.

DeRose, S. J. 1988. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, vol. 14, no. 1, pp. 31-39.

Dillon, M., and P. Federhart. 1982. The Use of Discriminant Analysis to Select Content-Bearing Words. *Journal of the American Society for Information Science*, vol. 33, no. 4, pp. 245-253.

Dillon, M., and L. K. McDonald. 1983. Fully Automatic Book Indexing. *The Journal of Documentation*, vol. 39, no. 3, pp. 135-154.

Duffy, G. 1986. Categorial Disambiguation. *Proceedings of the Fifth National Conference on Artificial Intelligence AAAI-86*, Philadelphia, pp. 1079-1082.

Earley, J. 1970. An Efficient Context-Free Parsing Algorithm. *Communications of the Association for Computing Machinery*, vol. 13, no. 2, pp. 94-102.

Fano, R. 1961. *Transmission of Information: A Statistical Theory of Communications*. Cambridge, MA: MIT Press.

Fournier, J. P., P. Herman, G. Sabah, A. Vilnat, N. Burgaud, and M. Gilloux. 1988. Processing of Unknown Words in a Natural Language Question-Answering System. *Computational Intelligence*, vol. 4, no. 2, pp. 205-211.

Greene, B. B., and G. M. Rubin. 1971. *Automatic Grammatical Tagging of English*. Brown University, Department of Linguistics.

Grishman, R., L. Hirschman, and N. T. Nhan. 1986. Discovery Procedures for Sublanguage Selectional Patterns: Initial Experiments. *Computational Linguistics*, vol. 12, no. 3, pp. 205-215.

Hayes, P. J., L. E. Knecht, and M. J. Cellio. 1988. A News Story Categorization System. *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, pp.9-17.

Hindle, D. 1989. Acquiring Disambiguation Rules From Text. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, pp. 118-125.

Hindle, D. 1990. Noun Classification from Predicate-Argument Structures. *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, Pittsburgh, pp. 268-275.

Hirschman, L., R. Grishman, and N. Sager. 1975. Grammatically-Based Automatic Word Class Formation. *Information Processing & Management*, vol. 11, pp. 39-57.

Kaplan, R. 1973. A General Syntactic Processor. In *Natural Language Processing*, ed. R. Rustin. Englewood Cliffs, NJ: Prentice-Hall.

Kay, M. 1973. The Mind System. In *Natural Language Processing*, ed. R. Rustin. Englewood Cliffs, NJ: Prentice-Hall.

Kay, M. 1980. *Algorithm Schemata and Data Structures in Syntactic Processing*. Xerox, Palo Alto Research Center.

Lanka, S. 1985. Automatically Inferring Database Schemas. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence IJCAI 85*, Los Angeles, pp. 647-649.

Luhn, H. P. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159-165.

Maarek, Y. S., D. M. Berry, and G. E. Kaiser. 1991. An Information Retrieval Approach for Automatically Constructing Software Libraries. *IEEE Transactions on Software Engineering*, vol. 17, no. 8, pp. 800-813.

Marshall, I. 1983. Choice of Grammatical Word-Class Without Global Syntactic Analysis: Tagging Words in the LOB Corpus. *Computers in the Humanities*, vol. 17, pp. 139-150.

Mellish, C. S. 1989. Some Chart-Based Techniques for Parsing Ill-Formed Input. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, pp. 102-109.

O'Shaughnessy, D. D. 1989. Parsing With a Small Dictionary for Applications Such as Text to Speech. *Computational Linguistics*, vol. 15, no. 2, pp. 97-108.

Pustejovsky, J. 1989. Type Coercion and Selection. *Proceedings of West Coast Conference on Formal Linguistics*, Vancouver.

Pustejovsky, J. 1989. *The Generative Lexicon.* MS, Brandeis University.

Rayner, M., A. Hugosson, and G. Hagert. 1988. Using a Logic Grammar to Learn a Lexicon. *Proceedings of the 12th International Conference on Computational Linguistics COLING 1988*, Budapest, pp. 524-528.

Salton, G. 1986. On the Use of Term Associations in Automatic Information Retrieval. *Proceedings of the 11th International Conference on Computational Linguistics COLING 1986*, Bonn, pp.380-386.

_____. 1988. Syntactic Approaches to Automatic Book Indexing. *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pp. 204-210.

Szpakowicz, S. 1990. Semi-Automatic Acquisition of Conceptual Structure from Technical Texts. *International Journal of Man-Machine Studies*, vol. 33, pp. 385-397.

van der Wouden, T., and D. Heylen. 1988. Massive Disambiguation of Large Text Corpora With Flexible Categorial Grammar. *Proceedings of the 12th International Conference on Computational Linguistics COLING 1988*, Budapest, pp. 694-698.

Wiren, M. 1987. A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, pp. 226-233.

Zernik, U., and P. Jacobs. 1990. Tagging for Learning: Collecting Thematic Relations from Corpus. *Proceedings of the 13th International Conference on Computational Linguistics COLING 1990*, vol. 1, Helsinki, pp. 34-39.

Appendix 1

The following is an example of the output of the optimal Classification by Sentence algorithm, using the dictionary flag of the head noun and the consolidated dictionary flag of the modifier, the total word frequency of each NP compared against two Minimum Bounds, set to 0.5% and 0.25% of the total number of words in the text, and the term percentage set to 100%. The sentences processed in this example were taken from portions of the first two chapters of the QUIZ manual, but the classification process used the word frequencies associated with chapters 1 through 6 of QUIZ. Since no single numeric score is assigned to each concept, the following are not ranked within each category.

TECHNICAL:

quiz
data
page
report
column heading
statement
read datum
quiz statement
statement line
employee information
file
employee file
report statement
select statement
key item
item
large file
access statement
position file

RATHER TECHNICAL:

datum dictionary
application designer
single report
employee
short report
process time
key value
option
record complex
one-to-many situation

POSSIBLY TECHNICAL:

computer
datum
information
key
basic rule
record
feature
entry
return key
uppercase
lowercase letter
user
manual
word access
access
screen
list
name
problem
sort-key
symbol
qualifier
ampersand
line

asterisk
advance feature
specific part
selection condition
example
compound record

NOT TECHNICAL:

question
answer
right question
standard
dictionary
good idea
group
minute
case
system
way
change
time
result
equal sign
large quantity
small amount
type
reason
thing
easy way
order
point

Appendix 2

The following is an example of the output of the Classification by Block algorithm, using the dictionary flags for the head noun and the last modifier, and the term frequency compared against one Minimum Bound, set to 5% of the total number of NPs in the text. The block of text processed in this example consists of the first two chapters of the QUIZ manual. Since no single numeric score is assigned to each concept, the following are not ranked within each category.

TECHNICAL:

application designer
quiz statement
select statement
access statement
employee file
one-to-many situation
quiz
column heading
file

PROBABLY TECHNICAL:

POSSIBLY TECHNICAL:

statement
data
computer
simplicity
relationship
datum dictionary
basic rule
element
feature
entry
uppercase
lowercase letter
policy
user
read datum
manual
statement line
employee information
word access
screen
problem
access
sort-key
abbreviation
symbol
qualifier
datum
ampersand
asterisk
large file
process time
advance feature
option
specific part
selection condition
employee
dallas
sort statement
record complex
item
key item
position file
report statement

NOT TECHNICAL:

question
answer
right question
page
standard
dictionary

return key
good idea
group
minute
case
list
name
change
time
sort
result
short report
equal sign
fact
large quantity
line
dot
flag
small amount
information
key value
single report
work
report
way
various level
manner
type
reason
key
system
thing
compound record
record
easy way
example
order
point

Appendix 3

The following is an example of the output of the Classification by Block algorithm, using the dictionary flags for the head noun and the last modifier, the term frequency of each term and the head noun frequency of each single noun compared against one Minimum Bound, set to 3% of the total number of NPs in the text, and the term percentage compared against 20%. The block of text processed in this example consists of the first two chapters of the QUIZ manual. Since no single numeric score is assigned to each concept, the following are not ranked within each category.

TECHNICAL:

application designer
feature
quiz statement
select statement
access statement
employee file
one-to-many situation
quiz
column heading
item
file
statement

PROBABLY TECHNICAL:

POSSIBLY TECHNICAL:

data
computer
simplicity
relationship
datum dictionary
basic rule
element
entry
uppercase
lowercase letter
policy
user
read datum
manual
statement line
employee information
word access
screen
problem
access
sort-key
abbreviation
symbol
qualifier
datum
ampersand
asterisk
information
large file
process time
advance feature
option
specific part
selection condition
employee
dallas
report
sort statement
record complex
record
key item
position file
report statement

NOT TECHNICAL:

question
answer
right question
page
standard
dictionary
return key
good idea
group
minute
case
list
name
change
time
sort
result
short report
equal sign
fact
large quantity
line
dot
flag
small amount
key value
single report
work
way
various level
manner
type
reason
key
system
thing
compound record
easy way
example
order
point

Appendix 4

The following is an example of the output of the Classification by Block algorithm, using the dictionary flags for the head noun and the last modifier, the head noun frequency of each term compared against two Minimum Bounds, set to 3% and 1.5% of the total number of NPs in the text, and the term percentage compared against 20%. The block of text processed in this example consists of the first two chapters of the QUIZ manual. Since no single numeric score is assigned to each concept, the following are not ranked within each category.

TECHNICAL:

application designer
feature
quiz statement
select statement
access statement
employee file
one-to-many situation
quiz
column heading
item
file
statement

PROBABLY TECHNICAL:

read datum
employee information
word access
access
datum
information
advance feature
report
record
key item

POSSIBLY TECHNICAL:

data
computer
simplicity
relationship
datum dictionary
basic rule
element
dictionary
entry
uppercase
lowercase letter
policy
user
manual
statement line
screen
problem
sort-key
abbreviation
symbol
qualifier
ampersand
asterisk
large file
process time
option
specific part
single report
selection condition
employee
dallas
sort statement
key
record complex
position file
report statement

NOT TECHNICAL:

question
answer
right question
page
standard
return key
good idea
group
minute
case
list
name
change
time
sort
result
short report
equal sign
fact
large quantity
line
dot
flag
small amount
key value
work
way
various level
manner
type
reason
system
thing
compound record
easy way
example
order
point

Appendix 5

The following is an example of the output of the optimal Classification by Block algorithm, using the dictionary flags for the head noun and the last modifier, the head noun frequency of each term compared against two Minimum Bounds, set to 1.5% and 0.75% of the total number of NPs in the text, and the term percentage compared against 10%. The block of text processed in this example consists of the first two chapters of the QUIZ manual. Since no single numeric score is assigned to each concept, the following are not ranked within each category.

TECHNICAL:

feature
select statement
employee file
quiz
column heading
item
file
statement
read datum
employee information
word access
access
datum
information
advance feature
report
record
key item
datum dictionary
dictionary
key
report statement

RATHER TECHNICAL:

quiz statement
access statement
computer
statement line
qualifier
ampersand
process time
option
single report
return key
short report
compound record

POSSIBLY TECHNICAL:

large file
sort statement
position file
question
right question
list
time
line
way
easy way

NOT TECHNICAL:

TECHNICALITY UNKNOWN:

application designer
one-to-many situation
data
simplicity
relationship
basic rule
element
entry

uppercase
lowercase letter
policy
user
manual
screen
problem
sort-key
abbreviation
symbol
asterisk
specific part
selection condition
employee
dallas
record complex
answer
page
standard
good idea
group
minute
case
name
change
sort
result
equal sign
fact
large quantity
dot
flag
small amount
key value
work
various level
manner
type
reason
system
thing
example
order
point

Appendix 6
Expected Classification Results
QUIZ Chapters 1-6

UNKN	abbreviation
NOT	abra
RATH	access datum
POSS	access
POSS	actual datum
POSS	actual heading
POSS	actual text
UNKN	act
UNKN	add extra
POSS	advance feature
POSS	age-date prompt
POSS	age routine
RATH	allowable entry
UNKN	alot
POSS	alphabetical list
POSS	alphabetical order
UNKN	alternative approach
NOT	amount
POSS	ampersand
POSS	ampersand symbol
NOT	answer
POSS	application designer
POSS	ascending order
POSS	associate condition
NOT	associate detail
POSS	asterisk
RATH	automatic feature
POSS	available option
UNKN	average
POSS	balance
NOT	basic rule
NOT	basis
POSS	batch
NOT	beginning
UNKN	benefit
POSS	billing file
POSS	billing
POSS	billing report
POSS	blank line
UNKN	blank
POSS	blank report-item
POSS	blank space
POSS	body
POSS	boston employee
NOT	boston
POSS	boston staff
NOT	bottom
UNKN	box
POSS	branch change
POSS	branch code
POSS	branch lastname
POSS	branchname change
POSS	branchname
POSS	branch name
POSS	branch report-item
POSS	branch start
POSS	branch
UNKN	breadth
BAD	break can
UNKN	break structure
POSS	break
NOT	canadian city
BAD	can match
BAD	can use
BAD	can work
BAD	can
UNKN	case modification
NOT	case
NOT	cause
UNKN	center
POSS	century prefix
NOT	certain item
NOT	certain thing
UNKN	chance
POSS	change value
POSS	change
NOT	chapter
POSS	character
UNKN	check
UNKN	choice
NOT	city
POSS	close balance
UNKN	collection
TECH	column heading
POSS	column
POSS	column number
POSS	combination
POSS	combine item
POSS	command
POSS	comment line
NOT	company
UNKN	company organization

POSS	compile report
NOT	complete detail
POSS	complete instruction
NOT	complete list
POSS	complete syntax
NOT	complex
POSS	complex report
POSS	complex statement
POSS	component
POSS	compound record
RATH	compound statement
UNKN	comptroller
POSS	computer
POSS	computer terminal
NOT	conclusion
POSS	condition
NOT	contents
POSS	control break
TECH	control footing
POSS	control group
RATH	control-heading line
TECH	control heading
POSS	control key
POSS	control
RATH	conversion statement
POSS	count record
UNKN	country
UNKN	cross-tabulation
POSS	crt terminal
POSS	currency symbol
POSS	customer
NOT	dallas
NOT	dallas office
POSS	data
BAD	date can
RATH	date format
POSS	datejoined
RATH	date-type item
POSS	date
RATH	datum dictionary
POSS	datum
POSS	day employee
POSS	day
UNKN	debit indicator
POSS	decimal point
RATH	default column
POSS	default date
RATH	default format
RATH	default heading
POSS	default
POSS	default picture
POSS	default value
RATH	define item
POSS	demonstration system
POSS	descending order
POSS	designate file
POSS	desire surname
NOT	desire value
POSS	detail amount
POSS	detail information
POSS	detail item
POSS	detail line
NOT	detail
POSS	dictionary manual
POSS	dictionary
POSS	different format
POSS	different result
BAD	difficulty here
UNKN	discount
NOT	discussion
UNKN	display
POSS	division
POSS	dollar sign
UNKN	dot
UNKN	ease
POSS	easy-to-read format
NOT	easy way
POSS	edited contents
RATH	editfile
UNKN	edit
POSS	editor command
POSS	editor
POSS	effect
NOT	efficient way
UNKN	effort
UNKN	eight-digit-date n
POSS	eight-digit date
POSS	eight-line page
NOT	elaborate use
UNKN	element
POSS	employee change
POSS	employee file
POSS	employee information
POSS	employee
POSS	employee name
POSS	employee number
POSS	employee report
NOT	end
POSS	entire file

NOT	entire list
POSS	entry
POSS	equal sign
UNKN	equation
NOT	every time
POSS	example access
POSS	example
RATH	execution-time parameter
POSS	execution time
UNKN	experimentation
NOT	explanation
BAD	expression can
POSS	expression
RATH	extract function
POSS	extra feature
POSS	extra position
UNKN	fact
POSS	fast response
POSS	feature
POSS	female employee
UNKN	female
POSS	fictitious company
BAD	file can
TECH	file
UNKN	filename
POSS	file name
RATH	final footing
POSS	firstname
POSS	fix position
POSS	fix value
UNKN	flag
NOT	follow example
POSS	follow message
POSS	follow page
POSS	follow report
BAD	footing can
TECH	footing
TECH	footing report-group
POSS	format
RATH	format option
RATH	format report
RATH	formatting option
POSS	form letter
POSS	form
NOT	full discussion
NOT	full explanation
BAD	function can
RATH	function
UNKN	future industry
NOT	future report
NOT	general principle
NOT	general rule
UNKN	good decision
NOT	good example
UNKN	good idea
UNKN	go
POSS	grand total
NOT	great detail
NOT	group
NOT	hand
UNKN	heading-body-footing concept
RATH	heading feature
TECH	heading
TECH	heading report-item
POSS	help system
POSS	help
UNKN	hyphen
POSS	important information
UNKN	inclusive category
POSS	information
UNKN	ingenuity
POSS	initial value
POSS	installation tape
NOT	instance
POSS	instruction
RATH	interactive session
UNKN	interesting exercise
POSS	interim item
UNKN	interruption
NOT	introductory explanation
POSS	introductory message
POSS	inventory
POSS	invoice-amount
POSS	invoice amount
BAD	invoice can
UNKN	invoice-detail
POSS	invoice example
POSS	invoice-master file
UNKN	invoice-master
POSS	invoice
UNKN	invoice-number
POSS	invoice number
POSS	invoice total
POSS	item balance
POSS	item billing
UNKN	item datatype
POSS	item employee
RATH	item heading

POSS	itemize detail
POSS	item lastname
POSS	item
UNKN	item sex
POSS	item type
UNKN	january
UNKN	job
UNKN	keyboard
POSS	key information
POSS	key item
POSS	key value
POSS	keyword
POSS	key
UNKN	knowledge
POSS	label form
POSS	label
POSS	languagetext
NOT	language
POSS	large file
UNKN	large quantity
POSS	lead sign
NOT	least detail
POSS	length
POSS	letter example
POSS	letter
NOT	level
RATH	line
POSS	linkage
RATH	link file
UNKN	list description
POSS	list
NOT	long list
POSS	lowercase letter
POSS	low level
POSS	mail label
POSS	mail list
UNKN	mail
NOT	major division
POSS	male employee
NOT	manner
POSS	manual
NOT	margaret
POSS	match item
POSS	match
NOT	matter
UNKN	maximum
POSS	maximum number
POSS	message
UNKN	michael
NOT	minute
POSS	mislead count
UNKN	money
POSS	month change
POSS	month number
POSS	month
NOT	name beginning
NOT	name
POSS	need information
NOT	needless repetition
NOT	need
POSS	negative number
POSS	new data
POSS	new line
POSS	new page
POSS	new report-group
POSS	new report
POSS	new statement
UNKN	new term
POSS	next-to-first place
POSS	next-to-last place
NOT	number name
RATH	number
POSS	numeric expression
POSS	numeric value
POSS	occurrence
NOT	office
POSS	one-level report
RATH	one-time heading
POSS	one-to-many situation
POSS	one-to-one linkage
UNKN	on-line inquiry
POSS	open balance
POSS	operation
UNKN	operator
UNKN	optional specification
BAD	option can
RATH	option keyword
RATH	option
UNKN	option n
POSS	order
BAD	organization can
UNKN	overdue account
POSS	own heading
POSS	own item
POSS	own report-item
UNKN	own security
POSS	page break
RATH	page heading
POSS	page length
POSS	page number
POSS	page width
POSS	page

UNKN	parenthesis
BAD	parenthesis need
UNKN	parse
POSS	parse time
POSS	particular operation
NOT	particular report
POSS	particular statement
NOT	part
POSS	pattern
UNKN	payroll
UNKN	people
UNKN	percentage
POSS	percent symbol
RATH	permanent file
UNKN	personnel
NOT	person
NOT	phone number
NOT	place
POSS	planned report
NOT	point
UNKN	policy
POSS	portion
POSS	position file
POSS	position line
RATH	position report-item
POSS	position
POSS	powerhouse date
UNKN	powerhouse
RATH	precede statement
UNKN	president
RATH	primary file
NOT	principle
POSS	printer
POSS	printing
UNKN	printout
NOT	problem
POSS	procedure
BAD	process can
UNKN	process
POSS	process time
POSS	produce report
RATH	production report
POSS	professional-looking report
POSS	project change
POSS	project
POSS	prompt character
POSS	prompt
UNKN	prototype
UNKN	provision
UNKN	prov-state
NOT	purpose
POSS	qualifier
NOT	question mark
NOT	question
BAD	quiz can
UNKN	quiz issue
POSS	quiz list
TECH	quiz
BAD	quiz need
RATH	quiz report
RATH	quiz session
POSS	quiz start
TECH	quiz statement
POSS	quiz use
POSS	quotation mark
UNKN	range
UNKN	readability
POSS	read datum
UNKN	real genius
NOT	reason
POSS	recent employee
POSS	record complex
POSS	record count
RATH	record
POSS	recurring requirement
UNKN	relationship
UNKN	remainder
POSS	repeat key
POSS	repetitive printing
BAD	report can
BAD	report consist
UNKN	report divisionname
TECH	report-group
UNKN	report image
TECH	report-item
RATH	report-item name
RATH	report item
POSS	report requirement
RATH	report statement
TECH	report
POSS	requirement
UNKN	respect
POSS	response
POSS	result report
POSS	result
POSS	return key
UNKN	return
POSS	right information
POSS	right portion
NOT	right question
NOT	room

POSS	routine
POSS	run count
POSS	run quiz
POSS	run subtotal
POSS	sale tax
POSS	sample report
POSS	screen
POSS	select information
RATH	selection condition
POSS	selective report
UNKN	select
POSS	select record
TECH	select statement
POSS	semicolon
NOT	series
BAD	session can
POSS	short report
POSS	sign
POSS	simple operation
UNKN	simplicity
POSS	single file
POSS	single report
NOT	situation
UNKN	six-digit-date
POSS	six-digit date
POSS	size
NOT	small amount
NOT	smith
UNKN	solution
POSS	sophisticated report
POSS	sort-key
RATH	sort-key option
RATH	source file
NOT	source
RATH	source statement
UNKN	source version
POSS	space
POSS	special character
POSS	special instruction
POSS	special statement
NOT	specific form
UNKN	specific limit
NOT	specific part
UNKN	specific section
POSS	specific value
NOT	staff
UNKN	stage
RATH	standard heading
POSS	standard page
UNKN	standard
POSS	start value
NOT	start
BAD	statement can
BAD	statement cause
BAD	statement draw
UNKN	statement influence
POSS	statement line
TECH	statement
POSS	status control
POSS	string expression
RATH	string function
POSS	string
RATH	styled heading
UNKN	subject
POSS	submission batch
UNKN	substance
POSS	subtotal
POSS	subtotal value
POSS	summary information
POSS	summary
RATH	summary operation
POSS	surname
UNKN	suspend
UNKN	switch
POSS	symbol
POSS	syntax
POSS	system manager
POSS	system
UNKN	table
POSS	tabs work
POSS	tape
POSS	task
POSS	tax
RATH	temporary file
POSS	terminal
POSS	terminal screen
POSS	test-pattern
POSS	text editor
RATH	text file
POSS	text
RATH	textual heading
NOT	thing
POSS	three-up form
POSS	time
POSS	title page
POSS	title
POSS	top line
POSS	total employee
POSS	total number
POSS	total
POSS	trailing space
POSS	transaction
UNKN	trouble

POSS	two-level report
POSS	two-up form
POSS	type size
UNKN	typewriter
NOT	type
POSS	unique branch
NOT	unique detail
POSS	unique key
NOT	unnecessary repetition
POSS	uppercase letter
POSS	uppercase
POSS	useful option
RATH	use option
POSS	user
NOT	use
RATH	value
NOT	various level
UNKN	view
NOT	way
UNKN	wide variety
RATH	widowed footing
RATH	widowed heading
NOT	word
NOT	work
UNKN	xbilling
BAD	xx
POSS	y key
UNKN	yymmdd
UNKN	yyyymmdd
UNKN	zero

Appendix 7
Expected Classification Results
Delisle's Parser Text

UNKN	abbreviation
UNKN	absence
POSS	acceptable input
POSS	access language
POSS	account capitalization
UNKN	account
RATH	acquisition facility
UNKN	acquisition laboratory
RATH	acquisition
RATH	acquisition task
POSS	active non-terminal
POSS	activespy list
POSS	actual grammar
POSS	actual implementation
NOT	actual number
POSS	actual parse
POSS	actual session
POSS	additional source
POSS	addition modification
NOT	addition
UNKN	adjustment
UNKN	advance
NOT	advantage
UNKN	advent
UNKN	adverb
POSS	allot time
POSS	alternative approach
RATH	ambiguity
NOT	amount
RATH	analysis
RATH	anaphoric reference
NOT	angle
POSS	approach
POSS	appropriate information
POSS	appropriate interpretation
POSS	appropriate module
NOT	aspect
RATH	assertion
POSS	assumption
UNKN	asterisk
NOT	attempt
UNKN	attention
UNKN	author
TECH	automatic acquisition
TECH	auxiliary
POSS	available grammar
POSS	availspies list
NOT	bad case
RATH	base manager
POSS	base
RATH	basic construction
UNKN	basic deduction
POSS	basic form
UNKN	basis
NOT	beginning
NOT	belief
UNKN	binding
RATH	bottom-up parsing
UNKN	brace
UNKN	bracket
UNKN	brief explanation
UNKN	build block
RATH	ca
BAD	call kate
POSS	call
RATH	candidate substring
BAD	can
POSS	capitalization
POSS	capital letter
RATH	case analyzer
RATH	case grammar
RATH	case representation
POSS	case
POSS	category
POSS	cause fragment
UNKN	cause
NOT	central part
POSS	certain form
POSS	certain input
NOT	certain kind
POSS	certain rule
NOT	certain situation
POSS	certain structure
POSS	certain substring
POSS	certain token
POSS	certain word
UNKN	chance
UNKN	chapter
BAD	characterization can
RATH	chart parsing
POSS	check
POSS	chronological order

NOT	claim
UNKN	clarification
RATH	clause grammar
RATH	clause
UNKN	clear-cut distinction
UNKN	close communication
UNKN	cluster
RATH	code
POSS	cognitive purpose
UNKN	cognos
UNKN	come month
UNKN	comma
POSS	command
UNKN	comparator
UNKN	comparison
TECH	complement
POSS	complete paragraph
RATH	complete sentence
UNKN	complex machinery
POSS	complex parser
RATH	complex semantics
RATH	complex structure
POSS	complex task
POSS	complicate input
RATH	component
UNKN	compositionality
RATH	compound sentence
NOT	comprehensible description
POSS	computerize description
TECH	conceptual analyzer
RATH	conceptual model
RATH	conceptual network
UNKN	concern
UNKN	condition
RATH	conjoined constituent
RATH	conjoined element
RATH	conjoined item
TECH	conjoined sentence
RATH	conjoined structure
RATH	conjoined unit
RATH	conjunction
NOT	consideration
RATH	constituent
POSS	constraint
POSS	construction
NOT	contents
TECH	context-free grammar
UNKN	contraction
UNKN	convenience
RATH	core grammar
POSS	core
UNKN	correctness
RATH	correct parse
POSS	correspond entry
UNKN	coupling
POSS	coverage
UNKN	cpu
POSS	crucial component
UNKN	current emphasis
POSS	current focus
POSS	current position
NOT	current remainder
NOT	current version
POSS	dash
UNKN	datum
RATH	dcg formalism
RATH	dcg grammar
RATH	dcg implementation
RATH	dcg
UNKN	dct
POSS	debug purpose
UNKN	decision
POSS	default mode
BAD	define
POSS	definition
NOT	delicate problem
NOT	description
POSS	design
POSS	desirable feature
POSS	detail information
NOT	detail
UNKN	detection
UNKN	development
UNKN	diagram
RATH	dialogue-related phenomenon
RATH	dictionary addition
RATH	dictionary
UNKN	dict
POSS	didactic purpose
NOT	different part
POSS	different phenomenon
NOT	different purpose
NOT	difficult aspect
NOT	difficult issue
NOT	difficult problem
BAD	difficulty here
NOT	difficulty
RATH	discontinuity
RATH	discontinuous constituent

POSS	discontinuous grammar
POSS	discourse consideration
RATH	discourse element
RATH	discourse model
RATH	discourse
RATH	discourse-related phenomenon
UNKN	discrepency
NOT	discussion
UNKN	disposal
POSS	distinct category
POSS	distinct mode
POSS	document
RATH	domain-independent grammar
RATH	domain-independent subset
TECH	domain knowledge
POSS	domain
TECH	domain text
UNKN	double quote
UNKN	drastic impact
RATH	driver
UNKN	ease
NOT	easy solution
POSS	efficiency
POSS	efficient solution
UNKN	efficient vehicle
NOT	efficient way
POSS	element
POSS	ellipsis
POSS	empty clause
POSS	endless computation
POSS	endless loop
NOT	end
RATH	english grammar
RATH	english input
RATH	english sentence
POSS	english
BAD	entry can
POSS	entry
UNKN	enumeration
POSS	error-flag-type parameter
POSS	evaluation
RATH	evaluation process
POSS	example input
UNKN	example log
POSS	example sentence
POSS	example session
POSS	example
UNKN	exception
POSS	excerpt
POSS	excessive ambiguity
RATH	existential construction
POSS	exist file
POSS	exit command
POSS	expansion facility
NOT	experience
RATH	experimental text
RATH	explicit reference
POSS	explicit representation
UNKN	exponential cost
POSS	expressive power
POSS	extend knowledge
RATH	extra-grammatical input
POSS	extraposition grammar
RATH	extraposition
NOT	extra step
NOT	extreme case
NOT	extreme situation
RATH	fact
UNKN	failure
POSS	feature
UNKN	feedback
UNKN	field
POSS	file
RATH	find-all-the-parses approach
UNKN	flexible access
POSS	focus facility
POSS	focus
NOT	follow assumption
UNKN	follow drawback
POSS	follow excerpt
UNKN	follow member
POSS	follow parse
NOT	follow subsection
POSS	follow type
UNKN	force
POSS	formal rule
POSS	form
POSS	fragment
POSS	free mode
POSS	free segment
UNKN	frequency
POSS	frequent type
POSS	full integration
RATH	function word
UNKN	fundamental characteristic

POSS	future work
NOT	future
POSS	generic type
RATH	gerundive construction
POSS	give input
POSS	give parse
NOT	give time
BAD	goal here
RATH	goal
NOT	good case
NOT	good idea
NOT	good way
POSS	grammar definition
POSS	grammar description
RATH	grammar formalism
RATH	grammar rule
TECH	grammar
RATH	graph-structured stack
NOT	great difficulty
UNKN	group
NOT	guarantee
RATH	head symbol
TECH	helper mechanism
POSS	helper
POSS	help
POSS	high level
NOT	hoc fashion
NOT	hole
POSS	human reader
NOT	human
NOT	idea
POSS	identical entry
UNKN	identical head
RATH	idiomatic construction
UNKN	illustration
POSS	implementation
BAD	implementer can
POSS	implementer
RATH	implicit complement
RATH	implicit knowledge
RATH	implicit subject
NOT	importance
POSS	important assumption
POSS	important element
POSS	important hypothesis
UNKN	important inadequacy
POSS	important issue
NOT	important modification
NOT	important one
POSS	important role
NOT	indication
RATH	inference mechanism
UNKN	inference
RATH	inflect form
POSS	info
POSS	information
RATH	input
RATH	input rejection
RATH	input string
TECH	input text
UNKN	insertspy
POSS	instance
POSS	instruction
NOT	insubstantial example
POSS	integration
UNKN	integrator
NOT	intelligent type
NOT	intelligent way
UNKN	intention
RATH	interaction manager
POSS	interaction
NOT	interesting approach
NOT	interesting question
NOT	interest
RATH	interface
POSS	intervention
NOT	investigation
UNKN	isolation
POSS	issue
UNKN	italics
POSS	item
POSS	joining-point
RATH	kate
RATH	kate system
NOT	kind
UNKN	know complexity
TECH	knowledge acquisition
RATH	knowledge-based text
UNKN	knowledge engineer
POSS	knowledge processor
RATH	knowledge
POSS	language grammar
POSS	language input
RATH	language parsing
POSS	language phenomenon
UNKN	language utterance
POSS	language
POSS	large grammar
UNKN	larger slowdown
POSS	larger substring
POSS	latter option
NOT	latter
RATH	legal particle
RATH	legal substring

POSS	length position
POSS	level
TECH	lexical analysis
TECH	lexical analyzer
RATH	lexical category
RATH	lexical entry
RATH	lexical information
RATH	lexical item
UNKN	lieu
POSS	limitation
UNKN	limit
UNKN	line
RATH	linguistic component
RATH	linguistic coverage
POSS	linguistic perspective
UNKN	linguistics
POSS	list
RATH	logic grammar
RATH	long-distance dependency
POSS	longer input
POSS	long sentence
TECH	look-ahead mechanism
UNKN	look-ahead
POSS	loosely-coupled integration
NOT	low-level detail
RATH	low-level non-terminal
RATH	lr parsing
POSS	main category
POSS	main component
UNKN	main justification
POSS	main menu
POSS	main module
NOT	main point
NOT	main reason
POSS	main rule
UNKN	main subsystem
TECH	main verb
NOT	major aspect
POSS	major component
NOT	major improvement
NOT	majority
POSS	major non-terminal
UNKN	major observation
NOT	major step
POSS	major substring
NOT	manner
POSS	marker
UNKN	matcher
NOT	maximum amount
NOT	maximum number
POSS	meaningful information
POSS	meaning
TECH	mechanism
RATH	memorize device
POSS	memorizer
RATH	memory assertion
POSS	memory
UNKN	menu-based environment
POSS	menu
UNKN	message
UNKN	mind
NOT	minor improvement
UNKN	minute
UNKN	modal
POSS	model
POSS	modification
RATH	modifier
POSS	modify sentence
POSS	modify version
RATH	module
TECH	morphological analysis
RATH	morphological process
RATH	movement
TECH	multiple parse
NOT	name
NOT	natural description
TECH	natural language
UNKN	naturalness
TECH	natural text
UNKN	nature
NOT	need
UNKN	negation
POSS	network
UNKN	new tendency
POSS	nlp literature
TECH	nlp
UNKN	no-crossing branch
POSS	non-standard component
POSS	non-terminal name
POSS	non-terminal sentence
RATH	non-textual element
UNKN	normal circumstances
POSS	normal input
NOT	note
UNKN	nounphrase
TECH	noun phrase
TECH	noun
POSS	number
UNKN	object

UNKN	occasion
UNKN	occurrence
UNKN	offer
NOT	one
POSS	operate system
POSS	operation
RATH	optional mechanism
POSS	option
POSS	order
NOT	organization
POSS	original form
POSS	original list
POSS	original string
RATH	original text
UNKN	orthographic bit
POSS	orthography
UNKN	ottawa
UNKN	outcome
NOT	own information
POSS	paragraph
POSS	parameter
NOT	paramount importance
POSS	parenthesis
RATH	parsable input
RATH	parsable sentence
UNKN	parse algorithm
RATH	parse efficiency
RATH	parse input
RATH	parse
POSS	parser attempt
TECH	parser
TECH	parse session
RATH	parse system
RATH	parse technique
TECH	parse tree
TECH	parsing
RATH	particle
RATH	particle table
POSS	part
UNKN	past
NOT	people
NOT	perennial problem
POSS	performance indication
POSS	performance
POSS	performance requirement
RATH	permanent dictionary
RATH	permissive grammar
NOT	perspective
NOT	pervasive problem
UNKN	phase-marker
RATH	phenomenon
POSS	phrase marker
RATH	phrase structure
UNKN	place
POSS	plan goal
UNKN	plan
TECH	plural noun
BAD	point here
NOT	point
UNKN	politeness convention
UNKN	portion
NOT	possibility
NOT	possible improvement
POSS	possible parse
POSS	possible phenomenon
NOT	possible situation
UNKN	potential extension
RATH	potential joining-point
POSS	potential parse
POSS	potential sentence
UNKN	potential utility
POSS	potential verb
POSS	practical nlp
NOT	practical point
NOT	precede part
NOT	precede subsection
POSS	precise coverage
NOT	predetermine amount
RATH	predicate
RATH	preference semantics
POSS	preliminary version
UNKN	preposing
TECH	prepositional phrase
RATH	preposition
UNKN	presence
POSS	presentation
NOT	present detail
NOT	present difficulty
POSS	present document
POSS	present form
RATH	pre-syntactic task
NOT	previous one
NOT	previous subsection
POSS	priori knowledge
UNKN	priori
POSS	problematic input
NOT	problematic issue
POSS	problematic substring
POSS	problematic word
NOT	problem
POSS	process input
POSS	process
POSS	process power

POSS	production
RATH	program code
POSS	programming language
UNKN	program output
RATH	prolog assertion
POSS	prolog description
RATH	prolog fact
RATH	prolog implementation
RATH	prolog
RATH	prolog predicate
TECH	proper noun
UNKN	property
POSS	punctuation mark
NOT	purpose
RATH	quantifier
NOT	question
UNKN	quintus
POSS	quiz command
POSS	quiz example
BAD	reader can
POSS	reader
NOT	read section
UNKN	real function
POSS	realistic strategy
NOT	reasonable amount
NOT	reason
NOT	recent note
POSS	recent rule
UNKN	recent year
POSS	reference
POSS	reference problem
TECH	regular verb
BAD	rejection can
POSS	rejection
RATH	relative clause
UNKN	relativizer
NOT	relax description
UNKN	relevancy
POSS	relevant information
POSS	relevant parse
POSS	relevant predicate
POSS	relevant work
POSS	remainder
NOT	remark
UNKN	removal
NOT	renew interest
UNKN	report generator
POSS	representation
POSS	representative example
POSS	representative input
POSS	representative sample
POSS	requirement
UNKN	research initiative
UNKN	research team
UNKN	respect
NOT	rest
POSS	rich syntax
UNKN	right-to-left direction
POSS	role
RATH	root form
UNKN	root
RATH	rule parameter
TECH	rule
TECH	sample text
RATH	scanner
UNKN	scope
UNKN	scratch
UNKN	search
UNKN	second
POSS	section
POSS	selective mechanism
TECH	semantic analysis
UNKN	semantic attachment
RATH	semantic component
RATH	semantic grammar
RATH	semantic information
RATH	semantic model
TECH	semantic module
POSS	semantic phenomenon
RATH	semantics
UNKN	semantic viewpoint
UNKN	sense
RATH	sentence modifier
RATH	sentence parse
RATH	sentence structure
TECH	sentence
NOT	separate part
POSS	sequence
UNKN	sequential read
NOT	serious difficulty
RATH	session
POSS	similar phenomenon
POSS	simple grammar
UNKN	simple heuristic
POSS	simple mechanism
POSS	simple reference
RATH	simple sentence
POSS	simple statistics
NOT	simple strategy
POSS	simplify version
RATH	single parse
POSS	single sentence
POSS	single token

UNKN	single transcript
NOT	situation
UNKN	so-called idiom
UNKN	software package
POSS	source
UNKN	special flag
POSS	special meaning
POSS	special mode
POSS	special treatment
UNKN	specification
UNKN	specific individual
POSS	specific location
POSS	specific non-terminal
POSS	specific word
UNKN	speech act
RATH	speech category
POSS	speech
UNKN	speedy recognition
UNKN	spirit
POSS	stack
RATH	stand-alone fragment
UNKN	standard
POSS	state
POSS	straightforward interpretation
POSS	strategy
UNKN	strict pattern
POSS	string
RATH	structure
RATH	structure rule
POSS	subcategory label
POSS	subjective constraint
TECH	subject
NOT	subsection
NOT	subsequent level
POSS	subset
POSS	substantial subset
RATH	substring
POSS	substring table
RATH	substructure
UNKN	sun
UNKN	superset
POSS	supplementary information
RATH	surface-syntactic dictionary
UNKN	surface-syntactic manifestation
UNKN	surface
POSS	symbolic name
POSS	symbol
UNKN	synonym
TECH	syntactic analysis
TECH	syntactic analyzer
RATH	syntactic construction
RATH	syntactic coverage
RATH	syntactic information
POSS	syntactic limitation
TECH	syntactic module
RATH	syntactic phenomenon
TECH	syntactic structure
RATH	syntactic substructure
POSS	syntactic type
RATH	syntactic unit
RATH	syntax
RATH	syntax semantics
BAD	system can
POSS	system
POSS	table
RATH	tabular parsing
POSS	task
RATH	tbm
RATH	tbm scanner
RATH	technical element
TECH	technical text
POSS	technique
RATH	tense information
UNKN	terminology
POSS	term
UNKN	textbase
UNKN	textbook
TECH	text
RATH	text sample
POSS	theoretical consideration
UNKN	there-subject
UNKN	thing
POSS	thorough presentation
UNKN	thought
UNKN	time check-point
POSS	time constraint
POSS	time device
RATH	time mechanism
POSS	time
UNKN	timing
POSS	title
POSS	token
POSS	tool
POSS	top-down fashion
UNKN	topicalization
POSS	topic
UNKN	trace control
RATH	trace facility
RATH	trace instruction

TECH	trace mechanism
RATH	transitive construction
POSS	treatment
RATH	tree
POSS	trivial task
NOT	type
POSS	typical input
POSS	typical parse
NOT	typical situation
POSS	ultimate goal
UNKN	undue delay
RATH	ungrammatical input
RATH	ungrammatical string
POSS	uninformative element
POSS	uninformative part
POSS	unit
UNKN	university
POSS	unknown part
TECH	unknown word
POSS	unordered list
NOT	unpleasant operation
POSS	unrecognizable part
POSS	upcoming integration
UNKN	useful comment
POSS	useful parser
POSS	useless computation
BAD	user can
POSS	user input
RATH	user interaction
RATH	user interface
RATH	user intervention
UNKN	user manual
RATH	user
NOT	use
POSS	usual location
POSS	usual sentence
POSS	valid input
POSS	various statistics
TECH	verbal clause
POSS	verbal part
UNKN	verbphrase
TECH	verb phrase
UNKN	verb report
TECH	verb sequence
TECH	verb
UNKN	verification
NOT	view
NOT	way
UNKN	well-established methodology
RATH	well-formed substring
POSS	well-known fact
NOT	well-known problem
POSS	well-known subset
UNKN	well-written technical-text
POSS	window
POSS	word problem
POSS	word sentence
RATH	word
POSS	work
POSS	wrong rule

Appendix 8
Classification by Block Results
QUIZ Chapters 1-6

TECHNICAL:

employee number
control heading
heading
footing report-group
final footing
column heading
file
option
page heading
number
define item
example
line
item
invoice number
statement
report
value
quiz

RATHER TECHNICAL:

installation tape
fictitious company
computer terminal
easy-to-read format
item lastname
boston staff
boston employee
select statement
ampersand symbol
application designer
entire file
allowable entry
option keyword
access datum
employee file
one-to-one linkage
one-to-many situation
primary file
item employee
billing file
unnecessary repetition
link file
item heading
temporary file
precede statement
currency symbol
complex statement
available option
formatting option
heading report-item
default column
one-time heading
extra feature
blank report-item
textual heading
needless repetition
report-item name
useful option
widowed heading
widowed footing
footing
interactive session
permanent file
quiz statement
summary information
invoice-master file
default format
summary operation
repetitive printing
sort-key option
interim item
percent symbol
item billing
format option
date-type item
option can
extract function
conversion statement
function
recurring requirement
automatic feature
quiz session
crt terminal
default heading
source file
designate file
text file
text editor

terminal screen
execution-time parameter
source statement
submission batch
heading feature
employee
report-group
report-item
control footing
age-date prompt

POSSIBLY TECHNICAL:

basic rule
phone number
sample report
canadian city
introductory message
entry
lowercase letter
read datum
select information
demonstration system
uppercase
uppercase letter
file name
component
y key
system manager
tape
data
staff
manual
actual datum
employee information
prompt
instruction
alphabetical list
new statement
ascending order
recent employee
selective report
qualifier
statement line
ampersand
compound statement
certain item
smith
select record
specific value
large file
process time
advance feature
specific part
dallas office
sophisticated report
particular statement
quiz list
special statement
prompt character
low level
return key
introductory explanation
keyword
single file
alphabetical order
list
linkage
match item
full discussion
dictionary manual
access
key
example access
margaret
position file
unique key
repeat key
abra
complex
default
screen
problem
special instruction
format report
standard heading
datum dictionary
maximum number
blank space
quiz start
own report-item
tabs work
quiz use
report statement
styled heading
portion
actual heading
quotation mark
symbol
position report-item
own heading
column

footing can
report item
employee report
major division
branchname change
boston
dallas
statement cause
professional-looking report
printing
branchname
control-heading line
associate detail
new report-group
blank line
session can
statement can
editor command
file can
editor
edited contents
editfile
employee name
use option
billing report
numeric value
billing
item balance
open balance
transaction
initial value
subtotal
branch lastname
break
run subtotal
branch report-item
unique branch
mislead count
subtotal value
invoice-amount
detail item
invoice amount
customer
operation
complete syntax
situation
control break
basis
sort-key
key item
next-to-last place
project change
next-to-first place
unique detail
new data
information
own item
combination
combine item
firstname
quiz need
detail
asterisk
record complex
negative number
date format
six-digit date
powerhouse date
eight-digit date
default date
item type
different format
format
default picture
extra position
series
syntax
fix value
languagetext
routine
occurrence
string function
actual text
parenthesis need
function can
record
surname
test-pattern
trailing space
desire surname
numeric expression
right portion
day employee
datejoined
can use
procedure
age routine
century prefix
male employee
column number
summary
female employee
total employee
chapter

quiz can
can
label form
terminal
specific form
printer
two-up form
three-up form
eight-line page
status control
default value
complex report
label
process can
computer
run quiz
parse time
quiz report
source
complete instruction
instance
company
requirement
time
execution time
mail label
branch
feature
branch code
selection condition
xx
entire list
user
discussion
compile report
batch
efficient way
project
name
two-level report
employee change
elaborate use
comment line
semicolon
itemize detail
one-level report
new page
invoice can
page
text
associate condition
conclusion
invoice example
invoice total
fix position
invoice
fast response
use
inventory
response
task
datum
report requirement

NOT TECHNICAL:

question
right question
office
command
control key
follow message
answer
new report
city
descending order
produce report
short report
equal sign
start
word
key information
number name
small amount
key value
group
question mark
need information
can work
various level
help system
help
explanation
need
match
system
complete list
compound record
single report
reason
position
easy way

result report
top line
amount
new line
simple operation
thing
position line
future report
dollar sign
standard page
title page
work
beginning
branch start
page break
break can
general rule
control
room
detail information
branch name
level
grand total
count record
control group
record count
sale tax
run count
start value
balance
close balance
total number
desire value
tax
detail amount
every time
general principle
planned report
great detail
least detail
principle
change value
month change
branch change
place
particular report
cause
way
certain thing
purpose
special character
sign
string expression
matter
type
full explanation
type size
lead sign
decimal point
language
condition
particular operation
title
follow page
expression can
size
expression
follow example
pattern
name beginning
character
space
can match
month number
division
result
day
date
dictionary
important information
good example
page width
page length
point
contents
manner
organization can
order
message
mail list
report can
complete detail
different result
form letter
month
end
part
string
body
detail line
page number
long list
hand
total

change
effect
letter example
length
letter
bottom
form
right information
person
minute
follow report
date can
case
production report

TECHNICALITY UNKNOWN:

simplicity
relationship
standard
element
check
powerhouse
keyboard
good idea
policy
interruption
future industry
michael
personnel
select
abbreviation
fact
large quantity
collection
people
payroll
dot
flag
item sex
go
statement draw
range
female
quiz issue
difficulty here
alot
maximum
display
subject
stage
chance
filename
add extra
new term
effort
readability
typewriter
experimentation
report divisionname
optional specification
debit indicator
zero
specific section
option n
interesting exercise
solution
suspend
edit
average
choice
money
invoice-master
invoice-detail
discount
inclusive category
hyphen
blank
yymmdd
yyyymmdd
item datatype
xbilling
country
prov-state
list description
center
parenthesis
specific limit
remainder
equation
january
substance
eight-digit-date
six-digit-date
cross-tabulation
percentage
respect
company organization
return
report image
operator
switch

statement influence
alternative approach
prototype
printout
process
table
parse
own security
source version
case modification
knowledge
mail
act
report consist
benefit
job
view
break structure
heading-body-footing concept
breadth
trouble
invoice-number
provision
box
real genius
good decision
ease
comptroller
overdue account
president
wide variety
on-line inquiry
ingenuity

1978 total NPs

Appendix 9
Classification by Block Results
Delisle's Parser Text

TECHNICAL:

helper mechanism
trace mechanism
syntactic module
look-ahead mechanism
parse session
multiple parse
semantic module
prepositional phrase
syntactic phenomenon
input
rule
conjoined sentence
sentence
technical text
grammar
parser
text

RATHER TECHNICAL:

crucial component
experimental text
knowledge-based text
explicit reference
automatic acquisition
basic construction
sample text
complex task
trivial task
complex semantics
parse technique
lr parsing
graph-structured stack
programming language
well-known subset
linguistic coverage
domain-independent subset
text sample
major component
appropriate module
syntactic analyzer
non-standard component
expansion facility
subcategory label
legal particle
additional source
idiomatic construction
syntactic construction
pre-syntactic task
morphological process
lexical analysis
identical entry
morphological analysis
unknown word
prolog fact
permanent dictionary
straightforward interpretation
syntactic unit
linguistic perspective
prolog implementation
prolog predicate
low-level non-terminal
memorize device
well-formed substring
prolog assertion
bottom-up parsing
chart parsing
larger substring
candidate substring
performance indication
useful parser
major substring
problematic substring
modify version
input rejection
ungrammatical input
extra-grammatical input
problematic issue
acceptable input
upcoming integration
user input
session
major aspect
acquisition facility
complex parser
trace facility
user interface
relevant predicate
trace instruction
specific non-terminal

example session
syntactic substructure
lexical analyzer
minor improvement
tbm scanner
loosely-coupled integration
substantial subset
discourse element
legal substring
tabular parsing
alternative approach
error-flag-type parameter
major improvement
interaction manager
user interaction
simplify version
dcg implementation
major non-terminal
optional mechanism
technical element
parse efficiency
conjoined unit
conjoined constituent
conjoined structure
conjoined item
lexical category
potential joining-point
verbal clause
mechanism
parse input
useless computation
implicit complement
parsable input
appropriate interpretation
complicate input
selective mechanism
semantic component
syntactic analysis
semantic information
syntax semantics
preference semantics
find-all-the-parses approach
relevant parse
evaluation process
realistic strategy
endless computation
potential parse
extreme situation
subjective constraint
allot time
user
parse
syntactic structure
endless loop
ultimate goal
natural text
valid input
original text
stand-alone fragment
semantic analysis
verb phrase
non-textual element
program code
specific location
normal input
distinct mode
fragment
discourse consideration
precede subsection
semantic phenomenon
previous subsection
process input
reference problem
anaphoric reference
explicit representation
precise coverage
well-known problem
efficient solution
default mode
desirable feature
longer input
focus facility
discourse-related phenomenon
approach
inference mechanism
dialogue-related phenomenon
syntactic coverage
user intervention
process power
similar phenomenon
lexical item
correspond entry
parse tree
uninformative element
domain text
complex structure
conjoined element
transitive construction
excessive ambiguity
noun phrase
existential construction
gerundive construction
pervasive problem
phenomenon

problematic input
syntactic limitation
long-distance dependency
grammar rule
discontinuous constituent
performance requirement
perennial problem
expressive power
linguistic component
module
conceptual analyzer
parsing
natural language
kate system
dcg formalism
main module
section
lexical entry
distinct category
theoretical consideration
acquisition task
preliminary version
input text
kate

POSSIBLY TECHNICAL:

discussion
call kate
conceptual model
important assumption
paramount importance
representation
semantic model
rich syntax
ambiguity
language input
technique
context-free grammar
language phenomenon
characterization can
insubstantial example
hoc fashion
core grammar
nlp literature
addition modification
definition
grammar definition
parse system
driver
debug purpose
surface-syntactic dictionary
function word
info
basic form
speech category
source
category
particle
particle table
single token
excerpt
task
inflect form
original list
entry can
original form
appropriate information
unit
follow excerpt
symbol
actual session
file
central part
efficient way
relevant work
dcg grammar
representative sample
parsable sentence
prolog description
remainder
syntactic type
goal here
actual implementation
symbolic name
read section
subsequent level
menu
assertion
syntactic information
phrase structure
substring table
process
simple statistics
meaningful information
give input
chronological order
stack
follow assumption
top-down fashion
bad case
previous one

certain substring
original string
different phenomenon
rejection
structure rule
easy solution
main menu
sentence parse
exist file
modify sentence
exit command
various statistics
difficult aspect
implementation
implementer can
parameter
requirement
instruction
availspies list
available grammar
implementer
predicate
activespy list
active non-terminal
non-terminal sentence
non-terminal name
current remainder
possible improvement
presentation
full integration
entry
tbm
word problem
unknown part
intervention
important modification
supplementary information
scanner
ca
well-known fact
difficulty here
domain-independent grammar
modification
intelligent way
unpleasant operation
difficult issue
memorizer
helper
memory assertion
performance
core
dcg
computerize description
natural description
subset
language
reader can
comprehensible description
low-level detail
rule parameter
relax description
permissive grammar
specific word
head symbol
typical input
check
input string
major step
potential sentence
verbal part
joining-point
parser attempt
potential verb
possible situation
typical situation
implicit subject
complement
verb sequence
auxiliary
item
representative input
possible phenomenon
actual parse
interface
follow parse
sentence modifier
typical parse
precede part
important element
human reader
simple mechanism
analysis
option
latter option
certain input
possible parse
interaction
integration
semantic grammar
correct parse
evaluation
difficult problem
time mechanism
subsection

reasonable amount
guarantee
conjunction
serious difficulty
predetermine amount
single parse
give parse
maximum amount
maximum number
time constraint
present difficulty
delicate problem
constraint
difficulty
case analyzer
special mode
tree
extra step
extreme case
time device
operate system
assumption
element
majority
cause fragment
sentence structure
simple strategy
discourse model
quiz example
role
hole
phrase marker
marker
feature
high level
follow subsection
syntax
semantics
thorough presentation
complete paragraph
simple reference
extend knowledge
main component
case representation
cognitive purpose
focus
indication
current focus
conceptual network
free mode
modifier
relative clause
problem
system
investigation
time
paragraph
free segment
sequence
intelligent type
knowledge processor
plan goal
implicit knowledge
important hypothesis
priori knowledge
main category
coverage
reader
account capitalization
relevant information
lexical information
base manager
problematic word
capitalization
quiz command
parenthesis
dash
ellipsis
consideration
angle
information
punctuation mark
certain structure
certain situation
situation
example input
domain knowledge
orthography
token
certain token
fact
generic type
uninformative part
didactic purpose
goal
example
part
perspective
knowledge acquisition
user can
unrecognizable part
empty clause
important role
unordered list

reference
great difficulty
quantifier
clause
construction
substructure
usual location
rejection can
word
substring
constituent
english input
term
discontinuity
preposition
movement
extraposition
issue
structure
nlp
renew interest
grammar formalism
interesting approach
extraposition grammar
discontinuous grammar
logic grammar
strategy
clause grammar
instance
language parsing
efficiency
ungrammatical string
order
domain
discourse
aspect
important issue
topic
limitation
current version
component
prolog
code
dictionary
present document
practical nlp
acquisition
belief
design
document
access language

NOT TECHNICAL:

system can
good way
meaning
manner
level
idea
language grammar
compound sentence
simple grammar
point here
grammar description
tense information
different part
important one
plural noun
regular verb
root form
addition
main point
dictionary addition
main reason
description
good idea
name
english sentence
length position
reason
wrong rule
state
recent note
good case
recent rule
formal rule
memory
main rule
one
give time
base
own information
certain word
capital letter
command
importance
usual sentence
present form
note
long sentence
large grammar

window
english grammar
english
people
actual grammar
detail
current position
different purpose
separate part
noun
simple sentence
attempt
string
certain rule
subject
main verb
point
purpose
number
practical point
amount
production
type
beginning
actual number
human
call
table
case grammar
verb
complete sentence
frequent type
special meaning
possibility
end
present detail
use
network
title
certain kind
rest
operation
single sentence
advantage
contents
need
help
speech
detail information
way
special treatment
interesting question
proper noun
work
can
kind
latter
example sentence
question
treatment
follow type
certain form
experience
list
representative example
claim
form
model
organization
future work
case
word sentence
interest
view
knowledge
future
tool
remark

TECHNICALITY UNKNOWN:

main subsystem
plan
acquisition laboratory
university
ottawa
thing
sequential read
main justification
priori
absence
come month
know complexity
exponential cost
exception
parse algorithm
standard
textbook
synonym
diagram
well-established methodology
define

scratch
follow drawback
superset
discrepency
root
frequency
contraction
example log
portion
convenience
dict
compositionality
fundamental characteristic
language utterance
property
linguistics
efficient vehicle
surface
brief explanation
failure
build block
spirit
offer
clarification
basis
group
identical head
simple heuristic
right-to-left direction
potential utility
textbase
flexible access
chapter
close communication
isolation
specification
binding
special flag
trace control
insertspy
nounphrase
verbphrase
detection
software package
adjustment
limit
lieu
nature
undue delay
account
comparison
larger slowdown
minute
menu-based environment
naturalness
dct
object
research initiative
past
attention
chance
verification
place
advance
look-ahead
speedy recognition
modal
adverb
drastic impact
message
surface-syntactic manifestation
asterisk
single transcript
timing
current emphasis
semantic viewpoint
force
mind
relevancy
search
feedback
concern
sense
respect
correctness
time check-point
enumeration
removal
italics
illustration
well-written technical-text
semantic attachment
disposal
program output
normal circumstances
condition
outcome
thought
clear-cut distinction
integrator
matcher
cluster
negation
scope

speech act
intention
inference
decision
major observation
complex machinery
abbreviation
orthographic bit
verb report
comma
brace
bracket
double quote
specific individual
basic deduction
real function
ease
politeness convention
strict pattern
presence
so-called idiom
occasion
terminology
comparator
relativizer
topicalization
preposing
there-subject
cause
no-crossing branch
phase-marker
occurrence
author
recent year
field
advent
important inadequacy
coupling
potential extension
quintus
line
second
cpu
sun
new tendency
knowledge engineer
follow member
research team
useful comment
cognos
user manual
report generator
datum
development

2030 total NPs