ACQUIRING DISAMBIGUATION RULES FROM TEXT

Donald Hindle
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974-2070

Abstract

An effective procedure for automatically acquiring a new set of disambiguation rules for an existing deterministic parser on the basis of tagged text is presented. Performance of the automatically acquired rules is much better than the existing hand-written disambiguation rules. The success of the acquired rules depends on using the linguistic information encoded in the parser; enhancements to various components of the parser improves the acquired rule set. This work suggests a path toward more robust and comprehensive syntactic analyzers.

1 Introduction

One of the most serious obstacles to developing parsers to effectively analyze unrestricted English is the difficulty of creating sufficiently comprehensive grammars. While it is possible to develop toy grammars for particular theoretically interesting problems, the sheer variety of forms in English together with the complexity of interaction that arises in a typical syntactic analyzer makes each enhancement of parser coverage increasingly difficult. There is no question that we are still quite far from syntactic analyzers that even begin to adequately model the grammatical variety of English. To go beyond the current generation of hand built grammars for syntactic analysis it will be necessary to develop means of acquiring some of the needed grammatical information from the regularities that appear in large corpora of naturally occurring text.

This paper describes an implemented training procedure for automatically acquiring symbolic rules for a deterministic parser on the basis of unrestricted textual input. In particular, I describe experiments in automatically acquiring a set of rules for disambiguation of lexical category (part of speech). Performance of the acquired rule set is much better than the set of rules for lexical disambiguation written for the parser by hand over a period of several rules; the error rate is approximately half that of the hand written rules. Furthermore, the error rate is comparable to recent probabilistic approaches such as Church (1987) and Garside, Leech and Sampson (1987). The current approach has the added advantage that, since the rules acquired depend on the parser's grammar in general, independent improvements in other modules of the parser can lead to improvement in the performance of the disambiguation component.

2 Categorial Ambiguity

Ambiguity of part of speech is a pervasive characteristic of English; more than a third of the word tokens in the million-word "Brown Corpus" of written English (Francis and Kucera 1982) are categorially ambiguous. It is possible to construct sentences in which every word is ambiguous, such as the following,

(1) Her hand had come to rest on that very book.

But even without such contrived exaggeration, ambiguity of lexical category is not a trivial problem. Nor can part of speech ambiguity be ignored in constructing models of natural language processing, since syntactic analysis (as well as higher levels of analysis) depends on correctly disambiguating the lexical category of both content words and function words like *to* and *that*.

It may seem that disambiguating lexical category should depend on complex reasoning about a variety of factors known to influence ambiguity in general, including semantic and pragmatic factors. No doubt some aspects of disambiguating lexical category can be expressed in terms of such higher level decisions. But if disambiguation in fact depends on such higher level reasoning, there is little hope of succeeding in disambiguation on unrestricted text.

Fortunately, there is reason to believe that lexical disambiguation can proceed on more limited syntactic patterns. Indeed, recent increased interest in the problem of disambiguating lexical category in English has led to significant progress in developing effective programs for assigning lexical category in unrestricted text. The most successful and comprehensive of these are based on probabilistic modeling of category sequence and word category (Church 1987; Garside, Leech and Sampson 1987; DeRose 1988). These stochastic methods show impressive performance: Church reports a success rate of 95 to 99%, and shows a sample text with an error rate of less than one percent. What may seem particularly surprising is that these methods succeed essentially without reference to syntactic structure; purely surface lexical patterns are involved. In contrast to these recent stochastic methods, earlier methods based on categorical rules for surface patterns achieved only moderate success. Thus for example, Klein and Simmons (1963) and Greene and Rubin (1971) report success rates considerably below recent stochastic approaches.

It is tempting to conclude from this contrast that robust handling of unrestricted text demands general probabilistic methods in preference to deeper linguistic knowledge. The Lancaster (UCREL) group explicitly takes this position, suggesting: "... if we analyse quantitatively a sufficiently large amount of language data, we will be able to compensate for the computer's lack of sophisticated knowledge and powers of inference, at least to a considerable extent." (Garside, Leech and Sampson 1987:3).

In this paper, I want to emphasize a somewhat different view of the role of large text corpora in building robust models of natural language. In particular, I will show that that large corpora of naturally occurring text can be used together with the rule-based syntactic analyzers we have today to build more effective linguistic analyzers. As the information derived from text is incorporated into our models, it will help increase the sophistication of our linguistic models. I suggest that in order to move from our current impoverished natural language processing systems to more comprehensive and robust linguistic models we must ask *Can we acquire the linguistic information needed on the basis of text?* If we can answer this question affirmatively – and this paper presents evidence that we can – then there is hope that we can make some progress in constructing more adequate natural language processing systems.

It is important to emphasize that the question whether we can acquire linguistic information from text is independent of whether the model is probabilistic, categorical, or some combination of the two. The issue is not, I believe, symbolic versus probabilistic rules, but rather whether we can acquire the necessary linguistic information instead of building systems completely by hand. No algorithm, symbolic or otherwise, will succeed in large scale processing of natural text unless it can acquire some of the needed knowledge from samples of naturally occurring text.

3 Lexical Disambiguation in a Deterministic Parser

The focus of this paper is the problem of disambiguating lexical category (part of speech) within a deterministic parser of the sort originated by Marcus (1980). Fidditch is one such deterministic parser, designed to provide a syntactic analysis of text as a tool for locating examples of various linguistically interesting structures (Hindle 1983). It has gradually been modified over the past several years to improve its ability to handle unrestricted text.

Fidditch is designed to provide an annotated surface structure. It aims to build phrase structure trees, recovering complement relations and gapped elements. It has

- a lexicon of about 100,000 words listing all possible parts of speech for each word, along with root forms for inflected words.
- a morphological analyzer to assign part of speech and root form for words not in the lexicon
- a complementation lexicon for about 4000 words
- a list of about 300 compound words, such as *of course*
- a set of about 350 regular grammar rules to build phrase structure
- a set of about 350 rules to disambiguate lexical category

Being a deterministic parser, Fidditch pursues a single path in analyzing a sentence and provides a single analysis. Of course, the parser is necessarily far from complete; neither its grammar rules nor its lexicon incorporate all the information needed

to adequately describe English. Therefore, it is to be expected that the parser will encounter structures that it does not recognize and will make errors of analysis. When it is unable to provide a complete analysis of text, it is designed to return a partial description and proceed. Even with the inevitable errors, it has proven useful for analyzing text. (The parser has been used to analyze tens of millions of words of written text as well as transcripts of speech in order to, for example, search for subject-verb-object triples.)

Rules for the parser are essentially pattern-action rules which match a single incomplete node (from a stack) and a buffer of up to three completed constituents. The patterns of parser rules can refer only to limited aspects of the current parser state. Rules can mention the grammatical category of the constituents in the buffer and the current incomplete node. Rules can also refer to a limited set (about 200) of specific words that are grammatically distinguished (e.g. *be*, *of*, *as*). Complementation rules of course refer to a larger set of specific lexical items.

The model of the parser is that it recognizes grammatical patterns; whenever it sees a pattern of its rule base, it builds the associated structure; if it doesn't see a pattern, it does nothing. At every step in the parse, the most specific pattern is selected. The more linguistic information in the parser, the better able it will be to recognize and describe patterns. But when it does not recognize some construction, it simply uses a more general pattern to parse it. This feature (i.e., matching the most specific pattern available, but always having default analyses as more general patterns) is necessary both for analyzing unrestricted text and for training on the basis of unrestricted text.

Disambiguation rules

One of the possible rule actions of the parser is to select a lexical category for an ambiguous word. In Fidditch about half of the 700 pattern-action rules are disambiguation rules.

A simple disambiguation rule, both existing in the hand-written disambiguation rules and acquired by the training algorithm, looks like this:

(2) [PREP+TNS] = TNS [N+V]

Rule (2) says that a word that can be a preposition or a tense marker (i.e. the word *to*) followed by a word which can be a noun or a verb is a tense marker followed by a verb. This rule is obviously not always correct; there are two ways that it can be overridden. For rule (2), a previous rule may have already disambiguated the PREP+TNS, for example by recognizing the phrase *close to*. Alternatively, a more specific current rule may apply, for example recognizing the specific noun *date* in *to date*. In general, the parser provides a window of attention that moves through a sentence from the beginning to the end. A rule that, considered in isolation, would match some sequence of words in a sentence, may not in fact apply, either because a more specific rule matches, or because a different rule applied earlier.

These disambiguation rules are obviously closely related to the bigrams and trigrams of stochastic disambiguation methods. The rules differ in that 1) they can refer to the 200 specified lexical items, and 2) they can refer to the current incomplete node.

Disambiguation of lexical category must occur before the regular grammar rules can run; regular grammar rules only match nodes whose lexical category is disambiguated. [1]

The grammatical categories

Fidditch has 46 lexical categories (including 8 punctuations), mostly encoding rather standard parts of speech, with inflections folded into the category set. This is many fewer than the 87 simple word tags of the Brown Corpus or of related tagging systems (see Garside, Leech and Sampson 1987:165-183). Most of the proliferation of tags in such systems is the result of encoding information that is either lexically predictable or structurally predictable. For example, the Brown tagset provides distinct tags for subjective and objective uses of pronouns. For *I* and *me* this distinction is predictable both from the lexical items themselves and from the structure in which they occur. In Fidditch, both subjective and objective pronouns are tagged simply as PRO.

One of the motivations of the larger tagsets is to facilitate searching the corpus: using only the elaborated tags, it is possible to recover some lexical and structural distinctions. When Fidditch is used to search for constructions, the syntactic structure and lexical identity of items is available and thus there is no need to encode it in the tagset. To use the tagged Brown Corpus for training and

[1] More recent approaches to deterministic parsing may allow categorial disambiguation to occur *after* some of the syntactic properties of phrases are noted (Marcus, Hindle, and Fleck 1983). But in structure-building deterministic parsers such as Fidditch, lexical category must be disambiguated *before* any structure can be built.

evaluating disambiguation rules, the Brown categories were mapped onto the 46 lexical categories native to Fidditch.

Errors in the hand-written disambiguation rules

Using the tagged Brown Corpus, we can ask how well the disambiguation rules of Fidditch perform in terms of the tagged Brown Corpus. Comparing the part of speech assigned by Fidditch to the (transformed) Brown part of speech, we find about 6.5% are assigned an incorrect category. Approximately 30% of the word tokens in the Brown Corpus are categorially ambiguous in the Fidditch lexicon; it is this 30% that we are concerned with in acquiring disambiguation rules. For these ambiguous words, the error rate for the hand constructed disambiguation rules is about 19%. That is, about 1 out of 5 of the ambiguous word tokens are incorrectly disambiguated. This means that there is a good chance that any given sentence will have an error in part of speech. Obviously, there is considerable motivation for improving the lexical disambiguation. Indeed, errors in lexical category disambiguation are the biggest source of error for the parser.

It has been my experience that the disambiguation rule set is particularly difficult to improve by hand. The disambiguation rules make less syntactic sense than the regular grammar rules, and therefore the effect of adding or deleting a rule on the parser performance is hard to predict. In the long run it is likely that these disambiguation rules should be done away with, substituting disambiguation by side effect as proposed by Milne (1986). But in the meantime, we are faced with the need to improve this model of lexical disambiguation for a deterministic parser.

4 The Training Procedure

The model of deterministic parsing proposed by Marcus (1980) has several properties that aid in acquisition of symbolic rules for syntactic analysis, and provide a natural way to resolve the twin problems of discovering a) when it is necessary to acquire a new rule, and b) what new rule to acquire (see the discussion in Berwick 1985). The key features of this model of parsing relevant to acquisition are:

- because the parser is deterministic and has a limited window of attention, failure (and therefore the need for a new rule) can be localized.
- because the rules of the parser correspond closely to the instantaneous description of the state of the parser, it is easy to determine the form of the new rule.
- because there is a natural ordering of the rules acquired, there is never any ambiguity about which rule to apply. The ordering of new rules is fixed because more specific rules always have precedence.

These characteristics of the deterministic parser provide a way to acquire a new set of lexical disambiguation rules. The idea is as follows. Beginning with a small set of disambiguation rules, proceed to parse the tagged Brown Corpus. Check each disambiguation action against the tags to see if the correct choice was made. If an incorrect choice was made, use the current state of the parser together with the current set of disambiguation rules to create a new disambiguation rule to make the correct choice.

Once a rule has been acquired in this manner, it may turn out that it is not a correct rule. Although it worked for the triggering case, it may fail on other cases. If the rate of failure is sufficiently high, it is deactivated.

An additional phase of acquisition would be to generalize the rules to reduce the number of rules and widen their applicability. In the experiments reported here, no generalization has been done. This makes the rule set more redundant and less compact than necessary. However, the simplicity of the rule patterns of this expanded rule set allow a compact encoding and an efficient pattern matching.

The initial state for the training has the complete parser grammar – all the rules for building structures – but only a minimal set of context independent default disambiguation rules. Specifically, training begins with a set of rules which select a default category for ambiguous words words ignoring all context. For example, the rule (3) says that a word that can be an adjective or a noun or a verb (appearing in the first buffer position) is a noun, no matter what the second and third buffer positions show and no matter what the current incomplete node is.

(3) **A default disambiguation rule**
[ADJ+N+V] = N [*] [*]

In the absence of any other disambiguation rules (i.e. before any training), this rule would declare

fleet, which according to Fidditch's lexicon is an ADJ+N+V, to be a noun. There are 136 such default disambiguation rules, one for each lexically possible combination of lexical categories.

Acquisition of the disambiguation rules proceeds in the course of parsing sentences. In this way, the current state of the parser – the sentence as analyzed thus far – is available as a pattern for the training. At each step in parsing, before applying any parser rule, the program checks whether a new disambiguation rule may be acquired. If neither the first nor the second buffer position contains an ambiguous word, no disambiguation can occur, and no acquisition will occur. When an ambiguous word is encountered in the first or second buffer position, the current set of disambiguation rules may change.

New rule acquisition

The training algorithm has two basic components. The first component – **new rule acquisition** – first checks whether the currently selected disambiguation rule correctly disambiguates the ambiguous items in the buffer. If the wrong choice is made, then a new, more specific rule may be added to the rule set to make the correct disambiguation choice. (Since the new rule is more specific than the currently selected rule, it will have precedence over the older rule, and thus will make the correct disambiguation for the current case, overriding any previous disambiguation choice).

The pattern for the new rule is determined by the current parse state together with the current set of disambiguation rules. The new rule pattern must match the current state and also must be be more specific than any currently matching disambiguation rule. (If an existing rule matches the current state, it must be doing the wrong disambiguation, otherwise we would not be trying to acquire a new rule). If there is no available more specific pattern, no acquisition is possible, and the current rule set remains.

Although the patterns for rules are quite restricted, referring only to the data structures of the parser with a restricted set of categories, there are nevertheless on the order of 10^9 possible disambiguation rules.

The action for the new rule is simply to choose the correct part of speech.

Rule deactivation

The second component of the rule acquisition – **rule deactivation** – comes into play when the current disambiguation rule set makes the wrong disambiguation and yet no new rule can be acquired (because there is no available more specific rule). The incorrect rule may in this case be permanently deactivated. This deactivation occurs only when the proportion of incorrect applications reaches a given threshold (10 or 20% incorrect rule applications).

Ideally we might expect that each disambiguation rule would be completely correct; an incorrect application would count as evidence that the rule is wrong. However, this is an inappropriate idealization, for several reasons. Most crucially, the grammatical coverage as well as the range of linguistic processes modeled in Fidditch, are limited. (Note that this is a property of any current or foreseeable syntactic analyzer.) Since the grammar itself is not complete, the parser will have misanalyzed some constructions, leading to incorrect pattern matching. Moreover, some linguistic patterns that determine disambiguation (such as for example, the influence of parallelism) cannot be incorporated into the current rules at all, leading to occasional failure. As the overall syntactic model is improved, such cases will become less and less frequent, but they will never disappear altogether. Finally, there are of course errors in the tagged input. Thus, we can't demand perfection of the trained rules; rather, we require that rules reach a certain level of success. For rules that disambiguate the first element (except the default disambiguation rules), we require 80% success; for the other rules, 90% success. These cutoff figures were imposed arbitrarily; other values may be more appropriate.

An example of a rule that is acquired and then deactivated is the following.

(4) [ADJ+N+V] = ADJ [N] [*]

This rule correctly disambiguates some cases like *sound health* and *light barbell* but fails on a sufficient proportion (such cases as *sound energy* and *light intensity*) that it is permanently deactivated.

Interleaving of grammar and disambiguation

One of the advantages of embedding the training of disambiguation rules in a general parser is that independent parser actions can make the disambiguation more effective. For example, adverbs

often occur in an auxiliary phrase, as in the phrase *has immediately left*. The parser effectively ignores the adverb *immediately* so that from its point of view, *has* and *left* are contiguous. This in turn allows the disambiguation rules to see that *has* is the left context for *left* and to categorize *left* as a past participle (rather than a past tense or an adjective or a noun).

5 The Training

The training text was 450 of the 500 samples that make up the Brown Corpus, tagged with part of speech transformed into the 46 grammatical categories native to Fidditch. Ten percent of the corpus, selected from a variety of genres, was held back for testing the acquired set of disambiguation rules.

The training set (consisting of about a million words) was parsed, beginning with the default rule set and acquiring disambiguation rules as described above. After parsing the training set once, a certain set of disambiguation rules had been acquired. Then it was parsed over again, a total of five times. Each time, the rule set is further refined. It is effective to reparse the same corpus because the acquisition depends *both* on the sentence parsed *and* on the current set of rules. Therefore, the same sentence can induce different changes in the rule set depending on the current state of the rule set.

After the five iterations, 35000 rules have been acquired. For the training set, overall error rate is less than 2% and error rate for the ambiguous words is less than 5%. Clearly, the acquired rules effectively model the training set. Because the rule patterns are simple, they can be efficiently indexed and applied.

For the one tenth of the corpus held back (the test set), the performance of the trained set of rules is encouraging. Overall, the error rate for the test set is about 3%. For the ambiguous words the error rate is 10%. Compared to the performance of the existing hand-written rules, this shows almost a 50% reduction in the error rate. Additionally of course, there is a great saving in development time; to cut the error rate of the original hand-written rules in half by further hand effort would require an enormous amount of work. In contrast, this training algorithm is automatic (though it depends of course on the hand-written parser and set of grammar rules, and on the significant effort in tagging the Brown Corpus, which was used for training).

It is harder to compare performance directly to other reported disambiguation procedures, since the part of speech categories used are different. The 10% error rate on ambiguous words is the same as that reported by Garside, Leech and Sampson (1987:55). The program developed by Church (1987), which makes systematic use of relative tag probabilities, has, I believe, a somewhat smaller overall error rate.

Adding lexical relationships

The current parser models complementation relations only partially and it has no model at all of what word can modify what word (except at the level of lexical category). Clearly, a more comprehensive system would reflect the fact, for example, that *public apathy* is known to be a noun-noun compound, though the word *public* might be a noun or an adjective. One piece of evidence of the importance of such relationships is the fact that more than one fourth of the errors are confusions of adjective use with noun use as premodifier in a noun phrase. The current parser has no access to the kinds of information relevant to such modification and compound relationships, and thus does not do well on this distinction.

The claim of this paper is that the linguistic information embodied in the parser is useful to disambiguation, and that enhancing the linguistic information will result in improving the disambiguation. Adding that information about lexical relations to the parser, and making it available to the disambiguation procedure, should improve the accuracy of the disambiguation rules. In the long run the parser should incorporate general models of modification. However, we can crudely add some of this information to the disambiguation procedure, and take advantage of complementation information.

For each word in the training set, all word pairs including that word that might be lexically conditioned modification or complementation relationships are recorded. Any pair that occurs more than once and always has the same lexical category is taken to be a lexically significant collocation – either a complementation or a modification relationship. For example, for the word *study* the following lexical pairs are identified in the training set.

[ADJ] [NOUN]	recent study, present study, psychological study, graduate study, own study, such study, theoretical study
[N] [N]	use study, place-name study, growth study, time-&-motion study, birefringence study
[VPPRT] [N]	prolonged study, detailed study
[PREP][N]	under study
[V][N]	study dance
[V][PREP]	study at
[N][PREP]	study of, study on, study by

Obviously, only a small subset of the modification and complementation relations of English are included in this set. But missing pairs cause no trouble, since more general disambiguation rules will apply. This is an instance of the general strategy of the parser to use specific information when it is available and to fall back on more general (and less accurate) information in case no specific pattern matches. permitting an incremental improvement of the parser. The set of lexical pairs does include many high frequency collocations involving potentially ambiguous words, such as *close to* (ADJ PREP) and *long time* (ADJ N).

The test set was reparsed using this lexical information. The error rate for disambiguation using to these lexically related word pairs is quite small (3.5% of the ambiguous words), much better than the error rate of the disambiguation rules in general, resulting in an improved overall performance in disambiguation. Although this is only a crude model of complementation and modification relationships, it suggests how improvements in other modules of the parser will result in improvements in the disambiguation.

Using grammatical dependency

A second source of failure of the acquired disambiguation rules is that the acquisition algorithm is not paying enough attention to the information the parser provides.

The large difference in accuracy between the training set and the test set suggests that the acquired set of disambiguation rules are matching idiosyncratic properties of the training set rather than general extensible properties; the rules are too powerful. It seems that the rules that refer to all three items in the buffer are the culprit. For example, the acquired rule

(5) [N][PREP+TNS] = TNS [N+V] = V

applies to such cases as

(6) Shall we flip a *coin to see* which of us goes first?

In effect, this rule duplicates the action of another rule

(7) [PREP+TNS] = TNS [N+V] = V

In short, the rule set does not have appropriate shift invariance.

The problem with disambiguation rule (5) is that it refers to three items that are not in fact syntactically related: in sentence (6), there is no structural relation between the noun *coin* and the infinitive phrase *to see*. It would be appropriate to only acquire rules that refer to constituents that occur in construction with each other, since the predictability of part of speech from local context arises because of *structural relations* among words; there should be no predictability across words that are not structurally related.

We should therefore be able to improve the set of disambiguation rules by restricting new rules to only those involving elements that are in the same structure. We use the grammar as implemented in the parser to decide what elements are related and thus to restrict the set of rules acquired. Specifically, the following restriction on the acquisition of new rules is proposed.

> All the buffer elements referred to by a disambiguation rule must appear together in some other single rule.

This rules out examples like rule (5) because no single parser grammar rule ever refers to the noun, the *to* and the following verb at the same time. However, a rule like (7) is accepted because the parser grammar rule for infinitives does refer to *to* and the following verb at the same time.

For training, an additional escape for rules was added: if the first element of the buffer is ambiguous a rule may use the second element to disambiguate it whether or not there is any parser rule that refers to the two together. In these cases, if no new rule were added, the default disambiguation rules, which are notably ineffective, would match. (The default rules have a success rate of only 55% compared to over 94% for the disambiguation rules that depend on context.) Since the parser is not sufficiently complete to recognize all cases where words are related, this escape admits some local

context even in the absence of parser internal reasons to do so.

The training procedure was applied with this new constraint on rules, parsing the training set five times to acquire a new rule set. Restricting the rules to related elements had three notable effects. First, the number of disambiguation rules acquired was cut to nearly one third the number for the unrestricted rule set (about 12000 rules). Second, the difference between the training set and the test set is reduced; the error rate differs by only one percent. Finally, the performance of the restricted rule set is if anything slightly better than the unrestricted set (3427 errors for the restricted rules versus 3492 errors for the larger rule set). These results show the power of using the grammatical information encoded in the parser to direct the attention of the disambiguation rules.

6 Conclusion

I have described a training algorithm that uses an existing deterministic parser together with a corpus of tagged text to acquiring rules for disambiguating lexical category. Performance of the trained set of rules is much better than the previous hand-written rule set (error rate reduced by half). The success of the disambiguation procedure depends on the linguistic knowledge embodied in the parser in a number of ways.

- It uses the data structures and linguistic categories of the parser, focusing the rule acquisition mechanism on relevant elements.
- It is embedded in the parsing process so that parser actions can set things up for acquisition (for example, adverbs are in effect removed within elements of the auxiliary, restoring the contiguity of auxiliary elements).
- It uses the grammar rules to identify words that are grammatically related, and are therefore relevant to disambiguation.
- It can use rough models of complementation and modification to help identify words that are related.
- Finally, the parser always provides a default action. This permits the incremental improvement of the parser, since it can take advantage of more specific information when it is available, but it will always disambiguate somehow, no matter whether it has acquired the appropriate rules or not.

This work demonstrates the feasibility of acquiring the linguistic information needed to analyze unrestricted text from text itself. Further improvements in syntactic analyzers will depend on such automatic acquisition of grammatical and lexical facts.

References

Berwick, Robert C. 1985. The Acquisition of Syntactic Knowledge. MIT Press.

Church, Kenneth. 1987. A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings Second ACL Conference on Applied Natural Language Processing.*

DeRose, Stephen J. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics* 14.1.31-39.

Francis, W. Nelson and Henry Kucera. 1982. *Frequency Analysis of English Usage.* Houghton Mifflin Co.

Garside, Roger, Geoffrey Leech, and Geoffrey Sampson. 1987. *The Computational Analysis of English.* Longman.

Greene, Barbara B. and Gerald M. Rubin. 1971. Automated grammatical tagging of English. Department of Linguistics, Brown University.

Donald Hindle. 1983. Deterministic parsing of syntactic non-fluencies. *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics.*

Klein, S. and R.F. Simmons. 1963. A computational approach to grammatical coding of English words. *JACM* 10:334-47.

Marcus, Mitchell P. 1980. *A Theory of Syntactic Recognition for Natural Language.* MIT Press.

Marcus, Mitchell P., Donald Hindle and Margaret Fleck. 1983. D-theory: talking about talking about trees. *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics.*

Milne, Robert. 1986. Resolving Lexical Ambiguity in a Deterministic Parser. *Computational Linguistics* 12.1, 1-12.