Development and Evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals

Ezra Black *John Lafferty* *Salim Roukos*

<black|jlaff|roukos>@watson.ibm.com

IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598

ABSTRACT

We present an approach to grammar development where the task is decomposed into two separate subtasks. The first task is linguistic, with the goal of producing a set of rules that have a large coverage (in the sense that the correct parse is among the proposed parses) on a blind test set of sentences. The second task is statistical, with the goal of developing a model of the grammar which assigns maximum probability for the correct parse. We give parsing results on text from computer manuals.

1. Introduction

Many language understanding systems and machine translation systems rely on a parser of English as the first step in processing an input sentence. The general impression may be that parsers with broad coverage of English are readily available. In an effort to gauge the state of the art in parsing, the authors conducted an experiment in Summer 1990 in which 35 sentences, all of length 13 words or less, were selected randomly from a several-million-word corpus of Associated Press news wire. The sentences were parsed by four of the major large-coverage parsers for general English.[1] Each of the authors, working separately, scored 140 parses for correctness of constituent boundaries, constituent labels, and part-of-speech labels. All that was required of parses was accuracy in delimiting and identifying obvious constituents such as noun phrases, prepositional phrases, and clauses, along with at least rough correctness in assigning part-of-speech labels, e.g. a noun could not be labelled as a verb. The tallies of each evaluator were compared, and were identical or very close in all cases. The best-performing parser was correct for 60% of the sentences and the the remaining parsers were below 40%. More recently, in early 1992, the creator of another well-known system performed self-scoring on a similar task and reported 30% of input sentences as having been correctly parsed. On the basis of the preceeding evidence it seems that the current state of the art is far from being able to produce a robust parser of general English.

In order to break through this bottleneck and begin making steady and quantifiable progress toward the goal of developing a highly accurate parser for general English, organization of the grammar-development process along scientific lines and the introduction of stochastic modelling techniques are necessary, in our view. We have initiated a research program on these principles, which we describe in what follows. An account of our overall method of attacking the problem is presented in Section 2. The grammar involved is discussed in Section 3. Section 4 is concerned with the statistical modelling methods we employ. Finally, in Section 5, we present our experimental results to date.

2. Approach

Our approach to grammar development consists of the following 4 elements:

- Selection of application domain.
- Development of a manually-bracketed corpus (*treebank*) of the domain.
- Creation of a grammar with a large coverage of a blind test set of treebanked text.
- Statistical modeling with the goal that the correct parse be assigned maximum probability by the stochastic grammar.

We now discuss each of these elements in more detail.

Application domain: It would be a good first step toward our goal of covering general English to demonstrate that we can develop a parser that has a high parsing accuracy for sentences in, say, any book listed in Books In Print concerning needlework; or in any wholesale footwear catalog; or in any physics journal. The selected domain of focus should allow the acquisition of a naturally-occuring large corpus (at least a few million words) to allow for realistic evaluation of performance and

[1] At least one of the parties involved insisted that no performance results be made public. Such reticence is widespread and understandable. However, it is nonetheless important that performance norms be established for the field. Some progress has been made in this direction [3, 4].

Fa	Adverbial Phrase
Fc	Comparative Phrase
Fn	Nominal Clause
Fr	Relative Clause
G	Possessive Phrase
J	Adjectival Phrase
N	Noun Phrase
Nn	Nominal Proxy
Nr	Temporal Noun Phrase
Nv	Adverbial Noun Phrase
P	Prepositional Phrase
S	Full Sentence
Si	Sentential Interrupter
Tg	Present Participial Clause
Ti	Infinitival Clause
Tn	Past Participial Clause
V	Verb Phrase
NULL	Other

Table 1: Lancaster constituent labels

AT1	Singular Article (a, every)
CST	*that* as Conjunction
CSW	*whether* as Conjunction
JJ	General Adjective (free, subsequent)
NN1	Singular Common Noun (character, site)
PPH1	the Pronoun "it"
PPY	the Pronoun "you"
RR	General Adverb (exactly, manually)
VBDZ	"was"
VVC	Imperative form of Verb (attempt, proceed)
VVG	-ing form of Verb (containing, powering)

Table 2: Sample of Lancaster part-of-speech labels

adequate amounts of data to characterize the domain so that new test data does not surprise system developers with a new set of phenomena hitherto unaccounted for in the grammar.

We selected the domain of computer manuals. Besides the possible practical advantages to being able to assign valid parses to the sentences in computer manuals, reasons for focusing on this domain include the very broad but not unrestricted range of sentence types and the availability of large corpora of computer manuals. We amassed a corpus of 40 million words, consisting of several hundred computer manuals. Our approach in attacking the goal of developing a grammar for computer manuals is one of successive approximation. As a first approximation to the goal, we restrict ourselves to sentences of word length 7 - 17, drawn from a vocabulary consisting of the 3000 most frequent words (i.e. fully inflected forms, not lemmas) in a 600,000-word subsection of our corpus. Approximately 80% of the words in the 40-million-word corpus are included in the 3000-word vocabulary. We have available to us about 2 million words of sentences completely covered by the 3000-word vocabulary. A lexicon for this 3000-word vocabulary was completed in about 2 months.

Treebank: A sizeable sample of this corpus is hand-parsed ("treebanked"). By definition, the hand parse ("treebank parse") for any given sentence is considered its "correct parse" and is used to judge the grammar's parse. To fulfill this role, treebank parses are constructed as "skeleton parses," i.e. so that all obvious decisions are made as to part-of-speech labels, constituent boundaries and constituent labels, but no decisions are made which are problematic, controversial, or of which the treebankers are unsure. Hence the term "skeleton parse": clearly not all constituents will always figure in a treebank parse, but the essential ones always will. In practice, these are quite detailed parses in most cases. The 18 constituent labels[2] used in the Lancaster treebank are listed and defined in Table 1. A sampling of the approximately 200 part-of-speech tags used is provided in Table 2.

To date, roughly 420,000 words (about 35,000 sentences) of the computer manuals material have been treebanked by a team at the University of Lancaster, England, under Professors Geoffrey Leech and Roger Garside. Figure 1 shows two sample parses selected at random from the Lancaster Treebank.

The treebank is divided into a training subcorpus and a test subcorpus. The grammar developer is able to inspect the training dataset at will, but can never see the test dataset. This latter restriction is, we feel, crucial for making progress in grammar development. The purpose of a grammar is to correctly analyze previously unseen sentences. It is only by setting it to this task that its true accuracy can be ascertained. The value of a large bracketed training corpus is that it allows the grammarian to obtain quickly a very large[3] set of sentences that

[2] Actually there are 18 x 3 = 54 labels, as each label L has variants L& for a first conjunct, and L+ for second and later conjuncts, of type L: e.g. [N[N& the cause N&] and [N+ the appropriate action N+]N].

[3] We discovered that the grammar's *coverage* (to be defined later) of the training set increased quickly to above 98% as soon as the grammarian identified the problem sentences. So we have been

```
[N It_PPH1 N]
[V indicates_VVZ
   [Fn [Fn&whether_CSW
             [N a_AT1 call_NN1 N]
             [V completed_VVD successfully_RR V]Fn&]
        or_CC
        [Fn+ if_CSW
              [N some_DD error_NN1 N]@
              [V was_VBDZ detected_VVN V]
              @[Fr that_CST
                    [V caused_VVD
                       [N the_AT call_NN1 N]
                       [Ti to_TO fail_VVI Ti]V]Fr]Fn+]
                                          Fn]V]._.
[Fa If_CS
    [N you_PPY N]
    [V were_VBDR using_VVG
        [N a_AT1 shared_JJ folder_NN1 N]V]Fa]
[,_,
[V include_VVC
   [N the_AT following_JJ N]V]:_:
```

Figure 1: Two sample bracketed sentences from Lancaster Treebank.

the grammar fails to parse. We currently have about 25,000 sentences for training.

The point of the treebank parses is to constitute a "strong filter," that is to eliminate incorrect parses, on the set of parses proposed by a grammar for a given sentence. A candidate parse is considered to be "acceptable" or "correct" if it is *consistent* with the treebank parse. We define two notions of consistency: *structure-consistent* and *label-consistent.* The *span* of a consitituent is the string of words which it dominates, denoted by a pair of indices (i, j) where i is the index of the leftmost word and j is the index of the rightmost word. We say that a constituent **A** with span (i, j) in a candidate parse for a sentence is *structure-consistent* with the treebank parse for the same sentence in case there is no constituent in the treebank parse having span (i', j') satisfying

$$i' < i \leq j' < j$$

or

$$i < i' \leq j < j'.$$

In other words, there can be no "crossings" of the span of **A** with the span of any treebank non-terminal. A grammar parse is structure-consistent with the treebank parse if all of its constituents are structure-consistent with the treebank parse.

The notion of *label-consistent* requires in addition to structure-consistency that the grammar constituent name is equivalent[4] to the treebank non-terminal label.

The following example will serve to illustrate our consistency criteria. We compare a "treebank parse":

> [NT1 [NT2 w1_p1 w2_p2 NT2] [NT3 w3_p3 w4_p4 w5_p5 NT3]NT1]

with a set of "candidate parses":

> [NT1 [NT2 w1_p1 w2_p2 NT2] [NT3 w3_p3 [NT4 w4_p4 w5_p5 NT4]NT3]NT1]
>
> [NT1 [NT2 w1_p6 w2_p2 NT2] [NT5 w3_p9 w4_p4 w5_p5 NT5]NT1]
>
> [NT1 w1_p1 [NT6 b_p2 w3_p15 NT6][NT7 w4_p4 w5_p5 NT7]NT1]

For the structure-consistent criterion, the first and second candidate parses are correct, even though the first one has a more detailed constituent spanning (4, 5). The third is incorrect since the constituent NT6 is a case of a crossing bracket. For the label-consistent criterion, the first candidate parse is the only correct parse, because it has all of the bracket labels and parts-of-speech of the treebank parse. The second candidate parse is incorrect, since two of its part-of-speech labels and one of its bracket labels differ from those of the treebank parse.

Grammar writing and statistical estimation: The task of developing the requisite system is factored into two parts: a linguistic task and a statistical task.

- The linguistic task is to achieve perfect or near-perfect *coverage* of the test set. By this we mean that among the n parses provided by the parser for each sentence of the test dataset, there must be at least one which is *consistent* with the treebank filter. [5] To eliminate trivial solutions to this task, the grammarian must hold constant over the course of development the geometric mean of the number of parses per word, or equivalently the total number of parses for the entire test corpus.

- The statistical task is to supply a stochastic model for probabilistically training the grammar such that the parse selected as the most likely one is a correct parse.[6]

continuously increasing the training set as more data is treebanked.

[4] See Section 4 for the definition of a many-to-many mapping between grammar and treebank non-terminals for determining equivalence of non-terminals.

[5] We propose this sense of the term *coverage* as a replacement for the sense in current use, viz. simply supplying one or more parses, correct or not, for some portion of a given set of sentences.

[6] Clearly the grammarian can contribute to this task by, among other things, not just holding the average number of parses con-

The above decomposition into two tasks should lead to better broad-coverage grammars. In the first task, the grammarian can increase coverage since he can examine examples of specific uncovered sentences. In the second task, that of selecting a parse from the many parses proposed by a grammar, can best be done by maximum likelihood estimation constrained by a large treebank. The use of a large treebank allows the development of sophisticated statistical models that should outperform the traditional approach of using human intuition to develop parse preference strategies. We describe in this paper a model based on probabilistic context-free grammars estimated with a constrained version of the Inside-Outside algorithm (see Section 4)that can be used for picking a parse for a sentence. In [2], we desrcibe a more sophisticated stochastic grammar that achieves even higher parsing accuracy.

3. Grammar

Our grammar is a feature-based context-free phrase structure grammar employing traditional syntactic categories. Each of its roughly 700 "rules" is actually a rule template, compressing a family of related productions via unification.[7] Boolean conditions on values of variables occurring within these rule templates serve to limit their ambit where necessary. To illustrate, the rule template below[8]

$$\begin{bmatrix} f1 : a \\ f2 : V1 \\ f3 : V2 \end{bmatrix} \rightarrow \begin{bmatrix} f1 : b \\ f2 : V1 \\ f3 : V3 \end{bmatrix} \begin{bmatrix} f1 : c \\ f2 : V1 \\ f3 : V2 \end{bmatrix}$$

where

$$(V2 = d \mid g \mid h) \ \& \ (V3 \neq k)$$

imposes agreement of the children with reference to feature f2, and percolates this value to the parent. Acceptable values for feature f3 are restricted to three (d,g,h) for the second child (and the parent), and include all possible values for feature f3 *except* k, for the first child. Note that the variable value is also allowed in all cases mentioned (V1,V2,V3). If the set of licit values for feature f3 is {d,e,f,g,h,i,j,k,l}, and that for feature f2 is {r,s}, then, allowing for the possibility of variables remaining as such, the rule template above represents 3*4*9 = 108 different rules. If the condition were removed, the rule template would stand for 3*10*10 = 300 different rules.

While a non-terminal in the above grammar is a feature vector, we group multiple non-terminals into one class which we call a *mnemonic*, and which is represented by the least-specified non-terminal of the class. A sample mnemonic is N2PLACE (Noun Phrase of semantic category Place). This mnemonic comprises all non-terminals that unify with:

$$\begin{bmatrix} pos : n \\ barnum : two \\ details : place \end{bmatrix}$$

including, for instance, Noun Phrases of Place with no determiner, Noun Phrases of Place with various sorts of determiner, and coordinate Noun Phrases of Place. Mnemonics are the "working nonterminals" of the grammar; our parse trees are labelled in terms of them. A production specified in terms of mnemonics (a *mnemonic production*) is actually a family of productions, in just the same way that a mnemonic is a family of non-terminals. Mnemonics and mnemonic productions play key roles in the stochastic modelling of the grammar (see below). A recent version of the grammar has some 13,000 mnemonics, of which about 4000 participated in full parses on a run of this grammar on 3800 sentences of average word length 12. On this run, 440 of the 700 rule templates contributed to full parses, with the result that the 4000 mnemonics utilized combined to form approximately 60,000 different mnemonic productions. The grammar has 21 features whose range of values is 2 - 99, with a median of 8 and an average of 18. Three of these features are listed below, with the function of each:

det_pos	Determiner Subtype
degree	Degreė of Comparison
noun_pronoun	Nominal Subtype

Table 3: Sample Grammatical Features

To handle the huge number of linguistic distinctions required for real-world text input, the grammarian uses many of the combinations of the feature set. A sample rule (in simplified form) illustrates this:

$$\begin{bmatrix} pos : j \\ barnum : one \\ details : V1 \\ degree : V3 \end{bmatrix} \rightarrow \begin{bmatrix} pos : j \\ barnum : zero \\ details : V1 \\ degree : V3 \end{bmatrix}$$

This rule says that a lexical adjective parses up to an adjective phrase. The logically primary use of the feature "details" is to more fully specify conjunctions and phrases

stant, but in fact steadily reducing it. The importance of this contribution will ultimately depend on the power of the statistical models developed after a reasonable amount of effort.

[7] *Unification* is to be understood in this paper in a very limited sense, which is precisely stated in Section 4. Our grammar is not a unification grammar in the sense which is most often used in the literature.

[8] where f1,f2,f3 are features; a,b,c are feature values; and V1,V2,V3 are variables over feature values

involving them. Typical values, for coordinating conjunctions, are "or" and "but"; for subordinating conjunctions and associated adverb phrases, they include e.g. "that" and "so." But for content words and phrases (more precisely, for nominal, adjectival and adverbial words and phrases), the feature, being otherwise otiose, carries the semantic category of the head.

The mnemonic names incorporate "semantic" categories of phrasal heads, in addition to various sorts of syntactic information (e.g. syntactic data concerning the embedded clause, in the case of "that-clauses"). The "semantics" is a subclassification of content words that is designed specifically for the manuals domain. To provide examples of these categories, and also to show a case in which the semantics succeeded in correctly biasing the probabilities of the trained grammar, we contrast (simplified) parses by an identical grammar, trained on the same data (see below), with the one difference that semantics was eliminated from the mnemonics of the grammar that produced the first parse below.

> [SC[V1 Enter [N2[N2 the name [P1 of the system P1]N2][SD you [V1 want [V2 to [V1 connect [P1 to P1]V1]V2]V1]SD]N2]V1]SC].

> [SCSEND-ABS-UNIT[V1SEND-ABS-UNIT Enter [N2ABS-UNIT the name [P1SYSTEMOF of [N2SYSTEM the system [SDORGANIZE-PERSON you [V1ORGANIZE want [V2ORGANIZE to connect [P1TO to P1]V2]V1]SD]N2]P1]N2]V1]SC].

What is interesting here is that the structural parse is different in the two cases. The first case, which does not match the treebank parse[9] parses the sentence in the same way as one would understand the sentence, "Enter the chapter of the manual you want to begin with." In the second case, the semantics were able to bias the statistical model in favor of the correct parse, i.e. one which does match the treebank parse. As an experiment, the sentence was submitted to the second grammar with a variety of different verbs in place of the original verb "connect", to make sure that it is actually the semanitc class of the verb in question, and not some other factor, that accounts for the improvement. Whenever verbs were substituted that were licit syntatically but not semantically (e.g. adjust, comment, lead) the parse was as in the first case above. Of course other verbs of the class "ORGANIZE" were associated with the correct parse, and verbs that did were not even permitted syntactically occasioned the incorrect parse.

We employ a lexical preprocessor to mark multiword units as well as to license unusual part-of-speech assignments, or even force labellings, given a particular context. For example, in the context: "How to:", the word "How" can be labelled once and for all as a General Wh-Adverb, rather than a Wh-Adverb of Degree (as in, "How tall he is getting!"). Three sample entries from our lexicon follow: "Full-screen" is labelled as an adjective which usually bears an attributive function, with the semantic class "Screen-Part". "Hidden" is categorized as a past participle of semantic class "Alter". "1983" can be a temporal noun (viz. a year) or else a number. Note that all of these classifications were made on the basis of the examination of concordances over a several-hundred-thousand-word sample of manuals data. Possible uses not encountered were in general not included in our lexicon.

full-screen	JSCREEN-PTB*
Hidden	VALTERN*
1983	NRSG* M-C-*

Table 4: Sample lexical entries

Our approach to grammar development, syntactical as well as lexical, is frequency-based. In the case of syntax, this means that, at any given time, we devote our attention to the most frequently-occurring construction which we fail to handle, and not the most "theoretically interesting" such construction.

4. Statistical Training and Evaluation

In this section we will give a brief description of the procedures that we have adopted for parsing and training a probabilistic model for our grammar. In parsing with the above grammar, it is necessary to have an efficient way of determining if, for example, a particular feature bundle $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N)$ can be the parent of a given production, some of whose features are expressed as variables. As mentioned previously, we use the term *unification* to denote this matching procedure, and it is defined precisely in figure 2.

In practice, the unification operations are carried out very efficiently by representing bundles of features as bit-strings, and realizing unification in terms of logical bit operations in the programming language PL.8 which is similar to C. We have developed our own tools to translate the rule templates and conditions into PL.8 programs.

A second operation that is required is to partition the set of nonterminals, which is potentially extremely large, into a set of equivalence classes, or *mnemonics*, as mentioned earlier. In fact, it is useful to have a tree, which hierarchically organizes the space of possible fea-

[9] [V Enter [N the name [P of [N the system [Fr[N you][V want [Ti to connect [P to]]]]]]].

```
UNIFY($\mathbf{A}, \mathbf{B}$):
    do for each feature $f$
        if not FEATURE_UNIFY($\mathbf{A}_f, \mathbf{B}_f$)
            then return FALSE
    return TRUE

FEATURE_UNIFY($a, b$):
    if $a = b$ then return TRUE
    else if $a$ is variable or $b$ is variable
            then return TRUE
    return FALSE
```

Figure 2

ture bundles into increasingly detailed levels of semantic and syntactic information. Each node of the tree is itself represented by a feature bundle, with the root being the feature bundle all of whose features are variable, and with a decreasing number of variable features occuring as a branch is traced from root to leaf. To find the mnemonic $\mathcal{M}(\mathbf{A})$ assigned to an arbitrary feature bundle **A**, we find the node in the mnemonic tree which corresponds to the smallest mnemonic that contains (subsumes) the feature bundle **A** as indicated in Fugure 3.

```
$\mathcal{M}(\mathbf{A})$:
    $n = root\_of\_mnemonic\_tree$
    return SEARCH_SUBTREE($n$, $\mathbf{A}$)

SEARCH_SUBTREE($n$, $\mathbf{A}$)
    do for each child $m$ of $n$
        if $Mnemonic(m)$ contains $\mathbf{A}$
            then return SEARCH_SUBTREE($m$, $\mathbf{A}$)
    return $Mnemonic(n)$
```

Figure 3

Unconstrained training: Since our grammar has an extremely large number of non-terminals, we first describe how we adapt the well-known Inside-Outside algorithm to estimate the parameters of a stochastic context-free grammar that approximates the above context-free grammar. We begin by describing the case, which we call unconstrained training, of maximizing the likelihood of an unbracketed corpus. We will later describe the modifications necessary to train with the constraint of a bracketed corpus.

To describe the training procedure we have used, we will assume familiarity with both the CKY algorithm [?] and the Inside-Outside algorithm [?], which we have adapted to the problem of training our grammar. The main computations of the Inside-Outside algorithm are indexed using the CKY procedure which is a bottom-up chart parsing algorithm. To summarize the main points in our adaptation of these algorithms, let us assume that the grammar is in Chomsky normal form. The general case involves only straight-forward modifications. Proceeding in a bottom-up fashion, then, we suppose that we have two nonterminals (bundles of features) **B** and **C**, and we find all nonterminals **A** for which $\mathbf{A} \rightarrow \mathbf{B}\ \mathbf{C}$ is a production in the grammar. This is accomplished by using the unfication operation and checking that the relevent Boolean conditions are satisfied for the nonterminals **A**, **B**, and **C**.

Having found such a nonterminal, the usual Inside-Outside algorithm requires a recursive update of the Inside probabilities $I_{\mathbf{A}}(i,j)$ and outside probabilities $O_{\mathbf{A}}(i,j)$ that **A** spans (i,j). These updates involve the probability parameter

$$Pr_{\mathbf{A}}(\mathbf{A} \rightarrow \mathbf{B}\ \mathbf{C}).$$

In the case of our feature-based grammar, however, the number of such parameters would be extremely large (the grammar can have on the order of few billion nonterminals). We thus organize productions into the equivalence classes induced by the mnemomic classes on the non-terminals. The update then uses mnemonic productions for the stochastic grammar using the parameter

$$Pr_{\mathcal{M}(\mathbf{A})}(\mathcal{M}(\mathbf{B}) \rightarrow \mathcal{M}(\mathbf{C})\ \mathcal{M}(\mathbf{C})).$$

Of course, for lexical productions $\mathbf{A} \rightarrow w$ we use the corresponding probability

$$Pr_{\mathcal{M}(\mathbf{A})}(\mathcal{M}(\mathbf{A}) \rightarrow w)$$

in the event that we are rewriting not a pair of nonterminals, but a word w.

Thus, probabilities are expressed in terms of the set of mnemonics (that is, by the nodes in the mnemonic tree), rather that in terms of the actual nonterminals of the grammar. It is in this manner that we can obtain efficient and reliable estimates of our parameters. Since the grammar is very detailed, the mnemonic map $\mathcal{M}$ can be increasingly refined so that a greater number of linguistic phenomena are caputured in the probabilities. In principle, this could be carried out automatically to determine the optimum level of detail to be incorporated into the model, and different parameterizations could be smoothed together. To date, however, we have only contructed mnemonic maps by hand, and have thus experimented with only a small number of parameterizations.

Constrained training: The Inside-Outside algorithm is a special case of the general EM algorithm, and as such, sucessive iteration is guaranteed to converge to a set of parameters which locally maximize the likelihood of generating the training corpus. We have found it useful to employ the treebank to supervise the training of these parameters. Intuitively, the idea is to modify the algorithm to locally maximize the likelihood of generating the training corpus using parses which are "similar" to the treebank parses. This is accomplished by only collecting statistics over those parses which are *consistent* with the treebank parses, in a manner which we will now describe. The notion of *label-consistent* is defined by a (many-to-many) mapping from the mnemonics of the feature-based grammar to the nonterminal labels of the treebank grammar. For example, our grammar maintains a fairly large number of semantic classes of singular nouns, and it is natural to stipulate that each of them is label-consistent with the nonterminal `NN1` denoting a generic singular noun in the treebank. Of course, to exhaustively specify such a mapping would be rather time consuming. In practice, the mapping is implemented by organizing the nonterminals hierarchically into a tree, and searching for consistency in a recursive fashion.

The simple modification of the CKY algorithm which takes into account the treebank parse is, then, the following. Given a pair of nonterminals **B** and **C** in the CKY chart, if the span of the parent is not structure-consistent then this occurence of **B C** cannot be used in the parse and we continue to the next pair. If, on the other hand, it is structure-consistent then we find all candidate parents **A** for which **A** → **B C** is a production of the grammar, but include only those that are label-consistent with the treebank nonterminal (if any) in that position. The probabilities are updated in exactly the same manner as for the standard Inside-Outside algorithm. The procedure that we have described is called *constrained training*, and it significantly improves the effectiveness of the parser, providing a dramatic reduction in computational requirements for parameter estimation as well as a modest improvement in parsing accuracy.

Sample mappings from the terminals and nonterminals of our grammar to those of the Lancaster treebank are provided in Table 5. For ease of understanding, we use the version of our grammar in which the semantics are eliminated from the mnemonics (see above). Category names from our grammar are shown first, and the Lancaster categories to which they map are shown second:

P1	P
FRV2	Fr
SD	Fr
IANYTI	Ti
JBVVN*	JJ

Table 5: Sample of grammatical category mappings

The first case above is straightforward: our prepositional-phrase category maps to Lancaster's. In the second case, we break down the category Relative Clause more finely than Lancaster does, by specifying the syntax of the embedded clause (e.g. FRV2: "that opened the adapter"). The third case relates to relative clauses lacking prefatory particles, such as: "the row *you are specifying*"; we would call "you are specifying" an SD (Declarative Sentence), while Lancaster calls it an Fr (Relative Clause). Our practice of distinguishing constituents which function as interrupters from the same constituents *tout court* accounts for the fourth case; the category in question is Infinitival Clause. Finally, we generate attributive adjectives (JB) directly from past participles (VVN) by rule, whereas Lancaster opts to label as adjectives (JJ) those past participles so functioning.

5. Experimental Results

We report results below for two test sets. One (Test Set A) is drawn from the 600,000-word subsection of our corpus of computer manuals text which we referred to above. The other (Test Set B) is drawn from our full 40-million-word computer manuals corpus. Due to a more or less constant error rate of 2.5% in the treebank parses themselves, there is a corresponding built-in margin of error in our scores. For each of the two test sets, results are presented first for the linguistic task: making sure that a correct parse is present in the set of parses the grammar proposes for each sentence of the test set. Second, results are presented for the statistical task, which is to ensure that the parse which is selected as most likely, for each sentence of the test set, is a correct parse.

Number of Sentences	935
Average Sentence Length	12
Range of Sentence Lengths	7-17
Correct Parse Present	96%
Correct Parse Most Likely	73%

Table 6: Results for Test Set A

Number of Sentences	1105
Average Sentence Length	12
Range of Sentence Lengths	7-17
Correct Parse Present	95%
Correct Parse Most Likely	75%

Table 7: Results for Test Set B

Recall (see above) that the geometric mean of the number of parses per word, or equivalently the total number of parses for the entire test set, must be held constant over the course of the grammar's development, to eliminate trivial solutions to the coverage task. In the roughly year-long period since we began work on the computer manuals task, this average has been held steady at roughly 1.35 parses per word. What this works out to is a range of from 8 parses for a 7-word sentence, through 34 parses for a 12-word sentence, to 144 parses for a 17-word sentence. In addition, during this development period, performance on the task of picking the most likely parse went from 58% to 73% on Test Set A. Periodic results on Test Set A for the task of providing at least one correct parse for each sentence are displayed in Table 8.

We present additional experimental results to show that our grammar is completely separable from its accompanying "semantics". Note that semantic categories are not "written into" the grammar; i.e., with a few minor exceptions, no rules refer to them. They simply percolate up from the lexical items to the non-terminal level, and contribute information to the mnemonic productions which constitute the parameters of the statistical training model.

An example was given in Section 3 of a case in which the version of our grammar that includes semantics outperformed the version of the same grammar without semantics. The effect of the semantic information in that particular case was apprently to bias the trained grammar towards choosing a correct parse as most likely. However, we did not quantify this effect when we presented the example. This is the purpose of the experimental results shown in Table 9. Test B was used to test our current grammar, first with and then without semantic categories in the mnemonics.

It follows from the fact that the semantics are not written into the grammar that the coverage figure is the same with and without semantics. Perhaps surprising, however, is the slight degree of improvement due to the semantics on the task of picking the most likely parse: only 2 percentage points. The more detailed parametrization with semantic categories, which has about 13,000 mnemonics achieved only a modest improvement in parsing accuracy over the parametrization without semantics, which has about 4,600 mnemonics.

January 1991	91%
April 1991	92%
August 1991	94%
December 1991	96%
April 1992	96%

Table 8: Periodic Results for Test Set A: Sentences With At Least 1 Correct Parse

Number of Sentences	1105
Average Sentence Length	12
Range of Sentence Lengths	7-17
Correct Parse Present (In Both Cases)	95%
Correct Parse Most Likely (With Semantics)	75%
Correct Parse Most Likely (No Semantics)	73%

Table 9: Test Subcorpus B With and Without Semantics

6. Future Research

Our future research divides naturally into two efforts. Our linguistic research will be directed toward first parsing sentences of any length with the 3000-word vocabulary, and then expanding the 3000-word vocabulary to an unlimited vocabulary. Our statistical research will focus on efforts to improve our probabilistic models along the lines of the new approach presented in [2].

References

1. Baker, J., *Trainable grammars for speech recognition.* In Speech Communication papers presented at the 97-th Meeting of the Acostical Society of America, MIT, Canbridge, MA, June 1979.
2. Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., and Roukos, S., *Towards History-based Grammars: Using Richer Models for Probabilistic Parsing.* Proceedings of Fifth DARPA Speech and Natural Language Workshop, Harriman, NY, February 1992.
3. Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T.. *A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars.* Proceedings of Fourth DARPA Speech and Natural Language Workshop, pp. 306-311, 1991.
4. Harrison, P., Abney, S., Black, E., Flickenger, D., Gdaniec, C., Grishman, R., Hindle, D., Ingria, R., Marcus, M., Santorini, B., and Strzalkowski, T.. *Evaluating Syntax Performance of Parser/Grammars of English.* Proceedings of Natural Language Processing Systems Evaluation Workshop, Berkeley, California, 1991.
5. Hopcraft, J. E. and Ullman, Jeffrey D. *Introduction to Automata Theory, Languages, and Computation,* Reading, MA: Addison-Wesley, 1979.
6. Jelinek, F., Lafferty, J. D., and Mercer, R. L. *Basic Methods of Probabilistic Context-Free Grammars.* Computational Linguistics, to appear.