A System for Computer Assisted Grammar Construction

H-H. Shih and S.J. Young

CUED/F-INFENG/TR.170
June 14, 1994

Abstract

A system of computer assisted grammar construction (CAGC) is presented in this paper. The CAGC system is designed to generate broad-coverage grammars for large natural language corpora by utilizing both an extended inside-outside algorithm [Pereira and Schabes, 1992] and an automatic phrase bracketing (AUTO) system which is designed to provide the extended algorithm with constituent information during learning. This paper demonstrates the capability of the CAGC system to deal with realistic natural language problems and the usefulness of the AUTO system in the inside-outside based grammar re-estimation. Performance results including the proportion of sentences parsed and precision are presented for a grammar constructed for the Wall Street Journal (WSJ) corpus.

Contents

1 Introduction

In natural language processing (NLP), a grammar is often required for a new corpus. The conventional way of generating a new grammar is to ask linguists to design a set of production rules by hand parsing the text corpus. However, to develop a useful grammar which covers the whole corpus is almost impossible, unless the size of the corpus is small. Additionally, a grammar is usually developed under the assumption that the sentences in the corpus are well-formed, but this is not true for most naturally-occurring corpora. Because of these factors, only limited coverage can be achieved by a hand-written grammar. Furthermore, as more sentences are added to the corpus, re-tuning the grammar becomes necessary to maintain coverage. This is usually done by asking linguists to re-write some of the production rules and probably add new ones. Development and later modification of a grammar by these means are labour intensive, time consuming and often introduce unwanted rule interactions.

To overcome these problems, automatic algorithms are needed to infer a grammar for a given corpus. Techniques for grammar inference (GI) [Fu and Booth, 1986a] [Fu and Booth, 1986b] [Lucas, 1993] can be used to replace or augment the conventional manual grammar construction. Given a set of data (training set), the aim of GI is to estimate or infer a grammar G such that the language L(G) generated by G not only includes all sentences in the training set but more importantly generalises unseen members of L(G). Several algorithms have been explored for this purpose, for example, genetic algorithms [Wyard, 1993], the error correcting grammar inference (ECGI) algorithm [Rulot et al., 1989] and the inside-outside algorithm [Baker, 1979].

The inside-outside algorithm was first proposed by Baker [Baker, 1979]. The algorithm is a version of the Baum-Welch re-estimation algorithm [Baum, 1972] designed to operate on stochastic context-free grammars (SCFGs) in Chomsky Normal Form (CNF) [Chomsky, 1957]. Investigations into the properties and applications of this algorithm have been published by a number of researchers [Jelinek, 1985] [Dodd, 1989] [Lari and Young, 1990] [Carroll and Charniak, 1992]. The algorithm was further extended by [Schabes, 1992] [Kupiec, 1992] and [Waegner, 1993] in order to handle non-CNF CFGs, which permits a more linguistically plausible representation of natural language.

In the inside-outside algorithm, there are two approaches for GI, namely explicit and implicit. The explicit approach is to manually produce a set of production rules for a language corpus and then use the inside-outside algorithm to estimate the required rule probabilities. The implicit approach starts with a grammar containing all possible rules, given a set of non-terminals and (pre)terminals. The grammar is then trained on a large training corpus, in the hope that the learning process will converge to a linguistically-motivated grammar which fits the training corpus.

However, two inherent problems of the inside-outside algorithm inhibit its practical application in NLP. Firstly, the algorithm has a high complexity of O(n^3m^3), where n is the length of sentences and m is the number of non-terminals in the grammar. With a complexity of O(n^3m^3), only inference of simple toy grammars is practicable if all the possible rules are allowed in the learning process as in the implicit approach. Secondly, the algorithm does not guarantee convergence on a linguistically-motivated maximum. As the number of non-terminals increases, the convergence property of the algorithm degrades due to the insufficiency of data. It becomes clear that the raw training text corpus from which the grammar is trained provides insufficient linguistic information.

To overcome these two key problems, a hybrid method integrating the explicit and implicit approaches is necessary. The hybrid method integrates a core grammar and a set of constrained implicit rules to reduce the size of the rule set as well as to give better bootstrapping in the learning process. Further improvement can be obtained by utilizing phrase bracketing information during training as proposed in [Pereira and Schabes, 1992] [Black et al., 1992]. In this approach, the inside-outside algorithm considers only those rules whose spans are compatible with the *a priori* bracketings derived for the training set. This significantly speeds up the learning process and also provides a tighter link between the hierarchical structure of the inferred grammar and that of the training set, promoting to convergence onto a linguistically-motivated set of rules.

This paper describes a complete system for computer assisted grammar construction (CAGC). It includes a new algorithm for automatic phrase bracketing and an evaluation of its usefulness within the context of grammar inference using the inside-outside algorithm. It also demonstrates that inside-outside based inference can be extended to deal with realistic problems using only tagged text data for training. It concludes with an experimental evaluation using a subset of the Wall Street Journal (WSJ) text corpus.

2 Overview of the CAGC system

The CAGC system is designed to infer linguistically-motivated SCFGs with broad-coverage for large corpora. The system takes advantages of both heuristic and stochastic approaches. Heuristic knowledge provides powerful and important constraints to the system, whereas stochastic information deals with situations which are too complex or too trivial for heuristic rules to handle. A block diagram of the system is shown in Figure 1. The first part of the system falls into two stages: construction of an initial SCFG and phrase-bracketing of the raw text data. In the second part of the system, a grammar is inferred by utilizing the inside-outside algorithm to re-estimate the initial SCFG from the bracketed text data.

The initial SCFG is derived from the core grammar (explicit part) and a set of CF rules (implicit part). The explicit part was manually developed using a grammar development environment tool (GDE) [Briscoe et al., 1987] to form a skeleton of the SCFG. The implicit part consists of all possible rules which do not appear in the core grammar but which are nevertheless linguistically plausible. This is done by filtering all possible rule forms using constraints which enforce structural features such as headedness [Briscoe and Waegner, 1993] [Jackendoff, 1977]. The explicit and implicit rules are then integrated into a hybrid SCFG along with an appropriate set of initial probabilities. In order to bias the learning process towards a linguistically-motivated optimum, the explicit rules are given higher initial probabilities whereas the implicit rules are assigned low probabilities to start with. Details of the grammar development and the calculation of the initial probabilities are described in Section 3.

In order to convert the raw text data into a constituent-rich training set, the AUTO system is designed to generate phrase bracketing information. The AUTO system utilizes heuristic knowledge to bracket the raw text data in a fashion of integrating top-down and bottom-up approaches. The training set with derived constituent information provides the additional constraints to the grammar re-estimation process in the second part of the CAGC system. The detailed bracketing algorithm and its justification are given later in Section 4.

In the second part of the CAGC system, the inside-outside inference procedure, incorporating a bottom-up chart parser [Gazdar and Mellish, 1989], iteratively re-estimates the probabilities of the production rules. The updated probabilities are calculated according to the weighted frequency counts of the rules used in parses licensed by the grammar and generated at the previous iteration. At the end of each iteration, the rules with probabilities falling below a pre-defined threshold are discarded. This reduces the size of the inferred grammar and the computational expense of estimation. The re-estimation process continues until either the change in the total log probability (sum of log probabilities of all possible parses generated for all the training data) between iterations is less than a minimum or the number of iterations reaches a maximum (set to 10). Both limiting values and the rule elimination threshold were determined empirically in the system development. The final inferred grammar is generated when either criteria is met.

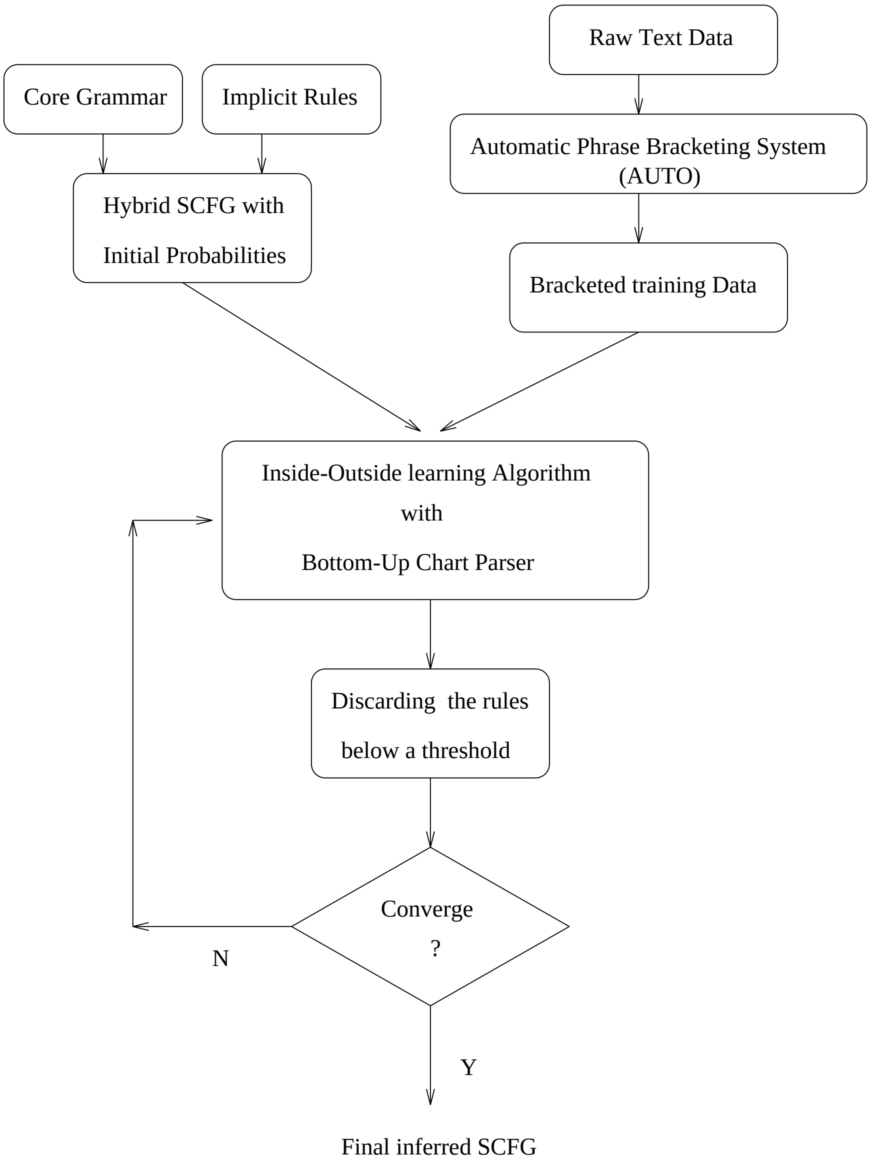

Figure 1: A Block Diagram of the CAGC System

3 Grammatical Inference

3.1 Extended Inside-Outside Algorithm

The original inside-outside algorithm was extended to take into account constituent information in a phrase-bracketed training set [Pereira and Schabes, 1992] [Black et al., 1992]. Given a series of observations (terminals) $O = O_1, O_2, ..., O_T$ the extended learning procedure considers only a rule i whose span $O_s, ..., O_t$ is compatible with the *a priori* bracketing of the observations. This constraint was implemented by a boolean function A(s,t):

$$\text{A(s,t)} = \begin{cases} 1 & \text{if (s,t) compatible with the } \textit{a priori} \text{ bracketing} \\ 0 & otherwise \end{cases}$$

The compatibility of two spans of the observations is defined as follows: given two spans (s,t) and (p,q), where (s,t) = $O_s, ..., O_t$ and (p,q) = $O_p, ..., O_q$, (s,t) and (p,q) are incompatible if they overlap in a situation, where either $O_s, .., O_p, .., O_t, .., O_q$ or $O_p, .., O_s, .., O_q, ..O_t$ where $p \leq t$, $s \leq q$, $s \neq t$ and $p \neq q$. For example, a bracketed training sentence with word indices:

$$(\ (The_1\ apples_2)\ (\ (are_3)\ (ripe_4)\ )\ )$$

provides a set of five bracketing spans:

$$\beta = \{\ (1,4),\ (1,2),\ (3,4),\ (3,3),\ (4,4)\ \}$$

The word string *'apples are'* whose span is (2,3) is not compatible with the brackets (1,2) and (3,4) in β.

As a result, the normal inside probability e(s,t,i), the probability of i generating $O_s, ..., O_t$, and the outside probability f(s,t,i), the probability of i being generated and not involved in the generation of $O_1, ..., O_{s-1}$ and $O_{t+1}, ..., O_T$, are modified as

$$e^{'}(s,t,i) = \text{A(s,t) e(s,t,i)}$$
$$f^{'}(s,t,i) = \text{A(s,t) f(s,t,i)}$$

Hence, the inside probability $e^{'}(s,t,i)$ is set to zero if the span (s,t) violates any of the *a priori* brackets otherwise it is unchanged, and similarly for the outside probability $f^{'}(s,t,i)$.

3.2 Core Grammar Development

A grammar G3 used throughout this work is a modified version of the grammar G2 defined by Briscoe and Waegner [Briscoe and Waegner, 1992]. In stead of lexical entries, parts-of-speech (POSs) are used in these grammars in order to reduce computational expense. G3 is a feature-based unification grammar developed using the Grammar Development Environment (GDE) [Carroll et al., 1991]. The GDE grammars take the Alvey Natural Language Tools Grammar formalism [Grover et al., 1993], which is similar to that of Generalized Phrase Structure Grammar (GPSG) [Gazdar et al., 1985]. In the GPSG's feature theory, each terminal and non-terminal can be viewed as a set of feature-value pairs. For example, NP (noun phrase) and VP (verb phrase) are two non-terminals and their corresponding sets of feature-value pairs might be

NP : [N +, V -, PLU +]
VP : [N -, V +, PLU -].

N, V and PLU (plural) are features and their possible values are + and -. [N +, V -, PLU +] indicates NP is a plural noun phrase such as *'the apples (DT NNS)'*, whereas [N -, V +, PLU -] represents a singular verb phrase such as *'is ripe (BEZ JJ)'*.

This feature theory is integrated with the operations of unification and propagation, which are the processes of checking and binding features and their values. For instance, the rule

$$S \rightarrow NP[PLU\ @x]\ VP[PLU\ @x]$$

embodies the principle of subject-verb agreement since it states that the values of the PLU feature in the NP and VP are bound to be the same (@x represents a variable). Therefore, the sentence *'The apples is ripe (DT NNS BEZ JJ)'* will fail during the unification process since the values of their PLU feature are not consistent. The value of a feature in a daughter node can also be carried up to its parent node. For instance, the rule

$$VP[PLU\ @x] \rightarrow V1[PLU\ @x]\ PP$$

indicates that V1 passes to its parent VP whatever the value of its PLU feature is.

Table 1 shows the broad characteristics of the G3 grammar. There are a total of 114 feature-based explicit rules using 6 non-terminals(NT) V2, V1, N2, N1, P1 and A1. There are also 12 pre-terminals(PT) whose main function is to bring the POSs into the grammar's formalism. Finally, there are 18 features whose number of values vary from 2 to 7, but the average number of values per feature is 3.

Rules	**NT**	**PT**	**POS**	**Features**	**(Average) Values/Feature**
114	6	12	62	18	3

Table 1: Broad Characteristics of the G3 Grammar

3.3 Implicit Rules Generation

The implicit extension to the G3 grammar was generated by the technique used in [Briscoe and Waegner, 1993] [Waegner, 1993]. This technique employed four templates to generate a set of context-free (CF) binary rules

$$NT \rightarrow NT\ NT$$
$$NT \rightarrow PT\ PT$$
$$NT \rightarrow PT\ NT$$
$$NT \rightarrow NT\ PT$$

where $NT \in \{V2, V1, N2, N1, P1, A1, H2, H1\}$
and $PT \in \{V0, N0, A0, P0, H0, DET, CSS, PT, CJ, IJ, NG, PDET\}$.
These four templates would generate 3200 CF rules if all possible combinations of NT and PT were allowed. However, a large proportion of these rules are linguistically implausible because they violate the constraint that a phrase must have a head, that is, a NP must contain a noun and a VP must contain a verb, and so forth. These

headedness constraints are therefore used to filter the rules generated by the above templates.

The full set of headedness constraints are derived from X-Bar Theory [Jackendoff, 1977]. In the theory, most phrasal contituents have heads on which elements of the contituents are dependent and certain elements of a contituent are more closely related to the head than the others. This can be explained using the diagram in Figure 2. X is any of the

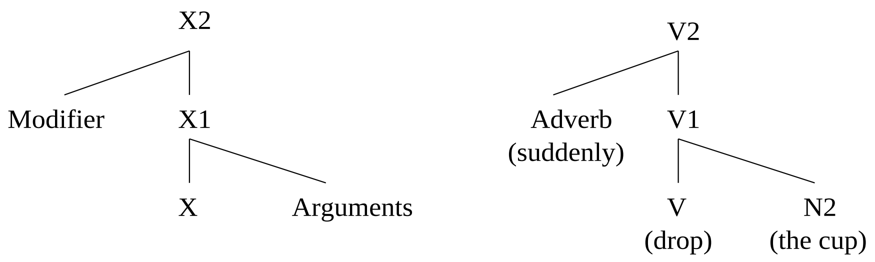

Figure 2: A Diagram of X-Bar Theory and an Example of Its Use

categories in the grammar, X1 (Bar 1) and X2 (Bar 2) are the phrasal categories. X1 is the mother of X (head) and X's argument(s); X2 is the mother of X1 (head) and X1's modifier(s). In the example, a verb *'drop'* and its N2 argument *'the cup'* form the V1, which takes the adverb *'suddenly'* as a modifier to form the V2, a verb phrase. The headedness constraints ensure that at least one daughter on the right-hand side of the rule has the same category as its mother and its Bar number is one less than that of the mother.

After applying the headedness constraints, 234 implicit CF rules remained. In order to be consistent with the core grammar, this set of the implicit CF rules was converted to the feature-based formalism by attaching the default features and values to each rule symbol. The two sets of rules, 114 explicit and 234 implicit, were then combined to form a hybrid feature-based grammar.

3.4 Hybrid SCFG

To convert the hybrid feature-based grammar into an SCFG, it was necessary to expand all possible values of every feature in the grammar. The size of the grammar after expansion depends on the number of features and the number of their corresponding values. However, in our grammar this expansion makes the size too large to handle and also reintroduces problems of data insufficiency. Instead, this expansion process was simplified by not utilizing the NTYPE (types of nouns) feature, which is relatively less important but tends to have a larger value set than other features. Although this increases the number of spurious analyses, the reduced size of the grammar makes the inside-outside re-estimation technique practical. After expansion on the 348 feature-based rules, a set of 14736 CF rules is generated, which is about 42 times more than the original set. This expended grammar includes 5949 explicit rules and 8787 implicit rules. A comparison between the feature-based core grammar and the hybrid SCFG is shown in Table 2, in terms of the number of rules, non-terminals, pre-terminals and parts-of-speech.

The initial probabilities of the rules in the SCFG are then calculated in the following manner: given a rule subset A, where the rules have the same non-terminals on the left-hand side (the same parent), the initial probability of each explicit rule (P_e) and implicit rule (P_i) is calculated as

	Rules	NT	PT	POS
Feature-Based Core Grammar	114	6	12	62
Hybrid SCFG	14736	280	128	62

Table 2: A Comparison between Feature-Based Core Grammar and Hybrid SCFG

$$P_e = \frac{0.9}{N} \qquad \text{and} \qquad P_i = \frac{0.1}{M},$$

where N and M are a total of explicit and implicit rules in A respectively. The weighting parameters for the explicit and implicit rules were empirically chosen in the grammar development. The above calculations ensure that the explicit rules get a better start during training, and also that the initial probabilities of all the rules with the same parent sum to 1.

4 Automatic Phrase Bracketing System (AUTO)

In the inside-outside based GI, constituent information of the training data provides a good constraint during the re-estimation process as stated in the introduction. However, generating a hand-parsed training corpus such as the Penn treebank [Santorini, 1991] used in [Pereira and Schabes, 1992] is very labour-expensive and this manual work must be repeated each time a new training corpus is required. In order to overcome this problem, an automatic algorithm is needed to derive the constituent information for the training corpus. The AUTO system has been designed for this purpose. The output of the AUTO bracketing of the corpus is used to assist the inside-outside based grammar re-estimation in the CAGC system.

The AUTO system is a surface-bracketing technique, which employs heuristic knowledge about phrase structures of sentences in a language, to group various kinds of constituents along a tagged sentence. The system applies the heuristic knowledge in a manner that combines both the top-down and bottom-up approaches. It is observed that some of the constituent information is easier to capture from a global view (the top-down approach) such as the subject and predicate relation, whereas other information can be identified locally (bottom-up approach) such as a noun phrase starting with a determiner or a verb phrase beginning with an auxiliary. Because of this, the AUTO system applies heuristic rules in both a top-down and a bottom-up fashion, whichever is more appropriate to deal with each type of constituent.

The various types of constituent cannot all be processed independently. Some larger constituents such as clauses rely on smaller constituents to be formed in advance. Also, interaction between two different constituents sometimes happen. As a consequence, the rules used in the AUTO system are applied in a prescribed order which has been determined empirically. The algorithm of the AUTO system is given in detail with an example in Section 4.1 and its performance is given in Section 4.2.

Alternative approaches to surface bracketing are based on the statistical properties of constituent information in sentences . An example of this kind is a constituent boundary parsing algorithm proposed in [Magerman and Marcus, 1990]. In this algorithm, generalized mutual information (GMI) was used to extract boundary information within a limiting window along a tagged sentence.

4.1 Bracketing Algorithm of the AUTO System

The bracketing algorithm of the AUTO system falls into stages: to begin with, the input (a tagged sentence in the WSJ corpus) is segmented into basic constituents of sentence structures. The linking of conjuncts is then introduced to construct conjunction. This bottom-up process creates subtrees on top of the input tags and gives a better view of the structure of the whole sentence than that which is provided by a bare input tag sequence. A top-down procedure starts with the whole sentence S and then finds the key constituents to form the clauses from which a subject and a predicate are bracketed. The flowchart shown in Figure 3 illustrates the processing stages in this bracketing system. Each stage is described in detail in the following sections.

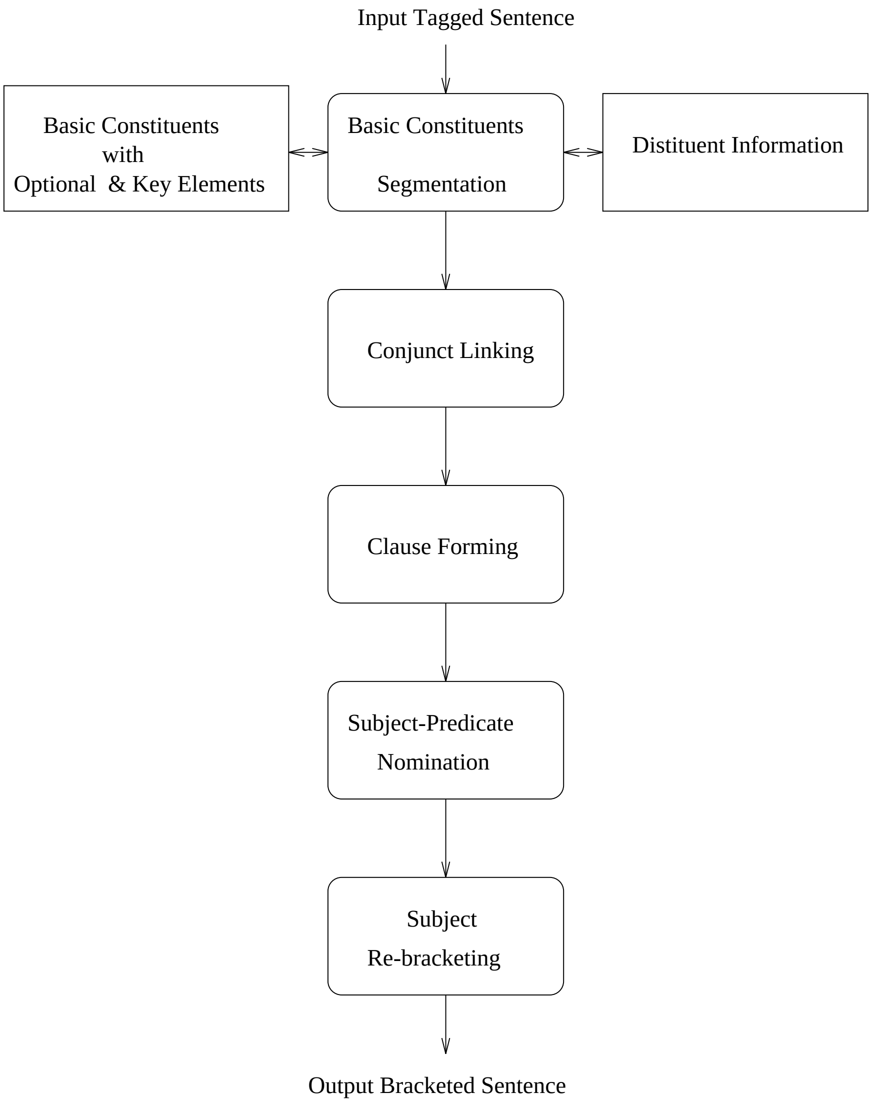

Figure 3: Flowchart of the AUTO System

4.1.1 Segmentation of Basic Constituents

As shown in the flowchart, there are two kinds of information needed in order to segment the input tag sequence into basic constituents. One is the list of labels of basic constituents and their associated optional and key elements. Basic constituents are phrases or tags that are the foundation of the sentence phrase structures defined in the AUTO system. An optional element of a basic constituent is a tag that is not essential to the basic constituent such as a determiner in a NP, and a key element is a vital tag (the head) of the basic constituent such a noun in a NP. The second kind of information is a list of distituents (a pair of tags that cannot be adjacent within a constituent as defined in [Magerman and Marcus, 1990]). This distituent list provides information about potential phrase boundaries.

The segmentation starts by matching each adjacent tag pair with the distituent list along the input sequence. When a potential phrase boundary is detected, the longest phrase is looked for among the processed tags. The longest phrase would be the constituent whose vital element(s) is in the proper position (usually at the right end, but not for all the cases.) For example, in the following NP

$$this_{DT}\ child_{NN}\ discarded_{VBN}\ by_{IN}\ his_{PP\$}\ parents_{NNS}$$

there are two distituents, (VBN,IN) and (IN PP$) found in the distituent list. When the first distituent (VBN,IN) is found, we look for a longest phrase among the data *'this child discarded'*. The word *'this'* can be a NP itself but *'this child'* is a longer NP. The tag sequence *'the child discarded'* cannot be a NP because the tag VBN is not vital but optional to NPs as in a phrase *'a discarded child'* and therefore it violates the principle that the head of NPs should be at the right end. As a result, *'this child'* is nominated as a NP at this stage.

Once the phrase is selected, the segmentation process resumes at the tag VBN which is right adjacent to the phrase found. By following this procedure, an input sentence *'She told the parents the child discarded by them was adopted by a couple who lost their son and daughter in a car accident.'* is bracketed as follows:

(S (She) (told) (the parents) (the child) (discarded) (by) (them) (was adopted) (by) (a couple) (who) (lost) (their son) (and) (daughter) (in) (a car accident)).

4.1.2 Conjunct Linking

After segmenting the input sentence into basic constituents, the next step is to join the conjuncts. Conjunction is a long-standing problem in parsing because of the variety of possible forms. In this system, the following two patterns are employed to deal with some well-formed conjunction structures:

X cj X
and
Y...cj Y... cj Y...

where X and Y are any basic constituents except for cj, a conjunction.

Pattern matching starts from right to left and finishes when no more patterns can be found. As a result, the latter half of the example sentence is further bracketed as

(S ..... (a couple) (who) (lost) *[(their son) (and) (daughter)]* (in) (a car accident)).

In order to emphasize the current step in the running example, the newly introduced constituent(s) at each stage are square-bracketed and in *italics*. Also, the brackets generated from the former stages may be omitted in the later stages.

Note that the phrase *'their son and daughter'* should be precisely bracketed as *(their (son and daughter))* . In other words, the step of conjunct linking in this case should be taken before the segmentation of basic constituents. However, there is not sufficient information in a tag sequence to decide which step should be taken first. The order of rule applications in the current system covers the majority of the cases and minor errors such as these are ignored.

4.1.3 Clause Forming

To process clauses, the leading constituents of the clauses, namely subordinating conjunctions (CSs) such as *because*, wh-adverbs (WHRBs) such as *when*, and wh-nouns (WHNPs) such as *who*, are searched from right to left along the output from the previous stage. When one of the clause leading constituents is found, the end of the clause is located in the following manner. For CS-clauses the ending is defined as the end of the sentence; for WH-clauses the ending is at the constituent just before the second verb counted from the leading constituent, otherwise it is at the end of the sentence. This way of finding the ends of the clauses correctly brackets the majority of the cases. In the running example, a WH-clause starts from the word *'who'* and ends at the end of the sentence since there is no second verb.

(S ..... (a couple) *[(who) (lost) (their son and daughter) (in) (a car accident)]*).

There is one auxiliary procedure to further process WH-clauses. WH-clauses are one type of NP modifiers and are usually referred as *relative clauses*. A WH-clause and its associated NP need to be bracketed as a constituent. As a result, the sentence is further bracketed as

(S ... *[(a couple) (who lost their son and daughter in a car accident)]*)

4.1.4 Subject-Predicate Nomination

Nomination of a subject (SUB) and a predicate (PRD) is done within each clause and the sentence itself. The boundary between them is set at a verb group (VG), one of the basic constituents. Any constituents before a VG are bracketed as a subject; constituents starting from the VG to the end of the clause/sentence are bracketed as a predicate. Within the predicate, an object (OBJ) is formed from the constituent adjacent to the VG to the end of the clause/sentence. The same procedure recursively applies to each object until no further VG is found in an object.

In order to make the following example clearer, the labels of SUB, PRD and OBJ constituents are shown, Also, to distinguish SUBs/PRDs/OBJs at different levels of the phrase structures, they are indexed according to the depth of the levels at which they are located. Following the principle of Subject-Predicate nomination, the running example is first bracketed as

(S [SUB_1 She] [PRD_1 told [OBJ_1 (the parents) (the child) (discarded) (by) (them) (was adopted) (by) (a couple who ..... in a car accident)]]).

and then the OBJ is recursively bracketed as

(S ..... (OBJ_1 [SUB_2 (the parents) (the child) (discarded) (by) (them)] [PRD_2 (was adopted) [OBJ_2 (by) (a couple who ..... in a car accident)])).

The same principle applies to the WH-clause, which is then bracketed as follows:

(S ..... (a couple (*[SUB_3 who] [PRD_3 lost [OBJ_3 (their son and daughter) (in) (a car accident)]]*)) ...).

Once the SUBs, PRD and OBJs are formed, one auxiliary function at this stage is evoked to process the constituents within the SUBs and OBJs. This function searches for four categories: gerunds, participles, infinitives and prepositions, from right to left within a SUB or OBJ. When one of the four categories is found, the category and the rest of the elements on its right-hand side are bracketed as a constituent. The search carries on until no more categories are found in the SUB/OBJ.

In the example sentence, the SUB_1 and SUB_3 remain unchanged since they do not contain any of the four categories. In the SUB_2, *(the parents) (the child) (discarded) (by) (them)*, the preposition *by* is first found and a preposition phrase is formed as

(S ... (SUB_2 (the parents) (the child) (discarded) *[(by) (them)]*)...).

Then, the participle *discarded* is found afterwards and a participle phrase bracketed as

(S ... (SUB_2 (the parents) (the child) *[(discarded) (by them)]*)...).

As for the OBJs, the OBJ_1 remains intact, while OBJ_2 and OBJ_3 have preposition phrases *((by) (a couple ...))* and *((in) (a car accident))* respectively.

In order to accommodate certain possible ambiguities, the bracket of an OBJ which does not contain any VG is undone at the end of this stage. This allows the PRD to have more than one possible constituent structures later on in the training process. In the running example, the bracket for OBJ_3 is undone and the new PRD_3 is formed as

(S *...[PRD_3 lost (their son and daughter) (in (a car accident))]*).

Note that the NP *(the parents)* in the SUB_2 is in fact another object at the level 1, and should be excluded from the SUB_2 and moved to the upper level as the other OBJ_1. For this kind of situation, the process of re-bracketing the subjects become necessary.

4.1.5 Subject Re-Bracketing

The procedure of re-bracketing subjects is to locate the most probable subject constituents within a SUB, and nominate the rest of the constituents as an object and move this object to the upper level. The most probable subject is defined as the last NP in a SUB. Any constituents before the last NP are bracketed as an OBJ and moved to the upper level. The NP and whatever follows in the SUB remain as the subject. According to this principle, *(the parents)* is moved from SUB_2 to the level 1 as another OBJ_1; whereas *(SUB_2 (the child) (discarded by them))* remains at the level 2.

At the end of this stage, the complete bracketed sentence is generated as follows:

```
(S (She)
   ((told)
    (the parents)
    (((the child)
        ((discarded) ((by) (them))))
     ((was adopted)
        ((by) ((a couple)
                  ((who)
                   ((lost)
                    ((their son) (and) (daughter))
                    ((in) (a car accident))))))))))).
```

4.2 Performance

The advantage of using the AUTO system in the CAGC task is that it is automatic and therefore there is no manual work in bracketing the training data for the extended inside-outside based grammar learning. Nevertheless, the system has a disadvantage: because it employs general heuristic rules, it cannot cope with sentences with unusual structures and which are relatively rare. As a result of this, high bracketing error rates occasionally occur in some sentences.

To evaluate the accuracy of the AUTO system, both the training and test data (2021 sentences in total), which were used respectively for the inside-outside learning process and later for the evaluations of the CAGC system, were bracketed by the AUTO system and the results were compared with their Penn tree bank counterparts. The details of these sentences including the average length and their distributions against the length are given in Section 5.1. Table 3 shows a performance evaluation of the AUTO system.

	Training set	**Testing set**	**Total**
Recall	84.51%	85.65%	85.08%
Precision	77.55%	78.33%	77.94%
Crossings	1.12	1.05	1.09

Table 3: Bracketing Accuracy of the AUTO system

Parseval, an evaluation method used in this paper, was developed by the Grammar Evaluation Interest Group (GEIG) [Black et al., 1991] [Thompson, 1992]. In order to create a basis for comparison between standard Penn treebank and evaluated parses, they suggest three preprocessing steps: firstly, erase auxiliaries, “not”, pre-infinitival “to”, null categories, possessive endings (’s and ’) and punctuation; secondly, delete bracket labels such as NP and VP; thirdly, remove all bracketings containing only one constituent. Parseval uses three metrics to evaluate the accuracy of a parse, namely, Recall, Precision and Crossing. Recall is the percentage of standard bracketings (in the Penn treebank) that are present in the evaluated parses. Precision is the percentage of bracketings in the evaluated parses that are present in the Penn treebank. Partial overlapping between a pair of standard:evaluated bracketings is called “cross” and Crossings is the average number of bracketings in the evaluated parses that cross the standard bracketings.

It can be seen from Table 3 that total Recall is over 85% which shows the high degree of closeness bwteen the Penn treebank and AUTO generated parses. Precision is about 78% which is lower than Recall due to the fact that the Penn treebank is only partially parsed and therefore the parses tend to be flatter than AUTO generated parses. On average there is only slightly more than two crossing errors in each sentence. Since the AUTO system derives constituent information for the training data from which a SCFG is trained later on in the CAGC task, the quality of the inferred grammar will depend on the extent to which crossing errors affect the subsequent re-estimation process.

5 Experimental Evaluation

5.1 Training and Testing Sets

Training and Testing data were randomly chosen from the Wall Street Journal(WSJ) text corpus. Since our current system cannot handle punctuation and brackets, sentences containing them were put aside for future work. In addition, quotation marks in sentences were ignored. In the final set, there were 1521 training sentences and 500 testing sentences in our WSJ subset.

There is no explicit limit on sentence length imposed by our system. Figure 4 shows the distributions of sentence length in the training and the test sets. The peak of the sentence length is around 12 and 13, and average length of the sentences is 16 words.

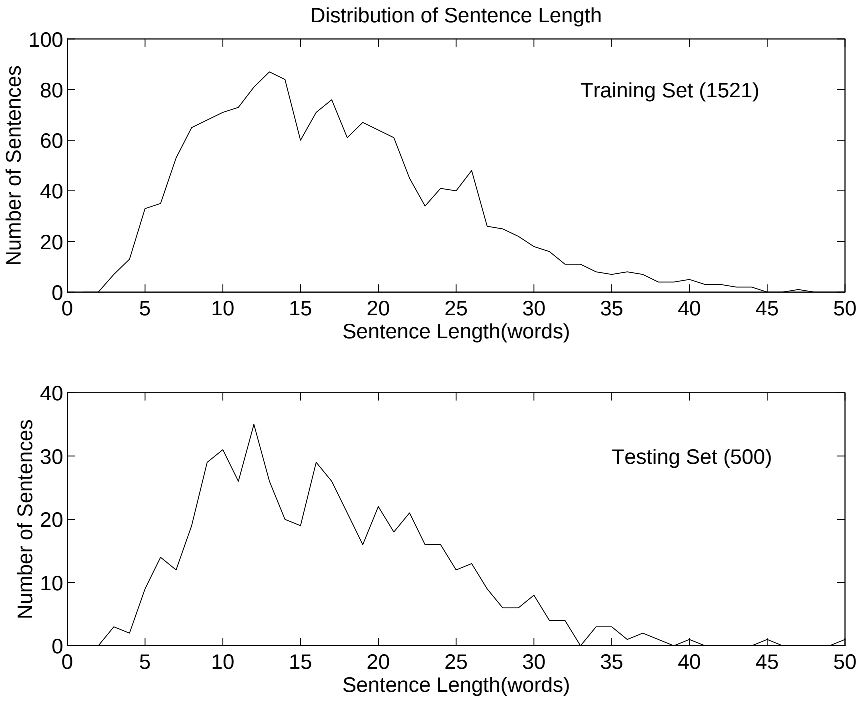

Figure 4: Distribution of Sentence Length

5.2 Modified Parts-of-Speech Set

In order to capture more detailed syntactic information, some of the POS used in the WSJ text corpus were subcategorized. This increased the number of POS from 48 to

62. Table 4 shows the original POSs which were modified and their new corresponding subcategorizations. It also shows examples of words for the new POS.

Old Part-of-speech	New Part-of-speech
CC	CC CC2(both,either,neither)
DT	DT AT(a,an)
IN	IN CS(although,etc) CSIN(until,etc) INA(than,as) INO(of)
NN	NN TMP(today,Monday,etc)
PP	PP PPI(she,he,it)
RB	RB NT(not)
TO	TO IN
VB	VB HV(have) DO(do) BE(be)
VBP	VBP BEP(are,am)
VBZ	VBZ HVZ(has) DOZ(does) BEZ(is)
VBD	VBD HVD(had) DOD(did) BED(were) BEDZ(was)
VBG	VBG BEG(being)
VBN	VBN BEN(been)

Table 4: Modified set of Parts-of-Speech

5.3 Training Grammars with/without Constituent Information

Three experiments were carried out to investigate the extent to which bracketing information is utilized during the inside-outside re-estimation process and its effect on the final trained grammar. In the first two experiments, the explicit and hybrid grammars were trained from the raw corpus (NULL_BC i.e. no bracketing constraint). In the third experiment, the hybrid grammar was trained from the AUTO bracketed corpus (AUTO_BC).

Grammar Types	Explicit	hybrid + NULL_BC	hybrid + AUTO_BC
Rules Remaining After Training	33.99% (2022/5949)	40.91% (6029/14736)	18.54% (2733/14736)
Exp:Imp rules	2022:0	3290:2739	1316:1417
Sents. Parsed	89.80% (449)	98.60% (493)	97.20% (486)
Recall	76.02%	70.76%	84.66%
Precision	58.56%	54.55%	62.50%
Crossings	2.86	3.49	2.14
Sents. All Parsed	89.20% (446)	89.20% (446)	89.20% (446)
Recall	76.11%	71.35%	85.58%
Precision	58.59%	55.23%	63.43%
Crossing	2.87	3.35	2.00

Table 5: Performance of the Explicit and Hybrid Grammars Trained From Raw/Bracketed Corpus

Table 5 shows the results for these three trained grammars. It records the number of SCF rules which survived after training, the ratio of explicit and implicit trained rules, the coverage of each grammar (the number of test sentences that each grammar successfully parsed), and their individual Recall, Precision and Crossings. The second section of the table records the number of test sentences successfully parsed by all three grammars and Recall, Precision and Crossings for these sentences only.

From Table 5, it can be seen that both the hybrid grammars achieved significantly greater coverage than the explicit grammar. After training, the AUTO_BC grammar has less than half the number of rules of the NULL_BC grammar and its only disadvantage is a slightly reduced coverage. Although the NULL_BC grammar has the best coverage among three, its performance is much worse than either the explicit or AUTO_BC grammars. It can be seen in the second section of the table that the AUTO_BC grammar has considerable improvement by about 14% and 9% in Recall, 8% and 5% in Precision, and 40% and 30% reduction in Crossings, compared to the NULL_BC and explicit grammars respectively.

5.4 Convergence of the Training Schemes

Figure 5 shows the convergence of the training process for the hybrid NULL_BC and AUTO_BC grammars, where the cross entropy is calculated as follows:

$$\text{Cross Entropy } \hat{H} = -\frac{\sum_{c \in C} logP^c}{\sum_{c \in C} |c|}$$

where C is the training corpus, $logP^c$ is the total log probability of a sentence c and $|c|$ is its length.

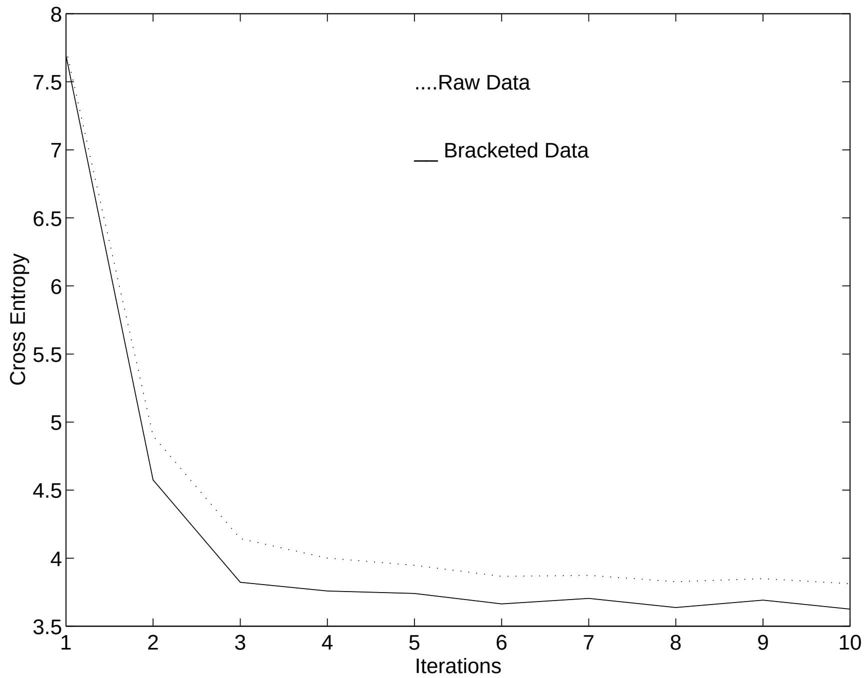

Figure 5: Convergence of the Re-estimation process for the NULL_BC and AUTO_BC hybrid grammars

As can be seen, both grammars had similar convergence properties. The cross entropies decreased rapidly for the first three iterations and moved slowly afterwards. However, with the constituent information, the AUTO_BC training process achieved a lower cross entropy at each iteration. The results of the experiment in Section 5.3 suggest that this lower cross entropy resulted in a superior convergence of the AUTO_BC training process.

5.5 Complexity of the Original and Extended Inside-Outside Algorithm

The complexities of the original and extended inside-outside algorithms were recorded and a comparison was made in Table 6. The NULL_BC hybrid grammar was used for calculating the CPU time of the original algorithm and the AUTO_BC grammar for that of the extended algorithm. Both were trained on an HP series 735 machine.

Algorithm	**Original**	**Modified**
CPU Time in Minutes	507.34	17.37

Table 6: Complexities of Original and Extended Inside-Outside Algorithm

As shown in the table, the original algorithm spent 8 hour and 27 minutes CPU time whereas the extended algorithm took less then 18 minutes. This significant improvement indicates that the modified algorithm overcomes the main inherent weakness of the inside-outside algorithm by making it computationally tractable for use in grammatical inference.

5.6 Training the Hybrid Grammar from Hand-Parsed Corpus

In this experiment, the hybrid grammar was trained using the hand-parsed Penn tree bank (PENN_BC) training corpus. The purpose of this experiment was to compare the effectiveness of the AUTO generated brackets with the Penn tree bank brackets. Table 7 shows the performance comparison between these two approaches.

Types of Training Corpora	**AUTO_BC**	**PENN_BC**
Rules Remaining After Training	18.54% (2733/14736)	21.29% (3137/14736)
Exp:Imp rules	1316:1417	1482:1655
Sent. Parsed	97.20% (486)	97.80% (489)
Recall	84.66%	84.65%
Precision	62.50%	64.06%
Crossings	2.14	1.92
Sent. Both Parsed	96.20% (481)	96.20% (481)
Recall	84.76%	84.76%
Precision	62.67%	63.98%
Crossings	2.13	1.93

Table 7: A Performance Comparison between AUTO_BC and PENN_BC Grammars

As can be seen, the performances of these two grammars are very competitive. The coverage of the PENN_BC grammar is only 0.6% ahead of the AUTO_BC grammar and their Recall and Crossings are only marginally different. The second section of the table shows that PENN_BC grammar has only 1.3% ahead of AUTO_BC grammar in Precision, 0.2 reduction in Crossings and there is no difference in terms of Recall. These results illustrate that although the AUTO system makes mistakes as shown in Table 3, they do not seem to undermine the learning process significantly.

5.7 Effects of Unique Rules in AUTO_BC/PENN_BC Hybrid Grammars

The inferred AUTO_BC and PENN_BC grammars contained rules which only existed in one but not the other. These rules are referred to here as unique rules. It was observed that the existence of the unique rules in the grammars arose because of the differences between the two bracketing systems. In the AUTO system, the unique rules were encouraged by the system's inherent errors (see Table 3). In the hand-parsed Penn tree bank, the unique rules emerged probably from two sources: one was sentences whose structures were rare and therefore required special rules during training, the other was partially-parsed phrases which allowed spurious rules to participate in re-estimation process.

To investigate the effects of these unique rules, they were separated from the AUTO_BC and PENN_BC grammars. Test sentences whose Viterbi parse involved at least one of the AUTO_BC unique rules were then extracted as the first set, and similarly the sentences involving the PENN_BC unique rules were extracted as the second set. Table 8 shows the performance metrics for these two sets of extracted sentences.

Types of Hybrid Grammars	**AUTO_BC**	**PENN_BC**
The Percentage of Unique Rules	20.20% (552/2733)	30.47% (956/3137)
Exp:Imp rules	179:373	345:611
Sent. Involved	19.80% (99)	44.20% (221)
Recall	73.63%	78.78%
Precision	52.22%	58.23%
Crossing/Sent.	3.97	2.86
Sent. Both Involved	11.40% (57)	11.40% (57)
Recall	72.52%	75.94%
Precision	50.51%	54.26%
Crossing/Sent.	4.54	3.65

Table 8: Results from Unique Rules

As shown in the table, about 20% of rules in AUTO_BC grammar were unique (ie. they do not exist in the PENN_BC grammar) and similarly 30% of rules in the PENN_BC grammar were unique. It can also be observed that there were more than twice as many test sentences that needed PENN_BC unique rules than sentences that needed AUTO_BC unique rules. However, this might be expected since the PENN_BC grammar contains a 10% greater proportion of unique rules. In addition, the PENN_BC's "custom tailored" unique rule set seemed to be more specific to certain types of uncommon syntactic structures than the AUTO_BC set. This feature can also explain why the PENN_BC set involved more test sentences.

Putting aside their individual performance, the second section of table 8 shows a comparison of the sentence parses which contain respective unique rules for both the grammars. It can be seen that the sentence parses which include PENN_BC unique rules performed better than the AUTO_BC set in all three metrics. It is believed that PENN_BC unique rules were more accurate models of some particular syntactic structures. This would help explain why the PENN_BC grammar performed 3.4% better than the AUTO_BC grammar in Recall, 3.8% in Precision, and 1.1 reduction in Crossings,

It is worth noting that the figures for unique sets shown in Table 8 are much worse than those given in Table 7 for the full rule sets. It was mentioned in previous section that the performances of the two grammars were very competitive in terms of the full rule sets. Therefore, there must be a subset of test sentences where the AUTO_BC grammar performed better than the PENN_BC grammar to compensate the figures in Table 8.

In Table 9, the results from four subsets of test sentences were calculated and compared. These four partitions were made according to properties of the rules (unique/common) used in Viterbi parses. For example, Set A contained 57 test sentences whose Viterbi parse involved at least one unique rule in the case of the AUTO_BC and also in the case of the PENN_BC (as in Table 8); whereas Set B was composed of sentences whose Viterbi parse involved only common rules in the AUTO_BC but at least one unique rule in the PENN_BC.

Types of Rules Involved	**No. of Sents.**	**Recall**		**Precision**		**Crossings**	
		auto	**penn**	**auto**	**penn**	**auto**	**penn**
Unique AUTO and Unique PENN (A)	57 11.9%	72.52	75.94	50.51	54.26	4.54	3.65
Common AUTO and Unique PENN (B)	156 32.4%	81.98	79.85	57.17	59.16	2.65	2.64
Unique AUTO and Common PENN (C)	37 7.7%	75.15	75.67	55.69	55.37	3.16	2.78
Common AUTO and Common PENN (D)	231 48.0%	91.20	91.71	70.51	71.02	1.01	0.89
TOTAL (A+B+C+D)	481	84.76	84.76	62.67	63.98	2.13	1.93

Table 9: Results from Four Combinations of rule types

From table 9 it can be seen that to varying extents the PENN_BC grammar performs better than the AUTO_BC grammar for all subsets except Precision in Set B. . This shows that the unique PENN rules somehow were misused and adversely affected performance in Set B. The performance of the grammars is comparable for Sets C and D; so an important difference between the grammars is the observation that the negative effects of unique rules are distributed in a larger proportion (32.4%) of the sentences for the PENN_BC grammar than the AUTO_BC grammar.

5.8 Performance vs. Sentence Length

It was stated previously that there was no limit on the length of sentences used in this work. Figure 6 shows the effect of the length of the test sentences on the performance of both the AUTO_BC and PENN_BC grammars. As can be seen, for the sentences of 15 words or less Recall is as high as almost 90%, Precision 70% and Crossings is below 1 for both grammars. However, the three performance metrics degraded as the length of the sentences increased.

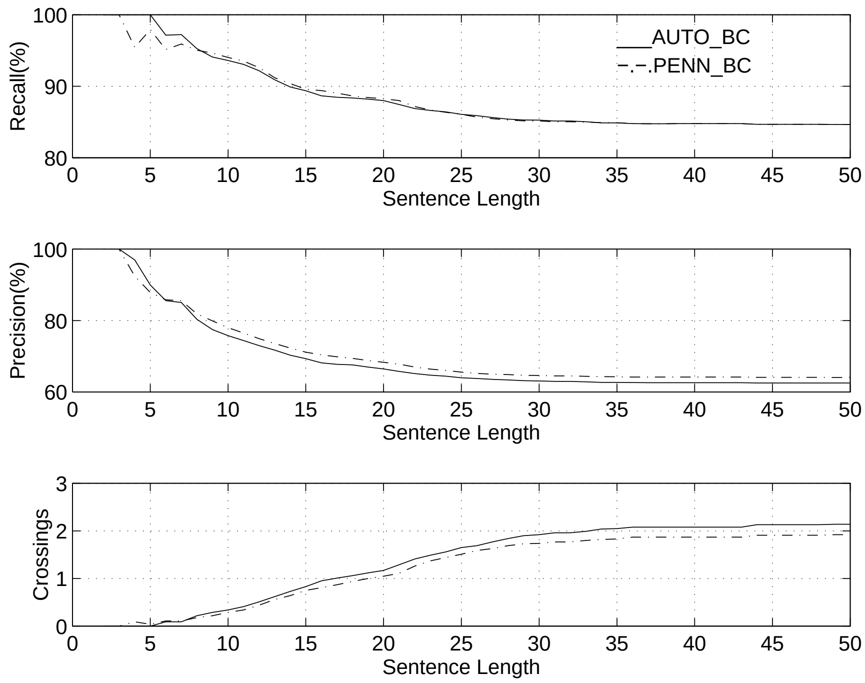

Figure 6: Performance vs. Sentence Length

5.9 Number of Parses Generated from Trained Grammars

The number of parses generated for a sentence varies for the differently trained grammars. This experiment was carried out to compare the differences among the four trained grammars and the results are shown in Figure 7.

The top part of Figure 7 shows a comparison of the three grammars trained as described in Section 5.3. The explicit grammar had far fewer parses simply because it did not contain any implicit rules. However, the AUTO_BC hybrid grammar generated more ambiguous parses than the NULL_BC hybrid grammar, although the size of the AUTO_BC grammar was less than that of the NULL_BC grammar. A similar result can be observed in the bottom part of Figure 7, where a comparison is made between the AUTO_BC and PENN_BC grammars. Hence, the AUTO_BC grammar generated more ambiguous parses than the NULL_BC grammar and the PENN_BC grammar, although both of these contained more rules than the AUTO_BC grammar.

This can be explained by our belief that the number of parses generated from a grammar for a sentence not only depends on the size of the grammar but also on the type of rules the grammar has. The AUTO_BC grammar contained more general rules than the other two, because the grammar was trained from a corpus whose bracketing information was provided by the heuristic-based AUTO system. However, the important point is that among the large set of ambiguous parses generated by the AUTO_BC grammar, the Viterbi algorithm still successfully chose parses that lead to good overall performance.

6 Conclusions

A system for computer assisted grammar construction has been presented in this paper. The aim in developing this system was to efficiently infer a broad-coverage and linguistically-motivated grammar for a large corpus without relying on significant

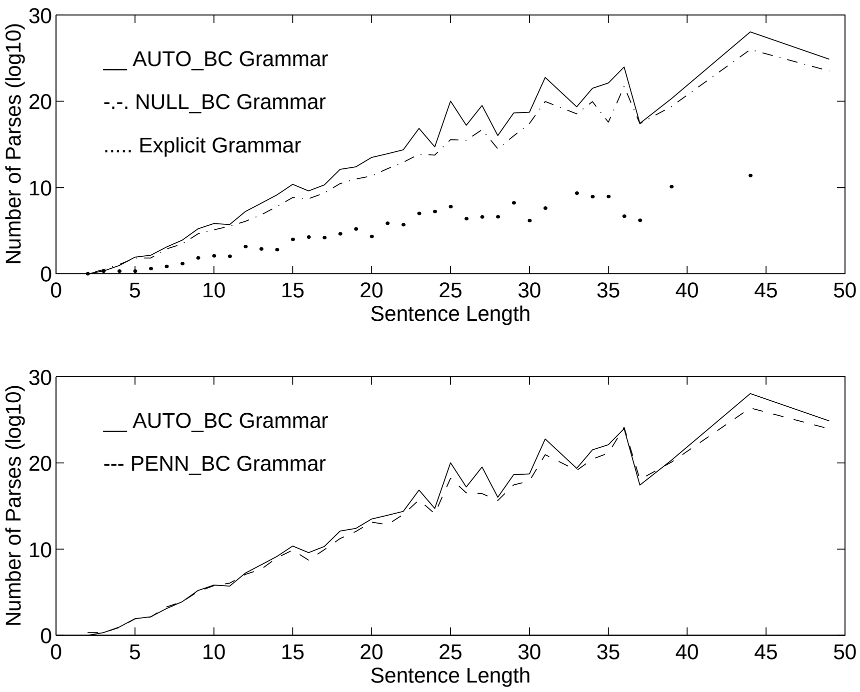

Figure 7: Number of Parses vs. Sentence Length in Differently Trained Grammars

manual labour. The experimental results demonstrate that the CAGC system can successfully infer a grammar for a subset of the WSJ corpus which has acceptable coverage and precision. Two techniques employed in the system contributed to this success.

Firstly, the method of generating an initial SCFG ensures broad-coverage of the inferred grammar and provides good bootstrapping for the learning process. The initial SCFG includes a set of explicit rules (the core grammar), which is hand-produced, and a set of implicit rules that compensates for the limiting coverage of the core grammar. Additionally, giving high initial probabilities for the explicit rules and low probabilities for the implicit rules means that the system is biased towards a linguistically-motivated grammar.

Secondly, the extended inside-outside algorithm used for the grammar re-estimation utilizes constituent information derived by the AUTO bracketing system to constrain the training process. The bracketed training data not only provides essential information for the extended algorithm to consider only the set of rules whose spans are compatible with the *a priori* bracketings, but also establishes similar constituent structures in the inferred grammar.

A variety of experimental results have been presented which show that, although the AUTO bracketing system does make errors, these errors do not appear to significantly degrade the inference procedure compared to the use of manually bracketed data. The use of automatically generated implicit rules allows coverage to increase substantially. The introduction of bracketing constraints generated from the AUTO system significantly improves the measured performance metrics and maintains this increased coverage. Finally, unlike the standard inside-outside inference procedure, the system described here is computationally tractable when applied to realistic tasks.

Acknowledgement

The authors wish to thank Gareth Jones of CUED for helpful discussions on the experimental work described in this report and for critical comments on the manuscript.

References

[Baker, 1979] Baker, J. (1979). Trainable grammar for speech recognition. In *Speech Communication Papers for the 97th Meeting of the acoustical Society of America (D. Klatt and J. Wolf, eds)*, pages 547–550.

[Baum, 1972] Baum, L. (1972). An inequality and associated maximisation technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8.

[Black et al., 1991] Black, E., Abney, S., Flickinger, 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. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammar. In *DARPA Speech and Natural Language Workshop*, pages 306–311.

[Black et al., 1992] Black, E., Lafferty, J., and Roukos, S. (1992). Development and evaluation of a broad-coverage probabilistic grammar of English-language computer manuals. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 185–192.

[Briscoe et al., 1987] Briscoe, E., Grover, C., Bogurraev, B., and Carroll, J. (1987). A formalism and environment for the development of a large grammar of english. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 703–708, Milan, Italy.

[Briscoe and Waegner, 1992] Briscoe, E. and Waegner, N. (1992). Robust stochastic parsing using the inside-outside algorithm. In *AAAI Symposium on Statistic Applications to Natural Language*.

[Briscoe and Waegner, 1993] Briscoe, E. and Waegner, N. (1993). Undergeneration and robust paring. In Arts, J., Haan, P. D., and Oostdijk, N., editors, *English language corpora: design, analysis and exploitation*, pages 14–19. Rodopi, Amsterdam.

[Carroll and Charniak, 1992] Carroll, G. and Charniak, E. (1992). Learning probabilistic dependency grammars from labelled text. In *AAAI Fall Symposium Series: Probabilistic Approaches to Natural Language*, pages 25–32.

[Carroll et al., 1991] Carroll, J., Briscoe, E., and Grover, C. (1991). A development environment for large natural language grammars. Technical Report 233, Computer Laboratory, Cambridge University, England.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structure*. The Hague: Mouton and Co.

[Dodd, 1989] Dodd, L. (1989). The inside-outside algorithm: further experiments on three and four letter word spelling and implementation on an array processor. Technical Report Memo 4256, RSRE Malvern.

[Fu and Booth, 1986a] Fu, K. and Booth, T. (1986a). Grammatical inference: Introduction and survey - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(3):343–359.

[Fu and Booth, 1986b] Fu, K. and Booth, T. (1986b). Grammatical inference: Introduction and survey - part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(3):360–375.

[Gazdar et al., 1985] Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalised Phrase Structure Grammar*. Blackwell, Oxford, England.

[Gazdar and Mellish, 1989] Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in PROLOG*. Addison-Wesley.

[Grover et al., 1993] Grover, C., Carroll, J., and Briscoe, E. (1993). The Alvey natural language tools grammar. Technical Report 284, Computer Laboratory, Cambridge University, England.

[Jackendoff, 1977] Jackendoff, R. (1977). *$\bar{X}$-Syntax: A Study of Phrase Structure*. MIT Press.

[Jelinek, 1985] Jelinek, F. (1985). Markov source modelling of text generation. In *NATO Advanced Study Institute Impact of Processing Techniques on Communication*, pages 569–598.

[Kupiec, 1992] Kupiec, J. (1992). Hidden markov estimation for unrestricted stochastic context-free grammars. In *Proceedings of the ICASSP 92*, volume 1, pages 177–180.

[Lari and Young, 1990] Lari, K. and Young, S. (1990). The estimation of stochastic context-free grammars using inside-outside algorithm. *Computer Speech and Language*, pages 35–56.

[Lucas, 1993] Lucas, S. (1993). New directions in grammatical inference. In *Colloquium on Grammatical Inference: Theories, Applications and Alternatives*.

[Magerman and Marcus, 1990] Magerman, D. and Marcus, M. (1990). Parsing a natural language using mutual information statistics. In *DARPA Workshop on Spoken Language Systems*.

[Pereira and Schabes, 1992] Pereira, F. and Schabes, Y. (1992). Inside-Outside reestimation for partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135.

[Rulot et al., 1989] Rulot, H., Prieto, N., and Vidal, E. (1989). Learning accurate finite-state structural models of words through the ECGI algorithm. In *Proceedings of the ICASSP 89*, volume 2, pages 643–646.

[Santorini, 1991] Santorini, B. (1991). Bracketing guidelines for the Penn treebank project (draft version). Technical report, Univ. of Pennsylvania, Philadelphia.

[Schabes, 1992] Schabes, Y. (1992). An inside-outside algorithm for estimating the parameters of a hidden stochastic context-free grammar based on earley's algorithm. Technical report, Dept. of Computer and Information Science, Univ. of Pennsylvania, Philadelphia.

[Thompson, 1992] Thompson, H. (1992). Parseval workshop. In *ELSNews Vol.1(2)*.

[Waegner, 1993] Waegner, N. (1993). *Stochastic Models for Language Acquisition*. PhD thesis, Cambridge University, England.

[Wyard, 1993] Wyard, P. (1993). Context free grammar induction using genetic algorithm. In *Colloquium on Grammatical Inference: Theories, Applications and Alternatives*.