ICAME NEWS

Newsletter of the International Computer Archive of Modern English (ICAME)

Published by: The Norwegian Computing Centre for the Humanities, Bergen
The Norwegian Research Council for Sience and the Humanities

NAVF

Machine-readable texts in English language research

No.7

May 1983

CONTENTS

Editor: Dr. Stig Johansson, Department of English,
University of Oslo, Norway.

SEMINAR ON THE USE OF COMPUTERS IN ENGLISH LANGUAGE RESEARCH

September, 13-15 1982, University of Stockholm, Sweden

This seminar - the third in the series starting with the Bergen seminars in 1979 and 1981 - was organised by Magnus Ljung, Department of English, University of Stockholm. Papers were read by academics from a number of different research teams working with corpora of machine-readable texts. The papers are listed below, with a short outline of the main points raised (based on a report by Eric Atwell, University of Lancaster). The two contributions by the Lancaster group (7 and 10) are reproduced in full in this issue of *ICAME News*. Stig Johansson's paper (8) is to appear in *Computers and the Humanities*. There are also plans to publish papers from the seminar in an issue of *Stockholm Papers in English Language and Literature*. The participants agreed to meet again May 30 - June 1, 1983, in Nijmegen. For information on the forthcoming seminar, contact Jan Aarts, English Department, University of Nijmegen, Erasmusplein 1, 6500 HD Nijmegen, Holland.

(1) Henry Kučera, Brown University
"Comments on the Brown Corpus"

HK talked about his forthcoming book (co-authored with W. Nelson Francis), containing lemmatized word frequency lists, and many interesting new statistics gleaned from the grammatically tagged Brown Corpus. For example, 'informative' prose has many more words per sentence, on average, than fictional texts, *but* we find roughly the same number of predicates per sentence (so 'informative' prose sentences are made very long by complex noun phrases). HK also discussed his recent work in the field of Word Processing. Many WP packages currently available include a 'spelling-check' feature which compares each word in a text with a Lexicon of valid words, and flags words not found as probable misspellings; HK has been developing an algorithm which suggests *corrections*, that is, given an 'invalid' word, it finds the nearest match in the lexicon. His

program will also be able to detect certain spelling-errors which current WP programs miss: certain 'valid' words are flagged if they are incompatible with their syntactic contexts (e.g. *my* in *I my consider*).

(2) Antoinette Renouf, University of Birmingham
"Aspects of the Building of a Large Computer-Held Corpus of Text"

AR reported on the collection of English texts at the University of Birmingham, and the computing facilities they have available during its compilation. The Birmingham Corpus consists mainly of written texts, but also includes about one and a half million words of spoken English. The current size is over ten million words, and it is still growing! However, in 1981, a Subcorpus of six million words (1.5M spoken, 4.5M written) was separated out, to act as a static 'Interim' Corpus for the production of concordances, etc. These texts are mainly British English (c 70%), and the remainder is mostly American English. There is no poetry, nor any 'technical' language; and the style is predominantly 'neutral' in formality. Most of the genres found in the LOB and Brown Corpora are represented in the Birmingham Corpus (except Category J), although no rigorous sampling technique has been used in selecting the texts to be added; the main criterion is simply that 'texts are chosen to add to the variety and completeness of the Corpus'. Unlike the LOB and Brown Corpora, which represent the English of a fixed period in time (1961), the Birmingham Corpus is being continually updated as more and more texts are added; almost all the texts were written after 1960, and most belong to the late 70s and later.

Most texts were added to the Corpus using a Kurtzweil Optical Scanner which could 'read' books directly; however, the performance was erratic. Another source was 'dump tapes' from computer typesetting of newspapers, etc. Concordancing was another big problem; they used the COCOA package, and hired the ICL 1900 Mainframe for exclusive use for a couple of weekends!

(3) John Sinclair, University of Birmingham
"The Implications for Linguistic Research of Access to a Large Computer-Held Corpus of Text"

In this talk, JS explained why he was building an 'open-ended' Corpus, with few restrictions on size or subject-matter (in contrast to the LOB and Brown Corpora). He argued that linguists still have no complete or adequate theory of language; and any restrictions in the design of a Corpus might tend to bias us towards one particular theory. Work has begun on analysing collocation lists, but at Birmingham they are not interested in merely collecting statistics: they also want to know *why* certain patterns occur in language. For example, many words (e.g. *matter*) could theoretically be used as a verb or a noun, but it turns out that in actual usage one of these two often predominates to a marked degree (e.g. of 1160 occurrences of *matter*, only 173 are verbs (and of these, all but 3 are negatives, mainly *it doesn't matter*)). At Birmingham JS is trying to explain why such things happen in usage; but at the moment, there is no theoretical framework to fit such explanations into. Instead, they run the risk of ending up with an open-ended list of unconnected 'theorems', such as "questions involving the word *matter* are usually rhetorical".

The selection of texts to be added to the Birmingham Corpus is *deliberately* free and fairly unrestricted, since, in a sense, they cannot be sure in advance just what they are looking for. JS ended by saying that linguists working on corpus-based research need to think carefully about the direction and purpose of their research; corpus-based linguistics at present lacks a clear general theory or set of guidelines.

(4) Jan Hultgren, University of Stockholm
"3RIP: An Interactive Text Data Base System"

3RIP (pronounced "thrip") is a system for storing and accessing large amounts of data, such as a corpus of English texts. A data base is subdivided into records, and each record has an extensive Reference Key to speed up and simplify sort and search operations. An application of 3RIP of particular interest here is BLOB, an implementation of the (untagged) Brown and LOB Corpora. In BLOB, each Category is treated as an independent data base (30 in all),

and each sentence is a separate record. The text is divided up into sentences rather than 'lines' as in the original Brown and LOB Corpora because this is more useful; e.g. in KWIC concordances, it is more useful to be given a complete sentence context.

3RIP has many commercial, academic and other applications. The largest 3RIP application at present comprises some 1.2M references (more than 1000 million characters of text) to scientific articles; this is available commercially in America. The smallest 3RIP 'Corpus' is a Shakespeare text of a few hundred records at Stockholm University.

(5) Benny Brodda, University of Stockholm
"The TAGGER: Presentation of a Tagging Program"

A technical problem in tagging programs is how to link tags with their corresponding syntactic units. In word-tagging, these units are the words, but in theory the term 'tagging' could be used to cover higher levels as well, with 'tags' assigned to larger constituents.

The TAGGER is an interactive semi-automatic tagging system which allows tags to be linked to words in a number of different ways. One method is to have two separate files, one containing words only, and the other containing tags only; corresponding tag and word have the same record reference number, and the same position within that record. Another method is to have only one file, with the tag immediately after the word (with a separator, /, between the two). A third format has a line of words with a line of tags immediately following (each tag is immediately below its corresponding word). These three alternatives each have different advantages; for example, the first format takes up less space in computer memory, but the third format is much more 'readable' to humans.

The TAGGER can take files in any format as input, and can produce files of any format as output. Initially, there will only be 'dummy' tags; the linguist must then go through the text interactively, making tagging decisions. Initially all 'dummy' tags start with a lower case letter; if any are 'accepted' or 'corrected' by the linguist, this is signalled by changing the initial letter to

upper case. The TAGGER also takes a Lexicon as input; this is a list of words and/or word-endings, each followed by one or more tags. The TAGGER uses this Lexicon to suggest tags to replace 'dummy' tags. However, unlike the LOB and Brown tagging systems, The TAGGER does not take syntactic context into account.

(6) Magnus Ljung, University of Stockholm
"SEMTAG: Semantic Tagging of the Brown and LOB Corpora"

ML talked about his project: the semantic tagging of a collection of texts from Categories A, G, J, and N (c 200,000 words in all). Unfortunately, unlike syntactic tagging, semantic tagging has no obvious tagset 'given'; there is no clear, closed set of 'semantic primitives'. Instead, ML used the classification system found in dictionaries. Each word was marked with its 'sense number' according to the Longman Dictionary available on magnetic tape. For example, *realized* in *They realized that he was drunk* might be REALIZE2. If a dictionary entry included two or more homonyms, then the homonym-number was given first; e.g. *centre* might be tagged CENTRE2S3 (meaning '3rd sense of 2nd homonym of *centre*' - note that S is just a separator).

Semantic tagging of this kind tends to involve much more subjective decision-making by linguists than the grammatical tagging of the Brown and LOB Corpora. One interesting fact that ML has found so far is that Longmans generally order the senses of a word correctly according to their frequency of occurrence: sense 1 is by far the most commonly used, followed by sense 2, and so on. In other words, the frequency statistics agree very well with the intuitions of Longman's lexicographers!

The tagging so far has been entirely 'manual', using The TAGGER described above. ML would like an automatic semantic tagging system analogous to the automatic grammatical tagging systems of Brown and LOB; but he has no idea yet just how this might work (since there are no obvious patterns in the semantic sense-numbers).

Unfortunately, ML had not been able to collate any statistics or draw any conclusions from the tagged subcorpus yet, since he has been preoccupied with the practical problems of actually getting the subcorpus tagged. However, the outlook seems promising.

(7) Geoffrey Leech, University of Lancaster
"Progress Report from the Lancaster Project (1)"

GL described the progress made so far in the project to grammatically tag the LOB Corpus. Each word in the Corpus is to be assigned a tag from a closed set of 134 (e.g. JJ=adjective, NN=singular noun, etc.). Unlike ML's Semantic Tagging, in the Lancaster Tagging project nearly all tagging decisions are made automatically, by a suite of programs. GL described the successive steps in the grammatical tagging process, and talked about the programs in the Tagging Suite.

The first step in the tagging procedure is Manual Pre-editing: various classes of words which would prove to be problematic for the Tagging Suite are dealt with 'by hand', that is, decisions are made by a (human) linguist. In fact, very few words actually require any 'manual pre-editing': the biggest problem was words beginning with a capital letter (the first word of a sentence must generally be converted to lower case).

Next, the text is 'verticalized': a program writes each word onto a separate line of its own. Then, a program called WORDTAG adds a list of one or more possible tags to each word (e.g. "go VB/NN", i.e. the word *go* can be a verb or a noun). WORDTAG adds this list of possible tags to a word without looking at the context. Occasionally, however, this is necessary: certain idiomatic sequences of words require 'syntactically uncharacteristic' tags (e.g. *one* is generally tagged CD1 (CarDinal 1), and *another* is generally tagged DT (singular DeTerminer), but the idiomatic combination *one another* is a reflexive personal pronoun (PPLS). The next program, IDIOMTAG, deals with such special cases. The final stage in the automatic analysis is CHAINPROBS; this looks at the set of possible tags with each word, and works out the 'relative likelihood' of each tag, by looking at the 'immediate syntactic context'.

After the Tagging Suite has run, a Manual Postediting stage is necessary to check the decisions made by the programs. In c 97% of words, the tag chosen as 'likeliest' by CHAINPROBS is actually the correct one; manual correction is needed in the remaining c 3%. (See further the paper by Geoffrey Leech *et al.* in this issue of *ICAME News*.)

(8) Stig Johansson, University of Oslo
"Word Frequency and Text Type - Some Observations Based on the LOB Corpus of British English Texts"

The LOB Corpus is subdivided into Categories, according to the type of text; SJ has been studying statistical differences in vocabulary between the different genres. The most striking contrasts were revealed when comparing the vocabulary of Category J (learned and scientific English) with that of Categories K-R (fiction); the other texts in the LOB Corpus were grouped under the broad headings 'newspapers' (Categories A-C), and 'miscellaneous informative prose' (Categories D-H).

SJ looked at the most frequent words in each of these four genres. A 'distinctiveness coefficient' can be calculated, indicating whether a word is unusually frequent in one particular genre; for example, words like *thus*, *therefore*, etc. are much more frequent in J than in K-R. This fact is not really so surprising; but SJ found various other statistics which are not so obvious, for example: J places more importance on NOUNS, whereas VERBS are more important in K-R; fiction texts are more 'singular prone' (i.e. the ratio of singular to plural nouns is significantly higher); personal pronouns are most common in fiction, except *we*; possessive and reflexive personal pronouns are also much commoner in K-R than in J, except *its*, *our*, *their*, *itself*, and *themselves*; and articles *the* and *an* are significantly more frequent in J, but article *a* is significantly more frequent in K-R!

(9) Knut Hofland, Norwegian Computing Centre for the Humanities
"Implementation of the Lancaster Tagging Programs on a Non-Corpus Text"

The LOB Corpus Grammatical Tagging Program Suite described by GL (7) was designed specifically to tag the texts of the LOB Corpus. Even in its 'raw' state before the automatic analysis, the Corpus contains various markers not found in ordinary texts (e.g. the start of each sentence is marked with a special symbol; headings and headlines are marked; etc.). Therefore, KH decided it would be interesting to see how well the LOB Tagging Suite could deal with something completely

different. The text he chose was the play *Look Back In Anger* by Osborne; the LOB Corpus contains no plays, so this would be an added difference.

KH first described various technical changes, needed to run the programs on a UNIVAC computer instead of an ICL machine. To circumvent the Manual Pre-editing stage (see (7)), KH wrote a program to deal with word-initial capitals automatically; this resulted in a number of errors (e.g. *I* was converted to lower case at the start of a sentence), but saved time.

The test run was very successful; c 90% of words were tagged correctly by the LOB Tagging programs in their original form. Minor changes could improve this success rate; clearly, KH had shown that the LOB Tagging Program Suite can be used to grammatically tag *any* English text with a high degree of accuracy.

(10) Eric Atwell, University of Lancaster
"Progress Report from the Lancaster Project (2)"

GL had briefly outlined the programs of the LOB Tagging Suite; EA explained in greater detail how the 'relative likelihood' of a tag is calculated. This is done using a general formula, which takes into account the word that the tag is assigned to, and the 'immediate syntactic context': the tag immediately before and the tag immediately after the current tag. If either of these tags are 'ambiguous', then two different formulae are used to calculate 'relative likelihood'. In almost all cases, both formulae agree as to which tag is likeliest with a given word.

This method of probabilistic analysis is new to linguistics; EA went on to mention some of its other potential applications. One of these is in Automatic Spelling-checking. Many Word Processors include a facility to check the spellings of words in a text, but these programs simply look up each word in a dictionary: syntactic context is not taken into account at all. This means that many glaring errors can still slip through; e.g. in *I am able to prophecy the whether*, no errors are detected by simple dictionary-lookup methods because *prophecy* and *whether* are valid words. However, syntactic analysis can show that these words are incompatible with their syntactic

context. EA has altered the programs in the LOB Tagging Suite to carry out such 'spelling-check syntactic analysis' on a small sample text. Further research is under way to create a generally-applicable WP syntax-analyser and -checker. (See further Eric Atwell's article in this issue of *ICAME News*.)

(11) Tore Jansson, University of Stockholm
"Simulation of Phonotactic Word Division in Perception"

The 'Corpus' TJ is working on is in fact just five lines of phonetic transcription. This test 'Corpus' is input to a program, which first inserts all *potential* word-boundaries: for example, the phrase *see his point* will be represented as "s-i:-h-I-z-p-oI-n-t". The program then uses a set of rules to try to divide this continuous sequence into separate words. One type of rule *replaces* a hyphen with a blank, if a word boundary is required; for example, one rule states that "h" should begin a word, so the hyphen before "h" must be replaced with a space. The second type of rule *deletes* a hyphen where a word boundary *cannot* occur; for example, there are rules which state, in essence, that each word must have at least one vowel, so the hyphen in "s-i: h..." is deleted. In the final output, the text should be correctly divided into words, with no hyphens remaining (i.e. "si: hIz poInt"). This is not always the case, as often some hyphens remain, denoting an 'ambiguity'; this type of ambiguity may be resolvable by syntactic analysis.

(12) Jan Svartvik and Gunnel Tottie, Universities of Lund and Uppsala
"English in Speech and Writing: Presentation of a Project"

"English in Speech and Writing" is a research project in progress at the Universities of Lund and Uppsala, based on two corpora of English texts: the LOB Corpus of written British English and the London-Lund Corpus of spoken British English (LLC). JS talked about how the LLC was compiled. Originally, spoken material was collected and transcribed at University College London by researchers working on the Survey of English Usage. The Survey of Spoken English, with which JS is principally concerned, is a sister project of the Survey of English Usage; its primary aim was to convert the spoken

material into machine-readable form, for use in computational analysis. JS reported that this has now been completed, and that the London-Lund Corpus is available for research, the whole corpus on magnetic tape, and about one third of it, also in printed form, as a book. The main points raised by JS concerned the grammatical analysis of the London-Lund Corpus, which includes both word-class tagging and higher-level syntactic tagging (in which the basic unit of analysis is not the sentence but the tone unit).

GT reported in greater detail on the English in Speech and Writing project. A 'Minicorpus' of 10,000 words and a 'Midicorpus' of 100,000 words (available on tape) have been compiled. Both contain a mixture of texts from the LLC and the LOB Corpus. The distinction between written and spoken texts is strengthened by the exclusion of LOB fiction texts (which contain dialogue). Some major areas which are currently being investigated in the project are modality, negation, and cohesion in speech and writing.

(13) Mats Eeg-Olofsson, University of Lund
"An Experiment in Statistical Tagging"

Researchers in Lund have 'word-tagged' part of the London-Lund Corpus 'by hand', that is, with a linguist making all tagging decisions interactively. ME used these tagged texts as the basis for an experimental statistical model for tagging. He designed an experimental automatic tagging program which was *integrated* (it can take into account syntactic, morphological, and other levels of information), *probabilistic* (rather than choosing one analysis to the exclusion of all other possibilities, it *orders* all analyses according to relative likelihood), *holistic* (words are not tagged independently, but according to context), and *formal* (testable by standard statistical methods).

In designing his experimental program, ME first observed the patternings that occurred in the tagged texts. The aim was to apply Bayes' formula, and write a probability function which 'encapsulated' the tag-patternings found. To write this probability function, ME first had to make some initial assumptions. He assumed that tag-pair

collocation was an important indicator of patternings, and that individual tag-pattern pairings were independent 'events', so that the conditional independence assumption formula could be applied (p = pattern, t = tag):

$$P(p[1]...p[n] \,/\, t[1]...t[n]) = \pi_{i=1..n} P(p[i] \,/\, t[i])$$

In other words, ME assumed that the tags on either side were a major factor in working out the likelihood of the 'target' tag. Another factor to be taken into account was the overall relative likelihood of the 'target' tag appearing with the particular word being tagged.

The tag-set used was analogous to the one used in the LOB tagging project, with about 100 tags denoting 'traditional' surface syntactic word classes. ME noted that the 'graphic word' is not the only pattern indicative of tagging; suffixes (e.g. "...-ly") or phrases (e.g. "as well") may also be important patterns. ME's tagging program could be described as equivalent to a Markov chain process or a probabilistic Finite State grammar.

To run the experimental program, ME first had to extract a Tag-Pair Frequency Table from a pre-tagged text. Interestingly, he found that most possible tag-pair combinations never actually occurred in the text: out of a theoretically possible c 10,000 tag-pair combinations, only 1487 actually occurred in the 5000-word text. ME also needed a tagged wordlist, stating which tags could occur with each word; he extracted a wordlist from the 5000-word text for this purpose.

The trial run of his program, on the text from which the tag-pair statistics and tagged wordlist had been extracted, was very impressive: the tag chosen as 'likeliest' was correct with 99% of words! However, we must remember that this was partly because the tag-pair table and wordlist were 'tailor-made' for the particular text. In particular, the wordlist gave many words only one unambiguous tag simply because it only ever appeared with that tag in the text: for example, if the word *bother* only appeared as a noun in the manually-tagged text, then it was unambiguously tagged as a noun in the wordlist.

To get a more 'realistic' success-rate figure, ME used the same tag-pair statistics table and tagged wordlist to tag a *different* 5000-word text. This time, the success rate was c 84%. Most of the

errors were due to omissions in the simple wordlist.

Overall, ME's tagger was very similar to the LOB Tagging Suite, in principle at least. However, the type of text it had to deal with was very different from that found in the LOB Corpus: the spoken English texts contained no punctuation (although they were divided up into *tone units*), and they were full of pauses, indistinct passages, phonetic transcriptions of 'unclear' words, hesitations, etc. It is interesting that basically the same method of analysis can be used for both written and spoken English.

THE AUTOMATIC GRAMMATICAL TAGGING OF THE LOB CORPUS

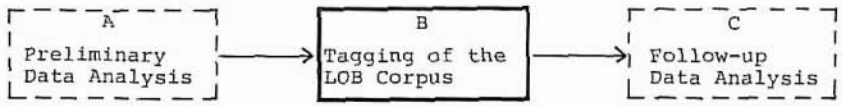
Geoffrey Leech, *Roger Garside*, and *Eric Atwell*
University of Lancaster, England

In collaboration with the English Department, University of Oslo,[1] and the Norwegian Computing Centre for the Humanities, Bergen,[2] we have been engaged in the automatic grammatical tagging of the LOB (Lancaster-Oslo/Bergen) Corpus of British English. The computer programs for this task are running at a success rate of approximately 96.7%,[3] and a substantial part of the 1,000,000-word corpus has already been tagged.[4] The purpose of this paper is to give an account of the project, with special reference to the methods of tagging we have adopted.

1 OVERVIEW OF THE PROJECT

To see the project in its overall context, we must give some attention to the preliminaries which preceded the tagging itself, and also to the follow-up work which we intend to undertake when the tagging is complete:

Fig. 1



1.1 Preliminaries

The first stage of our work was collecting and analysing data from the Tagged Brown Corpus. Our purpose was to make use of, and at the same time to improve on, the automatic tagging of the Brown Corpus (undertaken at Brown University 1971-8).[5] The Tagged Brown Corpus was kindly made available to us by Henry Kučera and Nelson Francis, who also provided us with a copy of the automatic tagging program TAGGIT written by Greene and Rubin (1971). An exploratory run of the program

on the LOB Corpus suggested that a new approach to tag selection would be needed if we were to improve substantially on TAGGIT's performance. For comparability with the Tagged Brown Corpus, we had decided to use largely the same set of tags as were used by TAGGIT; but in practice some changes were advisable, and as a result of these changes, the new Tagset (see Appendix A) consisted of 134 tags (including punctuation tags), as against Brown's 87. For example, we found it desirable to introduce a number of additional tags ("NPL", "NPT", "NNP", "JNP") where Brown had used only the one tag "NP" (proper noun). But where changes were made, we have been careful to preserve general comparability with the Brown Corpus, so that when the LOB tagging is complete, it will be possible to make systematic comparisons between the American and British corpora.

The chief advantage we derived from the Brown tagging project, however, was that we were able to make substantial use of the Tagged Brown Corpus itself as a database for our own Automatic Tagging. From lists provided by the Norwegian Computing Centre for the Humanities, our Oslo colleagues Stig Johansson and Mette-Cathrine Jahr derived lists of word-tag associations and suffix-tag associations which, after revision, formed the kernel of our Tag-Assignment program (see 3.1 below). Also, by means of a group of Context Collecting programs, we were able to derive from the corpus frequency lists of tag-sequences, and these were later adapted for inclusion in our Tag-Selection program (see 3.2).

1.2 Follow-up work

Just as the tagging of the Brown Corpus provided us with a headstart in our own project, so after the tagging of the LOB Corpus it will be possible to use the data derived from the LOB tagging project, including the tagged Corpus itself, as an input to further automatic tagging programs, which will improve on our programs just as these were an improvement on the Brown programs. Corpus-based automatic language analysis is one area of linguistic research where results are cumulative, so we hope, in a follow-up to this project, to revise and improve the programs for implementation on further corpora. For this to happen, however, various frequency listings must be obtained from the Tagged LOB Corpus. Such listings (in particular, a lemmatised word-frequency listing of the LOB Corpus) will also be useful

for other research purposes, e.g. for comparison with the Brown Corpus and with the London-Lund Corpus.

2 THE OVERALL PROCESS OF TAGGING

Having looked briefly at stages (A) and (C) in Fig. 1, we may now examine the middle box (B), dealing with the overall tagging process. The contents of this box we again divide into three stages:

Fig. 2

As may be expected with programs acting on unrestricted language input, the automatic tagging programs require both a pre-editing phase, where the human investigator prepares the corpus for input, and a post-editing phase, where he corrects any errors made by automatic tagging. Manual pre-editing and post-editing are both, however, carried out with the aid of computer programs. We give a brief account of these stages (A and C in Fig. 2) before dealing with the automatic tagging programs themselves.

2.1 Pre-editing

At the start of the process, the Raw Corpus (the Corpus in its untagged orthographic form) exists in a "horizontal" format; i.e. it reads from left to right in the normal way. A Verticalization Program converts this corpus into a "Vertical Corpus" in which one word occurs beneath another in a vertical column. At the same time, the Verticalization Program makes automatic changes which will later help the tagging. These include supplying missing punctuation, splitting enclitic words (*n't*, *'ll*, etc.) from their predecessors, changing capital letters to lower case at the beginning of sentences, in headings, etc.; and marking foreign words, formulae, and other exceptional features of the text. The Verticalization Program also creates a number of columns alongside the text, so that various kinds of information (orthographic, lexical, syntactic) can be recorded for future users of the corpus.

When the Verticalization of the corpus takes place, another set of programs produces "Editlists" of particular text features which have to be checked by a human editor to see whether they have to be altered in order to be suitable input to the Automatic Tagging. The most important lists are those of "CAPITALS" (non-sentence-initial words beginning with a capital letter) and "UNCAPITALS" (sentence-initial words whose capital letter will have been changed to lower case by the Verticalization Program). For example, if a sentence begins with a proper name such as *John*, the Program will have changed this to *john*, and a manual editor will then have to change it back again. The reason for these changes in capitalization is that the Automatic Tagging programs make use of word-initial capitals in deciding what kind of tags to assign to a word (most words beginning with a capital end up being tagged as proper names: see 3.1 and Appendix D).

Although the majority of pre-editing changes are made automatically by the Verticalization Program, Pre-editing has proved to be a time-consuming process, especially since all pre-editing decisions have had to be carefully standardized and entered in a "Pre-editing Manual". In any further tagging projects, we will try to eliminate manual pre-editing, e.g. by enabling the automatic tagging programs to accept a word with an initial capital as a possible variant of a lower case word. For example, if both *Rose* and *rose* occurred in the same text, the capital of the former word would be reduced to lower case; but if *Rose* only occurred in the capitalized version, the capital would be retained, and the word would be analysed as a proper noun. In this way, manual pre-editing could be replaced by automatic pre-editing, and any additional errors which resulted from this would simply add to the number of words requiring correction at the post-editing phase.[6]

2.2 Post-editing

Like Pre-editing, Post-editing currently has both an automatic and a manual aspect. The Vertical Corpus, after automatic tagging, contains, alongside each word, one or more grammatical tags, placed in order of their likelihood of occurring in this context. The tag which the programs have selected as the correct one is clearly indicated (see Fig. 4 below). Thus the task of the manual post-editor is to check the decisions made by the program, and to mark any corrections which

have to be made. With more than a million words to check, this is an exceedingly time-consuming task, and it is therefore worthwhile using the computer to ease the human editor's task in any practicable way. One way of doing this is to present the output in a special form in which the text is arranged in two vertical columns per page, the word and the tag lying alongside one another for ease of reading. Into this "Vertical Output" there is built an additional aid for the post-editor: it is possible to set a threshold below which the likelihood of error is low enough to be disregarded by the initial post-editor. Sample analyses have shown that 60% of the text-words are unambiguously tagged; that of the 40% which are ambiguously tagged, 64% have a likelihood, as calculated by the Tag Selection Program (see 3.2)[7], of more than 90%; and that these have only a 0.5% risk of being erroneous. This means that over the whole sample 86% of words can be unambiguously tagged with less than 1% error. In these relatively safe cases, the output listing simply assumes the one tag to be correct, and gives alternative taggings only for the 14% of words for which the risk of error is relatively high. A specimen of this "Vertical Output" is given in Appendix E.

This facilitates the first manual post-edit, but to ensure that all errors have been caught, a second stage of manual post-editing will take place, this time on a "rehorizontalized" version of the corpus, in which each word in a line has a single tag beneath it, as in Appendix F.

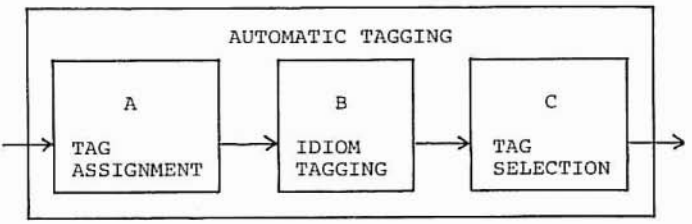
Once it has undergone manual correction, this version of the corpus will be available for distribution to users. There will also, however, be a vertical-format "Rolls-Royce" version of the corpus, which will contain all the information about the original text recorded in the columns of the Verticalization Program (see 2.1) as well as the grammatical tag of each word. This version is the authoritative tagged LOB Corpus, and will enable users to reconstruct the original text. For example, if one wants to study the relation between orthography and grammar, this version will preserve orthographic information excluded from the "rehorizontalized" version.

3 AUTOMATIC TAGGING

We now turn to the Automatic Tagging programs which form the heart of the project, and constitute its main contribution to research.

Once again, the contents of the middle box of the previous diagram (B in Fig. 2) must itself be broken down into three logically separable processes:

Fig. 3



For development purposes, it was convenient to write a separate program for each of these three processes;[8] but it would be easy enough in principle to combine them all into a single program. Logically speaking, the Automatic Tagging divides into Tag Assignment (whereby each word in the corpus is assigned one or more possible tags), and Tag Selection (whereby a single tag is selected as the correct one in context, from the one or more alternatives generated by Tag Assignment). It was as something of an afterthought that we added to the Tag Assignment program (WORDTAG) and the Tag Selection program (CHAINPROBS) a third, intermediate program (IDIOMTAG) to deal with various grammatically anomalous word-sequences which, without intending any technical usage of the term, we may call "idioms".

3.1 Tag Assignment

The simplest kind of Tag Assignment procedure would be just a look-up in a WORDLIST or dictionary specifying the tag(s) associated with each word. In addition to such a Wordlist, the Brown Tagging Program TAGGIT has a SUFFIXLIST, or list of pairings of word-endings and tags (for example, the ending -NESS is associated with nouns). We follow Brown in this, using a Wordlist of over 7000 words, and a Suffixlist of approximately 660 word-endings.[9] Further, the LOB Assignment Program contains a number of procedures for dealing with words containing hyphens, words beginning with a capital letter, words ending with *-s*, with *'s*, etc. The advantages of having a SUFFIXLIST are that (a) the WORDLIST can be shortened, since words whose wordclass is predictable from their ending can be omitted from it; and (b) the

set of words accepted by the program can be open-ended, and can even include neologisms, rare words, nonsense words, etc. These advantages also apply to the procedures for dealing with hyphenated and capitalized words.

The Tag Assignment Program reads each word in turn, and carries out a series of testing procedures, to decide how the word should be tagged. The procedures are crucially ordered, so that if one procedure fails to tag a word, the word drops through to the next procedure. If none of the tag-assignment procedures is successful, the word is given a set of default tags. The program's structure can be summarized at its simplest by listing the major procedures as follows (where W = the word currently being tagged):

(1) *Is W in the WORDLIST?*

If so, assign the tags given in the WORDLIST.

(2) *Is W a number, a single letter, or a letter preceded or followed by a number of digits?*

If so, assign special tags.

(3) *Does W contain a hyphen?*

If so, carry out the special procedure APPLYHYPHEN.

(4) *Does W have a word-initial capital (WIC)?*

If so, carry out the special procedure APPLYWIC.

(5) *Does W end with one of the endings in the SUFFIXLIST?*

If so, assign the tags specified in the SUFFIXLIST.

(6) *Does W end in -s?*

If so, apply an *-s* stripping procedure, and check again whether W is in the WORDLIST, or failing that, the SUFFIXLIST. If it is, apply the tags given in the WORDLIST or SUFFIXLIST, retaining only those tags which are compatible with *-s*.

If not, assign default tags for words ending in *-s*.

(7) *If none of the above apply*, assign default tags for words not ending in *-s*.

APPLYHYPHEN and APPLYWIC are 'macroprocedures' which themselves consist of a set of tests comparable to those of the main program. For further details, see the Flowcharts in Appendices B - D.

The output of the Tag Selection Program is a version of the Vertical Corpus in which one or more grammatical tags (with accompanying rarity markers @ or % if appropriate)[10] are entered alongside each word. As an additional useful feature, this program provides a diagnostic (in the form of an integer between 0 and 100) indicating the tagging decision which led to the tag-assignment of each word. This enables the efficacy of each procedure in the program to be monitored, so that any improvement effected by changes in the program can be measured and analysed. In this respect, the program is self-evaluating. It can also be readily updated through revisions to the Tag-set, Wordlist, or Suffixlist.

3.2 Tag Selection

If one part of the project can be said to have made a particular contribution to automatic language processing, it is the Tag Selection Program (CHAINPROBS), the structure of which is described in greater detail in Marshall (1982). This program operates on a principle quite different from that of the Tag Selection part of the program used on the Brown Corpus. The Brown program used a set of CONTEXT FRAME RULES, which eliminated tags on the current word if they were incompatible with tags on the words within a span of two to the left or two to the right of the current word (W). Thus assuming a sequence of words -2, -1, W, +1, +2, an attempt was made to disambiguate W on the evidence of tags already unambiguously assigned to words -2, -1, +1, or +2. The rules worked only if one or more of these words were unambiguously tagged, and consequently often failed on sequences of ambiguous words. Moreover, as many as 80% of the applications of the Context Frame Rules made use of only one word to the left or to the right of W. These observations, made by running the Brown Program over part of the LOB Corpus, led us to develop, as a prototype of the LOB Tag-Selection Program, a program which computes transitional probabilities between one tag and the next for all combinations or possible tags, and chooses the most likely path through a set of ambiguous tags on this basis.

Given a sequence of ambiguous tags, the prototype Tag-Selection Program computed all possible combinations of tag-sequences (i.e. all possible paths), building up a search tree. It treated each possible Tag Sequence or path as a First-order Markov chain, assigning to each

path a probability relative to other paths, and reducing by a constant scaling factor the likelihood of sequences containing tags marked with a rarity marker @ or %. Our assumption was that the frequency of tag sequences in the Tagged Brown Corpus would be a good guide to the probability of such sequences in the LOB Corpus; these frequencies were therefore extracted from the Brown Corpus data, and adjusted to take account of changes we had made to the Brown Tag-set. We expected that the choice of tags on the basis of first-order probabilities would provide a rough-and-ready tag-selection procedure which would then have to be refined to take account of higher-order probabilities. It is generally assumed, following Chomsky (1957:18-25), that a first-order Markov process is an inadequate model of human language. We therefore found it encouraging that the success rate of this simple first-order probabilistic algorithm, when tried out on a sample of over 15,000 words of the LOB Corpus, was as high as 94%. An example of the output of this program (from Marshall 1982) is given in Fig. 4:

Fig. 4

this	DT
task	NN
involved	[VBD]/90 VBN/10 JJ@/0
a	AT
very	[QL]/99 JJB@/1
great	[JJ]/98 RB/2
deal	[NN]/99 VB/1
of	IN
detailed	[JJ]/98 VBN/2 VBD/0
work	[NN]/100 VB/0
for	[IN]/97 CS/3
the	ATI
committee	NN

In this output, the tags supplied by the Tag Assignment Program are accompanied by a probability expressed as a percentage. For example, the entry for the word *involved* ([VBD]/90 VBN/10 JJ@/0) indicates that the tag VBD 'past tense verb' has an estimated probability of 90%; that the tag VBN 'past participle' has an estimated probability of 10%; and that the tag JJ 'adjective' has an estimated probability

of 0%. The symbol @ after JJ means that the Tag Assignment program has already marked the 'adjective' tag as rare for this word (see Note 10). The square brackets enclosing the 'past tense' tag indicate that this tag has been selected as correct by the Tag Selection Program. (The square brackets are used to indicate the preferred tag for every word which is marked as ambiguous; where the word has only one assigned tag, this marking is omitted as unnecessary.)

An improved Tag Selection Program was developed as a result of an analysis of the errors made by the prototype program. We realised that an attempt to supplement the first-order transition matrix by a second-order matrix would lead to a vast increase in the amount of data to be handled as part of the program, with only a marginal increase in the program's success. A more practical approach would be to concentrate on those limited areas where failure to take account of longer sequences resulted in errors, and to introduce a scaling factor to adjust such sequences in the direction of the required result. For instance, the occurrence of an adverb between two verb forms (as in *has recently visited*) often led to the mistaken selection of VBD rather than VBN for the second verb, and this mistake could be corrected by downgrading the likelihood of a triple consisting of the verb *be* or *have* followed by an adverb followed by a past tense verb. Similarly, many errors resulted from sequences such as *live and work*, where we would expect the same word-class to occur on either side of the coordinator - something which an algorithm using frequency of tag-pairs alone could not predict. This again could be handled by boosting or reducing the predicted likelihood of certain tag triples. A further useful addition to the program was an alternative method of calculating relative likelihood, making use of the probability of a word's belonging to a particular grammatical class, rather than the probability of the occurrence of a whole sequence of tags. This serves as a cross-check on the 'sequence probability' method, and appears to be more accurate for some classes of cases. These improvements, together with the introduction of an Idiom Tagging program (see 3.3 below), resulted in an overall success rate of between 96.5% and 97.0%.

Having tried out the heuristic principle that error-analysis of a program's output can be fed back into the program, enabling it to increase its accuracy, we anticipate that a further analysis of errors after post-editing of the LOB Corpus will lead to further improvements.

3.3 Idiom Tagging

The third tagging program, which intervenes between the Tag Assignment and Tag Selection programs, is an Idiom Tagging Program (IDIOMTAG) developed as a means of dealing with idiosyncratic word sequences, which would otherwise cause difficulty for the automatic tagging. One set of anomalous cases consists of sequences which are best treated, grammatically, as a single word: for example, *in order that* is tagged as a single conjunction, *as to* as a single preposition, and *each other* as a single pronoun. Another group consists of sequences in which a given word-type is associated with a neighbouring grammatical category; for example, preceding the preposition *by*, a word like *invoked* is usually a past participle rather than a past tense verb. The Idiom Tagging Program is flexible in the sorts of sequence it can recognize, and in the sorts of operation it can perform: it can look either at the tags associated with a word, or at the word itself; it can look at any combination of words and tags, with or without intervening words. It can delete tags, add tags, or change the probability of tags. It uses an Idiom Dictionary to which new entries may be added as they arise in the corpus. In theory, the program can handle any number of idiomatic sequences, and thereby anticipate likely mistaggings by the Tag Selection Program; in practice, in the present project, we are using it in a rather limited way, to deal with a few areas of difficulty. Although this program might seem to be an *ad hoc* device, it is worth bearing in mind that any fully automatic language analysis system has to come to terms with problems of lexical idiosyncrasy.

4 FUTURE PROSPECTS

Our present overriding objective (in cooperation with our collaborators in Norway) is to complete the grammatical tagging of the LOB Corpus by the summer of 1983, and to make it available for research, through the Norwegian Computing Centre for the Humanities. We hope that its value as a research facility will more than justify the research which has led to the development of the Automatic Tagging programs. But in addition, we believe that the considerable success of these programs has helped to vindicate the value of corpus-based research in the automatic analysis of texts. The strength of computa-

tional corpus-based research is that the programs have to be designed to operate on unrestricted input, and can be progressively enhanced by the 'recycling' of data already analysed into the database.

If resources are available for future research, we hope to eliminate manual pre-editing, and to reduce further the percentage of error to be corrected in post-editing. One method for reducing error would be to derive different tag-pair frequencies from different kinds of text, and to use these in a 'fine-tuning' of the transition matrix for various styles of input text. For example, the frequencies for scientific and for fictional writing can be supposed to differ considerably, and statistical adjustments of the program to deal with these differences can be expected to eliminate additional errors. Even so, there will still be errors which cannot be corrected by enhancement of the present programs. Like Kučera and Francis (see Francis 1980), we have found special problems with certain classes of ambiguity, where the choice of wordclass requires reference to a wide context. Three difficult ambiguities are:

(i) that between IN and CS (e.g. *after* can be a preposition or a conjunction);

(ii) that between IN and RP or RI (e.g. *in* can be a preposition or a prepositional adverb); and

(iii) that between VBD and VBN (e.g. *acquired* can be a past tense verb or a past participle).

The following example shows the sort of problem which arises with the last case:

> ... some local authorities ... *have* not only *carried* out a very good business deal for themselves but also *acquired* a beauty spot for their people.

It is notable that if the word *have* were omitted from this sentence, the word *acquired*, which is the fourteenth word following it, would be changed from a VBN to VBD. This is because *carried*, which by virtue of the coordinate construction must be matched by *acquired*, would no longer be marked as the second verb of a perfective (*have* + past participle) construction. In other words, for this disambiguation a span of 14 words to the left of the target word is needed.

Such difficulties inevitably lead us to consider the deficiencies of word-tagging as an autonomous level of analysis. The most obviously

valuable levels of analysis to be added to word-tagging would be (a) syntactic analysis or parsing of a corpus; and (b) semantic tagging, whereby senses of words, as well as their grammatical categories, would be identified. These additional levels, on which work with the LOB Corpus has only recently begun,[11] would have to be added to the LOB Automatic Tagging programs if success in word-tagging were to approach 100%. The VBD/VBN ambiguity cited above, for example, could be successfully resolved only by a program which carried out recognition and tagging of larger-than-word units. There are strong reasons, indeed, for believing that the tagging programs will only reach their full potential when they are implemented in parallel with syntactic and (possibly) semantic analysis programs. These further challenges will remain when the present project is completed.

NOTES

1 Stig Johansson and Mette-Cathrine Jahr (see Johansson and Jahr 1982) have made major contributions to the project in the preparation of the WORDLIST and SUFFIXLIST (see 3.1). They are also undertaking roughly half of the post-editing. The research at Lancaster has been conducted by Ian Marshall, as well as the present authors. The Lancaster project has been supported by the Social Science Research Council (Research Grant HR 7081/1).

2 The Norwegian Computing Centre for the Humanities (director Jostein Hauge) has provided text processing facilities essential to the project. We have particularly appreciated the programming support provided at the Centre by Knut Hofland.

3 The percentage of 96.7% is based on the post-editing of c. 100 texts (i.e. c. 200,000 text words, or 20% of the Corpus). These texts are from categories B, C, F, G and R, representing a varied cross-section of the Corpus. There is little variation in the tagging success-rate between different categories. The figure of 96.7% excludes errors in the output which are not due to automatic tagging (these are chiefly pre-editing errors, and account for c. 0.1% of all words). Punctuation tags (see Appendix A) are discounted in calculating the success-rate.

4 Approximately 55% of the Corpus has been automatically tagged by November 1982.

5 Reported in Francis (1980); for results and analysis of the automatic tagging, see Francis and Kučera (1982).

6 An experiment carried out by Knut Hofland at Bergen in 1982 gave encouraging support to the view that manual pre-editing could be dispensed with. The LOB tagging programs were applied to a machine-readable copy of John Osborne's *Look Back in Anger*, a text not included in the LOB Corpus. Automatic pre-processing followed by

automatic tagging resulted in a success-rate in the region of 90%. This was without modifications to the programs themselves, which are designed to accept the specially pre-edited text of the LOB Corpus. (See p. 7f. above.)

7 See Marshall (1982:10-12) for further details.

8 Each of the three programs was written by a different member of the research team: A by Roger Garside, B by Eric Atwell, and C by Ian Marshall.

9 The Brown Wordlist contained c. 3,000 words, and the Brown Suffix-list contained c. 450 word-endings. See Johansson and Jahr (1982) on the LOB Suffixlist.

10 The marker @ indicates that a tag has (notionally) an intrinsic likelihood of 10% or less; the marker % indicates that a tag has (notionally) an intrinsic likelihood of 1% or less. The tags are also output in order of likelihood, more likely tags being placed to the left of less likely ones. To this extent, the Tag Selection program makes use of probabilities.

11 Roger Garside and Fanny Leech are currently working on programs to be applied in the parsing of the LOB Corpus. Manual work on semantic tagging is being undertaken at Stockholm by Magnus Ljung.

REFERENCES

Chomsky, N. 1957. *Syntactic Structures*. The Hague: Mouton.

Francis, W. Nelson. 1980. 'A Tagged Corpus - Problems and Prospects'. In S. Greenbaum, G. Leech, and J. Svartvik, eds. *Studies in English Linguistics - for Randolph Quirk*. London: Longman. 192-209.

Francis, W. Nelson and Henry Kučera. 1982. *Frequency Analysis of English Usage: Lexicon and Grammar*. Boston: Houghton Mifflin.

Greene, Barbara B. and Gerald M. Rubin. 1971. 'Automatic Grammatical Tagging of English'. Providence, R.I.: Department of Linguistics, Brown University.

Johansson, Stig and Mette-Cathrine Jahr. 1982. 'Grammatical Tagging of the LOB Corpus: Predicting Word Class from Word Endings'. *In* S. Johansson, ed. *Computer Corpora in English Language Research*. Norwegian Computing Centre for the Humanities, Bergen. 118-46.

Marshall, Ian. 1982. 'Choice of Grammatical Word-Class without Global Syntactic Analysis for Tagging Words in the LOB Corpus'. Department of Computer Studies, University of Lancaster.

APPENDIX A: A SELECTION OF TAGS FROM THE LOB TAGSET

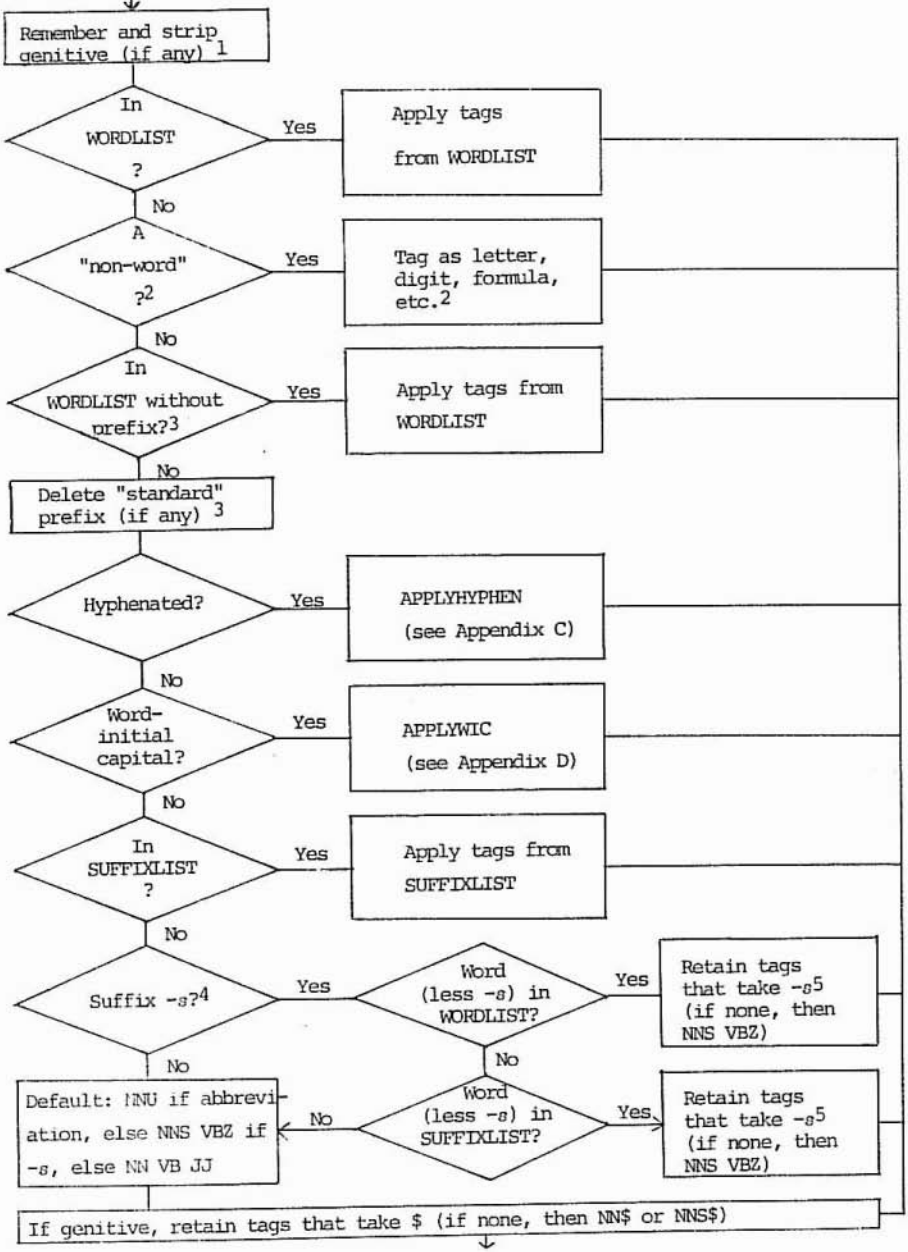
Note 1: The following punctuation tags represent themselves:
"." , "..." , "(" , ":" , "**'" , "*-" , "*'" , ")" , ";" , "?" , ":" , ","

Note 2: The letter "S" added to a tag marks it as plural; e.g. "NNS" = "plural common noun"

Note 3: The dollar sign added to a tag marks it as genitive or possessive; e.g. "NNS$" = "genitive plural common noun".

&FO	formula
AT	singular article (*a, an, every*)
ATI	singular or plural article (*the, no*)
CD	cardinal numeral
CD-CD	hyphenated pair of cardinal numerals
CS	subordinating conjunction
DT	singular determiner
DTI	singular or plural determiner
IN	preposition
JJ	adjective
JJB	attributive adjective
NNU	unit of measurement unmarked for number (e.g. *ft., cc., m.p.h.*)
NN	singular common noun
NNP	singular common noun with word-initial capital (e.g. *Irishman*)
NP	singular proper noun
NPL	singular locative noun with word-initial capital (e.g. *Square*)
NPT	singular titular noun with word-initial capital (e.g. *Mr, Lord*)
NR	singular adverbial noun (e.g. *north, home*)
OD	ordinal numeral
PP1A	*I*
PP1O	*me*
PP2	*you*
PP3	*it*
QL	qualifier (e.g. *very, more*)
RB	adverb
RI	prepositional adverb (homograph of preposition)
RP	prepositional adverb which can also be a particle
VB	verb (uninflected form)
VBD	past tense verb
VBN	past participle
VBZ	verb (3rd person singular present tense)

APPENDIX B: General flowchart of Tag Assignment Program (see 3.1)



NOTES

1 If the word ends in "*s* apostrophe" then strip the apostrophe; if the word ends in "apostrophe *s*" then strip both characters (and any preceding full-stop).

2 "Non-words" are the following:

a letter followed by zero or more digits (0 to 9), possibly followed by a single, double, or triple prime, tagged ZZ

a number* followed by "st", "nd", "rd" or "th", tagged OD

a number followed by "s" tagged CDS

a number containing "-", tagged CD-CD

a number followed by "apostrophe s", tagged CD$

a number followed (possibly) by a letter, tagged CD

a word containing a superscript or subscript, tagged &FO

a word containing letters and digits, but no hyphen, tagged &FO

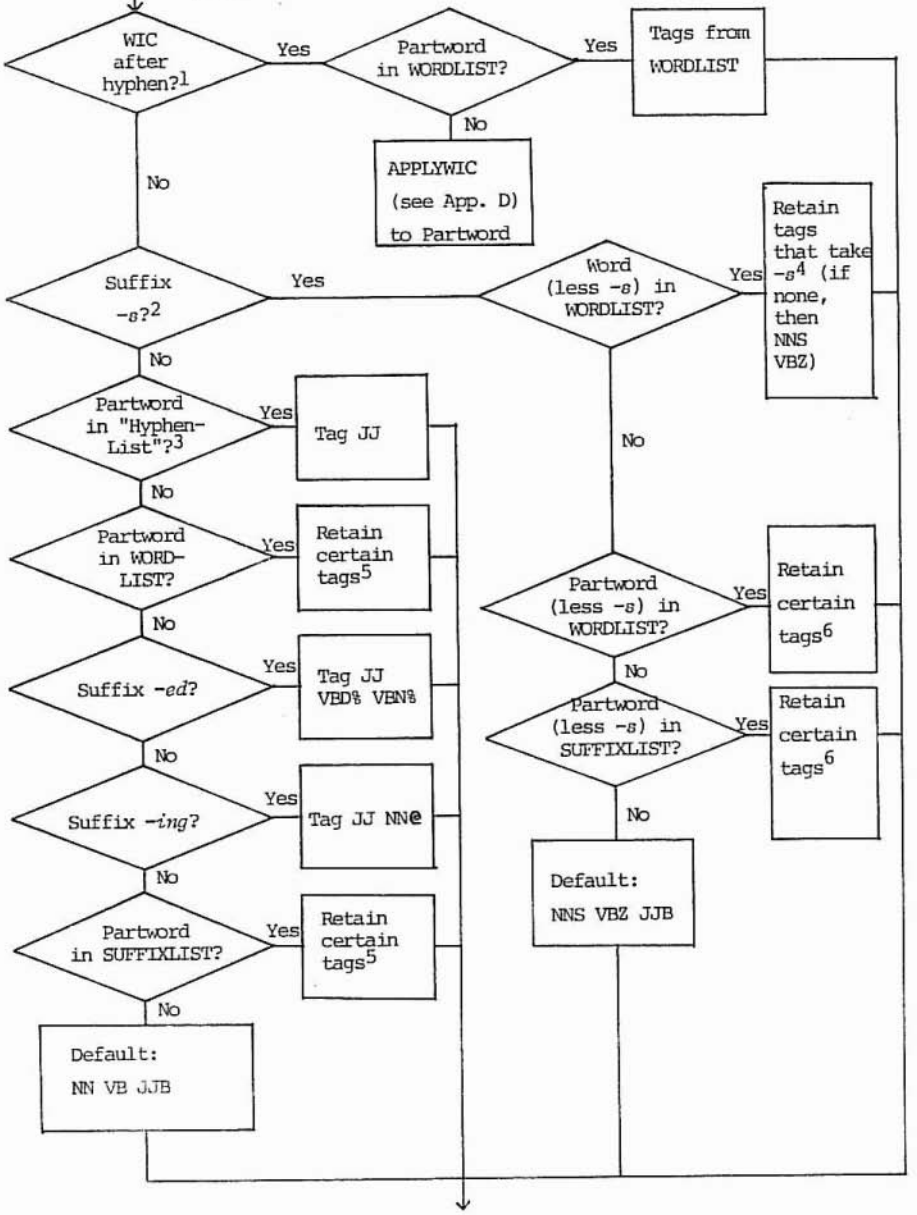
*In this context, a "number" means a sequence of digits (0-9) perhaps also including ".", "," and "/".

3 The "standard" prefixes include "a-", "co-", "counter-", "de-", "hyper-", "mis-", "out-", "over-", "re-", "retro-", "super-", and "trans-".

4 Words ending "ches", "shes", "sses", "zzes", "oes", "xes" have the "es" removed: words with 5 or more letters and ending in "ies" have the "ies" changed to "y"; words ending in "full-stop s" have both characters removed; other words ending in "s" (unless they end in "ss") have it removed.

5 Tags that take *-s* are VB (becoming VBZ) and CD, NN, NNP, NNU, NP, NPL, NPT, NR (becoming CDS, NNS, NNPS, NNUS, NPS, NPLS, NPTS, NRS).

APPENDIX C: Tagging decisions of APPLYHYPHEN
(Note: "Partword" means the characters after the last hyphen)



NOTES

1 "WIC" means "Word-initial Capital".

2 See Note 4, Appendix B.

3 The "Hyphen-List" consists of "class", "hand", "like", "price", "proof", "quality", "range", "rate", and "scale".

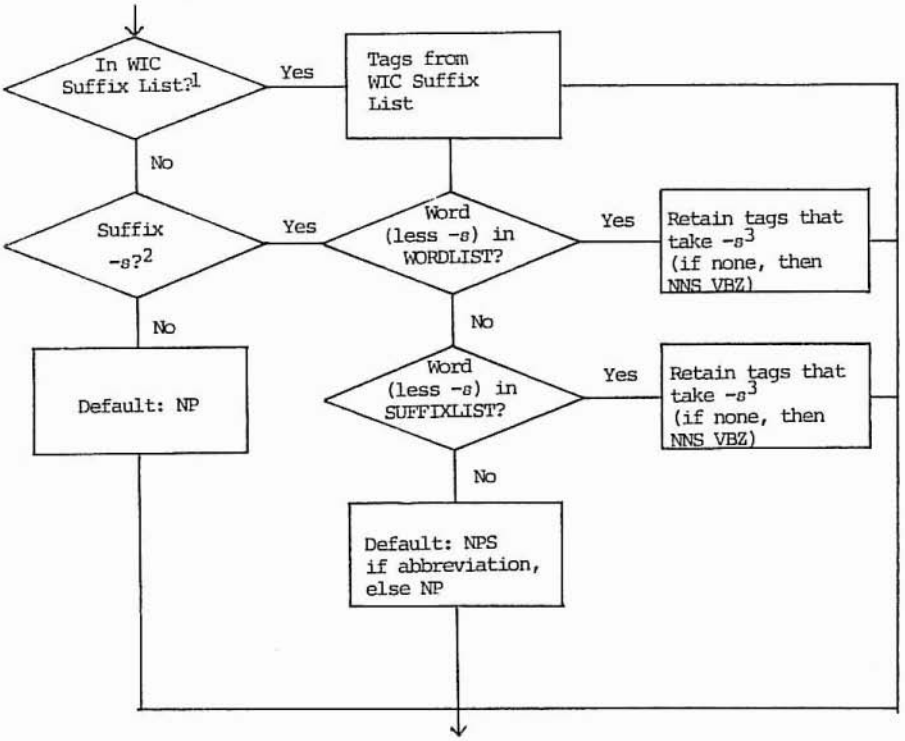
4 See Note 5, Appendix B.

5 For words not ending in "s", if IN is one of the tags, tag the word NN JJ@; if VBN is one of the tags, tag the word JJ; if VBG is one of the tags, tag the word JJ NN VBG@; if NNU is one of the tags, tag the word JJB; if NN with "normal" probability is one of the tags, tag the word NN JJB; otherwise leave the tags unchanged.

6 For words ending in "s", if IN is one of the tags, tag the word NNS; if VBG is one of the tags, tag the word NNS; if NNU is one of the tags, the tag is JJB; if NN with "normal probability" is one of the tags, the tag is NNS; otherwise retain tags that take "s" (see Note 5, Appendix B). If there are none, then tag the word NNS VBZ.

APPENDIX D: Tagging decisions of APPLYWIC

("WIC" means "Word-initial Capital")



Notes

1 The WIC Suffix List contains the following endings: "ic", "ese", "ite", "esque", "ish", "ism", "ean", "ian", "woman", "women", "ation", "ist".

2 See Note 4, Appendix B.

3 See Note 5, Appendix B.

APPENDIX E: SPECIMEN OF VERTICAL OUTPUT (before post-editing)

```
--------------------------------
        thus  RB
          it  PP3
          is  BEZ
       clear  [JJ]/73 RB%/23 NN@/3 VB@/1
        that  CS
         the  ATI
 predominant  JJ
organization  NN
           ,  ,
particularly  RB
          in  IN
         the  ATI
distribution  NN
          of  IN
manufactured  JJ
       goods  NNS
           ,  ,
          is  BEZ
         the  ATI
   wholesale  JJ
    merchant  NN
         who  WP
     carries  VBZ
      stocks  NNS
           .  .
```

APPENDIX F: THE SAME PASSAGE AS REHORIZONTALIZED OUTPUT

```
^ thus it  is  clear that the predominant organization, particularly
^  RB  PP3 BEZ  JJ    CS  ATI      JJ           NN     ,       RB

in the distribution of manufactured goods, is  the wholesale merchant
IN ATI      NN       IN      JJ       NNS , BEZ ATI     JJ       NN

who carries stocks.
WP    VBZ     NNS .
```

CONSTITUENT-LIKELIHOOD GRAMMAR

Eric Steven Atwell
University of Lancaster, England

A *INTRODUCTION*

The paper by Leech *et al.* describes the aims of the LOB Corpus Grammatical Tagging project, and explains the suite of programs we are using to achieve these aims. In this paper, I would like to look in greater detail at the theoretical basis of these programs; I shall attempt to explain exactly what *constituent-likelihood grammar* involves, and suggest some other applications of this probabilistic approach to natural language syntax analysis.

A.1 General principles of CL grammar

The CL grammar used in the LOB Corpus project is specifically designed to be used in tagging, that is, in assigning a grammatical-class marker to each word in a text. In fact, the basic principles could be generalized to apply to other levels of linguistic analysis (parsing, semantic analysis, etc.); in general, if the analysis involves assigning 'labels' to 'constituents', then a CL grammar could be devised for this analysis.

The CL method of grammatical analysis involves two steps:

(i) Each 'constituent' is first assigned a set of *potential* 'labels'. This can be done by some quite simple mechanism such as dictionary-lookup; this may well mean that some of the possible labels are in fact inappropriate in the given context, but this does not matter, since they will be eliminated during the second stage.

(ii) Each of the potential labels of a constituent is then assigned a *relative likelihood* figure, using a formula which takes into account contextual and other relevant factors; having done this, we can then choose the single 'best' label for the constituent, and disregard all the others (no matter how many others there happen to be).

Thus a CL grammar should not be viewed as a set of rules for generating sentences; rather, it is characterized by:

(i) an algorithm for assigning a set of possible 'labels' or tags to any given constituent; and

(ii) a general *relative likelihood formula* which can be used to calculate the relative likelihood of any given label or tag in any given context.

A.2 The LOB CL grammar

In the CL grammar used to analyse the LOB Corpus, the 'labels' are grammatical tags, and the 'constituents' are words (in this special case, all the 'constituents' are at the same 'level'; but this does not mean that CL grammar could not be generalized to deal with more complex structuring).

The tag-assignment algorithm is embodied in the program WORDTAG. Tags are assigned mainly by dictionary-lookup; but since the set of possible words in the English language is open-ended, the algorithm also includes a number of 'default' routines to deal with words which 'fall through the net' (as explained in Leech *et al.*). This means that the tag-assignment algorithm can be used to assign a set of potential tags to *any* word, and this set will almost always include the 'intuitively correct' tag.

Probably the most innovative part of the LOB CL grammar is the general *relative likelihood formula* used by the 'tag-disambiguation' program CHAINPROBS. When a word has been assigned more than one potential tag, this formula is used to find the relative likelihood of each candidate. We have found that a very simple formula, taking into account only the immediate context, will correctly choose the 'best' tag in c. 96-97% of all cases (moreover, this high success rate is consistent regardless of style: novels, newspapers, magazines, etc. all have approximately the same success rate). Section B explains the Tag Relative Likelihood formula in greater detail.

A.3 Other applications of CL grammar

The CL-grammar approach to language analysis was developed specifically for the LOB Corpus Grammatical Tagging research project. However, it has become clear that this method of analysis has many other possible areas of application. The two main advantages of CL grammar

over other methods of natural language analysis are:

(i) *Generality and robustness*: 'Rule-based' analysis algorithms tend to work only with sentences that 'follow the rules', and will fail if presented with 'non-standard' English, accidental misspellings, or other 'deviant' input. Unfortunately, as become clear when researching with a large corpus, 'real-life' English texts are often dotted with many of these 'imperfections'! In contrast, the LOB Corpus Tagging programs are extremely general and 'robust', since they will produce a reasonably acceptable analysis of *any* input (they have successfully dealt with newspaper 'telegraphese', 'foreigner English', Sci-Fi neologisms, and even a 'humorous' text peppered with *deliberate* misspellings!).

(ii) *Simplicity*: Most syntax-analysis programs build a complex 'parse tree' for each sentence, which requires much complicated and time-consuming computation. CL grammar, on the other hand, involves analysis at a 'local level' only; the tag-likelihood function looks at the immediate context only, *not* at a whole sentence; and even within this localized context, the computation is very straightforward. This means that the amount of computation is much less; the analysis is much simpler and faster.

These advantages make CL grammar particularly suitable for applications requiring a simple and fast analysis of a wide range of possible linguistic input. In sections C, D, and E I shall look briefly at three potential uses of CL grammar; a spelling and grammar 'checker' for use in Word Processors, a speech analysis program for converting from spoken to written English, and a general Grammatical Parser for the LOB Corpus.

B THE LOB TAG RELATIVE LIKELIHOOD FUNCTION: HOW WE DEVELOPED THE FORMULA

To give the reader a clearer idea of how 'likelihoods' are calculated in a CL grammar, I will attempt in this section to explain the Tag Relative Likelihood Function used in tagging the LOB Corpus; I will do this by explaining step by step how we developed the mathematical formula.

B.1 A formalism for words and tags

The programs before CHAINPROBS (where the likelihood formula is applied) divide the texts of the LOB Corpus into *records*, where each record contains a single word and a set of potential tags, and each record has a unique reference number (in fact, each record is a separate line of text; but I prefer the term 'record' (rather than 'line'), since this avoids confusion (different words which were on the same line in the original Corpus are in different records)). If we denote the record-number by r, the word by W<r>, and the set of tags by T<r,1>, T<r,2>, T<r,3>, ... T<r,n(r)>, where n(r) is the number of potential tags in the record r, then a typical sequence of records from the LOB Corpus is:

record-no.	word	tags
.		
.		
.		
r-1	W<r-1>	T<r-1,1>, T<r-1,2>, ... T<r-1,n(r-1)>
r	W<r>	T<r,1>, T<r,2>, ... T<r,n(r)>
r+1	W<r+1>	T<r+1,1>, T<r+1,2>, ... T<r+1,n(r+1)>
.		
.		
.		

B.2 Relative and absolute likelihood

CHAINPROBS assigns a percentage likelihood figure to each tag in a record. This percentage is the *relative likelihood* of the tag, relative to all the other potential tags in the record. The *relative likelihood* l of a tag T<r,a> is the *absolute likelihood* L of that tag, divided by the sum of the absolute likelihoods of all the potential tags in the record r:

$$l(T\langle r,a\rangle) = \frac{L(T\langle r,a\rangle)}{\sum_{i=1..n(r)} L(T\langle r,i\rangle)}$$

B.3 'Factorizing' likelihood: L = Lb * Lf * Lw

The absolute likelihood function must now be defined. Ideally, we would like this function to take into account *all* relevant contextual information; this would be the 'perfect' absolute tag-likelihood function. Unfortunately, it is not immediately apparent exactly what such a formula should look like. However, we *can* work towards this 'perfect' formula, step by step: first we must write some simple formula which approximates to the 'ideal'; then, we can add on extra factors to take into account more peripheral information.

To begin with, we can say that the absolute likelihood of a tag is dependent on the '*backward context*' (i.e. the preceding tags) and the '*forward context*' (i.e. the following tags); this allows us to separate out '*backward likelihood*' Lb and '*forward likelihood*' Lf. Another important factor in deciding the likelihood of a tag is of course the *word* it is to be assigned to: for example, with the word "water", the tag NN (noun) is likelier than the tag VB (verb). Thus the absolute likelihood formula must also take into account Lw, the '*word-tag likelihood*'.

The simplest formula for absolute likelihood which takes these three factors into account is:

L = Lb * Lf * Lw

(where * represents multiplication). This is our first approximation to a 'perfect' likelihood function.

B.4 Tag-pair bond B

To calculate the likelihood of a tag T<r,a>, let us assume to begin with that the records immediately before and after r each have only one unambiguous tag. Furthermore, let us assume that the only thing relevant in the 'backwards context' is the single tag in the previous record, T<r-1,1>; and likewise that the only relevant factor in the 'forward likelihood' is the tag T<r+1,1>.

This means that the 'backward likelihood' can be defined as simply the '*bond*' between T<r,a> and the preceding tag T<r-1,1>; and likewise, that Lf is simply the '*bond*' between T<r,a> and T<r+1,1>:

Lb(T<r,a>) = B(T<r-1,1>, T<r,a>)

Lf(T<r,a>) - B(T<r,a>, T<r+1,1>)

Values of the *tag-pair bond* function B are stored in a table with a row and column for every tag in the LOB tagset.

The 'bond' between a pair of tags T1, T2 is dependent on the frequency of cooccurrence, f(T1,T2), compared to the frequency of occurrence of each tag individually, f(T1) and f(T2). These statistics must be extracted from texts which have already been tagged unambiguously (in the LOB Corpus Grammatical Tagging project, we extracted these figures from the Brown Corpus initially (making adjustments where the tagsets differ), but later statistics include figures drawn from the first sections of the LOB Corpus to be analysed).

B.5 Calculating values of B for each tag-pair (T1,T2)

If tags were combined randomly (i.e. if context had no influence on the choice of tag with a word), then the ('random') probability of tag T1 being followed by tag T2 would be

$$P_{\langle random \rangle}(T1,T2) = \frac{f(T1) * f(T2)}{N1}$$

(N1 is a constant, dependent on the number of tags in the sample.)

The actual ('true') probability of the tag-pair (T1,T2) is

$$P_{\langle true \rangle}(T1,T2) = \frac{f(T1,T2)}{N2}$$

(N2 is another constant.)

If we divide P<true> by P<random>, we get a very simple measure of the 'correlation' or *bond* between T1 and T2; ignoring the constant factor (N1/N2) we get the formula:

$$B(T1,T2) = \frac{f(T1,T2)}{f(T1) * f(T2)}$$

The value of B(T1,T2) for any tag-pair (T1,T2) is thus dependent on the sample from which the frequency statistics are derived, so clearly it is important that the sample is representative, and reasonably large. However, even with a very large sample, we cannot be certain

that the figures are perfect, especially if a particular frequency figure happens to be very low or zero; for example, if for a given sample f(DT,DOD)=0, does this mean that the tag-pair (DT,DOD) can *never* cooccur in English, or is this simply a failing of this particular sample? It is safer to assume the latter; so we must add a constant k1 to *all* tag-pair frequency figures, to ensure that all values are greater than zero. Similarly, we should add a constant k2 to all single-tag frequency figures, to ensure that we can never divide by zero. Thus, the new definition of B is

$$B(T1,T2) = \frac{f(T1,T2) + k1}{(f(T1)+k2) * (f(T2)+k2)}$$

B.6 Word-tag likelihood Lw

'Word-tag likelihood' is the likelihood that a given word will have a given tag, regardless of other factors. Dictionary-lookup (or equivalent mechanisms) can give us a very crude measure of Lw: if the tag occurs with the word in the dictionary, then Lw is 1, otherwise 0 (e.g. Lw("the",ATI)=1, but Lw("the",VB)=0).

In the LOB Corpus CL grammar, we found that this 'binary' likelihood function was too crude and simplistic, so we included *four* 'levels' of word-tag likelihood. The 'binary' values of Lw, 0 and 1, are *implicitly* assigned by straightforward dictionary-lookup, as explained above; in addition, the Wordlist used in the LOB Corpus CL grammar has two *explicit* Lw 'weighting markers' (@ and %): if a tag appears with a word only rarely, then that tag is marked @, and if the tag is *very* rare with a given word, it is marked %, for example:

alert JJ VB NN@
water NN VB%
major JJ NN@ VB%

(Notionally @ means that the tag appears with the given word in 10% or less of all uses, and % means 1% or less. In fact often the assignment of weightings was based on 'intelligent guesses', particularly with rare words; this is one reason why we decided to limit ourselves to only four 'grades' of word-tag likelihood (this decision has since been vindicated by the consistently high success rate of the tagging programs: it is clear that a much more 'refined' system of gradations of Lw is unlikely to improve tagging results very

significantly.))

These weighting-markers appear in the LOB WORDLIST, SUFFIXLIST, and IDIOMLIST, and are assigned by WORDTAG (and IDIOMTAG). In fact, within the theoretical framework of a CL grammar, the assignment of these weightings is *not* a necessary part of the tag-assignment algorithm; more correctly, it 'belongs' with the mechanism for calculating tag likelihoods. In other words, if the two tasks of
(i) assigning potential tags to each word, and
(ii) calculating likelihoods for each potential tag
were autonomously dealt with by WORDTAG and CHAINPROBS respectively, then the @ and % 'weighting-markers' would *not* be assigned by WORDTAG; instead, every time CHAINPROBS applied the tag-likelihood function to a tag, it would have to find the appropriate value of Lw for that word-tag combination. Of course, this would require exactly the same word-tag lookup algorithm as was used by WORDTAG to assign the potential tag in the first place; so, to save time, WORDTAG assigns potential tags *and* Lw weighting-markers (where appropriate) in a single search.

B.7 Generalizing the formula to deal with ambiguous contexts

The formulae for Lb and Lf given in B.4 assume that the records immediately before and after the current record are unambiguously tagged, so that in working out the likelihood of tag T<r,a> the only tags we need take into account are T<r-1,1> and T<r+1,1>. However, if either of these records are in fact *ambiguous*, we must take the other tags into account also. For example, if the immediately *preceding* record is ambiguously tagged, then the formula for *backward likelihood* Lb must take into account not only T<r-1,1>, but also all the other potential tags in record r-1: T<r-1,2>, T<r-1,3>, ... T<r-1,n(r-1)>.

For each potential preceding tag T<r-1,i>, we must take into account the *bond* between T<r-1,i> and T<r,a>, 'weighted' by the Backward Likelihood in turn of T<r-1,i>, and also the Word-Tag Likelihood Lw of T<r-1,i>. Thus, *backward likelihood* must be redefined as a recursive function:

$$Lb(T\langle r,a\rangle) = \sum_{i=1..n(r-1)} \begin{pmatrix} B(T\langle r-1,i\rangle, T\langle r,a\rangle) \\ *Lw(W\langle r-1\rangle, T\langle r-1,i\rangle) \\ *Lb(T\langle r-1,i\rangle) \end{pmatrix}$$

Forward likelihood must also be redefined, so it can deal with sequences of tag-ambiguities:

$$Lf(T\langle r,a\rangle) = \sum_{i=1..n(r+1)} \begin{pmatrix} B(T\langle r,a\rangle, T\langle r+1,j\rangle) \\ *Lw(W\langle r+1\rangle, T\langle r+1,j\rangle) \\ *Lf(T\langle r+1,j\rangle) \end{pmatrix}$$

Notice that the recursive definition of Lb means that the *backward likelihood* of a tag T<r,a> theoretically takes into account *all* tags preceding T<r,a>; however, in calculating *relative* likelihood, the set of possible 'backward contexts' before the last *unambiguous* tag is the same for all the potential tags in record r, so this can be "cancelled out". Similarly, *forward likelihood* recursively defined should theoretically involve *all* tags after T<r,a>; but in calculating *relative* likelihood all bonds after the next unambiguous tag "cancel out" and can thus be ignored.

In other words, when calculating the relative likelihood of any tag using the general formulae for Lb and Lf, we need only 'look back' as far as the *last unambiguous tag*, and we need only 'look forward' as far as the *next unambiguous tag*. In general, tags are 'disambiguated' by looking *only* at the words in the *immediate context*.

B.8 The relative likelihood function

As an example, let us take a sequence of five records, with five consecutive words: A, B, C, D, E; and with tags: a, b, b', c, c', d, d', e (the first and last records are unambiguously tagged, while the intermediate records have two tags each):

record no.	word	tags
r1	A	a
r2	B	b, b'
r3	C	c, c'
r4	D	d, d'
r5	E	e

To show how the formulae are applied, let us calculate l(d), the relative likelihood of the tag d. The formula from B.2 tells us

$$l(d) = \frac{L(d)}{L(d) + L(d')}$$

L(d) and L(d') can be expanded using the formula from B.3:

$$L(d) = Lb(d) * Lf(d) * Lw(D,d)$$
$$L(d') = Lb(d') * Lf(d') * Lw(D,d')$$

Applying the recursive formulae for Lb and Lf from B.7, these equations expand to:

$$L(d) = Lb(a) * Lw(A,a) * Lf(e) * Lw(E,e) * \left(\begin{array}{l} B(a,b)*Lw(B,b)*B(b,c)*Lw(C,c)*B(c,d)*Lw(D,d)*B(d,e) \\ +B(a,b')*Lw(B,b')*B(b',c)*Lw(C,c)*B(c,d)*Lw(D,d)*B(d,e) \\ +B(a,b)*Lw(B,b)*B(b,c')*Lw(C,c')*B(c',d)*Lw(D,d)*B(d,e) \\ +B(a,b')*Lw(B,b')*B(b',c')*Lw(C,c')*B(c',d)*Lw(D,d)*B(d,e) \end{array} \right)$$

$$L(d') = Lb(a) * Lw(A,a) * Lf(e) * Lw(E,e) * \left(\begin{array}{l} B(a,b)*Lw(B,b)*B(b,c)*Lw(C,c)*B(c,d')*Lw(D,d')*B(d',e) \\ +B(a,b')*Lw(B,b')*B(b',c)*Lw(C,c)*B(c,d')*Lw(D,d')*B(d',e) \\ +B(a,b)*Lw(B,b)*B(b,c')*Lw(C,c')*B(c',d')*Lw(D,d')*B(d',e) \\ +B(a,b')*Lw(B,b')*B(b',c')*Lw(C,c')*B(c',d')*Lw(D,d')*B(d',e) \end{array} \right)$$

We can think of a term such as

B(a,b)*Lw(B,b)*B(b,c)*Lw(C,c)*B(c,d)*Lw(D,d)*B(d,e)

as a *chain*, represented by [abcde]. This notational simplification allows us to rewrite the equation for the relative likelihood thus:

$$l(d) = \frac{L(d)}{L(d) + L(d')}$$

$$= \frac{[abcde] + [ab'cde] + [abc'de] + [ab'c'de]}{[abcde] + [ab'cde] + [abc'de] + [ab'c'de] + [abcd'e] + [ab'cd'e] + [abc'd'e] + [ab'c'd'e]}$$

$$= \frac{\text{(SUM OF ALL POSSIBLE 'CHAINS' FROM a TO e THROUGH d)}}{\text{(SUM OF ALL POSSIBLE 'CHAINS' FROM a TO e)}}$$

This can be generalized to give us the relative likelihood of any tag T in terms of '*chains*':

$$l(T) = \frac{\text{(sum of all possible 'chains' from the last unambiguous tag to the next unambiguous tag THROUGH TAG T)}}{\text{(sum of all possible 'chains' from the last unambiguous tag to the next unambiguous tag)}}$$

CHAINPROBS actually uses a definition of the likelihood function in terms of 'chains', since it is computationally more efficient; but this new definition is entirely equivalent to the likelihood formulae previously given.

B.9 Modifying the 'one-step' formula in special cases

So far, we have assumed that the tag-likelihood function is a First-Order Markov process: we have assumed that a '*chain*' is composed of a sequence of independent 'links', *bonds* between *pairs* of tags. In trials on a section of the LOB Corpus (over 20,000 words), we found that the formulae above correctly yielded the 'best' tag for c 93-94% of words; so the 'one-step' function is in fact a very close approximation to the 'perfect' likelihood function (we were actually quite surprised that such a simple set of formulae could be

so successful!).

However, among the errors in the remaining 6-7%, there were a significant number of cases where the function clearly needed to look *two* tags backwards or forwards (rather than just one) to calculate the likelihood of a 'link' in a 'chain'. These exceptional cases fell into two main categories:

(i) tag-sequences involving a "noise-tag" such as RB (adverb), e.g. in

"she began to seductively reveal herself"
PP3A VBD TO RB VB PPL

the forward likelihood of TO is much more dependent on VB than on RB, and the backward likelihood of VB is more dependent on TO than RB. In effect, when calculating the likelihood of the tag-sequence, we would like to 'ignore' the "noise-tag" RB.

(ii) tag triples around CC (coordinating conjunction), of the form T<a> CC T<b> : Tag-triples in which T<a> and T<b> are in fact the same tag (e.g. NN CC NN, JJ CC JJ) are far likelier than tag-triples in which T<a> and T<b> differ (e.g. JJ CC NN).

The 'one-step' likelihood function can be used to calculate a likelihood figure for any sequence of three tags T1, T2, T3, essentially by multiplying B(T1,T2) * B(T2,T3). In a few special cases, this tag-triple likelihood must be modified by a *tag-triple scaling factor*, S(T1,T2,T3). These special cases are ones where the overall likelihood of the tag-triple depends on the 'bonding' of T1 and T3, rather than B(T1,T2) and B(T2,T3).

B.10 Summary of the final formula

How is S(T1,T2,T3) to be incorporated into the likelihood formulae? If the immediate context were assumed to be unambiguous, we could simply add a new factor to the formula for absolute likelihood (L(T<r,a>):

```
L(T<r,a>) = Lb(T<r,a>) * Lf(T<r,a>) * Lw(W<r>,T<r,a>)
            *S(T<r-1,1>,T<r,a>,T<r+1,1>)
```

To be able to deal with ambiguous contexts, we must generalize this formula to:

$$L(T\langle r,a\rangle) = Lw(W\langle r\rangle, T\langle r,a\rangle) * \sum_{i=1..n(r-1)} \left(\begin{array}{l} B(T\langle r-1,i\rangle, T\langle r,a\rangle) * Lw(W\langle r-1\rangle, T\langle r-1,i\rangle) * Lb(T\langle r-1,i\rangle) \\ * B(T\langle r,a\rangle, T\langle r+1,j\rangle) * Lw(W\langle r+1\rangle, T\langle r+1,j\rangle) * Lf(T\langle r+1,j\rangle) \\ * S(T\langle r-1,i\rangle, T\langle r,a\rangle, T\langle r+1,j\rangle) \end{array} \right)$$

$j=1..n(r+1)$

The formulae for Lb and Lf must be similarly modified to take S into account. The above formula for L is considerably more complex than that of B.3. However, since S(T1,T2,T3) only 'comes into play' in a few special cases, the extra computation is often redundant. There is an alternative (equivalent) formula which is computationally much more efficient (even though the formula looks more complicated at first sight); it contains a separate factor dealing with S, which 'cancels out' to 1 (and can thus be ignored) in most cases. This formula is given below, in the following summary of the LOB CL Grammar tag likelihood formulae:

Relative likelihood:

$$l(T\langle r,a\rangle) = \frac{L(T\langle r,a\rangle)}{\sum_{i=1..n(r)} L(T\langle r,i\rangle)}$$

Absolute likelihood:

$$L(T\langle r,a\rangle) = Lb(T\langle r,a\rangle) * Lf(T\langle r,a\rangle) * Lw(W\langle r\rangle,T\langle r,a\rangle) *$$

$$\sum_{\substack{i=1..n(r-1)\\ j=1..n(r+1)}} \left(\begin{array}{l} B(T\langle r-1,i\rangle,T\langle r,a\rangle) * Lw(W\langle r-1\rangle,T\langle r-1,i\rangle) * Lb(T\langle r-1,i\rangle) \\ * B(T\langle r,a\rangle,T\langle r+1,j\rangle) * Lw(W\langle r+1\rangle,T\langle r+1,j\rangle) * Lf(T\langle r+1,j\rangle) \\ * S(T\langle r-1,i\rangle,T\langle r,a\rangle,T\langle r+1,j\rangle) \end{array} \right)$$

$$\sum_{\substack{i=1..n(r-1)\\ j=1..n(r+1)}} \left(\begin{array}{l} B(T\langle r-1,i\rangle,T\langle r,a\rangle) * Lw(W\langle r-1\rangle,T\langle r-1,i\rangle) * Lb(T\langle r-1,i\rangle) \\ * B(T\langle r,a\rangle,T\langle r+1,j\rangle) * Lw(W\langle r+1\rangle,T\langle r+1,j\rangle) * Lf(T\langle r+1,j\rangle) \end{array} \right)$$

Backward likelihood:

$$Lb(T\langle r,a\rangle) =$$

$$\sum_{i=1..n(r-1)} \left(\begin{array}{l} \sum_{h=1..n(r-2)} \left(\begin{array}{l} B(T\langle r-1,i\rangle,T\langle r,a\rangle) * Lw(W\langle r-1\rangle,T\langle r-1,i\rangle) * Lb(T\langle r-1,i\rangle) \\ B(T\langle r-2,h\rangle,T\langle r-1,i\rangle) * Lw(W\langle r-2\rangle,T\langle r-2,h\rangle) * Lb(T\langle r-2,h\rangle) \\ * S(T\langle r-2,h\rangle,T\langle r-1,i\rangle,T\langle r,a\rangle) \end{array} \right) \\ \hline \sum_{h=1..n(r-2)} \left(B(T\langle r-2,h\rangle,T\langle r-1,i\rangle) * Lw(W\langle r-2\rangle,T\langle r-2,h\rangle) * Lb(T\langle r-2,h\rangle) \right) \end{array} \right)$$

Forward likelihood:

Lf(T<r,a>) =

$$\sum_{j=1..n(r+1)} \left(\frac{\sum_{k=1..n(r+2)} \left(\begin{array}{l} B(T\langle r,a\rangle, T\langle r+1,j\rangle) * Lw(W\langle +-1\rangle, T\langle r+1,j\rangle) * Lf(T\langle r+1,j\rangle) \\ B(T\langle r+1,j\rangle, T\langle r+2,k\rangle) * Lw(W\langle r+2\rangle, T\langle r+2,k\rangle) * Lf(T\langle r+2,k\rangle) \\ * S(T\langle r,a\rangle, T\langle r+1,j\rangle, T\langle r+2,k\rangle) \end{array} \right)}{\sum_{k=1..n(r+2)} \left(B(T\langle r+1,j\rangle, T\langle r+2,k\rangle) * Lw(W\langle r+2\rangle, T\langle r+2,k\rangle) * Lb(T\langle r+2,k\rangle) \right)} \right)$$

The alternative definition of relative likelihood in terms of '*chains*' is now:

```
l(T) =   sum of all possible 'CHAINS'
         FROM the LAST unambiguous tag
         not in the middle of a 'special case' tag-triple
         TO the NEXT unambiguous tag
         not in the middle of a 'special case' tag-triple
         THROUGH TAG T
        ------------------------------------------------
         sum of all possible 'CHAINS'
         FROM the LAST unambiguous tag
         not in the middle of a 'special case' tag-triple
         TO the NEXT unambiguous tag
         not in the middle of a 'special case' tag-triple
```

B.11 Potential for further improvement

The current success rate of CHAINPROBS is consistently 96.5-97%. Theoretically this could be improved by adding further factors to the formulae, taking more contextual information into account by going beyond the simple 'Augmented First-Order Markov' model (CL Grammar is ideally suited to 'enhancement through feedback').

However, the law of diminishing returns suggested to us that it would probably be easier simply to correct remaining tagging-errors 'by hand' than to spend time and effort enhancing the formulae further (at least, this is quicker in the short term, for the immediate task of tagging the LOB Corpus; for new corpora, improvements may well be worthwhile).

The types of construct in which the remaining errors tend to occur are listed in the Manual Postedit Handbook (Atwell *et al.*). In general, many of these problem-cases call for 'higher-level' grammatical or semantic analysis, which would require major enhancements of the present tagging programs. Nevertheless, we feel that our remarkable success rate using such a simple model of language is highly significant.

C ADAPTING THE LOB CL GRAMMAR TO DETECT SPELLING AND GRAMMATICAL ERRORS

As explained in section A, the LOB Grammatical Tagging Programs perform a very simple grammatical analysis of input texts. This 'surface' approach makes the programs much faster than 'full-blooded' parsers; so they are ideally suited to applications where a 'basic' level of grammatical analysis is all that is required.

One such application is in the automatic detection of spelling and grammatical errors in input English texts. In this section, I shall explain how the current LOB Grammatical Tagging programs have been superficially modified to detect such errors in a short sample text; and I shall discuss what further research is required to produce an efficient general-purpose *automatic error-detection* program for commercial Word Processing applications.

C.1 Current 'spelling-checkers' do not look at context

A number of programs are currently available which claim to 'check spelling' in English texts. However, these programs are limited to simple dictionary-lookup: each input word is checked against a large Lexicon, and any word not found is assumed to be misspelt. Unfortunately, this simple method allows many errors to 'slip through' undetected: if a misspelling happens to coincide with another valid word (as in "I now how to prophecy the whether!"), then it is accepted.

Errors such as "now", "prophecy", and "whether" in the example *could* be detected by simple grammatical analysis: for example, the subordinating conjunction "whether" is easily confused with the noun "weather"; and a noun is much likelier than a subordinating conjunction in the context

"... the X!"

so "whether" is probably a misspelling of "weather" in this context.

C.2 Adapting the LOB Grammatical Tagging Programs

Notice that this sort of error can be detected simply by comparing relative likelihoods of word-tags; no higher level of grammatical analysis is required. Clearly the LOB CL Grammar is ideally suited to this kind of analysis. Only a few superficial modifications were needed to convert the current Grammatical Tagging Programs into a prototype 'context-sensitive' spelling-checker (these mainly related to input/output formats).

More important than the adjustments to the programs was the change in the role of the wordlist. In Grammatical Tagging, wordlist-lookup is just one of several methods of tag-assignment available to WORDTAG: there were a number of 'default' routines for words not found in the wordlist. In a spelling-checker, these 'default' routines are not required, in fact, they must not be used at all: if a word is not found in the wordlist, then we can assume it is a misspelling immediately, without the need for 'context-compatibility' checking. Therefore, the Lexicon of a spelling-checker must be much larger than the current LOB wordlist.

Another difference is that each entry in the Lexicon must not only contain a word's 'own' tags, but also the tags of any similar words, the *error-tags*. For example, in the sentence given above ("I now how to prophecy the whether!"), the misspelt word "prophecy" can be detected by grammatical analysis *only* if we know that it is a noun, *and* that there exists a very similar verb ("prophesy"); so the Lexicon entry for "prophecy" must give not only the word's 'own' tag NN, but also the error-tag VB:

WORD	TAG(S)	ERROR-TAG(S)
prophecy	NN	VB£

Note that error-tags are marked with £ to distinguish them from 'own' tags.

C.3 Trial run of the adapted LOB tagging programs

To put the theory to the test, a short text was devised, full of deliberate spelling mistakes which could *only* be detected by grammatical analysis. Also, a sample Lexicon was compiled, with an entry for each word in the text. This text was then processed by the adapted LOB tagging programs:

(i) VERTICALIZE put each word on a separate line (record), and also tagged punctuation marks (so these do not have to be included in the Lexicon)

(ii) WORDTAG assigned a set of tags and error-tags to each word, by Lexicon-lookup (any word not found in the Lexicon can be marked as an error at this stage)

(iii) CHAINPROBS used the Tag Likelihood function to choose the 'best' tag for each word; if an error-tag (marked £) was chosen, then this indicated a probable misspelling

(iv) LOBFORMAT (renamed MARKERRORS) 'rehorizontalized' the text, writing the message "ERROR?" underneath all words which had been 'error-tagged'.

The output from this trial run is shown in Appendix A. Almost all the errors in the text are flagged; but *none* would be uncovered by current 'spelling-check' programs.

C.4 From prototype to general-purpose program

Much research still has to be carried out to transform a 'prototype' into a general-purpose spelling-checker for commercial Word Processing packages:

(i) Compile a very large Wordlist, much bigger than the current LOB wordlist.

(ii) Modify the LOB Tagset (and Tag-Pair Bond function table): the number of tags in the current LOB Tagset is 134, but experience has shown that many tags could be 'merged' or eliminated with little loss of accuracy (many of the finer distinctions drawn in the LOB Tagset are linguistically interesting, but not required for spelling-checking); this makes the program much smaller and more efficient.

(iii) A set of potential tags must be added to every word in the Lexicon: this can be done by running WORDTAG over the untagged Lexicon, and then 'manually' checking the decisions reached.

(iv) We must design an algorithm to discover, for each word in the Lexicon, a set of 'similar' words. This algorithm must find words which have very similar spellings to the 'target' word (e.g. *now* is a common 'typo' misspelling of *know*); and also, it must find words which can easily be confused because they *sound* the same (e.g. *there* vs. *their*).

(v) Using this 'similar-word-finding' algorithm, every word in the Lexicon must be assigned a set of *error-tags*: first, a set of similar words is associated with each 'target' word; then, the tags from these similar words become the error-tags of the 'target' word.

(vi) The current LOB Tagging programs were originally written to be run on University Mainframe computers, and we paid scant attention to questions of speed and efficiency; the programs contain a number of routines which, in the light of experience, are clearly not necessary in a spelling-checker (for example, the programs are designed to collect large amounts of statistical feedback; but once a satisfactory success level is achieved, this will not be needed). Everything but the essential 'core' of the analysis can be cut out, and the suite of programs can be combined into one single program, performing the analysis in a single pass. In effect, then, the LOB CL Analysis suite must be completely rewritten, to make it much faster and more efficient.

C.5 Checking grammar and style

So far, we have only discussed *spelling* errors which can be detected by grammatical analysis. In essence, such errors are detected because the misspelling causes an incongruity in the grammatical structure of the sentence; the position of the incongruity is marked by the warning message "ERROR?", which is to be interpreted as a spelling-error.

In general, though, *any* striking grammatical incongruity is liable to be marked by the warning message "ERROR?"; and although up till now we have assumed this indicates a spelling-error, this is not necessarily so: the user of the system must be aware that this warning may be triggered by a *grammatical* infelicity (for example, if a word is not just misspelt, but accidentally missed out altogether,

then if an 'ungrammatical' sentence results, an "ERROR?" warning will be triggered.

Rather more insidious and problematic than blatantly 'incorrect' grammar is the use of obscure and unnecessarily complex grammar, which can make documents unintelligible; this is a problem of *style* rather than simple grammaticality. Fortunately, the spelling-check program is readily adapted to check 'grammatical style' as well. Currently, the tagging programs choose the 'best' analysis by comparing the *relative likelihoods* of alternative analyses. A fairly simple modification would allow us to elicit an *absolute likelihood* figure for the 'best' analysis of each sentence (normalized to fall within the range 0 to 1). This figure amounts to a measure of *'grammatical deviance'*: sentences with a normalized absolute likelihood of nearly 1 have simple, 'ordinary' grammatical structure, while sentences with a normalized absolute likelihood near zero are highly 'deviant'.

Thus, the 'Automatic Text-Checker' will not only mark out blatant errors in spelling and grammar, but it will also grade sentences along a sliding scale according to 'grammatical deviance' (sentences which fall below an 'acceptability threshold' (chosen by the user) can even be specifically marked out). Word Processors equipped with this Automatic Text-Checker will hopefully encourage the use of Plain English in official and business documents!

D CL GRAMMAR IN SPEECH SYNTHESIS AND ANALYSIS

Converting between written and spoken English is a trivial operation for humans, but has proven extremely difficult for computers. CL Analysis may prove a useful tool in tackling this problem.

D.1 Graphemic to phonemic transcription

It is generally agreed that an important stage in speech synthesis is the translation of ordinary written text into some phonetic form, in which each symbol corresponds to some specific sound. Some simple speech-synthesis systems have a straightforward dictionary-lookup algorithm to do this, using a dictionary which gives a single phonetic equivalent of each written word. A more refined version of this algorithm also has a 'default' rule-system to translate words not found in the dictionary, so that *any* input word can be assigned a

phonetic transcription; this is analogous to the default routines in WORDTAG, which ensure that *any* input word is assigned a set of potential tags.

Unfortunately, some words turn out to be 'ambiguous', in that they can have varying pronunciation and/or stress, depending on their grammatical function, e.g.:

"John wanted to *read* the paper"
vs.
"Has he *read* it yet?"

"She seems to *reject* all my advances"
vs.
"I put the *reject* in the dustbin"

A grammatical tagging algorithm could be used to disambiguate such examples. The great advantage of CL Analysis is that we do not have to analyse a whole sentence, but only the immediate context; a 'Grapheme-to-Phoneme Transcription' program could 'turn on' the CL tagging and disambiguation algorithm whenever such an ambiguity arose, but keep it 'turned off' the rest of the time.

However, if we wish to include sentence intonation in our phonetic transcription, then grammatical analysis of the whole sentence clearly *is* required. For this, the CL Grammatical Parser to be described in Section E would be a useful tool.

D.2 Speech analysis in terms of constituent-likelihood analysis

CL Analysis plays an even more important part if we view the whole process of speech analysis, from sound to written form, in terms of 'tagging', that is, assignment of 'labels' to 'constituents'.

The first step in speech analysis is to convert 'raw' sound into a digital form which can be readily manipulated by digital computer (the Lancaster University Linguistics Department has an ACT Sirius 1 computer which has this facility). Next, this 'digital sound' must be converted into a sequence of phonetic symbols; and then, the sequence of phonetic symbols must be converted into normal written English. However, these two conversion processes are far from trivial. The 'units' of speech sound (phones) are of variable length (e.g. a vowel sound is longer than a plosive), and also, the 'same'

utterance recorded several times will yield a slightly different digital recording each time. This leads to uncertainty and ambiguity in the phonetic transcription of a digital recording of an utterance. Moreover, even if we could be sure of choosing the correct phonetic transcription, converting this to normal written English is still a big problem. Again, the 'units' are of variable length (unlike written English, spoken utterances generally have nothing like a space at every word-boundary). Also, there is another level of ambiguity, e.g. *make up* and *may cup* may both be valid interpretations of a given phonetic transcription.

This second level of ambiguity can only be resolved by grammatical analysis: the 'best' interpretation must be chosen on the basis of contextual compatibility. Clearly, this problem can be tackled in terms of CL Analysis:

(i) given a phonetic transcription of an utterance, assign a set of potential written English interpretations; then

(ii) assign a likelihood to each potential 'labelling' or grapheme-string, using a Likelihood Function (L<g>) which measures the internal grammatical consistency of the grapheme-string in terms of the contextual compatibilities of the constituent graphemes (so that grapheme-strings which constitute 'grammatical' sentences are assigned higher likelihoods than grapheme-strings which involve grammatical inconsistencies).

In fact, the first level of ambiguity, encountered when moving from digital recording to phonetic transcription, can also be dealt with in terms of CL Analysis:

(i) given a digital recording of an utterance, assign a set of potential phonetic transcriptions; then

(ii) assign a likelihood to each potential 'labelling' or phone-symbol-string using a Likelihood Function (L<p>) which measures the internal lexical consistency of the phone-symbol-string in terms of the contextual compatibilities of the constituent phone-symbols (so that phone-symbol-strings which constitute a sequence of valid lexical items (words) are assigned higher likelihoods than phone-symbol-strings which involve non-existent 'words').

A great advantage of this approach is that it allows both levels of disambiguation to be combined in an integrated analysis algorithm:

we can calculate the overall likelihood that a particular grapheme-string is the correct interpretation of a given digital recording, simply by multiplying L<p> by L<g>. This is useful for two reasons:

(i) the 'best' phonetic transcription of a digital recording may turn out to be grammatically inconsistent, while a 'less likely' phonetic transcription (rejected during the first stage of disambiguation) might have had some graphemic interpretation which is grammatically 'acceptable'. In other words, if the two stages of disambiguation are separate, we may eliminate some of our options 'too early'; by disambiguating only on the basis of 'overall' likelihood, we are effectively hedging our bets until *all* relevant factors have been taken into account.

(ii) The division of the problem of speech analysis into two main subtasks, as described above, is in fact contentious; for example, many linguists would say that the transition from phonetic transcription to phonemic transcription is an important separate subtask. However, if the aim of the CL Analysis is to assign some 'overall' Likelihood figure to any given mapping between digital recording and grapheme-string, then it does not really matter how many subtasks this 'overall' process is divided into: the 'overall' Likelihood is simply a product of a number of factors, one for each subtask.

D.3 A CL Grammar of spoken English

The CL Grammar used by the LOB Corpus Tagging program suite is based on statistics derived from written English texts (initially, texts from the Brown Corpus). In a sense, we can say that the CL Grammar was 'extracted' from these texts: although we decided upon the tagset (using 'intuitive' knowledge of the important grammatical word-classes of English), the texts provided the frequency statistics which constituted the 'rules' of syntactic patterning.

The grammar of spoken English is statistically different from the grammar of written English (for example, written English tends to include more lengthy, complex sentences); the CL approach allows us to quantify these differences systematically. First, a Corpus of spoken English is needed (the London-Lund Corpus of Spoken English could be used, or alternatively, if a sufficiently general and robust speech-analysis program could be devised, we might even compile a

new Corpus using this program (the actual compilation of this new Corpus would serve as a very thorough 'test' of such a program!). This Corpus must then be grammatically analyzed, by running the present LOB Grammatical Tagging programs over it, and then 'manually' correcting the errors (many of which will be due to the imposition of a Written English Grammar over Spoken English). From the analyzed Corpus, we can then 'extract' a CL Grammar of Spoken English, by gathering the relevant frequency statistics. The differences between this CL Grammar of Spoken English and the LOB CL Grammar of Written English will be reflected in the differences in Tag-Pair Bond function values for certain tag pairs, and also in other related statistical differences such as the average Absolute Likelihood assigned to a sentence.

Thus, a speech-analysis program can be used in the compilation of a Corpus of Spoken English, from which we can 'extract' a CL Grammar of Spoken English; and this grammar will then be very useful to researchers in speech analysis and synthesis, since it is specifically geared to spoken English. Potentially, the two fields of CL Grammar and Speech Synthesis and Analysis have much to offer each other.

E CL GRAMMATICAL PARSER

The current LOB Corpus Grammatical Tagging programs assign a grammatical tag to each word in a text, showing its grammatical function; but 'higher-level' constituents are not analysed. To do this, we need a *grammatical parser*; and it turns out that it should be possible to perform a grammatical parse of the LOB Corpus using algorithms very similar to those of the present tagging-suite.

E.1 Tags and hypertags

In general, each tag in the LOB Tagset can only appear in certain syntactic (syntagmatic) positions, for example:
AT (article) comes at the start of a Noun Phrase;
IN (preposition) comes at the start of a Prepositional Phrase;
CS (subordinating conjunction) comes at the start of a Subordinate Clause;
. (full stop) comes at the end of a Sentence;

NN (singular common noun) comes

(i) at the start of a Noun Phrase *or*

(ii) at the end of a Noun Phrase *or*

(iii) within a Noun Phrase *or*

(iv) as a Noun Phrase in its own right (i.e. start *and* end of a Noun Phrase)

These syntactic positions within higher-level constituents can be symbolized by 'higher-level tags' or *hypertags*. By analogy with the present WORDLIST (a list of words and their possible tags), we could construct a TAGLIST of tags and their possible hypertags, with entries such as

tag	possible hypertags
AT	[N
.	S]
IN	[P
CS	[F
NN	N] N [N] [N@
PP$	[N
VB	[V] V] V@ [V@

etc.

(NB [V]does *not* include the object Noun Phrase, but only Verb-constituents; however, [N] *does* include subordinate prepositional phrases, etc.)

As with tags in the WORDLIST, hypertags are ordered, with @ and % markers for rare syntagmatic functions.

E.2 Hypertag-assignment

A program analogous to WORDTAG could give each tag in a sentence its appropriate hypertags, as given by the TAGLIST (this program would in fact be much simpler than WORDTAG, as there are only 134 tags in the LOB Tagset, instead of an open-ended set of possible words).

Sometimes, the hypertag(s) required is(/are) indicated better by a particular *combination* of tags, rather than by the tags taken individually. For example, IN (preposition) is 'hypertagged' [P (open prepositional phrase), and WDT (WH-determiner) is 'hypertagged' [F[N (open subordinate clause *and* open noun phrase); but the combined

tag-pair IN WDT must be 'hypertagged' [F[P [N (this is for clauses beginning "of which...", "for what...", etc.). These 'special-case' tag-pairs and their corresponding hypertag-pairs must be listed in a TAG-PAIR-LIST, analogous to the current IDIOMLIST of exceptional word combinations; a program analogous to IDIOMTAG could 'overwrite' hypertags assigned by simple TAGLIST-lookup whenever a tag-pair matches an entry in this TAG-PAIR-LIST.

Since these two 'hypertag-assignment' programs will be considerably simpler than WORDTAG and IDIOMTAG, it will be practicable to combine them into a single program: each tag-pair in a text is first looked up in the TAG-PAIR-LIST; but if no match is found, then hypertags are assigned to the tags individually, according to the TAGLIST. This unified hypertag-assignment program will be much more efficient, since unnecessary lookups are avoided, and all hypertags are assigned in a single pass.

E.3 Hypertag-disambiguation

Each record has now been assigned a set of potential hypertags. Next, a program analogous to CHAINPROBS must assign a relative likelihood figure to each hypertag in a record, using a *hypertag likelihood function* very similar to the Tag Likelihood Function described in Section B. We can then choose a single 'best' sequence of hypertags. For example, the sentence "As I was eating my lunch I decided to get a cup of coffee" would be hypertagged as follows:

WORD	TAG	HYPERTAG
----------		[S
as	CS	[F
I	PP1A	[N]
was	BEDZ	[V
eating	VBG	V]
my	PP$	[N
lunch	NN	N]
I	PP1A	[N]
decided	VBD	[V]
to	TO	[T
get	VB	[V]
a	AT	[N
cup	NN	N
of	IN	[P
coffee	NN	[N]
.	.	S]

E.3 Building a syntactic parse tree

Tags have now been grouped into higher-level constituents (N, V, S, etc.); but there are still some 'unmatched brackets'. This is because certain tags specifically mark the *start* of a higher-level constituent (e.g. CS-[F; IN-[P; AT-[N), but often there is no such corresponding 'end-of-phrase word'.

What we need now is a program which can insert extra closing brackets where needed. One way to find out where to add these brackets is to try to convert the labelled bracketing into a tree data-structure, following simple 'conversion rules':

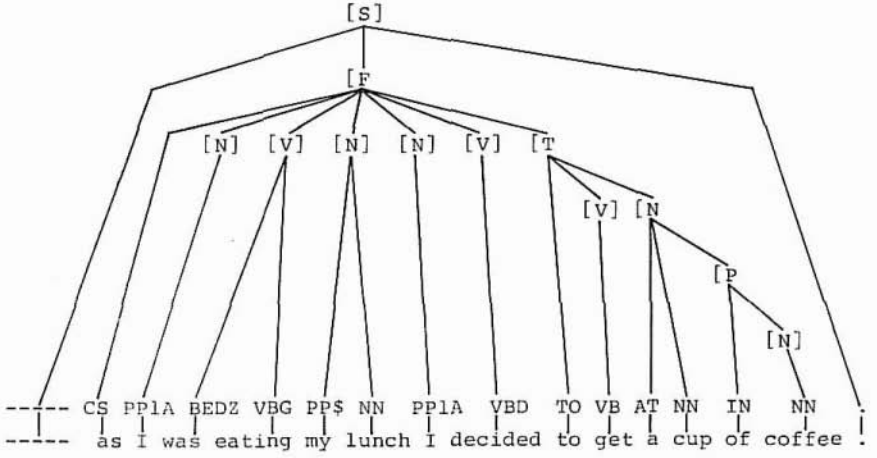
(i) X [Y means "Y is the daughter of X"

(ii) X] Y means "X is the daughter of Y"

(iii) X] [Y means "Y is the right sister of X"

(iv) X ... X is represented by a single node X *if* both Xs are at same 'level' of nested bracketing and they are not sisters . there is no][interposing between the two Xs *at the same level* as the Xs. Note that X ... Y (where X and Y are different, and X is at the same level as Y but not a sister) is *invalid*, since it requires a single node to be tagged both X and Y; this is an indication that some phrase-boundary (labelled bracket(s)) is missing.

Using such rules, we can build the following tree:



E.4 Inserting missing closing brackets

The 'conversion rules' carry on adding daughters to a node until that node's closing bracket is found; so, if the closing bracket is missing, the node will continue to have daughters attached to it until the sentence-end is reached. This means that the rightmost daughters of an 'unclosed' node are *suspect*: each non-leaf node in the tree should have at least one daughter (the first or left-most daughter), but the nodes further to the right could well be not daughters but right-hand sisters (or even 'aunts'!) of the 'unclosed' node.

An example of this is the unclosed [F node (marking a subordinate clause) in the tree above; its daughters are apparently

[N] [V] [N] [N] [V] [T

Clearly this is wrong - this sequence of daughter-constituents could not be a valid subordinate clause. The reason for this error in the tree is that the missing closing bracket F] should be inserted between *lunch* and *I*, so that the subordinate clause becomes

[N] [V] [N]

and the remaining 'daugthers' become *sisters* of [F]. However, the tree-building algorithm does not know this, so it carries on adding daughters to the unclosed [F node instead of the root [S].

Nevertheless, despite being 'lopsided', the tree built in this way is still useful. The tree shows us where missing closing brackets *might* be inserted: for example, the tree becomes well-formed only if the F] is inserted after a daughter of [F.

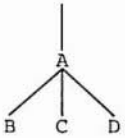
In general, an unclosed node [X with n daughters in the original tree can be 'closed' in n different ways, leading to n different parse-subtrees. So, if an 'unclosed' tree such as the one shown above has q 'unclosed' hypertag-nodes [H<1>, [H<2>, [H<3>, ..., [H<q>, where

[H<1> has n<1> daughters
[H<2> has n<2> daughters
[H<3> has n<3> daughters
⋮
[H<q> has n<q> daughters

then there are (n<1>*n<2>*n<3>*...*n<q>) *potential parse-trees*.

E.5 Choosing the 'best' parse-tree

The final stage in parsing is a program which, starting from an 'unclosed' tree such as the one above, generates all possible parse-trees and compares the likelihood of each (note the analogy with CHAINPROBS: this program effectively generates all possible tag-sequences and compares their likelihoods; the difference is that now we are dealing with trees rather than simple strings). To do this, we must be able to associate a likelihood with a potential parse-tree; this is done using a *hypertag-node likelihood function* Lhn which assigns any given node a likelihood figure dependent on its daughter nodes and their likelihoods in turn. If a node A has daughters B, C, D:



then at A we must store the likelihood that BCD is a 'valid' A (the *constituent likelihood* Lc(A,BCD)), multiplied by the hypertag-node likelihoods of B, C, and D in turn:

Lhn(A) = Lc(A,BCD) * Lhn(B) * Lhn(C) * Lhn(D)

This recursive definition allows us to calculate a likelihood figure for the root [S] node which takes into account all nodes and subtrees in the parse-tree.

E.6 The phrase dictionary

Values of the Constituent-Likelihood function Lc are stored in a *Phrase Dictionary*, which states, for each of the higher-level constituents (N, S, V, P, etc.), the set of possible 'daughter-constituent-sequences', along with the relative likelihood of each possible sequence. For example, the Phrase Dictionary will tell us that, in a subordinate clause (hypertagged [F]), the following daughter-constituent-sequences are very likely:

CS [N] [V] [N]

CS [N] [V]

; the following sequences are less likely, but still possible:

CS [N] [V] [P]

CS [N] [V] [T]

; but the following sequences are *very* unlikely:

CS [N]

CS [V]

CS [N] [N]

Any daughter-sequences not found in the Phrase Dictionary get a very low default probability (just above zero); in this way, we are ensured of *some* analysis for *any* sentence (the analogy in CHAINPROBS is that the Tag-Pair Bond function always has a value greater than zero, to ensure that no potential tag is ever assigned a zero likelihood; see Section B.5).

E.7 A parse in three passes

To summarize, the CL Grammatical Parser outlined above will build a syntactic parse-tree in three passes. First, every tag in a text is assigned a set of potential hypertaggings, using a *tag-pair-list* and *taglist*. Secondly, the set of hypertags at each tag is disambiguated, by eliminating all but the likeliest hypertag-sequence; this is done using a *hypertag-likelihood function* very similar to the tag-likelihood function currently used by CHAINPROBS. Thirdly, this 'disambiguated' hypertag-sequence is converted into a set of potential parse-trees, where each potential parse-tree has the missing closing brackets inserted differently; a *hypertag-node likelihood function* is used to compare likelihoods of competing potential parse-trees.

In the final output, it will probably be useful to include not only the single 'best' parse-tree, but also a number of 'runners-up' (say three), in case the 'best' parse is found to be incorrect in postediting. This can be done quite easily, if we adopt an output format similar to that shown in Section E.3: there are columns for *word*, *tag*, and *hypertag*; and in addition, we need three more columns to show the three 'likeliest' combinations of inserted closing brackets. For example, the final output of the 'parse' of our earlier example sentence might be:

WORD	TAG	HYPERTAG	THREE LIKELIEST PARSES		
			59%	39%	2%
----------		[S			
as	CS	[F			
I	PP1A	[N]			
was	BEDZ	[V			
eating	VBG	V]			
my	PP$	[N			
lunch	NN	N]	F]	F]	F]
I	PP1A	[N]			
decided	VBD	[V]			
to	TO	[T			
get	VB	[V]			
a	AT	[N			
cup	NN	N]		N]	N]T]
of	IN	[P			
coffee	NN	[N]	P]N]T]	P]T]	P]
.	.	S]			

This representation may seem difficult to understand at first; but hopefully posteditors will soon get to grips with it. The great advantage is of course the economy of space: to show three potential analyses, we do not need three complete trees.

E.8 Residual problems

Finally, it must be remembered that, of course, not all sentences will be as straightforward as the example above! There are many problems not touched upon (e.g. when the phrase-boundary is not explicitly marked, as in "I gave the baby milk to drink"); but then, *any* approach to syntactic parsing will encounter difficulties with these and other stumbling blocks. The success rate of CHAINPROBS turned out to be much higher than we expected; the lesson to be learnt was that in the 'real' language found in a corpus, very few 'pathological cases' actually turn up! Therefore, we have every hope that the CL Grammatical Parser will also be very successful.

F OTHER APPLICATIONS OF CL GRAMMAR

As explained in Section A.3, CL Grammar is generally applicable to many different forms of linguistic analysis. So far we have not explored all the possibilities: for example, CL Analysis may also be useful in formal semantic analysis. Other applications will doubtless suggest themselves as our research continues.

In general, we hope we have shown that *statistical, probabilistic* methods of analysis *do* have a place in linguistics, and specifically in the field of syntax. Furthermore, statistical analysis should *not* be seen simply as a 'heuristic' to fall back on when all else fails; CL analysis is entirely based on probabilities, and the Tagged LOB Corpus will be overwhelming evidence that this approach works.

REFERENCES

Atwell, Eric Steven. 1982. 'LOB Corpus Tagging Project: Manual Pre-edit Handbook'. Department of Linguistics and Modern English Language and Department of Computer Studies, University of Lancaster.

Atwell, Eric Steven. 1982. 'LOB Corpus Tagging Project: Manual Post-edit Handbook (A mini-grammar of LOB Corpus English, examining the types of error commonly made during automatic (computational) analysis of ordinary written English)'. Department of Linguistics and Modern English Language and Department of Computer Studies, University of Lancaster.

Francis, W. Nelson and Henry Kučera. 1964 (rev. eds. 1971 and 1979). *Manual of Information to Accompany a Standard Sample of Present-Day Edited American English, for Use with Digital Computers.* Department of Linguistics, Brown University.

Garside, Roger and Geoffrey N. Leech. 1982. 'Grammatical Tagging of the LOB Corpus: General Survey'. In Stig Johansson, ed. *Computer Corpora in English Language Research.* Norwegian Computing Centre for the Humanities, Bergen.

Greene, Barbara and Gerald Rubin. 1971. *Automatic Grammatical Tagging of English.* Department of Linguistics, Brown University.

Johansson, Stig, Leech, Geoffrey N. and Helen Goodluck. 1978. *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers.* Department of English, University of Oslo.

Johansson, Stig and Mette-Cathrine Jahr. 1982. 'Grammatical Tagging of the LOB Corpus: Predicting Word Class from Word Endings'. In Stig Johansson, ed. *Computer Corpora in English Language Research.* Norwegian Computing Centre for the Humanities, Bergen. 118-146.

Leech, Geoffrey N., Garside, Roger and Eric Steven Atwell. 1983. 'The Automatic Grammatical Tagging of the LOB Corpus' (pp. 13-33 in this issue of *ICAME News*).

Marshall, Ian. 1982. 'Choice of Grammatical Word-Class without Global Syntactic Analysis for Tagging Words in the LOB Corpus'. Department of Computer Studies, University of Lancaster.

Peterson, James. 1980. 'Computer Programs for Detecting and Correcting Spelling Errors'. In *Communications of the Association for Computing Machinery*, 23, 12. 676-87.

```
ATWELL1 my farther was very crawl . he bald at me if I dud anything wrong , and
          ERROR?          ERROR?      ERROR?          ERROR?

ATWELL2 sometimes he would hot and bit me , until I was so week and miserable
                          ERROR?  ERROR?                    ERROR?

ATWELL3 that I wanted to due . finally , won day , I decided to got my won back
                        ERROR?                                  ERROR?

ATWELL4 on him : *' I 'll mike him pay ; he will n't get away with this ! **' I stole
                          ERROR?

ATWELL5 a meat clever , and I maid several dense in his hid with it ! it must
            ERROR?          ERROR?      ERROR?          ERROR?

ATWELL6 have hurt a lit ! son *' the gruesome tame of Eroc Attwell **' appeared
                    ERROR?  ERROR?                ERROR?

ATWELL7 in all the papers . perhaps my friends would learnt to spell my name
                                                  ERROR?

ATWELL8 correctly at last !


END OF LISTING OF FILE :ENA001.ERIC(1,*,1).EXAMPLE6(5) FOR USER :ENA001 AT 1983/03/18__
```

MATERIAL AVAILABLE FROM BERGEN

The following material is currently available on tape from Bergen through the International Computer Archive of Modern English (ICAME):

Brown Corpus, text format I (without grammatical tagging): A revised version of the Brown Corpus with upper- and lower-case letters and other features which reduce the need for special codes and make the material more easily readable. A number of errors found during the tagging of the corpus have been corrected. Typographical information is preserved; the same line division is used as in the original version from Brown University except that words at the end of the line are never divided.

Brown Corpus, text format II (without grammatical tagging): This version is identical to text format I, but typographical information is reduced and the line division is new.

Brown Corpus, KWIC concordance (also on microfiche): A complete concordance for all the words in the corpus, including word statistics showing the distribution in text samples and genre categories. The microfiche set includes the complete text of the corpus.

LOB Corpus, text: The LOB Corpus is a British English counterpart of the Brown Corpus. It contains approximately a million words of printed text (500 text samples of about 2,000 words).

LOB Corpus, KWIC concordance (also on microfiche): A complete concordance for all the words in the corpus. It includes word statistics for both the LOB Corpus and the Brown Corpus, showing the distribution in text samples and genre categories for both corpora. The text of the LOB Corpus is not available on microfiche.

London-Lund Corpus, text: The London-Lund Corpus contains samples of educated spoken English, in orthographic transcription with detailed prosodic marking. It consists of 87 'texts', each of some 5,000 running words. The text categories represented are spontaneous conversation, spontaneous commentary, spontaneous and prepared oration, etc.

London-Lund Corpus, KWIC concordance I: A complete concordance for the 34 texts representing spontaneous, surreptitiously recorded

conversation (text categories 1-3), made available both in computerised and printed form (J. Svartvik and R. Quirk (eds.) *A Corpus of English Conversation*, Lund Studies in English 56, Lund: C.W.K. Gleerup, 1980).

London-Lund Corpus, KWIC concordance II: A complete concordance for the remaining 53 texts of the London-Lund Corpus (text categories 4-12).

The material has been described in greater detail in previous issues of *ICAME News*. Prices and technical specifications are given on the order forms which accompany this newsletter. *Note that the concordances are now also available on higher-density tapes at a lower price.*

A printed manual accompanies tapes of the LOB Corpus. Printed manuals for the Brown Corpus cannot be obtained from Bergen. Some information on the London-Lund Corpus is distributed together with copies of the text and the KWIC concordances for the corpus. Users of the London-Lund material are, however, recommended to order the recent book by Svartvik *et al.*, *Survey of Spoken English: Report on Research 1975-81*, Lund Studies in English 63, Lund: C.W.K. Gleerup, 1982. The grammatically tagged version of the Brown Corpus can only be obtained from: Henry Kučera, TEXT RESEARCH, 196 Bowen Street, Providence, R.I. 02906, U.S.A. The Syntax Data Corpus, which consists of part of the Brown Corpus, with detailed syntactic tagging, can only be obtained from: Alvar Ellegård, Department of English, University of Gothenburg, Lundgrensgatan 7, S-412 56 Göteborg, Sweden.

BIBLIOGRAPHICAL SURVEY

One of the main aims in establishing ICAME was 'to make possible and encourage the coordination of research effort and avoid duplication of research'. Since the start in 1977, material has been distributed to a range of research institutions in many countries, and it is becoming increasingly difficult to survey how it has been, and is being, used. Users are encouraged to *send in information on publications, reports, and work in progress* related to the material. There is no need to report on work contained in the bibliography in *Computer Corpora in English Language Research* (ed. by Stig Johansson, publ. by the Norwegian Computing Centre for the Humanities, Bergen 1982). An updated bibliography will be included in a later issue of *ICAME News*.

CONDITIONS ON THE USE OF ICAME CORPUS MATERIAL

The primary purposes of the International Computer Archive of Modern English (ICAME) are:

(a) collecting and distributing information on (i) English language material available for computer processing; and (ii) linguistic research completed or in progress on this material;

(b) compiling an archive of corpora to be located at the University of Bergen, from where copies of the material can be obtained at cost.

The following conditions govern the use of corpus material distributed through ICAME:

1 No copies of corpora, or parts of corpora, are to be distributed under any circumstances without the written permission of ICAME.

2 Print-outs of corpora, or parts thereof, are to used for bona fide research of a non-profit mature. Holders of copies of corpora may not reproduce any texts, or parts of texts, for any purpose other than scholarly research without getting the written permission of the individual copyright holders, as listed in the manual or record sheet accompanying the corpus in question. (For material where there is no known copyright holder, the person/s/ who originally prepared the material in computerized form will be regarded as the copyright holder/s/.)

3 Commercial publishers and other non-academic organizations wishing to make use of part or all of a corpus or a print-out thereof must obtain permission from all the individual copyright holders involved.

4 The person/s/ who originally prepared the material in computerized form must be acknowledged in every subsequent use of it.

EDITORIAL NOTE

Further ICAME newsletters will appear irregularly and will, for the time being be distributed free of charge. The Editor is grateful for any information or documentation which is relevant to the field of concern of ICAME.

ICAME NEWS is published by the Norwegian Computing Centre
for the Humanities (NAVFs EDB-senter for humanistisk forskning)
Address: Harald Hårfagresgate 31, P.O. 53, 5014 Bergen-University, Norway