I. INTRODUCTION

1.1. Approaches to Parsing

Parsing can be detined as the grammatical analysis oftext. A parser is a computer program which can carry out this analysis automatically on an input provided inmachine-readable form. Traditionally. parsers were desipned to accept as input a complete well-formed sentence and to produce as output a complete parse-tree for it if the sentence conformed to the parser's grammar. and to produce nothing at all if the sentence did not. There are two main disadvantages to this approach. Firstly. the input sentence may be structurally correct but still not conform to the prammar which the parser is using. Secondly. the input may be incorrect but still idiomatic. In both cases it desirable to be able to produce an output. As a result. two approaches to parsing have developed in recent years. *robust* parsing and *shallow* parsing. A robust parser attempts to produce an output however ill-formed the input. A shallow parser only attempts to carry out certain forms of analysis; typically. shallow parsers avoid problems of structural ambipuity caused by phenomena such a prepositional phrase attachment and instead concentrate on smaller regular constructs such as verb groups. noun phrases and prepositional phrases. Another development in parsing has been the adoption of part-of-speech taggers such as the Brill Tagger (1992) and the Lancaster CLAWS system (Garside. Leech and Sampson. 1987). These are programs which allow a part-of-speech such as noun or verb to be assigned quite reliably to each token in the input. The resultinp output can then be used as input to the parser. This technique considerably reduces the difficulty of the parsing task since candidate analyses of sentence constituents which are not compatible with the tags assigned by the tagger can be eliminated at a very early stage. This article describes a particular approach to robust shallow parsing. The method uses a part-of-speech tagging phase followed by multiple passes over the data during each of which some structural analysis is carried out.

1.2. Parser Evaluation

When considering the application of two different parsers to the same input. it is desirable to be able to make direct comparisons of performance. Until recently. the dominant approach to evaluation was to parse a set of test sentences and then to analyse the resulting output by hand (Carroll. Briscoe and Sanfilippo. 1998). However. this is very time consuming to do. In addition. whenever an incremental improvement to the parser or grammar is made. the output must be re-analysed. An alternative is to use a parsed corpus of test sentences. In such a corpus each sentence has already been assigned a reference parse tree by one or more linguistic

experts. During evaluation. the parser output for a sentence is conipared to its reference parse tree in the corpus. This approach to evaluation has the advantage that it is automatic once the basic software for carrying out the comparisons has been developed. On the otlier hand. it is difficult to assign reference parses in a manner which is entirely theory-neutral. A parser and its grammar are likely to be based on a grammatical theory which is different from that used for the parsed corpus. As a result. comparison of a candidate parse with a reierence parse can turn out to be a complex process. Nevertheless. in the evaluation of our parser. the reference parse tree method was applied. using a small treebank we developed for the purpose.

I.3. The Application Domain

Parsing can be applied to any kind of texts but the best results are likely to be obtained when all input to be analysed is from a restricted domain. Our work is based on the analysis of technical software nianuals. in particular the *Lotus Ami* Pro *for Windows User's Guide Release Three* (Ami Pro. 1993). There are several reasons for this. Firstly. there is a ready market for the application of language teclinology to software manuals. For example. most software is supplied witli an on-line help system combining a large body oftechnical text with a retrieval enpine for extracting parts which are relevant to a particular user query. There is considerable room for improvement in such systems.

Secondly. manuals have a clear purpose: to describe the procedures involved using the software to which they refer. A manual is effectively describing a complete. self-contained world. Such a world can also be modelled using scriptal datastructures which can then be applied to important tasks such as question answering. We are specifically interested in the automatic transformation of information expressed in textual form into such structures – a task which necessarily involves parsing.

Thirdly. a manual is typically available in a number of different languages. This is because computer software and its attendant documentation is usually localised to enable it to be marketed in different countries and cultures. Software nianuals can therefore readily be used to create multilingual resources for studying the application of linguistic techniques to different lanpuages. and for evaluating the results. For example. in recent work a parallel English-Japanese test collection for information retrieval was developed (Sutcliffe and Kurohashi. 3000). based on the Japanese version of the Ami Pro manual (Ami Pro. 1994). In another project. the approach to parsing English text described here was applied to Japanese (Sutcliffe and Nashimoto. 1999).

Fourthly. technical manual text is regular though complex. This makes parsing it a tractable task which allows engineering aspects such as grammar development and refinenient techniques to be studied without encountering insoluble instances of ambiguity. Fifthly. we have worked with technical manual texts already in several projects. The first was an attenipt

to produce a concept-based information retrieval system working with the Unix *Man* files (Sutcliffe. 1991). After this. the SIFT project (Selecting Information from Text) aimed to develop a text retrieval system based on Lotus Ami Pro (Hyland. Koch. Sutcliffe and Vossen. 1996). The text of the manual was scanned automatically and converted into an internal representation which attenipted to capture its meaning. When a query was input. it was also scanned and hence niatched with the representation of the manual in order to determine possible answers. Finally. the conference IPSM'95 (Industrial Parsing of Software Manuals) was concerned witli parser evaluation. Eight teams of computational linguists from seven countries applied their parsers to a set of 600 test sentences extracted from three technical manuals (Sutcliffe. Koch and McElligott. 1996: Sutcliffe. 1998). The texts were the Lotus manual mentioned earlier. the *Dynix Automated Library Systems Searching Manual* (Dynix. 1991). and the *Trados Translator's Workbench for Windows User's Guide* (Trados. 1995).

1.4. Structure of the Article

The remainder of this article is organised as follows. We first describe the robust layered parsingalgorithm which we developed following experiences with the IPSM project. Secondly. tlie characteristics of'the technical manual domain in which we are working are described. Thirdly. the approach taken in developing the grammar for the parser is described. Fourthly. the method adopted in creating a parsed test collection is presented. Fii'thly. we report on tlie evaluation procedure used to measure the performance of the parser when applied to the test collection. Finally. conclusions are drawn. based on the results ofthe work undertaiien.

II. ROBUST LAYERED PARSING

II.1. Motivation

In the IPSM parser evaluation study (Sutcliffe. Koch and McElligott. 1996). we conipeted using the excellent Link Parser of Sleator and Temperley (1991). This is a highly efficient parser which is supplied with a grammar of very wide coverage. However. we encountered a number of serious problems not only with this parser but with the entire approach on which it and many other parsers are based:

- It proved very difficult to modify the grammar in order to produce a slightly different analysis for a particular construction without affecting the analysis of'other constructions.

- It was very hard to predict the coverage of the grammar by inspecting its rules because they are all inter-related.
- While analysis was generally very fast. the parser could go into a loop unexpectedly when presented with an input which contained several instances ofstructural ambiguity. making it highly ambiguous. The most spectacular example of this phenomenon was: 'Displays the records that have a specific word or words in the TITLE. CONTENTS. SUBJECT. or SERIES fields of the BIB record. depending on which fields have been included in each index.'. The Link Parser found 4.424 parses of this utterance. a process which took 4.206 seconds (Sutcliffe and McElligott. 1996. p95).
- The parser often produced many different analyses for a given input. even though that input was not really structurally ambiguous.

Our objective. therefore was to develop aparser which would address some ofthese difficulties. In particular. it was to have the following characteristics:

- **Efficient** – capable of producing output quickly at all times.
- **Robust** – able to produce some analysis for any input whether grammatical or not.
- **Shallow** – only attempting to recognise certain constructs while ignoring others.
- **Layered** – carrying out analysis in a series of stages which would not interfere with each other.

The result of tning to meet these objectives was the Robust Layered Parser.

11.2. Parsing Algorithm

The algorithm can be summarised as follows:

- The input to be anal! sed is first tagged forpart-of-speech using a separate system such as the Brill Tagger (1 992);
- The resulting string is tlien tokenised. yielding a list of tokens and separated punctuation. each with a part-of-speech attached:
- The parser then scans the tagged input *n* times trying to analyse portions of it each time:
- Each scan looks for instances of one construct. Corresponding to that construct is a context-free grammar. For *ti* scans. therefore. there are *n* different grammars;

- A construct recognised in a particular scan must be defined in terms of some combination of constructs recognised in previous scans (i.e. non-terminal symbols), together with zero or more tagged words (i.e. terminal symbols);
- If an instance of a construct is found in the input to a scan, the instance is replaced by the parse tree for the construct;
- The output of parsing is the output produced by the last scan.

```
Input:
save the file under a new name with the same extension

PART-OF-SPEECH Analysis:
[save/'VB', the/'DT', file/'NN', under/'IN', a/'DT', new/'JJ', name/'NN',
with/'IN', the/'DT', same/'JJ', extension/'NN']

NOUN PHASE Analysis:
[save/'VB', [cnp, [np, [det, the/'DT'], [], [], [cn, [ns, [n,
file/'NN']]]]], under/'IN , [cnp, [np, [det, a/'DT'], [], [adj,
new/'JJ'], [cn, [ns, [n, name/'NN']]]]], with/'IN', [cnp, [np, [det,
the/'DT'], [], [adj, same/'JJ'], [cn, [ns, [n, extension/'NN']]]]]]

PREPOSITIONAL PHRASE Analysis:
[save/'VB', [cnp, [np, [det, the/'DT'], [], [], [cn, [ns, [n,
file/'NN']]]]], [cpp, [pp, [cp, [p, under,''IN']], [cnp, [np, [det,
a/'DT'], [], [adj, new/'JJ'], [cn, [ns, [n, name/'NN']]]]]]], [cpp, [pp,
[cp, [p, with/'IN']], [cnp, [np, [det, the/'DT'], [], [adj, same/'JJ'],
[cn, [ns, [n, extension/'NN'11111111

VERB GROUP Analysis:
[[cvg, vg, [v, save/'VB']], [cnp, [np, [det, the/'DT'], [], [], [cn, [ns,
[n, file/'NN']]]]], [cpp, [pp, [cp, [p, under/'IN']], [cnp, [np, [det,
a/'DT'], [], [adj, new/'JJ'], [cn, [ns, [n, name/'NN']111111, [cpp, [pp,
[cp, [p, with/'IN']], [cnp, [np, [det, the/'DT'], [], [adj, same/'JJ'],
[cn, [ns, [n, extension:'NN']]]]]]]]
```

Figure. 1 Example Output from the Kobust Parser

To understand these stages. consider the following example. Suppose the input to the parser is 'save the file under a new name with the same extension'. The output from each scan using the grammar defined for this study might be as shown in Figure 1. The first scan is Part-of-Speech Analysis. which is carried out by the Brill (1993) tagger. During this stage. a part-of-speech such as VB (verb) or NN (singular common noun) is associated with each token in the input. After this comes Noun Phrase Analysis. The tagged input is scanned looking for instances of noun phrases. the/'DT', file/'NN' is recognised as a noun phrase for which the analysis is [cnp, [np, [det, the/'DT'], [], [], [cn, [ns, [n, file/'NN']]]]], the/'DT',

file/'NN' is therefore replace by this analysis. Two othei noun phrases are recognised in this pass: a/'DI', new/'JJ', name/'NN' and the/'DT', same/'JJ', extension/'NN'.

Prepositional Phrase Analysis is now carried out. using as input the output produced from the previous scan. This now contains a mixture of terminal symbols such as under/'IN' together with parses of constructs such as [cnp, [np, [det, a/'DT'], [], [adj, new/'JJ'], [cn, [ns, [n, name/'NN']]]]]. Such parses also serve as non-terminal symbols to be recognised by future scans. In this case the construct is a Coordinated Noun Phrase denoted by the symbol cnp. During Prepositional Phrase Analysis. constructs denoted Coordinated Prepositional Phrase (cnp) are recognised. These broadly comprise a preposition (usually tagged IN) followed by a Coordinated Noun Phrase (cnp). One instance of such a sequence is under/'IN' followed by the cnp construct just discussed. The two are thus replaced by a parse of the resulting cpp. [cpp, [pp, [cp, [p, under/'IN']], [cnp, [np, [det, a/'DT'], [], [adj, new/'JJ'], [cn, [ns, [n, name/'NN']]]]]]]. A similar process is carried out on with/'IN' and [cnp, [np, [det, the/'DT'], [], [adj, same/'JJ'], [cn, [ns, [n, extension/'NN']]]]] resulting in their replacement by [cpp, [pp, [cp, [p, with/'IN']], [cnp, [np, [det, the/'DT'], [], [adj, same/'JJ'], [cn, [ns, [n, extension/'NN']]]]]]].

The final stage of processing is Verb Group Analysis. The output from the previous stage complete with its parses ior instances of cnp and cpp is scanned once more looking for verb groups. save/'VB' is duly recognised and replaced with [[cvg, vg, [v, save/'VB']]].

The output of the entire parsing process is the result produced by the final scan. in this case Verb Group Analysis. The output comprises a list oi'parse trees corresponding to constructs recognised. together with any tagged terminal symbols which were not recognised as part of any construct. The essence of this kind of parsing is that as much or as little analysis can be carried out on the input. depending on its form and on the constructs to be recognised. So. for example. if no scans carried out by the parser lind any instances in the input of the constructs they seek. the output of the entire parsing process is exactly the same as the input. At the other extreme. which corresponds to conventional parsing. all tagged terminal symbols in the input may be recognised as forming part of a single construct (such as Sentence) in the first scan. In other words. robust layered parsing allows complete analyses if this is what is required in a particular application. In fact. the version of the parser being used for Figure 1 will produce a single analysis of the entire input in certain cases. for example if the input is a solitary noun phrase such as the/'DT', same/'JJ', extension/'NN'. In the technical manual domain. inputs whichare not complete grammatical sentences occur very commonly (e.g. as heridings aiid bullet points) and so it is essential that we can analyse thein.

The output of robust parsing can take a wide variety of forms. depending on the grammar. The aniount of syntactic information returned can be as large or as small as is required to perform a particular task. For exaniple. Figure 2 shows the sanie stages of analysis applied to the same input but using another grammar. This grammar makes explicit the kind of and/or/comma

coordination which has been recognised and it includes in the construct `cn` the head noun, in the construct `cpp` the preposition and the head noun, and in the construct `cvg` the head verb. All other information such as determiners (`the`, `a`), the words indicating coordination (such as `and` or `or`), punctuation and so on are reinoved. The output is thus quite different even though the grammatical constructs being recognised are very similar.

```
Input:
save the file under a new name with the same extension

PART-OF-SPEECH Analysis:
[save/'VB', the/'DT', file/'NN', under/'IN', a/'DT', new/'JJ', name/'NN',
with/'IN', the/'DT', same/'JJ', extension/'NN']

NOUN PHASE Analysis:
[save/'VB', cnp(and, [np(cd(?, ?), cn(and, [file]))]), under/'IN',
cnp(and, [np(cd(?, ?), cn(and, [name]))]), with/'IN', cnp(and, [np(cd(?,
?), cn(and, [extension]))])]

PREPOSITIONAL PHRASE Analysis:
[save/'VB', cnp(and, [np(cd(?, ?), cn(and, [file]))]), cpp(and,
[pp(p(and, [under]), cnp(and, [np(cd(?, ?), cn(and, [name]))]))]),
cpp(and, [pp(p(and, [with]), cnpiand, [np(cd(?, ?), cn(and,
[extension]))])])]1

VERB GROUP Analysis:
[cvg(and, [save]), cnp(and, [np(cd(?, ?), cn(and, [file]))]), cppiand,
[pp(p(and, [under]), cnpiand, [np(cd(?, ?), cn(and, [name]))]))]),
cpp(and, [pp(p(and, [with]), cnpiand, [np(cd(?, ?), cn(and,
[extension]))]))]), []]
```

Figure. 2 Example of Different Output from the Same Input

Before leaving the Robust Layered Parsing algorithm, two points should be noted about it. Firstly, part-of-speech tagging is not necessary in order to apply this approach. For example, a pattern recognition task in information estraction might work with tokens directly if the patterns to be recognised were simple and the vocabulary highly constrained.

Secondly, standard language engineering processes such as advanced tokenisation, term recognition and case-frame extraction can all be combined nithin the same coinputational paradigm. Effectively, each is a diflerent scan. This makes robust parsing a very convenient way to solve practical problems.

II.3. Implementation

The parser is implemented in Quintus Prolog and runs under either Sunos or NT 4. The input to the iirst scan is a list of `Word . Tag` pairs derived from the part-of-speech tagger. The output of a scan is a list where each element is either one of the original `Word / Tag` pairs or is a structure representing a parse of some contiguous sublist of the input.

Analysis during each scan is carried out using a Definite Clause Grammar (DCG) which is conceptually separate from DCGs used in other scans. The DCG is defined to look i'or instances o f a particular construct (e.g. Coordinated Noun Group `cn`) and to replace each instance of the construct found by a parse tree for it (e.g. `[cn, [ns, [n, file/'NN']]]`). Each DCG is designed to look for an instance of its corresponding construct at the start of the current input list. Each scan is thus implemented via a driving routine which first tries to apply the DCG assigned to that scan to the very start of the input list provided for the scan. If this does not succeed. the first element of the list is passed unchanged to the output of the scan and the DCG is tried again on the list. starting from the second element. Wherever the DCG succeeds in recognising a construct and hence produces as output a parse tree. the elements in the input which have been recognised as comprising the construct are replaced by the parse tree for it. Processing then continues from the first element following the construct. The scan continues until there are no further elements in the input.

To take a concrete example. suppose the input to a scan is `[save/'VB', the/'DT', file/'NN']` and the construct to be recognised is `cn`. The DCG for `cn` is iirst applied to the input starting at `save/'VB'`. Since a verb can not start a cn. the DCG fails. `save/'VB'` is thus passed to the output of the scan. The DCG is now applied to the input starting at `the/'DT'`. This also fails since `the/'DT'` can not start a `ir`. It is thus passed to the output of the scan. Next. the DCG is tried on the input starting at `file/'NN'`. This time. analysis succeeds. resulting in the parse tree `[cn, [ns, [n, file/'NN']]]` being passed to the output instead of the input recognised within the tree. namely the single word `file/'NN'`. Analysis continues by trying to apply the DCG to the input starting after the end of the `cn` just recognised. In this example there is no input left. so the scan finishes. The output of the scan is thus `[save/'VB', the/'DT', [cn, [ns, [n, file/'NN']]]]`. The scan following the one being considered will thus be given an input comprising three elements. The first two (`save/'VB'` and `the/'3T'`) are terminal symbols. The third (`[sr, [ns, [n, filo/'NN']]]]`) is a parse tree corresponding to the non-terminal symbol `cn`.

The implementation of the parsing algorithm has an important ramification i'or the design and organisation of the DCGs for each scan. When a DCG looks for an instance of a construct. it succeeds as soon as one is found. No further searches take place and. in particular. no hacktracking is permitted. In the case where several forms of the same construct could be

recognised at a given point. the first construct found will be returned. It is often the case that there are both shorter and longer instances which could occur at a particular place. For example. suppose a scan is looking for a `cn` in the input `[system/'NN', file/'NN', directory/'NN']` and that `cn` is defined to comprise one. two or three nouns. There are thus three `cn` constmcts which could be recognised at the current point: `[cn, [ns, [n, system/'NN']]]`. `[cn, [ns, [n, system/'NN'], [n, file/'NN']]]`. and `[cn, [ns, [n, system/'NN'], [n, íile/'NN'], [n, directory/'NN']]]`. In all probability we will wish torecognise the third instance since it is the longest. To ensure that this occurs. the grammar rules in the DCG for `cn` need to be ordered so that the longest constmct is Iooked for first. In practice. we have found that this issue does not cause serious problems once it is recognised.

III. THE TEST COLLECTION

III.1. The Application Domain

As we have seen. the text domain used for this study is that of a software instmction manual. the *Ami Pro for Windows User's Guide Release Three* (Ami Pro. 1993). The characteristics of the manual can be summarised as follows. It is a comprehensive guide to the the Lotus word processor Ami Pro. and is intended to contain everything which a user needs to know to use the software. including both elementary and advanced features. The manual contains 621 pages. There are 32 chapters as well as contents pages. a reading guide. four appendices and an index. Each chapter is divided into sections and subsections. neither of which are numbered. Sections vary in length between one or two lines and a few pages. with the average length being around half a page. Each sectionísubsection is devoted to a particular topic.

III.2. The Sentences

During the Industria1 Parsing study (Sutcliffe. Koch and McElligott. 1996). 600 sentences were selected. 200 each derived from the Ami Pro. Trados (1995) and Dynix (1991) manuals. The 200 Ami Pro sentences were therefore used for this study. The main syntactic characteristics ofthese sentences are now summarised. Firstly. sentences come in a number of overall syntactic forms. summarised in Table 1. The forms are Sentence **S** (e.g. S199 'Several text formatting and enhancement commands are toggles.'). Imperative IMP (e.g. S24 'Drag the mouse until you reach the end of the text you want to select. and then release the mouse button.'). Infinitive Verb Phrase **IVP** (e.g. S5 'To scroll in a document'). Third Person Singular Verb Phrase **3PS** (e.g.

S157 'Permanently *inserts the date* the *current document was created.'). Progressive Verb Phrase* **PVP** (e.g. *S1 'Editing a* Document') *and* Noun *Phrase NP* (e.g. *S150* 'System *time'). As Table 1* shows, *less* than half the *200 sentences* (45.5%) *are* of *type S. This* underlines *the need for a* parser *to accept* incomplete inputs. Imperatives form 34% of the *collection and are* the *next largest* group *alier sentences.* After *this there is a* large *drop down to* the remaining *constructions IVP (9%). PVP* (6.5%). *and* 3PS *and NP (both 3.5%).*

	S	IMP	IVP	3PS	PVP	NP	Total
No	91	68	18	5	13	5	200
Percent	45 5	34.0	9.0	7.5	6.5	2 5	100

Table. 1 Breakdown of 200 Lotus Sentences by Syntactic Type

Secondly. it should be observed that the sentences use very complicated *coordination.* Almost *any constituent can be coordinated including determiners* (e.g. *S172 'You can select Off. 1. 2. 3. or 4* levels.'). adjectives (e.g. *S35* 'You *can select text on* multiple *pages by dragging the* mouse beyond *the top or bottom margins ol'the pages.'). nouns* (e.g. *S140 'To insert the date or time into a document').* noun *phrases* (e.g. *S19 'You can use either the mouse or rhe keyboard to select text.'). adverbs* (e.g. *S6* 'With *a mouse.* you can *use the vertical scroll* arrows. *scroll bar. or scroll box on* the *right* side of *the screen to* go *forward or backward* in *a document by lines. screens. or pages.). verbs* (e.g. *S1 14* 'You *can move or copy text* from *one* Ami *Pro document to another document.'). and verbphrases* (e.g. *S12 'To use these shortcuts. hold the first key and press the second key.'*).

Thirdly. *technical* terms usually *comprising* noun *compounds occur very* frequently (e.g. *S198* 'When *the insertion point is on text that has* been modified using SmartIcons. *the status bar.* or the *Text menu. a check mark appears* beside *the appropriate menu item when* you *access* the *Text menu.*').

Fourthly. *material enclosed in quotation marks or round brackets can occur* (e.g. *S21 'For* information about *changing the appearance* of *text.* refer *to "Understanding text formatting and text enhancements" in Chapter 6.'. S193 'Text formatting* is *any* typeface. point *size. color. attribute (bold. italic. underline, word underline). capitalization. or special* effect *you* apply *to selected text using either* SmartIcons. the *status bar. or the Text menu.'). Such text* requires *special* processing if *it is to be* parsed *correctly.*

Finally. notwithstanding *the* previous comments. Ami *Pro sentences are regular. In particular* phenomena such *as anaphors.* ellipsis *(material* implied *though* left *out)* slang expressions *and irregular constructions do not occur very* frequently. *Thus.* if *the points above are addressed* successfully *by the* parser. a *high* level of *performance can be* achieved.

III.3. *Markup* of *Sentences*

In order to carry *out the study. the set of* 200 *sentences* was first *marked up with two pieces of* information: part-of-speech *tags and syntactic phrase boundaries. The sentences were* first *run through the* Brill *Tagger* Version *1.0. This tagger has* not been trained *on test fiom* our *domain. and thus* accuracy is *not* very high. *The tagged text was therefore checked by hand and any erroneous tags* were *corrected. It should be pointed out of course that there is a certain* level *of' intrinsic ambiguity in tag assignment. It* therefore follows *that if the tag correction had* been *carried out by sonieone other than the author. the tagged text used Sor parsing* would have been slightly different.

Following *tag correction. phrase boundaries were assigned to sentences. 4 types of phrase are* marked *up as follows:* noun *group* **ng.** noun *phrase* **np.** *prepositional phrase* **pp.** *and verb group vg*. A noun *group is a sequence* of nouns *which* may *contain coordination. It* might *he a single* noun (e.g. 'information' *froni S190). a compound* noun (e.g. 'scroll bars' *fiom* S4). *ananie* comprising one *or* more *proper nouns* (e.g 'I-beam' *froni* S23 or *'Ami Pro' fromS89) a* name involving *a* number (e.g. *'Chapter* 3' from S74). *or a* pronoun (e.g. 'you' *from* S155). Noun *groups can* also *contain* and/or. *ampersand or* comma coordination (e.g. *'date or tinie'* from *S138.* 'SHIFT+DEL *or* CTRL+X' *Srom S117.* 'Drag & *Drop' fiom* S71. *or 'block. paragraph. or word'* from *S18). A contiguous sequence of nuun-type words can* constitute *more* than *one group. For* example. consider S88: *'Text you move* remains on *the clipboard* until you *copy or* cut *other text. data. or a* picture.'. Here. *'text' is one* noun *group while* 'you' is *another.* because *it* is the *start of a* relative clause *froni which* the *marker* 'which' *has* been left *out.* In *other words. 'text* you move...' is *an* elliptical *abbreviatioii* for *'text which* you *move...'*.

The noun *phrase includes* determiners. *adjectival* modifiers *and adjectives* together *with* one *or* more noun *groups. Determineis can be simple* (e.g. *'a* Document' *froni S1 or 'the date from* S139). involve *nunibers* (e.g. *'two* ways' from S161) *or* involve *coordination* (e.g. *'one or more* actions' *froni* S167. *'Off. 1. 2. 3. or- 4* levels' from S172). Adjectives *can occur* (e.g. 'the *previous word' from S70) as can adjectival modifiers* (e.g. 'the *most recent action'* from S176). *Examples* of noun *groups* have *already* been seen. Finally. noun phrases *can* themselves *be coordinated* (e.g. *'either the mouse or the keyboard' froni* S19. 'other *text. data. or a picture' fiom* S88).

The prepositional *phrase comprises a preposition* followed *by a* noun *phrase* (e.g. 'in *Ami Pro'* from S168. 'to *a specific block. paragraph. or word'* from S18). *In this* study we have *assumed that certain* multiple word *phrases are prepositions* (e.g. 'to the *righi* of' *in 'to the right of* a *return' in S15*).

The verb *group coniprises adverbs* (e.g. *'Permanently* inserts' from S146. 'affect *only' Iiom S188).* and various modals. auxiliaries *and particles* (e.g. *'being accessed* and edited' *froni*

S46. 'can type *over*' from S20. '*must* click' from S9. '*have been* saved' from S164. '*has been* modified' from S198). The main verb may be coordinated (e.g. 'adding *and* removing' from S200).

The verb group can be infinitive (e.g. 'to modify just' from S187). progressive (e.g. 'typing. deleting. moving. and copying' fiom S46). passive (e.g. 'can be reversed' from S165). copular (e.g. 'to be able' from S3. 'appears highlighted' from S37. 'is on text' from S198). and negative ('*cannot* edit' from S47. '*does not* place' from S75. '*do not* reverse' from S164). Adverbs can occur at the beginning (e.g. '*thereby* reducing' from S131) end (e.g. 'click *elsewhere*' from S42) and middle (e.g. 'can. *however*. use' in S16. 'can *also* hold' from S34) of verb groups. Isolated adverbs can also occur (e.g. S60 '*otherwise*. you type over existing text.') and we consider these as isolated verb groups even though the verb which they modify ('type over' in this example) is in another verb group.

Sentences are marked up with the grammatical information using Prolog lists. Figure 3 shows a sample of the 300 sentences. The Iirst element of'each list is the name of the construct it denotes and the following elements contain the construct itself. Thus. for example. `[ vg, 'Editing'/'VBG' ]` means that 'Editing' is a verb group `vg`. One construct can be nested within another. Thus `[ np, the/'DT', [ rg, pages/'NNS' ] ]` denotes the lact that 'the pages' is a noun phrase which contains the noun group 'pages'.

IV. DEVELOPMENT OF THE GRAMMAR

The starting point for this study was an initial grammar for the Ami Pro text which had been created for the robust parser as part of the SIFT project (Hyland. Koch. Sutcliffe and Vossen. 1996). The initial grammar was then refined using the following method. Firstly. the process of correcting the part-of-speech tags for the 300 sentences provided an opportunity to study their syntactic characteristics in detail. A list of syntactic phenomena likely not to be within the coverage of the initial grammar was prepared. When tag correction was complete. the constructs were considered in turn – noun group. noun phrase. prepositional phrase and verb group. For each construct. the phenomena on the list which related to it were studied and appropriate changes were then made to the grammar. The performance of the grammar was checked at various stages by an automatic comparison of the parses produced by it for each sentence with the refeience parses (illustrated in Figure 3). Once all the points had been addressed. the detailed output from the evaluation process was studied in order to detect instances of constructs which were not properly recognised. This led to further changes to the grammar.

```
[ [ vg, 'Editing'/'VBG' ], [ np, a/'DT', [ ng, 'Document''NNP' ] ] ]

[ [ vg, 'Moving'/'VBG',around/'RP' ], [ pp, in/'IN', [ np, a/'DT', [
ng, document''NN' ]  ]  ] ]

[ 'When'/'WRB', [ np, [ ng, you/'PRP' ] ], [ vg, are/'VBP',
editing/'VBG' ], [ np, a/'DT', [ ng, document/'NN' ] ], ',',',', [ np,
[ ng, you/'PRP' ] ], [ vg, want/'VBP' ], [ vg, to/'TO',
be/'VB',able/'JJ' ], [ vg, to/'TO', move 'VB', quickly/'RB' ], [ pp,
through/'IN',[ np, the''DT', [ ng, pages/'NNS' ] ] ], '.'/'.' ]

[ [ np, [ ng, 'You'/'PRP' ] ], [ vg, can/'MD', utilize/'VB' ], [ np,
the/'DT', [ ng, scroll/'NN', bars/'NNS' ] ], [ vg, to/'TO',
display/'VB' ], [ np, different/'JJ', [ ng, parts''NNS' ] ],[ pp,
of/'IN',[ np, the/'DT', [ ng, document/'NN' ] ] ],and/'CC', [ vg,
use/'VB' ], [ np, [ ng, keyboard/'NN', shortcuts/'NNS' ] ], [ vg,
to 'TO', navigate/'VB' ], [ np, a/'DT', [ ng, document/'NN' ] ],
'.'/'.' ]

[ [ vg, 'To'/'TO', scroll/'VB' ], [ pp, in/'IN', [ np, a/'DT', [ ng,
document/'NN' ] ] ] ]

[ [ pp, 'With'/'IN', [ np, a/'DT', [ ng, mouse/'NN' ] ] ], ','/',', [
np, [ ng, you/'PRP' ] ], [ vg, can/'MD', use/'VB' ], [ np, the/'DT',
vertical/'JJ', [ ng, scroll/'NN', arrows/'NNS', ','/',', scroll/'NN',
b  '', '  ', ','/',', or/'CC', scroll/'NN', box/'NN' ] ], [ pp, on/'IN',
[ np, the/'DT', right/'JJ', side/'NN' ] ], [ pp, of/'IN', [ np,
the/'DI', [ ng, screen/'NN' ] ] ], [ vg, to/'TO', go/'VB',
forward/'RB', or/'CC', backward/'RB' ], [ pp, in/'IN', [ np, a/'DT', [
ng, document/'NN' ] ] ],,[ pp, by/'IN', [ np, [ ng, lines/'NNS',
','/',', screens/'NNS',  ;'/', ',or/'CC', pages/'NNS' ] ] ], '.'/'.' ]

[ [ np, [ ng, 'You'/'PRP' ] ], [ vg, ca ''MD', also/'RB', use/'VB' ],
  np, the/'DT', horizontal/'JJ', [ ng, scroll''NN', arrows''NNS',
 ,'/',', scroll 'NN', bar/'NN', ','/',', or 'CC', scroll/'NN',
box/'NN' ] ], [ pp, at/'IN', [ np, the/'DT', [ ng, bottom/'NN' ] ] ],
[ pp, of/'IN', [ np, the/'DT', [ ng, screen/'NN' ] ] ], [ vg, to/'TO',
also/'VB' ], [ np, the/'DT', left/'JJ', or 'CC', right/'JJ', [ ng,
sides/'NNS' ] ] ,[ pp, of/'IN', [ np, your 'PRP$', wide/'JJ', [ ng,
documents/'NNS' ] ,or/'CC', enlarged/'VBN', [ ng, views/'NNS' ] ] ],
'.'/'.' ]

[ [ np, [ ng, 'Scrolling'/'NN' ] ], [ vg, changes/'VBZ' ], [ np,
the/'DT', [ ng, display/'NN' ] ], but/'CC', [ vg, does/'VBZ',
(not)/'RP', move/'VB'    [ np, the/'DI', [ ng, insertion/'NN',
point/'NN' ] ], '.'/'.' ]

[ [ np, [ ng, 'You'/'PRP' ] ], [ vg, must/'MD', click/'VB' ], [ pp,
in/'IN', [ np, the/'DT', [ ng, document/'NN' ] ] ], [ vg, to/'TO',
place/'VB' ], [ np, the/'DT', [ ng, insertion/'NN', point/'NN' ] ], [
pp, at/'IN', [ np, a/'DT', new/'JJ', [ ng, location/'NN' ] ] ],
'.'/'.' ]
[ [ vg, 'To'/'TO', use/'VB' ], [ np, [ ng, keyboard/'NN',
shortcuts 'NNS' ] ], [ vg, to/'TO', navigate/'VB' ], [ np, a/'DT',
document/'NN' ] ]
```

Figure. 3 Sample of Sentences Tagged for Part-of-Speech and Syntactic Construct

V. EVALUATION

V.1. Method

Evaluation of parser performance was carried out in the following manner. Firstly. the 200 test sentences were tagged Sor part-of-speech and then marked up with a bracketing to denote the boundaries of four constructs: noun group ng. noun phrase np. prepositional phrase pp and verb group vg. as already described.

The next stape was to develop an evaluation procedure. It was decided to adopta standard phrase-boundary identification method. working as follows. Firstly. for each of the four constructs ng. np. pp and vg. a construct was identified in the grammar which was going to identify the same phrase boundaries. The constructs were cn for ng. cnp for np. cpp for pp and cvg for vg. Evaluation then involved comparison of the constructs recognised by cn with those demarcated by ng. and similarly comparison between cnp and np. cpp and pp. and cvg and vg.

Comparison involved the following procedure. For each construct (e.g. cn/ng). each sentence was inspected in turn. A list of reference instances of the construct was extracted from the reference analysis of the sentence. A list of candidate instances was likewise extracted from the analysis produced by the parser. Each candidate instance was considered in turn. as follows: if the candidate spanned the same terminal symbols as one of the reference instances. then the candidate was marked correct and the reference instance was deleted from the list of reference instances. The next candidate instance was then considered in the same way. When this process was complete. the standard Precision and Recall measures were computed for the sentence: Precision is considered to be the number of correct candidate instances divided by the nunibei oi'candidate instances. Recall is the number of correct candidate instances divided by the number of reference instances. In addition. the counts of the number ofreference instances. the number of candidate instances and the number of correct candidate instances were stored in order to compute the overall Precision and Recall figures for the whole set of 200 sentences.

The above evaluation scheme equates to the well-established Parseval method (Black et al.. 1991) which advocates the use of Precision. Recall and bracket crossing. Given that the reference parses were prepared with the study in mind. we decided not to use the bracket crossing measure and instead to make a simple binary decision for each construct within a sentence.

V.2. Results

Results are summarised in Tables 2-5. Table 2 shows the initial scans which were used in the parser before the start of the study. As can be seen. there were only three scans. Sor coordinated noun phrases. coordinated prepositional phrases and verb groups respectively. Table 3 shows the

scans used in the final system. There are now five scans. Firstly, a scan for quoted expressions was added at the start. This recognises quotations such as are found in S21: 'For information about changing the appearance of text, refer to *"Understanding text formatting and text enhancements"* in Chapter 6.'. The entire quotation "'Understanding text formatting and text enhancements"' is automatically recognised as a noun phrase, and no further processing is done to the contents. The reason for doing this is that all quoted expressions in the text serve syntactically as noun phrases and so this technique leads to a correct analysis of all such sentences. The text within the quotation could also be analysed independently of the enclosing sentence but this was not done in the present study and neither was this text marked up with phrase boundaries in the reference parses. The second change is that there is an additional scan for prepositions themselves. The reason for this is that the assumption was built into the project that certain word sequences such as 'adjacent to', 'at the bottom of', 'from within', 'on top of', 'up to' and so on should be treated as prepositions. However, as can be seen, some of these 'prepositions' in fact contain nouns such as 'bottom' or 'top' and adjectives such as 'adjacent'. Since the noun phrase scan preceded the prepositional phrase scan (by definition) and the prepositions were originally not being recognised until the latter scan, this led to premature recognition of those nouns and adjectives as parts of noun groups and noun phrases. To correct this, the prepositions were recognised in a separate scan following the quoted expression analysis. This kind of problem will always occur in any parsing approach which is divided into discrete scans, but there are many advantages of the approach which we will summarise in the conclusions.

Table 3 Final Parsing Scans (Stage 5)

Table 4 summarises in narrative form the five stages involved in the development of the graminar. Stage Zero was the starting point of the project. namely tlie grammar developed as part of SIFT. Quoted expression processing was first added. enabling text such as '"Understanding text formatting and test enhancements"' within a sentence to be recognised as a noun phrase and not processed further. The result was Stage One. Work was then undertaken on the verb group which was originally ver); rudirnentan. A large nuniber of extra rules were added. leading to Stage Two. At this point the granimars for noun phrases and prepositional phrases were upgraded. A new scan for prepositions was also added in order to gel around the problem *o!'* preniature recognition of compouiid preposition constituents as noun phrase constituents. as discussed earlier. The result was Stage Three. At this point. a flaw in the verb group analysis scan became apparent. namely that words recognised as prepositions could not now be iiiterpreted as heralding inlinitives such as 'to be able'. Correction of the problem led to Stage Four. Finally. further work on tlie analysis of noun phrases and prepositional phrases led to Stage Five.

Table 4 Summary of Cianimar Development

Table 5 shows the numerical results of the evaluation by stage and construct. The first part of the table deals with the coiistruct ng. the second part with np. and the third and fourth parts with pp and vg. For each construct. evaluation results are shown for each stage of the project. For each construct. results are shown for each stage of grammar development. as outlined in the previous paragraph. Columns three to six show the number of reference constructs. caiididate constructs and correct candidate constructs. Naturally. the nuniber ofreference constructs is the same for a given construct regardless of tlie stage. since these were delined at the start of the project. The last two columns show the precision and recall. Precision is the nuniber ofcorrect candidates divided by the nuniber of candidates. Recall is the nuniber of correct candidates divided by tlie nuniber of reference instances. In a perfect parser. precision and recall should both be one for all constructs.

The changes in precision and recall Sor a construct from one stage to tlie next give an indication of the effect on parsing performance which was achieved by tlie work done on tlie grammar in that stage. For example. the initial performance on noun groups (ng) was P = 0.90. R = 0.78. In other words onl); 78% of all ngs were recognised initially. but for those which were

detected 90% were correct. The Iirst significant change came at Stage 3 where recall rose to 0.89 with a slight reduction in precision from 0.93 to 0.90, This corresponds to the first portion of work devoted to noun phrases and prepositional phrases. The next jump comes at Stage 5 where P = 0.94 and R = 0.97.

Regarding noun phrases (np) the change in performance over the project was from P = 0.89. R = 0.73 to P = 0.94. R = 0.95. For prepositional phrases (pp) the result was an increase from P = 0.83. R = 0.80 to P = 0.85. R = 0.90. Finally. the change for verb groups (vg) was from P = 0.53. R = 0.58 to P = 0.92. R = 0.91.

Stage	Construct	Total Reference	Total Candidate	Correct Candidate	Precision	Recall
0	ng	667	573	517	0.90	0.78
1	ng	667	564	517	0.92	0.78
2	ng	667	564	517	0.03	0.78
3	ng	667	658	594	0.90	0.89
4	ng	667	658	594	0.90	0.89
5	ng	667	687	648	0.94	0.97
0	np	667	553	400	0.89	0.73
1	np	667	551	408	0.90	0.75
3	np	667	551	498	0.90	0.75
›	np	667	645	579	0.90	0.87
4	np	667	645	579	0.90	0.87
5	np	667	675	624	0.94	0.95
0	PP	173	166	138	0.83	0.80
1	PP	173	174	146	0.84	0.85
3	PP	173	174	146	0.84	0.85
3	PP	173	181	147	0.81	0.85
4	PP	172	181	147	0.81	0.85
5	PP	172	183	155	0.85	0.90
0	vg	440	476	354	0.53	0.58
1	vg	440	467	354	0.54	0.58
3	vg	440	433	399	0.92	0.91
›	vg	440	433	339	0.78	0.77
4	vg	440	433	398	0.93	0.90
5	vg	440	423	399	0.93	0.91

Table 5 Results of Evaluation

What can be said about the change in performance of the parser? Firstly. there was a substantial improvement inrecall together with an increase in precision for all constructs. Inother words. the final parser was able to detect many more instances ol'constructs than was the original parser. At the same time. the accuracy in the recognitioii of those constructs also improved. The

biggest change was in the recognition of verb groups. This can be attributed to the fact that the original grammar was only designed to handle the simplest constructs while the final grammar was the result of a very detailed analysis of all the verb groups in the 200 sentence treebank. The smallest improvement was for prepositional phrases. The causes for this include incorrect recognition of prepositions (e.g. 'if' in S13 being taken as a preposition resulting in 'if you' being analysed as a pp) and discrepancies between what is considered a preposition in the treebanh as against the grammar. For example. in S7 'at the bottom of' is a single preposition in the grammar while the reference parse shows it as two. 'at' and 'of'. Such problems could of course be addressed.

The second point to he noted is that the improvement in performance was relatively easy to achieve. It is estimated that about three days were spent on the grammar in total during the project. However. the original grammar with its sophisticated handling of coordination was the result of perhaps one week's work which of course followed a number of years during which the syntax of Ami Pro sentences was a major focus of attention.

VI. CONCLUSIONS AND NEXT STEPS

In this section we attempt to draw some conclusions regarding the project as a whole as well as providing some pointers Sor further work. Firstly. what can be said about the strengths and weaknesses of robust parsing as an approach to the analysis of technical manual texts? The following are some conclusions:

- The approach is simple to use overall:
- Interference between stages of analysis is highly constrained:
- It is relatively easy to develop grammars for the individual constructs;
- Processes such as the tokenisation of intractable inputs. the treatment of bracketed and quoted material and the analysis of semantic case frames can be combined with parsing in the same paradigm;
- Non-linguistic techniques (e.g. the use of machine-learning algorithms) can readily be applied to specific aspects of parsing (e.g. prepositional phrase attachment) within the robust parsing paradigm:
- Parsing is efficient because the lack of interference between scans minimises the combinatorial complexity of the parsing task.

The main disadvantages can be summarised as follows:

- Accuracy of analysis is dependent on the quality of part-of-speech tagging. If this is low. then parsing performance will be also;
- Layered parsing can not handle mutually recursive structures to arbitrary depth. For example if a prepositional phrase can occur in a noun phrase which can occur in a prepositional phrase and so on. noun phrase and prepositional phrase analysis can not be handled in separate scans;
- It is necessan to decide how many scans to conduct and what analysis to carry out in each one. The ordering of scans can pose problems, as was shown in this project regarding the recognition of prepositions:
- Generally. the order in which constructs are recognised must be carefully thought out and may be anomalous. For example verbs (e.g. 'justified') can occur as adjectives in noun phrases (e.g. 'justified text') nhich implies that noun phrases must be recognised before verb phrases. On the other hand. prepositions used as particles (e.g. 'off' in 'take off') should be recognised as being within verb phrases before prepositional phrases are analysed. as such particles may otherwise be mistaken ior prepositions heralding prepositional phrases (e.g. 'off the cover' in 'take offthe cover').

Secondly. how applicable is the parsing approach to technical manual text? From this project it seems that robust parsing is highly suited to such text. probably because it is regular though complex. The most difficult aspect is undoubtedly the prevalence and complexity of coordination but this is amenable to treatment by the layered approach. The ability to include the processing ofquoted and bracketed text as an integral part of parsing is a major advantage as such material occurs frequently. On the negative side. we alluded earlier to the potential difficulties caused by mutually recursive structures. Fortunately. honever. such structures occur rarely in manuals.

Finally. what further work could be carried out based on the findings presented here? There are a number of experiments which we would like to carry out:

- The cost of developing the grammar has not been properly measured in this project. All we have presented is estimates of the time taken in days. It would be interesting to devise and use more accurate metrics.
- The nuinber ofrules used in a series ofgrammars (e.g. at different stages) could be measured and compared with the coverage of the grammar and the time taken to develop it. This could lead to metrics which predict both the precise cost of developing a grammar to a particular standard of accuracy and the size of that grammar.
- The size of a grammar (and hence its coverage) could be compared to the efficiency (i.e. speed) ofparsing. Efficiency definitely decreased in the course of this project as rules were

added. though it was not actually measured. *Such* a comparison *could* lead *to* the computation *of an optiniuni iiumber* of rules for *a particular task* which *gave* maximum *coverage within a particular* efficiency *threshold.*

- *Tlie* effect on *parsing* accuracy of errors *in part-of-speechtagging could be* measured. *In this* project, all such errors were corrected *by hand so that tlie parsing performance could be nieasured* independently *In a* practical *parsing* task, however, *we* probably wish *to know what* accuracy can *be attained* on *free* text using *automatic tagging.*

- *It would be* very interesting *to establish what* is *the limit of accuracy* which *can be attained by a* simple parsing *approach based on* grammars *and* at *what cost. This limit* was *not reached in the* current *project as* more *developnient work could* have been *done to the gramniar.* An example *of a phenoiiienon* which *can not* readily *be liandled by grarnrnars is* the *structural* ambiguity *ot'prepositional phrase attachmeiit. Following on* from *this. one could establish tlie limit of* accuracy *attainable using special techniques* to handle *such* ambiguity.

- Finally, *we would like to make a coinprehensive itineran of specinl phenomena which occur frequently in technical* manual *text. together with a palette of techniques* for *handling* them. *We* have *already looked at bracketed expressions. quotations and compound* prepositions *in this* project but *tliere are* probably *other constructions in* this category which *are* amenable to *special* processing.

*In conclusion. tlierefore.*the *project has* proved *niost interesting* but, *as* always, many new *problems* worthy of *investigation* have *come to light.*

REFERENCES

Ami Pro (1993). *Lotus Ami Pro for Windows User's Guide Release Three*. Atlanta. Georgia: Lotus Development Corporation.

Ami Pro. 1994. *Lotus Ami Pro Word Processor for Windows User's Guide Release 3.IJ* (Japanese Version). Cambridge. MA: Lotus Development Corporation.

Black. E.. Abney. S.. Flickinger. D.. Gdaniec. C.. Grishman. R.. Harrison. P.. Hindle. D.. Ingria. R.. Jelinek. F.. Klavans. J.. Liberman. M.. Marcus. M.. Roukos. S.. Santorini. B. & Strzalkowski. T. (1991). A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. *Proceedings of the DARPA Speech and Natural Language Workshop*. 306-311.

Brill. E. (1992). A simple rule-based part ofspeech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing. AC'L, Trento. Italy. 1992.*

Carroll. J.. Briscoe. T.. & Sanfilippo. A. (1998). Parser Evaluation: A Survey and a New Proposal. *Proceedings of the First International Conference on Language Resources and Evaluation. Granada. Spain. 28-30 May IYYX.* 447-454.

Dynix (1991). *Dynix Automated Library Systems Searching Manual*. Evanston. Illinois: Ameritech Inc.

Garside. R.. Leech. G.. & Sampson. G. (1987): *The Computational Analysis of English: A Corpus-Based Approach*. London. LJK: Longman.

Ilyland. P.. Koch. H.-D.. Sutcliffe. R. F. E.. & Vossen. P. (1996). *Selecting Information from Text (SIFT) Final Report* (LRE-62030 Deliverable D61). Luxembourg. Luxembourg: Commission oi'the European Communities. DGXIII/E5.

Sleator. D. D. K.. & Temperley. D. (1991). *Parsing English with a Link Grammar* (Technical Report CMU-CS-91-196). Pittsburgh. PA: Carnegie Mellon University. School of Computer Science.

Sutcliffe. R.F.E. (1991). Distributed Representations in a Text Based Information Retrieval System: A New Way of Using the Vector Space Model. In A. Bookstein. Y. Chiaramella. G. Salton and V. V. Raghavan (Eds.) *The Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval. Chicago. IL. October 13-16. 1991* (pp. 123-132). New York. NY: ACM Press.

Sutcliffe. R. F. E. (1998). Organising Parser Evaluation. Abstract in *Proceedings of the workshop 'Towards a European Evaluation Infrastructure for Natural Language and Speech. Granada, Spain. Wednesday 27 May 1998'* forming part of *The First International Conference on Language Resources and Evaluation. Granada. Spain. 28-29 May. 1998.*

Sutcliffe. R. F. E.. Koch. 11.-D.. & McElligott. A. (Eds.) (1996). *Industrial Parsing of Software Manuals*. Amsterdam. The Netherlands: Rodopi.

Sutcliffe. R. F. E.. & Kurohashi. S. (3000). A Parallel English-Japanese Query Collection for the Evaluation of On-Line Help Systems. *Proceedings of the Second International Conference on Language Resources and Evaluation. Athens. Greece. 31 May - 2 June 2000.*

Sutcliffe. R. F. E.. & McElligott. A. (1996). Using the Link Parser of Sleator and Temperley to Analyse a Sol'tware Manual Corpus. In R. F. E. Sutcliffe. H.-D. Koch and A. McElligott (Eds.) *Industrial Parsing of Software Manuals* (pp. 89-102). Amsterdam. The Netherlands: Rodopi.

Sutcliffe. R. F. E.. & Nashimoto. N. (1999). *Robust Parsing of Technical Japanese Text.* Teclinical Note. Department of Computer Science and Information Systems. University of Limerick. Ireland.

Trados (1995). *Trados Translator's Workbench for Windows User's Guide*. Stuttgart. Germany: Trados GmbH.