# A Robust Partial Parsing Strategy based on the Slot Unification Grammars

Palomar, M.[†]; Ferrández, A.[†]; Moreno, L.[‡]; Saiz-Noeda, M.[†]; Muñoz, R.[†]; Martínez-Barco, P.[†]; Peral, J.[†] and Navarro, B.[†]

[†]Dept. Lenguajes y Sistemas Informáticos
Universidad de Alicante
Apdo. de correos 99. 03080 – Alicante (Spain)
*{patricio, jperal, antonio, mpalomar}@dlsi.ua.es*

[‡]Dept. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Apdo. de correos 22012. 46071-Valencia (Spain)
*lmoreno@dsic.upv.es*

## Abstract

In this paper we present a robust partial parser (Slot Unification Partial Parser, SUPP) based on the Slot Unification Grammars, SUG. Our parsing strategy analyzes coordinated nouns and prepositional phrases and verbal chunks (verbs in their simple and compound forms and verbal periphrasis) and it's guided to the linguistic phenomena resolution. Its adaptability to different taggers or dictionaries makes it a general purpose system. To show this universality, the system has been evaluated with two Spanish corpora (LEXESP and Blue Book), achieving precision and recall values between 95%-97%.

## 1. Introduction

Most parsers that have been developed in the field of Natural Language Processing (NLP) in the recent years accomplished exact analysis of the input sentences, accepting only those that were recognized by the grammar. For this reason, people worked with closed discourse universes in which the grammar was capable of recognizing all its phrases. Therefore, the main objective to pursue was to find out the best parser for the grammar Abney (1997).

Nevertheless, the field of NLP includes unrestricted texts in which the lack of lexical information and grammatical rules that encompass all the discourse universe together with the difficulty that involves the length of the sentences and the ambiguity of the grammar make the use of complete parsing useless. As an alternative, the use of partial parsing techniques capable of retrieving syntactic and relevant information from the text sacrificing completeness and depth of the analysis is outlined. For this reason, a parsing of small structures of relevant information capable of being retrieved with little syntactic information is accomplished, instead of the complete analysis techniques where large structures that require the use of much lexical and semantic information were parsed.

In this work we outline a partial analysis system based on a set of grammatical rules in order to extract from unrestricted texts solely the syntactic information that is relevant according to these rules, using a technique based on what we call *slot structures* (henceforth *SS*) that generate the relevant elements of the sentence.

In this way, the parser that we have designed, *SUPP* (Slot Unification Partial Parser), is capable of extracting coordinated noun and prepositional phrases and verbal chunks[i] (formed by verbs in their simple forms, their compound forms and verbal periphrasis without preposition, also including the passive voice), widening considerably the set of syntactic elements that we are capable of extracting instead of other traditional partial parsers that were only working on the extraction of noun phrases. For this reason, our system becomes a good tool for subsequent applications such as information extraction and the constitution of superior order syntactic elements through stochastic techniques.

In addition to this important advantage, our system allows linguistic phenomena resolution modules such as anaphora in Ferrández et *al.* (1997), Ferrández et *al.* (1998), left extraposition, as well as the structural ambiguity to be fitted. For all of these, SUPP is a consistent analysis system. The capability of changing the definition of the interface for each set of tags, makes this system ready to use with any tagger or dictionary.

Next section shows main partial parsing strategies developed in this area. After this, we will present the formalism of Slot Unification Grammar (SUG) as base for the parser *SUPP*. In the third section, we will describe the complete system. To conclude, we will show the results obtained through applying *SUPP* to two different corpora: a fragment of the LEXESP[ii] Spanish corpus, manually checked, containing 71.849 words in 2.738 sentences, and a fragment of the Blue Book Spanish corpus, containing 9.407 words in 245 sentences. This will allow us to compare our system with other strategies.

## 2. Background

The idea of partial parsing consists of dividing the parsing into small structures that could be recovered with a small amount of syntactic information, while other types of parsing work with structures that require much lexical association information. Abney (1997) defines these small structures with the concept of chunks. Chunks are defined as the non recursive center of a intrasentence constituent, and it expands itself from its start to the kernel, without including post-modifiers. Another important concept in the partial parsing is the simplex clause. It is defined as a clause that includes non embedded syntactic structures, that is, chunks with no attachment between them.

Chunks and simplex clauses can be recovered with a small regular-expression grammar, postponing the attachment of the different constituents.

Within the different algorithms of partial parsing resolution existing in the current literature, we can differentiate two trends Moreno et *al.* (1999):

- *Finite states machines*. They use regular grammars to recognize each syntactic structure type.
- *Standard algorithms in natural language processing*. Define the language through context free grammars using classic algorithms as chart, LR, etc., adapted to the partial parsing.

[i] A chunk is defined as element sequence with certain syntactic sense around a core or head [1].

[ii] The Spanish Corpus LEXESP contains 5 million of lexically tagged words and belongs to the project of the same name carried out by the Departamento de Psicología of the Universidad de Oviedo and developed by the Grupo de Lingüística Computacional of the Universitat de Barcelona, with the collaboration of the Grupo de Tratamiento del Lenguaje of the Universitat Politécnica de Catalunya.

Furthermore there are other algorithms such as text tagging, hidden model Markov (HMM) techniques and syntactic tagging based on restrictions.

One of the first chunk recognizers was developed by Ross and Tukey (1975). This algorithm looks for stretches of stop words. These words cannot form part of a noun phrase. Chunks will be the words between two stretches. Bourigault (1992) used this technique for identifying noun phrases in French.

Church (1998) used a simple stochastic technique to construct a noun-chunk recognized that consists of codifying the chunks with brackets through a tagger based on the HMM. Ramshaw and Marcus (1995) also use this technique.

One of the most successful partial parsers is Fidditch, developed by Hindle (1983), Hindle (1984). It was not developed as a partial parser. It was designed to be used on unrestricted texts. This parser simplifies the rule formalism provided by the Marcus parser to allow an easier way for writing a very large grammar capable of recognizing clause boundary markers, subjects and predicates. This parser can be implemented as a finite state automaton. It is one of the fastest parsers achieving speeds of 5600 words by second.

Brill (1993) develops a mixed method based on a text tagging with brackets and the use of learning techniques from a training corpus. Vilain and Palmer (1996) develop some techniques for improving learning speeds.

Karlsson et *al*. (1995) describes the partial parser ENGCG developing the algorithm presented by Voutilainen et *al.* (1992). This parser uses a lexical parsing to assign its part of speech and a set of possible syntactic function tags to each word. The way of disambiguating the syntactic category is the same as part of speech disambiguation, through the application of pattern-matching rules. The syntactic parsing is a dependency analysis in the sense that only word-word relations are considered. There is no association between words and their governors, although it reduces the set of possible parses.

Joshi and Srinivas (1994) describe a partial parser based on Voutilainen tagging techniques through LTAG (Lexicalized Tree Adjoining Grammars) in which each elementary tree contains a unique lexical item. These items can be attached through a dependency graph. Each word can appear in multiple elementary trees that represent a different syntactic structures. An adaptation of Viterbi search is used to select a tree for each word.

Voutilainen and Padró (1997) show a complete hybrid parser where a partial parser is accomplished. This partial parsing follows the linguistic model with restriction grammars which is adapted for including the statistic information obtained from a training corpus. The parser is capable of identifying verbs, pre-modifiers, nominal and adverbial headings and some post-modifiers. It is a noun phrase parser.

The developments of Ejerhed and Church (1983) and Ejerhed (1998) as well as the studies of Abney (1990), Abney (1991), Abney (1996) and Abney (1997) describe the finite state machines. In these machines, a set of regular expression patterns for recognizing phrases defines a sequence of strata. Each stratum takes as input the output of the previous one, being the output of stratum 0 parts of speech. These patterns are translated into finite state automatons that are joined together forming an unique determinist and minimal automaton. Each prefix of the input takes the automaton to a unique state, hence there is a longest prefix that takes the stratum automaton into a final state, and this final state is unique. That final state corresponds to a set of final states from the pattern automaton allowing us to determine which pattern is the responsible for the matching. Abney (1997) there is an approaching to this system with the construction of a HMM from the stratum recognizer.

## 3. Slot Unification Grammar (SUG)

The SUG are developed as an extension of DCG and they are named in this way because of the *slot structures* (*SS*) that are automatically generated by the parser where the linguistic information that is needed for solving linguistic phenomena is included Ferrández et *al.* (1998).

A SUG can be defined as the quadruple: $(NT,T,P,H)$, where $NT$ and $T$ are a finite set of nonterminal and terminal symbols respectively; moreover $NT \cap T = \emptyset$. $P$ is a finite set of pairs $\alpha$ ++> $\beta$ where $\alpha \in NT$, $\beta \in (T \cup NT)^{*} \cup \{procedures\ calls\}$, and these pairs are called *production rules*. Finally $H$ is a set of production rules which only has the first member of the production rule, i.e. $\alpha$, and $\alpha$*'s* name is either *coordinated*[iii], *juxtaposition*[iv], *fusion*[v], *basicWord*[vi] or *isWord*[vii].

SUG's production rules add to those of DCG that each subconstituent of $\beta$ could be omitted in the sentence if it is noted between the *optional operator*: << *constituent* >>. It is a well-known fact that we can get optional constituents in DCG from making use of a nonterminal symbol (e.g. *optA*, with $optA \rightarrow A$ and $optA \rightarrow [\,]$). However this skill obliges us to add new nonterminal symbols, whereas SUG allows us to get it without adding any new one. We can get an example from Figure 1, in which we can see the reduction of grammatical rules in SUG.

```
DCG Grammar:

np –> subst.
np –> det, subst.
np –>det, adj, subst.
np –>det, subst, adj.
np –>det, subst, pp.
…..
```

```
SUG Grammar:

np ++> <<det>>, <<adj>>,
   subst, <<adj>>, <<pp>>.
```

```
DCG Grammar with optional constituents:

np –> optDet, optAdj, subst, optAdj, optPP.
optDet –> det.      optAdj –> adj.     …..
optDet –> [ ].      optAdj –> [ ].
```

*Figure 1. Comparison between DCG and SUG with reference to optional constituents.*

Furthermore, this optional operator has the possibility of reminding whether the optional constituent has been parsed in the sentence or not. This information will be very useful in the resolution of NLP problems such as ellipsis or extraposition. This fact is carried out by adding a label to the optional constituent, e.g. << ***SSNP :*** *np* >>. This label will be an uninstantiated Prolog variable if constituent *np* is missing, so Prolog predicate *var(SSNP)* would be successful.

iii *Coordinated* production rule solves coordination between constituents.
iv *Juxtaposition* is a special coordination without conjunction
v *Fusion* production rule entails others rules with common sub-constituent
vi *BasicWord* production rule defines nonterminal symbols
vii *IsWord* production rule specifies language word

## 4. Slot Unification Partial Parser (SUPP): a robust partial parsing strategy

We are going to explain the general system of linguistic processing (Figure 2), in which is included the partial analysis strategy together with other modules of resolution of NLP problems and its possible subsequent application in a semantic analysis module.

In the first place, a SUG grammar capable of recognizing noun phrases and verbal chunks is defined. We take this grammar as input for a translator that we have developed which turns SUG rules into Prolog clauses. This translator has been run under SICStus Prolog 2.1, Arity Prolog 5.1 and LPA WinProlog, and it will translate into Prolog each SUG production rule. This translator will provide us the *SS*. This *SS* stores the syntactic, morphologic and semantic information of every constituent.

From our parsing module (partial due to the introduced grammar) a syntactic analysis of the sentence obtaining its $SS_0$ is carried out. Then, the module of resolution of NLP problems using $SS_0$ will be applied. The solution will consist of a new *SS*, $SS_0'$, in which the linguistic problems (anaphora, ellipsis, wh-movement) have been eliminated.

We would like to emphasize that this skill of resolution allows us to produce modular NLP systems in which grammatical rules, parsing module, and the module of resolution of NLP problems are quite independent from each other.

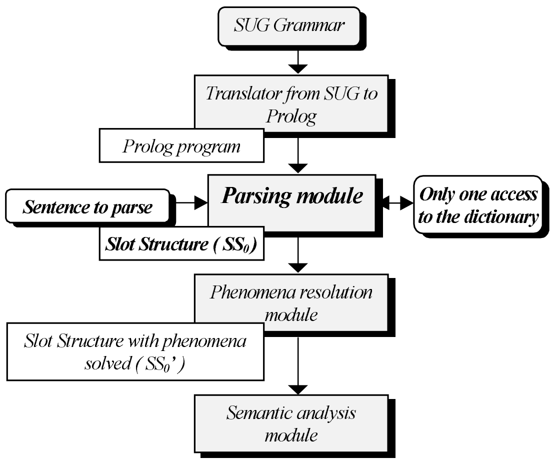
SUG Grammar
Translator from SUG to Prolog
Prolog program
Sentence to parse
Parsing module
Only one access to the dictionary
Slot Structure ($SS_0$)
Phenomena resolution module
Slot Structure with phenomena solved ($SS_0'$)
Semantic analysis module

*Figure 2. Complete application with SUPP.*

Our SUPP parser will access the dictionary only once during the whole process of parsing in order to avoid repeated access to the same word from the dictionary. It stores the information of each word on a list before starting the parse and it will work with this structure instead of the list of words of a DCG parser in Prolog; e.g. DCG list: *[this, book, is, mine]*, SUG list: ***[word** (this, **[**adj (sing, dem)**,** pron (sing, dem)**]** ), **word** (book, **[**noun (...)**]**), ...]*. Each element from the SUG list is a structure with name *word* and with two arguments. The first one corresponds to the same word of the sentence like a Prolog atom. The second one corresponds to a structure list which refers to the lexical entries of the word. That is to say that every time the parser has to access a lexical entry of a word, it will look it up in this list; it will not access the dictionary ever again.

In this work, we are going to apply a partial parsing strategy on the output of a part-of-speech (POS) tagger, in a similar way to the algorithm proposed by Kennedy and Boguraev (1996). We will work on the corpus used within the LEXESP project that consists of journalistic texts, articles on various subjects (human, political, sports, etc.) and some short literary extracts. This Spanish corpus consists approximately of 5M words tagged with their grammatical categories and morphological information.

As it can be observed in Figure 3, we will begin with a tagged sentence that is turned into the SUG list format, where each tag is mapped into the appropriate label in the SUG grammar. Finally, this SUG list of words will be taken as the input for the grammar described in Figure 4. This grammar will carry out the partial parsing of the text. This simple interface between the tagger and SUPP is one of the advantages of modularity that SUPP presents. This will allow us to work with different dictionaries or taggers with the same SUG grammar.

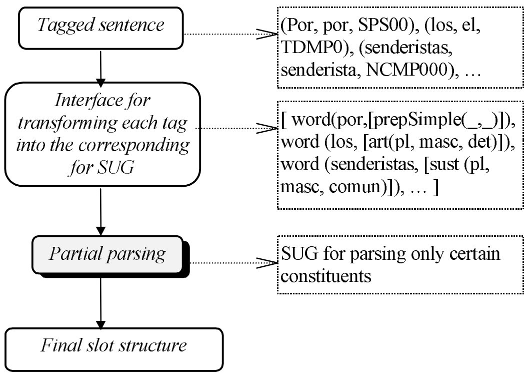
Tagged sentence
(Por, por, SPS00), (los, el, TDMP0), (senderistas, senderista, NCMP000), …
Interface for transforming each tag into the corresponding for SUG
[ word(por,[prepSimple(_,_)]), word (los, [art(pl, masc, det)]), word (senderistas, [sust (pl, masc, comun)]), … ]
Partial parsing
SUG for parsing only certain constituents
Final slot structure

*Figure 3. Interface between tagger and SUPP.*

A fragment of the SUG grammar that we have used in our tests is shown in Figure 4. This grammar will only parse coordinated prepositional phrases (PP), coordinated noun phrases (NP), and verbs in simple form, compound form, and verbal periphrasis (including passive voice) in whatever order that they appear in the text. The remaining words not recognized by the grammar are syntactically treated as loose words (without binding to any noun phrase or verb) and are represented in the output as 'PALABRA'.

## 5. Evaluation and Discussion

We have applied our partial parser, SUPP, to a fragment of the LEXESP corpus manually corrected that contains 71.849 words in 2738 sentences (medium length of a sentence is 26.24 words), obtaining the results that we will reveal below. To accomplish the values measurement, we have based it on the studies of Carroll and Briscoe (1998), in which it is shown that, even though there is no measurement system that faithfully represents the effectiveness of the system, precision and recall are very extended measures, allowing us to make comparisons with other systems. The evaluation system for our partial parser is based on precision and recall in extraction of simple noun phrases that may be included in another superior order noun phrase, constituted by a single core and constituents that complement it, i.e. *the white house*, and complete noun phrases (or of the most superior order) that can be constituted by other noun phrases and/or subordinated sentences and that do not belong to any noun phrase of superior order, i.e. *the red house which stood near the station*

We will define precision as the quotient between the number of correctly parsed noun phrases and the number of parsed noun phrases in the text and recall as the quotient between the number of correctly parsed noun phrases and the number of real noun phrases in the text. We will consider that a noun phrase has not been correctly parsed in 5 possible cases:

a) Error in the assigned syntactic category: when the assigned syntactic category does not correspond to the category of noun phrase.

b) Limit errors: when their limits do not coincide with the limits of real noun phrase (encompasses a greater or smaller length than the real length).

c) Multiplication errors: when a noun phrase is parsed as several.

d) Union errors: when several noun phrases are taken as one only.

e) Omission errors: when a noun phrase is not recognized as such.

```
partialSentence ++>
   << PP:pp >>, << NP:np >>, << V:verb >>,
   <# [ ],
      remainingSentence(PP,NP,V)
   #> .
remainingSentence(PP,NP,V) ++>
   <## ( {( var(PP), var(NP), var(V))}, [W]),
          ( _ , _  )
   ##>,
   partialSentence.

%----- Grammatical rules for each constituent to parse

coordinated( pp, simplePP ).
simplePP ++> preposition, np.
coordinated( np, simpleNP (_) ).
simpleNP (substantiveType) ++> <<determiner>>,
      <<adjective>>, noun, <<adjective>>, <<pp>>.
simpleNP (adjectiveType) ++> <<determiner>>,
      adjective, <<pp>>.
simpleNP (pronounType) ++> pronoun.
simpleNP (infinitiveType) ++> <<determiner>>,
      infinitiveVerb, <<pp>>.
verb ++> <# compVerb,periphrasisVerb,simpleVerb #>
compVerb ++> <<auxVerb>>, auxVerb,
      <# auxVerb, participleVerb #>.
periphrasisVerb ++> simpleVerb,
      <# gerundVerb, participleVerb #>.
…
```

*Figure 4. Partial parsing with SUG.*

In this way, after supervising manually the automatic parses carried out with the input corpus and comparing results, the following values have been obtained. For the simple noun phrases, a precision of 95% and a recall of 94% is achieved, and for the complex noun phrases, a precision of 80% and a recall of 79% is obtained[viii]. Also, for the verbal chunks the systems achieves a precision of 93.5% and a recall of 95%. Finally, for the prepositional phrases we have obtained a precision of 95% and a recall of 94%.

[viii] The decrease obtained from the complex noun phrases with respect to the simple is mainly due to errors produced by the linguistic phenomena action such as the structural ambiguity, the coordination, the ellipsis, etc., without solution in the partial analysis due to lack of semantic information in the discussed corpus, and that will have to be treated in the subsequent phase.

Furthermore, it has been detected that, in spite of being a manually corrected corpus, some isolated errors of lexical and morphological tagged items exist and alter the result in the case of simple noun phrases as well as complete noun phrases. So, better results in free tagged errors corpus would be obtained.

The main difference between our system and those developed previously is that we propose a parsing strategy that not only includes noun phrases but also prepositional phrases and verbal chunks, as opposed to systems that propose a much more limited partial parsing. On the other hand, in most cases, the other strategies show their results in words per second (wps). This parameter is directly dependent on the computer speed. Furthermore, our strategy works with a Spanish corpus, which is not used in the background systems. This makes comparison difficult to measure.

| *Program* | *depth* | *sw* | *hardware* | *w/s* | |
|---|---|---|---|---|---|
| **Fidditch3** | parse | C | SGI | 5600 | |
| Copsy | np | Pascal | BS2000 | 2700 | |
| CG | dep | | Sparc10 | 1550 | ±250 |
| Fidditch3 | parse | C | Sun4 | 1200 | |
| **SUPP** | **parse** | **Prolog** | **Pentium II** | **291** | |
| Pos | tag | | Sun4 | 240 | |
| Fidditch2 | parse | Lisp | Sun4 | 62 | |
| Cass | chunk | Lisp | Sun4 | 52 | |
| Clarit | np | Lisp | | 50 | |
| Fastus | chunk | Lisp | Sparc2 | 39 | |
| Cass | chunk | Lisp | UX400S | 32 | |
| Scisor | skim | | | 30 | |
| Fidditch1 | parse | Lisp | Sym-36XX | 28 | |
| McDonald | parse | | MacII | 14 | ±6 |
| Chupa | parse | Lisp | UX400S | 1,1 | |
| Traditional | parse | | | 0,20 | |

*Figure 5. Parsing speed comparison table*

Nevertheless, in comparing our method to other rules based methods, our parser uses a smaller quantity of rules (40 rules). Bourigault system uses 800 defined and hand-tested rules, Voutilainen's parser Voutilainen et *al.* (1992) uses 120 syntactic rules and 1300 morphological rules. This decrease in the number of rules is due to the logical formalism (SUG) used by the parser SUPP that permits optional components in rules.

As well as the capability of our system in the parsing of noun phrases, verbal chunks and prepositional phases, it is possible to add linguistic phenomena resolution modules to expand its power and to produce a more correct parsing. One of these tested phenomena is anaphora studied in .

Also, we cannot forget the easy way to work with any kind of corpus or tagger. Simply, we have to modify the tag interface (Figure 3). To illustrate this, we have used the same strategy with the Spanish corpus Blue Book, with 9407 words in 245 sentences (medium length of a sentence is 38.4 words), achieving precision and recall values of 88% and 87% for noun and prepositional phrases and 86% and 89% for the verbal chunks. The decrease of this values with reference to LEXESP is justified by the bigger length of the sentences.

Figure 5 shows a table comparing SUPP's parsing speed with other systems. We should remark that this measure has been obtained with an interpreted Prolog (LPA WinProlog) while other systems use compiled computing languages such as C and Pascal. Also, our system resolves the phrasal coordination. In LEXEPS there are 2558 phrasal coordination and in Blue Book there are 303 phrasal coordination.

## 6. Conclusion

We have developed a complete system of partial analysis capable of extracting coordinated noun and prepositional phrases and verbal chunks in an efficient way. We emphasize its universality because it can be adapted to different taggers due to the existence of a interface that can be defined for each set of tags, as well as the possibility of solving linguistic phenomena in subsequent analysis phases through the semantic information incorporation. The output of this parser can be employed as input in different applications: information extraction and information retrieval, the joining of syntactic trees for the resolution of complete syntactic analysis, as well as the process to obtain the logical formula through semantic analysis.

## Acknowledgements

This paper has been supported by the Comisión Interministerial de Ciencia y Tecnología with project number TIC97-0671-C02-01/02.

## References

ABNEY, S. (1990) Rapid Incremental parsing with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research,* Waterloo, Ontario. pp. 1-9

ABNEY, S. (1991) Parsing by chunks. In *Principle Based Parsing.* In ROBERT BERWICK, STEVEN ABNEY, AND CAROL TENNY, EDS., Kluwer Academic Publishers.

ABNEY, S. (1996) Partial parsing via finite-state cascades. In *Workshop on Robust Parsing (ESSLLI'96),* JOHN CARROLL, EDITOR. pp 8-15.

ABNEY, S. (1997) Part-of-Speech Tagging and Partial Parsing. In *Corpus-based Methods in Language and Speech Processing*. S. YOUNG AND G. BLOOTHOOFT, EDS., Kluwer Academic publishers. The Netherlands. pp. 119-136.

BOURIGAULT, D. (1992) Surface grammatical analysis for the extraction terminological noun phrase. In *International Conference on Computational Linguistics (COLING-92*). Vol. III, pp. 977-981.

BRILL, E. (1993) *Transformation-Based Learning*. PhD thesis. Univ. of Pensilvania..

CARROLL, J. AND BRISCOE, T. (1998) A Survey of Parser Evaluation Methods. In *Proceedings of the First Workshop On The Evaluation of Parsing Systems, in the First International Conference on Language Resources and Evaluation, LREC'98.* Granada, Spain. pp 1-5.

CHURCH, K. (1998) A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Processing Proceedings of the 2nd Conference on Applied Natural Language Processing ANLP'88*. Austin, Texas.

EJEHERD, E. AND CHURCH, K. (1983) Finite state parsing. In *Papers from the Seventh Scandinavian Conference of Linguistics*. FRED KARLSSON, EDITOR, Hallituskatu 11-13, SF-00100 Helsinki 10, Finland. University of Helsinki, Department of General Linguistics. pp. 410-432.

EJERHED, E. (1998) Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*. Austin, Texas.

FERRANDEZ, A., PALOMAR, M., AND MORENO, L. (1997) Slot Unification Grammar and Anaphora Resolution. In *Proceedings of Recent Advances in Natural Language Processing, RANLP'97.* Tzigov Chark, Bulgaria. pp. 294-299.

FERRANDEZ, A., PALOMAR, M., AND MORENO, L. (1998) Anaphor resolution in unrestricted texts with partial parsing. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, ACL'98 - COLING'98.* Montreal, Canada. pp. 385-391.

HINDLE, D. (1983) *User manual for Fidditch*. Technical Memorandum #7590-142, Naval Research Laboratory.

HINDLE, D. (1984) A parser for text corpora. In *Computational Aproaches to the Lexicon*. Zampolli, A. (ed.),. Oxford University Press, Oxford/New York.

JOSHI, A.K. AND SRINIVAS, B. (1994) Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *International Conference on Computational Linguistics (COLING-94)*. Kyoto, Japan.

KARLSSON, F., VOUTILAINEN, A., HEIKKILÄ, J. AND ANTTILA, A. Eds. (1995) *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text.* Mouton de Gruyter, Berlin & New York.

KENNEDY, C. AND BOGURAEV, B. (1996) Anaphora for everyone: Pronominal Anaphor resolution without a Parser. In *Proceedings of the 16th International Conference on Computational Linguistics .(COLING'96)*. Copenhagen, Denmark.

MORENO, L., PALOMAR, M., MOLINA, A., FERRANDEZ, A. (1999) *Introducción al Procesamiento del Lenguaje Natural.* Servicio de Publicaciones de la Universidad de Alicante. 1999.

RAMSHAW, L.A. AND MARCUS, M.P. (1995) Text Chunking using Transformation-Based Learning. In *Proceeding of the Third Workshop on Very Large Corpora of Association for Computational Linguistics (ACL-95)*. pp. 82-94.

ROSS, I.C. AND TUKEY, J.W. (1975) Introduction to these Volumenes. In *Index to Statistics and Probability*. The R & D. Press, Los Palos, CA. pp. iv-x.

VILAIN, M. AND PALMER, D. (1996) Transformational-Based Bracketing: Fast Algorithms and Experimental Results. In CARROLL, J. ED., *Workshop on Robust Parsing (ESSLLI'96)* pp. 93-102.

VOUTILAINEN, A., HEIKKILÄ, J. AND ANTTILA, A. (1992) *A. Constraint Grammar of English. A performance-oriented introduction*. Technical Report Publications 21, Department of General Linguistics, University of Helsinki.

VOUTILAINEN, A. AND PADRO, L. (1997) Developing a hybrid NP parser. In *Proceeding of Fifth Conference on Applied Natural Language Processing (ANLP'97).* Washington D.C., USA.