Integrating Shallow Linguistic Processing into a Unification–based Spanish Grammar

Montserrat Marimon
gilcUB
Grup d'Investigació en Lingüística Computacional
Universitat de Barcelona
`montse@gilc.ub.es`

Abstract

This paper describes to what extent deep processing may benefit from shallow processing techniques and it presents a NLP system which integrates a linguistic PoS tagger and chunker as a preprocessing module of a broad–coverage unification–based grammar of Spanish. Experiments show that the efficiency of the overall analysis improves significantly and that our system also provides robustness to the linguistic processing, while maintaining both the accuracy and the precision of the grammar.

1 Introduction

Deep linguistic processing produces a complete syntactic and semantic analysis of the sentences it processes, however it fails in producing a result when the linguistic structure being processed and/or words in the input sentences fall beyond the coverage of the grammatical resources. Natural Language Processing (NLP) systems with monolithic grammars, in addition, have to deal with huge search space due to several sources of non–determinism (i.e. ambiguity). This is particularly true of broad–coverage unification–based grammars where all dimensions of linguistic information are interleaved, as theories such as HPSG propose. Lack of robustness and inefficient processing make such systems inadequate for practical applications e.g. Natural Language Interfaces (NLI).

This paper presents a NLP system which integrates a linguistic Part–of–Speech (PoS) tagger and chunker (as opposed to data–driven) as a preprocessing module of a broad–coverage unification–based grammar of Spanish.

By integrating shallow and deep processing the efficiency of the overall analysis process improves significantly, since we can release the parser from certain tasks that may be efficiently and reliably dealt with by computationally less expensive techniques. The integration of shallow processing, in addition, provides the unification–based grammar with larger coverage for syntactic structures and allows us to implement default lexical entry templates for virtually unlimited lexical coverage while avoiding increase in ambiguity.

The system we present is inspired by (Abney, 1992) and it is in accordance with (Srinivas et al., 1997; Ciravegna and Lavelli, 1997; Yoon et al., 1999; Venkova, 2000; Watanabe, 2000; Prins and Noord, 2001; Grover and Lascarides, 2001; Crysmann et al., 2002).

In the following section we briefly present the unification–based grammar. Section 3 describes `latch`, the linguistic tagger and chunker. Section 4 discusses the extensions required by our system in order to transfer the information delivered by the tagger and chunker into the grammar. In section 5 we describe the default lexical entries we have defined. Results on the system performance are provided in section 6. This papers ends by presenting the general conclusions.

2 The Unification–based Grammar

The development of the grammar that served as the basis of our research work was done in the framework of the Advanced Language Engineering Platform (ALEP) (Simpkins et al., 1993) during the project LS–GRAM (LRE 61029) (Schmidt et al., 1996) and it was used in the project MELISSA (ESPRIT 22252) (Bredenkamp et al., 1998) for the first time in an industrial context. The grammar is currently being used in the project IMAGINE (IST–2000–29490). The main goal of the IMAGINE project is to develop software technology that allows the interaction with e–business applications by using a multi-lingual NLI from mobile devices and other appliances.[1]

2.1 Coverage of the Grammar

The range of linguistic phenomena that the grammar handles includes: all types of subcategorization structures, determination (simple and complex), a full coverage of agreement (subject–verb, subject–attribute, agreement within the NP), null–subjects (pro–drop, impersonal sentences), compound tenses and periphrastic forms, clausal complements (completive clauses and indirect questions), control and raising structures, support verb constructions, passive constructions (with the copula, with or without the 'by–agent' complement, and reflexive passive), modifiers of verbs, nouns, adjectives and adverbs, negation, sentential adjuncts, topicalization, relative and interrogatives clauses, surface word order variation, coordination (binary, enumeration and coordination of unlike categories), clitics (clitic–NP alternation, clitic doubling, clitic climbing, enclitics), NPs with no noun–head, non–sentential input strings and special constructions (number, dates, ...).

2.2 The ALEP Architecture

ALEP distinguishes preprocessing operations and linguistic processing operations. The former —Text Handling (TH) and orphographemic analyses— account for surface properties of input text (document formatting, delimitation of textual structural elements, orthographemic aspects of morphology), while the latter —parsing and refinement— deal with its non–surface properties (morphosyntactic analysis, constituent structure, semantic representation).[2] A special rule–based operation —Lifting— interfaces the output of the preprocessing operation with the parsing operation.

2.3 The ALEP Linguistic Formalism

The ALEP linguistic formalism has been developed on the basis of the specifications resulting from the ET–6 design study (Alshawi et al., 1991). It is a so called "lean" formalism compilable into first–order (Prolog) terms and thus avoiding computationally expensive formal devices.

An ALEP grammar is implemented by specifying lexical entries and grammar rules, based on a type system that constitutes a monotonic simple type hierarchy with appropriateness conditions.

Lexical entries are based on the data structure Linguistic Description (LD), collecting constraints on the type system. The lexical component of our grammar plays a crucial role in the grammatical description needed for processing. It is a highly lexicalized grammar where linguistic phenomena, such as subject–verb agreement, subcategorization, modification, control relations, etc., traditionally dealt with by means of specialized phrase structure rules, are treated in the lexicon. Grammar rules are thus reduced to a small set of binary–branching context–free phrase structure rules, which are based on the data structure Linguistic Structure (LS).[3]

The adopted approach in the grammar we present follows HPSG proposals (Pollard and Sag, 1994).

3 `Latch`: The Linguistic Tagger and Chunker

`Latch` was firstly conceived as a lexical disambiguation tool based on analyses promotion/reduction by means of weighted symbolic context rules (Porta, 1996).

It is a lean formalism where lexical information, including fullform, lemma and MorphoSyntactic Description (MSD), is expressed by regular expressions. The pivots of the rules, which specify the tokens to be disambiguated, are sequences of lexical elements that receive a vote on their morphosyntactic analyses. Votes may be positive or negative to promote or to eliminate them, respectively. In addition, a precondition may be expressed in the pivots to specify the type of ambiguity the rule is referred to. Linear generalizations are expressed by means of contextual operators for immediate, unbounded and constrained unbounded contextual conditions.

[1] See `http://www.rtd.softwareag.es/imagine`.

[2] A distinctive feature of the ALEP processing architecture is the division of the analysis task into two sub–tasks: 'parsing', which builds up a complete but shallow phrase structure tree, and 'refinement', which traverses the structure top–down, thus monotonically performing feature decoration, typically with semantic information.

[3] Besides phrase structure rules, a set of word structure rules are applied at the parsing component performing morphosyntactic analysis.

In a further development state, the Latch formalism was extended so that it can also be used to mark chunks (or intra–clausal partial constituents) (Abney, 1996) and use that information for PoS disambiguation. This interaction of PoS disambiguation and partial parsing reduces the effort needed for writing rules considerably and improves results (Marimon and Porta, 2000).[4]

4 Integrating PoS Tags and Chunks into the Grammar

The integration of shallow processing techniques (PoS tagging and partial parsing) is fully supported by the open architecture of ALEP, which allows easy integration of external modules.

Our system requires some changes to the default architecture of the ALEP system where both the TH system and the morphographemic analysis component are replaced by a unique external preprocessing module (Latch). It also requires the lifting component to be extended in order to transfer the information delivered by the external preprocessing module into the high–level linguistic processing components. The changes to be made in the high–level linguistic processing components, however, are very thin: word structure rules have to be extended, but phrase structure rules and lexical entries can be left untouched.

4.1 Text Structure to Linguistic Structure Rules

The integration of both the PoS tags and chunk mark–ups delivered by Latch is done by the lifting component of the ALEP system, which converts them into data structures suitable for deep linguistic analysis.

The lifting component is based on a particular set of rules, the so–called Text Structure to Linguistic Structure (TS–LS) rules.

Three levels are assumed at the lifting component —'M', 'W' and 'S'— which in the default architecture of the system were converted into LDs representing morphemes, fullforms, and the top node establishing the axiom of the grammar.[5] [6] Structure rules, then, are distributed according to the different types of structural units being involved in the parsing operation: 'morphemes to words' (word structure rules) or 'words to sentences' (phrase structure rules).

4.1.1 Lifting PoS Tags

Integrating PoS information in a system like ALEP means defining TS–LS rules propagating the morphosyntactic information associated to fullforms (i.e. PoS tag and lemma) delivered by the tagger to the relevant morphosyntactic features at the lexical entries of the grammar.

The integration of PoS tags into ALEP is done at the level 'M'. By using the lowest tag level to lift the lexical information associated to fullforms, we can propagate the ambiguities which can not be reliably solved by the shallow processing tool to the grammar component, thus ensuring that the accuracy of the grammar remains the same.

(1) shows the rule we defined to lift the tag 'Ncfs-'.

(1)

ts_ls_rule(

$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM}\,|\,\text{LOC:} & \begin{bmatrix} \textit{t\_local} & \\ \text{MORPH:} & \begin{bmatrix} \textit{t\_morph} \\ \text{LEMMA: } \boxed{2} \\ \text{MORPHEME: } \boxed{1} \\ \text{AGR: } \textit{(fem\&sing)} \end{bmatrix} \\ \text{CAT:} & \begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD:} & \begin{bmatrix} \textit{t\_noun} \\ \text{NCLASS: } \textit{common} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix},$$

'M', [POS = 'Ncfs-', LEMMA = $\boxed{2}$], $\boxed{1}$).

4.1.2 Lifting Chunks

Similar to the integration of PoS information, the integration of chunk mark–ups in the ALEP system requires TS–LS rules to convert them into LD data structures used by the linguistic processing components of ALEP.

[4] Latch is currently being used to annotate the 125 million word *Corpus Diacrónico del Español* (*CORDE*) and 125 million word *Corpus de Referencia del Español Actual* (*CREA*) by the Departamento de Lingüística Computational de la *Real Academia Española*. Some results on the first version of the tool can be found in (Sánchez et al., 1999).

[5] Normally, this will be the sentence node, though it can also be any phrasal node when partial input strings are to be processed.

[6] The output of the lifting process is a Partial Linguistic Structure (PLS) where the hierarchical relations between the different structural elements is expressed in terms of week dominance relations.

The integration of chunk mark–ups into ALEP is done at the level 'W'. By integrating chunk mark–ups at the intermediate level, we avoid modifying phrase structure rules which build up a LD on top of the converted LDs: (i) attaching post–head sisters (modifiers and/or complements to the right of the head element), (ii) and/or attaching modifiers and/or specifiers to the left of the head element when the chunk has only been partially recognized. Furthermore, we avoid interference with the set of phrase structure rules which build up the same type of LDs. These rules are maintained to build up nodes that have not been marked up by the preprocessing module.[7]

The system we propose, in addition, integrates into the high–level components of ALEP LDs which do not need to be re-built by phrase structure rules, since, even though they are quite underspecified w.r.t. the head element of the chunk (they only contain information about its part–of–speech), they already specify syntactic and semantic information about the non–head elements that have been attached to the head element.[8] This allows us to deal with low frequent syntactic structures whose coverage by means of our ALEP grammar, though feasible, would increase both the parsing search space and the ambiguity.[9]

(2) shows the rule for adjectival chunks which have the head element and a degree adverb.

4.2 Word Structure Rules

Besides the TS–LS rules we have presented, the strategy we propose also requires unary word structure rules to consolidate the structural nodes provided by the 'lift' operation for the new tags 'M' and 'W'.

These rules, in addition, are in charge of percolating the linguistic information of the head element of the chunk, which is encoded in the lexicon, to the mother node, which already contains information about the non–head elements already attached by the preprocessing tool.

(2)

ts_ls_rule(

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\scriptstyle \textit{ld}}\\
\multicolumn{2}{l}{\text{STRING: }\left\langle \textit{muy interesante}\right\rangle}\\
\ldots & \left[\begin{array}{ll}
\multicolumn{2}{l}{\scriptstyle \textit{t\_local}}\\
\text{CAT} & \left[\begin{array}{l}\scriptstyle \textit{t\_subst}\\ \text{HEAD } \textit{t\_adj}\end{array}\right]\\
\text{SEM} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\scriptstyle \textit{t\_sem}}\\
\multicolumn{2}{l}{\text{INDX}\,|\,\text{IND } \textit{sf(index(\_,}\boxed{1}\textit{))}}\\
\text{MODS} & \left\langle\left[\begin{array}{ll}
\multicolumn{2}{l}{\scriptstyle \textit{t\_sem\_mod}}\\
\multicolumn{2}{l}{\text{REL } \textit{sf(rel(degree,}\boxed{1}\textit{,}\boxed{2}\textit{))}}\\
\multicolumn{2}{l}{\text{INDX}\,|\,\text{IND } \textit{sf(index(nevent,}\boxed{3}\textit{))}}\\
\text{PREDARG} & \left[\begin{array}{l}\scriptstyle \textit{t\_predarg}\\ \text{PRED } \textit{sf(pred(}\boxed{3}\textit{,}\boxed{2}\textit{))}\end{array}\right]
\end{array}\right]\right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right],$$

'W', ['TYPE' = 'CHUNK', CHUNK-TY = 'AX', ADV = $\boxed{3}$]).

5 Default Lexical Entries

Supplementary to the integration of the shallow processing tool, default lexical entries have been implemented in our ALEP grammar to provide robust deep processing.

Default lexical entries are lexical entry templates which are activated when the system can not find a specific lexical entry to apply. Note that having default lexical entries in a system like ALEP increases ambiguity, and, thus, the parsing search space, unless a mechanism is used to restrict as much as possible the templates that are activated. The integration of the tagger, which supplies the PoS information to the linguistic processing modules of our system, allows us to increase robustness while avoiding increase in PoS ambiguity.

There are two basic ways to define default lexical entries. One is to implement underspecified lexical entry templates assigned to each major word class such that, while parsing, the system fills in the missing information of each unknown word (Horiguchi et al., 1995; Music and Navarretta, 1996; Mitsuishi et al., 1998; Grover and Lascarides, 2001). In the other approach, very detailed default lexical entries for each major word class are defined.

The approach we have followed falls under a middle type. We have defined several default lexical entry templates for the different major word classes —verbs, nouns, adjectives and

[7] These rules are applied when parsing words to sentences, whereas lifted chunk mark–ups are dealt with word structure rules (cf. section 4.2).

[8] This strategy, however, requires very specialized TS–LS rules not only w.r.t. the category of the head element (noun, verb, adjective, adverb) but also the number, category (determiner, adjective, adverb, auxiliary, ...) and type (definite, indefinite, ...) of non–head elements.

[9] Examples of such syntactic structures are given in section 6.

adverbs— which cover their most frequent subcategorization frames. These templates, however, are unspecified w.r.t. those features which encode the subcategorization restrictions imposed on their subjects and complements, e.g. marking prepositions, lexical semantics, etc. This information is filled by the application of phrase structure rules.

First experiments testing the effect of our default lexical entries, however, showed that, by covering the most frequent subcategorization frames, we ensured that the accuracy of the grammar —percentage of input sentences that received the correct analysis— remained the same. The precision of the grammar —percentage of input sentences that received no superfluous (or wrong) analysis—, however, was very low, since we could not restrict the lexical template to be activated for each word type.

To improve the precision of the system we extended the PoS tags of our external lexicon (i.e. the lexicon we use for morphosyntactic annotation in `Latch`) so that they included syntactic information about the subcategorized for elements (category, marking prepositions, ...). This allowed us to reduce the number of default lexical templates to be applied.[10]

6 Experiments and Results

The two experiments described in this section were used to evaluate the performance of the integrated system both w.r.t. efficient processing and robustness.

In the first experiment, our goal was to perform a comparative study of the processing time of our ALEP grammar before and after the integration of the PoS tagger and chunker. For this experiment, therefore, we required testing cases which were already fully covered by our grammar before the integration of the tagger and chunker. In this experiment, we used a subset of the test suites we have used in the LS–GRAM and the MELISSA projects.

In the second experiment, our goal was to investigate to what extent the ALEP grammar benefited from the default lexical entries in terms of robustness. In this experiment, we tested our system on test corpus which was selected randomly.[11]

a)– Experiment A

To evaluate the efficiency of the system, we defined two test suites and run them with our ALEP grammar both before and after the integration of the shallow processing tools.[12]

The first test suite included short instructive sentences or queries from the corpus of the MELISSA project[13] and sentences we selected from the different test suites we have used for diagnosis and evaluation purposes in the LS–GRAM and the MELISSA projects.[14] Test cases were selected according to: (i) the syntactic function of the chunk e.g. subject, complement and adjunct, for nominal chunks, complement and adjunct, for adjectival chunks, etc.; (ii) the position of the chunk in the sentence, and (iii) the category and the number of non–head elements. This test suite included 1500 cases.

In running the test suite with the new system, processing time of the overall process improved an average of 65% due to the reduction of both lexical ambiguity and sentence length.[15]

Once positive results were achieved with such type of sentential structures, we evaluated our system with much more complex sentences, showing a high interaction of phenomena. For this, we used an article —from the newspaper

[10] This information was not manually encoded, but it was extracted from the lexical resources developed in the project PAROLE (Melero and Villegas, 1998).

[11] Test suites and corpora are the two tools traditionally used for evaluating and testing NLP systems. The main properties of test suites are: systematicity, control over data, exhaustivity, and inclusion of negative data. Test corpora, by contrast, reflect naturally occurring date (cf. (Lehmann et al., 1996)).

[12] Experiments have been run in a 128 Mb Ultra Sparc–10. Mean CPU time values were calculated for 50 samples.

[13] NL utterances which users made in interacting with ICAD, an administrative purchase and acquirement handling system, employed at ONCE (Organización Nacional de Ciegos de España), dealing with budget proposals and providing information to help decision makers.

[14] These test suites are organized on the basis of a hierarchical classification of linguistic phenomena. Test suites including cases with interaction of phenomena and negative cases are also included.

[15] The reduction of the sentence length is due to the fact that elements that are wrapped together in a chunk by the preprocessing module are lifted to the parsing component of the grammar as a unique element.

"El Diario Vasco"— of 250 words from the LS–GRAM corpus.

Two experiments have been carried on, first by integrating the PoS tags into ALEP and then the chunk mark–ups. For the first experiment, the reduction of morphosyntactic ambiguity an average of 0.40 reduces the processing time of the overall process by 45.9% (35.9% on average per sentence). For the second experiment, the system processing time is reduced by 52.6% (an average of 42.7% per sentence). Here, parsing speed–up is due to the fact that by integrating chunk mark–ups, we do not only avoid generating irrelevant constituents not contributing to the final parse tree but we also provide part of the structure that the analysis component has to compute.[16]

b)– Experiment B

The evaluation of the effect of default lexical entries on the ALEP grammar was done with free input text. Here we used a 300 word article from "El Pais" (September 2001).

In running the second experiment we observed that our first approach ensured that the accuracy of the grammar —percentage of input sentences that received the correct analysis— remained the same, even though 67.7% of major words which appeared in the article was not encoded in the ALEP lexicon. The precision of the grammar —percentage of input sentences that received no superfluous (or wrong) analysis—, however, was be very low, we got an average of 8 analysis per sentence. By adding framing information to the PoS tags of our external lexicon we reduced overgeneration up to an average of 2.5 analysis per sentence.

Besides, our system provides structural robustness to the high–level processing. We observed that a number of linguistic structures which could not be handled by the grammar before the integration of the shallow processing tools are currently covered. Examples are:

(3) a. *No dieron* [*crédito alguno*] *a* ... ((they) did not believe in ...)

b. *Se incrementarán en* [*los próximos ocho meses*] (They will be increased in the following eight months)

(3.a) shows a nominal chunk where the indefinite *alguno* is postponed, (3.b) shows a nominal chunk where the canonical 'cardinal + adjective' order is inverted.

7 Conclusions

This paper has described research into the development of engineered large–scale grammar to provide more robust and efficient deep grammatical analysis of linguistic expressions in real–world applications e.g. NLI, while maintaining both the accuracy of the grammar and its precision.

We foresee to extend the chunker to cover ungrammatical or uncomplete intra–clausal partial constituents which can then be integrated into the ALEP linguistic processing components. Also we plan to add semantic information to the PoS+Frame tags encoded in the lexical resources developed in the project SIMPLE.

References

S. Abney. 1992. Prosodic Structure, Performance Structure and Phrase Structure. In *Proceedings of the Workshop on Speech and Natural Language*, Morgan Kaufmann Publishers, San Mateo, CA.

S. Abney. 1996. `http://www.sfs.nphil.uni-tuebingen.de/ abney/Papers.html#96i`.

H. Alshawi, D.J. Arnold, R. Backofen, D.M. Carter, J. Lindop, K. Netter, S. Pulman, J. Tsujii, and H. Uszkoreit. 1991. Eurotra ET6/1: Rule Formalism and Virtual Machine Design Study (final report). Commission of the European Communities, Luxembourg.

A. Bredenkamp, T. Declerck, P. Groenendijk, M. Phelan, S. Rieder, P. Schmidt, H. Schulz, and A. Theofilidis. 1998. Natural Language Access to Software Applications. In *Proceedings of COLING-ACL'98*, Montreal, Canada.

F. Ciravegna and A. Lavelli. 1997. Controlling Bottom–up Chart Parsers Though

[16] A detailed analysis of the results showed us that, while in processing simple sentences, as the ones we included in the first test suite, the most relevant factor for improving processing time was the reduction of the number of tokens of the sentences, in processing complex sentential constructions, e.g. sentences included embedded clauses, efficiency gains were mainly due to the reduction of the morphosyntactic ambiguity, since this drastically reduced the structural ambiguity.

Text Chunking. In *Proceedings of IWPT'97*, Boston, MA.

B. Crysmann, A. Frank, B. Kiefer, H.-U. Krieger, S. Müller, G. Neumann, J. Piskorski, U. Schäfer, M. Siegel, H. Uszkoreit, and F. Xu. 2002. An Integrated Architecture for Shallow Deep Processing. In *Proceedings of ACL'2002*, University of Pennsylvania, Philadelphia, PA.

C. Grover and A. Lascarides. 2001. XML–Based Data Preparation for Robust Deep Parsing. In *Proceedings of ACL–EACL 2001*, Toulouse, Franse.

K. Horiguchi, K. Torisawa, and J. Tsujii. 1995. Automatic Acquisition of Content Words Using an HPSG–Based Parser. In *Proceedings of the NLP Pacific Rim Symposium*, Seoul, Korea.

S. Lehmann, S. Oepen, S. Regnier-Prost, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion, J. Baur, L. Balkan, and D. Arnold. 1996. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING–96*, Copenhagen, Denmark.

M. Marimon and P. Porta. 2000. PoS Disambiguation and Partial Parsing: Bidirectional Interaction. In *Proceedings of LREC–2000*, Athens, Greece.

M. Melero and M. Villegas. 1998. Issues on the Encoding of a Computational Lexicon. In *Proceedings of LREC–1998*, Granada, Spain.

Y. Mitsuishi, K. Torisawa, and J. Tsujii. 1998. HPSG–Style Underspecified Japanese Grammar with Wide Coverage. In *Proceedings of COLING–ACL'98*, Montreal, Canada.

B. Music and C. Navarretta. 1996. Documentation of the LS-GRAM Danish Lingware. Deliverable E-D8-DK, Center for Sprogteknologi, Copenhagen.

C. Pollard and I.A. Sag. 1994. *Head–Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

J. Porta. 1996. Rtag. Technical Report, Grup de Investigació en Lingüística Computacional, Universitat de Barcelona.

R. Prins and G. van Noord. 2001. Unsupervised Post–Tagging Improves Parsing Accuracy and Parsing Efficiency. In *Proceedings of IWPT'2001*, Beijing, China.

F. Sánchez, J. Porta, J.L. Sancho, A. Nieto, A. Ballester, A. Fernández, L. Gómez, E. Raigal, and R. Ruiz. 1999. La anotación de los corpus CREA y CORDE. In *Proceedings of SEPLN'99*, LLeida, Spain.

P. Schmidt, A. Theofilidis, S. Rieder, and T. Declerck. 1996. Lean formalism, Linguistic Theory, and Applications. Grammar Development in ALEP. In *Proceedings of COLING–96*, Copenhagen, Denmark.

N. K. Simpkins, M. Groenendijk, and G. Cruickshank. 1993. ALEP User Guide. Commission of the European Communities, Luxembourg.

B. Srinivas, C. Doran, B.A. Hockey, and A. Joshi. 1997. An Approach to Robust Partial Parsing and Evaluation Metrics. In *Proceedings of IWPT'97*, Boston, MA.

T. Venkova. 2000. A Local Grammar Disambiguator of Compound Conjunctions as a Pre–Processor for Deep Analysis. In *Proceedings of Workshop on Linguistic Theory and Grammar Implementation. ESSLLI–2000*, Birmingham, UK.

H. Watanabe. 2000. A Method for Accelerating CFG–Parsing by Using Dependency Information. In *Proceedings of COLING–2000*, Saarbrücken, Luxembourg, Nancy.

J. Yoon, K.S. Choi, and M. Song. 1999. Three Types of Chunking in Korean and Dependency Analysis Based on Lexical Association. In *Proceedings of ICCPOL*.