Natural Language Engineering **8** (2/3): 121–144. Printed in the United Kingdom

Robustness beyond Shallowness: Incremental Deep Parsing

S. AIT-MOKHTAR, J.-P. CHANOD, C. ROUX
Xerox Research Centre Europe,
6, chemin de Maupertuis
38240 Meylan, France.

Abstract

Robustness is a key issue for natural language processing in general and parsing in particular, and many approaches have been explored in the last decade for the design of robust parsing systems. Among those approaches is shallow or partial parsing, which produces minimal and incomplete syntactic structures, often in an incremental way. We argue that with a systematic incremental methodology one can go beyond shallow parsing to deeper language analysis, while preserving robustness. We describe a generic system based on such methodology and designed for building robust analyzers that tackle deeper linguistic phenomena than those traditionally handled by the now widespread shallow parsers. The rule formalism allows the recognition of n-ary linguistic relations between words or constituents on the basis of global or local structural, topological and/or lexical conditions. It offers the advantage of accepting various types of inputs, ranging from raw to chunked or constituent-marked texts, so for instance it can be used to process existing annotated corpora, or to perform a deeper analysis on the output of an existing shallow parser. It has been successfully used to build a deep functional dependency parser, as well as for the task of co-reference resolution, in a modular way.

1 Introduction

Robustness is a key issue for natural language processing in general and parsing in particular. It can be broadly defined as the ability for a language analyzer to provide useful analyses of real-world input text, such as web pages, news, technical documentation, e-mail or FAQs. Many approaches have been explored in the last decade for the design of robust parsing systems. Among those approaches is shallow or partial parsing, which produces minimal and incomplete syntactic structures, often in an incremental way. After reviewing such approaches, this paper explores how with a systematic incremental methodology one can go beyond shallow parsing to deeper level of language analysis, while preserving robustness. We describe a generic system based on such methodology and designed for building robust analyzers that tackle deeper linguistic phenomena than those traditionally handled by the now widespread shallow parsers.

The rule formalism allows the recognition of n-ary linguistic relations between words or constituents on the basis of global or local structural, topological and/or lexical conditions. It offers the advantage of accepting various types of inputs, ranging from raw to chunked or constituent-marked texts, so for instance it can be used to process existing annotated corpora, or to perform a deeper analysis on the output of an existing shallow parser. It has been successfully used to build a deep functional dependency parser, as well as for the task of co-reference resolution, in a modular way. After introducing the underlying formalism, this paper illustrates how we can handle complex linguistic phenomena such as infinitive control and co-reference resolution. The final section provides some details of the implementation and the overall computational and linguistic performance of the system.

2 Background and Related Work

Since the Chomskyan generative paradigm (Chomsky 1957), linguistic theoretical trends have emerged, and abounded in the 80s, especially unification-based grammar theories (*e.g.* LFG (Kaplan and Bresnan 1982), GPSG (Gazdar *et al.* 1985), UCG (Calder *et al.* 1988), HPSG (Pollard and Sag 1994)). Those theoretical models were designed to describe and explain the principles and properties that govern language in all its aspects, although most studies mainly addressed the syntactic level and proposed formal grammar systems to describe the grammatical structure of language. However, the lack of robustness is the Achilles heel of parsing systems that were built on those paradigms. That weakness led researchers in the field, especially in the last decade, to explore new models and approaches for parsing that produce useful behavior on real-world texts. The concern has both theoretical and practical reasons: on the theoretical side, testing theories with real-world (non artificial) data is, by definition, a requirement for any scientific theory, as far as empirical sciences are concerned. In practice, the availability of huge amounts of electronic text documents raised the need for applications capable of processing those documents automatically, hence capable of parsing real-world texts. The research efforts in that direction have already brought robust parsing systems that successfully contributed either to the automatic acquisition of linguistic resources (Grefenstette 1992; Hindle 1994; Briscoe and Carroll 1997; Faure and Nédellec 1999) or to various NLP applications, such as information extraction (Appelt *et al.* 1993; Hobbs *et al.* 1996; Grishman 1995; Proux *et al.* 2000), translation memory (Gaussier *et al.* 2000) and question answering (Cardie *et al.* 2000; Moldovan *et al.* 2000).

There are many references to 'robustness' in the literature, but little agreement on the exact definition of that notion (Ballim *et al.* 1999; Menzel 1995). However, the idea of processing real-world textual data is widely shared. We think of robustness as the ability of a language analyzer to provide useful analyses for real-world input texts. By *useful analyses*, we mean analyses that are (at least partially) correct and usable in some automatic task or application. That definition implies two requirements:

- First, a robust system should produce (at least) one analysis for any real-world input. The 0-analysis situations are the main problem of traditional parsing

systems and they are not always due to ill-formed or ungrammatical input. In many cases, they are due to grammatical syntactic structures of the input that are not predicted by the linguistic model or the linguistic descriptions of the parser (Chanod 2000).
- A robust system should also limit the number of concurrent analyses it produces or at least give indications on which are the preferred ones. Traditional parsers may produce thousands of possible analyses for long (real-world) sentences, among them many duplicates (spurious ambiguity). Even though the use of packed representations can handle such ambiguity efficiently, the problem of selecting the correct analyses shows up when it comes to parsing-based applications.

Several proposals have been made for measuring the quality of robust parsers (Black 1996; Carroll *et al.* 1998). The analyses of the parser are compared against *treebanks*, *i.e.* corpora annotated with a reference annotation scheme (phrase structure trees) and corrected by humans (Marcus *et al.* 1994; Sampson 1995). The constituent-based evaluation methods measure the similarity between phrase structure or chunk trees by computing the number of matching brackets (Grishman *et al.* 1992; Magerman 1995; Collins 1996), while dependency-based methods compare functional dependency relations and compute *precision* and *recall* over them (Lin 1995; Srinivas *et al.* 1996; Carroll *et al.* 1998). In the latter case, the reference dependencies are often automatically extracted from treebanks (Lin 1995; Basili *et al.* 1999) when they exist for the processed language.

2.1 On the Road to Robustness

Attempts to reach a certain degree of robustness in parsing can be classified into three main trends. The first trend consists of the extension of parsing systems based on traditional theoretical models with special (and sometimes front-end) mechanisms in order to recover analysis when the core parsing system fails, or to rank and select the best analyses when the system excessively overgenerates. For instance, (Frank *et al.* 1998) propose an extension of the LFG projection architecture that incorporates ideas from Optimality theory. It improves parser robustness by adding low-ranked fallback rules that deal with common ungrammatical input or marginal constructions. (Ballim and Russell 1994) extend Prolog DCGs for incremental grammar development with a robust parsing method controlled by user-defined performance thresholds. As part of the VERBMOBIL project, (Kiefer *et al.* 1999) begins with an HPSG grammar development environment, and adds mechanisms for filtering out rule applications and computing best partial analyses in case of failure. In the framework of constraint-based dependency parsing, (Menzel and Schroder 1998; Foth *et al.* 2000) propose the use of graded constraints, which can be violated in a controlled way. Other methods integrate lexicalized descriptions and probability information in the parsing process: for instance, (Briscoe and Carroll 1993) describe a probabilistic parsing model for unification-based grammars. The *supertagging* approach of (Srinivas and Joshi 1999), based on the LTAG (Lexicalized Tree Adjoining Grammar) framework, integrates rich descriptions in the form of supertags (descriptions that combine both phrase structure and dependency information) and statistical distributions of supertag co-occurrences collected from annotated corpora.

The second trend is that of probabilistic parsing approaches, which assign the most probable structure to each input sentence (Magerman 1995; Collins 1996; Ratnaparkhi 1998; Charniak 2000), using weighted rules that are automatically extracted from annotated corpora. The rules are assigned probability weights depending on the frequency of the constituent structures and dependency relations they define. That probability information is used to resolve lexical and structural ambiguity. A similar idea can be applied to pure dependency structures. For instance, (Samuelsson 2000) proposes a statistical theory of dependency syntax where the probability of a dependency tree T is the product of all the probabilities of each node's having a specific label and being associated with a specific bag of dependencies, given its daughters in the tree T. Those probabilities are computed out of a treebank, for instance, and are used to define the most likely dependency tree for a given sentence. It is also worth mentioning here the connectionist approach to parsing (Jain and Waibel 1990; Henderson and Lane 1998) which, though basically different from probabilistic parsing, shares with it two important features: robustness is an inherent property of both models and both can be trained automatically from annotated corpora.

Finally, the third trend for robustness in parsing is the class of approaches known as (rule-based) shallow parsing. Shallow parsing is based on the idea of revising ambitions downwards with regard to the depth and completeness of syntactic analysis. The goal is to obtain minimal, underspecified though linguistically motivated syntactic structures that are useful by themselves for NLP applications, but are also viewed as syntactic preprocessing for further deep analysis.

Many parsing systems are hybrid in the sense that they combine different approaches from the above three classes. Nevertheless, our focus in this paper will be on robust rule-based parsing. In the following section, we describe in more details the notion of shallow parsing and give examples of existing shallow parsers.

2.2 Shallow Parsing

Shallow parsing is the incomplete grammatical analysis of texts (also called *light* or *partial* parsing). It produces minimal syntactic structures, where ambiguity is not resolved, nor fully and explicitly represented. The structures generally specified by shallow parsers include phrasal heads and their immediate and unambiguous dependents, and these structures are usually non-recursive. They are referred to as *clusters* (Joshi 1960), *chunks* (Abney 1991; Federici *et al.* 1996), *chains* (Grefenstette 1996) or *segments* (Aït-Mokhtar and Chanod 1997a). For instance, the analysis of the sentence *"Bill saw the man on the hill with the telescope"* will produce a structure similar to the following annotated sentence:

[Bill $_{N}P$] [saw $_{V}$] [the man $_{N}P$] [on the hill $_{PP}$] [with the telescope $_{PP}$]

which does not specify whether *with the telescope* is a modifier of *saw* (that is, Bill used the telescope in order to see the man) or a modifier of *man* (Bill saw a man who has a telescope). Attachment ambiguity is implicit within the flat minimal structures which, in fact, may represent multiple analyses. In the shallow parsing approach, such choices are left to more informed subsequent processes.

Another property shared by most shallow parsers is the use of limited linguistic resources: part-of-speech (POS) categories, morphological features (gender, number, etc.) and sometimes subcategorization.

Shallow parsers have two important advantages that make them suitable for practical, real-world NLP applications. First, *robustness*, which partially results from their under-specification. Second, *computational efficiency*: their speed may vary from dozens to thousands of processed words per second. Speed is necessary for applications such as data mining, dealing with huge amount of texts.

Joshi's parser (Joshi 1960) was certainly one of the first systems having the properties of what are now called shallow parsers, although it was not meant to process large electronic corpora, which were not available at that time. It was also the first application of finite-state techniques to parsing, designed as a *cascade* or *sequence* of finite-state transducers, each one performing a particular analysis task: rule-based POS disambiguation, simple noun phrase (NP) recognition, prepositional phrase (PP) and adverbial phrase (ADVP) recognition and verb cluster (VC) recognition. Clauses were annotated with a non-finite-state, iterative mechanism. The cascade of finite-state transducers allowed a division of the parsing task into subtasks ordered by increasing complexity. Parsers with quite similar architectures have been designed and implemented more recently (Ejerhed and Church 1983; Ejerhed 1988; Abney 1990; Hobbs *et al.* 1996; Grefenstette 1996; Aït-Mokhtar and Chanod 1997a; Ciravegna and Lavelli 1999).

Ejerhed's parser (Ejerhed 1988) is hybrid, extending the stochastic NP recognition of (Church 1988) with non-recursive clause boundary marking using finite-state machines. (Abney 1990; Abney 1991; Abney 1996) describe an incremental parser, Cass, designed as a cascade of finite-state transducers. The whole parsing process is not a strictly finite-state computation, but each transducer is built from regular expressions for recognizing phrases (chunks). When a given pattern matches multiple strings in the input, the longest one is selected. This results in fast parsing and relatively good accuracy. Processing steps include POS tagging, NP recognition, other chunkings (adverbial phrases, PPs, non-recursive verb clusters), simplex (non-recursive) clause recognition and finally function tagging. Cass2 (Abney 1996), the last version of the parser, is faster still (the finite-state cascade treats around 1300 w/s on a SparcStation 1).

ENGCG (English Constraint Grammar) (Karlsson *et al.* 1995) was designed as a shallow functional parser (function tags are assigned to tokens with no explicit attachment). A particular feature of the parser is that it assigns function tags in exactly the same way as POS tags. First, each token is given all its possible POS and function tags, then morphological and syntactic constraint rules are applied to eliminate incorrect tags. Processing is very efficient (around 1550 words per second on a SparcStation10). The non-projective Dependency Parser by (Tapanainen and Järvinen 1997) uses ENGCG for POS tagging but has its own dependency parser based on Tesnière's model (Tesnière 1959). The system is slightly less efficient (200 words/second, which does not include morphological analysis and POS disambiguation). However, it improves ENGCG since the analysis is deeper (outputting explicit dependencies) and more accurate, achieving reported precisions of 95-98% for subject relations and 89-94% for object relations.

Other parsers are based on modified or enriched versions of context-free grammars. The PLNLP system (Jensen *et al.* 1993) proposes a systematic approach to robust parsing. The initial shallow parses are produced by non-prescriptive binary rules and provide a compact representation of structural ambiguity (*e.g.* for PP attachment). Additional functionalities include a multi-pass algorithm for rule relaxation, ways to reconstruct full parses from partial parses and a ranking procedure to sort multiple parses. Post-processing procedures applied to the shallow parses produce richer linguistic representations such as predicate-argument structures. PLNLP grammars were developed for more than 10 different languages and were used in such applications as grammar and style checking and translation.

Finally, Fidditch (Hindle 1983; Hindle 1994) is one of the first efficient and large-scale text parsers (around 1200 words/second on a Sun4). Fidditch assigns phrase structure trees to sentences. The parser is deterministic so that only one parse is produced for each sentence. However, the tree structure may be underspecified in the sense that some phrase nodes are not attached because of ambiguity. The analysis is performed step by step, using a stack of phrase nodes under construction, as in Marcus parser (Marcus 1980). Fidditch has been used in the Penn Treebank Project (Marcus *et al.* 1994) to provide an initial parse of the corpus which was then hand-corrected by human annotators. It has also been used to extract subject-verb and verb-object relation pairs automatically from a 44-million word corpus (Church *et al.* 1989).

It is interesting to note that, while the term "shallow" usually refers to chunkers, some of the systems mentioned above perform regular dependency parsing, in the sense that they produce explicit functional relations between words, even though the output may be partial in the case of unpredicted or ungrammatical input. Examples of such parsers include the FDG parser (Tapanainen and Järvinen 1997), Sextant (Grefenstette 1996), the link grammar parser (Grinberg *et al.* 1995), the IFSP parser (Aït-Mokhtar and Chanod 1997b) and the Chaos parser (Basili *et al.* 1999).

2.3 Incrementality

Incrementality is a common feature of many approaches to shallow and robust parsing, although not always explicitly articulated as such. For instance, the relaxed approach advocated by Karen Jensen in the early days of PLNLP includes the notion that the first phase of parsing produces a preliminary syntactic sketch where certain linguistic phenomena are ignored or deliberately left underspecified. A second stage of syntactic elaboration revisits the information contained in the syntactic sketch and computes deeper predicate-argument structures, using lexical information that was mostly ignored during the first parsing phase. Additionally, the first phase of parsing is itself split into two parsing rounds, the second round allowing for rule relaxation in case the first round does not produce a full parse. Although this two-round mechanism is initially meant to parse ill-formed input (*i.e.* to provide for robustness in the narrow sense of accounting for ungrammatical utterances), it can also account for syntactic phenomena that are grammatical but only recognized if no alternative construction can be produced during the first parsing round.

Later work, largely based on finite-state machinery, is more explicitly based on the notion of incrementality. For instance FASTUS (Hobbs *et al.* 1996) includes different stages such as recognition of fixed expressions, recognition of basic phrases, recognition of more complex groups, and then recognition of patterns related to events of interest. In S. Abney's Cass parser (Abney 1996), cascaded transducers aim first at resolving small questions sequentially, and then at recognizing major constituents before analyzing the details of their internal structure. This strategy is characterized by the notion of islands of certitude, allowing reliable incremental decisions to be made for selected sub-strings. (Neumann *et al.* 2000) introduces a similar view of incrementality with a divide-and-conquer strategy for shallow parsing of German texts. In a first phase only the topological structure of a sentence (*i.e.* , verb groups, subclauses) is determined. In a second phase, phrasal grammars are applied to the content of the various parts of the main and sub-clauses.

One could even argue that the constraint-based approach advocated by (Karlsson 1990; Karlsson *et al.* 1995) is somewhat a mirror view on incrementality, as those constraints encode partial linguistic knowledge that incrementally reduces the scope of initially unrestricted syntactic analyses.

As those earlier systems demonstrate, incrementality is a methodological principle commonly used to build robust, broad-coverage parsers that rely on computationally tractable syntactic descriptions. This may be explained by two major properties that clearly distinguish incremental parsing from more traditional approaches to parsing: *self-containment* and *descriptive decomposition.*

2.3.1 Self-containment

Each incremental parsing operation (incremental rule) is a self-contained operation, whose result depends on the set of contextual restrictions stated in the rule itself. The constraints and restrictions are evaluated in light of the background knowledge available at the time the rule applies in the cascade. If a sub-string matches the contextual restrictions, the corresponding operation applies without later backtracking. If the rule does not match, the current accumulated linguistic interpretation remains unchanged and is passed to the next processing step without further ado. This leads to grammar readability and ease of maintenance.

This differs radically from classical approaches to parsing, where rules encode hypothetical local interpretations of sub-strings, yet to be validated by the production of a full parse. This requires that a series of independently formulated syntactic hypotheses finally combine and account for the whole input string in a consistent way. Such approaches do not *a priori* provide means to monitor the combination of independent local hypotheses. In a sense, it is implicitly assumed that underlying properties of the language, specified or not in the grammar, will spontaneously select and organize correct grammatical paths during the parsing process.

Such uncontrolled parsing schemes lead to well-known undesirable effects, such as combinatory explosion when local interpretations multiply, spurious ambiguity when different sets of rules produce equivalent syntactic interpretations, and finally parse failure, when a missing interpretation for a given sub-string prevents the parser from building a full parse. Such side-effects make it impossible, or at least extremely difficult to build robust grammars based on explicit linguistic descriptions, but in no way are such failures inherent in hand-crafted rule writing *per se*, as occasionally claimed. It is an issue of overall parsing strategy, which itself reflects strong assumptions about the epistemological value of a computational grammar, often seen as a mere illustration of a predefined linguistic theory, rather than a computational device aiming at exploring language constructions in their full diversity.

2.3.2 Descriptive decomposition

The fact that incrementality preserves parsing from the side effects mentioned above (combinatory explosion, spurious ambiguity, parse failure) does not alone account for robustness and broad coverage. To achieve robustness and broad coverage, one needs to take into consideration the great variety of linguistic constructions occurring in real texts. This requires fine granularity in the linguistic description and means to control hundreds of specific descriptions in a tractable way. Incrementality is a good methodology for that purpose. It allows one to break down the linguistic description of a given phenomenon into a large number of autonomous sub-descriptions.

Descriptive decomposition depends partially on distinctions supported by the linguistic theory, but also on distinctions imposed by the limitations of NLP systems themselves. Incremental rules encode observable linguistic facts in light of the background knowledge accumulated at a given stage of the parsing process. Linguistic facts and computers being what they are, an essential part of robust grammar writing has to be devoted to specific contextual restrictions, regardless of their theoretical value. This duality in grammar writing (linguistic observation on the one hand, restriction of the computational expressiveness on the other hand) is essential to robustness. And it is supported straightforwardly in an incremental framework.

Incrementality, as an additional strength, provides a natural framework for encoding underspecified linguistic descriptions: in an incremental framework, the scope and field of validity of any given statement is naturally restricted to sub-strings that have not yet received a competing interpretation in previous statements. The more explicitly constrained occurrences of linguistic phenomena can be described at earlier stages of the parsing process. The description of complex or atypical occurrences can be deferred to subsequent stages without interfering with phenomena earlier described. This leaves the possibility of underspecifying the constraints of deferred interpretations. But underspecification is not as loose as if constraints were stand-alone rather than incremental. As rules apply to strings that did not match competing rules in earlier stages of the incremental process, the underspecified constraints in deferred rules build up on the contraposition of constraints stated in previous steps. In a sense, the accumulated background knowledge is not limited to the linguistic representation built up as incremental parsing goes; it also includes some implicit knowledge derived from the fact that some rules did not apply at earlier stages.

This can be illustrated with agreement control. In various languages, established standard agreement principles state that noun modifiers must agree in number, gender and possibly case with the head noun they modify. Similarly, the subject agrees in number with the verb. Such principles have influenced various syntactic theories, especially in the paradigm of unification-based grammars. However, one may identify various causes for agreement failure (not even considering grammar errors). Such causes include phonological constraints, ellipsis, word order, interaction of syntactic phenomena imposing conflicting constraints, or semantic constraints prevailing over syntactic constraints (*e.g.* in English, such sentences as *three pints is not enough*, *Good and bad taste are inculcated by example* or *The team are full of enthusiasm*). While unification makes the relaxation of agreement control fairly cumbersome, if possible at all, such syntactic relaxation is straightforward in an incremental framework. Preliminary rules such as rule (1) below may impose agreement control to define a subject relation between `np#1` and `vp#2`:

```
rule (1)      subj(#2,#1) = np#1, vp#2    if agree(#2,#1)
```

while later rules, such as rule (2) below, may not:

```
rule (2)      subj(#2,#1) = np#1, vp#2    if ~subj(#2,?)
```

If a finite verb has not been assigned a subject after preliminary subject pickup rules (1) has applied, this finite verb becomes a likely candidate to accept a non-agreeing subject in rule (2). This rule (2) applies provided the verb has not yet received a subject, as stated by the condition: `if ~subj(#2,?)`.

Accepting such a relaxed construction in rule (2) has no implication on those verbs in the input sequence that follow agreement control and have already been assigned a subject by rule (1). The relaxation control does not reside in the constraints attached to the rule itself, but in the incremental organization of the rule system. This radically simplifies rule writing, as the precise and explicit conditions under which agreement control does not hold would be very difficult to enumerate in a computationally tractable way.

Underspecification, and more generally the process of descriptive decomposition as described above, do not imply a rejection of universal linguistic principles. Such generalizations are perfectly acceptable, in as much as they hold in corpora and do not preclude or complicate the description of exceptions and counter-examples. Such principles can still be exploited in an incremental framework, if they usefully restrict the scope of certain parsing stages. But they can be relaxed or ignored in later stages, if language as produced in real texts so requires.

One may regret that the importance of theoretical assumptions is somewhat minimized by the flood of erratic linguistic phenomena that abound in real text and do not conform to axioms but, in the end, this is what robustness is all about. In light of corpus analysis, the status of principles advocated by major linguistic theories appears to be more relative than expected, as such principles apply to core selected constructions rather than in full generality. While a relativization of linguistic principles is sometimes perceived as a weakness from a theoretical perspective, especially as it weakens the hope to promote an axiomatic approach to language processing, we claim that such a relativization gives principles their full epistemological value. Incrementality provides a natural way of restricting linguistic principles to limited and validated domains beyond which they no longer apply.

2.3.3 From shallow to deep parsing

As previously indicated in the section on background and related work, early work in robust parsing concentrated on shallow (*i.e.* partial or incomplete) parsing. Shallow parsers provide some level of syntactic information regarding the structure of the input string and, occasionally, the relations that hold between such structural elements. Notwithstanding the incompleteness of such information, it represents a gain with respect to the absolute shallowness of the input string. As once observed by J. Hobbs: “We were struck by the strong performance in MUC-3 that the group at the University of Massachusetts got out of a fairly simple system (Lehnert *et al.* 1991). It was clear they were not doing anything like the depth of preprocessing, syntactic analysis, or pragmatics that was being done by the systems at SRI, General Electric, or New York University. They were not doing a lot of processing. But they were doing the right processing for the task.”

Contrary to traditional parsers, shallow parsers have consistently departed from dominant linguistic theories. This may have led to the notion that robust parsing is doomed to remain shallow. In that view, robust and broad coverage parsing is sometimes seen as a result of overlooking complex linguistic phenomena. It would simply provide easy solutions to simple sub-problems.

In fact, very shallow parsers, restricted to chunking operations, or relying on poor and inextensible background lexical information, may not be able to scale up and tackle more complex phenomena. But such restrictions do not hold for parsers that achieve higher expressive power and can handle rich data structures. Especially in an incremental framework, deeper linguistic analysis is always made possible provided:

- no crucial information is lost during the preliminary parsing phase ;
- finer-grained background information, (*e.g.* sub-categorization features, corpus-based evidence) is accessible ;
- deeper linguistic relations can be derived based on a set of core relations extracted during a preliminary phase of robust analysis.

Indeed, the level of abstraction produced during a first phase of robust parsing, *e.g.* by representing the input as a set of dependency relations, facilitates the description of deeper syntactic relations. At this stage, the complexity and variability of the input string has a reduced impact on the complexity of the task. Accumulated syntactic information can be derived from a set of abstract relations, that are less sensitive to such surface phenomena as word order, long distance, embeddings, interference of modifiers, adverbials and non-essential complements.

The same incremental methodology can now apply to a somewhat normalized representation of the surface form. New, fine-grained relations can be derived from relations obtained at an earlier stage, whenever specific contextual restrictions hold. Again, it may be that not all the relevant syntactic information will be computed for any given input, at a given processing stage. But nothing in the incremental process prevents one from describing deep phenomena, such as infinitive control, paraphrastic construction, or co-reference, at least in those selected contexts where the necessary background information is available. And this, at no cost with respect to robustness.

3 Incremental Deep Parsing

Following the idea that the robustness of rule-based shallow parsing is not tied to its shallowness but mostly to the incrementality of its linguistic descriptions and subtasks, we have designed and implemented a generic system for building robust deep parsers. Incrementality is implemented as ordered rule layers and guarantees robustness. The input sequence goes through the rule layers, getting enriched when a rule applies, and passing through otherwise, like in cascaded finite-state parsers (Abney 1996; Grefenstette 1996; Aït-Mokhtar and Chanod 1997a). However, the rule formalism has been designed specifically for deep parsing and can handle rich and fine-grained lexical and dependency descriptions via feature lists. Linguistic descriptions are organized in ordered modules, depending on their depth level. Modularity facilitates the maintenance of linguistic data and makes the system easily customizable or reusable. For instance, a co-reference resolution module has been added (Trouilleux 2001) upon a dependency parsing module for French, using the same rule formalism.

3.1 Overview of the system

Figure 1 shows the global architecture of the system. Its input are sequences of linguistic objects (ranging from raw text to POS-tagged tokens or constituent structures) from which it produces a deep analysis. The analysis consists of a set of dependency relations between those objects, by applying dependency rules defined by a grammarian. By dependency, we mean any linguistic relation between n words or groups of words. The most obvious particular case is functional dependency relations. Before performing dependency analysis, the system has three optional modules (tokenization and morphological analysis, POS disambiguation and chunking) which are activated depending on the type of input.

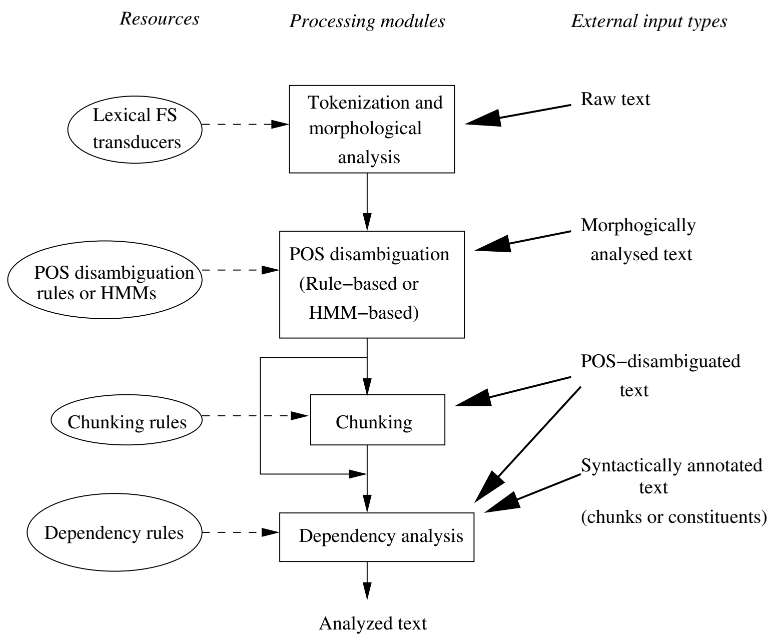
Resources
Processing modules
External input types
Lexical FS transducers
Tokenization and morphological analysis
Raw text
POS disambiguation rules or HMMs
POS disambiguation (Rule−based or HMM−based)
Morphogically analysed text
Chunking rules
Chunking
POS−disambiguated text
Dependency rules
Dependency analysis
Syntactically annotated text (chunks or constituents)
Analyzed text

Fig. 1. Architecture of the Incremental Deep Parsing System

3.2 A Generic System for Various Inputs

One of the main characteristics of our system is its ability to take as input different kinds of linguistic objects:

1. A raw ASCII text, possibly enriched with mark-up tags.
2. A sequence of tokenized and morphologically analyzed words. The words are ambiguously labeled with lexical readings. A lexical reading is a sequence of features that includes part-of-speech and possibly morpho-syntactic features such as number, gender, or subcategorization features.
3. A sequence of disambiguated words, where each word has one single lexical reading. The disambiguation may be performed with any POS tagger.
4. A sequence of constituent structures such as NP, PP, VP, S, etc. A sentence does not need to have a constituent structure where all the constituents are grouped into a single top node. Besides, the constituents may correspond to traditional phrase-structure constituents as well as to chunks (core constituents).

Hence, our system is able to handle an input sequence produced by any morphological analyzer, POS tagger or any full or partial constituent parser. Its input may also consist of POS or syntactically annotated corpora, such as the Brown corpus (Kucera 1992) or the Penn treebank (Marcus *et al.* 1994). The only condition is that the input sequence complies with the XML DTD that is predefined in the system. As partial or shallow parsers that produce chunked structures for raw

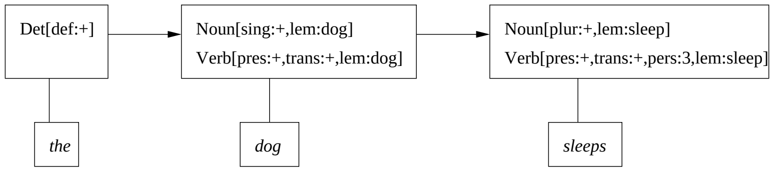
Det[def:+]
Noun[sing:+,lem:dog]
Verb[pres:+,trans:+,lem:dog]
Noun[plur:+,lem:sleep]
Verb[pres:+,trans:+,pers:3,lem:sleep]
the
dog
sleeps

Fig. 2. Internal representation of morphologically analyzed text

text are now widespread, the system's ability to process those structures is particularly suitable. It allows going beyond shallowness to a deeper syntactic analysis, featuring explicit functional relations between words, while preserving robustness. The system can also be used for inter-sentential analysis, as has been shown for co-reference resolution.

3.3 A Common Representation of the Input Sequence

The basic input sequence to the parser represents any segment of text, typically a sentence, but not necessarily (*e.g.* for the task of co-reference, the basic sequence is a paragraph or even the whole text). The basic input sequence is described by specific grammatical constraints, which identify the sequence delimiters (*e.g.* sentence or paragraph delimiters) used to split the input stream.

Whatever the type of input is, *i.e.* whatever the level of pre-processing performed on the input string and whatever the scope of the delimiters defined for the input sequence, the system initially relies on a unified representation: a sequence of constituent trees. The nodes of the trees are labeled with POS or constituent labels, and are associated with a bag of features, in the form of attribute-value pairs. Initially, the internal trees correspond to the input trees when the input is of type (4). When the input is of type (1), (2), or (3), each input word is assigned a pre-terminal node, *i.e.* it is mapped to a lexical tree where the root node is labeled with one or more POS labels together with their respective features, and the single leaf node is labeled with the wordform itself (see figure 2).

3.4 Pre-processing Modules

The system includes three optional pre-processing modules that can be activated before the dependency analysis itself. The activation of those modules depends on the nature of the input and on the linguistic strategy selected by the grammarian. Those modules are:

1. A tokenizer and a morphological analyzer based on finite-state lexical transducers (Karttunen 1994), to be used when the input is merely raw text.
2. A POS disambiguation module, optionally used when the input is a sequence of ambiguously tagged words.
3. A rule-based constituent/chunk analysis module, when the input is a sequence of lexical trees with no constituent structure.

Below, we briefly describe the optional disambiguation and chunking modules.

3.4.1 POS disambiguation

This module consists of a stochastic POS tagger (based on HMMs (Cutting *et al.* 1992)) and a rule-based POS disambiguation mechanism. Those two mechanisms can be used sequentially, and the latter can improve POS tagging precision when it comes to long-distance constraints and fine-grained lexical information (*e.g.* subcategorization) that cannot be captured easily with an HMM. The disambiguation rules apply on a single lexical node to keep that node's most likely part of speech, depending on its ambiguity class and the non-limited surrounding context.

3.4.2 Chunking

The role of the chunking rules is to group sequences of categories into structures (chunks) in order to facilitate the dependency analysis. Those rules are organized in layers that apply on the input sequences of categories one after the other. The parser manipulates a stack of lists of nodes, where each level in the stack corresponds to the application of a layer of rules. Each layer pushes onto that stack its own list that comprises two sorts of nodes:

- If a rule matches a sub-sequence of nodes, then the resulting mother node is copied into the list.
- The nodes that are not recognized by any rules are simply directly copied from the list of the previous level to the current list.

Hence, a new mother node is only available to the next layer, not to the current one, which means that a chunking rule is not recursive (*i.e.* it cannot apply more than once on the same part of the input). Therefore, the rules have less generative power than CFG rules. Still, it is possible to define embedded chunks by writing the relevant rules in different layers, since the number of embeddings in natural language constituents is finite in practice.

Two sorts of rules are available for chunking:

1. ID/LP (Immediate Dominance and Linear Precedence) rules (Gazdar *et al.* 1985) that identify partially ordered bags of nodes in the input sequence of categories;
2. Sequence rules that identify ordered sub-sequences of nodes

A layer can only accept one sort of rules, but both sorts of rules may be meshed together in a grammar on the condition that they are defined in distinct layers. Besides, chunking rules may include constraints on the context where they apply and on the feature values of selected nodes.

At each stage of the process, a parallel structure to the stack is built that connects mother nodes to their daughters and sister nodes together. When all layers have applied, a default TOP node is built upon the nodes of the top list to form a full tree.

3.5 Dependency Analysis

The system aims at producing dependency relations between words or chunks. We use the term dependency, though the modeled linguistic relations may go beyond functional or syntactic dependency relations. Such relations are defined by dependency rules that state topological and logical constraints for the relation to hold. The constraints involve linear and structural properties of the constituent trees and/or the set of dependency relations that have been computed prior to the current rule under scope. Hence, the parser has a bipartite input: the initial constituent tree and the incremental set of dependencies. This naturally makes the analysis sensitive to the order of the rules: dependency rules are applied in sequence and rely on the evolving background knowledge stored in the syntactic tree and in the dependency set. However, each rule applies to all input parts that match the conditions specified in the rule, and potentially yields multiple outputs. Additionally, the same dependency relation may be defined within different rules, corresponding to different topological contexts or to different logical constraints, especially as the dependency set increases. Contrary to most linguistic formalisms that define functional relations over trees, such as in LFG (Kaplan and Bresnan 1982), the structural constraints of our dependency rules are not limited to local trees (*i.e.* the current level of the words or constituents involved in the relation). Instead, they may cover any part at any level of the whole tree sequence. We can express complex constraints such as: a node X is followed by a node Y, both part of a local tree with a root node Z, while Z has a root node T which in turn is preceded by another node U. These constraints are expressed with regular tree matching patterns (Thatcher 1987), and may additionally refer to the feature-value pairs of selected nodes.

More formally, the syntax of a dependency rule is the following:

```
|<subtree_pattern>|  if  <conditions>  <d_term>
```

`<subtree_pattern>` is a tree matching expression that describes structural properties of part of the input tree (possibly the whole tree).

`<conditions>` is any Boolean expression built up from dependency terms, linear order statements and the operators `&` (conjunction), `|` (disjunction) and `~` (negation).

`<d_term>` is a dependency term of the form `name<f_list>(`$a_1, a_2, ..., a_n$`)`, where `name` is the name of the dependency relation, `<f_list>` is a list of features associated with that dependency relation, and $a_1, a_2, ..., a_n$ are the arguments.

A dependency rule states that a dependency term `d_term` is added to the set of dependency relations for the current input if `<conditions>` are satisfied within the current set of dependency relations and `<subtree_pattern>` matches successfully a part of the current input. All the arguments of `d_term` should be variables that are also expressed in the conditions and/or the pattern. Thus, the satisfaction of `<conditions>` and/or the successful match of `<subtree_pattern>`, as a side effect, instantiate the arguments of the new dependency term, which in turn is added to the current set of dependency relations. As a special case, dependency rules may also modify or even delete existing dependencies.

Since the application of a rule at a given stage depends on the background information produced by the previous stages, the correct choice of increments is important and is made on the basis of the identification of various linguistic phenomena and the (possibly multiple) configurations of each phenomenon. The incremental order is determined in such a way that basic and simple phenomena are always described before more complex ones. For a given phenomenon, general and default configurations are modeled before more specific cases and exceptions. The management of such incremental rules is relatively tractable, provided that each dependency relation in the analysis output is mapped with the identifier of the rule that generates it. It is indeed easy to examine the behavior of each individual rule by inspecting its output and comparing it with its input, *i.e.* the output of the preceding rules. If a given rule in the incremental sequence of rules produces an unexpected output while all the preceding rules behave correctly, then the property it models is not true, or at least not correctly formalized by the rule expression.

For more details on the syntax and semantics of the rules, we describe four examples that illustrate how dependency rules can be written to define different types of linguistic relations.

3.5.1 Example 1: surface relations and regular tree pattern

Dependency rules for surface syntactic relations usually feature a regular tree pattern for the constituent or chunk trees. The label of their top node represents the constituents. We refer to the features associated with the nodes (or with a dependency) using square brackets. Sister nodes (*i.e.* sharing the same mother node) are separated with a comma. It is also possible to go down and describe the internal structure of a constituent at any level, using curly brackets. The following rule defines a verb-complement dependency relation, as between enjoyed and wine in the chunked sentence:

SC[[NP John] [FV has always enjoyed]] [NP good wine].

The rule defines a dependency of type `vcomp` between 2 words `#1` (*enjoyed*) and `#2` (*wine*) if `#1` is the head of a finite verb chunk (`FV`) that has a `trans` feature (*i.e.* accepts a direct complement), and the `FV` is within an `SC` (clause chunk) followed by an `NP` chunk with no time feature (*i.e.* not a time expression), and the head of which is `#2`. The `vcomp` dependency relation is assigned the feature `dir` (for direct verb complement).

```
|SC{?*, FV[trans:+]{?*,#1[last:+]}},NP[time:~]{?*,#2[last:+]}|
                          vcomp[dir=+](#1,#2)
```

This sample rule has no condition on the current dependency relation set and would derive a relation noted `vcomp[dir](enjoyed,wine)` from the above example sentence.

3.5.2 Example 2: coordination and shared functions

Some dependency relations can be derived from logical constraints bearing on the current set of dependency relations. For instance, let us consider the sentence: *John peels and then eats an apple*, and let us assume that the current set of dependency relations include an object relation between #2 (*eats*) and #3 (*apple*), noted `vcomp[dir](eats, apple)` and a generic coordination relation between #1 (*peels*) and #2 (*eats*), noted `coorditems(peels, eats)`. Then, the following rule

```
if  ( coorditems(#1[trans:+],#2) & vcomp[dir:+](#2,#3) &
                        ~vcomp[dir](#1,?))
                        vcomp[dir=+](#1,#3)
```

infers an object relation between #1 (*peels*) and #3 (*apple*), on the condition that the first coordinated verb is transitive (`#1[trans:+]`) and has not yet been assigned a direct complement (`~vcomp[dir](#1,?)`).

3.5.3 Example 3: infinitive control

An example of a deep syntactic relation is the subject of infinitive verbs (infinitive control). The following rule describes infinitive control by objects, as in *Mary orders Fred to close the window.* If there is a predefined dependency relation of type `vcomp[inf]` (*i.e.* an infinitive complement) between a verb #1 (*orders*) and an infinitive verb #2 (*to close*), and if #1 has the feature `infctrl` set to `obj`, and #1 has another (non infinitive) complement #3 (*Fred*), then #3 is the subject of the infinitive complement #2. This can be stated as follows:

```
if  ( vcomp[inf](#1[infctrl:obj],#2) & vcomp([inf:~]#1,#3) )
                        subj(#2,#3)
```

3.5.4 Example 4: co-reference

The dependency rules can also define inter-sentential relations. The system has been successfully used for the task of co-reference resolution (Trouilleux 2001). An example of such rules is shown below. It shows conditions both on the current relation set and on the constituent structures. Let us assume that `S` is the top node of a sentence, `SC` is the top node of a clause, `FV` is the label of a finite verb chunk, `subj` is the label of the subject functional relation, and `within` is a structural relation that is true if the second argument is embedded within the first argument. Then the rule below selects the possible antecedent candidates for a pronoun.

```
                    |S#3,S{SC{?*,FV{?*,#1[last]}}}|
if  ( subj[imperso:~](#1,#2[pron,clit,p3,indef:~]) & within(#3,#4) )
                        coref(#2,#4)
```

The rule states that if a pronoun #2 is the subject of a verb #1 (and not an impersonal subject), and if there is a word #4 occurring within a sentence #3, expressed by the relation `within(#3,#4)`, then there is a potential co-reference relation between the pronoun #2 and the word #4, given that the structural conditions are satisfied, *i.e.* sentence S#3 precedes the sentence S where the verb #1 is embedded as the finite verb of the main clause. This rule is only a first step towards co-reference resolution. Next steps apply other constraints (*e.g.* agreement control) in order to eliminate unlikely candidates.

4 Implementation

The parser engine has been written in C++. This choice allows for a smooth integration of the parser as a library (a DLL for instance) in any projects that need linguistic tools. It includes preprocessing components such as a tokenizer, a morphological analyzer and a built-in part-of-speech disambiguator, which cane be intertwined with chunking rules. Since the parser sequentially processes a grammar, it does not require a complex algorithm to maintain concurrent analyses in memory. This reduces the memory footprint of the parser, since only one set of linguistic data is used throughout the whole analysis. For instance, the parser does not have to cope with the complex selection of a rule at a given moment; it rather applies those rules one after the other on the syntactic tree. Still, this architecture might face the same efficiency problem as more traditional approaches: a rule may be tried over the tree and fail, which proves to be a costly process. To overcome this problem, the parser relies on a binary coding of features and syntactic categories (Roux 1996; Roux 1999).

5 A Deep Parser for French

Among others, the system introduced in this paper has been used to build a broad coverage dependency parser for French. The input of the parser is a raw text, which is tokenized and morphologically analyzed by the integrated morphological analyzer. The latter is based on finite-state lexical transducers (Karttunen 1994) and produces a sequence of words labeled with (possibly) ambiguous readings. Beside the POS, words are associated with morphological features (number, tense, etc.), and subcategorization features.

5.1 POS Disambiguation and Chunking

The disambiguation module was fed with 292 rules that eliminate unlikely readings. Even though the methodology is the same as in the finite-state POS tagger described in (Chanod and Tapanainen 1995), we took advantage of the richer expressive power of the current formalism and the richer linguistic information available in the form of feature lists. In particular, we made use of the subcategorization features at this stage, for instance in order to disambiguate the frequent word *des* (prep+article or article). The overall precision rate of the disambiguation module is above 98%. The chunking module has 121 rules that describe chunks such NP, PP, FV (Finite verb chunk), SC (Sub-Clause). A chunk is defined as the core part of a traditional constituent phrase that starts at the beginning of the constituent and ends with its head (Aït-Mokhtar and Chanod 1997a).

5.2 Syntactic Dependency Parsing

We have defined a set of 22 syntactic dependencies; each of them may be refined with features. For instance, the dependency subject is marked with the feature passive when the clause is in passive voice. Some of the dependencies are surface dependencies, such as subject, verb complement, verb adjunct, noun modifier, coordination, etc. Others are deeper, such as the infinitive control dependency, shared dependency relations within coordinated items, or the antecedent dependency between relative pronouns and their antecedents. In the current version of the French parser, 199 dependency rules perform the recognition of all those dependency relations. They involve conditions on the structural properties of the words, their linear order, their categories and morphological features, their subcategorization properties and even some rough semantic features (time, location). Subcategorization is exploited as indicative information rather than strong constraints. For instance, in the case of ambiguous PP attachment, priority may be given, as a heuristic, to verb-complement or noun-complement relations, instead of noun-modifier relations.

6 Evaluation

The system has been tested with the grammar of French described above, on a corpus of newspaper articles (from *Le Monde*), containing 7300 sentences of 23 words on average. The parser requires 8 MB of memory (plus 12 MB for the tokenizer/morphological analyzer). It runs at a speed of 1300 words per second on a Pentium II 500. That speed includes tokenization, morphological analysis, POS disambiguation, chunking and dependency parsing.

As for the linguistic performance of the French parser, we have evaluated the precision and the recall of the subject dependency (including coordinated subjects, infinitive control and subject relative antecedent), and direct complements of verbs. For subject, precision and recall were respectively 93.45% and 89.36%, while the figures for verb complements were 90.62% and 86.56%. Using the system for the task of co-reference resolution, (Trouilleux 2001) reports a precision of 81.95% and a recall of 79.95

Since each dependency relation in the output is associated with the identifier of the rule that produces it, it is possible to evaluate the precision of each dependency rule and therefore track incorrect or unreliable linguistic (incremental) descriptions, which helps improving the overall quality of the analyzer.

7 Conclusion and Future Directions

This paper gave an overview of (rule-based) robust shallow parsing approaches, and pointed out incrementality in linguistic descriptions and processing as a widely shared property of those approaches. We investigated that property and argued that it is a key to robustness, which is not tied to shallowness. We described a generic system for robust incremental deep parsing that we have designed and implemented following that observation. We showed how incrementality and the rule formalism allow for the proper and robust handling of deeper linguistic phenomena than shallow/surface syntax. The incremental rules apply on a bipartite but consistent representation of the structure of the input (constituent tree and dependency set). The system accepts multiple input forms, ranging from raw to syntactically annotated texts, which makes it possible to reuse annotated corpora or existing shallow parsers as preprocessors. It allows for linguistically deep and computationally efficient grammars to parse unrestricted texts at high speed. This system opens the way to new research development and practical applications for content analysis. It has already been used over a variety of corpora, for linguistic tasks such as structural disambiguation and co-reference resolution, or applications such as knowledge extraction. Ongoing and future work includes, on the one hand, grammar development for a number of European and non-European languages, and, on the other hand, an extension of our incremental approach towards robust semantic processing, especially targeted at fine-grained knowledge extraction.

References

Abney, Steven P. (1990) Rapid incremental parsing with repair. In *Proceedings of the 6th New OED Conference: Electronic Text Research*, pp. 1–9, Waterloo, Ontario.

Abney, Steven P. (1991) Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny (eds.), *Principled-Based Parsing.* Dordrecht: Kluwer Academic Publishers.

Abney, Steven P. (1996) Partial Parsing Via Finite-State Cascades. In *ESSLLI'96 Workshop on Robust Parsing*, Prague, Czech Republic.

Aït-Mokhtar, Salah and Chanod, Jean-Pierre. (1997a) Incremental Finite-State Parsing. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC, USA.

Aït-Mokhtar, Salah and Chanod, Jean-Pierre. (1997b) Subject and Object Dependency Extraction Using Finite-State Transducers. *ACL workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications.* Madrid.

Appelt, Douglas E., Hobbs, Jerry R., Bear, John, Israel, David and Tyson, Mabry. (1993) FASTUS: A Finite-State Processor for Information Extraction from Real-World Text. In *Proceedings of IJCAI-93*, August 1993, Chambéry, France.

Ballim, A. and Russell, G. (1994) LHIP: Extended DCGs for Configurable Robust Parsing. In *Proceedings of the 15th international Conference on Computational Linguistics (COLING 94)*, Kyoto, Japan, pp 501–507.

Ballim, Afzal, Pallotta, Vincenzo and Lieske, Christian. (1999) *Robust Text Analysis: an Overview.* Swiss Federal Institute of Technology, Lausanne.

Basili, Roberto, Pazienza, Maria Tereza and Zanzotto, Fabio Massimo. (1999) Lexicalizing a shallow parser. In *Proceedings of TALN-99*, Cargèse, July 12th-17th.

Black, Ezra. (1996) Evaluation of Broad-overage Natural-language Parsers. In Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen and Victor Zue, (eds), *Survey of the State of the Art in Human Language Technology.* Cambridge University Press.

Briscoe, Ted and Carroll, John. (1993) Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars. In *Computational Linguistics*, **19**(1), pp. 25–59.

Briscoe, Ted and Carroll, John. (1997) Automatic Extraction of Subcategorization from

Corpora. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC, USA.

Calder, Jo, Klein, Ewan and Zeevat, Henk. (1988) Unification categorial grammar: A concise, extendable grammar for natural language processing. In *Proceedings of the 12th International Conference on Computational Linguistics*, pp 83–86, Budapest, Hungary.

Cardie, C., Ng, V., Pierce, D. and Buckley, C. (2000) Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, pp. 180–187. Association for Computational Linguistics/Morgan Kaufmann.

Carroll, J., Briscoe, T. and Sanfilippo, A. (1998) Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pp. 447–454, Granada, Spain.

Chanod, Jean-Pierre and Tapanainen, Pasi. (1995) Tagging French - comparing a statistical and a constraint-based method. In *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pp. 149–156, Dublin.

Chanod, Jean-Pierre and Tapanainen, Pasi. (1996) A Robust Finite-State Parser for French. In *ESSLLI'96 Workshop on Robust Parsing*, Prague, Czech Republic.

Chanod, Jean-Pierre. (2000) Robust Shallow Parsing and Beyond. In G. Van Noord and JC Junqua (eds.), *Robustness in Language Technology*, pp. 187–204, Kluwer.

Charniak, Eugene. (2000) A maximum entropy inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 132-139, Seattle, Washington, April 29th to May 4th.

Chomsky, Noam. (1957) *Syntactic Structures.* Mouton, The Hague, Holland.

Church, Kenneth W. (1988) A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin, Texas.

Church, Kenneth, Gale, William, Hanks, Patrick and Hindle, Donald. (1989) Parsing, word association and typical predicate-argument relations. In *Proceedings of the International Workshop on Parsing Technologies*, Pittsburgh, Carnegie Mellon University.

Ciravegna, Fabio and Lavelli, Alberto. (1999) Full text parsing using cascades of rules: An information extraction perspective. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway.

Collins, Michael J. (1996) A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 184–191, Santa Cruz, USA.

Cutting Doug, Kupiec Julian, Pedersen Jan and Sibun Penelope. (1992) A Practical Part-of-Speech Tagger. n *Third Conference on Applied Natural Language Processing*, pp. 133–140. Trento, Italy.

Ejerhed, Eva and Church, Kenneth. (1983) Finite state parsing. In Fred Karlsson (ed.), *Papers from the Seventh Scandinavian Conference of Linguistics*, pp. 410-432, Hallituskatu 11-13, SF-00100 Helsinki 10.

Ejerhed, Eva. (1988) Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin, Texas.

Ejerhed, Eva. (1993) Nouveaux courants en analyse syntaxique. *Traitement automatique des langues*, **34**(1).

Evans, R. (1987) *Theoritical and Computational Interpretations of Generalized Phrase Structure Grammar*. PhD, School of Cognitive Sciences, The University of Sussex, Brighton.

Faure, D. and Nédellec, C. (1999) Knowledge acquisition of predicate argument structures from technical texts using Machine Learning: the system ASIUM. In Dieter Fensel Rudi Studer (ed.), *11th European Workshop EKAW'99*, pp. 329-334, Springer-Verlag.

Federici, Stefano, Montemagni, Simonetta and Pirrelli, Vito. (1996) Shallow Parsing and Text Chunking: a View on Underspecification in Syntax. In *ESSLLI'96 Workshop on Robust Parsing*, Prague, Czech Republic.

Foth, Kilian, Menzel, Wolfgang and Schroder, Ingo. (2000) A transformation-based parsing technique with anytime property. In *Proceedings of the International Workshop on Parsing Technologies (IWPT-2000)*, pp 89–100, Trento, Italy.

Frank, A., King, T. H., Kuhn, J. and Maxwell, J. (1998) Optimality theory style constraint ranking in large-scale LFG grammars. In Miriam Butt and Tracy H. King (eds.), *Proceedings of the LFG98 Conference*, University of Queensland, Brisbane.

Gaussier, E, Hull, D. and Aït-Mokhtar, S. (2000) Term Alignment in Use: Machine Aided Human Translation. In J. Véronis (ed.), Parallel Text Processing, pp 253–274. Kluwer Academic Publisher, Dordrecht.

Gazdar, G., Klein, E., Pullum, G., Sag, A. I. (1985) *Generalized Phrase Structure Grammar*. Blackwell, Cambridge Mass., Harvard University Press.

Giguet, Emmanuel, Vergne Jacques. (1997) From Part of Speech Tagging to Memory-based Deep Syntactic Analysis. *Fifth International Workshop on Parsing Technologies*, Boston.

Grefenstette, Gregory. (1992) Exploring unexplored contexts for semantic extraction from syntactic analysis. In Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-92), pp. 324–326.

Grefenstette, Gregory. (1996) Light Parsing as Finite-State Filtering. In *Proceedings ECAI'96 workshop on "Extended finite state models of language"*, August 11-12, 1996, Budapest.

Grinberg D., Lafferty J., and Sleator D. (1995). A robust parsing algorithm for link grammar. In *Proceedings of the 4th International workshop on parsing tecnologies*, pp 111–125. Prague, Czech Republic.

Grishman, R., Macleod, C. and Sterling, J. (1992) Evaluating parsing strategies using standardized [arse files. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, pp. 156–161. Trento, Italy.

Grishman, Ralph. (1995) The NYU System for MUC6 or Where's the Syntax? In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*,pp. 167–175.

Heinecke, Johannes, Kunze, Jurgen, Menzel, Wolfgang and Schroder, Ingo. (1998) Eliminative parsing with graded constraints. In *Proceedings of 17th International Conference on Computational Linguistics, 36th Annual Meeting of the ACL, Coling-ACL'98*, Montréal, Canada.

Henderson, J. and Lane, P. (1998) A connectionist architecture for learning to parse. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98)*. University of Montréal, Canada.

Hindle, Donald. (1983) Deterministic parsing of syntactic non-fluencies. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL-83)*, pp 123–128, Morristown.

Hindle, Donald. (1994) A Parser for Text Corpora. In B.T.S. Atkins and A. Zampolli (eds.), *Computational Approaches to the Lexicon*, Oxford University Press, New York.

Hobbs, Jerry R., Appelt, Douglas E., Bear, John, Israel, David, Kameyama, Megumi, Stickel, Mark and Tyson, Mabry. (1996) FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

Jain, A. N. and Waibel, A. H. (1990) Incremental parsing by modular recurrent connectionist networks. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 364–371. Morgan Kaufmann, San Mateo, CA.

Jensen, Karen, Heidorn, George E, and Richardson, Stephen D (eds.) (1993) *Natural language processing: the PLNLP approach*, number 196 in The Kluwer international series in engineering and computer science, Kluwer Academic Publishers, Boston/Dordrecht/London.

Joshi, Aravind K. (1960) Computation of Syntactic Structure. In *Advances in Documentation and Library Science*, vol. III, part 2, Interscience Publishers, New York.

Joshi, Aravind K. A Parser from Antiquity: An Early Application of Finite State Transducers to Natural Language Parsing. In *Proceedings ECAI '96 workshop on "Extended finite state models of language"*, Budapest, August 11-12, 1996, Budapest.

Kaplan, R., Bresnan, J., (1982) Lexical-Functional Grammar: A Formal System for Grammatical Representation. In J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pp. 173–381, MIT Press, Cambridge, MA.

Karlsson, F. (1990) Constraint Grammar as a Framework for Parsing Running Text. In *Proceedings of COLING-90*, Helsinki.

Karlsson, F., Voutilainen, A., Heikkilä, J. and Anttila, A. (1995) *Constraint Grammar. A language-independent system for parsing unrestricted text.* Mouton de Gruyter, Berlin/New York.

Karttunen, Lauri. (1994) Constructing lexical transducers. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan.

Karttunen, Lauri. (1996) Directed replacement. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, Santa Cruz, USA.

Kiefer, Bernd, Krieger, Hans-Ulrich, Carroll, John, and Malouf, Rob. (1999) A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pp. 473–480. College Park, MD.

Koskenniemi, Kimmo, Tapanainen, Pasi and Voutilainen, Atro. (1992) Compiling and using finite-state syntactic rules. In *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92* vol. I, pp. 156–162, Nantes.

Kucera, H. (1992) Brown corpus. In S. Shapiro (ed.), *Encyclopedia of Artifical Intelligence*, vol.1, pp. 128–130. John Wiley & Sons, New York, NY.

Lehnert, W., Cardie, C., Fisher, D., Riloff, E. and Williams, R. (1991) Description of the CIRCUS System as Used for MUC-3. In *Proceedings of the Third Message Understanding Conference (MUC-3)*, pp. 223–233. San Diego, California.

Lin, Dekang. (1995) A Dependency-based Method for Evaluating Broad-Coverage Parsers. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1420–1425. Montréal, Canada.

Magerman, David. (1995) Statistical decisiontree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp 276–283. Cambridge, Massachusetts.

Marcus, Mitchell P. (1980) *A Theory of Syntactic Recognition for Natural Language*, Cambridge Mass., MIT Press.

Marcus, Mitchell P., Santorini, Beatrice and Marcinkiewicz, Mary Ann. (1994) Building a large annotated corpus of English: the Penn Treebank. In Susan Armstrong (ed.), *Using Large Corpora.* MIT Press.

Melcuk, I. (1988) *Dependency Syntax: Theory and Practice.* State University of NY Press.

Menzel, Wolfgang. (1995) Robust Processing of Natural Language. In Ipke Wachsmuth, Claus-Rainer Rollinger and Wilfried Bauer (eds.), *KI-95: Advances in Artificial Intelligence*, Springer, Berlin.

Menzel, Wolfgang and Schroder, Ingo. (1998) Decision procedures for dependency parsing using graded constraints. In Sylvain Kahané and Alain Polguère (eds.), *Proceedings of ColingACL Workshop on Processing of Dependency-based Grammars*, Montreal, Canada.

Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Girju, R., Goodrum, R. and Rus, V. (2000) The Structure and Performance of an Open-Domain Question Answering System. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (Acl-2000)*, pp 563–570. Hong Kong.

Neumann, G., Braun, C. and Piskorski, J. (2000) A divide-and-conquer strategy for shallow parsing of German free texts. In *Proceedings of ANLP-2000.* Seattle, Washington.

Oflazer, Kemal. (1999) Dependency Parsing with an Extended Finite-State Approach. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL).*

Pollard, Carl and Sag, Ivan. (1994) *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Proux, D., Rechenmann, F. and Julliard L. (2000) A Pragmatic Information Extraction Strategy for gathering Data on Genetic Interactions. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB'2000)*, pp. 279–285, La-Jolla, California, August 19-23.

Ratnaparkhi, A. (1998) *Maximum Entropy Models for Natural Language Ambiguity Resolution.* Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Roche, Emmanuel. (1993) *Analyse syntaxique transformationnelle du français par transducteurs et lexique-grammaire.* Ph.D. dissertation, Université de Paris 7.

Roche, Emmanuel. (1997) Parsing with Finite-State Transducers. In E. Roche and Y. Schabes (eds.), *Finite-State Language Processing*, MIT Press, Cambridge, Massachusetts.

Roux, Claude. (1996) *Une méthode de parsage efficace pour les Grammaires Syntagmatiques Généralisées.* Ph.D, Université de Montréal.

Roux, Claude. (1999) Phrase-driven Parser. In *Proceedings of VEXTAL'99*, Venice.

Sampson, Geoffrey. (1995) *English for the Computer: The SUSANNE Corpus and Analytic Scheme.* Oxford University Press, Oxford.

Samuelsson, Christer. (2000) A statistical theory of dependency syntax. In *Proceedings of COLING-2000.* ICCL.

Shieber, S. (1984) Direct parsing of ID/LP grammars. In *Linguistics and Philosophy*, **7**(2), pp. 135–154.

Srinivas, B., Doran, C., Hockey, B. and Joshi, A. (1996) An approach to robust partial parsing and evaluation metrics. In *Proceedings of the ESSLLI'96 Workshop on Robust Parsing.* Prague, Czech Republic.

Srinivas, Bangalore and Joshi, Aravind K. (1999) Supertagging: An approach to almost parsing. *Computational Linguistics*, **20**(3), pp. 331–378.

Tapanainen, Pasi, Järvinen, Timo. (1997) A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 64–71, Washington, D.C.

Tesnière, Lucien. (1959) *Eléments de syntaxe structurale.* Paris: Klincksiek.

Thatcher, James W. (1987) Tree automata: An informal survey. In Alfred V. Aho (ed), *Currents in the theory of computing*, pp. 143–172. Prentice-Hall.

Trouilleux, François. (2001) Thèse de doctorat en Sciences du langage, Université Blaise Pascal, Clermont-Ferrand, France.

Voutilainen, Atro and Tapanainen, Pasi. (1993) Ambiguity resolution in a reductionistic parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 394–403, Utrecht.

Voutilainen, A. and Heikkila J. (1994) An English constraint grammar (EngCG): a surface syntactic parser of English. In Fries, Tottie and Schneider (eds.), *Creating and using English language corpora.* Rodopi.