Natural Language Engineering **1** (1): 000–000. Printed in the United Kingdom
© 1998 Cambridge University Press

Parsing Engineering and Empirical Robustness

Roberto Basili and Fabio Massimo Zanzotto
University of Rome "Tor Vergata",
Department of Computer Science, Systems and Production,
00133 Rome (Italy),
{basili,zanzotto}@info.uniroma2.it

(*Received 9 February 2002*)

Abstract

Robustness has been traditionally stressed as a general desirable property of any computational model and system. The human NL interpretation device exhibits this property as the ability to deal with odd sentences. However, the difficulties in a theoretical explanation of robustness within the linguistic modelling suggested the adoption of an empirical notion.

In this paper, we propose an empirical definition of robustness based on the notion of performance. Furthermore, a framework for controlling the parser robustness in the design phase is presented. The control is achieved via the adoption of two principles: the *modularisation*, typical of the software engineering practice, and the availability of domain *adaptable* components. The methodology has been adopted for the production of CHAOS, a pool of syntactic modules, which has been used in real applications. This pool of modules enables a large validation of the notion of empirical robustness, on the one side, and of the design methodology, on the other side, over different corpora and two different languages (English and Italian).

1 Introduction

The ability of dealing with odd (i.e. ill-formed or simply partial) sentences is largely shown by humans. The human interpretation device is tolerant to phenomena like lack of the lexical information (e.g. foreign words), unknown words (e.g. proper nouns never encountered before), and odd grammatical constructions (e.g. gender disagreement, badly transcribed coordination structures or gaps in the information stream as in remote/telephonic dialogue). The above form of tolerance is what has been recently called *robustness* in NLP. The modelling of such phenomenon within computational devices (as early introduced in (Menzel1995)) is thus more than a relevant research area either for a better linguistic investigation as well as for design of large-scale NLP systems.

Robustness has been traditionally stressed as a general desirable property of any computational model and system. In the software engineering practice, *robustness* is (somewhat informally) defined as *the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental* *conditions* (IEEE1990). Although such a definition is satisfactorily used for any information system it requires more specification when used within a linguistic computational model or in NLP applications. In fact, it is needed a systematic definition of what kind of invalid input is here intended. For example, in syntax a formal definition of ungrammaticality is required. As this notion requires a complete definition of grammaticality, the circularity and moreover the criticality of the latter notion (largely debated within and among linguistic theories) prevent from a systematic analysis. Furthermore, the notion of *stressful environmental conditions* has to be interpreted in the linguistic analysis. The stress comes from two major sources:

- external/exogenous stress that is incompleteness in the context, i.e. missing information in the source sentences or lack of competence in a wider context (e.g. inter-phrasal context or discourse).
- internal/inner/endogenous stress that is wrong/odd/misleading information **in** the sentence. This can relate to *legal* information, e.g. high levels of syntactic and/or semantic ambiguity, or to *illegal* input evidence. As the latter has been above considered as a case of noisy or invalid input, with the stressful information we will refer to the former case where higher ambiguity levels are considered.

Stressful environmental conditions (endogenous and exogenous) are also critical to be formally defined since a comprehensive model at different linguistic levels (even pragmatic) would be required.

As robustness in humans is a typical empirical phenomenon a different definition is required. It should thus be tighter to linguistic observation and consequently not expressed formally, at least in a full way.

1.1 Robustness in NL Parsing

When looking only to NL parsing activity, robustness is possibly more specifically defined. Robustness in parsing is tight both to invalid input and endogenous stress. As ill-formedness is a characteristic form of invalid input, ungrammaticality captures only this first aspect. Endogenous stress is mainly related to sentence complexity where lack of information (e.g. in the lexicon) may be the source of the major failures. When the parser is faced with very complex structures, the result is a lower accuracy in the recognition. Robustness should be achieved by preserving most of the information and this results in the so-called *graceful* degradation of performance (Menzel1995).

The above (somewhat informal) definition of *robustness* in NL parsing has thus to take into account some notion of *performance*. Syntactic parsers indeed fail to systematically propose the *correct* interpretations, especially when exposed to large amounts of textual material. This is strictly true in real application environments. Performance criteria may largely vary among applications, where precision and coverage (two of the mostly used measures) may assume quite different relevance.

A coherent view on robustness for NL parsing is thus tightly related to a relative notion of performance and any attempt to study it should take this into account.

Previous approaches to NL parsing robustness have proposed extensions of the grammatical system (i.e. more rules) or changes in the representation (e.g. probabilistic layers around grammatical frameworks, as in PCFG). One of the more systematic approaches (Menzel1995) relies just on a layered representation and on a modified processing (i.e. preference based reasoning) to enforce autonomy and support expectation driven disambiguation in NL parsing. It is to be also noticed that changes in the representation (especially of the parser output) usually characterized the so-called robust parsers where partial interpretations (e.g. NP chunks) are produced.

Robust parsers are also based on the notion of *under-specification*. Parsers able to expose partial results are inherently more robust in the sense defined before with respect to "monolithic" parsers. Under-specification is obtained as, for example, some information (e.g. PP dependences) is left unanalysed and a not fully connected graph is output (e.g. (Abney1996; Basili et al.2000)).

Finally, robust parsers are usually based on complex parsing architectures where pools of modules are cooperatively applied to the source sentences. They add information (e.g. syntactic labels after POS tagging or syntactic dependencies after lexicalised analysis as in (Grinberg et al.1996)) or, in other cases, they prune and disambiguate over redundant representations (e.g. PP attachment disambiguation over parse forests).

The cooperation among modules usually relies on strategies like "*disambiguate as late as possible*", so that ultimate choices are made only when useful information (syntactic and semantic) is available.

When under-specified representations are adopted it is easier to allow different components to add evidence *incrementally* until disambiguation can be triggered.

The results of such strategy are *modular approaches to parsing*. The parsing process is decomposed in subtasks organized in pools (i.e. cascades or pipelines of individual/independent components). Each module tries to maximally confine part of the overall ambiguity: an example relates to the NP boundaries that in chunking (e.g. (Abney1996)) are detected first. Whenever the modules are able to limit all sources of a given type of ambiguity as soon as possible, all the later phases inherit a simpler representation where more constraints can be applied to reduce complexity.

The adoption of modular approaches to parsing raises the problem of flexible parsing architectures as no specific (deterministic) architecture is good for all cases and domains. A modular architecture is useful if it can be configured according to a specific notion of robustness, well suited for the performance required in the target domain.

Modular parsing architectures require the definition of possibly reusable modules. More robust parser can be obtained by re-configurable architectures/systems. The result is that also design methodologies are important for robustness. A methodology for building re-configurable parsers is more useful if it allows controlling the degree of robustness since the design phases.

The above observations about robustness (sources, limits and their influence on parsing) emphasized the need for a more operational notion of it able to also influence its direct measurement.

1.2 Robustness in NL Parsing: the attempt of an empirical definition

All the above observation did lead us to the following main assumptions:

- Robustness can be hardly defined in a fully formal way, as it requires linguistic and deviant phenomena to be modelled.
- Robustness has characteristics related to endogenous and exogenous stress, or invalidity in the source input.
- Robustness has to deal with performance, and this latter is tightly related to a corpus/domain and to the application/task.

Any serious attempt to deeply analyse and model robustness cannot neglect from all the above assumptions. Although other approaches to robustness have relied upon psycholinguistic analysis of its counterpart in human parsers, we will thus attempt a more data-driven analysis based on empirical evidence and measures.

It should be in fact noticed that robustness is an important issue when large-scale analysis is undertaken. Any large corpus exhibits a "noise" (i.e. deviation from linguistic principles and also non linguistic phenomena) that determines lack in robustness in the underlying parsing system. It is evident how corpus phenomena are difficult to be captured by a given theory. They in fact are irremediably changing throughout different corpora and sub-languages. Although the source theory is by itself very robust (as it is modelled on a subset of human language phenomena that are its final scope), corpora tend to significantly disclose from it. This tends to replicate whenever new corpora are approached.

In order to determine a more usable notion of robustness we can thus try a corpus-centred notion of it. A theory T' is thus *more robust* than a theory T, iff small changes in the corpus , i.e. $\Delta(C)$, implies small changes in the results, i.e. $T'(C)$ or $T(C)$, i.e.

$$\frac{\Delta T'(C)}{\Delta C} < \frac{\Delta T(C)}{\Delta C} \quad (1)$$

where ΔC roughly represents changes from a corpus C to a corpus C'. Notice how the above definition tries to capture the notion of graceful degradation. $T(C)$ here implies some notion of performance of the theory with respect to a data set. Although performance is **strictly** related to the target task, as different applications may require optimisation of different phenomena, we will leave this issue not specified at the moment. It does not prevent our analysis from drawing further (and useful) consequences.

However, it is evident that $T(C)$ is measurable in large only if a NLP system S is employed: S embodies T in its lexicons, grammars and control rules. When we make reference to $T(C)$, we are dealing indeed with a different function, $S(T, C)$, expressing a system S that, according to the theory T, is applied to the corpus C. This has consequences on the definition (1). Robustness of the process $S(T, C)$ can be now rewritten as:

(2) $$\frac{\Delta S(T',C)}{\Delta C} < \frac{\Delta S(T,C)}{\Delta C}$$

where $S(T,C)$ models the performance allowed by S in the application of a given T to C.

Equation (2) is useful as it allows decoupling the theory from its application to the corpus. This emphasizes the role of T and S independently. The performance $S(T,C)$ strictly depends on:

- The linguistic knowledge embedded in T that is its lexicons and rules, e.g. grammars
- The assumptions that T makes about the input and output representation. For example the output of a parser can range from a single (i.e. the best) tree to a parse forest or a redundant syntactic graph.
- The algorithmic assumptions implied by S. First of all, S can (or not) support a specific decomposition of the process in several linguistic levels. Second, it is very sensible to the adopted representation, where either single data structures (like charts) may serve all the process or independent representations are used by different subtasks.

In view of measuring and thus assessing a more precise notion of robustness we can now rely on the above three aspects: when a computational framework is available to design a system S able to include aspects of one (or more) linguistic theory(ies) T and to support large scale performance evaluation over different corpora, a (possibly) relative notion of robustness can be measured and exploited in view of target applications. Several systems S can be obtained via organizations of different architectures (e.g. cascades of different parsing modules). Different theories can be tested via tuning and adaptation of lexicons and grammars. Finally large-scale evaluation should be made available with respect to changes in the corpora or against some of their separate and independent subsets related to different syntactic aspects or built according to different complexity.

The purpose of this paper is to define and study:

- A *parsing framework* for design of systems S that support the application of different theories T (without major revisions). Notice that this does not reduce to defining a general formalism (or generalizing existing ones). Existing formalisms (e.g. feature structures as in HPSG) have been often criticized as they can be weak with respect to robustness: in (Menzel1995) the tight integration among syntactic and semantic language levels in HPSG is seen as a potential source of complexity for robustness. The required full constraint satisfaction (at syntactic and semantic level) can even prevent a suitable management of problematic situations where more flexibility is mandatory. Moreover, formalisms are often divergent and a single unifying formalism is not available. Trends in several NLP application areas (e.g. IE as in (MUC1995; Pazienza1997) suggest that heterogeneous architectures can be often successfully defined.

The definition of such a framework is instead mostly related to the design phase of the target NLP system S, where software infrastructures play a major role.

- A *unifying representation of grammatical information* for the target systems S able to transparently support the intermediate parsing phases.
- A suitable notion of *performance* by which $S(T, C)$ can be modelled. This notion will allow the systematic assessment of robustness in which applications will act as *Turing-like* tests.

The following sections will present, first, the principles underlying the required framework (section 2), some parsing architectures reflecting the framework, i.e. modular and lexicalised parsers (section 3.1), and, finally, experimental evidences derived from the latter (section 4) that will be discussed in the last section 5.

2 A modular, possibly pipelined, and lexicalised architecture for robust natural language parsing

The parsing design methodology should allow the production of systems that can be easily configured in order to achieve the desired degree of robustness. The proposed design methodology is based on two principles: the one inherited from the engineering practice, i.e. the *modularisation*, and the other more proper of the AI field, i.e. the availability of *"self"-adaptable components*. Modularisation imposes a clear separation between the activities performed by each module with evident benefits on reusability (modules are loosely coupled among them). Furthermore, the attention to "self"-adaptable components goes in the same direction. Knowledge-based approaches require an intensive work for tuning the general-purpose tool to the particular application environment (modules are loosely coupled with the knowledge domain).

2.1 Modular approaches: robust redundant voting policies vs. computationally attractive cascades

In the software engineering practice, *modularisation* is suggested as a method for the production of easy-to-reuse pieces of systems, i.e. the modules. In order to be re-usable, these modules have to be characterized by high internal *cohesion* and loose *coupling*. Modularisation speeding up the initial system construction allows concentrating the efforts in fine-tuning the system to the particular application scenario.

Once the modularisation is accepted as an added value of the design approach, the next step is deciding which is the desirable composition of the modules for the tasks that the overall systems are designed for. In the syntactic parsing system study, different approaches have been proposed for combining modules together: parallel vs. pipelined combining methods have been adopted.

Again from the engineering practice, *redundancy* is a well-known method against system failures. Hence, an increase on the "degree" of system robustness can be obtained duplicating the modules devoted to a particular task. This principle has been applied also to syntactic parsing in (Worm and Rupp1998). In (Worm and Rupp1998), syntactic processors implementing different theories/models independently produce competing interpretations of the sentence. A chart based uniform representation is envisaged and a voting mechanism is applied to decide which interpretation should be chosen or to combine different partial analysis. The "competing" parsers differ from the point of view of the information they produce over the input sentence: they range from *deep* parsers based on HPSG formalisms to shallow parsers based on finite-state cascades or HMM rules. A "prefer-the-deeper-analysis-whenever-available" is adopted and the sentence interpretations are obtained mixing together partial interpretations. The combined parser is "robust" in the sense that a reasonable (eventually degraded) response is (generally) produced.

Pipelined approaches (i.e. module cascades) have the disadvantage/advantage to be more deterministic. Processing redundancies are avoided and finite-state-automaton cascades generally adopted (Hobbs et al.1996) and (Aït-Mokhtar and Chanod1997). In the perspective of real world applications where time constraints are important, they result to be more appealing since their computation time is inherently lower with respect to redundant approaches. Moreover, the integration of different approaches in a cascade-fashion is postulated in (Abney1996; Collins1996). In (Collins1996), a stochastic approach is applied over symbolically processed textual material: an intermediate level of phrase interpretation is adopted (i.e. the NP kernels). This work suggests the possibility to integrate symbolic and sub-symbolic approaches. The same mixture exists in (Carroll and Briscoe1998) where sub-categorization frames and statistical parsing approach has been positively integrated.

The computational appealing and the suggested possibility of plugging modules inspired by different theories in the processing chain are nice features of pipeline approaches that can be capitalized in our robust methodology for building up re-configurable syntactic parsers.

The modularisation we want to push here is fine-grained: the components are responsible of the detection of some syntactic phenomenon and are interested on a syntactic representation of the sentence that disburdens their analysis. The observations are translated in requirements for the formalism that has to transfer the syntactic analysis among modules. The unifying formalism must exhibit the possibility of *data encapsulation* and *partial analysis storage*.

2.2 Grammars, Lexicons and "self"-adaptable components

Modularisation design principles (high cohesion and loose coupling) by themselves do not guarantee that the "linguistic" modules are conceived to be reusable in a given operational environment (i.e. sub-language/domain). It is also a wide shared perception that some shallower syntactic material can be produced with rules independent from the domain (for instance the NP-chunking). These latter approaches can lead to the definition of modules that have a low degradation of the performances when exposed to the new working conditions. However, it is a well-known limitation that, in order to obtain accurate syntactic parsers, current methodologies propose domain dependent approaches. The domain dependent resultant parsers are generally based on a wide knowledge of the domain. The challenge in this field is to propose approaches able to learn selective rules with the minimal supervision. This is undoubtedly a positive aspect in the perspective of speeding up the tuning to an operational scenario.

The knowledge based approaches need information on the given application domain in the form of distributional frequencies of linguistic phenomena (Collins1996) or lexicalised rules (Pollard and Sag1994). Generally statistical approaches are supervised: prediction rules are estimated on syntactically annotated corpora (the Penn Treebank (Marcus et al.1993), the Susanne corpus (Sampson1993), etc.). These latter are expensive extensional representations of the grammatical intuitions of the annotators over a large amount of textual material. On the other hand, lexicalised approaches are based on precise intuitions of the grammar writers inspired by "real" corpus textual material. Both the approaches provide high performance products. However, the tuning effort is high since, from the one side, portions of the corpora have to by annotated and, on the other side, grammatical rules have to be hand-written.

Knowledge based approaches are applicable in this framework if the required information can be learnt automatically with a *low* level of human supervision. Therefore, processors based on simple syntactical lexicalised sub-categorization frames (e.g. the verb lemma and the prepositions of the arguments) result to be applicable. In fact, this kind of "unpretentious" information is learnable with unsupervised algorithms (Brent1993; Basili et al.1997).

In the framework we propose that modules like:

- shallow analysers that take decisions over simple and domain independent phenomena

- lexicalised analysers based on syntactic sub-categorization frames and coupled with a weakly supervised learning modules

are more attractive since loosely coupled with the domain. They speed up the production of the system and the satisfaction of the performance (robustness) criteria.

3 Parsing Engineering in the practice: CHAOS, a pool of syntactic processors

The robust methodology for producing syntactic parsers proposed in the previous section foresees:

- the decomposition of the parsing process in (possibly) pipelined activities characterized by *high cohesion* and *low coupling*
- the definition of a uniform formalism supporting data exchange in the fine-grained decomposition

- the setting up of modules characterized by a low coupling with the application domain

In this section, we propose a case study where the above principles are applied in the production of CHAOS, a pool of syntactic parsing modules, which has been used in real applications (as text classification in TREVI (Basili et al.August 1998) and hyper-textual linking in NAMIC (Basili et al.2001)) throughout different domains (finance, sport, medicine, etc.) and different languages (English and Italian). The decomposition principles are discussed in sec. 3.1. The uniform formalism is introduced in sec. 3.2. The module pool is described in sec. 3.3 where grammatical and lexicalised modules are discussed.

3.1 Decomposition principles in CHAOS

The decomposition of a syntactic parsing process into different modules has to be motivated by the effective possibility of identifying sub-components with an high degree of *internal cohesion* and a loose degree of *coupling*. The wide shared assumption that *verbs* control the semantics of the sentence and, thus, their syntactic projections constrain the overall syntactic interpretation can be an interesting inspiring principle for the modularisation. For instance, in the sentence extracted from an economical newspaper article:

The executives and the employees say the Acme company, whose revenues plunged to $783 million for the quarter ended Dec. 31, 2000 from $1.67 billion for the comparable period in 1999, is furiously trying to cut costs.

the role of the verb *plunge* is central in the sub-sentence. If the sub-categorization frame

`( plunge, (Subj) (PP:from) (PP:to) )`[1]

related to the particular realisation were available, interpretations connecting together for example *for the quarter ended Dec. 31, 2000 from $1.67 billion* in a single prepositional phrase are obviously inadmissible. Since verbs play a key role in producing the correct interpretation of the sentence, a module devoted to this kind of phenomena is very appreciated. In fact, during the design activity it allows controlling the performances and thus the satisfaction of the constraints.

If this processor is available, the *loose coupling* principle imposes a first decomposition between processors devoted to the detection of phenomena influenced by the verb syntactic projections and those that are not. Then, since a pipeline is gracefully imposed, it should be decided what should be usefully done before the verb attachment detection and what should be done after. An interesting intermediate level

[1] The represented grammatical realization of *plunge* expects a subject, **(Subj)**, and two prepositional phrases, one with the preposition *from*, **(PP:from)**, and the other with the preposition *to*, **(PP:to)**.

between the words and the sentences is the notion of *chunk* (Abney1996). Chunks are both psycho-linguistically motivated and computationally attractive. These are generally *phrase kernels*, as NPs (Collins1996), whose boundaries can be detected via finite state automata. Furthermore, meaningful portions of these chunks, i.e. their syntactic heads and their potential governors, are emphasized. In our case, chunks are also the sentence fragments for which the spans are not influenced by any verbal syntactic projection. The activity of chunk detection should be done before the verb argument detection, since the knowledge gathered in the chunking phase obviously disburdens the verb argument detection. As already stated in sec. 2, the notion of phrase kernels has been used also in stochastic approach to parsing as in (Collins1996). This does not limit the integrability between symbolic and sub-symbolic modules. Moreover, the same consideration applies for the verb sub-categorization based module since, as argued in (Carroll and Briscoe1998), it does not prevent the effective integration of statistical processors. The un-retrieved verbal argument as well as the NP-modifier detection will have a clear benefit if done after the verb argument detection. The search spaces of the later processors are constrained by relations drawn by the verb argument matcher.

The inspiration principle for the design of the module pool is then the concern of using in the best way the disambiguating power of the verb sub-categorization frames. The module competences are partitioned accordingly and their positions in the pipeline chain are then derived.

3.2 An unifying formalism: XDG

The proposed fine-grained modularisation of syntactic parsing requires a uniform formalism able to, on the one hand, represent partial analysis flowing between the modules and, on the other, show to the (eventually pipelined) modules only the information relevant for the single steps. In fact, processors dealing with the verb argument detection as well as the pp-attachment problem are interested to be exposed to the input as a chain of VP, NP, and PP-kernels where relevant features as the phrase heads and the prepositions of the PPs are highlighted. This nice property is owned by constituency-based syntactic representation scheme as the one inspiring the *charts* underlying the VIT formalism (Worm and Rupp1998). In the software engineering, this *information-hiding* attitude is referred as *data encapsulation.*

However, the constituency-based approach has a limitation: the traditional notion of constituent as a subsequence of words in the analysed sentence. This limits its application in a fine-grained modularised framework. For instance, a pp-attachment resolution module should be free to draw the conclusion that a PP-kernel is the VP-kernel without postulating the structure of the rests of NPs/PPs between the two. A dependency-based annotation scheme (Tesniere1959; Grinberg et al.1996) is more indicated to cope with this kind of problem, but it is not well-suited for information hiding: the nodes of the graph are always words, no encapsulation of the information is foreseen. As an instance, the pp-attachment module has to navigate the structure in order to extract the key information to perform its choices (the preposition and the noun head of the PP-kernel as required by the pp-attachment resolution algorithm presented in (Brill and Resnik1994; Ratnaparkhi and Roukos1994)).

The formalism we have defined is a mixture inheriting the positive aspects of the two (apparently diverging) approaches: the *data encapsulation* and the *partial analysis storage attitude*. The proposed annotation scheme is an extended dependency graph (XDG). It is a dependency graph whose nodes C are *constituents* and whose edges D are the *grammatical relations* among the constituents, i.e.

$$\mathcal{XDG} = (C, D)$$

The $\mathcal{XDG}$ set is completely defined when the node tags, Γ, and the edge tags, Δ, are fully specified, i.e. it will be denoted by $\mathcal{XDG}_{\Gamma\Delta}$. The Γ and Δ tag sets depend upon the level of the syntactic analysis (and the underlying grammatical theory).

The XDG formalism efficiently models the syntactic ambiguity. In general, alternative interpretations for dependencies are represented by alternative $d \in D$. A useful property can be imposed on *xdg*s to select a single (partial) syntactic interpretation. A *planar xdg* is a single (although possibly partial) syntactic reading. *Planarity* (Grinberg et al.1996) interdicts *crossing links*, thus is can be used to select unambiguous sentence fragments. An unambiguous partial interpretation is any planar subgraph of an *xdg*.

3.3 The module pool

A module P of the modular syntactic parser is a processor that, using a specific set of rules R, adds syntactic information to the representation of the sentence, i.e.

$$P : R \times \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma'\Delta'} \tag{3}$$

so that $P(r, xdg) = xdg'$, where xdg and xdg' are the input and the enhanced graph, respectively. This implies that syntactic processors SPs are modelled as functions over XDGs, and their nature is reflected by properties of those functions. As any P_i module foresees the use of its own rule bases (elements in R_i), the first argument of a function P_i can be omitted for sake of synthesis, so that hereafter equation 3 will be written as

$$P_i : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma'\Delta'}$$

with $P_i(xdg) = P(xdg; r_i) = xdg'$.

Actions that a module P perform on the XDG can be *monotonic* or *non-monotonic*. *Monotonic modules* preserve all the choices (i.e. nodes and arcs, as constituents and dependencies already recognized) expressed by the input graph.
Furthermore, with respect to the input XDG, the ability of a module P refer to:

- *constituent gathering*, for processors grouping set of words into larger constituents;
- *dependency gathering*, where nodes are left untouched and only dependencies are added.

Finally, a further distinction can be done with respect to the parameter R_i of each processor P_i. P_i is a *lexicon-driven* processor when R_i is lexicalised (e.g. a verb sub-categorization lexicon, r_i). P_j is a *grammar-driven* processor when R_j does not include any lexicalised form of syntactic information (e.g. categorial or PSG rules).

According to the above definition several processors can be defined. An overall modular parser MP is thus defined as a cascade of processing modules $(P_1, ..., P_n)$, via composition of processors:

$$MP : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma'\Delta'}$$

with

$$MP(xdg) = P_n \circ P_{n-1} \circ \ldots \circ P_2 \circ P_1(xdg)$$

The modules actually used in CHAOS are described in the next sections.

3.3.1 Grammar-driven components

In order to be loosely coupled with the special language, the grammar-driven components should have general (and possibly under-specified) rules. Decisions will be taken by modules having high-expectations on the behaviour of the words (i.e. the lexicalised components). The two grammar-driven components adopted in the CHAOS pool are: (1) a chunker (Abney1996) and (2) a shallow syntactic analyser (Basili et al.1992).

The chunker is the component that has to pack ambiguity independent from verb valency information. It, thus, provides a set of (possibly) complex sentence fragments as kernels of nominal phrases NPK (e.g. *[The executives] and [the employees] say ...*) or prepositional modifiers PPK (e.g. *... plunged [to $783 million] [for the quarter] ...*). Basic information related to a chunk is a syntactic category (e.g. NPK, PPK, etc.), a potential governor and a grammatical handler possibly different from the governor. It recalls quite closely the notion of instance of morpho-syntactic *template* in most dependency based parser. In terms of the XDG notion introduced above, a chunking process matching grammatical rules (the *chunk prototypes*) over a part-of-speech tagged sentence (Γ'={Verb, Noun, Preposition, Adjective ...}) and produces an *xdg* whose nodes are chunks, characterized by a governor, and the syntactic category (Γ={VPK, NPK, PPK, ...}), i.e.:

$$Chunker : \mathcal{XDG}_{\Gamma'\Delta'} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$$

It is a machine computationally complex as a finite-state automaton since it is possible to express the chunk prototypes as regular expressions. The coupling with the domain is small since it postulates and uses only prototypical descriptions of simple structures.

The shallow syntactic analyser aims to draw relations among the chunks without using deep information (sub-categorization lexicons). The grammatical recognition is based on a shallow parsing strategy presented in (Basili et al.1992). A discontinuous logic grammar formalism is here used to model matching of non-adjacent (i.e.

expressed by gaps) modifiers and specifiers. Logical patterns as feature structures are used to express legal realizations of constituents with gaps: *skip* rules are used to express sentence fragments among head and dependents. Such fragments are simply skipped by the parser and left unanalysed although logical constraints (via unification) are imposed to their feature description. The result of the analysis is an XDG enriched with potentially ambiguous grammatical relations. The ambiguity is modelled via a *plausibility* score. In term of the formalism introduced, the shallow dependency parser is:

$$SSA : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$$

where $\mathcal{XDG}_{\Gamma\Delta}$ have chunks as nodes (Γ and Δ are Γ={VPK, NPK, PPK, ...} and Δ={SUBJ, DIROBJ, PPMOD, ...}).

3.3.2 Self-adaptable components

The precision of the whole syntactic parsing can be controlled if high expectations on the word behaviour are postulated. This is generally obtained by using lexicalised rules, i.e. rules activated by particular lexical items. Many of these rules depend tightly on the domain since they capture word meanings. This is particularly true for verbs. For instance, the verb *operate*, in the medical sub-language, can have the meaning of *"perform a surgery on"* and, thus, has the sub-categorization structure `(operate, (SUBJ,PP:on))`. In the finance sub-language it is likely to express the meaning of *"operate in a market sector"* and, consequently, the preferred reading is provided by the frame `(operate, (SUBJ,PP:in))`. This difference can result in very high performance variation. Furthermore, it is important to activate the subpart of the lexicon that can provide improvements in the particular domain. Modules based on sub-categorization lexicons are valuable in this framework if underlying lexicons are re-configurable and tuneable to the particular sub-language. In (Basili et al.1997; Basili et al.1999), the possibility of acquiring this form of knowledge has been demonstrated to be effective in a shallow parsing environment.

Therefore, a specific processor, the Verb Argument Syntactic Matcher,

$$VASM : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$$

is adopted in the pool. It matches verb argument structures and organizes the detected phrase fragments into a hierarchy of clauses. $VASM$ is a lexicalised processor able to work at different levels of lexicalisation that processes $\mathcal{XDG}_{\Gamma\Delta}$ whose nodes are chunks (Γ and Δ are Γ={VPK, NPK, PPK, ...} and Δ={SUBJ, DIROBJ, PP-MOD, ...}). Successful matches add to the target *xdg* dependency arcs also called *icd*s, i.e. *inter-chunk dependencies.* An original feature is the specific combination of the argument matching with the clause recognition. As sentences have more than one verb defining different sentence clauses, the matching of argumental *icd*s also determines the set of detected clause boundaries. In this perspective, coordination and subordination between clauses are approached on the basis of verb argument recognition. The recognition of the complete hierarchy of the sentence clauses is refined incrementally along with the matching of argumental *icd*s for the different verbs (see (Basili et al.1998a) for technical details). In $VASM$, the role of lexical information is not only to fill slots of lexical entries, but also to control, via planarity constraints, the matching for other verbs and the activity of the grammar-driven modules. The kind of suggested analysis has been also adopted for Italian where the relatively free order of arguments in sentence often require control rules to judge among competing slot fillers.

The use of sub-categorization frames introduces a *graceful* correlation among the syntax and the semantics analysis as sub-categorization frames *shallowly* convey the semantics of the words (verbs, in this case). The integration of the two level of analysis allows the propagation of semantic constraints in the later phases of the process of syntactic analysis.

4 Measuring Empirical Robustness

Once a framework for modular and lexicalised parsing has been settled, the study of robustness as it has been defined in section 1.2 can be carried out. The aim of the experiments is:

- to validate some of the proposed parsing architectures,
- to study the contribution of lexicon and grammar-driven modules to the overall robustness as an empirical validation,
- to assess the viability of the proposed approach to robustness and derive general principles about it.

For these targets, several corpora of two different languages (English and Italian) have been studied and contrastive analysis has been carried out. In order to fit the above objectives, a cross-domain analysis has been firstly applied. Different systems (i.e. parsing architectures S embodying different theories T) have been investigated across text collections in different domains. Then a large-scale reference corpus (the Penn Treebank, (Marcus et al.1993)) is used to evaluate the potentials of lexicalised architectures for parsing of English. Partitions of different complexity have been firstly derived. System performances (and thus robustness) are then systematically measured to get a quantitative evaluation of system degradation.

To assess the role of sub-categorization lexicon, two different parsing architectures are contrastively compared. The first lexicalised parser (hereafter referred as *Lex*) is characterized a cascade of chunking, verb argument detection, and shallow syntactic analysis:

$$Lex(xdg) = SSA \circ VASM \circ Chunker(xdg)$$

The verb argument detection, driven by a sub-categorization lexicon, has been carried out by adopting as a source an automatically induced verb lexicon. The acquisition model, described in (Basili et al.1997) has been independently carried out over the entire collections. The results are different lexicons independently adopted during testing. In this case the underlying theory T is changing among domains according to the specific nature of the sub-language, while the system S is expressed by a common architecture (i.e. the chain of processing steps), identical for the different domains.

As a contrastive architecture, a second parser (No_Lex) has been also applied made by the chunker and the shallow syntactic analysis:

$$No\_Lex(xdg) = SSA \circ Chunker(xdg)$$

Major differences between the two configurations, applied to XDG structures, is that the *Lex* parser depends strictly on a domain specific verb lexicon, while the second does not rely on any form of lexical knowledge.

In section 4.1 the first set of experiments is described. Section 4.2 discuss the results obtained from the Penn Treebank. An overall discussion is then carried out in Section 5.

4.1 Evaluating cross-domain Robustness: parsing Italian texts

The evaluation of parsing performances has been possible over three extensive collections of Italian texts:

- *Legal*, an excerpt of legal documents on Italian V.A.T. laws, of about $320,000$ words
- *ENEA*, a collection of technical and scientific papers on the environment (about $350,000$ words)
- *Sole24Ore*, a collection of news from the most important financial newspaper in Italy

These corpora are related to quite different topics, show very different styles and represent a good basis for cross-domain analysis. They all are not annotated. As extensive controlled annotated corpora were not still available at the time of the experiments, resources have been manually derived for them. A pre-existent constituency-based annotated collection of sentences for the *Legal* collection and two *ad hoc* annotated collections for the other domains have been used. Note that the annotated portions represent small subsets of the source collections. The main features of the annotated portions of the three corpora are reported in Tab. 1. The size of the entire source corpora is relevant instead as they influence the quality and coverage of the corresponding acquired lexical information (subcategorization frames).

Table 1. Features of the three annotated corpora

	ENEA	Sole24Ore	Legal
#words	1,149	494	1460
#sentences	56	22	80
av. #words per sentence	(20.51)	(22.45)	(18,25)
av. #verbs per sentence	2.14	3.1	2.2
average chunk length	1.53	1.44	1.54

Although no correspondence emerges between average sentence size and average chunk length (row 3 vs. row 5), both parameters are related to the overall *complexity* of the test sets. It seems that the *Legal* is simpler than the $ENEA$ corpus, while the more complex seems to be the *Sole*24*Ore* corpus.

The F-measure obtained by the two parsers (*Lex* and *No_Lex*) is reported in Fig. 1 for the three domains.

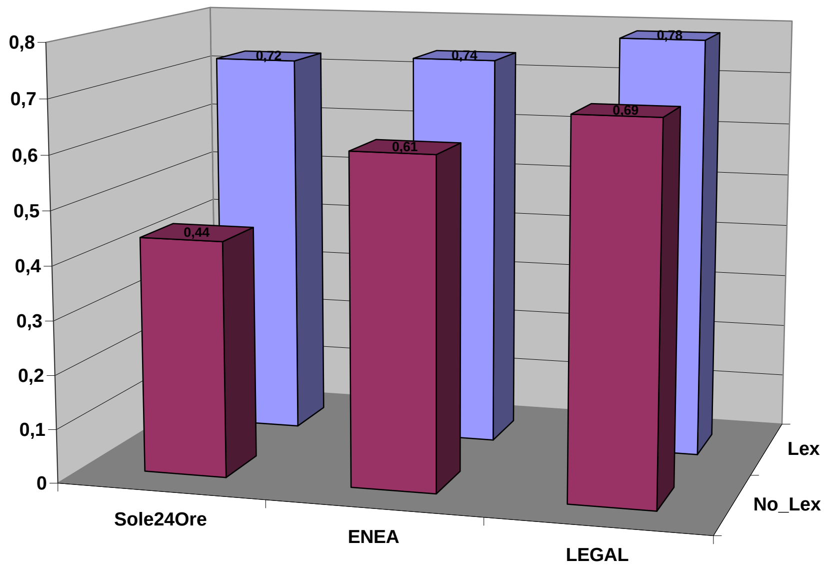

Fig. 1. F-measures of the *Lex* and *No_Lex* parsers over three Italian corpora

Results show that the lexicalised architecture significantly outperforms the shallow *No_Lex* parser. The synthetic data for the *Lex* parser include inter-chunk verb dependencies (argumental if postulated by the lexicon or ambiguous) as well as noun modifiers. Data suggest that chunk analysis provides an effective word grouping: at least two words over three appear in a non-singleton chunk (as shown in Table 1).

The *Lex* system is effective (F-value $> 70\%$) for all the different domains. The cross-domain analysis also shows that the relative complexity of the corpora (*Legal* $<$ $ENEA$ $<$ *Sole*24*Ore*) is also reflected by decreasing performance for both the two parsers. This suggests that the inherent complexity of the corpus can be well captured.

Table 2. Performance on the *Legal* corpus using two lexicons

Icd	Rec (Lex_{LIFUV})	Prec (Lex_{LIFUV})	Rec (Lex)	Prec (Lex)
Argumental	28.7 %	88.5 %	29.9 %	89.1 %
Unambiguous	53.6 %	85.8 %	54.4 %	86.3 %
All	67.4 %	71.6 %	68.1 %	72.1 %

By capitalizing on this we can thus evaluate the degradation in performance of the two parsers along the increase of complexity of the domain. The first major result is thus that equation (2) can be consistently used to express robustness. The outcome of the first experiment is thus that the lexicalised (*Lex*) parser is more robust as its performances do not decrease rapidly at the increase of the corpus complexity. This is strikingly clear if we compare this against the rapid degradation of the *No_Lex* parser.

The *Lex* parser is always fed with a domain specific lexicon, directly acquired on the text collections. Further evidence is thus needed to assess the role that the parser configuration (system S) plays on the resulting robustness. As the evaluation is biased by the lexical information, in order to ultimately validate the effectiveness of the *Lex* architecture over the other parser we should experiment with different lexicons.

As large sub-categorization lexicons are rare resources for the Italian language, we relied on LIFUV (Delmonte1992), a manually compiled lexical knowledge base that encodes syntactic and semantic selectional restrictions in lexicalised frames. The number of verbs covered by lexicalised frames is about #1,500. In order for the *Lex* architecture to use LIFUV, a compiled form with only sub-categorization frames has been extracted from the source syntactic-semantic frames. *Lex* has been thus fed with only syntactic constraints where sense information (expressed in LIFUV via aspectual categories and Jackendoff-like semantic primitives) has been neglected. We will refer hereafter this parser as the Lex_{LIFUV}.

The results obtained for the two parsers over the *Legal* corpus are reported in Table 2. Recall and precision over this corpus have been measured against the two sources lexical information. The different rows refer to argumental *icd*s (i.e. those dependencies postulated by the lexicon that are less frequent, so that their overall recall is low), unambiguous (i.e. dependencies not conflicting with other possible syntactic readings) and all dependencies in the classes: `V-SUBJ, V-OBJ, V-PP, N-PP`.

The performance results of Table 2 are very similar. Not surprisingly the use of data-driven learning provides even a better lexical information. This is a confirming evidence of the viability of lexicalised parsing architectures for NLP applications.

Moreover, the use of different (but consistent) sub-categorization information does not significantly modify performances. It seems that slight changes in linguistic theory, imposed by switching from a syntactic (automatically acquired) lexicon to another (richer) lexicon (i.e. LIFUV), are not impacting on robustness. This suggests that the architecture (shared by the two parsers as a cascade of chunking and lexicalised dependency analysis) is the major responsible for the robustness suggested in the first experiment (shown in Figure 1). Again, the role of the reference system S with respect to robustness is much more important than changes in the grammatical theory T.

4.2 Evaluating Robustness in English parsing

The evaluation of English parsing can rely on large syntactically annotated corpora (e.g. Penn Treebank (Marcus et al.1993) and Susanne (Sampson1993)). A major problem in reusing the annotated material in the evaluation activity is the actual mismatch of the grammatical hypothesis between the reference corpus (i.e. constituency based parse trees) and the parser outputs (i.e. XDGs). In fact, the grammatical hypothesis not only impacts on the reference representation but also on rules used to express grammatical phenomena via annotations.

The required mapping is not easy. Previous works suggested that the underlying grammar increases linearly in number of rules with the number of sentences (Gaizauskas et al.1998). Low error rate in translation can be guaranteed only for some given syntactic relations. Since the aim is to comprehend the effects of the lexical information in parsing, syntactic relations like `V-SUBJ`, `V-OBJ`, `V_PP`, and `N-PP` have been extracted with a suitable degree of confidence. Details of the rewriting algorithm are in (Basili et al.1998b).

One of the main benefits of using the Penn Treebank (PT) is that it provides consistent syntactic data (no grammatical noise) that can be exploited by the machine learning techniques needed for verb sub-categorization frame acquisition. The kind of lexical learning adopted in the *Lex* architecture is thus *optimal* as it is based on supervised data.

The experiments aim to demonstrate that the *Lex* architecture improves the parsing accuracy and robustness. Moreover, automatic acquisition of the sub-categorization lexicon is shown viable for a *bootstrapping* approach to parsing. In order to properly set-up the experiments several assumptions have been made:

- Different phenomena may exhibit different levels of complexity, so that individual measurements have been made of the different syntactic aspects. For example, independent evaluation of specific grammatical relations (e.g. `V-SUBJ`, `V-OBJ`, `V-PP` and `N-PP` has been carried out.
- Performance levels of the proposed architectures have been observed throughout all the available material (about 44,000 sentences in the PT)
- As robustness is to be investigated, we need to observe the behaviour of the system with respect to increasing levels of complexity. This allows quantifying robustness in terms of accurate performance with respect to stressful input.

The grammatical phenomena of interest in evaluating of the *Lex* architecture are those dependent on the verb sub-categorization information. Table 1 focuses on the set of verb dependencies, and provides evidence of the role of the lexicon in their recognition (i.e. `V-OBJ`, `V-OBJ` and `V-PP`). It also shows how the information available for verbs impacts on the ambiguity of strictly related phenomena, i.e. prepositional modifiers of noun (or nominal groups), `N-PP`.

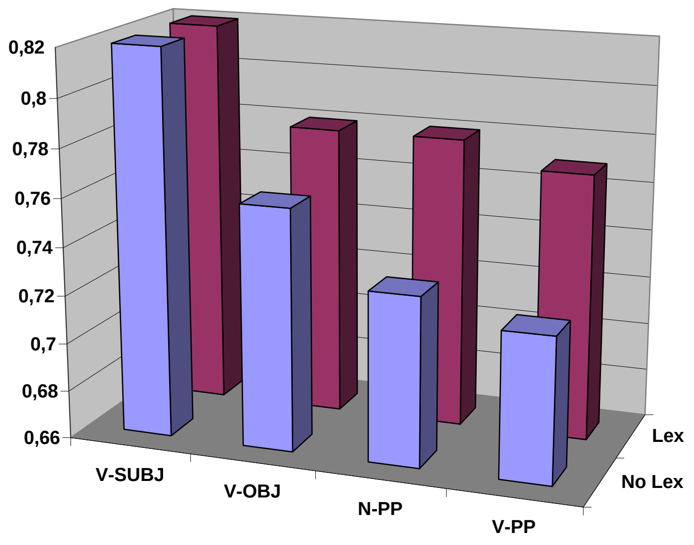

Fig. 2. Performance of Lex and No_Lex Architectures over different syntactic phenomena

In Fig. 2 the values of F-measure obtained for the different phenomena are shown. The performances of the two systems seem in agreement: a decreasing accuracy is shown correspondingly by both the parsers. This suggests a precise relative complexity of the observed phenomena. The different measures have been thus organized according to decreasing levels of the task complexity. As expected, the *Lex* parser is outperforming the non-lexicalised one. Except for the subject recognition task the *Lex* shows about 8-10% better performance on the other tasks.

Table 4.2 shows recall and precision scores of the two parsers on the most complex phenomena, `N-PP` attachments. Note that any decision made according to the verbal lexicon reflects, because of the planarity constraints, on the attachments of PPs to nouns. The Table shows an increase of the precision with a corresponding small loss in term of coverage. The global effect is described by an improvement of the F-measure: $F(\alpha) = 0.73$ without lexicon vs. $F(\alpha) = 0.78$ with lexicon.

In order to take into account robustness a second and more specific experiment

Lexicon	plaus	Link Type	R	P	$F(\alpha = 0.5)$
no	*any*	N-PP	0.85	0.65	0.73
yes	*any*	N-PP	0.82	0.75	0.78

Table 3. noun phrases-prepositional phrases attachment

Table 4. *Size of the the subcorpora*

Subcorpus	Number of syntactic dependencies
C_0	507
C_1	11,746
C_2	27,639
C_3	35,313
C_4	39,163
C_5	40,748
$C_6 = others$	41,799

has been run. Corpus has been split according to sentence complexity, and different sub-corpora have been derived. The behaviour of the two parsers has thus been studied over such subsets that express the different complexity levels of the target sentences. The estimation of the sentence complexity has been defined in order to capture aspects like average number of words, average number of syntactic dependencies and number of clauses. Given a sentence s, its complexity $SentenceComplexity(s)$ is defined by:

$$SentenceComplexity(s) = \frac{\#LV(s) + \#LN(s)}{\#Clauses(s)} \tag{4}$$

where $\#LV(s)$ and $\#LN(s)$ are the number of verbal and nominal links (i.e. V-PP and N-PP) defined by the oracle for s, while $\#Clauses$ is the number of clauses in the sentence s.

Each subcorpus C_i is thus made of the sentences whose complexity is below $i+1$, i.e.

$$C_i = \{s | SentenceComplexity(s) < i + 1\} \qquad i = 0, 1, 2, ..., 5 \tag{5}$$

The result is a set of 6 subcorpora that include sentences of increasing complexity. Table 4 reports the size of the different subsets in terms of total number of grammatical dependencies in the corresponding sentences. As expected very complex phenomena tend to be very rare. As values of $SentenceComplexity(s)$ up to 8 have been observed but for very few sentences, a seventh corpus *other* has been created by collapsing all the sentences in C_6, C_7 and C_8, as shown in the Table 4.

Over each of the created C_i corpus has been evaluated the performance of the systems.

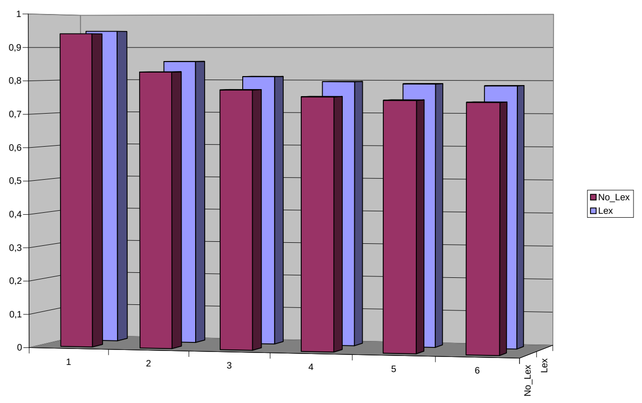

Fig. 3. Parser performance vs. sentence complexity in the Penn Treebank

As the Fig. 3 suggests decreasing values of F-measure according to increasing corpus complexity. Now robustness can be assessed by the simple observation that:

- it is proportional to a given parser performance index. F-measure is here used, although different criteria may be employed as well.
- it is inversely proportional to the factor expressed in Eq. 2, i.e.

$$\frac{\Delta S(T',C)}{\Delta C} < \frac{\Delta S(T,C)}{\Delta C}$$

Now, assume that $\Delta S(T,C)$ is given by ΔF, ΔC can be express as the percentage of new sentences in C_i with respect to C_{i-1}, i.e.

(6) $$\Delta C = \frac{|C_i| - |C_{i-1}|}{|C_i|} \qquad i = 1, ..., 6$$

Finally, the following robustness index *Rob* has been measured:

(7) $$Rob(i) = F(i) * \frac{\Delta C_i}{\Delta F_i} \qquad i = 1, ...$$

estimated by

(8) $$Rob(i) = F(i) * \frac{1}{|F(i) - F(i-1)|} * \frac{|C_i| - |C_{i-1}|}{|C_i|}$$

Figure 4 reports values of the Eq. 8 for the two parsers.

As expected the proposed score captures the tolerance of the *Lex* architecture to complex phenomena, i.e. those stressful conditions to which robustness should refer.

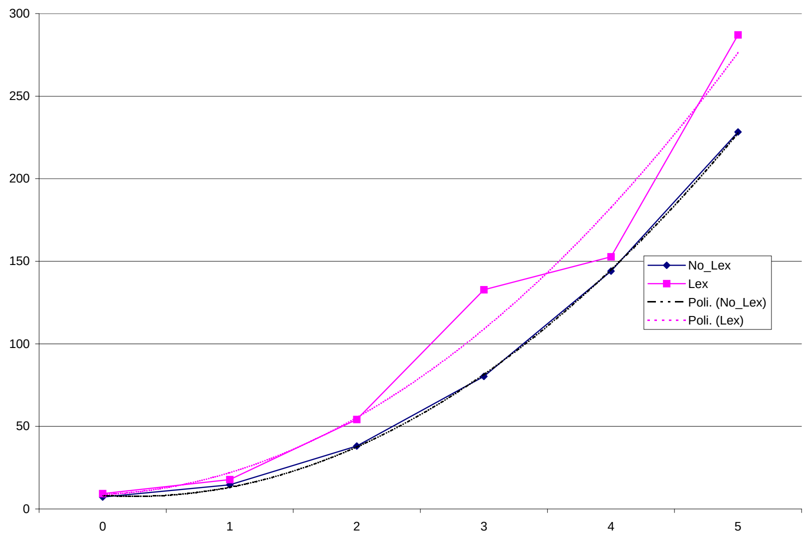

Fig. 4. Robustness of the two parsers

Polynomial interpolation is shown to suggest the superiority of the *Lex* parser, and also its trend with respect to asymptotic behaviour of complexity.

5 Discussion

The early proposal of this paper was that robustness could be usefully studied as an empirical phenomenon. The difficulties of other, more theoretical approaches, and the need for a systematic measurement of parser robustness suggested the use of a quantitative model of robustness close to the notion of graceful degradation of performance (used elsewhere, e.g. (Menzel1995)).

This view on robustness has been analysed against substantial experimental evidence over different corpora in two languages, Italian and English.

In all cases, a performance (and thus data) driven notion of robustness emerge from the tests, where increasing levels of complexity in the input are significant indicators for assessing robustness. The significance of such properties of the adopted corpora is enforced by the fact different architectures (i.e. independent parsers) result similarly weak on specific data sets. This suggests that performance indexes can be used to organize systematic experiments aiming to observe a resulting empirical notion of robustness.

Aspects like different syntactic phenomena or inherent sentence complexity have been used in Italian and English respectively to monitor degradation of performance. In Italian, corpora of different complexity were available and they have been used to evaluate the tolerance of the target parsers to increasing stressful input.

In English the availability of a (single) large-scale resource allowed a different experimental set-up. First, sentences have been separated according to an "inherent" notion of complexity (Eq. 4). Then the different sub-corpora obtained have been used to simulate increasing levels of stress, such that quantification of robustness was allowed.

All the tests suggested that architectural issues play a critical role on robustness (at least for parsing). Architectural choices are decomposition of the parsing task, uniqueness of the representation (against possibly conflicting approaches, e.g. probabilistic vs. logic models) as well as algorithmic principles. Among the latter, the exploitation of lexical knowledge suited for the target domain results an influential factor.

The adoption of domain specific lexicons is made available by the exploitation of machine learning techniques (see details in (Basili et al.1997)). Although viability of lexicalised parsing architectures was not the focus of this paper, experiments show that overall parsing performances are satisfactory (about 80% of F-measure on the Penn Treebank). As a further result all the tests suggest the strong beneficial impact of lexical information on robustness. This emphasizes researches like in (Carroll and Briscoe1998) dealing with the contribution of lexical syntactic knowledge to parsing.

As an overall result, the paper also emphasizes the role of parser design in the achievement of robustness. Whatever the formalism is, a design framework able to support modular parsing via uniform representations (i.e. constrained but underspecified data structures) and composition mechanisms plays a major role. This approach is not in contrast with previous work in this area (e.g. (Menzel1995; Worm and Rupp1998)). Every redundancy (or voting) approach to robustness is based in fact on modular architectures. Although two strictly pipeline architectures (i.e. deterministic cascades of different processors) have been experimented in this paper, the benefits of modularity on robustness have been proofed on a large scale. This validates the overall consistency of the proposed definition of robustness.

6 Conclusions

Robustness in parsing is a critical problem for linguistic modelling as well as for applications of natural language processing. The difficulties in a theoretical explanation of robustness suggested the adoption of a more empirical notion. The positive side effects of the proposed definition are:

- a systematic reference to performance, that is a desirable feature in view of specific applications where different kinds of robustness may be required
- a quantitative model able to capture some essential aspects of robustness with respect to linguistic input (i.e. invalid or complex sentences) and, mainly, exogenous stress, like lacks in available knowledge (e.g. poor lexical information)

- a usable notion of robustness aiming to inspire even the earlier design phases of the parser design

The extensive experiments carried out on two languages provided significant evidence on the empirical robustness of two independent parsing architectures. Although, especially for English, large resources are going to be more and more available, the needs for specific data sets for a larger study of robustness are still high. Future research will evaluate the use of other corpora for a larger contrastive analysis. Corpora with significant (and quantitatively measurable) differences will be retained for collecting further evidence. More experimentation is also needed for evaluating robustness of more complex framework, like redundant architectures based on voting schemes. The analysis of such pools of parsers will bring more insight on the architectural issues like optimal parser configurations and suitable lexical information.

References

Steven Abney. 1996. Part-of-speech tagging and partial parsing. In G.Bloothooft K.Church, S.Young, editor, *Corpus-based methods in language and speech.* Kluwer academic publishers, Dordrecht.

Salah Aït-Mokhtar and Jean-Pierre Chanod. 1997. Incremental finite-state parsing. In *Proceedings of ANLP97*, Washington.

Roberto Basili, Maria Teresa Pazienza, and Paola Velardi. 1992. A shallow syntactic analyser to extract word association from corpora. *Literary and linguistic computing*, 7:114–124.

Roberto Basili, Maria Teresa Pazienza, and Michele Vindigni. 1997. Corpus-driven unsupervised learning of verb subcategorization frames. Number 1321 in LNAI, Heidelberg, Germany. Springer-Verlag.

Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 1998a. Efficient parsing for information extraction. In *Proc. of the ECAI98*, Brighton, UK.

Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 1998b. Evaluating a robust parser for italian language. In *Proc. of the THE EVALUATION OF PARSING SYSTEMS Workshop, held jointly with 1st LREC*, Granada, Spain.

Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 1999. Lexicalizing a shallow parser. In *Proc. of the Traitement Automatique de la Langue Naturelle, TALN99*, Cargese, FR.

Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 2000. Customizable modular lexicalized parsing. In *Proc. of the 6th International Workshop on Parsing Technology, IWPT2000*, Trento, Italy.

Roberto Basili, Roberta Catizone, Luis Padro, Maria Teresa Pazienza, German Rigau, Andrea Setzer, Nick Webb, Yorick Wilks, and Fabio Massimo Zanzotto. 2001. Multilingual authoring: the namic approach. In *Proc. of the Human Language Technology and Knowledge Management held jointly with ACL2001*, Toulose, France.

R. Basili, M. Di Nanni, L. Mazzucchelli, M.V. Marabello, and M.T. Pazienza. August 1998. Nlp for text classification: the trevi experience. In *Proceedings of the Second International Conference on Natural Language Processing and Industrial Applications, Universite' de Moncton, New Brunswick (Canada).*

Michael R. Brent. 1993. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19-2.

E. Brill and P. Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of COLING-94*, Kyoto, Japan.

John Carroll and Ted Briscoe. 1998. A survey of parser evaluation methods. In *Proc. of FIRST INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, Granada, Spain.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34 th Annual Meeting of the Association for Computational Linguistics.*

R. Delmonte. 1992. *Linguistic and Referential Processes in Text Analaysis by computers.* UNIPRESS, Venezia.

R. Gaizauskas, M. Hepple, and C. Huyck. 1998. A scheme for comparative evaluation of diverse parsing systems. In *Proc. of FIRST INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION*, Granada, Spain.

D. Grinberg, J. Lafferty, and D. Sleator. 1996. A robust parsing algorithm for link grammar. In *4th International workshop on parsing tecnologies*, Prague.

Hobbs, Appelt, Bear, Israel, Kameyama, Stickel, and Tyson. 1996. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In Roche and Schabes, editors, *Finite State Devices for Natural Language Processing.* MIT Press, Cambridge MA.

IEEE. 1990. *IEEE Standard Glossary of Software Engineering Terminology.* IEEE Computer Society, IEEE Std 610.12-1990.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

W. Menzel. 1995. Robust processing of natural language. *Lecture Notes in Computer Science*, 981:19–34.

MUC. 1995. Proceedings of the sixth message understanding conference(MUC-6). In *Columbia, MD.* Morgan Kaufmann.

Maria Teresa Pazienza. 1997. *Information Extraction. A Multidisciplinary Approach to an Emerging Information Technology.* Number 1299 in LNAI. Springer-Verlag, Heidelberg, Germany.

C. Pollard and I.A. Sag. 1994. *Head-driven Phrase Structured Grammar.* Chicago CSLI, Stanford.

A. Ratnaparkhi and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *In Proceedings of the ARPA Workshop on Human Language Technology, Morgan Kaufmann.*

Geoffrey Sampson. 1993. The susanne corpus. *ICAME Journal*, 17.125-7.

L. Tesniere. 1959. *Elements de syntaxe structural.* Klincksiek, Paris, France.

Karsten L. Worm and C. J. Rupp. 1998. Towards robust understanding of speech by combination of partial analysis. In *Proc. of the ECAI98*, Brighton, UK.