Flexible Parsing Architectures for NLP Applications

Roberto Basili, Maria Teresa Pazienza, Fabio Massimo Zanzotto

Dipartimento di Informatica, Sistemi e Produzione,
Universita' di Roma Tor Vergata (ITALY)
{basili,pazienza,zanzotto}@info.uniroma2.it

Abstract. The requirements of different NLP applications have strong implications on the design and implementation of the related syntactic recognisers. In this paper, a fine-grained modular parser design framework is presented. Our aim is to reduce the design of a parsing processors to the composition of a pool of basic modules. Results over sample parsers and criteria for optimising coverage and accuracy are discussed.

1 Introduction

NLP applications (as Information Extraction, Text Classification, Document Indexing, etc.) employ grammatical analysers in activities ranging from the recognition of simple structures (Proper Nouns, noun phrases, etc.) to the detection of more complex phenomena (events). This heavily required flexibility has deep implications on the design and implementation of the target recognisers: different applications often have very different focus and performance needs.

The performance of the overall parsing process is a trade-off between linguistic quality of the produced material, on the one side, and the computational complexity, on the other. *Lexicalisation* of the grammatical rules and the *decomposition* of the parsing process are widely used approaches to control the quality of the results. However, sublanguage effects affecting subcategorisation lexicons have to be considered to make lexicalisation an effective means for controlling the performances. On the other hand, the *decomposition* of the parsing process (as often proposed in literature [1]) has suitable effects on the control of the overall parsing complexity, beeing too *a clear and reliable principle for the engineering of NLP applications.*

The design of a parsing processor can benefit from a fine-grained modular understanding of the syntactic recognition that models the overall (monolithic) process as a composition of dedicated components (e.g. POS tagging, noun phrase extraction, PP-attachment resolution). The design may simply proceed through elicitation of some of the available parsing components if they do not add value to the overall process in term of performance gain or recognition of interesting phenomena. Furthermore, in a modular framework, benchmarking of the different processors can be undertaken independently, and specific measures for the different potential architectures are enabled. Independent measurements

can thus justify the selection of an optimal architecture among a pool of potential configurations. Performance evaluation as well as optimisation can enter the design process since its earlier stages, as traditionally suggested in software engineering practice.

2 A Flexible Framework for Parser Design

A design model that supports a fine-grained modular parser development requires clear indications on the representation scheme of syntactic information exchanged between the modules and on a the compositions of the parser modules. Furthermore, a criterion inspiring the decomposition of the monolithic parsing process is desirable. As a consequence of this criterion, a pool of processors can be developed as the basic modules supporting for the composition of different parser instances.

2.1 Extended Dependency Graph

The uniform formalism should be able to represent partial analysis and show to the modules only the relevant information. The constituency-based approach has a limitation: the notion of *continuous* constituent. For instance, a pp-attachment resolution module should be free to draw the conclusion that a PP-kernel is attached to the VP-kernel without postulating the structure of the rests of NPs/PPs between the two. A dependency-based annotation scheme is more indicated to cope with this kind of problems, but it is not conceived for the information hiding: the nodes of the graph are always words, no encapsulation of the information is foreseen.

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): a 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 formalism efficiently models the syntactic ambiguity. In general, alternative interpretations for dependencies are represented by alternative $d \in D$. However, *planarity* [6] can be used to select unambiguous sentence (eventually partial) interpretations.

2.2 Composition of Parsing Modules

A generic syntactic processor MP is a *linguistic processing module* that (may) exploit syntactic knowledge stored in (possibly lexicalised) rule set R. MP processes a sentence S in order to augment it with syntactic information producing

a richer representation S'. The same applies to the parsing sub-processors P_i, i.e.

$$P : R \times \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma'\Delta'} \quad (1)$$

so that $P(r, xdg) = xdg'$, where xdg and xdg' are the input and the enhanced graph, respectively. The first argument of a function P_i can be omitted for sake of synthesis (since it is defined for a given parser instance). The equation 1 will be written $P_i(xdg) = P(xdg; r_i) = xdg'$. The overall modular parser MP is a cascade of processing modules $(P_1, ..., P_n)$ obtained via composition operator

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

Parsing sub-processors can be classified according to their attitude to preserve pre-existent choices (*monotonicity*), the actions performed on the graph (*constituent gathering*, for processors grouping set of words into larger constituents, and *dependency gathering*, where nodes are left untouched and only dependencies are added), and, finally, with respect the knowledge R_i used (*lexicalised* or *grammar-driven*). This classification gives indications both on the development and tuning costs and on the possible use in the chain of each single module.

2.3 Lexicalised Syntactic Processors

The decomposition adopted is inspired by the principle that subcategorisation lexicons are valuable resources for controlling the performances. Furthermore, the possibility of reducing the costs for building lexicons via automatic acquisition [4] makes *lexicalised* modules more attractive. In this framework, two lexicalised modules are adopted: a verb argument matcher and a adjective modifier matcher. These require a module, *the chunker* P_1 [1], for producing an intermediate sentence representation able to hide unrelevant ambiguities. The intermediate representation is $\mathcal{XDG}_{\Gamma\Delta}$ whose nodes are chunks (Γ and Δ are Γ={VPK, NPK, PPK, ...} and Δ={SUBJ, DIROBJ, PPMOD, }). More formally, $P_1 : \mathcal{XDG}_{\Gamma'\Delta'} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$ where Γ'={Noun, Prep, Verb, ...} and $\Delta' = \emptyset$.

The specific processor for matching verb argument structures, P_2, is a *dependency lexicalised* processor able to work at different levels of lexicalisation. It processes $\mathcal{XDG}_{\Gamma\Delta}$, i.e. $P_2 : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$. Successful matches add to the target xdg dependency edges also called *icd*s, i.e. *inter-chunk dependencies*. An original feature is the specific combination of the argument matching with the clause recognition [3]. The role of lexical information is not only to fill slots in valency lexical entries, but also to control, via planarity constraints, the matching for other verbs.

The second dependency-gathering lexicalised processor, P_3, has been designed to deal with adjectival subcategorisation. The recognition of dependencies whose heads are adjectives may well employ adjective subcategorisation frames. Let us take, as an example, an excerpt of the sentence # 0439(63) of the Penn Tree bank:

An increasing number of references by the Soviet press to opposition groups now active

in the U.S.S.R., particularly the Democratic Union, allege that ...
The processor will augment the grammatical information detected in the sentence by adding the dependency between *active* and *in the U.S.S.R.* according to the subcategorisation frame *[active [PP [P in] [NP _]]* .

3 Evaluating Alternative Parsing Architectures

The flexible architecture gives the possibility to investigate different parsing processors. In order to understand the value of the lexical information, the following configurations have been analysed:

- a base (non lexicalized) processor, i.e. $P_1 \circ P_4$, hereafter called $BASE$
- a strictly lexicalized processor, made of the composition of the chunker P_1 with P_2 and P_3 components referred to as $LEXICAL$
- a processor integrating lexicalized and shallow parsing, i.e. $P_1 \circ P_2 \circ P_3 \circ P_4$, hereafter called $COMBINED$

where P_4 ($P_4 : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$) is a non-lexicalized *shallow analyzer* [2], mainly used for lexical acquisition from corpora. Thus, it follows a $recall - first$ policy and retains ambiguous dependencies. Conflicting syntactic readings are preserved in the output graph. This affects the precision of the $BASE$ and $COMBINED$ configuration. As our goal is also to study precision, a PP-disambiguation module P_5 ($P_5 : \mathcal{XDG}_{\Gamma\Delta} \rightarrow \mathcal{XDG}_{\Gamma\Delta}$) has been also integrated in the parsing chains. P_5 implements a simple disambiguation strategy, i.e. the minimal attachment choice. P_5 is thus a non monotonic dependency-gatherer as it removes some of the previously assigned links in the input *xdg*. We have tested also two augmented configurations MA^- ($P_1 \circ P_4 \circ P_5$) and MA^+ ($P_1 \circ P_2 \circ P_3 \circ P_4 \circ P_5$) obtained by integrating the disambiguation processor.

For the performance evaluation in terms of quality of derived linguistic information, a metrics oriented to an *annotated-corpus* has been adopted. In particular the used formalism is *Parseval*-like where the *Parseval scheme* [5] has been adapted to the particular annotation paradigm. The comparison between the *oracle* information, i.e. the treebank-information, and the parser syntactic material is carried on a dependency-annotation base. The annotated corpus is the *Penn Treebank* [8], and the metrics adopted are then *R*ecall, *P*recision and *F*-measure. Constituency-based annotated information of the oracle has been translated in a dependency formalism. Translation algorithms have been settled in other works [7, 4]. In the present work the adopted translation algorithm left not translated about 10% of the *oracle trees* (i.e. reference corpus trees).

Metrics are settled on this representation and they are targeted over given grammatical relation τ (e.g. NP_PP), as follows:

$$R^\tau(S) = \frac{card((A_o^\tau(S) \cap A_s^\tau(S)))}{card(A_o^\tau(S))} \qquad P^\tau(S) = \frac{card((A_o^\tau(S) \cap A_s^\tau(S)))}{card(A_s^\tau(S))} \tag{2}$$

where $A_o^\tau(S)$ are the correct syntactic relations of type τ for the sentence S, and $A_s^\tau(S)$ are the syntactic relations of type τ extracted by the system. In particular,

given the set of test sentences Ω, global values for recall and precision are derived as follows:

$$R^{\tau}(\Omega) = \frac{1}{\sum_{s \in \Omega} \gamma_s} \sum_{s \in \Omega} \gamma_s R^{\tau}(s) \qquad P^{\tau}(\Omega) = \frac{1}{\sum_{s \in \Omega} \omega_s} \sum_{s \in \Omega} \omega_s P^{\tau}(s) \tag{3}$$

where γ_s and ω_s are weighting factors associated depending on the characteristics of sentences. Notice that another global definition, *cumulative recall* and *precision* respectively, is possible where cumulative phenomena are taken into account:

$$R^{\tau}(\Omega) = \frac{card(\bigcup_{S \in \Omega}(A_o^{\tau}(S) \cap A_s^{\tau}(S)))}{card(\bigcup_{S \in \Omega} A_o^{\tau}(S))} \qquad P^{\tau}(\Omega) = \frac{card(\bigcup_{S \in \Omega}(A_o^{\tau}(S) \cap A_s^{\tau}(S)))}{card(\bigcup_{S \in \Omega} A_s^{\tau}(S))} \tag{4}$$

The F-measure synthetic metric is defined for the whole corpus Ω as follows:

$$F_{\alpha}^{\tau}(\Omega) = \frac{1}{(\alpha \frac{1}{P^{\tau}(\Omega)} + (1-\alpha)\frac{1}{R^{\tau}(\Omega)})} \tag{5}$$

The target test bed Ω is a subset of 500 sentences out from the 40,000 translated interpretations. It has been set up in order to selectively measure the effects of the lexicalized processors in the parsing chains.

Our evaluation goal here is to compare the accuracy of different configurations. By using equation 3, we averaged precision and recall (i.e. γ_S and ω_S are set to 1, $\forall S$), and we measured the F-measure of the different architectures. Values of 72.23% and 78.19% characterizes the $BASE$ and $COMBINED$ configuration respectively. If disambiguation is adopted (i.e. in MA^{+}), the result is $F_{0.5}(\Omega)$ =81.48%. It is clear here that combination of lexicalized processors, plus a simple disambiguation strategy, outperforms in general the $BASE$ parser.

In order to selectively analyse syntactic dependencies (τ) several runs have been carried out. The measures of R_{τ} and P_{τ}, as in equations 4, are reported in Table 1.

The $BASE$ processor is the less precise, as it preserves all the potential attachments. Its recall represents an upper-bound for the other processors. Notice that the $LEXICAL$ configuration has an high precision (on specific phenomena, V_PP and Adj_PP) but its recall suggests that it is insufficient to fully "cover" the wide phenomena related to prepositional dependencies.

The best performing architecture is the $COMBINED$ cascade, where lexical knowledge is firstly employed and it constraints the scope of the shallow analyzer. This is true not only on prepositional modifiers but also on the attachment of direct object ($\tau = V\_NP$).

Several relevant consequences can be drawn. First, different types of phenomena with respect to the contributions of lexicalized processors can now be selectively studied. Secondly, optimal configuration for specific phenomena can be outlined, even in the design phase, and this supports and eases the systematic configuration of novel parsing architectures within new application scenarios. Finally, factorising lexical processes has proofed beneficial in the overall parsing process. This allows to (1) estimate the role of some lexicalised components and, if strictly required by the new application, (2) assess them via specific customisation actions during porting. The results suggest that modular parsing is viable

Table 1. Evaluation: Parsing configuration vs. different syntactic phenomena

	ADJ_PP		NP_PP		V_NP		V_PP	
	R	P	R	P	R	P	R	P
BASE	11,63%	100,00%	87,23%	26,81%	91,07%	44,94%	70,39%	37,50%
LEXICAL	25,58%	100,00%	-	-	86,51%	80,06%	38,55%	76,38%
COMBINED	27,91%	100,00%	80,84%	37,38%	87,74%	73,25%	68,16%	44,15%

within the defined framework and that even shallow approaches (widely employed within robust parsing systems) improve significantly their performances: they help to increase coverage, even when poor lexical knowledge is available.

References

1. Steven Abney. 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, 1996.
2. Roberto Basili, Maria Teresa Pazienza, and Paola Velardi. A shallow syntactic analyser to extract word association from corpora. *Literary and linguistic computing*, 7(2):114–124, 1992.
3. Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. Efficient parsing for information extraction. In *Proc. of the ECAI98*, Brighton, UK, 1998.
4. Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. Evaluating a robust parser for italian language. In *Proc. of the Workshop on Evaluation of Parsing Systems, held jointly with 1st LREC*, Granada, Spain, 1998.
5. E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proc. of the Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA, 1991.
6. D. Grinberg, J. Lafferty, and D. Sleator. A robust parsing algorithm for link grammar. In *4th International workshop on parsing tecnologies*, Prague, 1996.
7. D. Lin. A dependency-based method for evaluating broad-coverage parsers. In *Proc. of the 14th IJCAI*, pages 1420–1425, Montreal, Canada, 1995.
8. M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 1993.