# An evaluation of different symbolic shallow parsing techniques.

## Tristan VANRULLEN , Philippe BLACHE

Laboratoire Parole et Langage, UMR 6057 CNRS
Université de Provence, 29 Av. Robert Schuman, 13621 Aix-en-Provence, France
{tristan.vanrullen, blache}@lpl.univ-aix.fr

### Abstract

This paper presents an evaluation of four shallow parsers The interest of each of these parsers led us to imagine a parameterized multiplexer for syntactic information based on the principle of merging the common boundaries of the outputs given by each of these programs. The question of evaluating the parsers as well as the multiplexer came in the foreground with the problem of not owning reference corpora. We attempt here to demonstrate the interest of observing the 'common boundaries' produced by different parsers as good indices for the evaluation of these algorithms. Such an evaluation is proposed and tested with a set of two experiences.

## 1. Introduction

### 1.1. Why using different parsers

Shallow parsing usually relies on statistical techniques. In the case of symbolic shallow parsers, the method consists in using a reduced set of pre-compiled syntactic information. This information is generally at a very low level and specified in terms of filtering (e.g. constraint grammars). In such techniques, the linguistic information is heavily dependent from the parsing process. One consequence is that such systems are not modular nor reusable. There is another important question to be answered: what is the goal of shallow parsing? The classical answer is: an efficient and robust bracketing technique. Robustness is the most important aspect, shallow parsers must address this point, as well as efficiency: large and unrestricted corpora have to be treated. But the answer is not so obvious as for the last point: bracketing. We think that this constitutes only one aspect of the kind of information that can be built by shallow parsers: other kind of information such as dependency can also, under certain conditions, be built. Even more generally, we could imagine an integrated shallow parser generating syntactic (bracketing), semantic (dependency) and prosodic (intonative contours) information.

Such a goal absolutely requires for the parser to rely on high-level linguistic resources. The question is then: is it possible to develop an efficient and robust parsing strategy capable of integrating (if necessary) these different aspects? We propose in this perspective a strategy relying on a constraint-based representation. In such approach, all linguistic information is represented by means of constraints. All constraints being at the same level, it is then possible to verify only a subset of constraints. The idea consists in choosing the granularity of the parser in modifying such subset of constraints to be verified: there is a proportionality relation between the dimension of the set of constraints and the level of the parse. We can choose a very superficial granularity in verifying only one kind of constraints (for example the ones describing linearity) or to refine a little bit the parse in introducing other constraints. The main interest is that (1) the linguistic resource is the same in all cases (a set of constraints) and (2) the same system can be used for different granularity (i.e. different applications).

Such a goal doesn't mean that efficient and robust parsing don't require any more specific techniques. But we can make some proposals in this direction, for example implementing a deterministic strategy (ambiguity being in the end the main problem for parsing).

### 1.2. Improving parsers improves prosodic information for text-to-speech applications

Several domains in language technology can be improved by means of syntactic information. This is in particular the case for text-to-speech systems in which intonation generation can be driven with boundaries indication coming from shallow parsers (cf. [Allen], [Abney91], [Liberman92], or [DiCristo98]). However, if such systems have a larger scope than deep analysis techniques (they are in particular able to treat unrestricted texts in opposition to sublanguages), they also only provide poor linguistic information. The techniques generally used allow a simple chunking useful for some levels of speech synthesis, but too poor to give an actual account of more complex prosodic phenomena.

### 1.3. Several algorithms with a same goal

Some recent works (cf. [Hirshberg01]) showed that a finer analysis can significantly improve the prosodic quality. We propose in this paper a technique relying on the use of several symbolic shallow parsers (or more precisely deterministic parsers). Its particularity lies in the fact that it makes use of a linguistic formalism in spite of traditional stochastic information. Our goal is to improve quantity and quality of information likely to support intonation generation by means of surface analyzers. In this perspective, while preserving robustness and efficiency of the processing, we based our work on a linguistic formalism, called *Property Grammars* (cf. [Blache01b]) which main interest comes from the fact that any kind of input, even ill-formed, can be characterized with syntactic properties.

Three shallow parsers based on this formalism are presented and compared in this work. A fourth one, relying on a simple chunking approach is used in terms of reference.

### 1.4. Evaluation as a necessary crossroads

This paper addresses in particular the question of the interest of cascading several parsers in order to improve the result. Moreover, the evaluation problem itself is part of the work: due to the lack of a bracketed reference corpus for French, we present a 'subjective' evaluation (though automated) of these tools. Two experiences are described in order to test the behavior of these parsers.

### 1.5. An overview of Property Grammars

We propose to use a constraint-based formalism allowing to represent all kind of syntactic information by means of constraints. This formalism, called Property Grammars (cf. [Blache01b]) makes use of different types of constraints. The idea exposed above consists then in varying the granularity level in choosing the type of constraints to be verified. Let's present rapidly this formalism.

The representation of syntactic knowledge requires various types of constraints or properties, each one corresponding to a specific kind of information. There is a main difference from the usual presentation of Property Grammars in which the constituency information is not directly represented. In the following, and for efficiency reasons, we add this new type of property even if redundant. The following list presents these properties:

- *Constituency* (noted `Const`): Specifies the maximal set of categories that can appear in a category.
  Example: *Const(NP)={Det, AP, N, PP, Sup, Pro}*
- *Obligation* (noted `Oblig`): Specifies the possible heads. One of these categories (and only one) has to be realized.
  Example: *Head(NP) = {N, Pro}*
- *Uniqueness* (noted `Uniq`): Set of categories that cannot be repeated in a phrase.
  Example: *Uniq(NP) = {Det, N, AP, PP, Sup, Pro}*
- *Requirement* (noted ⇒): Cooccurrency between sets of categories.
  Example: *N[com]  Det*
- *Exclusion* (noted ): Cooccurrency restriction between sets of categories.
  Example:  *AP  Sup* (in a NP, a superlative cannot cooccur with an AP)
- *Linearity* (noted <): Linear precedence constraints.
- *Dependency* (noted →): Dependency relations between categories.

One of the originality of this approach is that a linguistic description is not presented in terms of grammaticality: parsing an input comes to verify the set of constraints. It is then possible to characterize each component of this input with the set of constraints that are satisfied (plus, eventually, the set of constraints that are violated). The core mechanism being a constraint satisfaction one, it is possible to verify only a subpart of the entire constraint system (in other words, the grammar).

## 2. Shallow, deep and granular parsers

### 2.1. A low-level shallow parser

The first technique described in the paper is inspired by Liberman & Church's Chink/chunk (1991) and by Di Cristo's Chink/chunk chunker (1998). Let's call **A1** this algorithm: the result is a segmentation of the text into *chunks*, according to a finite-state-automaton based on the concept of *function words* which plays the role of boundaries between blocks. An improvement of the concept of chunk is proposed, using conjunctions as neutralizing chunks under construction. For M sentences, each sentence consisting of Nm words, its complexity has an order of M*Nm*k (K < 10). That is to say a linear complexity.

The figure 1 below is the output of this parser for a sentence taken from the French newspaper 'Le Monde'.

```
[(bloc)La célébration]
[(bloc)de le dixième anniversaire]
[(bloc)de la mort]
[(bloc)de Max]
[(bloc)Pol Fouchet va commencer]
[(bloc)par un colloque universitaire]
[(bloc)à l' université Sorbonne nouvelle]
[(bloc)centre Censier]
[(bloc)13][(bloc)rue]
[(bloc)de Santeuil]
[(bloc)Paris]
[(bloc)5 e]
[(bloc)salle]
[(bloc)de les périodiques]
```

figure 1: Output for A1

The three other techniques described in the remaining of the paper are based on a compiled subset of Property Grammars birefly exposed below (see [Blache01a] for implementation aspects). All three build grammatically annotated blocks by traversing deterministically a sentence. During the process, blocks are opened (sometimes recursively) in a stack.

The tagsets used by each of these algorithms are rather different (depending on the granularity of the parser), which implies many differences between their results. These algorithms use different heuristics too. For the first two, opening and closing chunks depends on the precompiled grammar; for the last, the entire set of properties of the 'Property Grammars' is checked for each word.

### 2.2. A compiled subset of properties

In the second algorithm **A2**, a grammar based on left and right potential corners, and potential constituents of chunks, is generated with a tool compiling constituency, linear precedence, requirement and exclusion properties. In the worst case, for M sentences, each sentence consisting of Nw words, for a set of C precompiled categories, its complexity is $M*C*(Nw^2+Nw)*Constant$. That is to say a polynomial complexity.

Figures 2, 3 and 4 give the outputs of algorithms A2, A3 and A4 for the same sentence as for fig.1:

```
[(phrase)
   [(SN)La célébration]
   [(SP)de
      [(SN)le
         [(SA)dixième]
         anniversaire]]
   [(SP)de
      [(SN)la mort]]
   [(SP)de Max Pol Fouchet]
   [(SV)va commencer]
   [(SP)par
      [(SN)un colloque
         [(SA)universitaire]]]
   [(SP)à
      [(SN)l' université Sorbonne nouvelle
       centre   Censier 13 rue]]
```

```
       [(SP)de Santeuil Paris]
       [(SN)5 e salle]
       [(SP)de
          [(SN)les périodiques]]]]
```

figure 2: output for A2

## 2.3.  The whole set of properties

In **A3**, the parsing strategy relies on left corners, but verifies all the properties for each chunk. Finally, the last parser **A4** proposes a deterministic approach relying on the entire set of constraints proposed in a Property Grammar. Their complexity is still polynomial as discussed in a paper not yet published.

```
[(P)
  [(SN)La celebration
    [(SP)de
      [(SN)le
        [(SA)dixième]]]]
    [(SN)anniversaire
      [(SP)de
        [(SN)la mort
          [(SP)de
            [(SN)Max Pol Fouchet]]]]]
    [(SV)va commencer
      [(SP)par
        [(SN)un colloque
          [(SA)universitaire]]]]
    [(SP)à
      [(SN)l université Sorbonne centre Censier
            rue
        [(SP)de
          [(SN)Santeuil Paris salle
            [(SP)de
              [(SN)les périodiques]]]]]]]]
```

figure 3 : output for A3

```
[(P)
  [(SN)La célébration
    [(SP)de
      [(SN)le
        [(SA)dixième]]]]
  [(SN)anniversaire
  [(SP)de
    [(SN)la mort
      [(SP)de
        [(SN)Max Pol Fouchet]]]]]
  [(SV)va commencer
    [(SP)par
      [(SN)un colloque
        [(SA)universitaire]
      [(SP)à
        [(SN)l université Sorbonne centre Censier
              rue
          [(SP)de
            [(SN)Santeuil Paris salle
              [(SP)de
                [(SN)les périodiques]]]]]]]]]]]
```

figure 4 : output for A4

## 3.  How to evaluate empirically parsers without reference corpora

### 3.1.  A brief overview of the problem

The question of evaluating parsers (even shallow) is a problem in itself. At the difference of POS-tagging, many aspects can vary from one system to another, including the output itself. Before presenting more precisely our systems, we would like to give some general remarks about evaluating parsers.

Generally speaking, evaluating a system consists in comparing for a given input its output with a standardized reference output. In the case of parsing, the reference is a treebank, the comparison comes in comparing the respective bracketings. This means first the availability of a treebank (such resource only exists for few languages). This also means that the parser as to build the same kind of information as in the reference corpus. This can also be problematic. First, bracketing is not totally theory-free. The second problem is that such resource usually only indicates one solution. Finally, as explained above, bracketing is not the only kind of information that we would like to evaluate.

Moreover, it seems to us interesting not to limit an evaluation to the comparison of different outputs. It is also necessary in order to interpret such a comparison, to give some indications on the resources and the techniques involved in the system. For example, it is important to have indication on:

- an indication on the lexical coverage
    - o  number of entries
    - o  representations (lexical features)
- an indication of the syntactic coverage
    - o  the number of categories
    - o  the different syntactic phenomena
- the parsing strategy
    - o  robustness
    - o  efficiency

Our contribution in this paper lies in the possibility of extracting some evaluation information from a comparison technique. In other words, we show that comparing different parsers, provided that the method is systematic, allow in some cases to give some elements of evaluation.
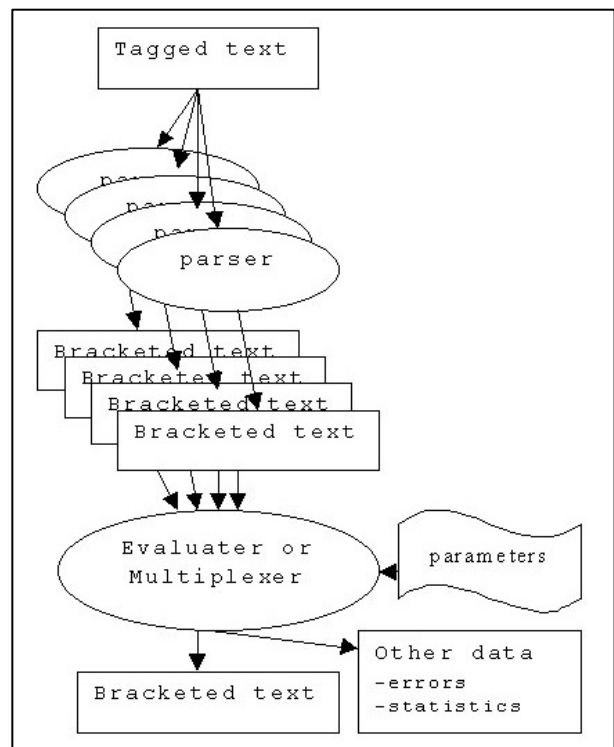
### 3.2.  Evaluating and /or multiplexing



figure 5 : To evaluate and/or multiplex parser's outputs

### 3.2.1. A multiplexer for bracketed texts

The idea of retrieving the same boundaries within texts bracketed with different parsers leds us to imagine a program able to merge in a parameterized way the outputs of these parsers. The goal is to keep the best information given by all of them and to let the worst be lost (see figure 5).

This program had to deal with sets of borders, that's why its parameters were of two kinds:

- set operators
  - union
  - intersection
  - complement
- weights
  - to balance the respective results of each parser for each syntactic category
  - to avoid the errors common to each parser

With such a program, we could exclude the worst and less significant borders and keep the best ones

### 3.2.2. An evaluator for the multiplexer as well as for each parser

But the parameters needed by this program could not be found without a good evaluation of the output of each parser.

These two needs are so closely related that we cannot distinguish them, except in an empirical step energy from the parameter setting to the evaluation and then in a retroactive way from the evaluation to the parameter setting.

Of course, even if all the preceding steps are automatic, the last one is an expert's work. while counting on the effects of the evaluator, we do not have any more but to check the relevance of its parameters.

In other words:

- the multiplexer program does the main part of the evaluation work by distinguishing the common borders and the less significant or the more particular: it informs us about the importance of each parser relatively to the others.
- A human feedback is still needed to improve each parser's outputs and the parameters of the multiplexer.

## 4. Experiments

The evaluation presented in this paper relies on two experiments: for each one, a tagged corpus of 13,236 French sentences (from the CLIF project, see http://www.talana.linguist.jussieu.fr) was used as input. Two kind of tagging of the lexical categories were used for these sentences: a manual tagging and an automatic one (realized with the french version of WinBrill).

The main objective of this experiment is to evaluate robustness and efficiency of algorithms for unspecified sentences automatically tagged.

To see better what can be found by such a program, we only used as parameters **the intersection set operator** and **the same weight** for each parser's output. Further studies should refine them.

### 4.1. First experiment

The first experiment set aims at comparing **block boundaries**, according to **the algorithm** and **the tagging**. To do this, we carry out a massive simplification of the blocks generated by programs A2, A3 and A4 in order to preserve only boundaries. Then we determine common borders, which constitutes a simple way of testing them without a reference corpus for these French sentences.
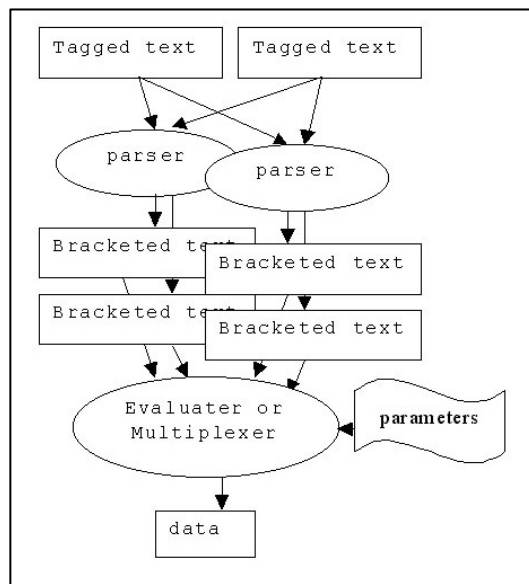


figure 6 : Is text tagging disturbing parsing?

### 4.1.1. 2 times 4 different outputs implies 64 evaluations

With a text tagged two times, we get eight outputs with the four parsers. The data to evaluate give 64 files were very large.

Because this experiment only aims at proving the good performance of our parsers with human tagged and automatically tagged texts, we only expose here the results of comparing A1 with A2.

Figure 6 shows the experiment procedure, figures 7 and 8 its results.

### 4.1.2. Results

| Algorithm | Words per chunk |
|---|---|
| A1 Human | 3.49 |
| A1 with WinBRILL | 3.34 |
| A2 Human | 2.02 |
| A2 with WinBRILL | 1.96 |

Figure 7: Results of the first experiment: words per chunk

The table below gives for each comparison between two bracketed outputs the number common borders (assuming that hierarchical brackets of A2, A3 and A4 are first transformed into linear ones by the program)

| First eval. | A1 Human | A1 with WinBRILL | A2 Human | A2 with WinBrill |
|---|---|---|---|---|
| A1 Human | 100% | 92% | 82% | 77% |
| A1 with WinBRILL | 89% | 100% | 75% | 81% |
| A2 Human | 50% | 47% | 100% | 82% |
| A2 with WinBRILL | 45% | 50% | 80% | 100% |

Figure 7: Results of the first experiment: common boundaries for two tagging methods

This table can be read this way:
- common boundaries of A1 with human Tagging and A2 with human tagging represent 82% of the amount of borders in A1's bracketed text
- common boundaries of A2 with human Tagging and A1 with human tagging represent 50% of the amount of borders in A2's bracketed text

Results are instructive, but two variables have to be isolated:
- the difference of behaviour for the same algorithm with two ways of tagging
- the difference between two algorithms, which outputs are very different.

It comes out from this experiment that boundary differences obtained by a **same parser** for the **two taggers** are from 2 to 3%, which indicates that automatic POS tagging remains relevant for the notion of border compared to an expert tagging.

This result is highlighted by another statistic given by the evaluator: the number of words per chunk (see figure 7).

A second conclusion is that the algorithms are sensitive to the tagging quality (i.e. they react to the variability).: these results indicate that A1 looses up to 10% of its borders when the tagging is not human, and A2 looses up to 20% of its borders.

A last conclusion is that the algorithms A1 and A2 really have from 47 to 82% common borders (according to what has already been said, these differences highlight the availability of using these common borders in order to harmonize and guarantee the efficiency of the diverse outputs). This point is discussed in the second experiment.

### 4.2. Second experiment

The second experiment set aims to compare the three approaches based on Property Grammars. Common boundaries are compared, category by category. This evaluation reveals several interests for each approach.

Figures 9 to 14 show the different data resulting of the avaluation.

| Algorithm | A2 | A3 | A4 |
|---|---|---|---|
| Chunks/ sentence | 15.03 | 19.04 | 18.97 |
| Words/chunk | 1.90 | 1.50 | 1.50 |

Figure 9: Statistics for the second experiment

| NP | A2 | A3 | A4 |
|---|---|---|---|
| A2 | 100% | 54% | 45% |
| A3 | | 100% | 100% |
| A4 | | | 100% |

Figure 10: NP common borders

| VP | A2 | A3 | A4 |
|---|---|---|---|
| A2 | 100% | 29% | 27% |
| A3 | | 100% | 75% |
| A4 | | | 100% |

Figure 11: VP common borders

| AP | A2 | A3 | A4 |
|---|---|---|---|
| A2 | 100% | 50% | 43% |
| A3 | | 100% | 86% |
| A4 | | | 100% |

Figure 12: AP common borders

| PP | A2 | A3 | A4 |
|---|---|---|---|
| A2 | 100% | 57% | 49% |
| A3 | | 100% | 85% |
| A4 | | | 100% |

Figure 13: PP common borders

| COORD | A2 | A3 | A4 |
|---|---|---|---|
| A2 | - | 0% | - |
| A3 | | 100% | 0% |
| A4 | | | - |

Figure 14: COORD common borders

Other results resulting of the evaluation are as significant as those shown in the tables 10 to 13.

The approaches A2 and A3 are rather different (48% average common categories). That partly comes from differences between the tagsets (A3 uses categories that A2 does not know). More precisely, NP, AP, the PP and

VP, have respectively up to 55%, 50%, 57% and 30% common borders.

A3 is closer to A4, which seeks to satisfy all constraints (90% average). NP, AP, the PP and the VP, have respectively up to 100%, 85%, 86% and 71% common borders.

These results imply two conclusions
- Common borders inform us about the originality or the conformism of a parser in comparison to another.
- A simple knowledge of what each parser does will allow us to parameterize the set operations and the weights associated to each one.

For an example, a guide to read these tables can reside in the fact that the algorithm A4 has given the best results in comparison with an expert evaluation of 10 sentences. It comes that most of the common boundaries A4 shares with A2 and A3 are carrying great weight and have to be merged with an 'intersection' set operator.

Another information resides in the fact that A3 knows categories that neither A2 nor A4 knows (see figure 14). This knowledge implies that COORD category has to be included in a multiplexing perspective with a weight of 100% and a 'union' set operator.

## 5. Conclusions

Several conclusions can be extracted from these experiments. In particular, it is possible to calculate efficiently in a deterministic way the syntactic categories constituting a sentence. Moreover it is possible to reduce errors by combining several parsers.

An interesting result for further studies lies in the fact that common boundaries obtained by two algorithms eliminates ill-formed and least remarkable boundaries. At the same time, it, increases the size of the blocks while keeping stored the linguistic information available.

Finally, the perspective of combining different approaches allows to propose a parameterized granularity in balancing the relative importance of different competing approaches.

Other experiments have to be done in order to know more things about multiplexing parsers outputs: cascaded multiplexing will reduce the quantity of chunks per sentence and cause a loss of data that has to be constrained and controlled.

## 6. References

Abney, S. (1991). Parsing by chunks. In *Berwick, R., Abney, S., Tenny, C. (Eds.). Principle-based parsing.* (pp. 257--278). Kluwer Academic Publishers, Dordrecht.

Abney, S. (1997). Part-of-speech tagging and partial parsing. In *Young, S., Bloothooft, G. Corpus-Based Methods in Language and Speech Processing*, (pp. 118--136). Kluwer Academic Publishers, Dordrecht.

Allen, J., Hunnincutt, S., Carlson, R., Granström, B. (1979). MITalk-79 : The 1979 MIT text-to-speech system. In *Wolf and Klatt (Eds.), Speech Communications* (pp. 507—510). Papers Presented at the 97th Meeting of the ASA.

Allen, J., Hunnincutt, S., Klatt, D. (1987). From text to speech : The MITalk system. Cambridge University Press.

Blache P. & J.-M Balfourier (2001a). "Property Grammars: a Flexible Constraint-Based Approach to Parsing", in proceedings of *IWPT-2001*.

Blache P. (2001b) *Les Grammaires de Propriétés : Des contraintes pour le traitement automatique des langues naturelles*, Hermès.

Di Cristo A., Di Cristo P, Campione E, Veronis J, (2000). A prosodic model for text to speech synthesis in French.

Di Cristo P, (1998). Génération automatique de la prosodie pour la synthèse à partir du texte. Thèse de Doctorat.

Hirschberg J., Rambow O (2001). Learning Prosodic Features using a Tree Representation. AT&T Labs Research. Eurospeech 2001 - Scandinavia.

Liberman, M., Church, K. (1992). Text analysis and word pronunciation in text-to-speech synthesis. In *Furui, S., Sondhi, M.M. (Eds), Advances in Speech Signal Processing*, New York: Dekker, 791-831.

Martin, L.E. (1990). Knowledge Extraction. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 252--262). Hillsdale, NJ: Lawrence Erlbaum Associates.