PCFG Extraction and Pre-typed Sentence Analysis

Noémie-Fleur Sandillon-Rezer
nfsr@labri.fr

LaBRI, CNRS
351 cours de la Libération, 33405 Talence

Abstract. We explain how we extracted a PCFG (probabilistic context-free grammar) from the Paris VII treebank. First we transform the syntactic trees of the corpus in derivation trees. The transformation is done with a generalized tree transducer, a variation from the usual top-down tree transducers, and gives as result some derivation trees for an AB grammar, which is a subset of a Lambek grammar, containing only the left and right elimination rules. We then have to extract a PCFG from the derivation tree. For this, we assume that the derivation trees are representative of the grammar. The extracted grammar is used, through a slightly modified CYK algorithm that takes in account the probabilities, for sentences analysis. It enables us to know if a sentence is include in the language described by the grammar.

1 Introduction

This article describes a method to extract a PCFG from the Paris VII treebank [1]. The first step is the transformation of the syntactic trees of the treebank into derivation trees representative of an AB grammar [13], which corresponds to the elimination rules of Lambek Calculus, as shown in Fig. 1. We chose an AB grammar because we want our approach to be potentially compatible with some usual learning algorithms, like the one of Buszkowski and Penn [4] or Kanazawa [12]. Once we have the derivation trees, we extracted the PCFG from them, and use it for sentence analysis. This analysis helps us to know how we can improve our grammar and all the processing line used to get it, by analyzing why some correct sentences cannot be parsed or why some incorrect ones are still parsed. In a more long-viewed aim, parsing french sentences can be use for grammatical checking, and with semantic information over a lexicon, the grammar could be used for generate coherent sentences.

$$\frac{A/B \quad B}{A}\ [/E] \qquad \frac{B \quad B\backslash A}{A}\ [\backslash E]$$

Fig. 1. Elimination rules of Lambek Calculus. An AB grammar is composed of instantiations of these two rules, where A and B are Lambek types.

The Paris VII treebank [1] contains sentences from the newspaper *Le Monde*, analyzed and annotated by the *Laboratoire de linguistique de Paris VII*. The flat shape of trees does not allow the direct application of a usual learning algorithm, so we decided to use a generalized tree transducer. For our work, we use a subpart of the treebank, on a parenthesized form, composed by $12,351$ sentences. Even if the whole treebank was in an **XML** form, the parenthesized form is easier to treat with the transducer. The 504 sentences left aside will be an evaluation treebank that we use as a control group.

Another new treebank, Sequoia [5], which is composed by $3,200$ sentences coming from different horizons, will also be used for experimentation. It is annotated using the same convention as the French Treebank.

This article will firstly overfly the transducer we use to transform syntactic trees into derivation trees, then we will focus on PCFG extraction. In a third part, we will detail the experimental results, obtained by using our PCFG to find the best analysis for a sentence, via the CYK algorithm [19].

2 Generalized Tree Transducer

It exists many way to make a syntactic analysis of a treebank, as we can see with the work of Hockenmaier [8], or Klein and Manning [10], but they were not applicable over the French Treebank or they did not gave simple AB grammar.

The transducer we created is the central point of the grammar extraction process. Indeed, the binarization of syntactic trees parametrize the extracted grammar. We based our works on usual derivation rules of an AB grammar used in computational linguistic and the annotations of the treebank itself [2]. The annotations give two types of information about the trees :

POS-tag: the **P**art-**Of**-**S**peech tags, booked for the pre-terminal nodes, indicates the POS-tag of the daughter node. For example *NC* will be used for Common Noun, *DET* for a determinant, etc.

Phrasal types: the nodes which are not a terminal or a pre-terminal node are annotated with their syntactic categories and sometime the role of the node. A *NP-SUJ* node will correspond to a Noun Phrase used as a subject for the sentence.

For the usual derivation rules, they are instantiation of elimination rules of Lambek calculus (see Fig. 1). We based ourselves on other annotation methods : a *NP* node will have the type np, a sentence type will be s, a preposition phrase taken as an argument will have the type pp, and so on.

A transducer, like defined by TATA [7], is an automaton which read an input and write something on output. It can be applied to trees and transform the shape of them. The transducers have some feature especially important for our work. Indeed, they are non erasing (it ensures us that we do not lost informations during the transduction), linear (the transduction will not change the order of the words in the sentence) and ϵ-free (gives more control over the transduction). The G transducer, developed by Sandillon-Rezer in [17], has additional features that feet better with the global shape of the treebank:

Recursivity: Our transducer can apply a rule to a node, looking only on its label but with an arbitrary arity. It generalize the usual definition of transduction rule, by matching node with an arbitrary number of daughters. The study of each specific case of the rule can transform the recursive rule into a set of ordinary rules.

Parametrization: We allow the rules to have some generic nodes which replace a node from a finite set of nodes. This quantification is equivalent to write each instantiation of the generic node.

Priority system: As our transducer needs to be deterministic, we decided to apply the rules always in a given order. It ensures us to have only one output tree.

The transduction rules have been written from a systematic analysis of the different shapes we could find in the treebank. As an example, a syntactic tree from the treebank and its transduction is shown Fig. 2.

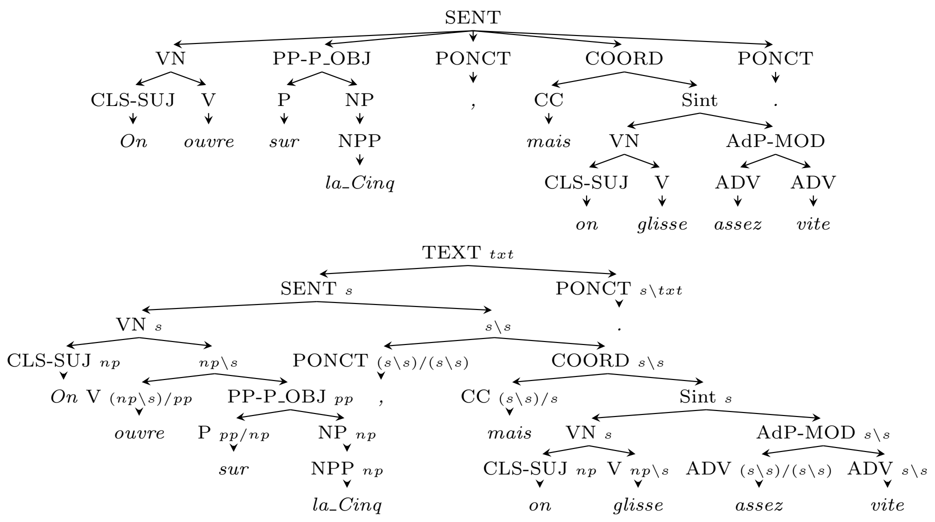

Fig. 2. Syntactic tree and derivation tree for the sentence: *"On ouvre sur la Cinq, mais on glisse assez vite."* (We open on *la Cinq*, but we slip fast enough.).

3 Grammar extraction

Even if the lexicon, extracted from the derivation trees, is representative enough of an AB grammar, and gives a probabilistic distribution of different types for words, it limits the sentence analysis to the sole lexicon of the treebank. This is the reason we decided to extract a PCFG from the derivation trees.

The output trees give both syntactic (we keep the initial labels) and structural information over sentences. We decided to proceed to a preprocessing step, in order for the user to control the extracted information. The only part of information we always keep is the type of nodes. The extracted grammar is a PCFG, the probabilities are computed based on their root node. We remind that our grammar is defined by a tuple

$\{N, T, S, R\}$, where N is the set of non-terminal symbols (internal nodes of trees), T the set of terminal symbols (typed leaves), S the initial symbol[1] and R the set of rules.

The extraction algorithm parses the trees and stores the derivation rules it sees. A rule is composed by a root and one or two daughters, then this is the usual case of a right or left elimination rule ($a \to a/b \quad b$ or $a \to b \quad a\backslash b$). Otherwise, it is only a type transmission which appears when a noun phrase is composed by a proper name only; or at the pre-terminal node level when the POS-tag node transmits type to the leaf. The probabilities are computed on a root related group.

The table 1 summarizes the grammars potentially generated. Each one presents useful information: the first one, from the derivation trees without preprocessing step, keeps the syntactic informations given by the treebank. The others are more useful for the application of a sentence analysis algorithm, like CYK (see section 4), on non-typed sentences. The table 2 shows some extracted rules.

Table 1. Extracted grammar. $n_i \in N$ and $t_i \in T$

Shape of trees	**Extracted rules**	**Specification**	**Number of rules**
Raw derivation trees	$n_1 \to n_2 \quad n_3$ $n_1 \to n_2$ $n_1 \to t_1$	Easy normalization in CNF: just need to remove some unary rules.	$63,368$
Removal of unary chains and labels exept POS-tags and words	$n_1 \to n_2 \quad n_3$ $n_1 \to t_1$	The grammar is in CNF.	$59,505$
Removal of unary chains and labels. No difference between N and T.	$n_1 \to n_2 \quad n_3$	The words do not even appear, we only have the skeleton of trees.	$3,494$

4 Sentence analysis

The analysis process can be subdivided in two parts. On the one hand, we have to type the words, while staying as close as possible to standard Lambek derivations. On the other hand, the needed rules must belong to the input grammar.

4.1 Word typing

By gathering the leaves of the derivation trees, we have a lexicon, as we can see Fig. 3. However, a typing system based only on this lexicon reduces the possibility of parsing to the sentences composed by words from the French Treebank. We decided to type words with the Supertagger (see Moot [15, 16]), which enabled us to validate the Supertagger results and to analyze sentences which did not occur in the Paris VII treebank.

The Supertagger is trained with the lexicon extracted from the transduced derivation trees.

[1] $S = \textit{TXT:txt}$ or *txt*, depending of the preprocessing step.

Table 2. Example of rules extracted from various input trees.

Raw derivation trees		
Rule example	$NP{:}np \rightarrow NPP{:}np$	$1.01e^{-1}$
	$NP{:}np \rightarrow DET{:}np/n\ NC{:}n$	$2.02e^{-1}$
Removal of unary chains and labels but POS-tags and words		
Rule example	$:(np\backslash s_i)/(np\backslash s_p) \rightarrow VINF{:}(np\backslash s_i)/(np\backslash s_p)$	$9.53e^{-1}$
	$:(np\backslash s)/(np\backslash s_p) \rightarrow CLR{:}cl_r\ \ :cl_r\backslash((np\backslash s)/(np\backslash s_p))$	$2.88e^{-2}$
Removal of unary chains and labels.		
Rule example	$np \rightarrow np/n\ n$	$8.02e^{-1}$
	$s \rightarrow np\ np\backslash s$	$3.81e^{-1}$
	$s \rightarrow s\ s\backslash s$	$2.65e^{-1}$
	the sentence has a "sentence modifier" at its end.	
	$s \rightarrow np\backslash s_p\ (np\backslash s_p)\backslash s$	$1.13e^{-3}$
	the type $np\backslash s_p$ corresponds to a past participle, used as an argument of the whole sentence.	

```
5968:le:det: - 5437:np/n, 140:(n\n)/n, 79:(s/s)/n, 68:(s\s)/n
```

Fig. 3. Extract of the lexicon. It gives the occurrence in the derivation trees, the POS-tag of the words and the associated types. Here, the determiner "le" occurs 5968 times in the derivation trees, and the most probable type is *np/n*, the usual one for a determiner at the beginning of a noun phrase. The three other types correspond to noun phrases used as modifiers.

4.2 Typed sentence analysis

We decided to use the CYK [19] algorithm, already tested and considered as a reference, with a probabilistic version for the parsing of sentences. We removed the typing step done initially by CYK with the rules $n_1 \rightarrow t_1$ 1, replaced by the Supertagger work. We use the simplest grammar (of 3,494 rules) for the analysis. The first test, to assure the correct running of the program, was to re-generate the trees from the transduced sentences with the grammar extracted from the derivation trees. Then, we tested the analysis with sentences typed by the Supertagger, using the grammar extracted from the main treebank (see results section 5.

The derivation trees corresponding to the sentence *"Celui-ci a importé à tout va pour les besoins de la réunification."* ("This one imported without restraint for the need of reunification need.") are shown in Fig. 4. We took the two most probable trees, typed by the Supertagger and analyzed with the grammar extracted from the main treebank. Two types of information are relevant to choose the best trees: we took into account the probability and the complexity of types. However, it is known that the comparison between two trees that do not have the same shape or leaves is complex. The main difference between the two trees is the prepositional phrase attachment; the most probable tree is more representative of the original treebank.

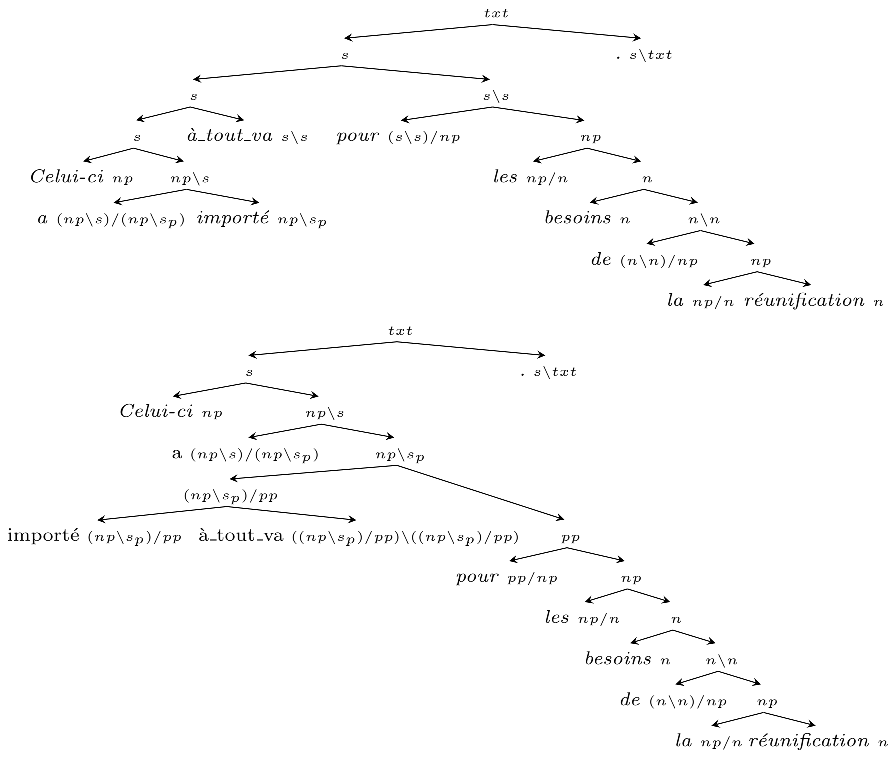

Fig. 4. Probability of the first tree: $5.12e^{-05}$; probability of the second one: $2.29e^{-08}$.

5 Results and evaluation

5.1 *G*-Transducer

From now, the transducer treats at least 88% of the corpora (see Table 3 for details). The lower percentage on the evaluation treebank can be explained by the study of the remaining sentences: they are colloquial and complex. On Sequoia, the better results can be explain by the greater simplicity of sentences.

Table 3. Coverage of the *G*-Transducer

Treebank	Number of sentences	Transduced sentences	Percentage
Main treebank	12,348	11,456	92.6%
Evaluation treebank	504	446	88.5%
Sequoia	3200	3014	94.2%

The use of rules, for the main treebank, is summarized in Table 4. We note that even if many rules are used infrequently, they do not have a real weight in the global use of the rules. Some of the important rules are shown in the Table 5 in the Tregex [14] parenthesized way. We were surprised too by the few occurrences of the rule that treats the determiner at the beginning of a noun phrase, despite the amount of *NP* in the treebank, but there is a rule which treats only the case of a noun phrase composed by a determinant and a noun, called 8,892 times.

The derivation trees of the three corpora allow us to extract three different grammars, and in addition, lexicon were created, containing words and their formulas. The lexicon covers 96.6% of the words of the Paris VII treebank, i.e. 26,765 words on 27,589, and for Sequoia, it covers 95.1%, i.e. 18,350 word on 19,284.

Table 4. Summary of the rule usage.

Number of rules	Minimal and maximal occurrence.	Number of applications
1,148	between 1 and 20	5,818
473	between 21 and 1,000	68,440
41	between 1,001 and 10,000	**125,405**
4	greater than 10,000	60,779

Table 5. Some important rules of the transducer, including the four most important.

input pattern	output pattern	applications
(*NP:* NC PP*)	(*NP:* NC:n PP:n\**)	17,767
(*NP:* DET tree*)	(*NP:* DET:np/n NP:n*)	16,232
(*PP:* P NP*)	(*PP:* P:*/np NP:np*)	16,037
(*SENT tree PONCT*)	(*TEXT:txt SENT:s PONCT:s\ txt*)	10,819
(*NP:* tree* (*COORD CC NP*))	(*NP:* (:* NP:** (*COORD:*\* CC:(*\*)/np NP:np*)))	2,511
(*SENT:* NP-SUJ VN NP-OBJ*)	(*SENT:* NP-SUJ:np* (*:np\* VN:(np\*)/np NP-OBJ:np*))	1,820

$$\frac{NP:* \qquad \dfrac{CC:(*\backslash *)/np \quad NP:np}{COORD:*\backslash *}\,[/E]}{NP:*}\,[\backslash E]$$

Fig. 5. Rule for the coordination between two *NP*, as shown in Table 5.

5.2 Sentence parsing

The parsing of typed sentences is done with the grammar extracted from the main treebank, which is the most covering one. Each treebank has been divided in two part, the transduced one and the non-transduced one. For the Supertagger, we used a β equal to 0.01: even if the supertagging and the parsing steps are slower, the results are much better than a β of 0.05. The results are gathered in Table 6. We note that non-transduced sentences are nevertheless analyzed, even if the results are less accurate than for the other sentences.

We also tested the precision of the Supertagger (for the whole tests, see [18]): the Supertagger can adjust the number of types given to a word, with the β parameter. It enables to select formula which the confidence of the Supertagger is at least β times the confidence of the first supertag. We summarize the time spent to analyze the fragment of 440 sentences of the evaluation corpus, the effectiveness of the algorithm by modifying β in Table 7. The high number of types is due to the limitations of AB-grammar. When the Supertagger is used for multimodal categorial grammars, the average number of formulas is around 2.4 with a β equal to 0.01 and 4.5 with $\beta = 0.001$; the correctness is better too, with respectively a rate of 98.2% and 98.8%.

Table 6. Results.

Origin of sentences	Number of sentences	**Success Rate**
Main treebank	11,456	90.1%
Evaluation treebank	446	83.6%
Sequoia treebank	3,014	95.5%
Non transduced main treebank	892	62.5%
Non transduced evaluation treebank	98	52.0%
Non transduced Sequoia treebank	198	94.4%

Table 7. Summary of time spent, given the average number of types for a word. The correctness of types is evaluated by the Supertagger.

Average number of types	Correctness	Execution time	Analyzed sentences
1 ($\beta = 1$)	76.9%	0.31 sec	30.2%
1.8 ($\beta = 0.1$)	87.0%	0.78 sec	65.7%
2.4 ($\beta = 0.05$)	88.9%	1.22 sec	72.7%
4.5 ($\beta = 0.01$)	91.7%	4.27 sec	83.6%
10.6 ($\beta = 0.001$)	93.4%	20.18 sec	96.8%
19.2 ($\beta = 0.0001$)	93.8%	55.32 sec	98.0%

6 Conclusion and prospects

In this article, we have briefly introduced the G-transducer principle, that we used to transform syntactic trees into derivation trees of an AB grammar. Then we explained how we extracted a PCFG and used it in the sentence analysis. The experimental results of the CYK algorithm used with typed sentences enabled us to compare the annotation from the transducer and the Supertagger.

However, the work is still ongoing, and opens many horizons. Of course, we want to extend the coverage of the transducer to exceed 95% and simplify the types of words. The main problem is that only complex cases remain, but we should be able to find derivation trees, given that we can analyze a part of them with the CYK algorithm. In order to improve parsing precision, we intend to integrate modern techniques such as those of [3, 20] into our parser. Using the Charniak method [6], we would like to transform our grammar into a highly lexicalized grammar.

Given that AB grammars may seem limiting in the case of a complex language, we wish to transform our transducer into a tree to graph transducer. This way, we would be able to use the whole Lambek calculus.

The *XML* version of the Treebank gives more informations on the words, like the tense of verbs. A major evolution would be to reflect this information into our transducer, even if it implies many transformation for it.

Our work and programs are available on [21], under *GNU General Public Licence.*

References

1. Abeillé, A., Clément, L., Toussenel, F.: Building a treebank for French. Treebanks, Kluwer, Dordrecht (2003).
2. Abeillé, A., Clément, L., Toussenel, F.: Annotation Morpho-syntaxique. http://llf.linguist.jussieu.fr (2003).
3. Auli M. and Lopez A.: Efficient CCG Parsing: A* versus Adaptive Supertagging. In Proceedings of the Association for Computational Linguistics (2011).
4. Buszkowski, W., Penn, G.: Categorial grammars determined from linguistic data by unification. Studia Logica (1990).
5. Candito M. and Seddah D.: Le corpusSequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical TALN'2012 proceedings, Grenoble, France (2012).
6. Charniak E. A maximum-entropy-inspired parser. In Proceedings of the 1st Annual Meeting of the North American Chapter of the ACL (NAACL), Seattle (2000).
7. Comon, H., Dauchet, M., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (1997).
8. Hockenmaier J.: Data and Models for Statistical Parsing with Combinatory Categorial Grammar. PhD thesis, 2003.
9. Hopcroft J.E. and Ullman J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company (1979).
10. Klein D. and Manning C.: Accurate Unlexicalized Parsing. In Proceedings of the Association for Computational Linguistics (2003).
11. Knuth D.E.: The Art of Computer Programming Volume 2: Seminumerical Algorithms (3rd ed.) Addison-Wesley Professional (1997).
12. Kanazawa, M.: Learnable Classes of Categorial Grammars. Center for the Study of Language and Information (1998).
13. Lambek, J.: The Mathematics of Sentence Structure. (1958).
14. Levy R., Andrew G.: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. http://nlp.stanford.edu/software/tregex.shtml (2006).

15. Moot, R.: Automated extraction of type-logical supertags from the Spoken Dutch Corpus. Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach (2010).
16. Moot, R.: Semi-automated Extraction of a Wide-Coverage Type-Logical Grammar For French. Proceedings TALN 2010, Montreal (2010).
17. Sandillon-Rezer, N.F.: Learning categorial grammar with tree transducers. ESSLLI Student Session Proceedings (2011).
18. Sandillon-Rezer, N-F.: *Extraction de PCFG et analyse de phrases pré-typées* Recital 2012, Grenoble (2012).
19. Younger D.H.: Context Free Grammar processing in n^3. (1968).
20. Zang Y. and Clarck S.: Shift-Reduce CCG Parsing. In Proceedings of the Association for Computational Linguistics (2011).
21. Sandillon-Rezer, N.F.: http://www.labri.fr/perso/nfsr/ (2012).