Learning categorial grammar with tree transducers

Noémie-Fleur Sandillon-Rezer
nfsr@labri.fr
LaBRI, CNRS, INRIA

Abstract. In this article, we describe how to use tree transducers to transform linguistic annotations, in our case the Paris VII corpus [1], into AB grammar derivations. Our implementation of a generalisation of standard tree transducers and a set of transduction rule we create allows us to handle 96% of the total number of words in the corpus.

1 Introduction

The main goal of our current work is to extract a French categorial grammar from the annotated corpus of Paris VII.

In 1958, Lambek [9] introduced an algorithm to distinguish sentences from non-sentences. The basic principle of categorial grammar is that the lexicon assigns a finite set of types to words. Types are recursively defined from a set of basic types, like \( s \) for sentence (e.g., “Jean aime Marie”), \( np \) for noun phrase (e.g., “Jean” or “l’étudiant”), and so on.

For the recursive case, if \( A \) and \( B \) are types, then both \( A/B \) and \( B\backslash A \) are types as well. The derivation rules, which model the intuitions behind the types just given, are shown in Figure 1. The fragment of the Lambek calculus containing only these rules is often called AB.

\[
\frac{A/B}{A} \quad \frac{B}{[\mathbf{E}]}
\]

\[
\frac{B\backslash A}{A} \quad \frac{B}{[\mathbf{\backslash E}]}
\]

Fig. 1. The elimination rules for AB

There exists many approaches of learning a categorial grammar. The basis of the theoretical branch is the identification in the limit introduced by Gold [5]\(^1\). This work inspired technical applications as the Buskowski and Penn algorithm [2] used for the learning of rigid AB-grammar\(^2\), or the work of Kanazawa [7], used on \( k \)-valued grammars\(^3\). These two algorithms cannot be applied to the Paris VII treebank, mainly because of the shape of the sentences (these two algorithms

---

\(^1\) Much like a child learns his mother tongue: a sequence of correct sentences is presented to a learner, and he learns a grammar that generates the examples. The more sentences the learner sees, the more his grammar is precise. Eventually, the grammar converges, and can generate any unseen sentences.

\(^2\) The words of the lexicon can only have one type.

\(^3\) The words of the lexicon can have at most \( k \) types.
need binary trees as input, and the corpus trees are planar). The more applied research, such as [6], [11] and [12], uses special-purpose algorithms which apply only to the corpus in question, with little hope of reuse of tools for other corpora. Given the differences in annotation formats between corpora and the grammatical differences between languages, adapting a tool from one corpus to another will always be a labour-intensive enterprise. However, it is our hope that the use of the right kind of formal tool can allow one implementation, though with different sets of parameters and rules files, to serve as the basis for corpus extraction over different types of corpora.

This paper explains our method of obtaining derivation trees, from the sentences of the Paris VII corpus. It also evaluates the performance of the method. First, we introduce the corpus. Then we present our generalized tree transducers which we use to learn an AB grammar and our current results. The last section will conclude and discuss some of the possible extensions of the current work.

2 Corpus Presentation

As the starting point for our grammatical inference, we have used the Paris VII Corpus, containing 12,351 French sentences of the newspaper Le Monde from the period between 1989 and 1994, annotated by the Paris VII Formal Linguistics laboratory “Laboratoire de Linguistique Formelle” [1]. The sentences are annotated as planar trees, as shown by the Figure 2. The root is labeled SENT for “sentence”, and the leaves represent the words. Internal nodes give information about their subtrees. It will guide us in our transformation of the corpus into an AB-grammar:

**POS tags:** the part-of-speech tags (POS) information, only given to preterminal nodes. It gives the POS tag of its daughter. For example the label NC announces that the word is a common noun (“journal” in Figure2).

**Phrasal types:** there are the labels of the others internal nodes. They give the syntactic type of the node, like NP for noun phrase (type of “un journal” in the example) or VN for verbal nucleus (like “a crée”). In addition, a second set of labels can give the “role” of the subtree in the sentence, like -SUI, that announces that the subtree will be the subject. Thus, we will write NP-OBJ for a noun phrase which serves as object to its parent node (such as “un journal”).

![Fig. 2. Part of an annotation tree from the Paris VII treebank.](image-url)
3 G-transducer

The main idea of this paper is to use a tree transducer to automatically convert the planar trees of corpora into derivation trees of an AB grammar. Though tree transducers are part of the standard tools of natural language processing (see [8] for an overview oriented towards statistical NLP and [3] for a general overview of tree transducers), they have as far as we know not yet been use for grammar extraction.

A tree transducer is similar to a tree automaton, adding the feature of writing on the output tape while reading a symbol on the input tape. We apply our transducer to sentences from the corpus, and the output will be derivation trees from an AB grammar: a binary tree, where internal nodes are assigned both a formula corresponding to the rules \( /E \) and \( \backslash E \) from Figure 1.

Note that this is the form required for many of the learning algorithms (see [2, 7]), so our transducers are compatible with Gold-style learning as well.

In order to render writing of the transduction rules more convenient, we introduce a slight generalization of the standard top-down tree transducer, which we will call the G-transducer. Instead of writing many different rules for only marginally different cases, these rule generalizations of the G-transducer allow us to apply rules to nodes of arbitrary arity and to have a form of restricted quantification over node labels.

3.1 Formal definition

Definition. A generalized transducer is a tuple \((Q, q_i, Q_f, X, \delta)\) where:

- \(Q\) is the set of states; \(q_i \in Q\) the initial state and \(Q_f \subseteq Q\) the set of final states.
- \(X\) is the alphabet for reading and writing. \(X = \{A \cup M \cup T\}\) with \(A\) for the POS-tags (\(\text{SENT}, \text{NP}, \ldots\)), \(M\) for French words, and \(T\) for types (\(np/n\), etc).
- \(\delta\) is the set of transduction rules, of the form
  \[q(f(x_1, \ldots, x_n)) \rightarrow u[q_1(t_1), \ldots q_p(t_p)]\]
  where for each \(i \in [1, p]\), either \(t_i \in \{x_1, \ldots, x_n\}\) or \(t_i = f_i(x_{i,1}, \ldots x_{i,m})\) with \(x_{i,j} \in \{x_1, \ldots, x_n\}; m \leq n - 1\). In our case, we use a sub-class of these rules:

- \(q(f(x_1, \ldots, x_n)) \rightarrow u[q_1(t_1), \ldots q_p(t_p)]\)
- for all \(i \in [1, p]\),

  \(t_i \in \{x_1, \ldots, x_n\}\)

with \((x_{i,1}, \ldots x_{i,m})\) subsequences of \((x_1, \ldots, x_n)\). We will call these rules “recursive rules”, since \(q(f(\ldots))\) calls \(q(f(\ldots))\).

Properties of the transducer. The G-transducer, as we will use it in what follows, has a number of properties which are shared by other tree transducers:

- \(\varepsilon\)-free: there is no \(\varepsilon\)–rule in our transducer.
- Linear: in the right-hand-side of each rule, each node is unique.
- Complete (non-deleting): the nodes which appear in the left-hand-side of the rule appear in the right-hand-side as well.
- Deterministic: at each state and input, only a single transition applies.
- Finite look-ahead: we allow each transduction rule to have a complex tree as its left-hand-side, i.e. \(f\) and the \(f_i\) can be complex trees with the indicated yield. This corresponds to having finite (as opposed to regular) look-ahead.
Original features. The features which make the $G$-transducer original, and which allow for a compact specification of the transduction rules, are the following.

**Recursivity:** A recursive rule applies to a node with a specified label but with an arbitrary arity. Figure 3 shows an example where the matched pattern consists of the rightmost daughters only (just for explanatory purposes: the pattern can occur on both sides, as shown in Figure 4(b)). The daughters 1 to $n$ of the root $P$ stand for any sequence of nodes, but nodes $n + 1$ to $n + k$ match the specified pattern. This pattern will be treated, then the transducer will search for a new rule to apply to the second node $P$. Though these recursive rules generalize the standard definition of transduction rule in allowing rules to match nodes of arbitrary arity, we can, when given the maximum node arity of the input tree, convert a recursive rule into a number of “ordinary” transduction rules.

![Fig. 3.](image)

**Parametrization:** We allow rules with a restricted quantification over node labels. An example is shown in Figure 4(a). Here, the variable $X$ can range over several node labels. Note that is equivalent to spelling out each of the different instantiations of $X$ into its separate rule, but an important convenience when writing conversion rules.

**Priority System:** To avoid non-determinism, the rules are always applied in the same order (see Figure 4(b)). The only disadvantage of this method is that it always gives wide scope to the same nodes.

![Fig. 4.](image)

3.2 Transduction rules

We have created a number of rules staying as close as possible to the “standard” way of analysing sentences and complex expressions in categorial grammars (see [9, 10, 13]). For example:
– Common nouns (NC in the corpus) will generally be assigned the type $n$.
– The noun phrases (NP) will generally be assigned the type $np$.
– The adjectives will generally have either type $n/n$ or type $n\backslash n$ (French adjectives can occur both before and after the noun), etc.

The output of one of our transducer for a given input tree is a binary branching tree, where all nodes are assigned a type and where each local tree corresponds to an elimination rule. In other words, an AB derivation tree. We will return to the effectiveness of the transducer in Section 3.3.

3.3 Implementation

System overview. To evaluate our method on the whole corpus, we decided to implement a robust system which reads and writes parenthesized format and applies, mainly, the transduction. In addition, our tool includes a corpus corrector, which corrects the most frequent mistakes done during the annotation of the corpus.

![Fig. 5. The corrector transforms the input corpus into a corrected corpus, which is given as an argument to the transducer.](image)

Output. The transducer’s output is divided in several files. The output file and the log file contain respectively the successfully binarized sentences with all nodes typed and the untreated subtrees; the lexicon files contain each word that appears in the output file with the different types they can have (see Figure 6), and so on with the POS tags.

```
2370: est - 398: (np\s)/np, 393: (np\s)/(n\n), 230: (np\s)/(np\s),
166: (np\s)/pp, 114: ((np\s)/pp)/(np\s)...
```

![Fig. 6. Extract of the lexicon: the verb “est” (is), used 2370 times in the corpus, and its most frequent types in the analysis. The type (np\s)/np occurs 398 times in the corpus. It comes from sentences like “la communication est un secteur d’avenir” (communication is a promising sector).](image)

Evaluation. The evaluation of our method is still ongoing. For now, our transducer generates a lexicon of 26,733 words (96.8% of the complete corpus, which contains 27,589 words), even if it analyzes 92% of the sentences completely. The current version of our transducer uses 1,598 rules to analyze 11,369 sentences. For the remaining 994 sentences, due to their complexity, we gauge to annotate them manually.

The information included in the binarised sentences can and will be used to construct a stochastic context-free grammar.

\(^4\) For example, the PONCT node must be the daughter of SENT, in practice it is sometimes more deeply nested. In these cases, we move the punctuation under SENT.
4 Conclusion and Future Work

We have introduced a generalization of the top-down tree transducer and shown how it can be used as a device for automated grammar extraction. We have described the transducer formally and talked about the way it is implemented.

This work opens up several possibilities for future research. The most obvious is the expansion of the coverage towards the entire corpus and the comparison of our results with the semi-automatically extracted types which have been obtained in [12]. Comparing the types obtained using both methods would be a useful way of validating both approaches.

In addition, we want to make the types more detailed, distinguishing for example between verbs which take an infinitival group as an argument and those which take a past participle as an argument.

A final interesting line of research to follow would be to extend the current work on tree-to-tree transducers to tree-to-graph transducers [4]. This would allow us to move from the simple AB grammars towards the different types of Lambek grammars and their modern incarnations.

Our work is available at [14], under the GNU General Public License.

References