

Tree k -Grammar Models for Natural Language Modelling and Parsing

Jose L. Verdú-Mas, Mikel L. Forcada,
Rafael C. Carrasco, and Jorge Calera-Rubio*

Departament de Llenguatges i Sistemes Informàtics, Universitat d'Alacant
E-03071 Alacant, Spain
{verdu,mlf,carrasco,calera}@dlsi.ua.es

Abstract. In this paper, we compare three different approaches to build a probabilistic context-free grammar for natural language parsing from a tree bank corpus: (1) a model that simply extracts the rules contained in the corpus and counts the number of occurrences of each rule; (2) a model that also stores information about the parent node's category, and (3) a model that estimates the probabilities according to a generalized k -gram scheme for trees with $k = 3$. The last model allows for faster parsing and decreases considerably the perplexity of test samples.

1 Introduction

Context-free grammars are the customary way of representing syntactical structure in natural language sentences. In many natural-language processing applications, obtaining the correct syntactical structure for a sentence is an important intermediate step before assigning an interpretation to it. Choosing the correct parse for a given sentence is a crucial task if one wants to interpret the meaning of the sentence, due to the *principle of compositionality* [13, p. 358], which states, informally, that the interpretation of a sentence is obtained by *composing* the meaning of its constituents according to the groupings defined by the parse tree.

But ambiguous parses are *very* common in real natural-language sentences (e.g., those longer than 15 words). Some authors (e.g. [7]) propose that a great deal of syntactic disambiguation may actually occurs without the use of any semantic information; that is, just by selecting a preferred parse tree. It may be argued that the preference of a parse tree with respect to another is largely due to the relative frequencies with which those choices have led to a successful interpretation. This sets the ground for a family of techniques which use a probabilistic scoring of parses to the correct parse in each case.

Probabilistic scorings depend on parameters which are usually estimated from data, that is, from parsed text corpora such as the Penn Treebank [11]. The most straightforward approach is that of *tree bank grammars* [6]. Treebank

* The authors wish to thank the Spanish CICYT for supporting this work through project TIC2000-1599.

grammars are probabilistic context-free grammars in which the probabilities that a particular nonterminal is expanded according to a given rule are estimated as the relative frequency of that expansion by simply counting the number of times it occurs in a manually-parsed corpus. This is the simplest probabilistic scoring scheme, and it is not without problems. Better results were obtained with *parent-annotated* labels [8] where each node stores contextual information in the form of the category of the node's parent. This fact is in agreement with the observation put forward by Charniak [6] that simple PCFGs, directly obtained from a corpus, largely overgeneralize. This property suggests that, in these models, a large probability mass is assigned to incorrect parses and, therefore, any procedure that concentrates the probability on the correct parses will increase the likelihood of the samples.

In this spirit, we introduce a generalization of the classic k -gram models, widely used for string processing [2], to the case of trees. The PCFGs obtained in this way consist of rules that include information about the context where the rule is applied. One might call these PCFGs *offspring-annotated* CFGs (by analogy to Johnson's [8] *parent-annotation* concept).

2 A Generalized k -Gram Model

Recall that k -gram models are stochastic models for the generation of sequences $s_1, s_2, \dots$ based on conditional probabilities, that is:

1. the probability $P(s_1 s_2 \dots s_t | M)$ of a sequence in the model M is computed as a product $p_M(s_1) p_M(s_2 | s_1) \dots p_M(s_t | s_1 s_2 \dots s_{t-1})$, and
2. the dependence of the probabilities p_M on previous history is assumed to be restricted to the immediate preceding context, in particular, the last $k - 1$ words: $p_M(s_t | s_1 \dots s_{t-1}) = p_M(s_t | s_{t-k+1} \dots s_{t-1})$.

Note that in this kind of models, the probability that the observation s_t is generated at time t is computed as a function of the subsequence of length $k - 1$ that immediately precedes s_t (this is called a *state*). However, in the case of trees, it is not obvious what context should be taken in to account. Indeed, there is a natural preference when processing strings (the usual left-to-right order) but there are at least two standard ways of processing trees: ascending (or bottom-up) analysis and descending (or top-down) analysis. Ascending tree automata recognize a wider class of tree languages [12] and, therefore, they allow for richer descriptions.

Therefore, our model will compute the expansion probability for a given node as a function of the subtree of depth $k - 2$ that the node generates¹, i.e., every *state* stores a subtree of depth $k - 2$. In the particular case $k = 2$, only the label of the node is taken into account (this is analogous to the standard bigram model for strings) and the model coincides with the simple rule-counting approach used

¹ Note that in our notation a single node tree has depth 0. This is in contrast to strings, where a single symbol has length 1.

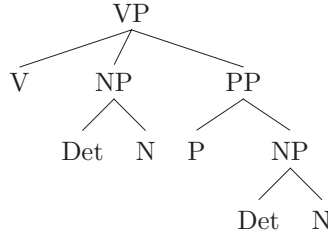


Fig. 1. A sample parse tree of depth 3

in treebank grammars. For instance, for the tree depicted in Fig. 1, the following rules are obtained:

$$\begin{aligned}
 \text{VP} &\rightarrow \text{V NP PP} \\
 \text{NP} &\rightarrow \text{Det N} \\
 \text{PP} &\rightarrow \text{P NP}
 \end{aligned}$$

However, in the case $k = 3$, which will be called *child-annotated* model, the expansion probabilities depend on states that are defined by the node label, the number of descendents the node and the sequence of labels in the descendents (if any). Therefore, for the same tree the following rules are obtained in this case:

$$\begin{aligned}
 \text{VP}_{\text{V,NP,PP}} &\rightarrow \text{V NP}_{\text{Det,N}} \text{PP}_{\text{P,NP}} \\
 \text{NP}_{\text{Det,N}} &\rightarrow \text{Det N} \\
 \text{PP}_{\text{P,NP}} &\rightarrow \text{P NP}_{\text{Det,N}}
 \end{aligned}$$

where each state has the form $X_{Z_1, \dots, Z_m}$. This is equivalent to performing a relabelling of the parse tree before extracting the rules.

Finally, in the parent-annotated model (PA) described in [8] the states depend on both the node label and the node's parent label:

$$\begin{aligned}
 {}^s\text{VP} &\rightarrow \text{V } {}^{\text{VP}}\text{NP } {}^{\text{VP}}\text{PP} \\
 {}^{\text{VP}}\text{NP} &\rightarrow \text{Det N} \\
 {}^{\text{VP}}\text{PP} &\rightarrow \text{P } {}^{\text{PP}}\text{NP} \\
 {}^{\text{PP}}\text{NP} &\rightarrow \text{Det N}
 \end{aligned}$$

It is obvious that the $k = 3$ and PA models incorporate contextual information that is not present in the case $k = 2$ and, then, a higher number of rules for a fixed number of categories is possible. In practice, due to the finite size of the training corpus, the number of rules is always moderate. However, as higher values of k lead to a huge number of possible rules, huge data sets would be necessary in order to have a reliable estimate of the probabilities for values above $k = 3$. A detailed mathematical description of *offspring-annotated* models can be found in [14].

3 Experimental Results

3.1 General Conditions

We have performed experiments to assess the structural disambiguation performance of k -gram models as compared to standard treebank grammars and Johnson's [8] parent-annotation scheme, that is, to compare their relative ability for selecting the best parse tree. We have also used the perplexity as an indication of the quality of each model. To build training corpora and test sets of parse trees, we have used English parse trees from the Penn Treebank, release 3, with small, basically structure-preserving modifications:

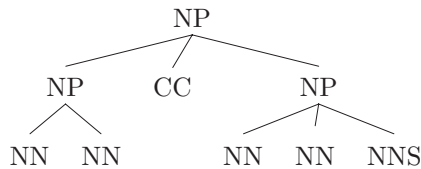
- insertion of a root node (ROOT) in all sentences, (as in Charniak [6]) to encompass the sentence and final periods, etc.;
- removal of nonsyntactic annotations (prefixes and suffixes) from constituent labels (for instance, NP-SBJ is reduced to NP);
- removal of empty constituents; and
- collapse of single-child nodes with the parent node when they have the same label (to avoid rules of the form $A \rightarrow A$ which would generate an infinite number of parse trees for some sentences).

In all experiments, the training corpus consisted of all of the trees (41,532) in sections 02 to 22 of the *Wall Street Journal* portion of Penn Treebank, modified as above. This gives a total number of more than 600,000 subtrees. The test set contained all sentences in section 23 having less than 40 words.

3.2 Structural Disambiguation Results

All grammar models were rewritten as standard context-free grammars, and Chappelier and Rajman's [5] probabilistic extended Cocke-Younger-Kasami parsing algorithm was used to obtain all possible parse trees for each sentence in the test sets and to compute their individual and total probabilities; for each sentence, the most likely parse was compared to the corresponding tree in the test set using the customary PARSEVAL evaluation metric [1,10, p. 432] after eliminating any parent and child annotation of nodes in the most likely tree delivered by the parser. PARSEVAL gives partial credit to incorrect parses by establishing three measures:

- *labeled precision* (P) is the fraction of correctly-labeled nonterminal bracketings (constituents) in the most likely parse which match the parse in the treebank,
- *labeled recall* (R) is the fraction of brackets in the treebank parse which are found in the most likely parse with the same label, and
- *crossing brackets* (X) refers to the fraction of constituents in one parse cross over constituent boundaries in the other parse.

**Fig. 2.**

The crossing brackets measure does not take constituent labels into account and will not be shown here. Some authors (see, e.g. [4]) have questioned partial-credit evaluation metrics such as the PARSEVAL measures; in particular, if one wants to use a probability model to perform structural disambiguation before assigning some kind of interpretation of the parsed sentence, it may well be argued that the exact match between the treebank tree and the most likely tree is the only possible relevant measure. It is however, very well known that the Penn Treebank, even in its release 3, still suffers from problems. One of the problems worth mentioning (discussed in detail by Krotov et al. [9]) is the presence of far too many partially bracketed constructs according to rules like $\text{NP} \rightarrow \text{NN NN CC NN NN NNS}$, which lead to very flat trees, when one can, in the same treebank, find rules such as $\text{NP} \rightarrow \text{NN NN}$, $\text{NP} \rightarrow \text{NN NN NNS}$ and $\text{NP} \rightarrow \text{NP CC NP}$, which would lead to more structured parses such as the one in Fig. 2. Some of these flat parses may indeed be too flat to be useful for semantic purposes; therefore, if one gets a more refined parse, it may or may not be the one leading to the correct interpretation, but it may never be worse than the flat, unstructured one found in the treebank.

For this reason, we have chosen to give, in addition to the exact-match figure, the percentage of trees having 100% recall, because these are the trees in which the most likely parse is either exactly the treebank parse or a refinement thereof in the sense of the previous example.

Here is a list of the models which were evaluated:

- A standard treebank grammar, with no annotation of node labels ($k=2$), with probabilities for 15,140 rules.
- A child-annotated grammar ($k=3$), with probabilities for 92,830 rules.
- A parent-annotated grammar (PARENT), with probabilities for 23,020 rules.
- A both parent- and child-annotated grammar (BOTH), with probabilities for 112,610 rules.

As expected, the number of rules obtained increases as more information is conveyed by the node label, although this increase is not extreme. On the other hand, as the generalization power decreases, some sentences in the test set become unparseable, that is, they cannot be generated by the grammar. The results in table 1 show that:

- The parsing performance of parent-annotated and child-annotated PCFG is similar and better than those obtained with the standard treebank PCFG.

Table 1. Parsing results with different annotation schemes: labelled recall R , labelled precision P , fraction of sentences with total labelled recall $f_{R=100\%}$, fraction of exact matches, fraction of sentences parsed, and average time per sentence in seconds

MODEL	R	P	$f_{R=100\%}$	EXACT	PARSED	t
$k=2$	70.7%	76.1%	10.4%	10.0%	100%	57
$k=3$	79.6%	74.3%	19.9%	13.4%	94.6%	7
PARENT	80.0%	81.9%	18.5%	16.3%	100%	340
BOTH	80.5%	74.5%	22.7%	15.5%	79.6%	4

This performance is measured both with the customary PARSEVAL metrics and by counting the number of maximum-likelihood trees that (a) match their counterparts in the treebank exactly, and (b) contain all of the constituents in their counterpart (100% labeled recall, $f_{R=100\%}$). The fact that child-annotated grammars do not perform better than parent-annotated ones may be due to their larger number of parameters compared to parent-annotated PCFG, which may make them hard to estimate accurately from currently available treebanks (there are, on average, only about 6 subtrees per rule in the experiments).

- The average time to parse a sentence shows that child annotation leads to parsers that are much faster. This is not surprising because the number of possible parse trees considered is drastically reduced; this is, however, not the case with parent-annotated models.

It may be worth mentioning that an analysis of parse trees produced by child-annotated models tend to be more structured and refined than parent-annotated and unannotated parses which tend to use rules that lead to flat trees in the sense mentioned.

3.3 Perplexity Results

We have also used the perplexity of a test sample $S = \{w_1, \dots, w_{|S|}\}$ as an indication of the quality of the model, $P = \frac{1}{|S|} \sum_{l=1}^{|S|} \log_2 p(w_l|M)$, where $p(w_l|M)$ is the sum of the probabilities of all of the parse trees of the sentence w_l . Since unparsable sentences would produce an infinite perplexity, we have studied the perplexity of the test set for linear combinations of two models M_i and M_j with $p(w_l|M_i, M_j) = \lambda p(w_l|M_i) + (1 - \lambda)p(w_l|M_j)$. The mixing parameter $\lambda \in [0, 1]$ was chosen, in steps of 0.05, in order to minimize the perplexity.

The best results were obtained with a mixture of the child-annotated ($k = 3$) and the parent-annotated models with a heavier component (65%) of the first one. When parsing, the recall and precision of that mixture were respectively 82.1% and 81.0% and the fraction of sentences with total labelled recall $f_{R=100\%}$ scored 22.2%, similar to using both annotation models at the same time but

covering all the test set. The minimum perplexity $P_{\min}$ and the corresponding value of λ obtained are shown in the table 2.

Table 2. Mixture parameter $\lambda_{\min}$ that gives the minimum test set perplexity for each linear combination. The lowest perplexity was obtained with a combination of the $k=3$ and parent-annotation models. All mixture models covered all the set test

MIXTURE MODEL	$P_{\min}$	$\lambda_{\min}$
$k = 2$ and $k = 3$	90.8	0.25
$k = 2$ and PARENT	108.7	0.6
$k = 2$ and BOTH	94	0.3
$k = 3$ and PARENT	88	0.65

4 Conclusion

We have introduced a new probabilistic context-free grammar model, offspring-annotated PCFG, in which the grammar variables are specialized by annotating them with the subtree they generate up to a certain level. In particular, we have studied offspring-annotated models with $k = 3$, that is, child-annotated models, and have compared their parsing performance to that of unannotated PCFG and of parent-annotated PCFG [8]. Child-annotated models are related to probabilistic bottom-up tree automata [12]. The experiments show that:

- The parsing performance of parent-annotated and child-annotated PCFG is similar.
- Parsers using child-annotated grammars are much faster because the number of possible parse trees considered is drastically reduced; this is, however, not the case with parent-annotated models.
- Child-annotated grammars have a larger number of parameters than parent-annotated PCFG which makes it difficult to estimate them accurately from currently available treebanks.
- Child-annotated models tend to give very structured and refined parses instead of flat parses, a tendency not so strong for parent-annotated grammars.
- The perplexity of the test sample decreases when a combination of models with child-annotated and parent-annotated is used to predict string probabilities.

We plan to study the use of statistical confidence criteria as used in grammatical inference algorithms [3] to eliminate unnecessary annotations by merging states, therefore reducing the number of parameters to be estimated. Indeed, offspring-annotation schemes (for a value of $k \geq 3$) may be useful as starting

points for those state-merging mechanisms, which so far have always started with the complete set of different subtrees found in the treebank (ranging in the hundreds of thousands). We also plan to study the smoothing of offspring-annotated PCFGs and to design parsers which can profit from these.

References

1. Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proc. Speech and Natural Language Workshop 1991*, pages 306–311, San Mateo, CA, 1991. Morgan Kauffmann. 59
2. Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based n -gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992. 57
3. Rafael C. Carrasco, Jose Oncina, and Jorge Calera-Rubio. Stochastic inference of regular tree languages. *Machine Learning*, 44(1/2):185–197, 2001. 62
4. John Carroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: A survey and a new proposal. In *Proceedings of the International Conference on Language REsources and Evaluation*, pages 447–454, Granada, Spain, 1998. 60
5. J.-C. Chappelier and M. Rajman. A generalized CYK algorithm for parsing stochastic CFG. In *Actes de TAPD'98*, pages 133–137, 1998. 59
6. Eugene Charniak. Treebank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036. AAAI Press/MIT Press, 1996. 56, 57, 59
7. L. Frazier and K. Rayner. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14:178–210, 1982. 56
8. Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998. 57, 58, 59, 62
9. Alexander Krotov, Robert Gaizauskas, Mark Hepple, and Yorick Wilks. Compacting the Penn Treebank grammar. In *Proceedings of COLING/ACL'98*, pages 699–703, 1998. 60
10. Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 59
11. Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19:313–330, 1993. 56
12. Maurice Nivat and Andreas Podelski. Minimal ascending and descending tree automata. *SIAM Journal on Computing*, 26(1):39–58, 1997. 57, 62
13. A. Radford, M. Atkinson, D. Britain, H. Clahsen, and A. Spencer. *Linguistics: an introduction*. Cambridge Univ. Press, Cambridge, 1999. 56
14. J.R. Rico-Juan, J. Calera-Rubio, and R.C. Carrasco. Probabilistic k -testable tree-languages. In A.L. Oliveira, editor, *Proceedings of 5th International Colloquium, ICGI 2000, Lisbon (Portugal)*, volume 1891 of *Lecture Notes in Computer Science*, pages 221–228, Berlin, 2000. Springer. 58