A Gibbsian Context-Free Grammar for Parsing

Antoine Rozenknop[1]

Ecole Polytechnique Fédérale de Lausanne, I&C-IIF-LIA, CH-1015 Lausanne, Suisse,
Antoine.rozenknop@epfl.ch

Abstract. Probabilistic Context-Free Grammars can be used for speech recognition or syntactic analysis thanks to especially efficient algorithms. In this paper, we propose an instanciation of such a grammar, whose mathematical properties are intuitively more suitable for those tasks than SCFG's (Stochastic CFG), without requiring specific analysis algorithms. Results on Susanne text show that up to 33% of analysis errors made by a SCFG can be avoided with this model.

1 Motivations

Stochastic Context-Free Grammars (SCFGs) are far from beeing up-to-date models for the desciption of natural languages, but they still remain interesting models for Parsing and Speech Recognition [4], thanks to particularly efficient algorithms they provide for those tasks [6, 11, 2]. They can also be used as computational representations of richer grammars, such as Polynomial Tree Substitution Grammars [3].

However, their mathematical properties seem a bit strange and lower the quality of the results they provide when used for such tasks [10].

This article describes a new probabilisation of Context-Free Grammars, which essentially is a non-generative variation of the SCFGs, and which will be denoted as GCFG (for "Gibbsian CFG"). In this model, each context-free rule of the CFG is mapped to a "potential" instead of a probability, and the learning criteria is turned to fit the analysis task instead of a generative task. This model should have a better parsing behaviour while taking benefit of the efficiency of SCFGs parsing algorithms.

In the remaining of this paper, an example of a non-intuitive behaviour of SCFGs is shown in section 2, the GCFG model is described, along with a learning algorithm, in section 3, SCFGs and GCFGs are experimentally compared in section 4, and conclusions are given in section 5.

2 Non-intuitive Behaviour of a SCFG

This example is extracted from M. Johnson's study [7], and illustrates a seemingly paradoxal behaviour of SCFGs. Suppose we have a treebank $\widetilde{\tau}_1$ with two trees (A) et (B) (Fig.1), where (A) appears with relative frequency f. An SCFG is trained on this corpus with the usual method, which consists in assigning to

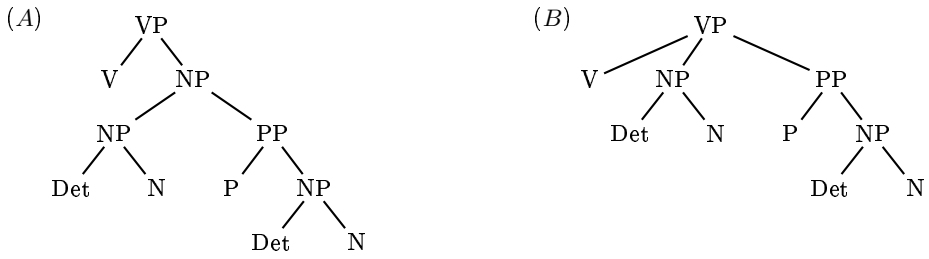

Fig. 1. Training corpus $\widetilde{\tau}_1$. This corpus contains the tree (A) with relative frequency f, and (B) with r.f. $(1-f)$.

the rules probabilities proportionnal to their frequency in the corpus. Rules probabilities ($\hat{P}_1$) obtained from this corpus are as follows : $\hat{P}_1(\text{VP} \rightarrow \text{V NP}) = f$, $\hat{P}_1(\text{VP} \rightarrow \text{V NP PP}) = 1-f$, $\hat{P}_1(\text{NP} \rightarrow \text{Det N}) = 2/(2+f)$ and $\hat{P}_1(\text{NP} \rightarrow \text{NP PP}) = f/(2+f)$. Thus, the probablitities of producing (A) and (B) with the resulting model are : $\hat{P}_1(A) = 4f^2/(2+f)^3$ and $\hat{P}_1(B) = 4(1-f)/(2+f)^2$. The estimated relative frequency of (A) among trees yielding `V Det N P Det N` is : $\hat{f}_1 = \hat{P}_1(A)/(\hat{P}_1(A) + \hat{P}_1(B)) = f^2/(2-f)$. Ideally, $\hat{f}_1$ should be close to the observed relative frequency f. Fig.2 shows $\hat{f}_1$ as a function of f; as can be seen, $\hat{f}_1$ and f can differ substantially. For instance, if $f = 0,75$, $\hat{f}_1 = 0,45$, i.e. even if (A) appears three times as often as (B) in the training corpus, the yields `V Det N P Det N` will be parsed as (B).

M. Johnson suspects that this behaviour is due to the non-systematicity of the structures in the training corpus : in tree (A), (NP $\Rightarrow^*$ Det N PP) follows Chomsky's Adjonctive Form, whereas (VP $\Rightarrow^*$ V NP PP) has a flat structure in (B). To test this hypothesis, the corpus is modified, either by flattening (NP $\Rightarrow^*$ Det N PP) in (A) (i.e. representing the structure by the unique rule NP $\rightarrow$ Det N PP), or by replacing (VP $\rightarrow$ V NP PP) in (B) with (VP $\rightarrow$ VP PP) and (VP $\rightarrow$ V NP). Each of the two modified corpora thus obtained are used to train an SCFG, and the estimated relative frequency of the first tree as a function of its observed frequency is respectively : $\hat{f}_2 = (f^2-2f)/(2f^2-f-2)$, and $\hat{f}_3 = f^2/(2-3f+2f^2)$. Those estimated frequencies are closer to f than $\hat{f}_1$, but remain lower, as illustrated by Fig.2. In each case, when the observed frequency of the NP-attachment of PP is 0.6, the computed model will affect a higher score to the VP-attachment of PP.

3 Gibbsian Context-Free Grammar (GCFG)

We now describe the GCFG model, which is strongly inspired from SCFGs. The grammar is composed of a set of N_r rewriting rules $X \rightarrow Y_1 ... Y_{|r|}$, N_s terminal and non-terminal symbols, terminal symbols only appearing in right parts of rules. Moreover, each rule r_i is associated with a potential value λ_i (instead of a probability as in SCFGs).

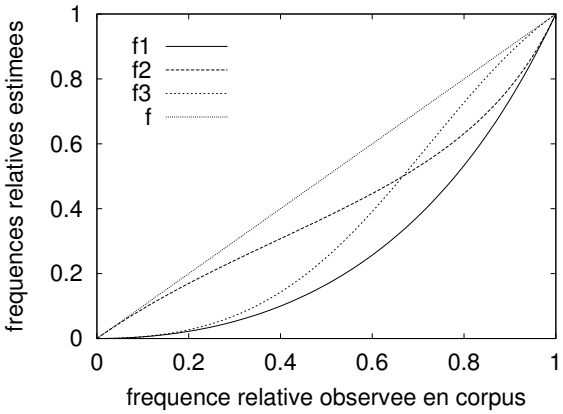

Fig. 2. Estimated relative frequencies of NP-attachment of the PP group, as a function of its observed relative frequency in the different training corpora.

3.1 Potential of a Tree and Conditional Probability

Contrary to a SCFG, this model is not generative. One thus does not seek to define the probability of producing a tree with this model. On the other hand, one defines the **potential of an analysis tree** x as the sum of the potentials of the rules which constitute it. It is thus the scalar product $\lambda \cdot f(x)$ of two vectors of size N_r (the number of context-free rules), component i of λ being the potential λ_i of rule r_i, and component i of $f(x)$ being the number of occurrences f_i of the rule r_i in the tree x.

One defines moreover the probability of an analysis tree x conditionnally to its yields $w = (w_1 ... w_n)$ (i.e. w stands for the analysed sentence) with the formula :

$$p(x|w) = \frac{e^{\lambda \cdot f(x)}}{\sum_{y \Rightarrow^* w} e^{\lambda \cdot f(y)}}$$

where $\sum_{y \Rightarrow^* w}$ is the sum over all trees y of the grammar whose yields are w.

3.2 Syntactic Analysis with a GCFG

The syntactic analysis of a sentence w consists in finding the tree $\bar{x}$ with the highest potential among possible analysis trees. As stated by the previous formula, this is equivalent to finding the tree with the highest conditionnal probability :

$$\bar{x} = \underset{x \Rightarrow^* w}{\text{Argmax}}\, \lambda \cdot f(x) = \underset{x \Rightarrow^* w}{\text{Argmax}}\, p_\lambda(x|w) = \underset{x \Rightarrow^* w}{\text{Argmax}} \prod_{r_i \in x} e^{\lambda_i f_i(x)}$$

The last expression shows how close this model is to a SCFG, where the solution of the syntactic analysis is : $\bar{x} = \text{Argmax}_{x \Rightarrow^* w}\, p(x) = \text{Argmax}_{x \Rightarrow^* w} \prod_{r_i \in x} p(r_i)^{f_i(x)}$

GCFGs thus do not require the development of specific analysis algorithms : one can use as is any SCFG's algorithm, provided it does not make use of the condition $p(r_i) \leq 1$. In particular, $A*$-algorithms use this condition by making the assumption that the score monotonically decreases with the length of the hypotheses, and cannot be used here. A bottom-up chart parser, described in [2], has been used for our tests, by simply replacing the rule probabilities $p(r_i)$ with e^{λ_i}.

3.3 Parameters Learning

Learning Principle : From a learning corpus, constituted of sentences (W) and their analysis trees (X), we seek to compute the parameters λ so as to maximize the probability of the trees conditionnally to the sentences :

$$\bar{\lambda} = \underset{\lambda}{\text{Argmax}}\, p_\lambda(X|W) = \underset{\lambda}{\text{Argmax}} \sum_{x \in X} \ln p_\lambda(x|w(x)) = \underset{\lambda}{\text{Argmax}}\, \mathcal{A}(\lambda)$$

(by writing $w(x)$ the yields of x, i.e. its leaves.) $\mathcal{A}(\lambda)$ is the "conditionnal log-likelihood" of the corpus; note that this is one of the main difference between GCFGs and SCFGs, for which one usually seeks to maximize the probability $p_\lambda(X)$ of the corpus, the solution beeing easily obtained by affecting to each rule a probability proportionnal to its observed frequency.

Improved Iterative Scaling principle : As directly maximizing $\mathcal{A}(\lambda)$ is too difficult, we start from an initial model λ_0 (the choice of this initial model can be crucial, but is not discussed here), and iteratively improve it. A step of iteration consists in passing from a model λ to a model λ', by trying to maximize $\mathcal{D}_\lambda(\lambda') = \mathcal{A}(\lambda') - \mathcal{A}(\lambda) = \sum_{x \in X} \ln \frac{p_{\lambda'}(x|w(x))}{p_\lambda(x|w(x))}$. As this maximization is again untractable, a step of IIS will maximize the intermediary function $\mathcal{B}_\lambda(\lambda')$ which is a minorant of $\mathcal{D}_\lambda(\lambda')$:

$$\mathcal{B}_\lambda(\lambda') = \sum_{x \in X} (\lambda' - \lambda) \cdot f(x) - \sum_{x \in X} \sum_{y \to w(x)} p_\lambda(y|w(x)) \sum_i \frac{f_i(y)}{f^\#(y)} e^{(\lambda'_i - \lambda_i) f^\#(y)} + |X|$$

where $f^\#(y) = \sum_i f_i(y)$ is the number of rules appearing in a tree y. Details about the mathematical derivations of $\mathcal{B}_\lambda(\lambda')$ from $\mathcal{D}_\lambda(\lambda')$ are well explained in [1, 9, 8].

Maximizing $\mathcal{B}_\lambda(\lambda')$ is done by finding the point where its partial derivatives are null, i.e. by solving for each rule r_i :

$$0 = -\sum_{x \in X} f_i(x) + \sum_{x \in X} \sum_{y \to w(x)} p_\lambda(y|w(x)) f_i(y) (e^{\lambda'_i - \lambda_i})^{f^\#(y)} \qquad (1)$$

Due to the convexity of the polynoms involved in these equations, their solutions can easily be computed with a Newton's method.

Inside-Outside Algorithm : The first term of the polynom, $-\sum_{x \in X} f_i(x)$, is trivially obtained as the frequency of r_i in the training corpus. But the other coefficients are far more complex to compute : the term $\sum_{y \Rightarrow^* w} ...$ requires a summation over all possible analyses of w, which can be an exponential problem. This step is sometimes approximatively solved by sampling methods [9]. Here, we can happily factorize the computation by using an Inside-Outside algorithm, as shown by the following manipulations.

Let us rewrite the third sum of (1) :

$$\mathcal{S}_{w,\lambda}(\alpha) = \sum_{y \to w} p_\lambda(y|w) f_i(y) \alpha^{f^\#(y)} = \Big(\sum_{y \to w} e^{\lambda \cdot f(x)} \Big)^{-1} \sum_{y \to w} e^{\lambda \cdot f(y)} f_i(y) \alpha^{f^\#(y)}$$
$$= Z_{\lambda,w}^{-1} \sum_{1 \le j \le k \le |w|} \sum_{y \Rightarrow^* w} V(y) C(y, [j, r_i, k])$$

where $C(y, [j, r_i, k])$ is the number of occurrences of r_i in y at position (j, k) (i.e. when r_i yields $w_j ... w_k$ in y), where $V(y) = e^{\lambda \cdot f(y)} \alpha^{f^\#(y)} = \prod_{r_i \in y} (f_i \alpha)^{f_i(y)}$ is the product of the polynoms $P_i(\alpha) = (\lambda_i \alpha)$ associated with the rules r_i that constitute y, and $Z_{\lambda,w} = \sum_{y \Rightarrow^* w} e^{\lambda \cdot f(x)}$.

Following [5] (pp.26-57) $\sum_{y \Rightarrow^* w} V(y) C(y, [j, r_i, k])$ can then be computed for each r_i, j and k with an Inside-Outside algorithm, which we can sketch by :

- For each triplet (j, r_i, k), compute $inside_r[j, r_i, k]$ as the sum of the values $V(y)$ of trees y whose top rule is r_i, and whose leaves are $w_j ... w_k$.
- For each triplet (j, A, k), compute $outside[j, A, k]$, as the sum of the values $V(y)$ of trees y whose leaves are $w_1 ... w_{j-1} A w_{k+1} ... w_n$.
- Noting $G(r_i)$ the left-side symbol of r_i, the result is obtained with : $\sum_{y \Rightarrow^* w} V(y) C(y, [j, r_i, k]) = inside_r[j, r_i, k] * outside[j, G(r_i), k]$

Improved Iterative Scaling Algorithm : Finally, the training algorithm can be summed up as follows :

`Define an initial model` λ'
Repeat :
  $\lambda \leftarrow \lambda'$.
  `/* from` λ `to` λ' `*/`
  `Initialize to zero the polynoms` $S^r(\alpha)$ `associated with rules` r.
  For each `example` x `of the learning corpus :`
    Analyze $w(x)$ `with the Inside-Outside algorithm.`
    `Compute` $Z_{\lambda,w(x)}$ `as the sum of the coefficients of the polynom` $\sum_r inside_r[1, r, n]$.
    Foe each `element` $[j, r, k]$ `of the Inside-Outside chart`
      $S^r(\alpha) := S^r(\alpha) + Z_{\lambda,w(x)}^{-1} \sum_{y \Rightarrow^* w} V(y) C(y, [j, r_i, k])$
        $(= S^r(\alpha) + Z_{\lambda,w(x)}^{-1} inside_r[j, r, k] * outside[j, G(r), k])$
  Foe each `rule` r_i `of the CFG :`
    `Solve :` $S^{r_i}(\alpha_0) = -sum_{x \in X} f_i(x)$
    `Compute the model parameter` λ' `with :` $\lambda'_i = \lambda_i + \log \alpha_0$
until `convergence of criterium` $\mathcal{A}(\lambda)$.

4 Experimental Results

The GCFG model has been tested on a SUSANNE-derived corpus, containing 4292 trees, 1920 non-terminal symbols, 11935 terminal symbols, 17669 rules. Some unary rules were manually removed so as to obtain a non-looping grammar.

The first test consists of learning the model from the complete corpus, parsing the sentences of the corpus with the obtained parameters, and comparing the resulting trees with the reference trees. Results are grouped under the label $Test = Learn$ in Fig.3. Column `Tx(Par)` represents the rate of sentences receiving a parse, and among them `Tx(Cor)` represents the rate of those whose parse is the correct one. Precision and Recall rates (columns `Pre` et `Rec`) are obtained by considering the sequence of parse trees $\widetilde{\tau}$ as a set $E(\widetilde{\tau})$ of triplets $< g, N, d >$, where N is a non-terminal symbol, and g et d are the positions in the corpus of the first and last words yielded by N. When comparing with the reference trees $\widetilde{\tau}'$, la precision and recall are computed as :

$$Tx(Pre)(\widetilde{\tau}) = \frac{|E(\widetilde{\tau}) \cap E(\widetilde{\tau}')|}{|E(\widetilde{\tau})|}, \quad Tx(Rap)(\widetilde{\tau}) = \frac{|E(\widetilde{\tau}) \cap E(\widetilde{\tau}')|}{|E(\widetilde{\tau}')|}$$

The second test is identical to the first one, except that the training process takes place on 9 tenth of the corpus (randomly chosen), and the test is done on the remaining part. The results appearing in Fig.3 are means of results obtained for ten random splits of the initial corpus. Precision and Recal rates are computed on the basis of trees that receive at least one parse.

The SCFG model was tested on the same corpora, and we give the *diminution of Error-Rates* that GCFG provide in comparison. These values are computed as : $1 - \frac{1-Tx[GCFG]}{1-Tx[SCFG]}$.

	$Test = Learn$				$Test \neq Learn$			
Lexicalized corpus								
Model	Tx(Par)	Tx(Cor)	Pre	Rec	Tx(Par)	Tx(Cor)	Pre	Rec
SCFG	1	0,788	0,989	0,988	0,062	0,408	0,878	0,893
GCFG	1	0,860	0,994	0,994	0,062	0,398	0,874	0,889
Diminution of Error-Rate		33%	48%	49%		−2%	−3%	−3%

Fig. 3. Compared results of SCFGs and GCFGs, for a parsing task.

4.1 Discussion

The example of Sec.2 shows that a SCFG whose parameters are learned from a treebank can exhibit an unexpected behaviour, affecting higher probabilities to seldom forms. On the same example, a GCFG conforms to intuition, i.e. the estimated and observed frequencies of (A) are the same (which we do not

demonstrate here). This difference is mainly explained by the difference between their learning criteria : SCFGs are usually considered as generative models, and their learning stage maximizes the probability of generating the corpus with an underlying stochastic process starting from the root of a tree. Their learning criterium thus is : $p_\lambda(X)$[1]. Such a model makes the assumption that language is generated by a grammatical process.

In the contrary, GCFGs are designed as analysis models, and their learning criterium is $p_\lambda(X|W)$, which intuitively corresponds to the probability of generating the corpus from its leaves. If possible, this criterium is maximized when the estimated frequencies of trees of same yields are proportional to their frequency in the corpus. As there are enough parameters to reach this result in the example, this explains the perfect adequation of the observed and estimated frequencies.

Sec.2 also relates the fact that representing the structures (NP $\Rightarrow^*$ Det N PP) and (VP $\Rightarrow^*$ V NP PP) in a similar way in the corpus (i.e. either both flattened or both under Chomsky's Adjonctive Form) enhances the performance of the trained SCFGs. Nothing comparable can be observed for GCFGs, which behave in the same way in every case. However, tests with more realistic corpora [7] show that only the flattening scheme really enhances the performance of SCFGs. M. Johnson suspects the reason is linked to the weakening of independance assumptions of CF grammars this flattening scheme induces. It should be interesting to measure in the same way the impact of this flattening operation on a GCFG, for which independance assumption are not so strong : the potential of a rule is computed in comparison with all rules of the grammar, and not only with the rules that share the same head, as in SCFGs.

GCFGs perform well in self-test ($Test = Learn$ column of Fig.3) : with the same number of parameters as a SCFG trained in the same conditions, the number of incorrect parses is multiplied by $2/3$, and the label precision and recall error rate are multiplied by $1/2$: GCFGs "stick" undoubtedly better to learning data than SCFGs.

In the generalisation scheme ($Test \neq Learn$ column), the mean behaviour of GCFGs seems slightly worse than for SCFGs, but the ranking differs from one experiment to the other : the observed difference between both models is not significative. This comes from the very low coverage of the underlying CFG : only 6% of test sentences are parsed by this CFG (26 sentences out of 429). Moreover, the number of context-free rules (17669) is huge as compared to the size of the learning corpus ($4292 - 429$ trees), and the computed parameters are likely to be insignificant for generalization.

5 Conclusion

This contribution presents a valuation method for Context-Free Grammars, which differs from SCFGs by its training criterium, suited to a parsing task,

[1] cf Sec.3 for notations

and by the relaxation of stochastic constraints ($p_i \leq 1$, et $\sum p = 1$) imposed to the parameters of a SCFG. The resulting grammars (GCFG) having essentially the same form as SCFGs, we are provided with standard efficient parsing algorithms that can be used without modification. We have also detailed an algorithm for training the GCFG model within a reasonnable time.

Experimental studies show that GCFG parameters better stick to training data and intuition than SCFG ones. On the other side, we have not shown any improvement when parsing unseen sentences, due to the weak coverage of the common underlying CFG. Tests with more covering corpora are under process.

Other applications of the exposed principle are foreseen, such as its adaptation to Polynomial Tree Substitution Grammars [3], which enclose richer linguistic informations than CFGs, but are not theoretically well defined from a probabilistic point of view. We also plan to adapt GCFGs for dealing with a Speech Recognition task, by changing its learning criterium.

References

[1] Adam Berger. Convexity, maximum likelihood and all that.

[2] J.-C. Chappelier and M. Rajman. A generalized CYK algorithm for parsing stochastic CFG. In *Proc. of 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 133–137, Paris (France), apr 1998.

[3] J.-C. Chappelier and M. Rajman. Grammaire à substitution d'arbre de complexité polynomiale : un cadre efficace pour dop. In *Actes de la 8ème conférence sur le Traitement Automatique des Langues Naturelles (TALN'2001)*, volume 1, pages 133–142, 2001.

[4] J.-C. Chappelier, M. Rajman, R. Aragüés, and A. Rozenknop. Lattice parsing for speech recognition. In *Proc. of 6ème conférence sur le Traitement Automatique du Langage Naturel (TALN99)*, pages 95–104, Cargèse (France), jul 1999.

[5] Joshua T. Goodman. *Parsing Inside-Out*. PhD thesis, Harvard University, Cambridge, Massachusetts, May 1998.

[6] F. Jelinek, J. D. Lafferty, and R. L. Mercer. Basic methods of probabilistic context-free grammars. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding: Recent Advances, Trends and Applications*, volume 75 of *F: Computer and System Science*. Springer, 1992.

[7] Mark Johnson. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, December 1998.

[8] John Lafferty. Gibbs-markov models. In *Computing Science and Statistics*, volume 27, pages 370–377, 1996.

[9] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[10] Antoine Rozenknop and Marius-Calin Silaghi. Algorithme de décodage de treillis selon le critère du coût moyen pour la reconnaissance de la parole. In *Actes de la 8ème conférence sur le Traitement Automatique des Langues Naturelles (TALN'2001)*, number 1, pages 391–396, Tours, juillet 2001. Association pour le Traitement Automatique des Langues.

[11] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.