Chapter 11

Gibbsian Tree Substitution Grammars
A discriminant probabilistic model for TSGs.

ANTOINE ROZENKNOP, JEAN-CÉDRIC CHAPPELIER, MARTIN RAJMAN

LIA – I&C – EPFL (Artificial Intelligence Laboratory
of the Swiss Federal Institute of Technology)
Lausanne, Switzerland
E-mail: {Antoine.Rozenknop, Jean-Cedric.Chappelier, Martin.Raijman}@epfl.ch

ABSTRACT. Standard stochastic grammars use generative probabilistic models, focusssing on rewriting probabilities conditioned by the symbol to be rewritten. Such grammars therefore tend to give penalty to longer derivations of the same input, which could be a drawback when they are used for analysis (e.g. parsing, language modelling for speech recognition). In this contribution, we propose a novel non-generative probabilistic model of TSGs, where probabilities are conditioned by the leaves (i.e. the input symbols) rather than by the root of the parse tree. Several experiments with this new model are presented.

11.1 Motivations

Standard stochastic grammars use generative probabilistic models, in which rule probabilities are conditioned by the root symbols of the rules. For instance, the probabilities in Stochastic Context-Free Grammars (SCFGs) actually represent the probabilities of the right-hand side of each rule knowing the left-hand side (i.e. the root) of the rule. The purpose of this contribution is to address this question in the framework of Stochastic Tree Substitution Grammars (STSGs). STSGs, mainly used in Data-Oriented Parsing (DOP) (Bod 1998), are grammars in which the rules consist of syntactic trees, called "*elementary trees*". These elementary trees are combined[1] with the substitution operator to give derivations of complete parse trees. In addition, a probability $p(\tau)$ is assigned to each elementary tree τ of the grammar. In the standard model this probability represents the probability

[1] leftmost first, as in CFG derivations.

Proceedings of Formal Grammar 2003
G. Jäger, P. Monachesi, G. Penn & S. Wintner (editors).

of using the elementary tree to rewrite its root in some (leftmost) derivation[2]. Although SCFGs and STSGs are equivalent from a structural point of view[3], STSGs are clearly different at the probabilistic level. Indeed, the availability of elementary trees of arbitrary depth enables STSGs to capture a much larger set of probabilistic dependencies than SCFGs, for which the probabilities are restricted to context-free (CF) rules (equivalent to depth-1 elementary trees).

When used in a speech recognition task or a syntactic parsing task, both STSGs and SCFGs present several drawbacks: the standard Expectation Maximization parameter estimation procedure used for SCFGs lead to models that assign biased probability estimations to syntactic trees (Johnson 1998) while DOP STSGs tend to overestimate the probabilities of deep syntactic trees (this behaviour is related to the lack of justification of their training procedure (Bonnema et al. 1999)).

The goal of this contribution is to present Gibbsian Tree Substitutions Grammars (GTSGs), a log-linear discriminant variation of STSGs, that does not suffer from the above mentioned drawbacks. Our approach is quite similar to the one that Johnson et al. (1999) expose for Unification-Based Grammars (UBG), also based on Gibbsian probability models and Maximum Conditional Likelihood Training. The two main differences with UBGs are (1) that we provide a different smoothing method (IIS), and (2) that both training and parsing can be achieved in polynomial time with a certain kind of TSGs (namely polynomial TSG), while for UBGs only training can be achieved in polynomial time under certain assumptions (Geman and Johnson 2002).

This paper first provides a formal definition of GTSGs. It next presents the key aspects of learning the parameters from a corpus. Finally, it concludes by giving the results of several experiments comparing GTSGs to standard STSGs.

11.2 The GTSG Model

The starting point is to consider a probabilization of TSGs which does not focus on the probability of a tree conditionally to its root symbol, but rather on what is the actual matter of parsing: the probability of a parse knowing the input sentence.

In order to do so, we focus on a Gibbs-Markov approach (Lafferty 1996) which allows to choose the desired features to be introduced in the model. In the analysis case that is considered here, the model is a set of conditional probabilities of parses knowing the input sentences, and the features are the elementary trees appearing in the derivations of those parses.

This means we choose *a priori* the form of the conditional probabilities of a derivation d knowing the input sentence w as a *Gibbsian distribution*. let us denote by $D(w)$ the set of derivations leading to w, $\mathcal{R}(\mathcal{G})$ the set of elementary trees of grammar $\mathcal{G}$, $f_\tau(d)$ the number of occurrences of elementary tree τ in derivation d, λ_τ the parameter associated with elementary tree τ in grammar

[2]The sum of the probabilities of elementary trees that have the same root node is therefore 1.

[3]The same language is recognized and the same parse trees are produced for a given sentence.

$\mathcal{G}$, $\lambda = (\lambda_{\tau_1}, \ldots, \lambda_{\tau_n})$ the vector of these parameters for all elementary trees in $\mathcal{R}(\mathcal{G})$, $f(d) = (f_{\tau_1}(d), \ldots, f_{\tau_n}(d))$ the vector of f_τ for all elementary trees τ in $\mathcal{R}(\mathcal{G})$, $Q(d) = \lambda \cdot f(d) = \sum_i \lambda_{\tau_i} f_{\tau_i}(d)$ the *potential* of derivation d and $Z_\lambda(w) = \sum_{d \in D(w)} e^{Q(d)}$ a partition function. We then pose:

$$p(d|w) = \frac{e^{Q(d)}}{\sum_{d' \in D(w)} e^{Q(d')}} = \frac{1}{Z_\lambda(w)} e^{\lambda \cdot f(d)}. \tag{11.1}$$

Furthermore, as in standard STSGs, the conditional probability of a parse tree t knowing the sentence w is defined as the sum of the conditional probabilities of all its derivations[4]:

$$p_\lambda(t|w) = \sum_{d \in D(t)} p_\lambda(d|w) \tag{11.2}$$

A TSG with such a probabilization model will be called a *Gibbsian Tree Substitution Grammar*.

Notice that the main difference between GTSGs and STSGs lies in their probabilization model and the associated learning algorithms, not in their structural aspect: this allows us to keep for GTSGs the same parsing algorithms as for STSGs for finding the most probable parse (MPP) tree knowing the input sentence to be analyzed.

A Toy Example

The following example illustrates in a simple way the differences between STSGs and GTSGs. Consider a TSG containing the elementary trees given in figure 11.1.[5]

The STSG model associates the probabilities $p_1, \ldots, p_5$ to the elementary trees, whereas the GTSG model uses unconstrained real values $\lambda_1, \ldots, \lambda_5$, called *potentials* and integrated in the probabilistic model through equation (11.1).

Suppose now that we want to use this grammar for parsing the sentence "*watch the man with a hat*". This sentence accepts the two parse trees (A) and (B) given in figure 11.2.

With the STSG, the probability of the derivations are:[6]

$$\begin{aligned} p_{STSG}(d_1(A)) &= p_1 \cdot p_3 \cdot p_4^2 \cdot p_5 \\ p_{STSG}(d_1(B)) &= p_1 \cdot p_4^2 \cdot p_5 \end{aligned}$$

In the GTSG, we first compute the potential of each derivation, which is by definition the sum of the potentials of its elementary trees:

$$\begin{aligned} Q(d_1(A)) &= \lambda_1 + \lambda_3 + 2\lambda_4 + \lambda_5 \\ Q(d_1(B)) &= \lambda_1 + 2\lambda_4 + \lambda_5 \end{aligned}$$

[4] With STSGs, a given parse tree can indeed have several different leftmost derivations.

[5] As these trees are of depth 1, this grammar can also be considered as a CFG; but this is not the case in general. This simple example was chosen for the sake of illustration only.

[6] Note that with this simple example, each parse tree has only one derivation. But this is not the case in general. With a more complex grammar, we should take every other derivation into account as well.

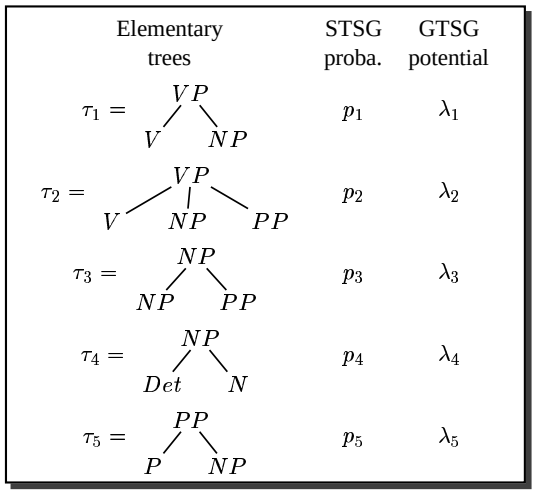

Figure 11.1: STSG/GTSG example

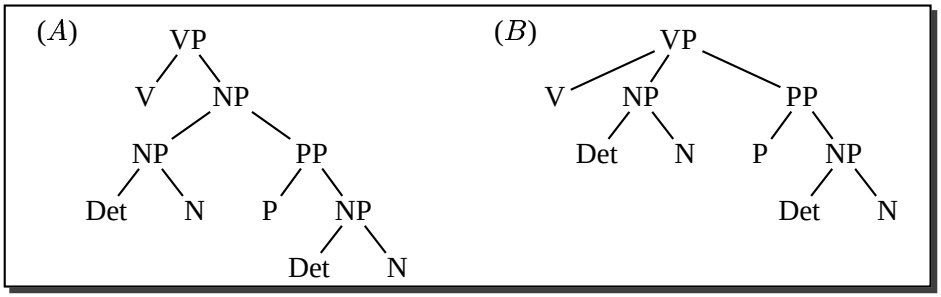

Figure 11.2: Two parse trees: (A) corresponds for instance to a sentence like "*watch this man with a hat*", while (B) corresponds to "*watch this man with your glasses*".

The probability of derivation $d_1(A)$ *conditionally to the leaves* is then, as stated in equation (11.1):

$$p_{GTSG}(d_1(A)|w) = \frac{e^{(Q(d_1(A)))}}{e^{Q(d_1(A))} + e^{Q(d_1(B))}}$$

Let us now use our simple example to illustrate some limitations of STSGs for parsing. Consider for instance the case where no further information on the language is available. In such a case,we would like the grammar to assign the same probability to each parse tree. With an STSG, the most "impartial" assignment of elementary probabilities seems to be the one corresponding to $p_1 = p_2 = p_3 = p_4 = \frac{1}{2}$; $p_5 = 1$: all elementary trees with the same root are assigned the same probability, and the sum of their probabilities is 1.

But in fact, the obtained grammar is anything but genuinely impartial: due to

the stochastic constraints and to the fact that the grammar only contains depth-1 elementary trees, some trees are favored by construction, namely the "smallest" ones (i.e. the ones that can be produced with the minimal number of derivation steps). For instance, with the above mentioned assignments, the STSG probability of tree (A) and (B) are respectively $(\frac{1}{2})^4$, and $(\frac{1}{2})^3$: (B) is therefore twice as probable as (A) with this "impartial" STSG!

With the GTSG however, an impartial assignment could consist in $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0$. With these values, the conditional probabilities of $d_1(A)$ and $d_1(B)$ relative to the sentence will be $1/2$, which appears to be effectively much fairer.

This example only gives a feeling of the differences between standard STSGs and GTSGs, without trying to demonstrate the superiority of the latter. However, obvious drawbacks of STSGs, pointed out by Johnson (1998) and Bonnema et al. (1999), can indeed be avoided with the discriminative training method for GTSGs described in the next section (as demonstrated by Rozenknop (2003)).

11.3 Learning parameters from a corpus

In order to learn the parameters of our model from a treebank corpus $\mathcal{C}$ (i.e. made of parse trees), we use a Maximum-Likelihood (ML) approach (Dempster et al. 1977).

We are therefore searching for the parameters λ^* of the GTSG which maximize the conditional log-likelihood of the learning corpus $L_{\tilde{p}}(p_\lambda)$, where $\tilde{p}(w,t)$ represents the relative frequencies in $\mathcal{C}$ of a tree t with leaves w:

$$\lambda^* = \underset{\lambda}{\mathrm{Argmax}} \sum_{w,t} \tilde{p}(w,t) \log p_\lambda(t|w),$$

In simpler terms, we search for the parameters that maximize the probability of the trees present in the corpus to be produced by the model, *knowing the sentences present in the corpus*.

11.3.1 Improved Iterative Scaling algorithm

In order to solve this non-trivial problem[7], we applied Lafferty's generalization of Improved Iterative Scaling algorithm (IIS) (Lafferty 1996) to STSG (where the hidden variables are the derivations).

Instead of maximizing $L_{\tilde{p}}(p_\lambda)$ directly, this approach iteratively improves the model λ, starting from an initial model λ_0.

[7]This problem is indeed non trivial since we are learning from a tree-bank in which only the parse trees of the sentences are given, and the derivations are not explicited. Part of the information is therefore lacking: the derivations leading to the observed parses.

In our case, this method leads, for every elementary tree $\widehat{\tau}$, to looking for the solution $\hat{x}$ of the polynomial equation (in x)

$$\begin{aligned}\sum_{w,t} \tilde{p}(w,t) \sum_{d \in D(w)} p_\lambda(d|w) f_{\widehat{\tau}}(d) x^{f^{\#}(d)} \\ = \sum_{w,t} \tilde{p}(w,t) \sum_{d \in D(t)} p_\lambda(d|t) f_{\widehat{\tau}}(d) \end{aligned} \tag{11.3}$$

In each step of the iteration, the new model is then obtained by replacing $\lambda_{\widehat{\tau}}$ with $\lambda_{\widehat{\tau}} + \log \hat{x}$.

Since all its coefficients are non-negative, equation (11.3) can easily be solved using Newton method. However, the key aspect is to be able to compute these coefficients in a "realistic" time, i.e. quickly enough for the method to be applicable in practice. This is precisely the aim of the following section.

11.3.2 Inside-Outside Algorithm

Left-hand side The left-hand side term of re-estimation formula (11.3) relies on a double sum: one sum over all samples in the learning set, and another sum over all possible derivations of each given sentence.

The computation of the latter sum has to be factorized in order to be tractable[8]. Indeed, the most demanding term of the left-hand side of (11.3) can be written as:

$$\begin{aligned}\sum_{d \in D(w)} p_\lambda(d|w) f_{\widehat{\tau}}(d) x^{f^{\#}(d)} &= \frac{1}{Z_\lambda(w)} \sum_{d \in D(w)} e^{\sum_{\tau \in \mathcal{R}(\mathcal{G})} \lambda_\tau f_\tau(d)} f_{\widehat{\tau}}(d) x^{f^{\#}(d)} \\ &= \frac{1}{Z_\lambda(w)} \sum_{d \in D(w)} f_{\widehat{\tau}}(d) x^{f^{\#}(d)} \prod_{\tau \in d} \left(e^{\lambda_\tau}\right)^{f_\tau(d)} \\ &= \frac{1}{Z_\lambda(w)} \sum_{d \in D(w)} f_{\widehat{\tau}}(d) \prod_{\tau \in d} \left(e^{\lambda_\tau} x\right)^{f_\tau(d)} \\ &= \frac{1}{Z_\lambda(w)} \sum_{d \in D(w)} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v(\tau)^{f_\tau(d)}\end{aligned}$$

with $v(\tau) = e^{\lambda_\tau} x$.

We are then able to compute $\sum_{d \in D(w)} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v(\tau)^{f_\tau(d)}$ using the *Inside-Outside algorithm* in the semi-ring of polynomials $\mathbb{P}[v(\tau)]$ (Goodman 1998).

Furthermore, we have by definition:

$$Z_\lambda(w) = \sum_{d \in D(w)} \prod_{\tau \in d} \left(e^{\lambda_\tau}\right)^{f_\tau(d)}$$

[8] the number of derivations of a given parse tree can indeed become very huge, in general exponential with the number of internal nodes of the parse tree; and moreover the number of different parse trees of one given sentence can also be very huge

So, $Z_\lambda(w)$ appears to be the sum of the coefficients associated with the $\widehat{\tau}$ such that $f_{\widehat{\tau}}(d) = 1$, which have already been computed in the chart used for the Inside-Outside algorithm. Therefore, $Z_\lambda(w)$ can be computed directly, without an extra Inside-Outside pass.

Right-hand side Similarly, we can write the right-hand side part of (11.3) as

$$\sum_{d \in D(t)} p_\lambda(d|t) f_{\widehat{\tau}}(d) \quad = \quad \frac{1}{Z_\lambda(t)} \sum_{d \in D(t)} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v'(\tau)^{f_\tau(d)}$$

with $v'(\tau) = e^{\lambda_\tau}$.

Once again, $\sum_{d \in D(t)} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v'(\tau)^{f_\tau(d)}$ can be computed by means of the Inside-Outside algorithm, provided that the chart used for factorizing the computation is "cleaned up" at each iteration, in such a way that it contains only (but all) derivations of parse tree t and not other derivations of other possible parse trees of the input sentence w.

11.3.3 Increasing Depth Learning (IDL)

The ML learning presented so far suffers from a major drawback, similar to the one pointed out by Bonnema and Scha (2002) for DOP: in the case where the set of elementary trees $\mathcal{R}(\mathcal{G})$ of the grammar contains the complete parse trees of the training corpus, the model over-estimates the parameters in such a way that the probability of any parse tree not present in the corpus becomes arbitrarily small.

One way to avoid this behavior is to decide that not all trees of the corpus can be elementary trees, in particular to avoid the full parse trees to be considered as one single elementary tree.

Another possibility is to slightly change the learning criterion. Instead of maximizing the ML of the corpus with the whole grammar, we can maximize the ML of the corpus iteratively while introducing new elementary trees in the grammar. In our case, we choose to consider the growing set of elementary trees with increasing depth (first all depth-1 trees are considered, then all depth-1 and depth-2, and so on).

More precisely, we consider the sequence of TSGs $(\mathcal{G}_p)_{p=1...pmax}$: $\mathcal{G}_p$ is the grammar associated with the set $\mathcal{R}_p$ of elementary trees with maximal depth p and with the set $\lambda_{\mathcal{R}_p}$ of corresponding parameters.

The Increasing Depth Learning (IDL) consists in computing the parameters $\lambda_{\mathcal{R}_1}$ of $\mathcal{G}_1$ (using the previously detailed algorithms), and, for increasing values of p, to learn the parameters $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ of $\mathcal{G}_p$,[9] keeping $\lambda_{\mathcal{R}_{p-1}}$ constant.

The parameters $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ are determined by maximizing the conditional likelihood $L_{\tilde{p}}(p_{\lambda_{\mathcal{R}_p}})$ of the learning set according to model $\mathcal{G}_p$, $\lambda_{\mathcal{R}_{p-1}}$ being known.

[9] $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ represents the parameters in $\lambda_{\mathcal{R}_p}$ which are not in $\lambda_{\mathcal{R}_{p-1}}$

In practice, IDL requires only small adaptations of the former algorithms. Only initialization and update steps are changed; all the computations remain the sames:

- Initialization of step p:
  - parameters $\lambda_{\mathcal{R}_{pmax}} \backslash \lambda_{\mathcal{R}_p}$ are set to $-\infty$;
  - parameters $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ are set to 0;
  - parameters $\lambda_{\mathcal{R}_{p-1}}$ keep the values obtained during the previous step.
- Update: Only the parameters $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ are updated.

Then, IIS is repeated $pmax$ times, with $pmax$ the maximal depth of trees in the learning set. This makes the learning algorithm quite long[10]. However, the deeper the trees the faster the convergence of parameters: learning can therefore be optimized by adapting the number of steps in IIS to the depth of the trees.

11.4 Experiments

The above exposed training procedure relies on the principle of maximizing the probability of the corpus *trees* conditionally to their leaves. To be coherent with this principle, syntactic analysis must also rely on the probability of the trees conditionally to the sentence to be parsed. Therefore, the chosen criterion for selecting one parse among all possible parses of a given sentence is the most probable parse (MPP) criterion.

11.4.1 Polynomial STSGs

STSGs in general suffer from a major drawback: finding the most probable parse (MPP) when parsing with a STSG has been proved to be an NP-hard problem in the most general case (Sima'an 1996). Various approximated MPP search algorithms have already been developed (Bod 1992; Goodman 1996; Chappelier and Rajman 2000). Another alternative, *Polynomial Tree Substitution Grammars* (pSTSG), first introduced by Chappelier and Rajman (2001), consists in choosing a set of elementary trees of the STSG in such a way that finding the MPP is no longer NP-hard but polynomial. This is the framework we will focus on in our experiments.

Min-Max pSTSG Min-Max STSGs are obtained by extracting from the training corpus two types of subtrees as elementary trees for the grammar: minimal (i.e. depth-1) subtrees, corresponding to context-free syntactic rules, and maximal subtrees, which correspond to all subtrees of the corpus whose leaves all are terminal symbols (i.e. words). Min-Max STSGs have been proved to be polynomial STSGs (Chappelier and Rajman 2001).

[10] one or two days of computation on a Sun Sparc Ultra 10, for 3000 trees in the learning corpus, $pmax = 16$, and doing 200 IIS passes per depth.

Head-Driven pSTSG Head-Driven pSTSGs (Chappelier et al. 2002) are obtained in a similar way as Min-Max pSTSGs: all depth-1 subtrees of the corpus are first extracted; then the other elementary trees are all subtrees of the corpus in which lexical heads (as defined by Collins (1999) and only heads are extended at every node.

11.4.2 Experimental protocol for evaluation

Bod's version of the ATIS corpus was used for this evaluation. This corpus consists of 750 syntactic trees from which the lexical leaves have been removed, keeping only part-of-speech tags. The extracted Min-Max pSTSG contains 2434 elementary trees, 381 of them being depth-1 trees and the other 2053 maximal trees. The extracted Head-Driven pSTSG only contains 930 elementary trees, with 381 depth-1 trees and 549 lexical-head trees.

The maximum depth of the trees was 11. Thus, the maximum depth of the elementary trees in the Min-Max models was 11, but only 5 for the Head-Driven TSGs.

Two kinds of experiments were conducted. In the "test 10%" experiment, we train the models on 90% of the corpus, and test it on the remaining 10%. The reported results are average values computed on 10 "90%-10%" random splits. In the "self-test" experiment, training and testing take place on the same (whole) corpus. The second type of experiments only serves as a baseline for the evaluation of the maximum improvement we can expect with Gibbsian syntactic models.

In each experiment, the GTSG model is compared with an STSG obtained with the standard training method, where elementary trees are parameterized by probabilities proportional to their frequency in the corpus.

For Head-Driven GTSG models, two kinds of training methods were tested on the "test 10%" experiments: the former one used the IDL procedure previously described, and the latter did not. Indeed, the IDL can be avoided for Head-Driven GTSG models, since in such models the complete trees of the corpus do not belong to the set of elementary trees of the grammar. Notice however that this procedure is not useful for "self-test" experiments, where the generalization ability of the grammar is not of interest.

For evaluation, we used the standard PARSEVAL measures (Manning and Schütze 1999). They include: (1) "*Coverage*", the number of sentences that have at least one (good or bad) analysis, (2) "*Good*", the number of sentences correctly parsed (with regard to the reference); the rate is the ratio between this number and the "coverage", (3) "*Cross*", the number of sentences receiving an incompatible bracketing; i.e. when the list of words spanned by some syntactic group in a tree intersects the list of words spanned by some syntactic group in the reference tree, and one list is not included in the other; the "cross" rate is the ratio between this number and the coverage, (4) "*Precision*" and (5) "*Recall*" rates, obtained by considering the sequence of all the trees $\tilde{\tau}$ produced by the analyzer as a set $E(\tilde{\tau})$ of triplets $< N, f, l >$, where N is a syntactic label attached to a node in a tree, and f

	Head-Driven models					Min-Max models				
	Cov.	Good	CrossP	P	R	Cov.	Good	CrossP	P	R
Self-Test										
STSG	750	412	57			750	664	0		
Rate (in %)		54.9	7.6	96.9	95.2		88.5	0.0	99.7	99.2
GTSG	750	479	43			750	691	0		
Rate (in %)		63.8	5.7	97.6	97.0		92.1	0.0	99.6	99.4
Gain (%)		**8.9**	–1.9	0.7	1.8		**3.6**		–0.1	0.2
Test 10%						**Test 10%**				
STSG	73.9	30.4	10.4			73.9	35.6	11.6		
Rate (in %)		41.1	14.0	93.7	93.6		48.2	15.7	93.6	94.4
GTSG-ML	73.9	35.7	10.0							
Rate (in %)		48.3	13.5	94.2	95.1					
Gain (%)		**7.2**	0.5	0.5	1.5					
GTSG-IDL	73.9	35.2	10.2			73.9	36.4	9.7		
Rate (in %)		47.6	13.8	94.2	94.8		49.3	13.1	93.5	95.6
Gain (%)		**6.5**	0.2	0.5	1.2		**1.1**	–2.6	–0.1	1.2

Table 11.1: Parsing performance: Head-Driven GTSG compared with Head-Driven STSG, Min-Max GTSG compared with Min-Max STSG.

and l are the indexes of the first and last words spanned by N; in comparison with a sequence of reference trees $\tilde{\tau}'$, precision and recall rates are calculated as:

$$P(\tilde{\tau}) = \frac{|E(\tilde{\tau}) \cap E(\tilde{\tau}')|}{|E(\tilde{\tau})|},\ R(\tilde{\tau}) = \frac{|E(\tilde{\tau}) \cap E(\tilde{\tau}')|}{|E(\tilde{\tau}')|}.$$

11.4.3 Results

Results are summarized in table 11.1 for Min-Max and Head-Driven pSTSGs.

For each model, the training process took approximately 4 hours of computation on a Sun Blade 60, on the above mentioned tree-bank. For the IDL procedure, 20 iterations of the IIS algorithm were performed for each elementary tree depth. Without IDL, 200 IIS iterations were performed for the whole set of elementary trees.

11.4.4 Discussion

- GTSGs reach the maximal score[11] in self-test for Min-Max grammars. This is an illustration that the theoretical skews pointed out by Bonnema et al. (1999) affect STSGs and not GTSGs. The suppression of this drawback has

[11] It can seem astonishing that the maximum score is $92,1\%$, and not 100%. This comes from the fact that the "sentences" of the corpus are PoS sequences: some sequences appear several times, with different analyses in the reference corpus. The grammar being deterministic, it assigns the same analysis for each occurrence of a "sentence".

almost no effect on the results in the "test-10%" experiments. Indeed, this effect is spread over various other error factors that occur in this situation, for instance the great variability of the corpus used.

- The advantage of Head-Driven GTSGs over Head-Driven STSGs is much more obvious. The "Good parse" error-rate is reduced by approximately 12% on the "test-10%" experiments. Cross bracketing, label recall and label precision rates also prove to be better with GTSGs. As theoretically expected, the IDL procedure is useless with Head-Driven GTSGs: the models obtained with standard ML (no IDL) lead to better results.

11.5 Conclusion

To conclude, the non-generative training developed in this presentation seems to be all the more interesting as grammars count less parameters: Head-Driven grammars benefit more from it than Min-Max grammars.

Results obtained when the training corpus and the test corpus are the same show that non-generative STSGs do not suffer the theoretical "skews" noted for standard stochastic grammars. This advantage is due to the difference between a training paradigm conditioned by the root and a use in parsing conditioned by the leaves of the trees. The suppression of these skews has less influence in the case of distinct training and test corpus, especially for Min-Max models, where several other effects also take place. Still, in this configuration, our Head-Driven GTSG performs better than the Min-Max STSG with much less parameters.

This type of tests does not show the advantage one could expect from the other characteristic of GTSGs parameterization, namely the absence of normalization: in models based on stochastic processes, this normalization is necessary due to the fact that the parameters of these models are probabilities. It triggers a considerable amount of problems when one tries to mix these models with others, as commonly done in speech recognition. It would then be very interesting to be able to set up non-generative models such as those exposed here in this context.

Bibliography

Bod, R. (1992). Applying Monte Carlo techniques to Data Oriented Parsing. In *Proceedings Computational Linguistics in the Netherlands*. Tilburg (The Netherlands).

Bod, R. (1998). *Beyond Grammar, An Experience-Based Theory of Language*. Number 88 in CSLI Lecture Notes. CSLI Publications, Standford (CA).

Bonnema, R., P. Buying, and R. Scha (1999). A new probability model for Data Oriented Parsing. In P.Dekker and G.Kerdiles, eds., *Proceedings of the 12th Amsterdam Colloquium*. Institute for Logic, Language and Computation, Amsterdam.

Bonnema, R. and R. Scha (2002). Reconsidering the probability model of Data-Oriented Parsing. In R. Bod, R. Scha, and K. Sima'an, eds., *Data-Oriented Parsing*, chapter I.3, pp. 25–41. CSLI Publications.

Chappelier, J.-C. and M. Rajman (2000). Monte-Carlo sampling for NP-hard maximization problems in the framework of weighted parsing. In D. Christodoulakis, ed., *Natural Language Processing – NLP 2000*, number 1835 in Lecture Notes in Artificial Intelligence, pp. 106–117. Springer.

Chappelier, J.-C. and M. Rajman (2001). Polynominal Tree-Substitution Grammars: an efficient framework for Data-Oriented Parsing. In *Proc. of Recent Advances in Natural Language Processing (RANLP 2001)*.

Chappelier, J.-C., M. Rajman, and A. Rozenknop (2002). Polynomial Tree Substitution Grammars: Characterization and new examples. In *Proc. of 7th conference on Formal Grammar*, pp. 29–39. Trento, Italy.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Dempster, M. M., N. M. Laird, and D. B. Jain (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society*, **39**:1–38.

Geman, S. and M. Johnson (2002). Dynamic programming for parsing and estimation of Stochastic Unification-Based Grammars. In *ACL 2002*, pp. 279–286.

Goodman, J. T. (1996). Efficient algorithms for parsing the DOP model. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pp. 143–152.

Goodman, J. T. (1998). *Parsing Inside-Out*. Ph.D. thesis, Harvard University, Cambridge, Massachusetts.

Johnson, M. (1998). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, **24**(4):613–632.

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler (1999). Estimators for Stochastic "Unification-Based" Grammars. In M. Kaufmann, ed., *Proceedings of the 37th Annual Meeting of ACL*, pp. 538–541. San Francisco.

Lafferty, J. D. (1996). Gibbs-Markov models. In *Computing Science and Statistics*, volume 27, pp. 370–377.

Manning, C. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge.

Rozenknop, A. (2003). *Modèles syntaxiques probabilistes non-génératifs*. Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne.

Sima'an, K. (1996). Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of COLING'96*, volume 2, pp. 1175–1180. Copenhagen (Denmark). Cmp-lg/9606019.