Smoothing Techniques for Tree-*k*-Grammar-Based Natural Language Modeling

Jose L. Verdú-Mas, Jorge Calera-Rubio, and Rafael C. Carrasco[⋆]

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

Abstract. In a previous work, a new probabilistic context-free grammar (PCFG) model for natural language parsing derived from a tree bank corpus has been introduced. The model estimates the probabilities according to a generalized k-grammar scheme for trees. It allows for faster parsing, decreases considerably the perplexity of the test samples and tends to give more structured and refined parses. However, it suffers from the problem of incomplete coverage. In this paper, we compare several smoothing techniques such as *backing-off* or *interpolation* that are used to avoid assigning zero probability to any sentence.

1 Introduction

Some previous works ([1], [2], [3]) have explored the performance of parsers based on a probabilistic context-free grammar (PCFG) extracted from a training corpus. The results show that the type of tree representation used in the corpus can have a substantial effect in the estimated likelihood of each sentence or parse tree. According to the conclusions weaker independence assumptions —such as decreasing the number of nodes or increasing the number of node labels— improve the efficiency of the parser. The best results were obtained with offspring annotated labels where each node stores contextual information in the form of the category of the node's parent or the node's descendents. This is in agreement with the observation put forward by Charniak [4] 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, a generalization of the classic k-gram models, widely used for string processing [5], was introduced to the case of trees [3]. The PCFG 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

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

is, child-annotated models, and we have compared their parsing performance to that of unannotated PCFG –or $k = 2$, in our notation– and of parent-annotated PCFG [2]. The experiments showed that:

– The parsing performance of unannotated model is worse than any annotated model.
– The parsing performance of parent-annotated and child-annotated PCFG are 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.

On the other hand, the smaller ambiguity of child-annotated model leads to unparsable sentences and, then, smoothing is essential in the construction of an efficient tree-k-grammar language model. A language model is a probability distribution over strings $P(s)$ that describes the frequency with which each string s occurs as a sentence in natural text [6].

In this work, we carry out a comparasion of three smoothing techniques. Two of them are well known: linear interpolation and tree-level back-off. In addition, we introduce a new smoothing technique: rule-level back-off. While being relatively simple to implement, we show that all these methods yield good performances with tree-k-grammar language models applied to structural, syntactical or lexical disambiguation.

The experiments were performed using the Wall Street Journal (WSJ) corpus of the University of Pennsylvania [7] modified as described in [4] and [2].

2 The Tree-*k*-Grammar Model

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

1. the probability $P(s_1 s_2 \ldots s_t|M)$ of a sequence in the model M is computed as a product $p_M(s_1)p_M(s_2|s_1)\cdots p_M(s_t|s_1 s_2 \ldots s_{t-1})$, 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: $p_M(s_t|s_1 \ldots s_{t-1}) = p_M(s_t|s_{t-k+1} \ldots s_{t-1})$.

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

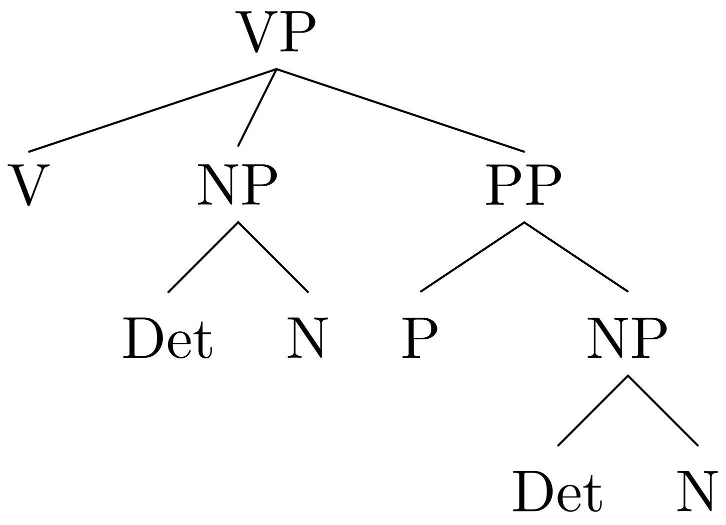

Fig. 1. A sample parse tree of depth 3

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 [8] 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 generates i.e., every *state* stores a subtree of depth $k-2$ ([3]). 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 in treebank grammars. For instance, for the tree depicted in Fig. 1, the following rules are obtained:

$$\begin{aligned}\mathrm{VP} &\to \mathrm{V\ NP\ PP}\\ \mathrm{NP} &\to \mathrm{Det\ N}\\ \mathrm{PP} &\to \mathrm{P\ NP}\end{aligned}$$

However, in the case $k=3$, 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:

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

where each state has the form $\mathrm{X}_{\mathrm{Z}_1,\ldots,\mathrm{Z}_\mathrm{m}}$. This is equivalent to performing a relabelling of the parse tree before extracting the rules.

It is obvious that the $k = 3$ model 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 $k = 3$.

3 Smoothing

Statistical approaches to efficient parsing offer the advantage of making the most likely decision on the basis of available parsed text corpora.

Although the $k = 3$ model yields a good performance (in terms of both parsing and speed), their rules are very specific and, then, some events (subtrees, in our case) in the test set are not present in the training data, yielding zero probabilities. Due to data sparseness, this happens often in reality. However, this is not the case of the $k = 2$ model, with total coverage but with worse performance. This justifies the need for smoothing methods.

In the following, three smoothing techniques are described. Two of them are well known: linear interpolation and tree-level back-off. In addition, we introduce a new smoothing technique: rule-level back-off.

3.1 Linear Interpolation

Smoothing through *linear interpolation* [9] is performed by computing the probability of events as a weighted average of the probabilities given by different models. For instance, the smoothed probability of a $k = 3$ model could be computed as a weighted average of the probability given by the model itself, and that given by the $k = 2$ model, that is,

$$p(t) = \lambda p_3(t) + (1 - \lambda) p_2(t) \tag{1}$$

The mixing parameter $\lambda \in [0, 1]$ was chosen to minimize the perplexity of a sample.

3.2 Tree-Level Back-Off

Back-off allows one to combine information from different models. In our case, the highest order model such that the probability of the event is greater than zero is selected. Some care has to be taken in order to preserve normalization.

$$p(t) = \begin{cases} (1-\lambda) p_3(t) \text{if} p_3(t) > 0 \\ \Lambda p_2(t) \text{if} p_3(t) = 0 \end{cases} \tag{2}$$

where

$$\Lambda = \frac{\lambda}{\sum_{t:p_3(t)=0} p_2(t)}. \tag{3}$$

In our experiments, we will assume that a λ may be found such that no sentence s in the test set having a tree with $p_3(t) > 0$ has another tree t' with $p(t') > p(t)$. Therefore, p_2's will only be compared for trees with $p_3(t) = 0$. This leads to the following efficient parsing strategy: $k = 2$ (unannotated, slow) parsing is not launched if the $k = 3$ (annotated, fast) parser returns a tree, because the $k = 3$ tree will win out all $k = 2$ trees; therefore, for parsing purposes, the actual value of λ is irrelevant.

3.3 Rule-Level Back-Off

Our back-off model builds a new PCFG from the rules of the tree-k-grammar models and adding new rules which allow to switch beetween those models. In particular, the new PCFG consists of three different kinds of rules:

1. $k = 3$ rules with modified probability,
2. back-off rules that allow to switch to the lower model, and,
3. modified $k = 2$ rules to switch-back to the higher model.

This is done as follows (for the sake of simplicity, only a kind of binary rules are shown):

1. Add the rules of the $k = 3$ model with probability:

$$p(X_{Y,Z} \to \alpha) = p_3(X_{Y,Z} \to \alpha)(1 - \lambda(X_{Y,Z})) \tag{4}$$

2. For each non-terminal symbol, $X_{Y,Z}$, of the $k = 3$ model, add a *back-off rule* $X_{Y,Z} \to Y\ Z$ with probability:

$$p(X_{Y,Z} \to Y\ Z) = \frac{\lambda(X_{Y,Z})}{\Lambda(X_{Y,Z})} \tag{5}$$

where

$$\Lambda(X_{Y,Z}) = 1 - \sum_{X_{Y,Z} \to \alpha_Y \alpha_Z \in \{k=3\}} p_2(Y \to \alpha_Y) p_2(Z \to \alpha_Z) \tag{6}$$

3. Add the $k = 2$ rules as unary rules, that is, if the rule is $X \to Y\ Z$, then, add $X \to X_{Y,Z}$ with probability:

$$p(X \to X_{Y,Z}) = p_2(X \to Y\ Z) \tag{7}$$

The grammar is normalized provided that parses of the form $X_{Y,Z} \to Y\ Z \to \alpha_Y \alpha_Z$ are assigned a zero probability if $X_{Y,Z} \to \alpha_Y\ \alpha_Z$ exists in the grammar.

4 Experimental Results

4.1 General Conditions

We have performed experiments to assess the structural disambiguation performance of tree-k-grammar smoothed models as compared to the ones unsmoothed, that is, to compare their relative ability for selecting the best parse tree. 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 [4]) 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 no more than 40 words.

4.2 Structural Disambiguation Results

All grammar models were written as standard context-free grammars, and Earley's probabilistic extended parsing algorithm [10] was used to obtain, for each sentence, the most likely parse that was compared to the corresponding tree in the test set using the customary PARSEVAL evaluation metric [11, 12, 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 these two 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

As baseline, three non smoothed models 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.

Table 1. Parsing results with different annotation models: labelled recall R, labelled precision P, fraction of exact matches, fraction of sentences parsed, and average time per sentence in seconds

Model	R	P	exact	parsed	t
k=2	70.7%	76.1%	10.0%	100%	57
k=3	79.6%	74.3%	13.4%	94.6%	7
Parent	80.0%	81.9%	16.3%	100%	340

Table 2. Parsing results with different smoothed models

Model	R	P	exact	parsed	t
M1	80.2%	78.6%	17.4%	100%	57
M2	78.9%	74.2%	17.1%	100%	9.3
M3	82.4%	81.3%	17.5%	100%	68

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 unparsable, that is, they cannot be generated by the grammar. The results in table 1, that were analyzed in detail in [3], show that the parsing performance of parent-annotated and child-annotated PCFG is similar but parsers using child-annotated grammars are much faster because the number of possible parse trees considered is drastically reduced.

Those smoothed models were evaluated:

- A linear interpolated model, M1, as described in section 3.1 with $\lambda = 0.7$ (the value of λ selected to minimize the perplexity).
- A tree-level back-off, M2, as described in section 3.2.
- A rule-level back-off, M3, as described in section 3.3. This model has 92,830 $k = 3$ rules, 15,140 $k = 2$ rules and 10,250 back-off rules. A fixed parameter λ (0.005) was selected to maximize labelled recall and precision).

The results in table 2 show that:

- M2 is the fastest but its performance is worse than that of M1 and M3.
- M1 and M3 parse sentences at a comparable speed but recall and precision are better using M3.

Compared to un-smoothed models, smoothed ones:

- Cover the whole test set ($k = 3$ did not).
- Parsed at reasonable speed (compared to Parent).
- Achieved acceptable performance ($k = 2$ did not).

5 Conclusions

We have compared several smoothing techniques for tree-k-grammar-based natural language modeling and parsing that are used to avoid assigning zero probability to any sentence. In particular, we have introduced a new smoothing technique: a rule-level back-off that builds a new PCFG from the rules of the tree-k-grammar models and adding new rules which allow to switch beetween those models. The new grammar cover the whole test set and improve the performance in terms of parsing.

References

[1] E. Charniak and G. Carroll. Context-sensitive statistics for improved grammatical language models. In *Proceedings of the 12th National Conference on Artificial Inteligence, AAAI Press*, pages 742–747, Seattle, WA, 1994. 1057

[2] Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998. 1057, 1058

[3] Jose L. Verdu-Mas, Mikel L. Forcada, Rafael C. Carrasco, and Jorge Calera-Rubio. Tree k-grammar models for natural language modelling and parsing. In Terry Caelli, Adnan Amin, Rober P. W.Duin, Mohamed Kamel, and Dick de Ridder, editors, *Proceedings of the Joint IAPR International Workshops on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 56–63, Windsor-Canada, 2002. Springer. 1057, 1059, 1063

[4] E. Charniak. Treebank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1031–1036. AAAI Press/MIT Press, 1996. 1057, 1058, 1062

[5] 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. 1057

[6] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco, 1996. Morgan Kaufmann Publishers. 1058

[7] 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. 1058

[8] Maurice Nivat and Andreas Podelski. Minimal ascending and descending tree automata. *SIAM Journal on Computing*, 26(1):39–58, 1997. 1059

[9] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. A tree-based statistical language model for natural language speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 507–514. Kaufmann, San Mateo, CA, 1990. 1060

[10] Andreas Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. In *Computational Linguistics, MIT Press for the Association for Computational Linguistics*, volume 21. 1995. 1062

[11] 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 syntatic coverage of english grammars. In *Proc. Speech and Natural Language Workshop 1991*, pages 306–311, San Mateo, CA, 1991. Morgan Kauffmann. 1062

[12] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 1062