PARSING WITH PCFGS

JOAKIM NIVRE

Contents

1. Grammar Formalism 1
2. Parsing Model 3
3. Parsing Algorithms 4
4. Learning with a Treebank 4
5. Learning without a Treebank 5
References 5

The PCFG model is without doubt the most important formal model in syntactic parsing today, not only because it is widely used in itself but also because many later developments start from it. In this lecture, I will first introduce the basic formalism (§1) and the parsing model that naturally follows from it (§2). I will then give an overview of standard techniques for parsing (§3), for supervised learning from a treebank (§4), and for weakly supervised learning using expectation-maximization (§5). In the next lecture, we will look at some of the more advanced techniques developed to improve parsing performance with PCFGs.

1. Grammar Formalism

A *probabilistic context-free grammar* (PCFG) is a simple extension of a context-free grammar in which every production rule is associated with a probability (Booth & Thompson, 1973). Formally, a PCFG is a quintuple $G = (N, \Sigma, R, S, Q)$, where N is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols, R is a finite set of production rules of the form $A \rightarrow \alpha$ (where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$), $S \in N$ is the start symbol, and $Q : R \rightarrow [0, 1]$ is a function that assigns a probability to each member of R. Figure 1 shows a PCFG capable of generating the sentence in Figure 2 with its associated parse tree. Although the actual probabilities assigned to the different rules are completely unrealistic because of the very limited coverage of the grammar, it nevertheless serves to illustrate the basic form of a PCFG.

As usual, we use $L(G)$ to denote the string language generated by G, that is, the set of strings over the terminal alphabet Σ for which there exists a derivation $S \Rightarrow^* x$ using rules in R. In addition, we use $T(G)$ to denote the tree language generated by G, that is, the set of parse trees corresponding to valid derivations of strings in $L(G)$. Given a parse tree $y \in T(G)$, we use $\textsc{yield}(y)$ for the terminal string associated with y, $\textsc{count}(i, y)$ for the number of times that the ith production rule $r_i \in R$ is used in the derivation of y, $\textsc{lhs}(i)$ for the nonterminal symbol in the left-hand side of r_i, and q_i for the probability of rule r_i (that is, $q_i = Q(r_i)$).

Date: 2013-02-01.

S	→	NP VP .	1.00	JJ	→	Economic	0.33
VP	→	VP PP	0.33	JJ	→	little	0.33
VP	→	VBD NP	0.67	JJ	→	financial	0.33
NP	→	NP PP	0.14	NN	→	news	0.5
NP	→	JJ NN	0.57	NN	→	effect	0.5
NP	→	JJ NNS	0.29	NNS	→	markets	1.0
PP	→	IN NP	1.0	VBD	→	had	1.0
PU	→	.	1.0	IN	→	on	1.0

FIGURE 1. Probabilistic context-free grammar for a fragment of English.

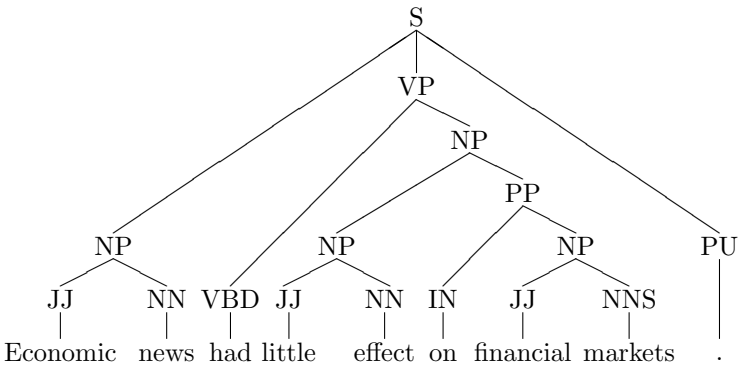

FIGURE 2. Constituent structure for an English sentence taken from the Penn Treebank (Marcus et al., 1993, 1994).

The probability of a parse tree $y \in T(G)$ is defined as the product of probabilities of all rule applications in the derivation of y:

$$P(y) = \prod_{i=1}^{|R|} q_i^{\text{COUNT}(i,y)} \tag{1}$$

This follows from basic probability theory on the assumption that the application of a rule in the derivation of a tree is independent of all other rule applications in that tree, a rather drastic independence assumption that we will come back to. Since the yield of a parse tree uniquely determines the string associated with the tree, the joint probability of a tree $y \in T(G)$ and a string $x \in L(G)$ is either 0 or equal to the probability of y, depending on whether or not the string matches the yield:

$$P(x, y) = \begin{cases} P(y) & \text{if } \text{YIELD}(y) = x \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

It follows that the probability of a string can be obtained by summing up the probabilities of all parse trees compatible with the string:

$$P(x) = \sum_{y \in T(G): \text{YIELD}(y) = x} P(y) \tag{3}$$

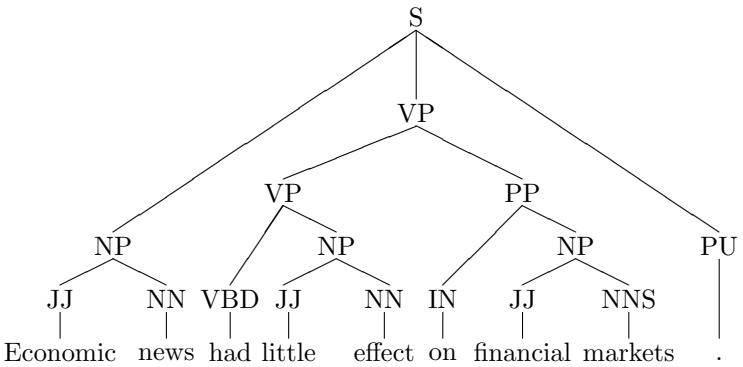

FIGURE 3. Alternative constituent structure for an English sentence taken from the Penn Treebank (cf. figure 2).

A PCFG is *proper* if Q defines a proper probability distribution over every subset of rules that have the same left-hand side $A \in N$:

$$\sum_{r \in R : \textsc{lhs}(r) = A} Q(r) = 1 \tag{4}$$

A PCFG is *consistent* if it defines a proper probability distribution over the set of trees that it generates:

$$\sum_{y \in T(G)} P(y) = 1 \tag{5}$$

Consistency can also be defined in terms of the probability distribution over *strings* generated by the grammar. Given Equation 3, the two notions are equivalent.

2. Parsing Model

PCFGs have many applications in natural language processing, for example, in language modeling for speech recognition or statistical machine translation, where they can be used to model the probability distribution of a string language. In this course, however, we are primarily interested in their use as parsing models, which can be conceptualized as follows:

(1) The input space is the set of all strings over Σ; $\mathcal{X} = \Sigma^*$.
(2) The output space is the set of all parse trees over R; $\mathcal{Y} = R^*$.[1]
(3) The generative component is the context-free grammar defining a set of parse trees for each input; $\text{GEN}(x) = \{y \in T(G) \mid \textsc{yield}(y) = x\}$.
(4) The evaluative component is the function Q defining a probability distribution over the parse trees for an input; $\text{EVAL}(y) = P(y) = \prod_{i=1}^{|R|} q_i^{\textsc{count}(i,y)}$.

For example, even the minimal PCFG in Figure 1 generates two trees for the sentence in Figure 2, the second of which is shown in Figure 3. According to the grammar, the probability of the parse tree in Figure 2 is 0.0000794, while the probability of the parse tree in Figure 3 is 0.0001871. In other words, using this PCFG for disambiguation, we would prefer the second analysis, which attaches the PP *on financial markets* to the verb *had*, rather than to the noun *effect*. According to the gold standard annotation in the Penn Treebank, this would not be the correct choice.

[1]Parse trees can be identified with sequences of production rules given some canonical form of derivation, such as leftmost derivations.

Note that the score $P(y)$ is equal to the joint probability $P(x, y)$ of the input sentence and the output tree. For ranking in a parsing model, it may seem more natural to use the conditional probability $P(y|x)$ instead, since the sentence x is given as input to the model. The conditional probability can be derived as shown in Equation 6, but since the probability $P(x)$ is a constant normalizing factor, this will never change the internal ranking of analyses in $\text{GEN}(x)$.

$$P(y|x) = \frac{P(x, y)}{\sum_{y' \in \text{GEN(x)}} P(y')} \tag{6}$$

3. Parsing Algorithms

The *parsing* or *decoding* problem for a PCFG model is to compute, given a specific grammar G and an input sentence x, the set $\text{GEN}(x)$ of candidate representations and to score each candidate by the probability $P(y)$, as defined by the grammar. The first part is simply the parsing problem for CFGs, and many of the standard algorithms CFG parsing have a straightforward extension that computes the probabilities of parse trees in the same process. This is true, for example, of the CKY algorithm (Ney, 1991), Earley's algorithm (Stolcke, 1995), and the algorithm for bilexical context-free grammars described in Eisner & Satta (1999).[2]

These algorithms are all based on dynamic programming, which makes it possible to compute the probability of a substructure at the time when it is being composed of smaller substructures and use Viterbi search to find the highest scoring analysis in $O(n^3 \cdot |G|)$ time, where n is the length of the input sentence x and $|G|$ is the size of the grammar G. However, this also means that, although the model as such defines a complete ranking over all the candidate analyses in $\text{GEN}(x)$, these parsing algorithm only computes the single best analysis. In other words, the algorithms solve the following optimization problem:

$$y^* = \underset{y \in \text{GEN}(x)}{\text{argmax}}\ \text{EVAL}(y)$$

Nevertheless, the inference is *exact* in the sense that the analysis returned by the parser is guaranteed to be the most probable analysis according to the model. There are generalizations of this scheme that instead extract the k best analyses, for some constant k, with varying effects on time complexity.

4. Learning with a Treebank

The *learning* problem for PCFGs is to learn a grammar from a sample of sentences $X = \{x^1, \ldots, x^m\}$. If the sentences are annotated with their correct parse trees, the simplest method is to extract a so-called *treebank grammar* (Charniak, 1996), where the context-free grammar contains all and only the symbols and rules needed to generate the trees in the training set $Y = \{y^1, \ldots, y^m\}$, and where the probability of each rule is estimated by its relative frequency among rules with the same left-hand side:

$$Q(r_i) = \frac{\sum_{j=1}^{m} \text{COUNT}(i, y^j)}{\sum_{j=1}^{m} \sum_{r_k \in R:\text{LHS}(r_k)=\text{LHS}(r_i)} \text{COUNT}(k, y^j)} \tag{7}$$

This is an example of *supervised learning*, since we require the inputs in the training corpus to be labeled with their correct outputs.

To give a simple example, the grammar in Figure 1 is in fact a treebank grammar for the treebank consisting of the two trees in Figure 2 and Figure 3. The grammar contains exactly the rules needed to generate the two trees, and rule probabilities are estimated by the frequency of each rule relative to all the rules for the same nonterminal. Treebank grammars have a number

[2]For a description of the CKY algorithm for PCFGs, see Collins (2013b) or Jurafsky & Martin (2009).

of appealing properties. First of all, relative frequency estimation is a special case of maximum likelihood estimation (MLE), which is a well understood and widely used method in statistics. Secondly, treebank grammars are guaranteed to be both proper and consistent. Finally, both learning and decoding is simple and efficient. However, although early investigations reported encouraging results for treebank grammars, especially in combination with other statistical models (Charniak, 1996, 1997), empirical research has clearly shown that they do not yield the most accurate parsing models, for reasons that we will return to in the next lecture.

5. Learning without a Treebank

If our training corpus is not a treebank, but the context-free grammar is given, then the standard method is to use expectation-maximization (EM) to learn the rule probabilities. EM is an approximate method for maximum likelihood estimation, but instead of maximizing the joint likelihood of inputs and outputs, as in a treebank grammar, it attempts to maximize the marginal likelihood of inputs (by summing over all outputs for a given input). The basic structure of the EM algorithm (as applied to PCFG learning) is as follows:

(1) Guess a probability q_i for each rule $r_i \in R$.
(2) Repeat until convergence:
   (a) E-step: Compute the expected count $f(r_i)$ of each rule $r_i \in R$:

$$f(r_i) = \sum_{j=1}^{m} \sum_{y \in \text{GEN}(x^j)} P(y \mid x^j, Q) \cdot \text{COUNT}(i, y)$$

   (b) M-step: Reestimate the probability q_i of each rule r_i to maximize the marginal likelihood given expected counts:

$$q_i = \frac{f(r_i)}{\sum_{r_j \in R:\text{LHS}(r_j)=\text{LHS}(r_i)} f(r_j)}$$

The tricky part is to compute the expected rule counts in an efficient way, which can be done using the inside-outside algorithm, another dynamic programming algorithm.[3] EM training with the inside-outside algorithm was used in early work on PCFG parsing to estimate the probabilistic parameters of hand-crafted context-free grammars from raw text corpora (Fujisaki et al., 1989; Pereira & Schabes, 1992). Like treebank grammars, PCFGs induced in this way are guaranteed to be proper and consistent.

References

J. Baker (1979). 'Trainable Grammars for Speech Recognition'. In *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.

D. M. Bikel (2004). 'Intricacies of Collins' Parsing Model'. *Computational Linguistics* **30**:479–511.

T. L. Booth & R. A. Thompson (1973). 'Applying Probability Measures to Abstract Languages'. *IEEE Transactions on Computers* **C-22**:442–450.

E. Charniak (1996). 'Tree-Bank Grammars'. In *Proceedings of AAAI/IAAI*, pp. 1031–1036.

E. Charniak (1997). 'Statistical Parsing with a Context-Free Grammar and Word Statistics'. In *Proceedings of AAAI/IAAI*, pp. 598–603.

E. Charniak & M. Johnson (2005). 'Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking'. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 173–180.

[3]For a description of the inside-outside algorithm, see Collins (2013a), who also discusses its use in EM for PCFGs.

Z. Chi & S. Geman (1998). 'Estimation of Probabilistic Context-Free Grammars'. *Computational Linguistics* **24**:299–305.

M. Collins (2000). 'Discriminative Reranking for Natural Language Parsing'. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 175–182.

M. Collins (2013a). 'The Inside-Outside Algorithm'. Lecture Notes.

M. Collins (2013b). 'Probabilistic Context-Free Grammars (PCFGs)'. Lecture Notes.

J. Eisner & G. Satta (1999). 'Efficent Parsing for Bilexical Context-Free Grammars and Head Automaton Grammars'. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 457–464.

T. Fujisaki, et al. (1989). 'A Probabilistic Method for Sentence Disambiguation'. In *Proceedings of the 1st International Workshop on Parsing Technologies*, pp. 105–114.

L. Huang & D. Chiang (2005). 'Better k-Best Parsing'. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

V. M. Jiménez & A. Marzal (2000). 'Computation of the n Best Parse Trees for Weighted and Stochastic Context-Free Grammars'. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*.

D. Jurafsky & J. H. Martin (2009). *Speech and Language Processing*. Pearson Education, second edition edn.

M. P. Marcus, et al. (1993). 'Building a Large Annotated Corpus of English: The Penn Treebank'. *Computational Linguistics* **19**:313–330.

M. P. Marcus, et al. (1994). 'The Penn Treebank: Annotating Predicate-Argument Structure'. In *Proceedings of the ARPA Human Language Technology Workshop*, pp. 114–119.

H. Ney (1991). 'Dynamic Programming Parsing for Context-Free Grammars in Continuous Speech Recognition'. *IEEE Transactions on Signal Processing* **39**:336–340.

F. C. Pereira & Y. Schabes (1992). 'Inside-Outside Reestimation from Partially Bracketed Corpora'. pp. 128–135.

J. A. Sánchez & J. M. Benedí (1997). 'Consistency of Stochastic Context-Free Grammars from Probabilistic Estimation Based on Growth Transformations'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**:1052–1055.

A. Stolcke (1995). 'An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities'. *Computational Linguistics* **21**:165–202.