A generalized CYK algorithm for parsing stochastic CFG(*)

J.-C. Chappelier and M. Rajman
EPFL – DI-LIA, Écublens, CH-1015 Lausanne, Suisse

Abstract

We present a bottom-up parsing algorithm for stochastic context-free grammars that is able (1) to deal with multiple interpretations of sentences containing compound words; (2) to extract N-most probable parses in $\mathcal{O}(n^3)$ and compute at the same time all possible parses of any portion of the input sequence with their probabilities; (3) to deal with "out of vocabulary" words. Explicitly extracting all the parse trees associated to a given input sentence depends on the complexity of the grammar, but even in the case where this number is exponential in n, the chart used by the algorithm for the representation is of $\mathcal{O}(n^2)$ space complexity.

1 Introduction

This article presents CYK+, a bottom-up parsing algorithm for stochastic context-free grammars that is able:

1. to deal multiple interpretations of sentences containing compound words;
2. to extract N-most probable parses in $\mathcal{O}(n^3)$ and compute at the same time all possible parses of any portion of the input sequence with their probabilities;
3. to deal with "out of vocabulary" words.

Explicitly extracting all the parse trees associated with a given input sentence or to any substring of the input sentence is furthermore possible. The cost of this step depends on the complexity of the grammar, but even in the case where the number of parse trees is exponential in n, the chart used by the CYK+ algorithm for their representation remains of $\mathcal{O}(n^2)$ space complexity.

2 The CYK+ algorithm

The algorithm we present, called hereafter CYK+, uses a CYK table as a chart which contains generalized Earley-like items. It is therefore quite similar to the bottom-up Earley algorithm [5, 13, 4]. More precisely, in terms of parsing schemata [10], it can be seen as a reduction of it to a specific class of grammars (less restrictive than the CNF grammars usually considered for CYK).

The grammar class accepted by CYK+ is a subclass of CFG, denoted hereafter nplCFG and consisting of "non partially lexicalized rules"[1]. This means that terminals can only appear in rules of type $X \rightarrow w_1 ... w_n$ where X is a non terminal and w_i are terminals. In practice such lexical rules are even not written as such in the grammar but are part of the lexicon[2]. The restriction to the nplCFG class is however not "critical" for the algorithm. It was introduced because it allows to restrict the processing of "lexicalized rules" to the initialization step only. The algorithm could nevertheless be easily be extended to deal with any CFG.

Our algorithm can be seen as derived either from a bottom-up Earley (with generalized items and without prediction) [4, 13]; or from a CYK algorithm (but performing dynamic binarization of the grammar). It could also be seen as an extension of the algorithm presented in [5]. A CYK-like point of view will be used hereafter to describe the CYK+ algorithm.

The basic data structure used by the algorithm is a triangular matrix[3] with $n\,(n+1)/2$ cells, where n is the size of the input string $w_1 ... w_n$ to parse. Each cell $T_{i,j}$ of the chart contains two lists of items:

(*)This research was funded by "slp-Infoware", France.

[1] and without ε rules

[2] most often with $n = 1$. $n > 1$ occurs for compound words or lexicalized expressions.

[3] We call this matrix a "chart".

- items of the first list (called the "type-1 list") are the non-terminals that parse the substring $w_j...w_{j+i-1}$. More formally, if A is such a non-terminal: $A \Rightarrow^* w_j...w_{j+i-1}$
- items of the second list (called the "type-2 list") represent partial parses of the substring $w_j...w_{j+i-1}$, that is strings α of non-terminal symbols such that $\alpha \Rightarrow^* w_j...w_{j+i-1}$ and for which there is a rule in the grammar of the form $A \rightarrow \alpha\beta$ with β a non-empty string[4] of non-terminal symbols. Such items will be denoted by $\alpha \bullet ...$ and represent $\bigcup_{A \in \mathcal{NT}} \bigcup_{\substack{\beta \\ A \rightarrow \alpha\beta \in \mathcal{R}}} \left\{ A \rightarrow \alpha \bullet \beta \right\}$ where $\mathcal{NT}$ is the set of non-terminals of the grammar and $\mathcal{R}$ the set of rules. Notice that these items represent a generalization of doted rules used in Earley-like parsers, since only the beginning (i.e. parsed) part of the rule is represented, independently both of the left-hand side and of the end of the rule. This provides a much more compact representation of dotted rules.

Each element of any of these two lists is itself a list of all the possible productions of the corresponding item. For extraction purposes, each element of those sublists contains an explicit reference to the items that were used for its production.

The **initialization step** of the algorithm consists in building the type-1 lists for all the cells of the chart for which there exists a lexical rule for the associated word or sequence of words. More precisely, if the rule $X \rightarrow w_j...w_{j+n}$ is in the grammar, the item X will be added to the type-1 list of cell $T_{n+1,j}$. This $\mathcal{O}(n^2)$ initialization step allows the algorithm to parse all the interpretations of sequences of words potentially corresponding to compound forms[5]. The initialization step is then completed by a self-filling step that will be explained below.

The **parsing step** of the algorithm consists in applying, row-wise to all the cells in the table, the two procedures described hereafter for a given cell (i, j) of the chart:

- first, a "standard" cell filling procedure in which all the combinations of two cell (k, j) and $(i - k, j + k)$ that could produce a new (eventually partial) interpretation are explored. More precisely a combination of a type-2 item $\alpha \bullet ...$ of cell (k, j) with a type-1 item B of cell $(i - k, j + k)$ is realized if and only if there is a rule of the form $A \rightarrow \alpha B \gamma$ in the grammar. If γ is empty, A is added to the type-1 list of cell (i, j) otherwise $\alpha B \bullet ...$ is added to the type-2 list of cell (i, j).
- second, a "self-filling" procedure in which, for each item B of the type-1 list, if there exists a rule of the form $A \rightarrow B\gamma$, then if γ is empty A is added to the type-1 list of the cell otherwise $B \bullet ...$ is added to the type-2 list. This second step is necessary both to deal with rules of the form $A \rightarrow B$ and to maintain the type-2 lists up to date. Notice that due to the fact that an item is never put twice in a list, cycles in the grammar do not represent problems for the non-probabilistic bottom-up phase. However they require specific processing both for correct probability calculation and for recursive enumeration of the syntactic trees stored in the chart.

The input string is syntactically correct with respect to the grammar if and only if the top-level non-terminal S is produced in the cell $T_{n,1}$. The CYK+ algorithm, like all standard CYK algorithms, not only determines whether a string is syntactically correct or not, but also produces a compact representation ($\mathcal{O}(n^2)$ space-complexity) of all parses for all the substrings of the input string.

Furthermore, it is particularly easy to extract out of the table all the possible parse trees associated with a sentence. The extraction procedure simply starts from the top cell type-1 list of items and follows the associated links stored in the sublists during the parsing phase. The looping problem that may occur in the case of cycles in the grammar (the extraction process may enter an infinite loop) is solved by forbidding to the algorithm to follow the same link twice. This corresponds to conventionally restrict to the shortest syntactic tree in case of cycles.

In term of parsing schemata, and using the same notations as in [10], our algorithm can be represented as the parsing system $\langle \mathcal{I}, H, D \rangle$ extended to any grammar in nplCFG, where :

[4] it is the role of the type-1 list cover the cases where β is empty.

[5] This initialization has even been generalized in a speech processing framework to the parsing at the same time of the multiple sequences coming from the same acoustic support.

$\mathcal{I}$ is the domain

$$\{[A \to \alpha \bullet \beta, i, j] \mid A \to \alpha\beta \in P, i \geq 1, j \geq 1\}$$

H is the following hypotheses set :[6]

$$\{[w, i, j] \mid w = w_j ... w_{j+i-1}, 1 \leq j \leq n, 1 \leq i \leq n - j + 1\}$$

D is the deduction set made of the union of

$$D^{\text{Init}} = \{[w, i, j] \vdash [A \to w\bullet, i, j] \mid A \to w \in P\}$$

and

$$D^{\text{Compl}} = \{[A \to \alpha \bullet B\beta, i, j][B \to \gamma\bullet, k, j + i] \vdash [A \to \alpha B \bullet \beta, i + k, j]\}$$

In can easily be seen that the bottom-up Earley parsing scheme is close to CYK+ without compound words treatment (i.e. $i = 1$ in the former hypotheses set). This is precisely the reason why we restrict to the nplCFG grammar class: it allows a reduction of the number of steps to be performed and do correspond the kind of grammars we are currently dealing with. The reduction is due to the fact that the treatment of non-terminal can be restricted to the initialization part. In addition, this initialization step has been extended (in a $\mathcal{O}(n^2)$ time-complexity) to the treatment of compound forms.

3 Dealing with probabilities

The purpose of this section is to present the extension of the CYK+ algorithm to the class of SCFG[7]. It is not in the scope of this paper to argue on the status of SCFGs which have been thoroughly investigated for many years [1, 2, 3, 6, 8, 9, 12]. We will use the SCFG paradigm for itself, as, even if it is not a perfect model of natural language, SCFGs are still superior to non-probabilistic CFGs [11].

The computation of the maximum probability for each of the parse trees is rather straightforward since in the SCFG context the probability of a non-terminal is the product of the conditional probability of the rule that produces it by the product of the probabilities of the right-hand side non-terminals of this rule. Since bottom-up approaches always construct right-hand side non-terminals first, and since the maximum of a product is obtained at the maxima of its constituents, the algorithm is able to compute the probability of a new non-terminal with few overhead during the parsing phase.

In addition to the fact that our algorithm is not restricted to CNF grammars, another of its advantages over the other known probabilistic variants of CYK [1, 7, 14] is that it is able to compute, during the parsing phase, the N most probable parses (hereafter called N-best parses). This computation relies on the fact that items are always created by combining only two previously created items (one from the type-2 list and one non-terminal from the type-1 list). It is therefore possible to compute, for each such production, the corresponding N-best probability values since, being a product of two members, the N biggest values are always included in the N^2 products of the N-best probabilities associated with each of the composing items. The algorithm therefore provides, at the end of the parsing phase[8], the N-best parses of the input sentence (as well as of each of its substrings).

In cases where the N-best approach is not sufficient, it is of course furthermore possible recursively to extract *all* the parses of the sentence (as explained in the previous section), their probabilities being calculated during extraction by the top-down product of the probabilities of their constituents.

The only remaining problem to be discussed here concerns the computation of probabilities in the case of cycles in the grammar. This is done as in [11] by pre-computing, once for all for a given grammar,

[6] Usual hypotheses restricts to $i = 1$. Hypotheses where $i > 1$ corresponds here to compound words.

[7] more precisely to SnplCFG which is the same reduction of SCFG as nplCFG is for CFG.

[8] i.e. without going into the recursive extraction of all the possible parses.

the matrix of the probabilities of the cycles of the grammar (refer to [11] for further details). This matrix is then used as a lookup table the algorithm refers to when detecting a cycle (the same item twice).

For practical NLP applications, the above algorithm has to be extended by the possibility of dealing with "out of vocabulary" words. In our approach, the treatment of unknown word is embedded into the stochastic framework.

It is part of the definition of SCFGs that for each given non-terminal (A) of the grammar the conditional probabilities for producing A have to verify the stochastic constraint, namely that:

$$\sum_{\alpha} \mathcal{P}(A \to \alpha) = 1$$

To implement a treatment for unknown words, we relax this constraint to

$$\sum_{\alpha} \mathcal{P}(A \to \alpha) \leq 1$$

and define a "open non-terminal" to be any non-terminal for which the above inequality is strict. For such non-terminals, it is therefore possible to introduce the quantity $p_{\underline{u}}(A) = 1 - \sum_{\alpha} \mathcal{P}(A \to \alpha)$ which can be interpreted as the probability that "the unknown word" (noted $\underline{u}$) might have been produced by the (forgotten) rule $A \to \underline{u}$. The unknown word is a new terminal (not already existing in the terminals of the grammar) to which is mapped every actually encountered unknown word[9]. Whenever an unknown word is encountered in the input sequence, all the rules $A \to \underline{u}$ where A is an "open non-terminal" are taken into account by the algorithm as any other "standard" termination rule.

4 Conclusion

In the present contribution, we have presented a variation of the CYK algorithm, called the CYK+ algorithm, which is able to parse a class of CF grammars larger than the CNF grammars usually accepted by standard CYK implementations. This class of grammars, called the nplCFG ("non partially lexicalized CF grammars"), allows easier grammar writing (less constrained than with CNF grammars) while preserving the algorithmic efficiency of the parsing algorithm.

Furthermore, this parsing algorithm deals with stochastic CF grammars and is able to produce the N-best probabilistic syntactic interpretations for any portion of the input sequence as well as dealing with unknown words.

Finally, CYK+ is able to take compound forms into account (with the integration of rules of the form $X \to w_1 w_2 ... w_n$). It turned out that this new functionality can also be used to adapt the CYK+ algorithm for the parallel processing of the word graphs produced by acoustic modules of speech recognizers; this opens promising perspectives for the use of the CYK+ algorithm for the integration of a syntactic module within a speech recognizer.

References

[1] J. K. Baker. Trainable grammars for speech recognition. In J. J. Wolf and D. H. Klatt, editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, 1979.

[2] T. L. Booth and R. A. Thompson. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May 1973.

[3] E. Charniak. *Statistical Language Learning*. MIT Press, 1993.

[4] G. Erbach. Bottom-up Earley deduction. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, 1994.

[9] This is done so in order to avoid tricky infinite summations of probabilities over actual unknown words.

[5] S. L. Graham, M. A. Harrison, and W. L. Ruzzo. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems*, 2(3):415–462, July 1980.

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

[7] B. Krenn and C. Samuelsson. The linguist's guide to statistics. available at `http://www.coli.uni-sb.de/~christer/stat_cl.ps`, 1997.

[8] J. Kupiec. A trellis-based algorithm for estimating the parameters of a hidden stochastic context-free grammar. In *Proceedings of the Speech and Natural Language Workshop*, pages 241–246. DARPA, 1991.

[9] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[10] K. Sikkel and A. Nijholt. Parsing of context-free languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, pages 61–100. Springer, 1997.

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

[12] P. Suppes. Probabilistic grammars for natural languages. *Synthese*, 22:95–116, 1970.

[13] F. Voisin and J. Raoult. A new, bottom-up, general parsing algorithm. In *Journées AFCET-GROPLAN, les Avancées en Programmation, Nice*, 1990.

[14] D. Wu. Stochastic inversion transduction grammars with application to segmentation, bracketing and aligment of parallel corpora. In *IJCAI'95*, volume 2, pages 1328–1335, Montréal (Canada), Aug. 1995. Morgan Kaufmann.