K-best Iterative Viterbi Parsing

Katsuhiko Hayashi and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{hayashi.katsuhiko, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper presents an efficient and optimal parsing algorithm for probabilistic context-free grammars (PCFGs). To achieve faster parsing, our proposal employs a pruning technique to reduce unnecessary edges in the search space. The key is to repetitively conduct Viterbi inside and outside parsing, while gradually expanding the search space to efficiently compute heuristic bounds used for pruning. This paper also shows how to extend this algorithm to extract K-best Viterbi trees. Our experimental results show that the proposed algorithm is faster than the standard CKY parsing algorithm. Moreover, its K-best version is much faster than the Lazy K-best algorithm when K is small.

1 Introduction

The CKY or Viterbi inside algorithm is a well-known algorithm for PCFG parsing (Jurafsky and Martin, 2000), which is a dynamic programming parser using a chart table to calculate the Viterbi tree. This algorithm is commonly used in natural language parsing, but when the size of the grammar is extremely large, exhaustive parsing becomes impractical. One way to reduce the computational cost of PCFG parsing is to prune the edges produced during parsing. In fact, modern parsers have often employed pruning techniques such as *beam search* (Ratnaparkhi, 1999) and *coarse-to-fine search* (Charniak et al., 2006).

Despite their practical success, both pruning methods are approximate, so the solution of the parser is not always optimal, i.e., the parser does not always output the Viterbi tree. Recently, another line of work has explored *A* search* algorithms, in which simpler problems are used to estimate heuristic scores for prioritizing edges to be processed during parsing (Klein and Manning, 2003). If the heuristic is *consistent*, A* parsing always outputs the Viterbi tree. As Tsuruoka and Tsujii (2004) mentioned, however, A* parsing has a serious difficulty from an implementation point of view: "One of the most efficient way to implement an agenda, which keeps edges to be processed in A* parsing, is to use a priority queue, which requires a computational cost of $O(\log(n))$ at each action, where n is the number of edges in the agenda. The cost of $O(\log(n))$ makes it difficult to build a fast parser by the A* algorithm."

This paper presents an alternative way of pruning unnecessary edges while keeping the optimality of the parser. We call this algorithm *iterative Viterbi parsing* (IVP) for the reason that the iterative process plays a central role in our proposal. The IVP algorithm conducts repetitively Viterbi inside and outside parsing, while gradually expanding the search space to efficiently compute lower and upper bounds used for pruning. IVP is easy to implement and is much faster in practice than the standard CKY parsing algorithm.

In addition, we also show how to extend the IVP algorithm to extract K-best Viterbi parse trees. The idea is to integrate Huang and Chiang (2005)'s K-best algorithm 3, which is called as *Lazy*, with the iterative parsing process. Lazy performs a Viterbi inside pass and then extracts K-best lists in a top-down manner. Although especially the first Viterbi inside pass is a bottleneck of the Lazy algorithm, the K-best IVP algorithm avoids its amount of work as well as in the 1-best case.

2 Iterative Viterbi Parsing

Following Pauls and Klein (2009), we define some notations. The IVP algorithm takes as input a

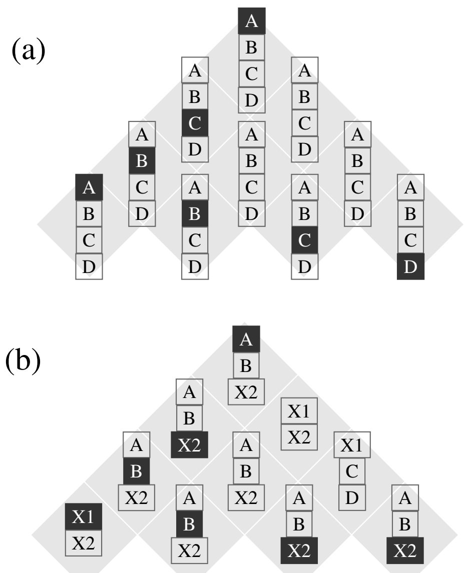
Figure 1: (a) An original chart table consisting of non-terminal symbols only. (b) A coarse chart table consisting of both non-terminal symbols and shrinkage symbols. There exists a corresponding derivation A(X2(B(X1 B) X2) X2) in (b) to a derivation A(C(B(A B) C) D) in (a), both consist of black-shaded symbols.

PCFG G and a sentence x consisting of terminal symbols $t_0 \dots t_{n-1}$. Without loss of generality, we assume Chomsky normal form: each non-terminal rule r in G has the form $r = \mathrm{A} \to \mathrm{B}\ \mathrm{C}$ with log probability weight $\log q(r)$, where A, B and C are elements in N, which is a set of non-terminal symbols. *Chart edges* are labeled spans $e = (\mathrm{A}, i, j)$. *Inside derivations* of an edge $e = (\mathrm{A}, i, j)$ are trees rooted at A and spanning $t_i \dots t_{j-1}$. The score of a derivation d is denoted by $s(d)$[1]. The score of the best (maximum) inside derivation for an edge e is called the *Viterbi inside score* $\beta(e)$. The goal of 1-best PCFG parsing is to compute the Viterbi inside score of the *goal edge* $(\mathrm{TOP}, 0, n)$ where TOP is a special root symbol. For the goal edge, we call its derivation *goal derivation*. The score of the best derivation of TOP $\to t_0 \dots t_{i-1}$ A $t_j \dots t_{n-1}$ is called the *Viterbi outside score* $\alpha(e)$.

We assume $N = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$. By grouping several symbols in the same cell of the chart table, we can make a smaller table than the original one. While the original chart table in Figure 1 (a) contains non-terminal symbols only, the chart table in Figure 1 (b) contains not only non-terminal symbols but also new symbols X1 and X2. The new symbols, which are made by grouping several non-terminal symbols, are refered to as *shrinkage symbols*. For example, the shrinkage symbols X1 and X2 consist of non-terminal symbols $\{\mathrm{A}, \mathrm{B}\}$ and $\{\mathrm{C}, \mathrm{D}\}$, respectively.

[1]The score of a derivation is the sum of rule weights for all rules used in the derivation.

Level	0	1	2
	Op_	ADJ_	JJ
			JJR
			JJS
		ADV_	RB
			RBR
			RBS
			WRB
		NOUN_	NN
			NNP
			NNPS
			NNS
		VERB_	MD
			VB
			VBD
			VBG
			VBN
			VBP
			VBZ
	Cl_		IN
			CC
			CD
		DET_	DT
			EX
			PDT
			WDT
		PRON_	PRP
			PRP$
			WP
			WP$
		PRT_	POS
			RP
			TO
	Ot_	X_	FW
			LS
			SYM
			UH
		PUNC_	#
			$
			”
			“
			-LRB-
			-RRB-
			,
			:
			.

Figure 2: The levels of non-terminal symbols.

In this paper, to make shrinkage symbols, we use hierarchical clustering of non-terminal symbols defined in (Charniak et al., 2006). Figure 2 shows a part of the hierarchical symbol definition. Formally, we hierarchically cluster N into $m + 1$ sets $N_0 \dots N_m$ where $N = N_m$. For some $i \in [0 \dots m - 1]$, we call an element in N_i *i-th layer shrinkage symbol*. For some $0 \le i \le j \le m$,

Algorithm 1 Iterative Viterbi Parsing

```
 1: $lb \leftarrow$ det$(x, G)$ or $lb \leftarrow -\infty$
 2: chart $\leftarrow$ init-chart$(x, G)$
 3: for all $i \in [1 \ldots]$ do
 4:     $\hat{d} \leftarrow$ Viterbi-inside(chart)
 5:     if $\hat{d}$ consists of non-terminals only then
 6:         return $\hat{d}$
 7:     if $lb <$ best(chart) then
 8:         $lb \leftarrow$ best(chart)
 9:     expand-chart(chart, $\hat{d}, G$)
10:     Viterbi-outside(chart)
11:     prune-chart(chart, $lb$)
```

we define a mapping $\pi_{i \to j} : N_i \mapsto \mathfrak{P}(N_j)$ where $\mathfrak{P}(\cdot)$ is the power set of $\cdot$. Taking a symbol HP in Figure 2 as an example, $\pi_{0 \to 1}(\text{HP}) = \{\text{S}_{-}, \text{N}_{-}\}$. When $i = j$, for some i-th layer shrinkage symbol A $\in N_i$, $\pi_{i \to j}(\text{A})$ returns a singleton $\{\text{A}\}$. For all $0 \leq i, j, k \leq m$, the rule parameter associated with symbols $\text{X}_i \in N_i$, $\text{X}_j \in N_j$, $\text{X}_k \in N_k$ is defined as the following:

$$\log q(\text{X}_i \to \text{X}_j \; \text{X}_k) = \max_{\substack{\text{A} \in \pi_{i \to m}(\text{X}_i) \\ \text{B} \in \pi_{j \to m}(\text{X}_j) \\ \text{C} \in \pi_{k \to m}(\text{X}_k)}} \log q(\text{A} \to \text{B C}).$$

By this construction, each derivation in a coarse chart gives an upper bound on its corresponding derivation in the original chart (Klein and Manning, 2003) and we can obtain the following lemma:

Lemma 1. *If the best goal derivation $\hat{d}$ in the coarse chart does not include any shrinkage symbol, it is equivalent to the best goal derivation in the original chart.*

Proof . Let $\mathcal{Y}$ be the set of all goal derivations in the original chart, $\mathcal{Y}' \subset \mathcal{Y}$ be the subset of $\mathcal{Y}$ not appearing in the coarse chart, and $\mathcal{Y}''$ be the set of all goal derivations in the coarse chart. For each derivation $d \in \mathcal{Y}'$, there exists its unique corresponding derivation d' in $\mathcal{Y}''$ (see Figure 1). Then, we have

$$\forall d \in \mathcal{Y}, \exists d' \in \mathcal{Y}'', \; s(d) \leq s(d') < s(\hat{d})$$

and this means that $\hat{d}$ is the best derivation in the original chart. □

Algorithm 1 shows the pseudo code for IVP. The IVP algorithm starts by initializing coarse chart, which consists of only 0-th layer shrinkage symbols. It conducts Viterbi inside parsing to find the best goal derivation. If the derivation does not contain any shrinkage symbols, the algorithm returns it and terminates. Otherwise, the chart table is expanded, and the above procedure is repeated until the termination condition is satisfied.

For efficient parsing, we integrate a pruning technique with IVP. For an edge $e = (\text{A}, i, j)$, we denote by $\alpha\beta(e) = \alpha(e) + \beta(e)$ the score of the best goal derivation which passes through e, where $\beta(e)$ and $\alpha(e)$ are Viterbi inside and outside scores for e. Then, if we obtain a lower bound lb such that $lb \leq \max_{d \in \mathcal{Y}} s(d)$ where $\mathcal{Y}$ is the set of all goal derivations in the original chart, an edge e with $\alpha\beta(e) < lb$ is no longer necessary to be processed. Though it is expensive to compute $\alpha\beta(e)$ in the original chart, we can efficiently compute by Viterbi inside-outside parsing its upper bound in a coarse chart table:

$$\alpha\beta(e) \quad \leq \quad \widehat{\alpha}(e) + \widehat{\beta}(e) = \widehat{\alpha\beta}(e)$$

where $\widehat{\alpha}(e)$ and $\widehat{\beta}(e)$ are the Viterbi inside and outside scores of e in the coarse chart table. If $\widehat{\alpha\beta}(e) < lb$, we can safely prune the edge e away from the coarse chart. Note that this pruning simply reduces the search space at each IVP iteration and does not affect the number of iterations taken until convergence at all.

We initialize the lower bound lb with the score of a goal derivation obtained by deterministic parsing det() in the original chart. The deterministic parsing keeps only one non-terminal symbol with the highest score per chart cell and removes the other non-terminal symbols. The det() function is very fast but causes many search errors. For efficient pruning, a tighter lower bound is important, thus we update the current lower bound with the score of the best derivation, having non-terminals only, obtained by the best() function in the current coarse chart, if the former is less than the latter.

At line 9, IVP expands the current chart table by replacing all shrinkage symbols in $\hat{d}$ with their next layer symbols using mapping π. While this expansion cannot derive a reasonable worst time complexity since it takes many iterations until convergence, we show from our experimental results that it is highly effective in practice.

3 K-best Extension

Algorihtm 2 shows the K-best IVP algorithm which applies the iterative process to the Lazy K-best algorithm of (Huang and Chiang, 2005). If the best derivation is found, which consists of non-terminal symbols only, this algorithm calls the

Algorithm 2 K-best IVP

```
1: $lb \leftarrow$ beam$(x, G, k)$ or $lb \leftarrow -\infty$
2: chart $\leftarrow$ init-chart$(x, G)$
3: **for all** $i \in [1 \ldots]$ **do**
4:     $\hat{d}_1 \leftarrow$ Viterbi-inside(chart)
5:     **if** $\hat{d}_1$ consists of non-terminals only **then**
6:         $[\hat{d}_2, \ldots, \hat{d}_k] \leftarrow$ Lazy K-best(chart)
7:         **if** All of $[\hat{d}_2, \ldots, \hat{d}_k]$ consist of non-terminals only **then**
8:             return $[\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_k]$
9:         **else**
10:            $\hat{d}_1$ = getShrinkageDeriv$([\hat{d}_2, \ldots, \hat{d}_k])$
11:    **if** $lb <$ k-best(chart, $k$) **then**
12:        $lb \leftarrow$ k-best(chart, $k$)
13:    expand-chart(chart, $\hat{d}_1, G$)
14:    Viterbi-outside(chart)
15:    prune-chart(chart, $lb$)
```

Lazy K-best algorithm. If all of the K-best derivations do not contain any shrinkage symbol, it returns them and terminates.

The K-best IVP algorithm also prunes unnecessary edges and initializes the lower bound lb with the score of the k-th best derivation obtained by beam search parsing in the original chart. For efficient pruning, we update lb with the k-th best derivation, which consists of non-terminals only, obtained by the k-best() function in the current coarse chart. The getShrinkageDeriv() function seeks the best derivation, which contains shrinkage symbols, from $[\hat{d}_2, \ldots, \hat{d}_k]$. The K-best IVP algorithm inherits the other components from standard IVP.

4 Experiments

We used the Wall Street Journal (WSJ) part of the English Penn Treebank: Sections 02–21 were used for training, sentences of length 1–35 in Section 22 for testing. We estimated a Chomsky normal form PCFG by maximum likelihood from right-branching binarized trees without function labels and trace-fillers. Note that while this grammar is a proof-of-concept, CKY on a larger grammar does not work well even for short sentences.

Table 1 shows that the number of edges produced by the IVP algorithm is significantly smaller than standard CKY. Moreover, many of the edges are pruned during the iterative process. While IVP takes many iterations util convergence, it is about 8 times faster than CKY. The fact means that the computational cost of the Viterbi inside and outside algorithms on a small chart is negligible.

Next, we examine the K-best IVP algorithm. Figure 3 shows parsing speed of Lazy and K-best IVP algorithms for various k ($2 \sim 128$). When k is small ($2 \sim 64$), K-best IVP is much faster than Lazy. However, K-best IVP did not work well when setting k to more than 128. We show the reason in Figure 4 where we plot the number of edges in chart table at each K-best IVP iterations for some test sentence with length 28. It is clear that the smaller k is, the earlier it is convergent. Moreover, when setting k too large, it is difficult to compute a tight lower bound, i.e., K-best IVP does not prune unnecessary edges efficiently. However, in practice, this is not likely to be a serious problem since many NLP tasks use only very small k-best parse trees (Choe and Charniak, 2016).

	CKY		IVP			
len.	edges	time	edges	pruned	iters	time
20	10590	1.25	2864	2089	68	0.13
23	13938	1.76	2219	1462	41	0.06
22	12771	1.52	2204	1425	46	0.05
17	7701	0.72	1526	1119	32	0.03
28	20538	3.14	7306	5338	144	1.18
34	30141	5.44	6390	4634	98	0.49
⋮	⋮			⋮		
21	12801	1.77	3502	2456	70	0.21

Table 1: The number of the edges produced in 1-best parsing on testing set. Many of the edges are pruned during the IVP parsing iterations. The last row denotes the mean values.

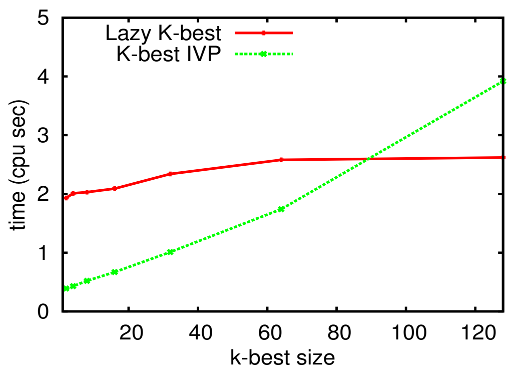

Figure 3: K-best Parsing time for various k.

5 Related Work

Huang and Chiang (2005) presented an efficient K-best parsing algorithm, which extracts K-best lists after a Viterbi inside pass. Huang (2005) also described a K-best extension of the Knuth parsing algorithm (Knuth, 1977; Klein and Manning, 2004). Pauls and Klein (2009) successfully integrated A* search with the K-best Knuth algorithm.

Tsuruoka and Tsujii (2004) proposed an itera-

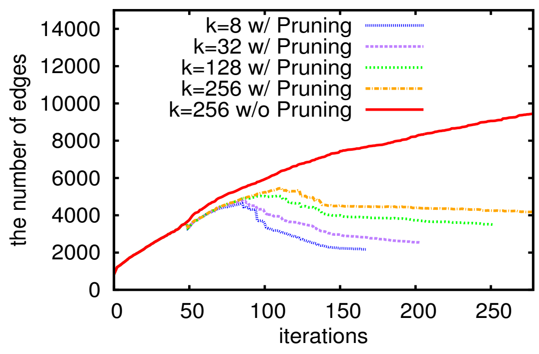

Figure 4: The plot of the number of edges in chart table at each K-best IVP parsing iteration.

tive CKY algorithm, which is similar to our IVP algorithm in that it conducts repeatedly CKY parsing with a threshold until the best parse is found. The main difference is that IVP employs a coarse-to-fine chart expansion to compute better lower and upper bounds efficiently. Moreover, Tsuruoka and Tsujii (2004) did not mention how to extend their algorithm to K-best parsing.

The coarse-to-fine parsing (Charniak et al., 2006) is used in many practical parsers such as Petrov and Klein (2007). However, the coarse-to-fine search is approximate, so the solution of the parser is not always optimal.

For sequential decoding, Kaji et al. (2010) also proposed the iterative Viterbi algorithm. Huang et al. (2012) extended it to extract K-best strings by integrating the backward K-best A* search (Soong and Huang, 1991) with the iterative process. Our proposed algorithm can be regarded as a generalization of their methods to the parsing problem.

6 Conclusion and Future Work

This paper presents an efficient K-best parsing algorithm for PCFGs. This is based on standard Viterbi inside-outside algorithms and is easy to implement. Now, we plan to conduct experiments using latent-variable PCFGs (Matsuzaki et al., 2005; Cohen et al., 2012) to prove that our method is useful for a variety of grammars.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was supported in part by JSPS KAKENHI Grant Number 26730126.

References

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 168–175, New York City, USA, June. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas, November. Association for Computational Linguistics.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable pcfgs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–231, Jeju Island, Korea, July. Association for Computational Linguistics.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Vancouver, British Columbia, October. Association for Computational Linguistics.

Zhiheng Huang, Yi Chang, Bo Long, Jean-Francois Crespo, Anlei Dong, Sathiya Keerthi, and Su-Lin Wu. 2012. Iterative viterbi a* algorithm for k-best sequential decoding. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 611–619, Jeju Island, Korea, July. Association for Computational Linguistics.

Liang Huang. 2005. K-best knuth algorithm. `http://cis.upenn.edu/~lhuang3/knuth.pdf`.

Daniel Jurafsky and James H Martin. 2000. *Speech and Language Processing*. Prentice Hall.

Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and Masaru Kitsuregawa. 2010. Efficient staggered decoding for sequence labeling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 485–494, Uppsala, Sweden, July. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 40–47, Edmonton, USA, May-June. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2004. Parsing and hypergraphs. In *New developments in parsing technology*, pages 351–372. Springer.

Donald E Knuth. 1977. A generalization of dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Adam Pauls and Dan Klein. 2009. K-best a* parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 958–966, Suntec, Singapore, August. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Frank K Soong and E-F Huang. 1991. A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 705–708, Toronto, Ontario, Canada, May. IEEE.

Yoshimasa Tsuruoka and Junichi Tsujii. 2004. Iterative cky parsing for probabilistic context-free grammars. In *International Conference on Natural Language Processing*, pages 52–60, Hyderabad, India, December. Springer.