Implementation of CKY Parsing

Wei Peng
University of Pittsburgh
wpeng1@andrew.cmu.edu

Abstract

Cocke-Kasami-Younger (CKY) parsing is one the earliest recognition and parsing algorithm. In this work, we implement CKY parsing with Penn Treebank PCFG and investigate how markovization (grammar refinement) affects the parsing performance measured by F1 score.

1 CKY Parsing

CKY parsing is well studied and developed in the literature by (Manning and Schütze, 1999) , (Charniak et al., 2006) and (Klein and Manning, 2003). As for parsing, the input is a sentence, and the output is a constituent tree where the internal nodes correspond to phrases (nonterminals), and nodes immediately above words are PoStag (preterminals). The basic CKY parsing supports only rules in the Chomsky Normal Form (CNF):

$$C \rightarrow w \tag{1}$$

$$C \rightarrow C_1\ C_2, \tag{2}$$

which was first introduced by (Chomsky, 1959). However, any Context Free Grammar (CFG) can be converted to an equivalent CNF and be converted reversely. We can take a grammar straight off a tree, using frequencies to estimate probabilities of $\mathrm{P(X \rightarrow Y, Z)}$, $\mathrm{P(X \rightarrow Y)}$ and $\mathrm{P(X \rightarrow w)}$. And then fill in Unary and Binary score charts in CKY:

$$\mathrm{S}_u(\mathrm{X,i,j}) = \max_{\mathrm{X \rightarrow Y}} \mathrm{P(X \rightarrow Y)S}_b(\mathrm{Y,i,j}) \tag{3}$$

$$\mathrm{S}_b(\mathrm{X,i,j}) = \max_{\mathrm{X \rightarrow Y,Z}} \max_{\mathrm{i<k<j}} \mathrm{P(X \rightarrow Y,Z)} \mathrm{S}_u(\mathrm{Y,i,k})\mathrm{S}_u(\mathrm{Z,k,j}). \tag{4}$$

Finally, $S_u(\mathrm{ROOT}, 0, n)$ will store the probability of the most probable full parse tree.

1.1 TreeAnnotate

Annotation refines base treebank symbols. There are multiple ways to convert a CFG tree to a CNF tree. We choose the vertical markovization of order v, where rewrites depend on past v ancestor nodes, and the horizontal markovization of order h, where rewrites depend on the h nodes to the left. For example, for v=2 and h=2, the nonterminals in the grammar falls within the following formats:

ROOT	S ∧ ROOT
NP∧ VP	@NP∧ S_DT
VP∧ S_VBD_PP	NP∧VP... _NNP _NNP

Table 1: Nonterminal Formats (v=2, h=2)

Additionally, we reduce the grammar size by avoiding binarizing the rule further if it is already binary, without reducing the order of markovization.

1.2 Implement Details

Files 200 to 2199 of the Penn Treebank are the training data (File 2300-2399 are read as testing data). The method `annotate` in *annotateTrees* is used to binarize CFG trees with markovization level v=2 and h=2. We use `double[][][] Chart` to store S_u in equation 3 and S_b in equation 4 simultaneously. We then use `List<Integer>[][][] back` to store the information for tracing back UnaryRules and BinaryRules. `Boolean[][][]` toApplyUnary is used to store the parents of BinaryRules that needed to apply UnaryRules on in each cell. In the end, we dynamically call method of `buildTree` to build the most probable tree, which is further unannotated back to CFG tree and used to evaluate the parser performance.

We also did additional analysis by comparing the performance of CKY parser for different levels

of grammar refinement. The results are shown in Table 3.

2 Experiment Results

The CKY performance we obtained for v=2 and h=2 is shown in the following table.

P	R	F1	EX	TIME(millis)
80.57	78.84	79.7	20.8	752608

Table 2: CKY Performance (v=2, h=2)

We get 79.7 F1 with `GenerativeParser` on trees of length 40 on the development set.

We also investigate how the refinement level affects F1 score given by the following table and figure.

(v,h)	P	R	F1	EX
(1,0)	64.62	52.11	57.7	2.8
(1,1)	73.82	67.24	70.38	7.92
(1,2)	74.72	69.07	71.79	9.97
(1,∞)	74.79	69.31	71.94	10.15
(2,0)	72.27	71.53	71.9	12.42
(2,1)	80.03	78.08	79.04	19.24
(2,2)	**80.57**	**78.84**	**79.7**	**20.8**

Table 3: CKY without Pruning

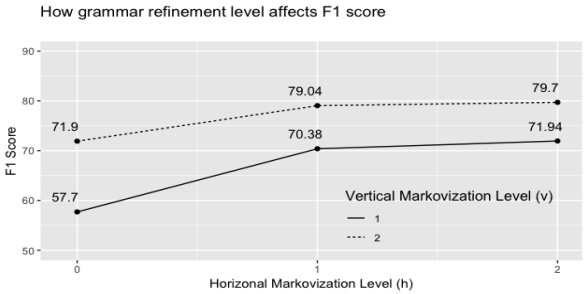

Figure 1: F1 score vs. refinement level

According to Figure 1, we found that for any given v, the marginal benefit of horizontal markovization is small. On the contrary, for any given h, the marginal benefit of vertical markovization is remarkable. This phenomenon can also be observed by Figure 2.

No matter whether v=1 or 2, when h grows form 0 to 2, the number of states increases a lot, but the F1 score just improves with small amount. However, no matter whether h=0 or 1, when v grows from 1 to 2, F1 score improves with substantial

(v,h)	0	1	2
1	99 (57.7)	707 (70.38)	2779 (71.79)
2	936 (71.9)	2337 (79.04)	6186 (79.7)

Table 4: Number of States

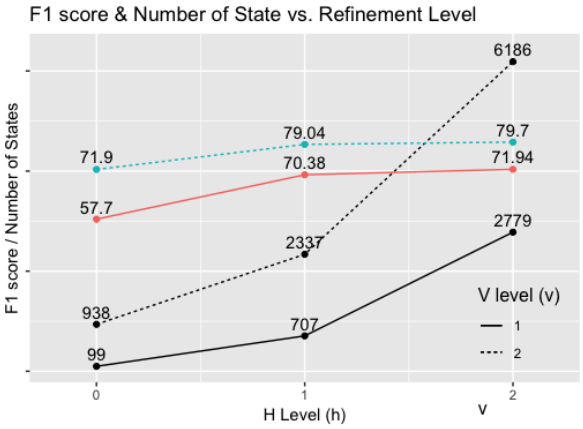

Figure 2: The Effect of Refinement Level

amount as the number of states increases. Thus we can roughly conclude that vertical markovization is more substantial than horizontal markovization in CKY parsing.

3 Coarse-to-Fine Pruning

We have built *CoarseToFineParser*. But unfortunately, it does not work properly and we fail to fix it by the deadline. More work needs to be done there.

Acknowledgments

The author would thank Taehee Jung for helpful conversation and feedback.

References

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R Shrivaths, Jeremy Moore, Michael Pozar, et al. 2006. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 168–175. Association for Computational Linguistics.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and control*, 2(2):137–167.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the*

41st Annual Meeting on Association for Computational Linguistics-Volume 1, pages 423–430. Association for Computational Linguistics.

Christopher D Manning and Hinrich Schütze. 1999. Foundations of statistical language processing.