Constituency Parsing with PCFG and CYK parser

Arun Govind M
arungm@iisc.ac.in

1 Introduction

Constituency parsing aims to extract a constituency-based parse tree from a sentence that represents its syntactic structure according to a phrase structure grammar. A constituency parse tree breaks a text into sub-phrases. Non-terminals in the tree are types of phrases, the terminals are the words in the sentence, and the edges are unlabeled.

In this assignment, CYK parser is implemented to do constituency parsing using Probabilistic context free grammars (PCFGs).

2 Data

10% sample of the Penn Treebank data from nltk interface is used to learn the grammar. The data contains 3914 sentences and their respective parse trees spread across 200 documents. PCFG rules are learned from on 80% of this data and the CYK parser is tested on the held out 20% of the data.

3 Implementation

The constituency parsing is done using two main components viz. the grammar (PCFG) and the CYK parser.

3.1 PCFG

A PCFG is defined by the following quintuple

$$G = (T, N, S, R, P)$$

where
T : set of terminals
N : set of non-terminals
S : start symbol
R : set of rules or productions
P : probability of each rule in R

3.2 Chomsky Normal Form

CYK parser requires that the PCFG be rendered in Chomsky normal form. A grammar in CNF is one wherein every rule in R is of the following form

$$X- > YZ$$

$$X- > t$$

where $X, Y, Z \in N$ and $t \in T$
A grammar in CNF makes parsing easier and more efficient.

3.3 CYK parser

CYK is a parsing algorithm for context-free grammars which employs bottom-up parsing and dynamic programming. CYK is an efficient parsing polynomial time PCFG parsing algorithm with a complexity of $O(n^2|G|)$ where n is the length of the parsed string and $|G|$ is the size of the CNF grammar G. algorithm considers every possible substring of the input string and sets $P[l, s, v]$to be true if the substring of length l starting from s can be generated from nonterminal variable R_v. Once it has considered substrings of length 1, it goes on to substrings of length 2, and so on. For substrings of length 2 and greater, it considers every possible partition of the substring into two parts, and checks to see if there is some production P Q R such that Q matches the first part and R matches the second part. If so, it records P as matching the whole substring. Once this process is completed, the sentence is recognized by the grammar if the substring containing the entire input string is matched by the start symbol.

4 Smoothing

Given the limited amount of data, smoothing is necessary for generating a good constituency parse. Without smoothing, the parser would fail to generate a parse tree for out-of-vocabulary (OOV) words.

4.1 Add one smoothing

One approach to deal with the aforementioned problem is to assign increment the frequency of every production rule that leads to a terminal by 1.

$$P(X->oov) = \frac{1}{count(X->t)+1}$$

4.2 Transition probability

Alternately, probability of a non-terminal X generating an unknown terminal token oov can be approximated as the likelihood that X generates any terminal.

$$P(X->oov) = \frac{count(X->t)}{count(X->YZ)+count(X->t)}$$

5 Results

The results of CYK parser for both smoothing methods adopted are reported below.

metric	value
labelled recall	53.85
labelled precision	43.75
F-1 score	48.28
tag accuracy	88.24

Table 1: results for add one smoothing

metric	value
labelled recall	53.21
labelled precision	43.45
F-1 score	47.84
tag accuracy	87.24

Table 2: results for transition probability

Input	he is living in my house
CYK parser	(TOP (S (NP-SBJ (PRP he)) (VP (VBZ is) (VP (VBG living) (PP-CLR (IN in) (NP (PRP$ my) (NN house)))))))
Stanford parser	(ROOT (S (NP (PRP he)) (VP (VBZ is) (VP (VBG living) (PP (IN in) (NP (PRP$ my) (NN house)))))))
Input	Arun submitted the assignment two days late
CYK parser	(TOP (S (NP-SBJ (NNP Arun) (NNP submitted)) (ADJP-PRD (NP-ADV (NP (DT the) (NN assignment)) (NP-ADV (CD two) (NNS days))) (JJ late))))
Stanford parser	(ROOT (S (NP (NNP Arun)) (VP (VBD submitted) (NP (DT the) (NN assignment)) (NP (CD two) (NNS days) (RB late)))))

Table 3: Comparison of add-one smoothing model with Stanford Parser

Input	He is dead
CYK parser	(TOP (S (NP-SBJ (PRP He)) (VP (VBZ is) (ADJP-PRD (JJ dead)))))
Stanford parser	(ROOT (S (NP (PRP He)) (VP (VBZ is) (ADJP (JJ dead)))))
Input	I am all powerful
CYK parser	(TOP (S (NP-SBJ (PRP I)) (VP (VB am) (NP (DT all) (JJ powerful)))))
Stanford parser	(ROOT (S (NP (PRP I)) (VP (VBP am) (RB all) (ADJP (JJ powerful)))))
Explanation	NP $->$ PRP 0.0162 whereas NP-SBJ $->$ PRP 0.1766
Editing rule	NP-SBJ $->$ PRP 0.1000
Result of edit	(TOP (S (NP-SBJ-1 (PRP He)) (VP (VBZ is) (ADJP-PRD (JJ dead)))))
Result of edit	(TOP (S (NP-SBJ-1 (PRP He)) (VP (VBZ is) (ADJP-PRD (JJ dead)))))

Table 4: short sentences with errors

Input	You will be required to share the path of a shell script on the server, which will take a sentence as input and produce the parsed output
CYK parser	(TOP (S (NP-SBJ (PRP You)) (VP (MD will) (VP (VB be) (VP (VBN required) (PP-CLR (TO to) (NP **(NN share)**)) (NP (NP (DT the) (NN path)) (PP (IN of) (NP (NP (DT a) (NN shell)) (PP (VBG script) (PP (IN on) (NP (DT the) (NN server,))))))) (S (NP-SBJ (WDT which)) (VP (MD will) (VP (VB take) (NP (NP (DT a) (NN sentence)) (PP (IN as) (NP (NNP input))))))) (CC and) (VP (VB produce) (NP (NP (DT the) **(NN parsed)**) (NN output)))))))
Stanford parser	(ROOT (S (NP (PRP You)) (VP (MD will) (VP (VB be) (VP (VBN required) (S (VP (TO to) (VP (VB share) (NP (NP (DT the) (NN path)) (PP (IN of) (NP (DT a) (JJ shell) (NN script)))) (PP (IN on) (NP (NP (DT the) (NN server)) (, ,) (SBAR (WHNP (WDT which)) (S (VP (MD will) (VP (VP (VB take) (NP (DT a) (NN sentence) (PP (IN as) (NP (NN input)))) (CC and) (VP (VB produce) (NP (DT the) (JJ parsed) (NN output))))))))))))))))
Explanation	’Share’ is incorrectly tagged as a NN i.e noun when it is a verb
Input	Find two short and two long sentences (from outside the treebank) where your parser produces errors but an existing parser doesn’t
CYK parser	(TOP (S (NP-SBJ (NP (NNP Find) (CD two) (JJ short) (CC and) (CD two) (JJ long) (NNP sentences) (NNP (from)) (PP (IN outside) (NP (NP (DT the) (NN tree-bank))) (SBAR (WHADVP-1 (WRB where)) (S (NP-SBJ (PRP$ your) (NNS parser)) (VP (VB produces) (UCP (JJ errors) (CC but) (NP (DT an) (VBG existing) (NN parser))))))))) (VP (VBZ does) (RB not))))
Stanford parser	(ROOT (VP (VB Find) (NP (CD two) (JJ short) (CC and) (CD two) (JJ long) (NNS sentences)) (PRN (-LRB- -LRB-) (PP (IN from) (PP (IN outside) (NP (DT the) (NN treebank)))) (-RRB- -RRB-)) (SBAR (WHADVP (WRB where)) (S (NP (PRP$ your) (NN parser)) (VP (VBZ produces) (SBAR (S (NP (NP (NNS errors)) (PP (CC but) (NP (DT an) (VBG existing) (NN parser)))) (VP (VBZ does) (RB n’t)))))))))
Explanation	’Find’ is incorrectly tagged as a NNP i.e proper noun

Table 5: Long sentences with errors