Experiment No. 2*

Parsing using CYK Algorithm

27 March 2012

1 Objective:

To implement and analysis parsing using CYK algorithm.

2 Requirements:

Python 2.6.5.

3 Theoretical Background:

The Cocke–Younger–Kasami (CYK) algorithm (alternatively called CKY) is a parsing algorithm for context–free grammars(CFG). It employs bottom-up chart parsing and dynamic programming. The algorithm requires the context-free grammar to be rendered into Chomsky normal form (CNF), because it tests for possibilities to split the current sequence in half. Any context-free grammar that does not generate the empty string can be represented in CNF using only production rules of the forms $A \rightarrow \alpha \ and \ A \rightarrow BC$ [1].

*Dept. of CSE, GEC Sreekrishnapuram

CYK computes a table summarizing the possible parses for each substring. From the table, we can quickly tell whether an input has a parse and extract one representative parse tree. Each row of the table corresponds to one length of substrings, bottom row contain string of length 1, second from bottom contain strings of length 2 and so on. The figure 1 shows an example of chart used in CYK parsing. Each X_{ij} in the table is filled by the following relation:

X_{ii} is the set of variables A such that $A \rightarrow w_i$ is a production in Grammar G.

$X_{ij} = \bigcup(\{Combination of X_{ik}, X_{kj}, i \leq k \leq j\}$

The CYK Algorithm correctly computes X_{ij} for all i and j; thus w is in L(G)

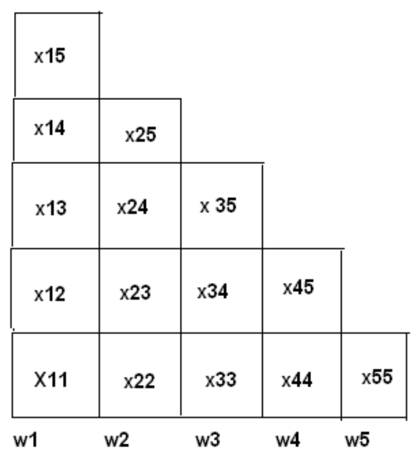

Figure 1: Parse Table for a string of length 5.

if and only if S is in X_{1n} .

4 Algorithm and Datastructure design:

Algorithm *CYK Parsing*
Input: G:CFG in CNF, string $w_1, w_2, ..., w_n$
Output: CYK Chart

$B \leftarrow \phi$
for each production of the form $A \rightarrow w_i$
    $B[i][i] \leftarrow B[i][i] \bigcup \{A\}$

for $i = 2$ *to* n
for $j = 1$ *to* $n - i + 1$
for $k = 1$ *to* $i - 1$
if $C \rightarrow B[i][k]\ B[k][j] \in G$
$B[i][j] \leftarrow B[i][j] \bigcup C$
if $Startsymbol, S \in B[1][n]$
then Accept the string
else Reject the string

5 Experimental setup:

The experiment is carried out by implementing CYK algorithm in Python. The chart of the parser is a a two dimensional array of size equal to number of words. The input is a CFG in CNF, written in another text file. The grammar file is given as command line argument to the program. The output will be the final chart, with decision on parse status of the input string.

6 Observations/Results:

1. The complexity of CYK algorithm is $O(n^3)$.
2. The CYK parsing eliminate the generation of unwanted subtrees, by keeping the parse history in chart. This makes main difference with top down parsers.
3. The algorithm can be easily modify for Probability CFG (PCFG) by adding probability computation in the algorithm.
4. Figure 2 shows the final chart after parsing grammar in Table 1.

Rules
$S -> NP\ VP$
$NP -> ART\ N$
$VP -> V\ N$
$ART -> 'the'$
$N -> 'boy'$
$N -> 'mango'$
$V -> 'eats'$

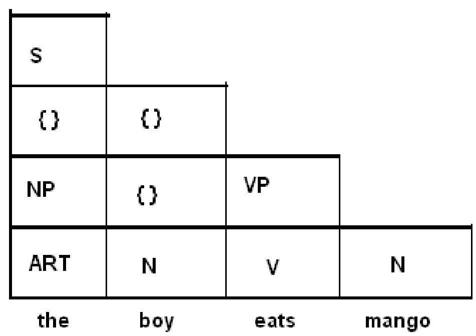

Figure 2: Final Chart for the grammar in Table 1.

References

[1] Jurafsky D.,and Martin J. H., *Speech and Language Processing,* Pearson Education, Inc. 2008.

[2] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python.* O'Reilly Media Inc., 2009.