CKY Algorithm for Chinese

ICS611
Final Project Paper
Tianli Mo

Introduction

In computer science, the Cocke-Kasami-Younger (CKY) algorithm (alternatively called CYK) is a syntactic parsing algorithm for context-free grammars (CFG). It employs bottom-up parsing and dynamic programming. It has been wildly used for syntactically parsing natural language sentences. The standard version of CYK operates only on context-free grammars given in Chomsky normal form (CNF).

In ICS611 class I have implemented assignment 3 and assignment 4 for English language, my final project aims to expand them to apply CKY parser for Chinese language. It is universally acknowledged that Chinese is one of the most complicated languages in the world. Chinese sentences not only consist of more various elements, more sophisticated grammar but also they have words segmentation problem which is hard to cope with. Therefore, many works and researchers focus on coming up with a powerful tool that can parse and understand Chinese sentences. Table 1 gives a simple comparison between English and Chinese.

Language	**Word**	**Phonology**	**Grammar**
English	Alphabet	Stress	Complicated
Chinese	Logographic system	Tone	More complicated

Table 1: Comparison between English and Chinese

Chinese sentences have 5 main structures:

◆ Sentence with Verbal Predicate

- Sentence with an Adjectival Predicate
- Sentence with a Nominal Predicate
- Sentence with a Subject-Predicate Phrase
- Subjectless Sentences

and 5 main phrases:

- Subject-Predicate phrases
- Verb-Object phrases
- Coordinative phrases
- Endocentric phrases
- Prepositional phrases

Description

CKY parser takes two things as input:

1. Context-Free Grammar;

2. Lexicon list.

The first dilemma I go into is that I am not able to find a standard context-free grammar for Chinese. In other words, I am not succeeded in finding a formal grammar for Chinese (Mandarin or Zhongwen) online or from books. Hence, I have to generate the Context-free grammar for Chinese by myself. By reading Chinese linguistic books and asking my Chinese friends' help, an "imperfect" Chinese context-free grammar will be utilized as one input. A part of the grammar shows in Table 2:

● S -> NP VP	● Nominal -> Nominal Noun
● S -> NP VP Emp	● Nominal -> PP Aux Nominal
● S -> VP	● VP -> Verb
● NP -> Pronoun	● VP -> Verb NP
● NP -> Proper-Noun	● VP -> Verb Nominal
● NP -> Det Qua Nominal	● VP -> PP Verb
● NP -> Dets Nominal	● VP -> VP PP
● Nominal -> Noun	● PP -> Preposition-Front NP

● PP -> Preposition-Front NP ● PP -> NP PrePosition-Back	PrePosition-Back

Table 2: CFG for Chinese

A part of lexicon is as follows (Table 3):

● Det -> 这 | 那 | 一
● Dets -> 这些 | 那些
● Cla -> 本 | 架 | 张 | 个
● Noun -> 书 | 飞机 | 晚餐 | 钱
● Verb -> 预定 | 包括 | 喜欢 | 跑步
● Pronoun -> 我 | 他 | 你 | 的
● Proper-Noun -> 休士顿 | 夏威夷 | 广州酒家
● Aux -> 应该 | 必须 | 的
● Preposition-Front -> 从 | 到 | 在
● Preposition-Back -> 上面 | 旁边 | 下面 | 来

Table 3: Lexicon for Chinese

By given Grammar and Lexicon, we can start to design the program to implement CKY parser. The input of the program is a context-free grammar of Chinese and lexicon of Chinese. Because CKY algorithm only operates on CNF form, therefore firstly the program converts the CFG into Chomsky Normal Form. The convert algorithm can be described briefly as follows:

- Introduce S_0

  - New S_0 -> S
- Eliminate all ε rules

  - Remove form A -> ε
- Eliminate all unit rules

  - Remove form A -> B
- Clean up remaining rules that are not in Chomsky normal form.

  - Remove form A -> $u_1u_2..u_k(k>2)$

Then the program prompts to get a Chinese sentence and use the CKY Algorithm 1 to parse the sentence. Afterwards it outputs whether this sentence meets the grammar or not, while giving a positive answer the program displays all possible parse trees of the sentence.

```
function CKY-PARSE(words, grammar) returns table

  for $j \leftarrow$ from 1 to LENGTH(words) do
      $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
      for $i \leftarrow$ from $j-2$ downto 0 do
          for $k \leftarrow i+1$ to $j-1$ do
              $table[i,j] \leftarrow table[i,j] \cup$
                              $\{A \mid A \rightarrow BC \in grammar,$
                                 $B \in table[i,k],$
                                 $C \in table[k,j]\}$
```

Algorithm 1: CKY Parser algorithm

Analysis

For program language I choose Python, which will considerably reduce my coding work when compare to if using C or Java. Instead of Python 3 I prefer to code on Python 2.7. My implementation environment is based Ubuntu 12.04 64bit by reason that Linux operating system always support python better than Windows. My laptop is: Inter Core i7 2.20GHZ, 8GB RAM, 500GB hard driver.

The program contains two parts. One is CNF.py, the other is CKY.py. As the name implies, CNF.py takes two input files: Context-Free Grammar *cfg* and lexicon *lexicon*. CNF.py convert the Context-Free Grammar into new Chomsky Normal Form *cnf*.

Next, CKY.py read the Chomsky Normal Form file *cnf* and prompts to get a Chinese sentence for input. CKY.py apply Cocke-Kasami-Younger algorithm on given sentence by referring Chomsky Normal Form. The output will be a positive answer and parse tree (bracket form) means the given sentence can be accepted by the CNF and can be parsed as well, otherwise will output a negative response.

Here is an example:

Input	他喜欢在夏威夷的饭店
Out	[他[喜欢[[在夏威夷]的饭店]]]

For comparing, here is the same meaning sentence in English:

Input	He likes the restaurants at Hawaii
Out	[He [likes [the restaurants] [at Hawaii]]]

In order to analysis my CKY parser I ask two of my friends, Xiaomeng Gao and Danyu Mo, to test it by using 10 sentences(these sentences have to only use the words in the lexicon) they like.

The first run is based on 10 logical sentences (right sentences). The result shows in Figure 1. It means Xiaomeng use 10 logical sentences which he think are good sentence to test the program, the program accept 8 sentences and output a parsing tree, the rest of two sentences are identified as wrong sentences. For Danyu's 10 sentences, the program accept 9 out of 10 sentences. Generally speaking, for this accept rate, the higher the better.

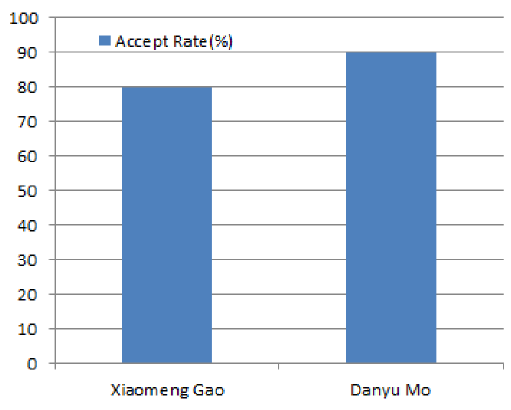

Figure 1: Accept Rate on 10 logical sentences

The second run take 10 unlogical sentences as input. The result shows in Figure 2. Similarly, this run takes each 10 unlogical (wrong) sentences from Xiaomeng and Danyu. For instance, the program accepts both 2 out of 10 sentence given by Danyu and Xiaomeng. Generally speaking, for this accept rate, the lower the better.

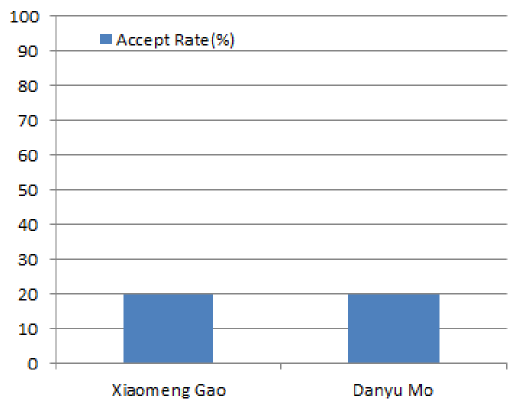

Figure 2: Accept Rate on 10 unlogical sentences

Figure 3 indicates the accurate rate that three users (including me) give to the program base their own sense about the result of parsing tree. 70% means Xiaomeng thinks 7 out of 10 parsing tree are right and the rest of 3 have problems.

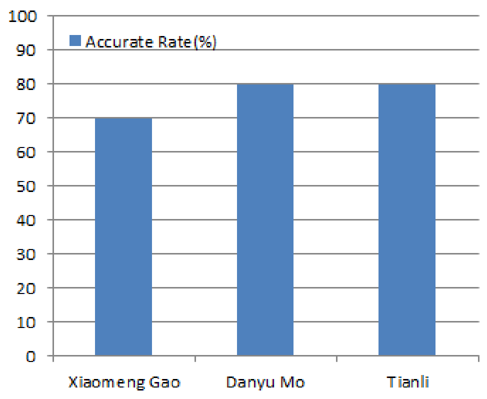

Figure 3: Accurate Rate on the parsing tree of 10 sentences

Conclusion

Overall, although my program is not good enough since only have approximately 80% accurate rate, this project really help me to understand how syntactic parser works. In coding the program, step by step, I increasingly understand Chinese Grammar more, and gained knowledge on natural language processing.

In my opinion, by modifying the Context-Free Grammar, maybe the performance will look better. The future work will focus on

- Probabilistic Context-Free Grammar

- Segmentation

References

1. Jurafsky, Daniel, and Martin, James H. *Speech and Language Processing*, 2nd ed. New Jersey: Pearson Education, Inc. 2009
2. *CYK algorithm* Wikipedia URL: http://en.wikipedia.org/wiki/CYK_algorithm
3. Sipser, Michael (1997), *Introduction to the Theory of Computation (1st ed.)*, IPS, p. 99, ISBN 0-534-94728-X

Appendix (Sample runs)

我喜欢在飞机上看书------------[我[喜欢[[在飞机上]看书]]]

那些夏威夷的酒家很好---------[[那些[夏威夷的酒家]]很好]

书在桌子上-----------------------[书在[桌子上]]

预定飞机票-----------------------[预定[飞机票]]

需要预定飞机票-----------------[需要[预定[飞机票]]]

一张钱------------------------------[一张钱]

她喜欢跑步和运动--------------[她[喜欢[[跑步]和[运动]]]]

晚餐包括鸡蛋和番茄------------[晚餐[包括[[鸡蛋]和[番茄]]]]

这架飞机在宾馆旁边------------[[这架飞机][在宾馆旁边]]

你喜欢他---------------------------[你[喜欢他]]