

Computational Linguistics

Assignment 3

Context-free grammars and CKY parsing

Sangeet Sagar
sasa00001@stud.uni-saarland.de

December 26, 2020

1 Introduction

This assignment implements the CKY algorithm for bottom-up CFG parsing and applies it to the word and the parsing problem of English. Developed in 1960, the CKY algorithm is the most used chart parser for CFGs (context-free grammars) in CNF (Chomsky normal-form). It uses a dynamic programming algorithm to tell whether a string is in the language of grammar.

2 Requirements

1. Python: [3.8.3](#)
2. NLTK: [3.5](#)
3. Texttable: [1.6.3](#). Install: `pip install texttable`

3 Project file structure

```
├── atis
│   ├── atis-grammar-cnf.cfg
│   ├── atis-grammar-original.cfg
│   ├── atis-test-sentences.txt
│   └── other_bad_sentences.txt
├── cky.py
├── README.md
└── results
    ├── summary_bad_sentences.txt
    ├── summary_tree_counts.txt
    ├── ten_sents_cyk_chart.txt
    └── ten_sents_parsed_trees.txt
```

4 Usage

- **Help:** for instructions on how to run the script with appropriate arguments.
`python cky.py -help`

```
python cky.py --help
usage: cky.py [-h]
              [-show_chart SHOW_CHART]
              [-show_tree SHOW_TREE]
              [-show_summary SHOW_SUMMARY]
              grammar_f sents_f
```

Cocke–Kasami–Younger (CKY) algorithm for bottom–up CFG parsing

Goals:

- > Write CKY algorithm and use it as a recognizer of CFG.
- > Extend it to a parser by adding back pointers
- > Get counts of all possible CKY parse trees for each sentence that is in the language of CFG

Functionalities:

- > Create CKY chart
- > Create CKY parsed trees
- > Get runtimes

positional arguments:

grammar_f	path to grammar file
sents_f	path test sentences file

optional arguments:

-h, --help	show this help message and exit
-show_chart SHOW_CHART	display CYK parsed chart
-show_tree SHOW_TREE	display CYK parsed tree
-show_summary SHOW_SUMMARY	

- **Run CYK parser:** Given CNF grammar and set of test sentences, check if these sentences are in the language of grammar and also display counts of all possible CKY parsed trees.
`python cky.py atis/atis-grammar-cnf.cfg atis/atis-test-sentences.txt`
- Run and test the parser on some self-made sentences that are ungrammatical (i.e. not in the language of given CFG)
`python cky.py atis/atis-grammar-cnf.cfg atis/other_bad_sentences.txt`

5 Runtime

- **Total** runtime: 20.51 s
- **CYK parser** runtime: 17.76 s
- **Backpointer** runtime: 0.015 s

However, if you use optional arguments `-show_chart` or `-show_tree`, the total runtime is as follows:

- Total runtime: `-show_chart`: 23.67 s
- Total runtime: `-show_tree`: 285.27 s

6 Results Contents

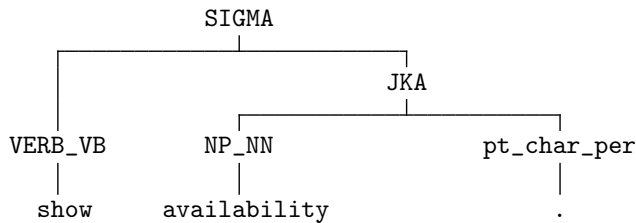
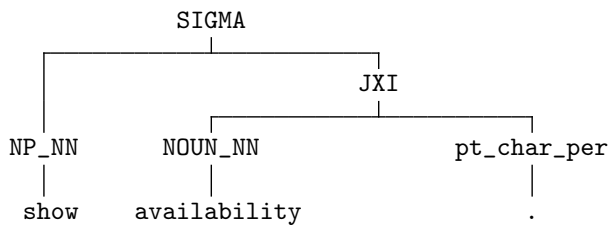
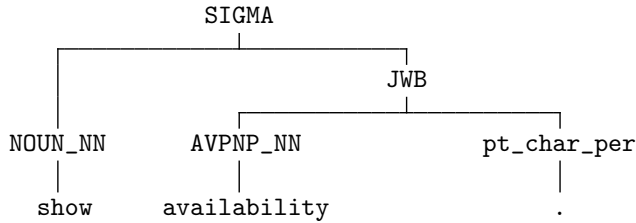
- `summary_tree_counts.txt`: Summary table of given ATIS test set with 98 sentences. Display if the sentence is in the language of CFG and counts of all possible CYK parse trees.
- `ten_sents_parsed_trees.txt`: Shows CYK parsed trees of the first 10 sentences from the ATIS test-set
- `ten_sents_cyk_chart.txt`: Shows CYK chart of first 10 sentences from the ATIS test-set
- `summary_bad_sentences.txt`: (Summary table of some self-made sentences) Shows if the sentences are in the language of CFG and counts of the parse tree for each.

7 Glimpse of results

- CKY tree of the sentence `show availability .`
A total of 3 trees are observed and they are:

(1) `show availability .`

Given sentence is in the language of CFG



- Summary table for first 10 sentences.

S.No.	test sentence	CFG	parse tree
			counts
1	prices .	True	2
2	show availability .	True	3
3	show the flights .	True	2
4	milwaukee to detroit .	True	2
5	indianapolis to seattle .	True	2
6	list round trips .	True	11
7	list saturday flights .	True	5
8	what aircraft is this .	False	0
9	list these economy fares .	False	0
10	list these city destinations .	False	0