Name: Sanket S. Fulzele

Roll No: 371018

PRN: 22110728

Div: A

The CKY (Cocke-Kasami-Younger) algorithm and its probabilistic variant, probabilistic CKY (PCYK), are parsing algorithms used in natural language processing (NLP) for syntactic parsing. Both algorithms are based on dynamic programming techniques and are particularly efficient for parsing context-free grammars (CFGs).

CKY Algorithm

The CKY algorithm is a bottom-up parsing algorithm that constructs a parse tree for a given sentence based on a CFG. It uses dynamic programming to efficiently compute and store partial parse results, reducing the time complexity of parsing.

Key Steps:

1. Initialization: Initialize a parse table with cells representing different spans of the sentence.
2. Parsing Table Filling: Fill in the table bottom-up by combining smaller spans to create larger spans based on CFG rules.
3. Backtracking: Trace back through the table to reconstruct the parse tree. The CKY algorithm is efficient and can handle ambiguous grammars and sentences with multiple parse trees. However, it requires the CFG to be in Chomsky Normal Form (CNF), where production rules are either of the form A -> BC or A -> w, where A, B, C are non-terminals and w is a terminal.

Probabilistic CKY (PCYK) Algorithm

The PCYK algorithm extends the CKY algorithm to handle probabilistic context-free grammars (PCFGs), where each production rule has associated probabilities. PCYK computes the most probable parse tree for a given sentence based on these probabilities.

Key Differences:

1. Probabilistic Scoring: PCYK assigns probabilities to each parse tree and selects the most probable parse tree based on these scores.
2. Parsing Table Initialization: The parse table in PCYK includes probabilities for each cell, representing the probability of generating a span with a specific non-terminal.
3. Probabilistic Parsing: During parsing table filling, PCYK computes probabilities for each possible split and combines probabilities according to CFG rules and production probabilities.

Comparison and Applications

1. Efficiency: Both CKY and PCYK are efficient parsing algorithms suitable for parsing CFGs and PCFGs, respectively.

2. Ambiguity Handling: CKY and PCYK can handle ambiguous grammars and sentences with multiple parse trees.
3. Probabilistic Modeling: PCYK is particularly useful for probabilistic parsing tasks, such as statistical parsing and machine translation, where probabilities play a crucial role in selecting the most likely parse tree.

```
import nltk

# Define CFG rules
cfg_rules = """
    S -> NP VP
    NP -> Det N | Det N PP | 'I'
    VP -> V NP | VP PP
    PP -> P NP
    Det -> 'a' | 'an' | 'the' | 'my'|'The'
    N -> 'dog' | 'cat' | 'ball' | 'park'|'bone'
    V -> 'chased' | 'ate' | 'threw'
    P -> 'in' | 'on' | 'at'
"""

# Create a CFG parser
cfg_parser = nltk.ChartParser(nltk.CFG.fromstring(cfg_rules))

# Define sentences to parse
sentences = [
    "I chased the dog in the park",
    "The cat ate my ball",
    "The dog chased the cat in the park",
    "I threw the ball at the cat",
    "The dog ate a bone"
]

# Define a function to parse sentences using CKY algorithm
def parse_sentence(sentence):
    tokens = nltk.word_tokenize(sentence)
    parsed_trees = cfg_parser.parse(tokens)
    return list(parsed_trees)

# Parse and print the parse trees for each sentence
for sentence in sentences:
    print(f"Sentence: {sentence}")
    parse_trees = (parse_sentence(sentence))
    for tree in parse_trees:
        print(tree)
    print()
```

```
Sentence: I chased the dog in the park
(S
  (NP I)
  (VP
    (VP (V chased) (NP (Det the) (N dog)))
    (PP (P in) (NP (Det the) (N park)))))
(S
  (NP I)
  (VP
    (V chased)
    (NP (Det the) (N dog) (PP (P in) (NP (Det the) (N park))))))

Sentence: The cat ate my ball
(S (NP (Det The) (N cat)) (VP (V ate) (NP (Det my) (N ball))))

Sentence: The dog chased the cat in the park
(S
  (NP (Det The) (N dog))
  (VP
    (VP (V chased) (NP (Det the) (N cat)))
```

```
    (PP (P in) (NP (Det the) (N park)))))
(S
  (NP (Det The) (N dog))
  (VP
    (V chased)
    (NP (Det the) (N cat) (PP (P in) (NP (Det the) (N park))))))

Sentence: I threw the ball at the cat
(S
  (NP I)
  (VP
    (VP (V threw) (NP (Det the) (N ball)))
    (PP (P at) (NP (Det the) (N cat)))))
(S
  (NP I)
  (VP
    (V threw)
    (NP (Det the) (N ball) (PP (P at) (NP (Det the) (N cat))))))

Sentence: The dog ate a bone
(S (NP (Det The) (N dog)) (VP (V ate) (NP (Det a) (N bone))))
```