Laporan Analisis Constituency Parsing

Constituency parsing adalah adalah tugas memecah teks menjadi sub-frasa, atau konstituen. Non-terminal dalam pohon parse adalah jenis frase, terminal adalah kata-kata dalam kalimat.

Bagian 1

1. Buat definisi grammar CFG yang mengandung ambiguitas

```
grammar_1  = nltk.CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP PP
    PP -> P N
    V -> "menembak" | "melihat" | "mengambil"
    NP -> "aziz" | "polisi" | "irawan" | Det N | Det N PP
    Det -> "seekor" | "seorang" | "semangkok"
    N -> "gajah" |"penjahat" | "piyamanya" |"majalah" |"bakso" | "kaki" | "pistolnya"
    P -> "dengan" | "oleh" | "di" | "pada"
    """)
```

2**.** Contoh 3 kalimat, yang telah di*parsing*/penguraian berdasarkan *grammar* .Sertakan keterangan pohon hasil dari proses penguraian, beri keterangan jika terdapat ambiguitas.

Kalimat 1: `sent_1 = 'irawan menembak seekor gajah dengan piyamanya'.split()`

Kalimat 2 : `sent_2 = 'aziz mengambil semangkok bakso dengan kaki'.split()`

Kalimat 3 : `sent_3 = 'polisi melihat seorang penjahat dengan pistolnya '.split()`

-Parser dengan Top Down Chart untuk kalimat 1

```
(S
  (NP irawan)
  (VP
    (V menembak)
    (NP (Det seekor) (N gajah) (PP (P dengan) (N piyamanya)))))
(S
  (NP irawan)
  (VP
    (V menembak)
    (NP (Det seekor) (N gajah))
    (PP (P dengan) (N piyamanya))))
```

Berdasarkan output dari parser tersebut terdapat ambiguitas yaitu "**irawan menembak seekor gajah saat menggunakan piyama/pakaian malam**" atau "**irawan menembak seekor gajah yang menggunakan piyama"**

-Parser dengan Top Down Chart untuk kalimat 2

```
(S
  (NP aziz)
  (VP
    (V mengambil)
    (NP (Det semangkok) (N bakso) (PP (P dengan) (N kaki)))))
(S
  (NP aziz)
  (VP
    (V mengambil)
    (NP (Det semangkok) (N bakso))
    (PP (P dengan) (N kaki))))
```

Berdasarkan output dari parser tersebut terdapat ambiguitas yaitu "**aziz mengambil semangkok bakso menggunakan kakinya**" atau "**aziz mengambil semangkok bakso dengan berjalan kaki"**

-Parser dengan Top Down Chart untuk kalimat 3

```
(S
  (NP polisi)
  (VP
    (V melihat)
    (NP (Det seorang) (N penjahat) (PP (P dengan) (N pistolnya)))))
(S
  (NP polisi)
  (VP
    (V melihat)
    (NP (Det seorang) (N penjahat))
    (PP (P dengan) (N pistolnya))))
```

Berdasarkan output dari parser tersebut terdapat ambiguitas yaitu "**polisi melihat penjahat dengan pistolnya. Siapa yang memiliki pistolnya? Polisi atau penjahat yang melihat dengan pistolnya?**

3. Periksa apakah grammar tersebut memenuhi syarat CNF?

```
print(grammar_1.is_chomsky_normal_form())
```

```
False
```

Grammar tersebut belum memenuhi syarat CNF karena masih terdapat lebih dari 2 non terminal

1. A → B C (ruas kanan terdiri atas 2 simbol non terminal)
2. A → w (ruas kanan terdiri atas 1 symbol terminal)

4. **Ubah menjadi CNF**

```
grammar_1  = nltk.CFG.fromstring("""
    S -> NP VP
    VP -> V NP | V NP1
    NP1 -> NP PP
    PP -> P N
    V -> "menembak" | "melihat" | "mengambil"
    NP -> "aziz" | "polisi" | "irawan" | Det N | Det N1
    N1 -> N PP
    Det -> "seekor" | "seorang" | "semangkok"
    N -> "gajah" |"penjahat" | "piyamanya" |"majalah" |"bakso" | "kaki" | "pistolnya"
    P -> "dengan" | "oleh" | "di" | "pada"
    """)
print(grammar_1.is_chomsky_normal_form())
```

```
True
```

Output berubah jadi true artinya memenuhi syarat CNF

Bagian 2

1. **Lakukan induksi grammar PCFG dari Treebank constituency Bahasa Indonesia**

```
from nltk import Nonterminal, nonterminals, Production, PCFG, induce_pcfg
S = Nonterminal('S')
productions = []
for t in ptb20:
    productions += t.productions()
grammar_3 = induce_pcfg(S, productions)
print(grammar_3)
```

```
Grammar with 302 productions (start state = S)
    NP -> NN SBAR [0.0215054]
    NN -> 'Kera' [0.00980392]
    SBAR -> IN S [0.65]
```

Induksi grammar hanya 20 array/ 20 kalimat diawal dari Treebank constituency Bahasa Indonesia

2. Lakukan Parsing dengan bottom-up parser

Analisis hasil dari parsing tersebut adalah untuk kalimat yang ada diconstituency Treebank sedangkan untuk kalimat uji **bukan** dari *constituency Treebank*

```
sent_3 = 'Jumlah monyet di ibukota India mencapai ribuan'.split()
# contoh menggunakan bottom-up parser
bu_parser = nltk.parse.BottomUpChartParser(grammar_3)

for tree in bu_parser.parse(sent_3):
    print(tree)
```

```
Streaming output truncated to the last 5000 lines.
      (NP
        (NN Jumlah)
        (NP
          (NP (NN monyet))
          (PP (IN di) (NP (NP (NN ibukota) (NP (NP (NNP India)))))))))
  (VP (VB mencapai) (NP (CD ribuan))))
```

Gambar1 kalimat yang ada diconstituency Treebank

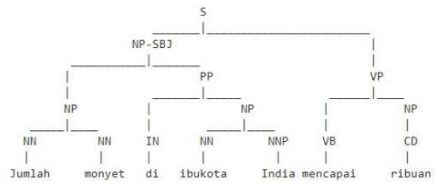
```
sent_4 = 'ribuan Pemkot mengusir monyet kecil'.split()
# contoh menggunakan bottom-up parser
bu_parser = nltk.parse.BottomUpChartParser(grammar_3)

for tree in bu_parser.parse(sent_4):
    print(tree)
```

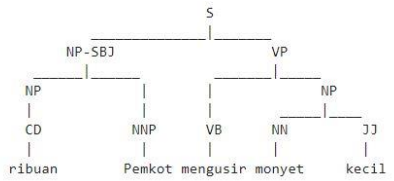
```
(S
  (NP-SBJ (NP (CD ribuan)) (NP (NN Pemkot)))
  (VP (VB mengusir) (S (NP-SBJ (NN monyet)) (ADJP-PRD (JJ kecil)))))
(S
  (NP-SBJ (NP (CD ribuan)) (NP (NN Pemkot)))
  (VP
    (VB mengusir)
    (S (NP-SBJ (NP (NN monyet))) (ADJP-PRD (JJ kecil)))))
```

Gambar 2 kalimat yang bukan dari constituency Treebank

2.1 Lakukan Parsing dengan viterbi parser



Gambar3 kalimat yang ada diconstituency Treebank



Gambar 4 kalimat yang bukan dari constituency Treebank

Berdasarkan output dari parsing menggunakan bottom-up parser untuk kalimat uji yang ada pada constituency yang dihasilkan sangatlah banyak sampai terpotong hingga 5000 baris terakhir sedangkan untuk kalimat uji yang bukan dari constituency treebank pohon tree yang dihasilkan lebih sedikit sekitar 70 Tree artinya ambiguitas pada kalimat yang ada diconsituency Treebank lebih banyak karena setiap node akan dikalikan terminal ataupun non terminal. Parsing dengan Viterbi parser untuk kalimat uji yang ada diconsituency dan bukan dari constituency sesuai grammar yang sudah ditentukan sebelumnya. Dan hasil parsing tree menggunakan viterbi itu ada pada hasil parsing menggunakan metode bottom-up parser.