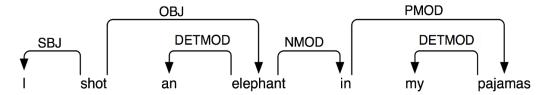
## **Laporan Analisis Dependency Parsing**

# Bagian 1

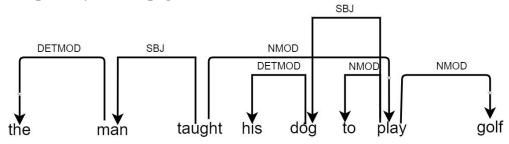
## 1. - Dependecy Tree dep\_grammar\_1



### Relasi head-dependent

- Headnya kata shot dan dependentnya adalah "I" relasinya "I" sebagai SBJ(Subjek) terhadap kata shot, Headnya kata shot dan dependent kata elephant relasinya "elephant" sebagai OBJ(Object) dari kata shot dan juga "in" relasinya sebagai(Noun modifier)
- Headnya kata elephant dan dependentnya adalah kata "an" relasinya sebagai det modifier dan kata "in" relasinya sebagai NMOD(Noun Modifier)
- Headnya kata "in" dan dependentnya adalah kata "pajamas" relasinya sebagai PMOD(Pronoun modifier)
- Headnya kata "pajamas" dan dependentnya adalah kata"my" relasinya sebagai DETMOD(Determinan Modifier)

# - Dependecy Tree dep\_grammar\_2



## Relasi head-dependent

- Headnya kata "man" dan dependentnya adalah "the" relasinya "the" sebagai DETMOD(Determinan Modifier)
- -Headnya kata "taught" dan dependent kata " man" relasinya sebagai SBJ(Subjek) dan dependentnya play relasinya sebagaiNMOD(Noun Modifier).
- -Headnya kata "dog" dan dependent kaya "his" relasinya DETMOD(Determinan Modifier)
- -Headnya kata "play" dan dependentnya kata "golf" relasinya sebagai NMOD(Noun Modifier), dependentnya kata "to" relasinya sebagai NMOD(Noun Modifiers dan dependentnya kata "dog" relasinya sebgai SBJ(Subject)

## 2. - Tipe Relasi Dependecy Tree dep\_grammar\_1

- Tipe SBJ menunjukan peran kata I sebagai subject terhadap kata want, Tipe OBJ menunjukan peran kata elephant sebagai oject dari kata shot, tipe DETMOD menunjukan peran kata my dan an

untuk kepunyaan/kepemilikan, tipe PMOD menunjukan peran kata pajamas sebagai keterangan dan tipe NMOD menunkan peran kata "in" sebagai pengubah kata benda

# - Tipe Relasi Dependecy Tree dep\_grammar\_2

-Tipe SBJ menunjukan peran kata man sebagai subject terhadap kata taught peran kata dog sebagai subject terhadap kata play, tipe DETMOD menunjukan peran kata his menerangkan kepunyaan/kepemilikan, dan tipe NMOD menunjukan peran kata dog, golf, dan to sebagai pengubah kata benda

# 3. Perbedaan Projective parser dan Non-Projective parser

Projective parser ketika semua kata ditulis dalam urutan linear yang sesuai dengan tata bahasa struktur frase sedangkan Non-Projective parser tata bahasa struktur frase bersifat lebih fleksibel.

### 4. Format CONLL

Adalah sebuah format yang mewakili korpus dengan satu kata per baris, setiap baris kata berisi 10 kolom yang dipisahkan tab dengan informasi tentang kata tersebut. Atributnya terdiri dari ID yaitu indeks kata, FORM yaitu bentuk kata, LEMMA yaitu bentuk kata yang disederhanakan, CPOSTAG yaitu POST-tag terperinci, FEATS yaitu kumpulan fitur yang diabaikan secara default, HEAD yaitu kepala token saat ini , DEPREL yaiutu relasi ketergantungan, PHEAD yaitu kepala proyektif, PDEPREL yaitu relasi ketergantungan untuk PHEAD

#### Bagian 2

## 5. Dependecy Grammer untuk parsing Projective Parser

## 1. Definisikan Dependency Parser

```
[106] dep_grammar_1 = nltk.DependencyGrammar.fromstring("""
    'melihat' -> 'aziz' | 'seorang' | 'dengan'
    'seorang' -> 'penjahat' | 'dengan'
    'dengan' -> 'pistol'
    'pistol' -> 'miliknya'
    """)
```

Alasan grammar tersebut sesuai karena berdasarkan penulisan kata ditulus dalam urutan linear yang sesuai dengan tata bahasa . Kegunaan Dependecy grammar disini untuk mendefinisikan kata mana saja yang akan dilakukan parsing

#### 2. Print Dependency Grammar

```
Dependency grammar with 7 productions
'melihat' -> 'aziz'
'melihat' -> 'seorang'
'melihat' -> 'dengan'
'seorang' -> 'penjahat'
'seorang' -> 'dengan'
'dengan' -> 'pistol'
'pistol' -> 'miliknya'
```

Ini berguna untuk melihat head(kepala) dan juga dependentnya (yang bergantung dengan head)

### 3. Lakukan parsing dengan Parser Dependecy Projective

```
pdp = nltk.ProjectiveDependencyParser(dep_grammar_1)
sent = 'aziz melihat seorang penjahat dengan pistol miliknya'.split()
trees = pdp.parse(sent)

for tree in trees:
    print(tree)

(melihat aziz (seorang penjahat) (dengan (pistol miliknya)))
(melihat aziz (seorang penjahat (dengan (pistol miliknya))))
```

Berdasarkan hasil parsing diatas sebuah kata dan semua turunannya membentuk urutan kata yang berdekatan dalam kalimat . Struktur ketergantungan dalam tanda kurung ini juga dapat ditampilkan sebagai pohon, di mana tanggungan ditampilkan sebagai dependect dari headnya.

- Headnya yaitu kata "melihat" dan dependentnya yaitu "aziz", "seorang", dan "dengan"
- Headnya yaitu kata "seorang" dan dependentnya yaitu "penjahat", "dengan" dan "pistol"
- Headnya yaitu kata "dengan" dan dependentnya yaitu 'pistol"
- Headnya yaitu kata "pistol" dan dependentnya yaitu "miliknya"

## 5. Dependecy Grammer untuk parsing Non-Projective Parser

## 1. Definisikan Dependency Parser untuk Parser NOn-Projective

```
dep_grammar_2 = nltk.DependencyGrammar.fromstring("""
   'mengajak' -> 'bermain' | 'wanita'
   'wanita' -> 'itu'
   'bermain' -> 'basket' | 'teman' | 'untuk'
   'teman' -> 'kecilnya'
   """)
```

Alasan grammar tersebut tata bahasa struktur frase bersifat lebih fleksibel dan ditulis tidak dengan tata bahasa . Kegunaan Dependecy grammar disini untuk mendefinisikan kata mana saja yang akan dilakukan parsing

## 2. Print Dependency Grammar

```
print(|dep_grammar_2|)

Dependency grammar with 7 productions
  'mengajak' -> 'bermain'
  'mengajak' -> 'wanita'
  'wanita' -> 'itu'
  'bermain' -> 'basket'
  'bermain' -> 'teman'
  'bermain' -> 'untuk'
  'teman' -> 'kecilnya'
```

Ini berguna untuk melihat head(kepala) dan juga dependentnya (yang bergantung dengan head)

### 3. Lakukan parsing dengan Parser Non Dependecy Projective

```
{0: {'address': 0,
    'ctag': 'TOP',
    'deps': defaultdict(<class 'list'>, {'ROOT': [3]}),
    'feat': None,
    'bad': None.
3: {'address': 3,
'ctag': None,
'deps': defaultdict(<class 'list'>, {'': [1, 4, 8]}),
         'feats': None,
'head': None,
'lemma': None,
                                                                                                                                                                             'lemma': None.
                                                                                                                                                                            'rel': None,
'tag': 'TOP',
'word': None},
'rel': None,
'tag': None,
'word': 'mengajak'},
4: {'address': 4,
                                                                                                                                                                   1: ('address': 1,
    'ctag': None,
    'deps': defaultdict(<class 'list'>, {'': [2]}),
        'deps': defaultdict(<class 'list'>, {'': [7, 9]}),
'feats': None,
'head': None,
'lemma': None,
                                                                                                                                                                             'feats': None,
'head': None,
'lemma': None,
'rel': None,
'rel': None,
'tag': None,
'word': 'bermain'},
5: {'address': 5,
                                                                                                                                                                            'rel': None,
'tag': None,
'word': 'wanita'},
                                                                                                                                                                   2: {'address': 2,
'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
        'address': s,
'ctag': None,
'deps': defaultdict(<class 'list'>, {'': [6]}),
'feats': None,
'head': None,
'lemma': None,
'rel': None,
'tag': None,
'word: 'teman'},
                                                                                                                                                                            'feats': None,
'head': None,
                                                                                                                                                                             'lemma': None.
                                                                                                                                                                            'rel': None,
'tag': None,
'word': 'itu
                                                                                                                                                                                           'itu'},
 6: {'address': 6,
                                                                                                                                                                   word . 10 },
3: {'address': 3,
   'ctag': None,
   'deps': defaultdict(<class 'list'>, {'': [1, 4, 8]}),
         'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
        'feats': None,
'head': None,
'lemma': None,
                                                                                                                                                                           'feats': None,
'head': None,
'lemma': None,
        'rel': None,
                                                                                                                                                                             9: {'address': 9,
    4: {'address': 4,
                                                                                                                                                                                      'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
'feats': None,
           { dubress : 4,
  'ctag': None,
  'deps': defaultdict(<class 'list'>, {}),
  'feats': None,
  'head': None,
  'lemma': None,
  'rel': None,
                                                                                                                                                                                     'head': None,
'lemma': None,
                                                                                                                                                                        'rel': None,
'tag': None,
'word': 'basket'}})
(<function DependencyGraph.__init__.<locals>.<lambda> at 0x7fbe92d271e0>,
   'tag': None,
'word': 'bermain'},
5: {'address': 5,
                                                                                                                                                                           {0: {'address': 0,
    'ctag': 'TOP',
    'deps': defaultdict(<class 'list'>, {'ROOT': [3]}),
            claudiess : 5,
'ctag': None,
'deps': defaultdict(<class 'list'>, {'': [6]}),
'feats': None,
                                                                                                                                                                                    'feats': None,
'head': None,
'lemma': None,
'rel': None,
'tag': 'TOP',
'word': None},
            'head': None,
'lemma': None,
           'rel': None,
'tag': None,
'word': 'teman'},
    6: {'address': 6,
                                                                                                                                                                           1: {'address': 1,
           'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
'feats': None,
'head': None,
'lemma': None,
                                                                                                                                                                                    'ctag': None,
'deps': defaultdict(<class 'list'>, {'': [2]}),
'feats': None,
                                                                                                                                                                                    'head': None,
'lemma': None,
   remma: None,
'rel': None,
'tag': None,
'word': 'kecilnya'},
7: {'address': 7,
                                                                                                                                                                                     'rel': None,
'tag': None,
'word': 'wanita'},
                                                                                                                                                                           2: {'address': 2,
            'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
'feats': None,
'head': None,
                                                                                                                                                                                     'ctag': None,
'deps': defaultdict(<class 'list'>, {}),
'feats': None,
             'lemma': None
                                                                                                                                                                                      'lemma': None,
             'rel': None,
```

Berdasarkan hasil parsing diatas hasil parsing berupa dictonery berisi key dan value, dimana key berisi address, ctag, deps, feats, head, lemma, rel, tag dan word. Kita bisa mengakses data melalui address dan juga terdapat informasi root dari parsing tersebut. Hasil dari **Non Dependecy Projective** membentuk banyak parses karena struktur frase bersifat lebih fleksibel dan ditulis tidak dengan tata bahasa