FAKULTAS ILMU KOMPUTER, UNIVERSITAS INDONESIA

CSC4602354 NATURAL LANGUAGE PROCESSING

TUGAS 1 STATISTICAL LANGUAGE MODELLING (POS TAGGING & PROBABILISTIC PARSING)

Yohanes Gultom

1506706345

April 18, 2016

I. POS Tagging

1. Deskripsi Penanganan Format

Format yang digunakan pada korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv` adalah:

```
<kalimat id=1>
Kera  NN
untuk SC
amankan VB
pesta olahraga  NN
</kalimat>
<kalimat id=2>
Kata1 POS1
kata2 POS2
kata3 POS3
</kalimat>
```

1.1. NLTK

Untuk NLTK[1] (Python), API yang ada untuk melakukan *training POS Tagger* menerima `list of list of tuples` yang memiliki representasi seperti:

```
[[(Kera, NN), (untuk, SC), (amankan, VB), (pesta olahraga, NN)], [(Kata1, POS1), (Kata2, POS2), (Kata3, POS3)]]
```

Untuk membaca file korpus dan mengubahnya menjadi objek yang sesuai dengan argumen masukan API, digunakan fungsi `parse_train_data` dengan kode sebagai berikut:

```
def parse_train_data(train_file_path):
  p = re.compile('<kalimat id=(.+)>((.|\n)*?)</kalimat>')
  file = open(train_file_path, 'r')
  raw = file.read()
  sents = []
```

[1] `http://www.nltk.org/`

```
  for s in p.findall(raw):
    words = s[1].strip().split('\n')
    sent = []
    for w in words:
      array = w.split('\t')
      sent.append((array[0], array[1]))
    sents.append(sent)
  file.close()
  return sents
```

Kalimat diekstrak menggunakan regex `<kalimat id=(.+)>((.|\n)*?)</kalimat>`, kemudian dibaca per baris dan dipisahkan dengan karakter `\t`. NLTK mengakomodasi nilai terminal yang dipisahkan spasi jadi tidak ada perlakuan khusus untuk terminal seperti `pesta olahraga`.

1.2. OpenNLP

Sedangkan jika gunakan kakas OpenNLP[2] (Java), untuk melakukan *training POS Tagger* dibutuhkan format file input sebagai berikut:

```
Kera_NN untuk_SC amankan_VB pesta-olahraga_NN
Kata1_POS1 Kata2_POS2 Kata3
```

Untuk mengubah format dari tiap kalimat pada `Indonesian_Manually_Tagged_Corpus_ID.tsv` ke format yang dibutuhkan *POS Tagger* OpenNLP digunakan fungsi `convertTrainString` berikut:

```
private static final String XML_TAG_PATTERN = "<(\"[^\"]*\"|'[^']*'|[^'\">])*>";

/* other code.. */

public String convertTrainString(String line, boolean firstWord) {
  String toWrite = null;
  matcher = pattern.matcher(line);
  if (!matcher.matches()) {
    // read the word if it's not an xml tag
    // replace tab with underscore
    if (line.contains(" ")) {
      // split compound word to multiple words with same tag
      String[] temp = line.split("\t");
      String tag = temp[1];
      String[] words = temp[0].split(" ");
      for (int i = 0; i < words.length; i++) {
        words[i] += "_" + tag;
      }
      toWrite = StringUtils.join(words, " ");
    } else {
      toWrite = line.replace("\t", "_");
    }
    if (!firstWord) {
      toWrite = " " + toWrite;
    }
  } else if (!firstWord) {
```

[2] `https://opennlp.apache.org/`

```
    // write newline every time we see an xml tag
    toWrite = "\n";
  }
  return toWrite;
}
```

Regex `<(\"[^\"]*\"|'[^']*'|[^'\">])*>` digunakan untuk mengenali dan melewati tag `<kalimat>`. Kemudian tiap kata dipisahkan dengan tag-nya menggunakan karakter `\t`. Untuk menghindari kegagalan parsing, karakter spasi diubah menjadi -. Hal ini mengakibatkan kata tersebut menjadi tidak berguna dalam proses *tagging* tetapi kata lainnya dalam kalimat tersebut tetap bisa berguna.

2. Hasil Uji Coba dan Analisis

2.1. NLTK

Program yang dibuat menggunakan API NLTK adalah `tagger.py`[3] yang menggunakan `nltk.tag.tnt`. Uji coba pertama adalah melakukan POS *tagging* pada paragraf Listing 1 yang diambil dari Wikipedia Indonesia[4] dengan tagger yang dilatih menggunakan korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv`. Perintah yang dijalankan adalah:

```
python tagger.py ../data/pos-tagging/Indonesian_Manually_Tagged_Corpus_ID.tsv ../data/
    pos-tagging/Wikipedia.txt
```

Hasil dari eksekusi perintah ini adalah file bernama `Wikipedia.nltk.tagged` yang berisi hasil *tagging* seperti pada Listing 2. Waktu yang dibutuhkan program untuk melakukan *tagging* adalah 1.960335 detik.
Pada saat melakukan *parsing* korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv`, ternyata jumlah kalimat yang diekstraksi adalah 10030. Hal ini terjadi karena terdapat kalimat yang memiliki id yang sama seperti 14a dan 14b.

```
Koperasi di Indonesia
Koperasi di Indonesia, menurut UU tahun 1992, didefinisikan sebagai badan usaha yang
    beranggotakan orang-seorang atau badan hukum koperasi dengan melandaskan
    kegiatannya berdasarkan prinsip-prinsip koperasi sekaligus sebagai gerakan ekonomi
    rakyat yang berdasar atas asas kekeluargaan .
Di Indonesia, prinsip koperasi telah dicantumkan dalam UU No . 12 Tahun 1967 dan UU No .
     25 Tahun 1992 .
Prinsip koperasi di Indonesia kurang lebih sama dengan prinsip yang diakui dunia
    internasional dengan adanya sedikit perbedaan, yaitu adanya penjelasan mengenai SHU
     (Sisa Hasil Usaha) .

Sejarah koperasi di Indonesia
Sejarah singkat gerakan koperasi bermula pada abad ke-20 yang pada umumnya merupakan
    hasil dari usaha yang tidak spontan dan tidak dilakukan oleh orang-orang yang
    sangat kaya .
Koperasi tumbuh dari kalangan rakyat, ketika penderitaan dalam lapangan ekonomi dan
    sosial yang ditimbulkan oleh sistem kapitalisme semakin memuncak .
```

[3] https://github.com/yohanesgultom/nlp-experiments/blob/master/python/tagger.py
[4] https://id.wikipedia.org

```
Beberapa orang yang penghidupannya sederhana dengan kemampuan ekonomi terbatas ,
    terdorong oleh penderitaan dan beban ekonomi yang sama, secara spontan
    mempersatukan diri untuk menolong dirinya sendiri dan manusia sesamanya .
```

Listing 1: Wikipedia.txt

```
Koperasi_NN di_IN Indonesia_NNPKoperasi_NN di_IN Indonesia_NNP ,_Z menurut_IN UU_NNP
    tahun_NN 1992_CD ,_Z didefinisikan_VB sebagai_IN badan_NN usaha_NN yang_SC
    beranggotakan_VB orang-seorang_Unk atau_CC badan_NN hukum_NN koperasi_NN
    dengan_IN melandaskan_Unk kegiatannya_Unk berdasarkan_VB prinsip-prinsip_NN
    koperasi_NN sekaligus_RB sebagai_IN gerakan_NN ekonomi_NN rakyat_NN yang_SC
    berdasar_VB atas_IN asas_Unk kekeluargaan_Unk ._ZDi_IN Indonesia_NNP ,_Z
    prinsip_NN koperasi_NN telah_MD dicantumkan_VB dalam_IN UU_NNP No_NNP ._Z 12
    _CD Tahun_NN 1967_NNP dan_CC UU_NNP No_NNP ._Z 25_CD Tahun_NNP 1992_CD .
    _ZPrinsip_Unk koperasi_NN di_IN Indonesia_NNP kurang_RB lebih_RB sama_JJ
    dengan_IN prinsip_NN yang_SC diakui_VB dunia_NN internasional_JJ dengan_IN
    adanya_NN sedikit_CD perbedaan_NN ,_Z yaitu_SC adanya_NN penjelasan_NN
    mengenai_IN SHU_Unk (_Z Sisa_NN Hasil_NN Usaha_NN )_Z ._ZSejarah_NN koperasi_NN
    di_IN Indonesia_NNPSejarah_NN singkat_JJ gerakan_NN koperasi_NN bermula_Unk
    pada_IN abad_NN ke-20_Unk yang_SC pada_IN umumnya_NN merupakan_VB hasil_NN
    dari_IN usaha_NN yang_SC tidak_NEG spontan_Unk dan_CC tidak_NEG dilakukan_VB
    oleh_IN orang-orang_NN yang_SC sangat_RB kaya_JJ ._ZKoperasi_NN tumbuh_VB dari_IN
     kalangan_NN rakyat_NN ,_Z ketika_SC penderitaan_NN dalam_IN lapangan_NN
    ekonomi_NN dan_CC sosial_JJ yang_SC ditimbulkan_VB oleh_IN sistem_NN
    kapitalisme_NN semakin_RB memuncak_VB ._ZBeberapa_CD orang_NN yang_SC
    penghidupannya_Unk sederhana_JJ dengan_IN kemampuan_NN ekonomi_NN terbatas_JJ ,
    _Z terdorong_VB oleh_IN penderitaan_NN dan_CC beban_NN ekonomi_NN yang_SC
    sama_JJ ,_Z secara_IN spontan_Unk mempersatukan_VB diri_NN untuk_SC menolong_VB
    dirinya_Unk sendiri_NN dan_CC manusia_NN sesamanya_Unk ._Z
```

Listing 2: Wikipedia.nltk.tagged

Kemudian untuk mengevaluasi kinerja *POS tagger*, pengujian dilakukan menggunakan 1000 kalimat yang diekstrak dari korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv` dan *training* dilakukan pada 9030 kalimat sisanya. Perintah yang dijalankan adalah:

```
python tagger.py ../data/pos-tagging/Indonesian_Manually_Tagged_Corpus_ID.tsv 1000
    sentences.tag
```

Hasil *tagging* dicetak pada `sentences.tag` (sesuai argumen eksekusi) dengan format seperti Listing 3. Sedangkan statistik eksekusi program:

```
Accuracy: 91.7577559878 %
Total time: 111.527937 s
```

```
Mengajari_VB anak_NN mengenai_IN kebiasaan_NN yang_SC sehat_JJ memerlukan_VB
    keterlibatan_NN seluruh_CD anggota_NN keluarga_NN anak-anak_NN tak_NEG kan_RP
    mempelajari_VB semua_CD ini_PR dengan_IN sendiri_NN -nya_PRP ,_Z kata_VB -
    nya_PRP ._Z
Untuk_SC kedua_OD kali_NND -nya_PRP batalion_Unk teknik_NN mesin_NN China_NNP
    bertolak_VB ke_IN Lebanon_NNP 22_NNP Januari_NNP untuk_SC bergabung_VB dengan_IN
     misi_NN penjaga_NN perdamaian_NN PBB_NNP di sana_PR ._Z
Batalion_Unk teknik_NN kedua_CD militer_NN China_NNP ini_PR terdiri_VB 275_CD
    personel_NN di_IN samping_NN satu_CD tim_NN medis_JJ yang_SC beranggotakan_VB
    60_CD orang_NN ._Z
```

```
4
5 /* remaining omitted */
```

Listing 3: sentences.tag

2.2. OpenNLP

Program yang dibuat menggunakan API OpenNLP adalah `yohanes.nlp`[5] yang menggunakan `opennlp.tools.postag.POSTaggerME`.
Uji coba pertama adalah melakukan POS*tagging* pada paragraf Listing 1 yang diambil dari Wikipedia Indonesia[6] dengan tagger yang dilatih menggunakan korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv`. Perintah yang dijalankan adalah:

```
1 sh target/appassembler/bin/nlp pos-tag ../../data/pos-tagging/
      Indonesian_Manually_Tagged_Corpus_ID.tsv ../../data/pos-tagging/Wikipedia.txt
```

Hasil dari eksekusi perintah ini adalah file bernama `Wikipedia.opennlp.tagged` yang berisi hasil *tagging* seperti pada Listing 4. Waktu yang dibutuhkan program untuk melakukan *tagging* adalah 70.589 detik. Waktu yang dibutuhkan lebih lama karena proses *training* dengan algoritma *maximum-log-likelihood* dengan 100 iterasi.

```
1 Koperasi_NN di_IN Indonesia_NNP
2 Koperasi_NN di_IN Indonesia_NNP ,_Z menurut_IN UU_NNP tahun_NNP 1992_CD ,_Z
      didefinisikan_VB sebagai_IN badan_NN usaha_NN yang_SC beranggotakan_VB orang_NN -_Z
       seorang_NND atau_CC badan_NN hukum_NN koperasi_NN dengan_IN melandaskan_VB
      kegiatannya_NN berdasarkan_VB prinsip_NN -_Z prinsip_FW koperasi_NN sekaligus_RB
      sebagai_IN gerakan_NN ekonomi_NN rakyat_NN yang_SC berdasar_VB atas_IN asas_NN
      kekeluargaan_NN ._Z
3 Di_IN Indonesia_NNP ,_Z prinsip_NN koperasi_NN telah_MD dicantumkan_VB dalam_IN UU_NNP
      No_NNP ._Z 12_CD Tahun_NN 1967_CD dan_CC UU_NNP No_NNP ._Z 25_CD Tahun_NN 1992_CD .
      _Z
4 Prinsip_NN koperasi_NN di_IN Indonesia_NNP kurang_RB lebih_RB sama_JJ dengan_IN
      prinsip_NN yang_SC diakui_VB dunia_NN internasional_JJ dengan_IN adanya_NN
      sedikit_CD perbedaan_NN ,_Z yaitu_SC adanya_NN penjelasan_NN mengenai_IN SHU_NNP (
      _Z Sisa_NN Hasil_NN Usaha_NN )_Z ._Z
5 Sejarah_NN koperasi_NN di_IN Indonesia_NNP
6 Sejarah_NN singkat_NN gerakan_NN koperasi_NN bermula_VB pada_IN abad_NN ke_IN -_NNP 20
      _CD yang_SC pada_IN umumnya_NN merupakan_VB hasil_NN dari_IN usaha_NN yang_SC
      tidak_NEG spontan_JJ dan_CC tidak_NEG dilakukan_VB oleh_IN orang_NN -_NNP orang_NN
      yang_SC sangat_RB kaya_JJ ._Z
7 Koperasi_NN tumbuh_VB dari_IN kalangan_NN rakyat_NN ,_Z ketika_SC penderitaan_NN
      dalam_IN lapangan_NN ekonomi_NN dan_CC sosial_JJ yang_SC ditimbulkan_VB oleh_IN
      sistem_NN kapitalisme_NN semakin_RB memuncak_VB ._Z
8 Beberapa_CD orang_NN yang_SC penghidupannya_NN sederhana_JJ dengan_IN kemampuan_NN
      ekonomi_NN terbatas_JJ ,_Z terdorong_VB oleh_IN penderitaan_NN dan_CC beban_NN
      ekonomi_NN yang_SC sama_JJ ,_Z secara_IN spontan_NNP mempersatukan_VB diri_NN
      untuk_SC menolong_VB dirinya_NN sendiri_NN dan_CC manusia_NN sesamanya_NN ._Z
```

Listing 4: Wikipedia.opennlp.tagged

[5] `https://github.com/yohanesgultom/nlp-experiments/tree/master/java/nlp`
[6] `https://id.wikipedia.org`

Kemudian untuk mengevaluasi kinerja *POS tagger*, pengujian dilakukan menggunakan korpus `Indonesian_Manually_Tagged_Corpus_ID.tsv` dengan perbandingan 9 (*training*) : 1 (*testing*). Perintah yang dijalankan adalah:

```
sh target/appassembler/bin/nlp pos-tag -split 9:1 ../../data/pos-tagging/
    Indonesian_Manually_Tagged_Corpus_ID.tsv
```

Hasil *tagging* dicetak pada `sentences.tag` (sesuai argumen eksekusi) dengan format seperti Listing 5. Sedangkan statistik eksekusi program:

```
Accuracy: 95.89224%
Total time: 42.179 s
```

```
Kera_NNP untuk_SC amankan_NN pesta_NN olahraga_NN
Pemerintah_NN kota_NN Delhi_NNP mengerahkan_VB monyet_NN untuk_SC mengusir_VB monyet-
    monyet_NN lain_JJ yang_SC berbadan_VB lebih_RB kecil_JJ dari_IN arena_NN Pesta_NN
    Olahraga_NNP Persemakmuran_NNP ._Z
Beberapa_CD laporan_NN menyebutkan_VB setidaknya_RB 10_CD monyet_NN ditempatkan_VB di_IN
    luar_NN arena_NN lomba_NN dan_CC pertandingan_NN di_IN ibukota_NN India_NNP ._Z
Pemkot_NN Delhi_NNP memiliki_VB 28_CD monyet_NN dan_CC berencana_VB mendatangkan_VB 10
    _CD monyet_NN sejenis_NN dari_IN negara_NN bagian_NN Rajasthan_NNP ._Z
Jumlah_NN monyet_NN di_IN ibukota_NN India_NNP mencapai_VB ribuan_CD ,_Z sebagian_NN
    besar_NN berada_VB di_IN kantor-kantor_NN pemerintah_NN dan_CC hewan_NN ini_PR
    dianggap_VB mengganggu_VB ketertiban_NN umum_JJ ._Z

/* remaining omitted */
```

Listing 5: sentences.tag

3. Kesimpulan

Berdasarkan hasil uji coba *tagging* di atas, berikut kesimpulan yang bisa diambil mengenai kedua kakas NLP tersebut:

OpenNLP

1. Relatif lebih akurat
2. Relatif lebih cepat untuk banyak kalimat (> 1000)
3. API yang tersedia kurang fleksibel (contoh: hanya bisa membaca dari `InputStream`)
4. Lebih kompleks untuk digunakan karena berb
5. Dokumentasi kurang lengkap

NLTK

1. Relatif kurang akurat
2. Relatif lebih lambat untuk banyak kalimat tapi lebih cepat untuk sedikit kalimat (< 10)

3. API yang tersedia lebih fleksibel (contoh: bisa membaca dari struktur data `Tree`) dan lengkap (terdapat banyak jenis implementasi *tagger*)

4. Lebih mudah untuk digunakan (karena berbasis Python)

5. Dokumentasi lebih lengkap

Selain itu, mengenai *task POS Tagging* secara keseluruhan ada beberapa yang perlu diperhatikan:

1. Hasil *parsing* korpus harus selalu diuji dengan *unit test* yang memadai untuk memastikan semua kasus teridentifikasi dan tertangani. Contoh: adanya terminal yang dipisahkan spasi (pesta olahraga), adanya id kalimat yang mengandung alfanumerik (14a, 14b .dst)

2. Kakas yang berbasis Java umumnya akan lebih cepat dalam melakukan *POS Tagging* dari bahasa lain seperti Python karena proses kompilasi yang lebih mangkus dan optimasi internal pada JVM. Kelebihan kakas berbasis Python adalah kemudahan sintaks bahasa, dokumentasi dan API yang lebih banyak karena lebih popular sebagai kakas NLP

II. Probabilistic Parsing

Untuk kasus *Probabilistik Parsing*, hanya NLTK yang digunakan karena OpenNLP dan Stanford NLP sulit digunakan untuk membangun *parser* yang model bahasa nya belum disediakan.

1. Deskripsi Penanganan Format

Format yang digunakan pada korpus `150324.001-300.bracket` adalah:

```
(NP (NN (Kera)) (SBAR (SC (untuk)) (S (NP-SBJ (*)) (VP (VB (amankan)) (NP (NN (pesta olahraga)))))))
(S (NP-SBJ (NNP (Pemerintah)) (NNP (kota)) (NNP (Delhi))) (VP (VB (mengerahkan)) (NP (NN (monyet))) (SBAR (SC (untuk)) (S (NP-SBJ (*)) (VP (VB (mengusir)) (NP (NP (NN (monyet-monyet)) (JJ (lain))) (SBAR (SC (yang)) (S (NP-SBJ (*)) (VP (VB (berbadan)) (ADJP (RB (lebih)) (JJ (kecil))))))) (PP (IN (dari)) (NP (NN (arena)) (NP (NNP (Pesta Olahraga)) (NNP (Persemakmuran))))))))) (Z (.)))
```

Pada format ini jika dilihat sekilas sepertinya tiap *parse tree* kalimat dipisahkan oleh baris baru tapi ternyata dalam satu baris bisa terdapat beberapa kalimat (memiliki lebih dari 1 root).
Untuk NLTK, API yang ada untuk melakukan *training parsing* menerima `list of nltk.Tree` yang memiliki representasi seperti:

```
(S
  (NP-SBJ (NN Kera))
  (SBAR-PRD
    (SC untuk)
    (S
      (NP-SBJ *)
      (VP (VB amankan) (NP (NN pesta) (NN olahraga))))))
```

Kode fungsi yang digunakan untuk mengkonversi korpus ke format tersebut dideskripsikan Listing 6.

```
def corpus2trees(text):
  """ Parse the corpus and return a list of Trees """
  rawparses = text.split('\n')
  trees = []
  for rp in rawparses:
    if not rp.strip():
      continue

    try:
      for s in split_tree(rp):
        s = remove_terminal_tree_format(s)
        t = Tree.fromstring(s)
        trees.append(t)
    except ValueError:
      logging.error('Malformed parse: "%s"' % rp)
```

```
16
17   return trees
```

Listing 6: corpus2trees.tag

Seperti yang dijelaskan di atas, setiap baris bisa berisi lebih dari satu *parse tree* (root S) jadi ketika di-*parse*, 300 baris korpus menghasilkan 321 *parse tree*.

2. Hasil Uji Coba dan Analisis

Program yang dibuat menggunakan API NLTK adalah `parser.py`[7] yang menggunakan `nltk.ViterbiParser` yang diekstensi untuk melakukan *smoothing* sederhana untuk terminal yang tidak ada pada *grammar*.
Uji coba pertama adalah melakukan *parsing* pada paragraf Listing 1 yang juga digunakan pada uji coba *POS tagging*. Perintah yang dijalankan adalah:

```
python parser.py ../data/prob-parsing/150324.001-300.bracket ../data/prob-parsing/
    Wikipedia.txt
```

Hasil dari eksekusi perintah ini seharusnya adalah file bernama `Wikipedia.nltk.bracket`. Tetapi pada pengujian ini, program gagal menemukan *parse tree* yang cocok untuk kalimat-kalimat pada Listing 1 sekalipun sudah digunakan teknis *smoothing*. Kejadian ini kemungkinan besar terjadi karena tidak ada *production rules* yang sesuai dari hasil *training* dengan `150324.001-300.bracket`.
Uji coba yang kedua adalah melakukan *parsing* terhadap 50 *parse tree* yang diekstraksi dari korpus `150324.001-300.bracket` setelah sebelumnya melakukan training dengan 271 *parse tree* sisanya. Perintah yang digunakan adalah:

```
python parser.py ../data/prob-parsing/150324.001-300.bracket 50 sentences.bracket
```

Hasil *parsing* dicetak pada `sentences.bracket` (sesuai argumen eksekusi) dengan format seperti Listing 7. Sedangkan statistik eksekusi program:

```
Accuracy: 90.0 %
Total time: 1213.776889 s
```

Akurasi baru dihitung dengan melihat berapa jumlah *parse tree* yang ditemukan (tidak dibandingkan dengan korpus).

```
(S
  (NP-SBJ (NN Kera))
  (SBAR-PRD
    (SC untuk)
    (S
      (NP-SBJ *)
      (VP (VB amankan) (NP (NN pesta) (NN olahraga))))))) (p=4.33627e-29)
(S
  (S
    (NP-SBJ (NN Monyet) (PR ini))
    (VP
      (VB diikat)
```

[7] `https://github.com/yohanesgultom/nlp-experiments/blob/master/python/parser.py`

```
13        (NP *-1)
14        (PP (IN dengan) (NP (NN tali) (JJ panjang)))))
15    (CC dan)
16    (S
17      (NP-SBJ
18        (NP
19          (NN pelatih)
20          (SBAR
21            (SC yang)
22            (S
23              (NP-SBJ *)
24              (VP
25                (VB mengawasi)
26                (S
27                  (NP-SBJ (PRP mereka))
28                  (VP
29                    (MD akan)
30                    (VP
31                      (VB melepas)
32                      (NP (NN tali))
33                      (ADVP (RB begitu)))))))))
34      (NP (NN monyet-monyet) (JJ kecil)))
35    (VP (ADJP (JJ lain)) (VB mendekat)))
36  (Z .)) (p=9.05711e-86)
37
38  /* remaining omitted */
```

Listing 7: sentences.bracket

3. Kesimpulan

Kesimpulan yang bisa diperoleh dari uji coba ini adalah:

1. NLTK menyediakan API yang fleksibel untuk melakukan *parsing* dengan banyak algoritma
2. Untuk memperoleh hasil *parsing* dibutuhkan korpus yang menghasilkan variasi *production rules* / *PCFG* yang cukup
3. Proses parsing dengan NLTK cukup lambat (1200 detik untuk 50 kalimat) jadi perlu dicoba alternatif lainnya
4. Sebelum memlih kakas NLP, perlu dilakukan penelaahan terhadap dokumentasi fungsi yang ditawarkan. Karena, sebagai contoh, kakas OpenNLP dan Stanford NLP tidak memiliki API yang cukup fleksibel untuk melakukan *probability parsing* dalam bahasa selain Inggris