Sistem Pengubah Kalimat Bahasa Indonesia ke dalam Bentuk Parse Tree Menggunakan Metode Bottom-Up Parsing

Tubagus Rifky Fajar Kusumah Sakti[1] , Moch Arif Bijaksana[2], Anisa Herdiani[3]

[1,2,3] Fakultas Informatika, Universitas Telkom, Bandung
[1]teberifky@students.telkomuniversity.ac.id, [2]arifbijaksana@telkomuniversity.ac.id,
[3]anisaherdiani@telkomuniversity.ac.id

Abstrak

Natural Language Processing (NLP) adalah area dari penelitian interdisipliner yang memungkinkan komputer untuk berkomunikasi dengan manusia dengan cara yang sama seperti bagaimana manusia berkomunikasi dengan manusia lain. Salah satu tahapan dalam Natural Language Processing (NLP) adalah parsing. Parsing telah berkembang pesat meliputi cakupan yang luas, akurasi yang tinggi, dan efisiensi dalam beberapa periode terakhir. Banyak sekali penelitian sebelumnya yang telah berhasil mengubah sebuah kalimat dari berbagai bahasa ke dalam bentuk parse tree, tapi hanya sedikit untuk Bahasa Indonesia. Dalam rangka sarana pendukung berbagai aplikasi pengolah bahasa guna meningkatkan penelitian di bidang Natural Language Processing (NLP), maka parser Bahasa Indonesia diperlukan dan perlu dikembangkan keberadaannya. Pada penelitian tugas akhir ini, dibangun sebuah sistem untuk mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree dengan menggunakan metode Bottom-Up Parsing. Bentuk parse tree yang dihasilkan sistem ini dievaluasi pengujian performansi dengan menggunakan PARSEVAL metrics. Dengan total hasil pengujian didapatkan precision dan recall sebesar 40,94% dan 72,26%.

Kata kunci : natural language processing, parsing, parse tree, bottom-up parsing, PARSEVAL.

Abstract

Natural Language Processing is an area of interdisciplinary research that enables computers to communicate with humans in the same way as humans communicate with other humans. One of the stages in Natural Language Processing is parsing. Parsing has developed rapidly covering wide, high accuracy, and efficiency in the last few periods. Many previous studies have succeeded in converting a sentence from various languages into the form of a parse tree, but only a little for Indonesian. In order to support various language processing applications and in order to enhance research in the field of Natural Language Processing, an Indonesian parser is needed and its existence needs to be developed. In this final project research, a system will be built to convert Indonesian sentences into parse tree form using the Bottom-Up Parsing method. The form of the parse tree produced by this system will be evaluated for performance testing using PARSEVAL metrics. With the total performance testing results obtained precision and recall of 40,94% and 71,59%.

Keywords: natural language processing, parsing, parse tree, bottom-up parsing, PARSEVAL.

1. Pendahuluan

Latar Belakang

Dalam beberapa periode terakhir, parsing telah berkembang pesat meliputi cakupan yang luas, akurasi yang tinggi, dan efisiensi dalam melakukan parsing [1]. Parsing memiliki beberapa aplikasi, diantaranya adalah question answering, information extraction, semantic disambiguation, machine translation, dan speech synthesis [2]. Terdapat 2 (dua) teknik utama dalam parsing, yaitu Rule-based dan Statistical Parsing [3] [4]. Natural Language Processing (NLP) adalah area dari penelitian interdisipliner dengan tujuan untuk mengembangkan program komputer yang dapat menghasilkan teks dalam bahasa alami [5], yang memungkinkan komputer untuk berkomunikasi dengan manusia dengan cara yang sama seperti bagaimana manusia berkomunikasi dengan manusia lain [6]. Natural Language Processing (NLP) melakukan prosesnya dengan beberapa tahapan dan diimplementasikan kedalam suatu aplikasi. Beberapa tahapan tersebut antara lain, yaitu parsing yang memeriksa kebenaran suatu struktur kalimat berdasarkan grammars, part of speech tagger yang memberi penanda POS untuk tiap kata di dalam korpus, named entity recognizer yang mendeteksi dan mengklasifikasikan named-entity pada suatu teks, context-independent yang merepresentasikan arti dari kalimat secara context-independent, dan contextual interpretation yang untuk merepresentasikan arti secara context-dependent [7].

Parse tree adalah pohon yang mewakili sintaksis struktur kalimat sehubungan dengan tata bahasa formal, yang dapat dihasilkan secara otomatis oleh parser [6]. Diberikan kalimat gramatikal yang benar, parser menghasilkan parse tree yang sesuai dengan aturan produksi yang ada pada grammars. Parser begitu penting dalam Natural Language Processing (NLP) yang berguna untuk memahami arti dari bahasa manusia dengan menentukan grammar dari bahasa tersebut. Parser membentuk parse tree dari sebuah kalimat, dan bisa diketahui ide utama dari kalimat tersebut [8]. Berbagai aplikasi menggunakan parser sebagai sarana pendukung untuk menemukan cara kalimat itu dibentuk dengan pendekatan tata bahasa yang benar dan melibatkan pengetahuan linguistik [9].

Banyak sekali penelitian sebelumnya yang telah berhasil mengubah sebuah kalimat dari berbagai bahasa ke dalam bentuk parse tree, sebagai contohnya adalah Maad Shatnawi dan Boumediene Belkhouche yang berhasil mengubah kalimat bahasa Arab ke dalam bentuk parse tree dengan menggunakan Top-Down (Recursive Descent) Parsing [2] dan Weiwei Sun, Yufei Chen, Xiaojun Wan, dan Meichun Liu yang berhasil mengubah kalimat bahasa Chinese ke dalam bentuk parse tree dengan Grammatical Relations [10]. Untuk Bahasa Indonesia sendiri, hanya sedikit yang melakukan penelitian terhadap topik ini. Sebagai contoh, terdapat 2 (dua) penelitian yang berhasil mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree dengan menggunakan Collins's Parser [6] [11]. Namun 2 (dua) penelitian tersebut tidak dilakukan pengujian performansi, yang berfungsi untuk mengetahui performansi dari pembentukan parse tree yang berhasil dibuat oleh sistem tersebut.

Pada penelitian tugas akhir ini, dibangun sebuah sistem untuk mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree dengan menggunakan metode Bottom-Up Parsing. Bottom-Up Parsing adalah sebuah metode parsing berbasis rule [12] dengan membentuk parse tree dari leaf (bottom) lalu ke root (akar) [13]. Cara kerja metode ini begitu sederhana, namun performa yang dihasilkan begitu baik dan efisien. Sistem ini melakukan proses parsing dengan berdasarkan pada grammars yang didapatkan dari proses ekstraksi treebank, lalu menghasilkan output berupa parse tree dengan bentuk constituency tree. Bentuk parse tree yang dihasilkan sistem ini dievaluasi pengujian performansi dengan menggunakan PARSEVAL metrics.

Topik dan Batasannya

Parsing merupakan langkah penting dalam beberapa aplikasi yang melibatkan analisis dokumen, diantaranya adalah question answering, information extraction, semantic disambiguation, summarization, dan filtering [2]. Parser membentuk parse tree dari sebuah kalimat, dan bisa diketahui ide utama dari kalimat tersebut [8]. Berbagai aplikasi menggunakan parser sebagai sarana pendukung untuk menemukan cara kalimat itu dibentuk dengan pendekatan tata bahasa yang benar dan melibatkan pengetahuan linguistik [9].

Dalam rangka sarana pendukung berbagai aplikasi pengolah bahasa guna meningkatkan penelitian di bidang Natural Language Processing (NLP), maka parser bahasa Indonesia diperlukan dan perlu dikembangkan keberadaannya. Untuk memperkecil cakupan dari tugas akhir ini diperlukan batasan-batasan. Adapun batasan masalah yang diambil untuk tugas akhir ini adalah sebagai berikut:

- Output sistem hanya menampilkan 1 (satu) bentuk parse tree paling atas (dengan kemungkinan paling benar), jika suatu kalimat menghasilkan lebih dari 1 (satu) parse tree.
- Hanya bisa melakukan parsing jika kata (lexicon) dalam kalimat terdapat pada grammars. Jika kata (lexicon) dalam kalimat tidak terdapat pada grammars, sistem menghasilkan output berupa message tidak dapat melakukan parsing karena kata pada kalimat tidak ada pada grammars.

Tujuan

Tujuan dibuatnya sistem ini adalah untuk menghasilkan bentuk parse tree dari sebuah kalimat Bahasa Indonesia menggunakan metode Bottom-Up Parsing, kemudian mengevaluasi bentuk parse tree yang telah dibuat oleh sistem dengan menghitung nilai precision, recall, dan f1 score menggunakan pengukuran PARSEVAL metrics.

Organisasi Tulisan

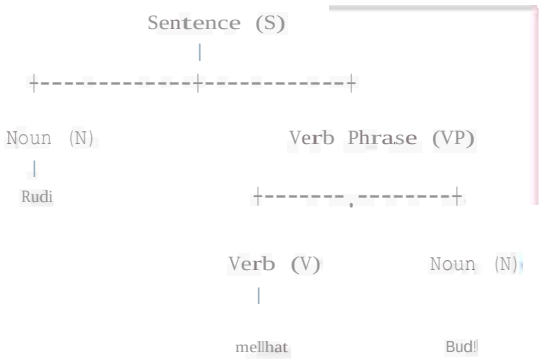
Penulisan tugas akhir ini dibagi menjadi 5 (lima) bagian, yaitu:

1. Bagian 1 berisi latar belakang, topik dan batasannya, tujuan, dan organisasi tulisan.
2. Bagian 2 berisi topik-topik terkait dengan tugas akhir ini.
3. Bagian 3 berisi alur sistem yang dibangun pada tugas akhir ini.
4. Bagian 4 berisi hasil pengujian sistem yang sudah dibangun beserta analisis terkait hasil yang sudah ada.
5. Bagian 5 berisi kesimpulan dari seluruh bagian penelitian tugas akhir yang sudah dilakukan. Terdapat saran untuk mengembangkan sistem ini di kemudian hari.

2. Studi Terkait

2.1 Constituency Parsing

Parsing merupakan langkah penting dalam beberapa aplikasi yang melibatkan analisis dokumen, diantaranya adalah question answering, information extraction, semantic disambiguation, summarization, dan filtering [2]. Constituency Parsing merupakan pendekatan yang digunakan untuk melakukan analisis sintaksis otomatis bahasa alami dengan berdasarkan pada grammars yang berbentuk constituency tree. Tujuan dari Constituency Parsing adalah untuk mengekstrak parse tree berbasis konstituensi dari kalimat yang mewakili sintaksisnya menurut phrase structure grammar [14]. Contoh bentuk contituency tree dapat dilihat pada Gambar 1.

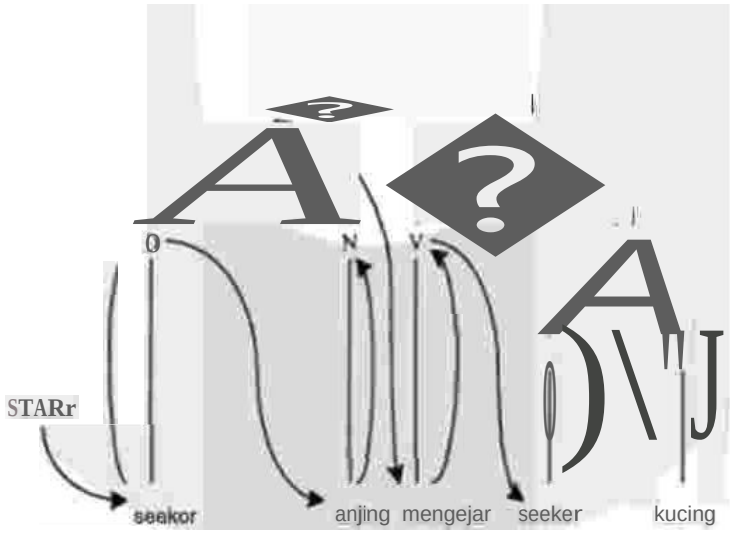


Gambar 1. Constituency Tree [14]

Pendekatan terbaru dengan mengubah bentuk parse tree menjadi sebuah urutan yang mengikuti depth-first traversal, pendekatan tersebut dilakukan agar dapat menerapkan model sequence-to-sequence [14]. Versi linier dari bentuk parse tree pada Gambar 1 adalah sebagai berikut: (S (N) (VP (V) (N))).

2.2 Bottom-Up Parsing

Bottom-Up Parsing adalah sebuah metode parsing berbasis rule [12] dengan membentuk parse tree dari leaf (bottom) lalu ke root (akar). Metode ini melakukan proses parsing dengan berdasarkan pada grammars yang didapatkan dari proses ekstraksi treebank, lalu menghasilkan parse tree dengan bentuk constituency tree. Parser ini melakukan proses parsing dengan cara mengambil satu demi satu kata dari sebuah kalimat secara terus-menerus, yang nantinya dirangkai menjadi sebuah constituent tree [13]. Cara kerja parser ini dapat dilihat pada Gambar 2.



Gambar 2. Cara Kerja Bottom-Up Parsing [13]

Parser ini tidak membedakan antara rule (grammar) dan word (lexicon) sehingga cara kerjanya sederhana, namun performa yang dihasilkan begitu baik dan efisien. Kesederhanaan metode ini terletak pada predikat untuk

parsing, yaitu parse yang hanya memiliki sebuah argumen. Argumen ini berisi kalimat yang di parse dalam bentuk list dari symbol. Kata yang terdapat pada kalimat di rangkai sembari mencari aturan yang lebih luas, sampai pada akhirnya hanya tinggal sebuah simbol saja dalam list tersebut, yaitu ”S” [13].

2.3 Context-free Grammar (CFG)

Context-free Grammar (CFG) adalah jenis formal bahasa tertentu yang digunakan untuk menggambarkan aturan dari suatu objek. Terdapat 4 (empat) atribut pada context-free grammar yang dapat menggambarkan suatu pola. Atribut tersebut adalah Start (S), kumpulan non-terminal (V), kumpulan terminal (Σ), hubungan antara V dan V/Σ [15] [16] [17]. Persamaan 1 merupakan aturan dalam CFG [18] [19].

$$G = (V, \Sigma, R, S) \tag{1}$$

2.3.1 Grammars

Grammars adalah aturan-aturan pembentukan suatu kalimat yang biasa disebut tata bahasa [13]. Proses parsing dengan membentuk struktur kalimat mengikuti suatu aturan produksi yang ada pada grammars. Beberapa tags pada grammars yang digunakan pada tugas akhir ini, dapat dilihat pada Tabel 1.

Tabel 1. Penn Treebank II Constituency Tags [20]

Tag	Meaning	Tag	Meaning
S	simple declarative clause	QP	quantifier phrase
SBAR	clause	UCP	unlike coordinate phrase
SBARQ	direct question	VP	verb phrase
SINV	inverted yes/no question	X	unknown
ADJ	adjective phrase	-ADV	marks a constituent
ADVP	adverb phrase	-NOM	marks free (”headless”) relatives
CONJP	conjunction phrase	-LGS	mark the logical subject in passives
FRAG	fragment	-PRD	marks any predicate that is not VP
INTJ	interjection	-SBJ	marks the structural surface subject
NAC	not a constituent	-TPC	marks elements that appear before the subject
NP	noun phrase	-VOC	marks nouns of address
PP	prepositional phrase	-TMP	marks temporal or aspectual adverbials
PRN	parenthetical	-TTL	attached to the top node of a title

2.4 Treebank Indonesia

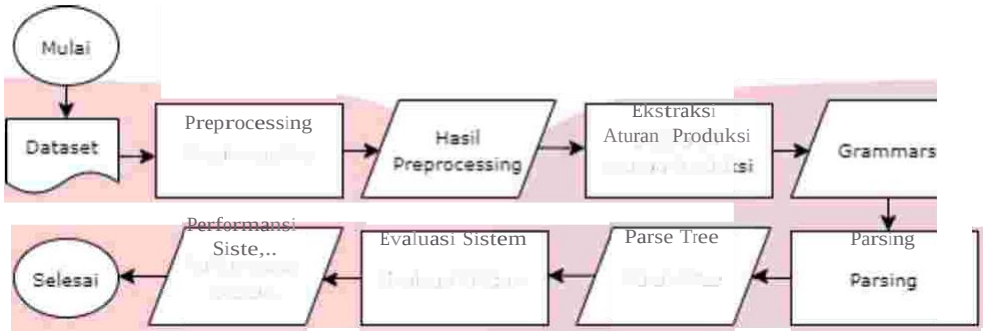
Treebank adalah kumpulan dari sintaksis kalimat yang diperiksa secara manual. Treebank merupakan hal yang penting dalam sumber daya untuk membangun dan mengevaluasi parser, dan digunakan dalam beberapa aplikasi seperti part-of-speech (POS) tagging, morphological disambiguation, chunk parsing, dan tokenizing [21]. Treebank Indonesia adalah korpus teks yang terdiri atas kalimat yang telah diurai dan dianotasikan berdasarkan proyek the Penn Treebank sebagai dasar dalam perancangan pedoman anotasi. Terdapat perbedaan dalam struktur kalimat Bahasa Indonesia dengan Bahasa Inggris, sehingga perlu penyesuaian. Perumusan struktur kalimat dalam pedoman anotasi Treebank Indonesia menggunakan label kategori sintaktis dan tag fungsi the Penn Treebank [20].

2.5 PARSEVAL Metrics

PARSEVAL metrics masih menjadi standar pengukuran evaluasi untuk kualitas parser. PARSEVAL memeriksa label dan identitas wordspan pada parse tree hasil output sistem yang dibandingkan dengan parse tree gold standard [22]. PARSEVAL metrics menghitung precision dan recall dengan melihat konstituen yang ada pada parse tree. Konstituen dalam parse tree hasil output sistem benar jika terdapat konstituen pada parse tree gold standard yang sesui dengan urutan simbol terminal yang sama dan memiliki label yang sama. PARSEVAL metrics tidak menghitung accuracy dari konstituen pre-terminal [23]. Terdapat 2 (dua) pengukuran PARSEVAL, yaitu precision dan recall. Precision dan recall pada dasarnya merupakan cara pengukuran yang sama dengan information retrieval. Precision adalah jumlah konstituen yang benar dalam parse tree yang dihasilkan oleh sistem. Konstituen dianggap benar jika konstituen tersebut sama dengan parse tree gold standard. Recall adalah jumlah konstituen

yang benar dalam parse tree yang dihasilkan oleh sistem dibandingkan dengan jumlah konstituen pada parse tree gold standard. Terdapat juga f1 score, yaitu harmonik mean dari precision dan recall [22].

3. Sistem yang Dibangun



Gambar 3. Gambaran Umum Sistem

Gambaran umum sistem yang dibuat dapat dilihat pada Gambar 3. Dataset melalui tahap preprocessing terlebih dahulu. Pada tahap preprocessing, dilakukan 3 (tiga) proses, yaitu cleaning data, case folding, dan pengambilan standardized treebank pada dataset. Kemudian setelah tahap preprocessing selesai, dilanjutkan dengan tahap ekstraksi aturan produksi. Pada tahap ekstraksi aturan produksi, seluruh standardized treebank yang telah didapatkan pada tahap preprocessing dijadikan aturan produksi dan diubah kedalam bentuk CFG (context-free grammar) yang menghasilkan sebuah grammars. Di dalam grammars ini menghasilkan non-terminal dan terminal. Setelah didapatkan grammars dari tahap ektraksi aturan produksi, dilakukan tahap parsing menggunakan metode Bottom-Up Parsing. Selanjutnya pada proses parsing menghasilkan output dengan bentuk parse tree. Selanjutnya dilakukan evaluasi sistem dengan menggunakan PARSEVAL Metrics.

Dataset

Dataset yang digunakan adalah dataset berisi treebank dari kalimat Bahasa Indonesia yang diambil dari Korpus Treebank Bahasa Indonesia Universitas Indonesia [24], dengan format BRACKET. Korpus Treebank Bahasa Indonesia Universitas Indonesia merupakan korpus yang telah diurai secara manual oleh manusia, dengan data kalimat yang didapatkan dari PAN Localization [25]. Dataset ini total berjumlah 1,030 treebank dari kalimat Bahasa Indonesia, dengan kata berjumlah 28,117. Pada proses pengujian performansi diambil data dengan total 500 treebank untuk train set dan total 50 treebank yang diambil secara manual untuk test set. Gambar 4 merupakan contoh dataset yang digunakan pada tugas akhir ini.

```
. (S (NP-SBJ (NNP\t (Pemerintah)) (tlNP\t (kota)) (NNP\t (Delhi))) (VP (VB\t
(mengerahkan)) (NP (NN\t (llonyet))) (SBAR (SC\t (untuk)) (S (tlP-SBJ (•)) (VP
(VB\t (llengusir)) (NP (NP (NN\t (llonyet-monyet)) (JJ\t (lain))) (SBAR (SC\t (y
ang)) (S (tlP-SBJ (*)) (VP (VB\t (berbadan)) (A0JP (RB\t (lebih)) (JJ\t (keci
1)))))))) (PP (IN\t (dari)) (NP (NN\t (arena)) (NP (NllP (Pesta Olahraga)) (NtlP\t (Per
semakmuran))))))))) (Z\t (.)))\n", '(S (NP-SBJ (CO\t (Beberapa)) (llll\t (laporan))) (VP (VB\t
(menyebutkan)) (SBAR (0) (S (NP-SBJ-1 (QP (RB\t (setidaknya)) (CO\t (10))) (Nll\t (monyet)))
(VP (VB\t (ditempatkan)) (HP ("-1)) (PP (!N\t (di)) (NP (llN\t (luar)) (NP (tlP (llN\t
(arena)) (NP (NN\t (lomba)) (((\t (dan)) (NN\t (pertandingan)))) (PP (ltl\t (di)) (NP (NNP
```

Gambar 4. Contoh Dataset

Preprocessing

Pada tahap preprocessing dilakukan cleaning data, case folding, dan pengambilan standardized treebank pada dataset. Preprocessing penting untuk dilakukan agar data yang diolah menjadi lebih terstruktur dan memastikan dataset siap untuk diproses [26].

1. Cleaning Data

   Berdasarkan pada Gambar 4, dapat dilihat bahwasanya treebank yang ada pada dataset mengandung karakter-karakter yang harus dihilangkan diantaranya yang dapat dilihat pada Gambar 4 adalah karakter ”\t” (escape character untuk tabs), dan whitespace yang ada pada dataset belum terstruktur dengan baik. Maka dari itu, pada tahap ini dilakukan proses cleaning data pada whitespace yang ada agar dataset siap untuk diproses. Selain karakter tersebut, beberapa aturan produksi non-terminal yang ada pada dataset tidak sesuai dengan bracketing guidelines [20] untuk treebank seperti ”NP=2” yang dirubah ke ”NP”.

Gambar 5. Contoh Dataset Setelah Cleaning Data

Terdapat perbedaan dari Gambar 4 dan Gambar 5 yang sudah dilakukan proses cleaning data. Sebagai contoh, pada Gambar 5 karakter ”\t” sudah dihilangkan dan whitespace menjadi lebih terstruktur.

Tabel 2. Perbedaan Setelah Cleaning Data

Treebank
(S (NP-SBJ (NN (Pemerintah)) (NN (kota))))....
Treebank (Cleaning Data)
(S (NP-SBJ (NN Pemerintah) (NN kota)))....

Perbedaan lainnya seperti yang dapat dilihat pada Tabel 2, (NP-SBJ (NN (Pemerintah)) (NN (kota)) diubah menjadi (NP-SBJ (NN Pemerintah) (NN kota)). Hal ini dikarenakan nantinya saat pembuatan grammars, ”Pemerintah” dan ”kota” dianggap sebagai non-terminal jika proses cleaning data tidak dilakukan. Bentuk setelah dilakukan proses cleaning data ini sama dengan output sistem yang dibangun. Dilakukannya proses cleaning data ini memudahkan pada saat tahap ekstraksi aturan produksi.

2. Case Folding

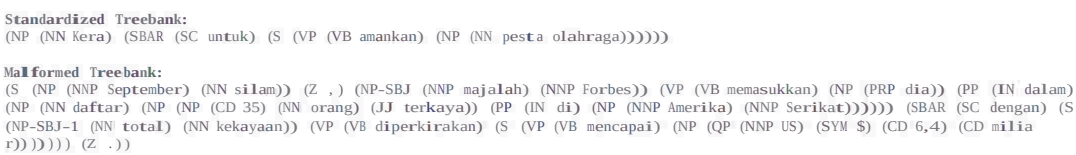
Tujuan dari proses case folding adalah untuk mengubah semua terminal (kata/lexicon) yang ada di dalam dataset menjadi lower case [27]. Hanya huruf ”A” sampai ”Z” yang diterima. Contoh kasus dari diperlukannya proses case folding ini, yaitu ”Kota” dan ”kota” dianggap sebagai 2 (dua) kata yang berbeda. Tabel 3 merupakan contoh treebank pada dataset sebelum dan setelah dilakukannya proses case folding.

Tabel 3. Contoh Proses Case Folding

Sebelum Case Folding
(S (NP-SBJ (NN Pemerintah) (NN kota)))....
Setelah Case Folding
(S (NP-SBJ (NP pemerintah) (NN kota)))....

3.Pengambilan Standardized Treebank

Pada proses ini, treebank yang ada pada dataset dibagi ke dalam 2 (dua) bagian, yaitu standardized treebank dan malformed treebank. Standardized treebank adalah treebank dengan format yang benar. Malformed treebank adalah treebank dengan format yang salah, seperti contohnya kurangnya tanda kurung tutup ”)” pada treebank tersebut. Treebank tersebut tidak dapat dijadikan aturan produksi, karena menyebabkan exception error. Gambar 6 merupakan output dari proses pengambilan standardized treebank.



Gambar 6. Contoh Standardized Treebank dan Abandoned Treebank

Dapat dilihat pada Gambar 6, dimana pada contoh malformed treebank seharusnya ada 2 (dua) tanda kurung tutup ”)” di akhir treebank tersebut. Maka dari itu, tidak semua aturan produksi dapat diambil pada dataset. Untuk menghindari exception error dalam mendapatkan grammars, malformed treebank dihilangkan dan hanya standardized treebank yang digunakan sebagai aturan produksi.

Ekstraksi Aturan Produksi

Pada tahap ini dilakukan ekstraksi aturan produksi dari seluruh standardized treebank ke dalam bentuk CFG (context-free grammars), dan menghasilkan sebuah grammars. Grammars dapat dihasilkan dengan membaca aturan produksi dari sebuah treebank [5]. Dalam melakukan proses parsing dari sebuah kalimat ke dalam bentuk parse tree, dibutuhkan sebuah grammars. Parse tree dapat dibentuk berdasarkan pada sebuah grammars dalam bentuk CFG (context-free grammars). Contoh grammars dapat dilihat pada Gambar 7.

```
Grammar with 5820 productions (start state = S)
    S -> NP-SBJ-1 PP VP
    S -> NP-SBJ PP-PRO
    S -> NP-1 VP
    S -> NP-SBJ-1 VP
    S -> PP Z PP NP-SBJ-1 VP Z
    S -> S Z S Z S CC S Z
    S -> NP-SBJ PP NP-PRO
    S -> NP-SBJ SBAR-PRO
    S -> SBAR Z PP Z NP-SBJ VP Z
    S -> S S
    S -> NP-SBJ AOVP VP
    S -> PP PP NP-SBJ-1 VP
    S -> PP NP-SBJ-1 VP Z
    S -> CC NP-SBJ PP PP-PRO Z
    S -> NP-SBJ-1 UCP-PRO
    S -> pp S CC S Z
    S -> NP-SBJ UCP-PRO
    S -> S NP-SBJ AOJP-PRO Z
```

Gambar 7. Contoh Grammars

Seperti yang dapat dilihat pada Gambar 7, grammars berhasil dibuat. Grammars tersebut menghasilkan 5820 aturan produksi yang didapat dari standardized treebank yang ada pada dataset.

```
[1619] PP -> IN NP
[1224] , -> ','
[847] Z -> '.'
[837] S -> VP
[711] SBAR -> SC S
[616] VP -> VB NP
[546] NP -> NP NP
[540] NP -> NN
[530] NP -> NN NN
[498] NP -> NP PP
```

Gambar 8. 10 Aturan Produksi (grammars) Dengan Kemunculan Terbanyak

Gambar 8 menunjukan statistik banyaknya kemunculan 10 grammars yang sama beberapa kali. Aturan produksi ”PP → IN NP” adalah aturan produksi yang paling banyak muncul dengan total kemunculan sebanyak 1619 kali. Statistik ini diambil berdasarkan stardardized treebank yang sudah diekstraksi menjadi sebuah grammars.

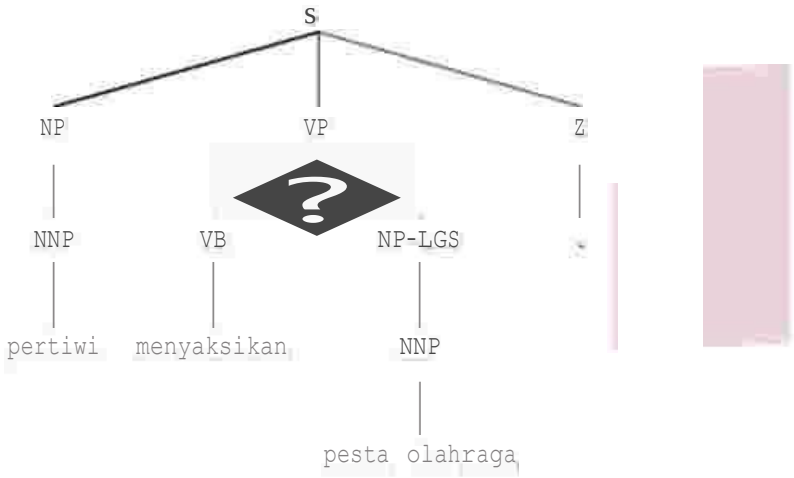
Parsing

Tahap terakhir dari sistem ini adalah melakukan parsing. Menggunakan metode Bottom-Up Parsing, sistem mengubah input berupa kalimat Bahasa Indonesia ke dalam bentuk Parse Tree. Pada tahap sebelumnya, telah dilakukan ekstraksi aturan produksi yang menghasilkan sebuah grammars. Bottom-Up Parsing menggunakan grammars untuk membuat parse tree dari sebuah kalimat, dengan dimulai dari leaf (bawah) lalu ke root (atas). Hasil parsing menggunakan input berupa kalimat Bahasa Indonesia dapat dilihat pada Gambar 9.

```
Pertiwi menyaksikan pesta olahraga .
(S
  (NP (NNP pertiwi))
  (VP (VB menyaksikan) (NP-LGS (NNP pesta olahraga)))
  (Z .))
```

Gambar 9. Hasil Parsing

Gambar 9 menunjukan input dari sistem berupa kalimat Bahasa Indonesia dan output dari sistem berupa bentuk parse tree. Setiap kata pada kalimat tersebut terdapat di dalam grammars. Sistem yang dibuat hanya bisa melakukan parsing jika setiap tag dan kata (lexicon) pada kalimat terdapat di dalam grammars. Jika setiap kata atau salah satu kata pada kalimat tidak terdapat di dalam grammars, sistem menampilkan message. Visualisasi dari bentuk parse tree tersebut dapat dilihat pada Gambar 10.



Gambar 10. Hasil Parsing (Tree Visualization)

Gambar 10 menghasilkan tree dengan bentuk constituent tree. Hasil parse tree yang didapat pada Gambar 9 dapat divisualisasikan. Terdapat bentuk lain dari Gambar 9 dan Gambar 10 adalah bentuk CFG (context-free grammar). Bentuk CFG (context-free grammar) dari hasil parse tree yang didapat dapat dilihat pada Gambar 11.

```
S -> NP VP Z
NP -> NNP
NNP -> 'pertiwi' VP
-> VB NP-LGS VB ->
'menyaksikan' NP-LGS
-> NNP
NNP -> 'pesta' 'olahraga'
Z -> '.'
```

Gambar 11. Hasil Parsing (CFG)

4. Evaluasi

4.1 Hasil Pengujian

Untuk menguji sistem dalam mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree, dilakukan dengan total 500 data (450 untuk train set , 50 untuk test set). Terdapat 2 (dua) kriteria dalam pengujian ini, yaitu Berhasil saat sistem berhasil mengubah kalimat ke dalam bentuk parse tree dan Tidak Berhasil saat sistem tidak berhasil mengubah kalimat ke dalam bentuk parse tree. Hasil dari pengujian dengan menghitung berapa banyak sistem berhasil mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree dapat dilihat pada Tabel 4.

Tabel 4. Hasil Pengujian Dengan 500 data

Test Set	Berhasil	Tidak Berhasil
50	42	8

Hasil yang diperoleh didapatkan persentase keberhasilan sistem dalam mengubah kalimat ke dalam bentuk parse tree sebesar 84%, dimana persentase tersebut didapat dengan cara membagi jumlah sistem berhasil mengubah kalimat ke dalam bentuk parse tree (42) dengan jumlah kalimat yang diuji (50) lalu dikali 100%.

Pengujian selanjutnya menggunakan PARSEVAL metrics untuk mengihitung nilai precision, nilai recall, dan nilai f1 score. Rumus untuk menghitung nilai precision, recall, dan f1 score adalah sebagai berikut [28] [29] [30]:

$$\text{Precision} = \frac{\text{count(matching constituents)}}{\text{count(predicted constituents)}} \times 100\% \quad (2)$$

$$\text{Recall} = \frac{\text{count(matching constituents)}}{\text{count(gold standard constituents)}} \times 100\% \quad (3)$$

$$\text{F1 Score} = \frac{\text{precision x recall}}{(\text{precision} + \text{recall}) / 2} \quad (4)$$

Untuk mengevaluasi sistem yang dibuat, terdapat 5 (lima) skenario pengujian yang dilakukan. Tujuan dari dibuat 5 (lima) skenario pengujian karena banyaknya train set mempengaruhi hasil dari parsing. Skenario pertama dilakukan dengan sejumlah 100 data (90 untuk train set, 10 untuk test set), skenario kedua dilakukan dengan sejumlah 200 data (180 untuk train set, 20 untuk test set), skenario ketiga dilakukan dengan sejumlah 300 data (270 untuk train set, 30 untuk test set), skenario keempat dilakukan dengan sejumlah 400 data (360 untuk train set, 40 untuk test set), dan skenario kelima dilakukan dengan sejumlah 500 data (450 untuk train set, 50 untuk test set). Masing-masing skenario dilakukan 1 (satu) kali percobaan. Hasil dari kelima skenario yang telah dilakukan dengan menghitung nilai precision, recall, dan f1 score menggunakan PARSEVAL metrics dapat dilihat pada Tabel 5 - 9. Pengujian ini membandingkan parse tree dari output keluaran sistem dengan parse tree gold standard.

Tabel 5. Hasil Skenario Pertama Dengan 100 data

Test Set	Precision	Recall	F1 Score
10	54,63%	87,13%	0,66

Tabel 6. Hasil Skenario Kedua Dengan 200 data

Test Set	Precision	Recall	F1 Score
20	43,36%	71,53%	0,53

Tabel 7. Hasil Skenario Ketiga Dengan 300 data

Test Set	Precision	Recall	F1 Score
30	37,79%	66,28%	0,44

Tabel 8. Hasil Skenario Keempat Dengan 400 data

Test Set	Precision	Recall	F1 Score
40	41,06%	72,26%	0,49

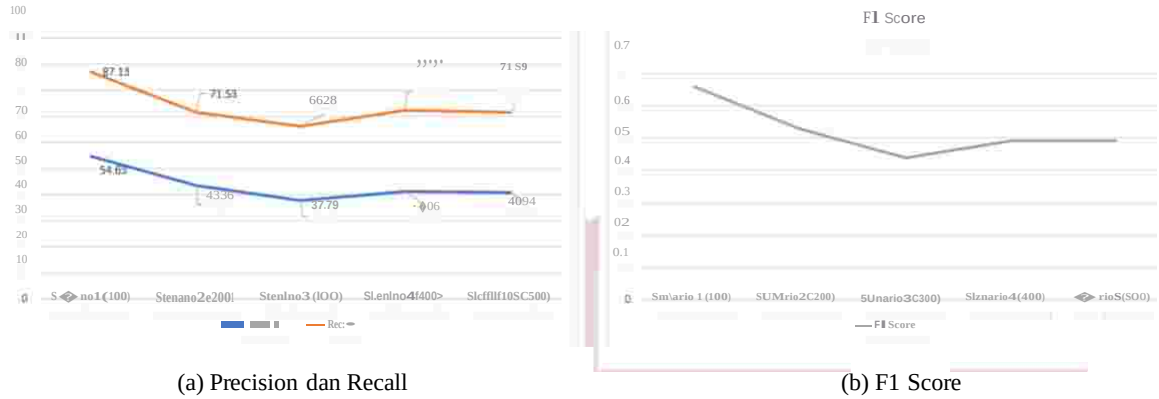
Tabel 9. Hasil Skenario Kelima Dengan 500 data

Test Set	Precision	Recall	F1 Score
50	40,94%	71,59%	0,49

4.2 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan dengan menghitung berapa banyak sistem berhasil mengubah kalimat Bahasa Indonesia kedalam bentuk parse tree, didapat persentase keberhasilan sistem sebesar 84%. Semakin banyak aturan produksi dan kata (lexicon) yang terdapat pada grammars, maka semakin besar kemungkinan sistem berhasil dalam melakukan parsing.

Banyaknya dataset begitu penting dalam melakukan parsing, karena proses parsing dilakukan berdasarkan aturan produksi dan kata (lexicon) yang terdapat pada grammars. Grammars di dapat dengan cara ekstraksi aturan produksi dari treebank yang ada pada dataset. Begitu minim treebank Indonesia yang tersebar secara publik, menjadi kendala dalam penelitian ini yang membutuhkan banyak data untuk memudahkan dalam penelitian.



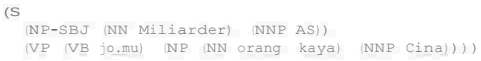
(a) Precision dan Recall (b) F1 Score

Gambar 12. Grafik Persentase Evaluasi

Berdasarkan hasil pengujian yang telah dilakukan dengan dengan menghitung nilai precision, nilai recall, dan nilai f1 score menggunakan PARSEVAL metrics, jumlah banyaknya train set mempengaruhi hasil parsing. Bentuk parse tree yang dihasilkan berbeda-beda tergantung dari banyaknya train set, semakin banyak jumlah train set maka semakin banyak pula jumlah aturan produksi yang terdapat pada grammars. Hal tersebut dapat dilihat pada Gambar 12 yang menunjukan grafik persentase dari precision, recall, dan f1 score yang cenderung menurun saat bentuk parse tree yang dihasilkan sistem dibandingkan dengan parse tree gold standard. Sebagai contoh, kalimat ”Miliarder AS jamu orang kaya Cina” pada skenario pertama dan skenario kedua mengalami perbedaan pada bentuk parse tree yang dihasilkan, dan akan dibandingkan dengan gold standard untuk mengetahui skenario mana yang paling mendekati benar. Perbedaan tersebut dapat dilihat pada Gambar 13 dan Gambar 14.

(a) Hasil Parse Tree Pada Skenario Pertama (b) Hasil Parse Tree Pada Skenario Kedua

Gambar 13. Perbandingan Hasil Parse Tree Pada Skenario Pertama dan Kedua



Gambar 14. Gold Standard

Bentuk parse tree yang dihasilkan tergantung pada treebank yang digunakan, yang dimana akan berdampak pada keputusan untuk memilih grammar yang paling valid untuk sebuah kalimat. Masalah pada pengujian ini diakibatkan bentuk parse tree yang dihasilkan berdasarkan pada treebank yang hanya sedikit berisi kalimat yang kompleks dengan struktur SPOK (Subjek, Predikat, Obyek, Deskripsi). Masalah akan terselesaikan jika tersedia treebank lengkap yang berisi kalimat yang kompleks dengan struktur SPOK (Subjek, Predikat, Obyek, Deskripsi), maka besar kemungkinan parse tree yang dihasilkan benar.

5. Kesimpulan

Kesimpulan yang diperoleh dari hasil pengujian, sistem berhasil melakukan parsing dalam mengubah kalimat Bahasa Indonesia ke dalam bentuk parse tree dengan persentase keberhasilan sistem sebesar 84%. Jumlah data yang digunakan mempengaruhi keberhasilan sistem dalam melakukan parsing, lalu dari hasil pengujian menggunakan PARSEVAL metrics didapat hasil precision, recall, dan f1 score. Pengujian terbaik pada skenario 1 yang mendapatkan 54,69% precision, 87,13% recall, dan 0,66 f1 score.

Saran untuk penelitian selanjutnya mencari dataset yang berisi treebank Bahasa Indonesia lengkap yang berisi kalimat yang kompleks dengan struktur SPOK (Subjek, Predikat, Obyek, Deskripsi) agar masalah yang timbul pada pengujian ini dapat terselesaikan. Kedepannya proses parsing dapat dilakukan dengan metode yang berbeda, salah satunya adalah Top-Down Parsing, dan pengujian hasil parsing dapat dilakukan dengan menggunakan pengukuran lain, salah satunya adalah leaf-ancestor (LA) metric.

Daftar Pustaka

[1]Mark Johnson Matthew Lease, Eugene Charniak. Parsing and it's application for conversational speech. Brown Laboratory for Linguistic Information Processing Brown University, 2005.

[2]Boumediene Belkhouche Maad Shatnawi. Parse trees of arabic sentences using the natural language toolkit. College of IT, UAE University, Al Ain, 2012.

[3]E. Brill. Some advances in rule-based part of speech tagging. AAAI-94 Proceedings, pages 722–727, 1994.

[4]G. Satta H. Bunt, J. Carroll. New Developments In Parsing Technology. Springer, 2005.

[5]Sara Stymne. Treebank grammars and parser evaluation. Department of Linguistics and Philology, 2014.

[6]Rosa Ariani Sukamto. Penguraian bahasa indonesia dengan menggunakan pengurai collins. Master Thesis, Institut Teknologi Bandung, 2009.

[7]Holon Institute od Technology. Information system. `http://www.hit.ac.il/staff/leonidm/information-systems/ch68.html`, 2012. Online; Accessed 29 July 2020.

[8]Alexander Gelbukh. Special issue: Natural language processing and its applications. Instituto Politécnico Nacional Centro de Investigación en Computación México, Mexico, 2010.

[9]J. E. Schmidt, F. L. The validity and utility of selection methods in personnel psychology: Practica and theoretical implications of 85 years of research findings. Springer, 1998.

[10]Xiaojun Wan Meichun Liu Weiwei Sun, Yufei Chen. Parsing chinese sentences with grammatical relations. Institute of Computer Science and Technology and Center for Chinese Linguistics, City University of Hong Kong Department of Linguistics and Translation, 2018.

[11]Elisa Margareth Sibarani. A study of parsing process on natural language processing in bahasa indonesia. IEEE 16th International Conference on Computational Science and Engineering, 2013.

[12]Alejandro Pazos Juan Ram ón Rabuñal Dopico, Julian Dorado. Rule Engines and Agent-Based Systems. IGI Global, 2008.

[13]James Suciadi. Studi analisis metode-metode parsing dan interpretasi semantik pada natural language processing. Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra, 2001.

[14]NLP-progress. Automatic parser evaluation. `http://nlpprogress.com/english/constituency_parsing.html`. Online; Accessed 1 August 2020.

[15]M. Arimura. A grammar-based compression using a variation of chomsky normal form of context free grammar. In 2016 International Symposium on Information Theory and Its Applications (ISITA), pages 246–250, October 2016.

[16]S. Lappin A. Clark, C. Fox. The handbook of computational linguistics and natural language processing. In Blackwell Handbooks, Linguistics, October 2010.

[17]A. D. Gaikwad R. C. Dharmik, R. S. Bhanuse. Ambiguity of context free grammar using the cyk algorithm. In 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), pages 1–6, February 2016.

[18]J. D. Ullman J. E. Hopcroft. Introduction to automata theory, languages, and computation. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, page 1st Edition, 1990.

[19]P. Barabs L. Kovcs. Experiences in building of context-free grammar tree. In 2011 IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pages 67–71, January 2011.

[20]K. Katz R. MacIntyre A. Bies, M. Ferguson. Bracketing guidelines for treebank ii style penn treebank project. http://catalog.ldc.upenn.edu/docs/LDC99T42/prsguid1.pdf, 1995. Online; Accessed 30 June 2020.

[21]Habash N. Introduction to arabic natural language processing. Morgan Claypool Publishers, 2010.

[22]Josef van Genabith Ines Rehbein. Evaluating evaluation measures. NODALIDA 2007 Conference Proceedings, pp. 372–379, 2007.

[23]Flickinger Ezra Black, Steve Abney. A procedure for quantitatively comparing the syntactic coverage of english grammarss. Proceedings of the 1991 DARPA Speech and Natural Language Workshop, 1991.

[24]Arawinda Dinakaramani Farm Rashel Ruli Andry Luthfi. Indonesian pos tagged corpus. http://bahasa.cs.ui.ac.id/treebank/corpus, 2017. Online; Accessed 20 May 2020.

[25]Dr. Sarmad Hussain. Pan localization. http://www.panl10n.net/, 2010. Online; Accessed 5 August 2020.

[26]V. Gurusamy. Preprocessing techniques for text mining. Conference Paper RTRICS, India, 2014.

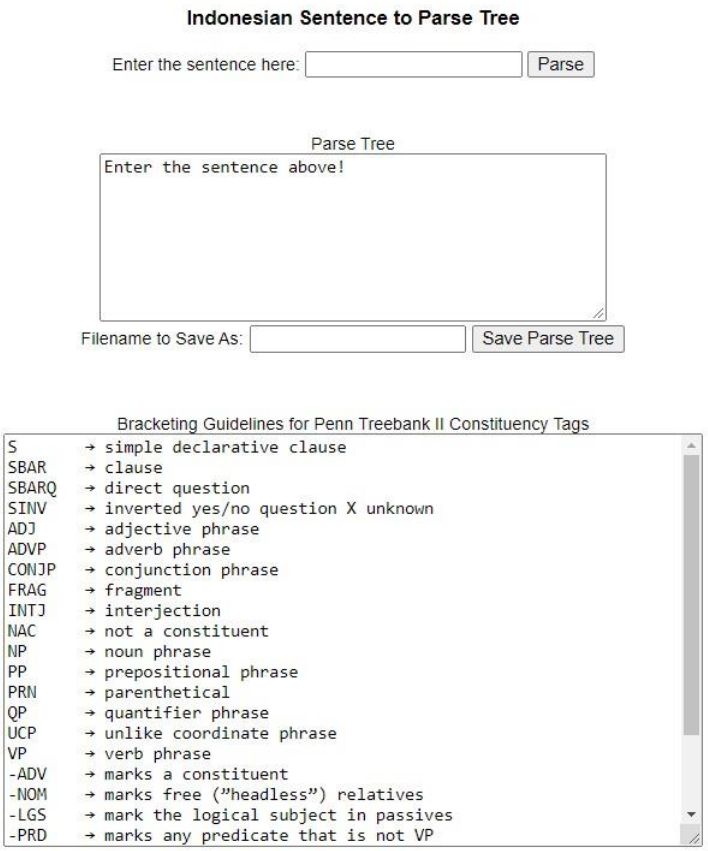
[27]D. Manning. Introduction to information retrieval. Cambridge University Press New York, NY, USA, 2008.

[28]Philipp Koehn. Advanced natural language processing lecture 17 statistical parsing ii. http://www.inf.ed.ac.uk/teaching/courses/anlp/slides/anlp14b-2x2.pdf, 2013. Online; Accessed 15 July 2020.

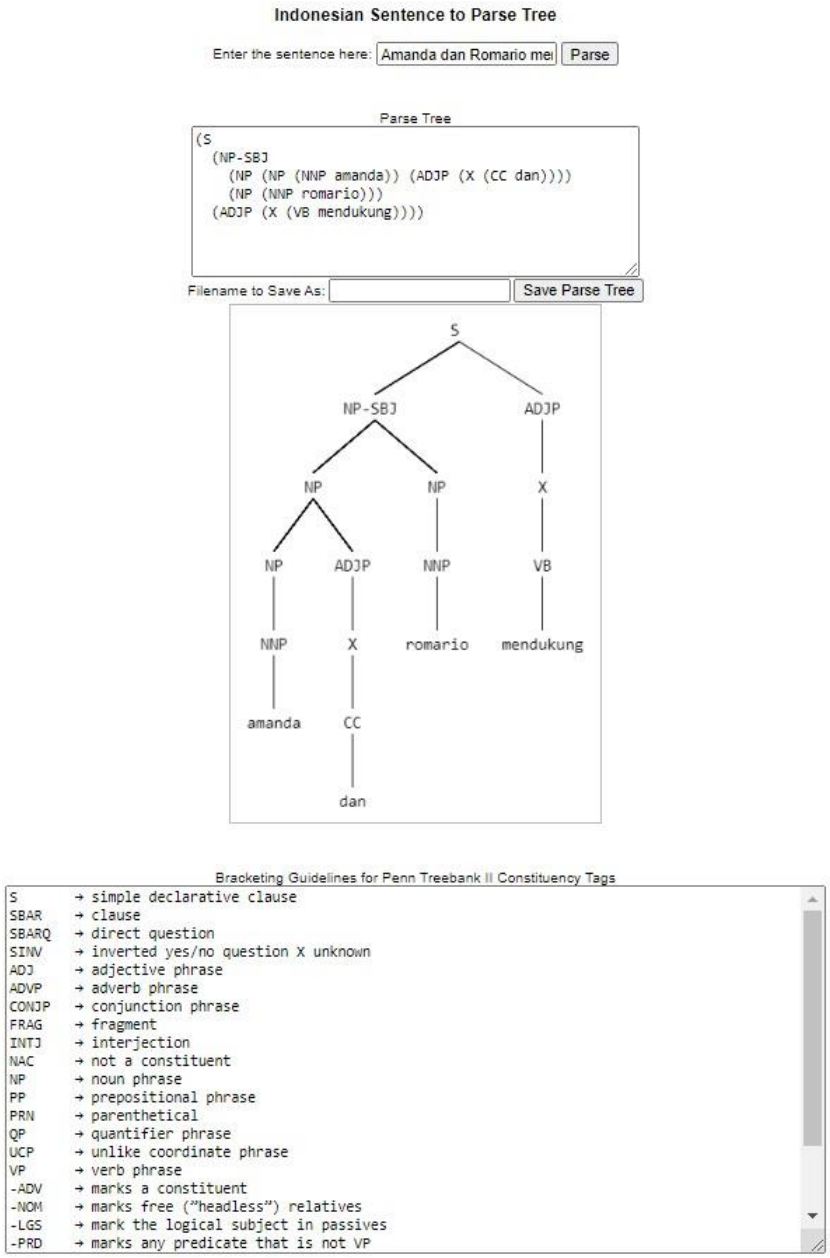
[29]Indiana University Dept. of Linguistics. Parser evaluation. https://wiki.eecs.yorku.ca/course_archive/2014-15/W/6339/_media/10e-eval-2x3.pdf, 2009. Online; Accessed 15 July 2020.

[30]Margus Treumuth Taavet Kikas. Constituency parsing. http://kodu.ut.ee/~treumuth/NLP/syntax.pdf, 2007. Online; Accessed 20 July 2020.

Lampiran



Gambar 1. Web-based



Gambar 2. Web-based (Output)

	Precision	Recall	F1 Score
0	0.555556	0.909091	0.689655
1	0.750000	1.000000	0.857143
2	0.461538	0.857143	0.600000
3	0.500000	1.000000	0.666667
4	0.571429	0.666667	0.615385
5	0.700000	1.000000	0.823529
6	0.285714	0.666667	0.400000

```
f"Precision = {table.mean()['Precision'] * 100}%"
```

```
'Precision = 54.63195534624106%'
```

```
f"Recall = {table.mean()['Recall'] * 100}%"
```

```
'Recall = 87.13667285095858%'
```

```
f"F1 Score = {table.mean()['F1 Score']}"
```

```
'F1 Score = 0.664625531910377'
```

Gambar 3. Evaluasi Skenario 1

	Precision	Recall	F1 Score
0	0.555556	0.909091	0.689655
1	0.166667	0.333333	0.222222
2	0.461538	0.857143	0.600000
3	0.250000	0.500000	0.333333
4	0.428571	0.500000	0.461538
5	0.375000	0.750000	0.500000
6	0.700000	1.000000	0.823529
7	0.166667	0.333333	0.222222
8	0.437500	0.777778	0.560000
9	0.500000	0.800000	0.615385
10	0.555556	0.909091	0.689655
11	0.375000	0.750000	0.500000
12	0.714286	1.000000	0.833333
13	0.500000	0.800000	0.615385
14	0.090909	0.333333	0.142857
15	0.428571	0.750000	0.545455
16	0.666667	0.857143	0.750000

```
f"Precision = {table.mean()['Precision'] * 100}%"
```

```
'Precision = 43.367571970513154%'
```

```
f"Recall = {table.mean()['Recall'] * 100}%"
```

```
'Recall = 71.53085476614889%'
```

```
f"F1 Score = {table.mean()['F1 Score']}"
```

```
'F1 Score = 0.5355629557836933'
```

Gambar 4. Evaluasi Skenario 2

	Precision	Recall	F1 Score
0	0.555556	0.909091	0.689655
1	0.166667	0.333333	0.222222
2	0.461538	0.857143	0.600000
3	0.250000	0.500000	0.333333
4	0.428571	0.500000	0.461538
5	0.375000	0.750000	0.500000
6	0.700000	1.000000	0.823529
7	0.166667	0.333333	0.222222
8	0.375000	0.333333	0.352941
9	0.062500	0.200000	0.095238
10	0.555556	0.909091	0.689655
11	0.375000	0.750000	0.500000
12	0.714286	1.000000	0.833333
13	0.500000	0.800000	0.615385
14	0.090909	0.333333	0.142857
15	0.428571	0.750000	0.545455
16	0.666667	0.857143	0.750000
17	0.200000	0.500000	0.285714
18	0.166667	1.000000	0.285714
19	0.300000	0.750000	0.428571
20	0.125000	0.250000	0.166667
21	1.000000	1.000000	1.000000
22	0.090909	1.000000	0.166667
23	0.071429	1.000000	0.133333
24	0.500000	0.285714	0.363636
25	0.500000	0.333333	0.400000

```
f"Precision = {table.mean()['Precision'] * 100}%"
```

```
'Precision = 37.79419832304448%'
```

```
f"Recall = {table.mean()['Recall'] * 100}%"
```

```
'Recall = 66.28787878787878%'
```

```
f"F1 Score = {table.mean()['F1 Score']}"
```

```
'F1 Score = 0.4464487667288416'
```

Gambar 5. Evaluasi Skenario 3

7	0.285714	0.666667	0.400000
8	0.500000	0.571429	0.533333
9	0.437500	0.777778	0.560000
10	0.333333	0.600000	0.428571
11	0.555556	0.909091	0.689655
12	0.300000	0.750000	0.428571
13	0.500000	0.800000	0.615385
14	0.272727	0.600000	0.375000
15	0.100000	0.333333	0.153846
16	0.428571	0.750000	0.545455
17	0.666667	0.857143	0.750000
18	0.333333	0.500000	0.400000
19	0.166667	1.000000	0.285714
20	0.400000	1.000000	0.571429
21	0.250000	0.500000	0.333333
22	0.750000	1.000000	0.857143
23	0.090909	1.000000	0.166667
24	0.071429	1.000000	0.133333
25	0.750000	0.428571	0.545455
26	0.666667	0.333333	0.444444
27	0.600000	0.857143	0.705882
28	0.454545	0.833333	0.588235
29	0.300000	0.500000	0.375000
30	0.357143	0.625000	0.454545
31	0.500000	0.571429	0.533333
32	0.727273	1.000000	0.842105
33	0.272727	0.600000	0.375000
34	0.300000	0.500000	0.375000
35	0.500000	0.800000	0.615385

```
f"Precision = {table.mean()['Precision'] * 100}%"
```

```
'Precision = 41.06895804812471%'
```

```
f"Recall = {table.mean()['Recall'] * 100}%"
```

```
'Recall = 72.26060205226872%'
```

```
f"F1 Score = {table.mean()['F1 Score']}"
```

```
'F1 Score = 0.4951201112993398'
```

Gambar 6. Evaluasi Skenario 4

20	0.400000	1.000000	0.571429
21	0.250000	0.500000	0.333333
22	0.750000	1.000000	0.857143
23	0.090909	1.000000	0.166667
24	0.071429	1.000000	0.133333
25	0.750000	0.428571	0.545455
26	0.666667	0.333333	0.444444
27	0.750000	0.750000	0.750000
28	0.600000	0.857143	0.705882
29	0.454545	0.833333	0.588235
30	0.300000	0.500000	0.375000
31	0.357143	0.625000	0.454545
32	0.500000	0.571429	0.533333
33	0.727273	1.000000	0.842105
34	0.272727	0.600000	0.375000
35	0.300000	0.500000	0.375000
36	0.500000	0.800000	0.615385
37	0.083333	0.250000	0.125000
38	0.333333	0.600000	0.428571
39	0.545455	0.857143	0.666667
40	0.428571	1.000000	0.600000
41	0.272727	0.600000	0.375000

```
f"Precision = {table.mean()['Precision'] * 100}%"
```

```
'Precision = 40.94820193034479%'
```

```
f"Recall = {table.mean()['Recall'] * 100}%"
```

```
'Recall = 71.59752284752285%'
```

```
f"F1 Score = {table.mean()['F1 Score']}"
```

```
'F1 Score = 0.4945133833812936'
```

Gambar 7. Evaluasi Skenario 5