

Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks

Tugas Akhir
diajukan untuk memenuhi salah satu syarat
memperoleh gelar sarjana
dari Program Studi Teknologi Informasi
Fakultas Informatika
Universitas Telkom

1103130095

Mukhtar Haris



Program Studi Sarjana Teknologi Informasi
Fakultas Informatika
Universitas Telkom
Bandung

2018

LEMBAR PENGESAHAN

Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks

Suggestion and Warning System on Syntax Tree Maker Application

NIM: 1103130095

Mukhtar Haris

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh
gelar pada Program Studi Sarjana Teknologi Informasi
Fakultas Informatika
Universitas Telkom

Bandung, 29 Agustus 2018

Menyetujui

Pembimbing I

Pembimbing II

Dr. Moch Arif Bijaksana

NIP: 03650029

Totok Suhardijanto, M.Hum., Ph.D.

NIP: 0706050101

Ketua Program Studi
Sarjana Teknologi Informasi,

Said Al Faraby, S.T., M.Sc.

NIP: 15890019

LEMBAR PENGESAHAN

Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks

Suggestion and Warning System on Syntax Tree Maker Application

NIM: 1103130095

Mukhtar Haris

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh
gelar pada Program Studi Sarjana Teknologi Informasi
Fakultas Informatika
Universitas Telkom

Bandung, 29 Agustus 2018

Menyetujui

Pembimbing I

Dr. Moch Arif Bijaksana

NIP: 03650029

Pembimbing II

Totok Suhardijanto, M.Hum., Ph.D.

NIP: 0706050101

**Ketua Program Studi
Sarjana Teknologi Informasi,**

Said Al Faraby, S.T., M.Sc.

NIP: 15890019

LEMBAR PERNYATAAN

Dengan ini saya, Mukhtar Haris, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul **” Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks ”** beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang belaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika dikemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya.

Bandung, 29 Agustus 2018

Yang Menyatakan,

Mukhtar Haris

LEMBAR PERNYATAAN

Dengan ini saya, Mukhtar Haris, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul **"Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks"** beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang belaku dalam masyarakat keilmuan. Saya siap menanggung resiko/sanksi yang diberikan jika dikemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya.

Bandung, 29 Agustus 2018

Yang Menyatakan,



Mukhtar Haris

Daftar Isi

1. Pendahuluan	11
2. Studi Terkait	12
2.1 Context Free Grammar	12
2.2 Teori Kebahasaan	13
2.3 Treebank	13
2.4 POS Tagging	13
2.5 Konvensi Treebank INACL	13
2.6 Komparasi Aplikasi Sejenis	13
2.7 TIKSentence TreeMaker	14
3. Membangun Sistem Saran dan Peringatan	14
3.1 <i>Input-Output</i> Sistem	14
3.2 Proses Melatih Aturan Frasa Sistem	15
3.2.1 Mengambil Simbol dan Label dari Kalimat	15
3.2.2 Mengubah Kalimat Kedalam Bentuk CFG	15
3.2.3 Menyimpan Seluruh Aturan CFG Data Latih	15
3.3 Proses Pengecekan Data Uji	15
3.3.1 Mengambil Simbol dan Label dari Kalimat	16
3.3.2 Mengubah Kalimat Kedalam Bentuk CFG	16
3.3.3 Mengubah Pola Sesuai Saran	16
4. Evaluasi	16
4.1 Hasil Pengujian	16
4.2 Analisis Hasil Pengujian	17
5. Kesimpulan	17

Daftar Gambar

1	Aturan CFG	12
2	Pohon Kalimat	24
3	Pohon Kalimat (Bentuk Lain)	24
4	Flowchart Umum Sistem	25
5	Flowchart Proses Melatih Aturan Frasa Sistem	26
6	Flowchart Mengubah Data Uji Menjadi Bentuk CFG	27
7	Flowchart Pengecekan dan Pemberian Saran	28
8	Kalimat 1 : Bentuk pohon	29
9	Kalimat 1 : Bentuk teks	29
10	Kalimat 1 : Bentuk label teks	29
11	Kalimat 1 : Bentuk CFG	30
12	Kalimat 1 : Bentuk CFG 2	30
13	Kalimat 2 : Bentuk pohon	31
14	Kalimat 2 : Bentuk teks	31
15	Kalimat 2 : Bentuk label teks	31
16	Kalimat 2 : Bentuk CFG	31
17	Kalimat 2 : Bentuk CFG 2	32
18	Gabungan CFG dari 2 kalimat	32
19	Kalimat data uji : Bentuk pohon	33
20	Kalimat data uji : Bentuk teks	33
21	Kalimat data uji : Bentuk label teks	33
22	Kalimat data uji : Bentuk CFG	33
23	Kalimat data uji : Bentuk CFG 2	34
24	Ilustrasi Stack Kata dan Aturan CFG Data Latih dan Data Uji	34
25	Flowchart Cara Kerja Mengubah Data Ke CFG	35
26	Daftar aturan yang gagal diubah oleh sistem	36
27	Tampilan aplikasi ketika baru dibuka	36
28	Tampilan aplikasi ketika melakukan labelling	37

29	Tampilan aplikasi ketika kalimat belum diberi label jenis frasa	37
30	Tampilan aplikasi ketika kalimat telah diberi label jenis frasa	38
31	Contoh Pohon yang Ditutup Untuk Survey	39
32	Contoh Pertanyaan pada Survey	39

Daftar Tabel

1	Tabel komparasi aplikasi	14
2	Accuracy, Recall dan Precision, F1 score table	16
3	Tabel nomina	19
4	Tabel pronomina	19
5	Tabel verba	19
6	Tabel adjektiva	20
7	Tabel adverbia	20
8	Tabel konungsi	20
9	Tabel preposisi	20
10	Tabel interjeksi	20
11	Tabel katagorifatis	21
12	Tabel determina	21
13	Tabel partikel	21
14	Tabel bilangan	21
15	Tabel lambang satuan	21
16	Tabel mata uang	22
17	Tabel lambang karakter	22
18	Tabel label katagori frasa	23
20	Tabel label penggabungan kata	24
22	Tabel Plot Klasifikasi Recall and Precision	38
23	Detail Accuracy, Recall dan Precision, F1 score table	39

Sistem Saran dan Peringatan Pada Aplikasi Alat Bantu Pembuatan Pohon Sintaks

Mukhtar Haris¹, Moch Arif Bijaksana², Totok Suhardijanto³

^{1,2}Fakultas Informatika, Universitas Telkom, Bandung

³Departemen Linguistik, Fakultas Ilmu Pengetahuan Budaya, Universitas Indonesia

¹mukhtarharis@student.telkomuniversity.ac.id, ²arifbijaksana@telkomuniversity.ac.id,

³suhardiyanto@gmail.com

Abstrak

TIKSentence TreeMaker adalah aplikasi yang dibuat oleh *Indonesia Association of Computational Linguistics* (INACL) yang berfungsi untuk memberi label pada satuan frasa dalam kalimat. Aplikasi ini memiliki kelebihan dibidang tampilan atau *User Interface* (UI), dimana struktur kata dan frasa didalam kalimat dapat dibentuk menjadi struktur pohon secara langsung, sehingga pengguna lebih mudah melihat dan memberi label frasa pada struktur kalimat tersebut. Selain tampilan atau UI, kelebihan aplikasi ini adalah mengikutsertakan pengguna dalam proses pelabelan sehingga pengguna dapat menyesuaikan aturan-aturan frasa dengan kondisi bahasa tersebut. Setiap aplikasi pasti memiliki kelebihan dan kekurangan, TIKSentence TreeMaker pun memiliki kekurangan dimana aplikasi ini masih belum dapat membantu pengguna dibidang pemberian saran label jenis frasa dimana saat ini pengguna diberi kebebasan dalam menentukan label frasa baik label jenis frasa itu cocok atau tidak dengan aturan yang diberinya. Penulis mencoba untuk melengkapi beberapa bagian dari kekurangan aplikasi ini dengan cara membuat sistem yang dapat mendeteksi aturan frasa dan memberi saran label jenis frasa apa yang tepat ketika aturan frasa tersebut salah tanpa mengubah susunan struktur frasanya. Sistem dibangun untuk mengecek label jenis frasa secara sintaktik menggunakan metode *Context Free Grammar* (CFG) dimana berguna untuk mengubah bentuk pohon pada aplikasi menjadi bentuk yang lebih mudah dibaca oleh sistem. Sistem mengecek setiap aturan frasa yang diujikan lalu mengubah label jenis frasanya jika aturan tersebut dianggap salah oleh sistem dengan label jenis frasa yang disarankan oleh sistem. Hasil akurasi dari sistem yang telah dibuat sekitar 95% sedangkan akurasi untuk data yang diujikan pada keadaan awal sekitar 87% yang berarti sistem yang dibuat telah berhasil mengubah beberapa label jenis frasa pada aturan frasa yang salah menjadi label jenis frasa pada aturan frasa yang benar.

Kata kunci : label frasa, bahasa Indonesia, context free grammar, INACL

Abstract

TIKSentence TreeMaker is an application created by Indonesia Association of Computational Linguistics (INACL) that used to giving phrase label units in sentences. This application has advantages in user interface (UI), where word and phrase structures in sentences can be shaped into tree structures in real time, so users easier to see and giving phrase labels on the sentence structure. Besides the UI, another advantage of this application is include users in labelling process so that users can adjust the phrase rules with language conditions. Every application always have advantages and disadvantages, TIKSentence TreeMaker also has drawback where this application cannot help users in providing prhase labels which users are given freedom in determine the phrase labels wether it is match or not with the given rules. Author tries to solve some parts of this application's shortage by creating a system that can detect phrase rules and suggest phrase labels that is correct base n system when the phrase rule is considered false without changing its phrase structure. The system is built to check phrase labels in syntactic manner with context free grammar (CFG) method where it is useful for changing the tree structure shapes from application into another shapes that system easier to read. The system check every phrase rules that tested and change its phrase labels if the rule are considered incorrect by the system with phrase label that suggested y the system. The accuracy from the system that has been made is around 95% while the accuracy for data test in initial condition is 87% which means the system succeeded in changing several incorrect phrase labels in phrase rules into correct phrase labels in phrase rules.

Keywords: phrase labels, Indonesian language, context free grammar, INACL

1. Pendahuluan

Latar Belakang

Dalam pemrosesan teks, memberi label pada satuan teks dapat membantu mengenali macam-macam karakteristik satuan teks. Label dapat diberikan pada satuan teks berupa artikel, paragraf, kalimat, frasa, dan yang lainnya [5]. TIKSentence TreeMaker adalah salah satu aplikasi yang digunakan untuk memberi label pada salah satu satuan teks, yaitu frasa. Frasa adalah satuan teks yang berupa gabungan dari 2 (dua) atau lebih kata yang bersifat non-prediktif [10]. Fungsi dari aplikasi TIKSentence TreeMaker adalah membuat dataset yang berupa kalimat-kalimat yang telah lengkap dengan label frasa dan label kata. Selain TIKSentence TreeMaker, terdapat pula beberapa aplikasi lain yang berfungsi memberi label frasa pada satuan kalimat seperti *Stanford Syntax Parser*, namun aplikasi *Stanford Syntax Parser* hanya dapat digunakan pada beberapa bahasa diantaranya bahasa inggris, arab, cina, perancis dan spanyol. Aplikasi TIKSentence TreeMaker memungkinkan setiap kalimat bahasa (yang telah dilengkapi label kata) untuk dibuat label frasanya. Untuk dapat melakukan hal tersebut, aplikasi TIKSentence TreeMaker dibuat menjadi aplikasi yang memerlukan bantuan pengguna dalam pelabelannya. Aplikasi TIKSentence TreeMaker memudahkan pengguna dalam memberi label frasa dari sebuah kalimat dengan cara tampilan dari aplikasi itu, dimana setiap kalimat diubah menjadi bentuk pohon sehingga pengguna tidak sulit melihat bentuk dari kalimat tersebut. Pengguna yang menggunakan TIKSentence TreeMaker untuk melakukan pelabelan frasa tidak semua adalah seorang ahli bahasa atau orang-orang yang memiliki keahlian atau latar belakang pendidikan dibidang tersebut, namun pengguna TIKSentence TreeMaker adalah seluruh masyarakat yang memiliki kemampuan dalam berbahasa yang baik dan benar terhadap bahasa yang akan dilabeli. Karena melibatkan pengguna tersebut, TIKSentence TreeMaker memiliki kelemahan atau kekurangan yang membutuhkan pengembangan lebih lanjut. Kekurangan tersebut adalah tidak adanya sebuah sistem pada aplikasi TIKSentence TreeMaker yang dapat memberi saran dan peringatan kepada pengguna ketika pengguna tersebut terdeteksi melakukan kesalahan ketika memberi label.

Pada tugas akhir ini penulis mencoba untuk sedikit melengkapi kekurangan dari aplikasi TIKSentence Treemaker dengan cara membuat sebuah sistem yang dapat mendeteksi aturan frasa dari kalimat dan mengganti label jenis frasa dengan label jenis frasa yang disarankan oleh sistem jika aturan frasa tersebut terdeteksi salah oleh sistem. Sistem yang dibuat memerlukan data berupa berkas-berkas hasil pelabelan dari aplikasi TIKSentence TreeMaker yang berupa berkas dengan ekstensi .tre untuk digunakan baik untuk melatih sistem dalam membuat aturan frasa atau untuk proses mendeteksi serta memperbaiki kalimat. Dataset yang digunakan berupa kalimat-kalimat yang telah dilengkapi dengan label jenis frasa dan label jenis kata serta seluruh berkas yang digunakan sebagai dataset pada sistem ini telah divalidasi oleh ahli bahasa (Totok Suhardijanto, M.Hum., Ph.D.). Sistem yang dibangun merupakan sistem yang berfungsi untuk mengganti label jenis frasa tanpa merusak atau mengubah struktur pohnnya ketika terdapat struktur pohon yang dianggap salah oleh sistem. Struktur pohon yang dibaca oleh sistem dianggap sebagai struktur yang berdiri sendiri (independen), dan sistem membaca pohon dimulai dari daun/leaf/label jenis kata menuju ke akar/root/label jenis frasa S. Sistem dibangun untuk mengecek label jenis frasa secara sintaktik menggunakan metode *Context Free Grammar* (CFG) dimana metode ini berguna untuk mengubah bentuk pohon pada aplikasi menjadi bentuk yang lebih mudah dibaca oleh sistem. Sistem mengecek setiap aturan frasa yang diujikan lalu mengubah label jenis frasanya jika aturan tersebut dianggap salah oleh sistem dengan label jenis frasa yang disarankan oleh sistem.

Topik dan Batasannya

Topik yang penulis angkat pada tugas akhir ini adalah bagaimana cara membuat sistem yang dapat mendeteksi aturan frasa dari berkas data uji yang telah disediakan dan memperbaiki label jenis frasa dari aturan tersebut ketika sistem mendeteksi label jenis frasa yang salah pada aturan frasa tanpa mengubah struktur atau bentuk dari aturan frasa tersebut.

Terdapat banyak faktor yang harus dipenuhi dalam sebuah sistem saran dan peringatan yang layak seperti pengecekan secara semantik, penyarianan struktur pohon, dan lain sebagainya. Karena terbatas waktu dalam pengajaran tugas akhir ini, maka penulis memberikan beberapa poin yang bertujuan untuk memperkecil *scope* bahasan dari tugas akhir ini. Beberapa poin tersebut adalah :

- Dataset yang digunakan adalah 120 kalimat valid yang telah dilengkapi dengan label jenis frasa dan label jenis kata. 100 kalimat digunakan untuk data latih, 20 kalimat digunakan untuk data uji dan validasi.
- Label jenis kata dan atau label jenis frasa pada kalimat dataset sudah dianggap benar sehingga tidak perlu adanya pengecekan terlebih dahulu terhadap label jenis kata dan label jenis frasa.

- Sistem yang dibangun tidak terintegrasi dengan aplikasi TIKSentence Treemaker karena masih banyak faktor yang belum tercakup didalam sistem yang dibuat ini.
- Sistem ini hanya mengecek label jenis frasa, klausa dan atau kata secara sintaksis atau secara aturan-aturan label berdasarkan data latih.
- Sistem ini hanya mengecek aturan frasa dan mengubah label jenis frasa tanpa mengubah struktur aturan tersebut jika aturan frasa tersebut dianggap salah oleh sistem (struktur bersifat statis).
- Kalimat dataset yang digunakan pada sistem ini dianggap hanya memiliki 1 (satu) variasi pohon.
- Untuk satu aturan frasa (satu jenis frasa dan komposisi jenis frasa nya) dengan aturan frasa yang lainnya tidak terikat ketika mengecek kesalahan dan mencari saran untuk aturan frasa tersebut.

Tujuan

Tujuan yang penulis capai pada tugas akhir ini adalah membuat sistem peringatan dan saran yang dapat mendeksi aturan frasa pada struktur pohon kalimat, lalu mengganti atau memperbaiki variabel label jenis frasa nya namun tidak mengubah komposisi aturan atau struktur pohon/frasa nya jika aturan frasa tersebut dianggap salah oleh sistem.

Organisasi Tulisan

Terdapat beberapa bagian didalam jurnal ini, diantaranya pendahuluan, studi terkait, sistem yang dibangun, evaluasi, kesimpulan. Pada bagian pendahuluan membahas latar belakang, topik, batasan sistem, tujuan serta organisasi tulisan yang digunakan dalam tugas akhir ini. Pada bagian studi terkait membahas materi-materi atau teori-teori apa saja yang digunakan pada tugas akhir ini dalam membangun sistem peringatan dan saran. Pada bagian sistem yang dibangun membahas alur serta contoh data yang terlibat didalam sistem secara berurutan. Pada bagian evaluasi membahas masalah hasil pengujian yang didapat serta analisis terhadap hasil tersebut. Bagian akhir atau bagian kesimpulan akan membahas kesimpulan akhir tentang sistem ini serta saran pengembangan untuk kedepannya.

2. Studi Terkait

2.1 Context Free Grammar

Context Free Grammar (CFG) adalah sebuah tata bahasa formal yang digunakan untuk mendeskripsikan seluruh kalimat yang mungkin. CFG dapat digunakan untuk menentukan sebuah pola yang dimiliki oleh sebuah kalimat sehingga dapat menentukan jenis frasa dan klausa yang dimiliki oleh kalimat tersebut [4] [9] [6] [3] [2] [8].

$$G = (V, \Sigma, R, S) \quad (1)$$

Berdasarkan rumus 1, terdapat 4 atribut yang dapat mendeskripsikan suatu *context free grammar* (G), yaitu V atau kumpulan non-terminal, Σ atau kumpulan terminal, R atau relasi antara V ke V/Σ , serta S atau start yang menandakan dimana CFG dimulai. Pada tugas akhir ini, notasi-notasi tersebut diubah sedemikian rupa sehingga dapat cocok untuk studi kasus dimana penulis akan mengubah aturan frasa yang berbentuk pohon menjadi CFG Gambar 1.

```

S -> NNO VP
VP -> VBT NP
NP -> NP NP | NNO NNO | NNP NP | NNO ADJ
  
```

Gambar 1. Aturan CFG

Dengan mengganti beberapa notasi dan bentuk aturan-aturan CFG, sistem dapat mengubah bentuk dari dataset yang menjadi masukan. Adapun notasi CFG Gambar 1 yang telah diperbaharui adalah :

- Simbol terminal (Σ) diubah menjadi label *Part of Speech* atau jenis kata yang menandakan akhir dari aturan tersebut seperti (NNO, VBT, NNP, ADJ, dan sebagainya).
- Simbol non-terminal (V) diubah menjadi label jenis frasa. Label ini menandakan bahwa dapat memanggil aturan lain dengan memiliki label jenis frasa yang sama seperti (NP, VP, dan sebagainya).
- Simbol start. Simbol start berguna untuk menandakan dimana letak aturan yang harus dimulai. Pada tugas akhir ini, selain start, S juga menandakan simbol S (Sentence) yang berarti root atau gabungan dari seluruh jenis kata dan jenis frasa.
- Simbol R diganti dengan $- >$. Simbol tersebut menandakan bahwa non-terminal (V/jenis frasa) memiliki rangkaian aturan atau relasi sebagai berikut.

2.2 Teori Kebahasaan

Teori kebahasaan adalah sebuah teori yang digunakan dalam sebuah bahasa yang berguna untuk mempermudah membaca atau mengenali sebuah bahasa. Dalam pelabelan frasa pada sebuah kalimat, biasanya kalimat akan dibuat pohon dimana akar dari pohon tersebut merupakan kalimat itu sendiri serta daun dari pohon tersebut merupakan jenis kata dari setiap kalimat [12]. Berdasarkan teori bahasa, lampiran Gambar 2 dapat diubah bentuknya menyesuaikan dengan keperluan tertentu, seperti diubah menjadi *context free grammar* (CFG) sehingga menjadi lampiran Gambar 3. Bentuk perubahan pada Gambar 3 tidak memiliki bentuk yang umum, sehingga bisa diubah menjadi bentuk apapun selama bentuk tersebut konsisten dari awal hingga akhir [12].

2.3 Treebank

Treebank adalah dataset-dataset yang berupa teks yang sudah dipisah atau sudah diproses menjadi struktur semantik [3]. Dalam komputasi linguistik, treebank ini sering digunakan sebagai dataset awal dalam membangun sistem pemrosesan teks. Satu treebank biasanya mewakili 1 (satu) bahasa tertentu, oleh karena itu terdapat banyak treebank dengan versi bahasa masing-masing. Pada tugas akhir ini penulis menggunakan treebank bahasa indonesia yang dikeluarkan oleh INACL [13] [11].

2.4 POS Tagging

Part-Of-Speech (POS) tagging adalah kegiatan menandai kata dengan jenis kata nya masing-masing. Part-of-speech adalah katagori kata-kata yang memiliki sifat kosa kata bahasa yang sama. Dalam konvensi INACL (Indonesian Association of Computational Linguistics) membagi part of speech menjadi beberapa aturan kelas yang dapat dilihat pada lampiran bagian 5.1-5.3 [9] [7] [14] [1] [15].

2.5 Konvensi Treebank INACL

Dalam bahasa indonesia, rangkaian kata dengan pola tertentu yang mempunyai sebuah makna disebut kalimat. Sebuah kalimat dapat dipecah menjadi kata-kata atau sekumpulan kata. Frase dan klausa adalah bagian dari kalimat atau sekumpulan kata yang memiliki makna yang tidak penuh. Daftar lengkap konvensi Treebank INACL terdapat pada bagian lampiran bagian 5.4. [14] [1] [15].

2.6 Komparasi Aplikasi Sejenis

Sudah ada beberapa aplikasi atau alat bantu pembangunan pohon sintaks baik dari indonesia maupun dari negara lain. Berikut adalah beberapa alat bantu pembuatan pohon sintaks serta kelebihan dan kelemahan nya :

Nama aplikasi	Platform	Kelebihan	Kelemahan
TIKSentence Tree-Maker	desktop	+mudah digunakan +posisi pohon dari kiri ke kanan (menyamping)	-belum memiliki sistem untuk memberikan saran atau peringatan -setiap kata harus sudah memiliki jenis katanya
Stanford Syntax Parser	website	+auto generate +memiliki beberapa bahasa +mengeluarkan beberapa informasi tambahan selain pohon sintaks	-hasil pohon yang diberikan tidak dilengkapi UI
Syntax Tree Generator	website	+auto generate	-hanya mengubah dari pola kalimat kedalam pohon

Tabel 1. Tabel komparasi aplikasi

Berdasarkan Table 1, tugas akhir kali ini akan mengembangkan aplikasi TIKSentenceTreemaker dengan membuat sistem saran dan peringatan menggunakan dataset yang sudah diberikan sebelumnya.

2.7 TIKSentence TreeMaker

TIKSentence TreeMaker adalah aplikasi yang dimiliki oleh *Indonesian Association of Computational Linguistic* (INACL) yang merupakan perkumpulan atau asosiasi yang bergerak dibidang komputasi linguistik di Indonesia. TIKSentence TreeMaker digunakan untuk membantu memberikan label pada tingkatan frasa atau klausa. Cara kerja TIKSentence TreeMaker adalah menggabungkan 2 (dua) atau lebih kata atau frasa menjadi jenis frasa tertentu sehingga kalimat tersebut akan membentuk sebuah pohon sintaks yang menunjukkan susunan frasa apa saja yang terdapat didalam kalimat tersebut.

3. Membangun Sistem Saran dan Peringatan

Sistem saran dan peringatan yang dibuat pada tugas akhir ini dibagi menjadi 2 (dua) proses, yaitu proses melatih aturan frasa sistem, proses pengecekan data uji serta memperbaiki formula atau aturan pada data uji serta 1 (satu) proses evaluasi sistem untuk menghitung nilai akurasi, *recall*, *precision* dan *F1 score*. Flowcart umum sistem terdapat pada bagian lampiran Gambar 4.

3.1 Input-Output Sistem

Sistem yang dibangun memerlukan berkas yang akan digunakan dalam proses melatih aturan frasa pada sistem serta proses pengecekan label jenis frasa. Berkas yang digunakan adalah berkas dimana berisi 120 (seratus dua puluh) kalimat yang telah dilengkapi dengan label jenis frasa dan jenis kata dan sudah divalidasi oleh ahli bahasa. Berkas ini dibagi menjadi 2 (dua) bagian, dimana 100 (seratus) kalimat digunakan untuk proses melatih aturan frasa sistem dan berkas yang berisi kalimat-kalimat ini akan disebut data latih, serta 20 (dua puluh) kalimat digunakan untuk proses pengecekan data uji dan berkas yang berisi kalimat-kalimat ini akan disebut data uji. Secara gambar, alur keluar masuk data dalam sistem dapat dilihat pada lampiran Gambar 4.

Pada proses melatih aturan frasa sistem, sistem menerima *input* data latih yang berisi kalimat-kalimat yang telah dilengkapi dengan label jenis frasa dan label jenis kata, serta pada proses ini akan mengeluarkan sebuah *Context Free Grammar* (CFG) dimana setiap aturan-aturan yang terdapat pada CFG ini mewaliki seluruh aturan-aturan frasa yang ada pada data latih dan CFG ini akan disebut sebagai CFG data latih.

Pada tahap proses pengecekan data uji, sistem menerima *input* data uji yang berisi kalimat yang telah dilengkapi dengan label frasa dan label jenis kata serta CFG data latih yang akan digunakan untuk proses pengecekan aturan frasa data uji dan pemberian saran label jenis frasa ketika aturan frasa tersebut dianggap salah oleh sistem. Proses pengecekan data uji ini akan mengeluarkan hasil berupa *Context Free Grammar* (CFG) dimana aturan-aturan CFG berdasarkan aturan-aturan frasa pada data uji dan label jenis frasa yang dianggap salah telah diperbaiki oleh sistem lalu CFG ini akan disebut sebagai CFG hasil.

3.2 Proses Melatih Aturan Frasa Sistem

Pada proses melatih aturan frasa sistem adalah proses dimana sistem akan menggunakan dataset yang berupa 100 (seratus) kalimat yang telah diberi label jenis frasa dan jenis kata dan mengubah kalimat-kalimat tersebut menjadi aturan-aturan CFG yang dapat dibaca oleh sistem. Proses ini mengubah data latih lampiran Gambar 9 menjadi CFG yang berisi aturan-aturan CFG lampiran Gambar 12. Flowchart alur kerja proses melatih aturan frasa sistem ada pada lampiran Gambar 5 dan adapun contoh untuk setiap masukan dan keluaran pada proses ini ada pada lampiran bagian "Contoh Data Input Output Proses Melatih Aturan Frasa Sistem".

3.2.1 Mengambil Simbol dan Label dari Kalimat

Langkah awal dalam proses melatih aturan frasa pada sistem adalah dengan melakukan mengambil beberapa simbol dan label-label yang ada pada kalimat yang sedang dibaca oleh sistem. Pada proses ini, setiap kata didalam kalimat dihilangkan sehingga pada kalimat tersebut hanya tersisa simbol ('(', ')', ',') serta label (label jenis frasa dan label jenis kata). Dari contoh lampiran Gambar 9, kata-kata yang dihapus adalah kata-kata yang memiliki pasangan dengan jenis kata seperti (NNO Konsep) diubah menjadi (NNO) sehingga akan menyisakan label jenis kata. Hal ini mempermudah dalam proses pembuatan aturan-aturan CFG. Menghapus kata pada sistem ini membantu dalam penyederhanaan proses. Secara sintaksis, kata-kata seperti (Konsep, untuk, dan sebagainya) tidak dibutuhkan karena sistem cukup melihat secara label jenis frasa atau label jenis katanya saja. Hasil akhir dari tahap ini berupa kalimat dimana hanya tersisa simbol ('(', ')', ',') dan label (label jenis frasa dan label jenis kata) lampiran Gambar 10 .

3.2.2 Mengubah Kalimat Kedalam Bentuk CFG

Setelah menghapus setiap kata, langkah berikutnya adalah membuat aturan-aturan CFG data latih untuk setiap kalimat pada data latih. Sistem menggunakan 2 (dua) buah stack dimana satu stack bertugas untuk membaca data uji yang berbentuk string simbol dan spasi ('(', ')', ',') serta label jenis frasa dan label jenis kata lalu menghasilkan satu aturan CFG sedangkan satu stack lainnya bertugas untuk membaca aturan CFG tersebut dan menentukan apakah aturan CFG yang sedang dibuat sudah selesai dan pindah untuk membuat aturan CFG selanjutnya, atau aturan CFG belum selesai dan masih berlanjut. Keseluruhan cara kerja sistem dalam mengubah data menjadi bentuk CFG dapat dilihat pada lampiran Gambar 24 dan lampiran Gambar 25. Hasil setelah kalimat berubah menjadi bentuk CFG terdapat pada lampiran Gambar 11. Setelah terbentuk CFG dari kalimat tersebut, sistem menghapus simbol SYM dikarenakan tidak berdampak besar pada nilai kebenaran secara sintaksis dan cenderung memperbanyak variasi aturan. Selain simbol SYM, simbol CCN dengan kondisi simbol tersebut harus berada diantara 2 (dua) seperti (NP CCN NNO), maka diubah menjadi (NP NNO). Aturan CFG yang memiliki jenis frasa S dihapus karena sifatnya yang sebagai root dan tidak termasuk kedalam frasa atau klausa. Pada akhir dari tahap ini, kalimat yang dibaca akan berubah bentuknya menjadi lampiran Gambar 12.

3.2.3 Menyimpan Seluruh Aturan CFG Data Latih

Setelah setiap kalimat pada data latih sudah diubah menjadi CFG-CFG yang merepresentasikan setiap kalimatnya lampiran Gambar 12 dan lampiran Gambar 17, langkah selanjutnya adalah menggabungkan CFG-CFG tersebut menjadi 1 (satu) kumpulan CFG utama. Cara menggabungkan CFG tersebut adalah dengan cara mengecek setiap aturan apakah aturan tersebut sudah pernah dimasukan kedalam data CFG utama atau belum. Jika aturan CFG tidak ditemukan didalam data CFG utama, maka aturan tersebut dimasukan kedalam CFG utama sehingga tidak terdapat aturan CFG ganda didalam CFG utama. CFG utama ini akan disebut sebagai CFG data latih dimana CFG ini berasal dari kalimat pada data latih. Hasil *output* dari tahap ini dapat dilihat pada lampiran Gambar 18.

3.3 Proses Pengecekan Data Uji

Ketika sistem selesai menjalankan proses melatih aturan frasa sistem atau sistem telah berhasil membuat CFG data latih, sistem akan masuk kedalam tahap proses pengecekan kesalahan pada data uji. Berdasarkan flowchart umum sistem, pada tahap ini terdapat 2 (dua) flowchart yang digunakan, yaitu flowchart untuk mengubah bentuk data uji menjadi CFG data uji serta flowchart untuk melakukan pengecekan dan pemberian saran pada data uji lampiran Gambar 6 dan lampiran Gambar 7. Pada lampiran Gambar 6, terdapat beberapa tahap yang sama dengan tahap mengubah data latih menjadi CFG data latih pada proses melatih aturan frasa sistem sehingga tidak akan dibahas panjang lebar untuk tahap-tahap yang sama tersebut. Contoh data pada proses pengecekan data uji terdapat pada lampiran Gambar 19 hingga lampiran Gambar 23.

3.3.1 Mengambil Simbol dan Label dari Kalimat

Langkah awal dalam tahap proses sama dengan langkah awal pada proses melatih aturan frasa sistem, yaitu mengambil label jenis frasa dan label jenis kata serta simbol ('(, ')', ' ').

3.3.2 Mengubah Kalimat Kedalam Bentuk CFG

Setelah kalimat yang dibaca hanya tersisa simbol dan label, tahap berikutnya adalah membuat aturan-aturan CFG untuk setiap kalimat pada data uji yang mana sama dengan langkah mengubah kalimat menjadi CFG dalam langkah pada proses melatih aturan frasa sistem yang telah dijelaskan sebelumnya. Hasil CFG dari langkah ini akan disebut sebagai CFG data uji dimana aturan CFG ini berisi aturan-aturan frasa dari 20 (dua puluh) kalimat data uji.

3.3.3 Mengubah Pola Sesuai Saran

Setelah data uji telah berubah menjadi bentuk CFG, sistem akan mulai mendeteksi aturan CFG data uji tersebut lampiran Gambar 7. Sistem akan membaca setiap pola pada CFG data uji dan dicocokan kedalam aturan CFG data latih, jika pola pada CFG data uji terdapat pada aturan CFG data latih, maka pola tersebut sukses atau dianggap benar oleh sistem dan pola tersebut tidak diubah jenis label nya. Jika pola pada CFG data uji tersebut tidak ditemukan pada aturan CFG data latih, sistem akan mencoba mencari saran jenis label frasa yang memungkinkan dengan cara mengecek rumus pola tersebut dengan setiap rumus pada aturan CFG data latih. Setelah mendapatkan saran jenis label frasa, sistem akan mengubah jenis frasa pada pola CFG data uji tersebut dengan saran yang dianjurkan sistem.

4. Evaluasi

Menentukan seberapa bagus sistem dalam melakukan pendekripsi aturan frasa yang salah serta mengganti label jenis frasa yang dianggap sesuai oleh sistem dapat menggunakan metode *recall and precision*. *Recall and precision* adalah metode yang digunakan pada pengenalan pola untuk menghitung ukuran suatu keterkaitan objek. *Recall* adalah nilai yang digunakan untuk menentukan berapa banyak objek yang seharusnya dipilih itu benar-benar terpilih. *Precision* adalah nilai yang digunakan untuk menentukan berapa banyak objek yang benar. *Recall and Precision* menggunakan pengklasifikasian nilai kebenaran menggunakan matrix antara data valid atau aktual dan data prediksi yang akan menghasilkan 4 (empat) kriteria pengklasifikasian, *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Pada tugas akhir ini, dalam menentukan klasifikasi kebenaran, penulis mengubah bentuk klasifikasi kebenaran menjadi lampiran Table 22. Adapun rumus *recall*, *precision*, *accuracy* serta *F1 score* dapat dilihat pada lampiran rumus 2 hingga rumus 5. Pada tugas akhir ini, sebagai pembanding penulis meminta 12 (dua belas) masyarakat indonesia dengan profesi minimal mahasiswa tingkat akhir (semester 8) untuk mengisi survei yang berkaitan dengan data uji. Dalam melakukan survei, data yang digunakan adalah data uji yang juga data uji digunakan untuk sistem, dimana label-label jenis frasa yang telah penulis ubah, penulis tutup dan para responden mengisi label yang tertutup itu dengan pilihan label jenis frasa yang tersedia lampiran gambar.

4.1 Hasil Pengujian

Berdasarkan hasil yang didapatkan setelah menjalankan sistem dan melakukan survei, didapatkan nilai *recall and precision*, akurasi serta *F1 score* dari sistem dan responden. Hasil data uji tersebut dicocokan dengan data uji yang telah divalidasi oleh ahli bahasa sebelumnya. Berikut adalah hasil dari pengujian dan Tabel 23:

Data test	Accuracy	Recall	Precision	F1 Score
Lower Bound	80.00%	Na	Na	Na
Upper Bound	100.00%	100.00%	100.00%	100.00%
Data uji awal	87.88%	Na	Na	Na
Hasil saran sistem	95.84%	100.00%	65.71%	79.31%
Rata-rata responden	91.75%	67.45%	58.33%	61.97%

Tabel 2. Accuracy, Recall dan Precision, F1 score table

4.2 Analisis Hasil Pengujian

Pada CFG data latih sangat memungkinkan ada nya aturan yang tidak lengkap atau sistem tidak dapat memberi label untuk aturan frasa tertentu lampiran Gambar 26. Ada beberapa hal yang menyebabkan ini terjadi, yang pertama jumlah aturan pada CFG data latih, cara pembangkitan CFG data latih dari data latih pun mempengaruhi jumlah aturan yang terdapat pada CFG data latih. Pada sistem ini, pembangkitan CFG data latih dilakukan secara langsung, dimana seluruh data latih digunakan sebagai dasar dalam proses pembangkitan CFG data latih tanpa disertai metode lain yang menyertainya. Salah satu metode yang dapat dilakukan untuk memperdalam CFG data latih adalah dengan cara melakukan pengecekan secara semantik. Berbeda dengan sintaktik yang mana mengecek secara aturan-aturan yang ada, semantik digunakan dalam mengecek berdasarkan hubungan arti/makna dari suatu kata/gabungan kata dengan yang lainnya. Yang kedua adalah cara pengecekan data uji. Seperti cara pembangkitan CFG data latih, cara pengecekan pun dilakukan dengan secara langsung tanpa ada metode tambahan sehingga ketika aturan dari sebuah aturan frasa yang dianggap salah tidak ditemukan di CFG data latih, maka sistem tidak dapat memproses lebih dalam lagi dan tidak menghasilkan saran label jenis frasa.

Akurasi yang didapat pada hasil sistem pun masih cukup rentan. Hal ini disebabkan karena sistem saat ini hanya bisa mengubah jenis label frasa pada suatu aturan frasa, tanpa mengubah isi atau struktur dari aturan frasa tersebut. Dibutuhkan penelitian lebih lanjut tentang bagaimana cara melakukan pemberian saran berupa isi atau struktur dari aturan frasa tersebut. Selain itu dengan sistem saat ini hanya bisa melakukan pengecekan dan atau pemberian saran label jenis frasa dimana aturan tersebut independen atau dianggap bahwa aturan frasa lainnya sudah benar. Dalam implementasi dunia nyata, sangat memungkinkan terdapat suatu kondisi dimana anak dari aturan frasa salah, namun jika isi dari aturan frasa tersebut digabungkan secara independen benar.

5. Kesimpulan

Secara sederhana, dengan keadaan sistem saat ini berhasil memperbaiki sebagian aturan frasa dengan cara mengganti label jenis frasa yang salah menjadi label jenis frasa yang benar dimana menaikan nilai akurasi dari data yang diujikan (87%) menjadi (95%). Hasil akurasi yang didapat dari sistem saat ini dan rata-rata responden menunjukkan bahwa sistem ini dapat bersaing dengan keputusan dari rata-rata responden. Hasil dari sistem saat ini masih sangat rentan karena faktor-faktor yang belum dapat dibahas pada tugas akhir ini, dimana beberapa faktor itu adalah :

- Pengecekan aturan frasa berdasarkan makna (semantik).
- Melakukan pendektsian secara terikat antar aturan CFG yang terhubung.
- Memperdalam pengambilan informasi dari data latih ke CFG besar. Cth : probabilistik, artificial-intelligence.
- Memperdalam pengecekan data uji terhadap sistem. Cth : probabilistik, artificial-intelligence.
- Sistem dapat mengecek dan menyarankan struktur pohon.

Daftar Pustaka

- [1] Inacl pos tagging convention konvensi pelabelan kelas kata inacl/malkin. 2017.
- [2] 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, Oct 2016.
- [3] A. Clark, C. Fox, and S. Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. Blackwell Handbooks in Linguistics. Wiley, 2010.
- [4] R. C. Dharmik, R. S. Bhanuse, and A. D. Gaikwad. 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, Feb 2016.
- [5] K. M. A. Hasan, A. Mondal, and A. Saha. A context free grammar and its predictive parser for bangla grammar recognition. In *2010 13th International Conference on Computer and Information Technology (ICCIT)*, pages 87–91, Dec 2010.
- [6] J. E. Hopcroft and J. D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.
- [7] P. P. B. Indonesia. *Pedoman umum ejaan bahasa Indonesia yang disempurnakan / Panitia Pengembangan Bahasa Indonesia, Pusat Pembinaan dan Pengembangan Bahasa*. Departemen Pendidikan dan Kebudayaan Jakarta, 1975.
- [8] L. Kovács and P. Barabás. Experiences in building of context-free grammar tree. In *2011 IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 67–71, Jan 2011.
- [9] R. Manurung. Tutorial: Pengenalan terhadap pos tagging dan probabilistic parsing. 2016.
- [10] T. P. K. P. Pembinaan. Kamus besar bahasa indonesia, 1989.
- [11] H. Riza, M. Purwoadi, Gunarso, T. Uliniansyah, A. A. Ti, S. M. Aljunied, L. C. Mai, V. T. Thang, N. P. Thai, V. Chea, R. Sun, S. Sam, S. Seng, K. M. Soe, K. T. Nwet, M. Utiyama, and C. Ding. Introduction of the asian language treebank. In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 1–6, Oct 2016.
- [12] Samsuri. *Analisis Bahasa*. Erlangga, 1994.
- [13] A. Taylor, M. Marcus, and B. Santorini. *The Penn Treebank: An Overview*, pages 5–22. Springer Netherlands, Dordrecht, 2003.
- [14] G. Z. N. Totok Suhardijanto, Ayu Purwantiari. Inacl syntax parsing convention konvensi pelabelan struktur kalimat inacl/malkin. 2017.
- [15] Y. YUSUF. Analisis pembentukan pola graf pada kalimat bahasa indonesia menggunakan metode knowledge graph, 2014.

Lampiran

Kata Konten (Nomina)

Kata konten adalah kelas kata yang mempunyai makna yang terdapat pada kamus nasional negara tertentu (contoh : KBBI). Kelas konten terutama adalah nomina, verba, adjektiva yang menjadi konsep dalam representasi antarbahasa.

Nomina

Tabel 3. Tabel nomina

kelas kata	Kode	contoh	keterangan
nomina	NNO	buku, mobil, malaikat, pikiran	
nomina nama diri	NNP	Jakarta, Indonesia, Burhan Silalahi	Nomina yang merupakan individu yang unik, misalnya nama kota, nama geografi, nama orang dan sebagainya.

Pronomina

Tabel 4. Tabel pronomina

kelas kata	Kode	contoh	keterangan
pronomina		saya, anda, kamu, sesuatu, seseorang	
pronomina relatif	PRR	yang	

Verba

Tabel 5. Tabel verba

kelas kata	Kode	contoh	keterangan
verba intransitif	VBI	duduk, menangis, bergembira, berlari, bertanam, percaya, tinggal, berasal	
verba transitif	VBT	membaca, menyirami, membelikan, memperistri, memperbarui, memperdayakan, memberlakukan	Verba pasif seperti dipukul, dipenuhi, disembuhkan, terselamatkan, dst masuk ke dalam kategori ini.
verba penghubung	VBL	adalah, [yaitu], ialah, merupakan, [menjadi]	Merupakan verba yang menghubungkan dua bagian, yaitu SUBJEK dan KOMPLEMEN SUBJEK, misalnya pada kalimat “Penduduk Miskin (SUBJ) adalah penduduk yang memiliki rata-rata pengeluaran per kapita per bulan di bawah garis kemiskinan (KOMP).”
verba eksistensial	VBE	ada	Ada merupakan verba eksistensial pada kalimat seperti, “Adegan ini selalu ada di pembukaan Galnas.”

Adjektiva

Tabel 6. Tabel adjektiva

kelas kata	Kode	contoh	keterangan
adjektiva	JJJ	biru, sakit, bersemangat, gelisah	

Adverbia

Tabel 7. Tabel adverbia

kelas kata	Kode	contoh	keterangan
adverbia modal	ADV		
negasi	NEG	tidak, bukan, tak	Kata tiada digolongkan ke dalam verba eksistensial.

Kata Fungsi

Kata fungsi adalah kelas kata yang digunakan untuk menunjukkan hubungan antarkonsep di dalam sebuah kalimat. Beberapa kata fungsi digunakan untuk memperlihatkan aspek dan waktu dalam kalimat.

Konjungsi

Tabel 8. Tabel konungsi

kelas kata	Kode	contoh	keterangan
konjungsi	CNN	dan, atau, tetapi, jika, sejak, meskipun, baik ... maupun, sebaliknya, oleh karena itu	Konjungsi baik ... maupun, bukan (hanya) ... melainkan (juga) ... merupakan konjungsi dengan bentuk terpisah. Lihat penggunaannya dalam kalimat.

Preposisi

Tabel 9. Tabel preposisi

kelas kata	Kode	contoh	keterangan
preposisi	PPN	di, ke, dari, tentang, untuk	

Interjeksi

Tabel 10. Tabel interjeksi

kelas kata	Kode	contoh	keterangan
interjeksi	INT	aduh, astaga, wah	Merupakan kelas kata yang menunjukkan emosi atau perasaan si pembicara/penulis, misalnya pada kalimat, " Wah, Australia akan membuat kereta yang lebih cepat dari pesawat terbang."

Katagorifatis

Tabel 11. Tabel katagorifatis

kelas kata	Kode	contoh	keterangan
katagori fatis	PHA	harap, tolong, maaf	Merupakan kelas kata yang digunakan untuk menyatakan, menciptakan, atau menjaga atmosfer kesamaan perasaan, niat baik, atau keakraban daripada menjadi bagian dari informasi.

Determina

Tabel 12. Tabel determina

kelas kata	Kode	contoh	keterangan
determina	DET	sesuatu, semua, beberapa, beberapa, sebagian, satu, dua, sebuah, seorang, seekor, dll	Yang termasuk ke dalam Determina adalah 1. Kuantifikator; 2. Numeral; 3. Artikel

Partikel

Tabel 13. Tabel partikel

kelas kata	Kode	contoh	keterangan
partikel	PAR	-lah, -kah, pun, -nya	Bentuk terikat -nya yang tidak terkait dengan "dia"/"ia" seperti dalam "Hadirnya pesaing telah diantisipasi PT AHM." dilabeli sebagai determina.

Lambang Lain

Lambang lain merupakan kata-kata atau simbol-simbol yang tidak dapat dimasukan kedalam kata fungsi ataupun kata konten

Bilangan

Tabel 14. Tabel bilangan

kelas kata	Kode	contoh	keterangan
bilangan	123	1, 2, 1/2, 412	

Lambang Satuan

Tabel 15. Tabel lambang satuan

kelas kata	Kode	contoh	keterangan
lambang satuan	UNS	W, kg, km, meter	lambang satuan

Mata Uang

Tabel 16. Tabel mata uang

kelas kata	Kode	contoh	keterangan
mata uang	\$\$\$	\$, Rp	simbol mata uang

Lambang Karakter

Tabel 17. Tabel lambang karakter

kelas kata	Kode	contoh	keterangan
lambang karakter	SYM	? , . ! -	

Daftar Label

Label Katagori Frasa

Tabel 18. Tabel label katagori frasa

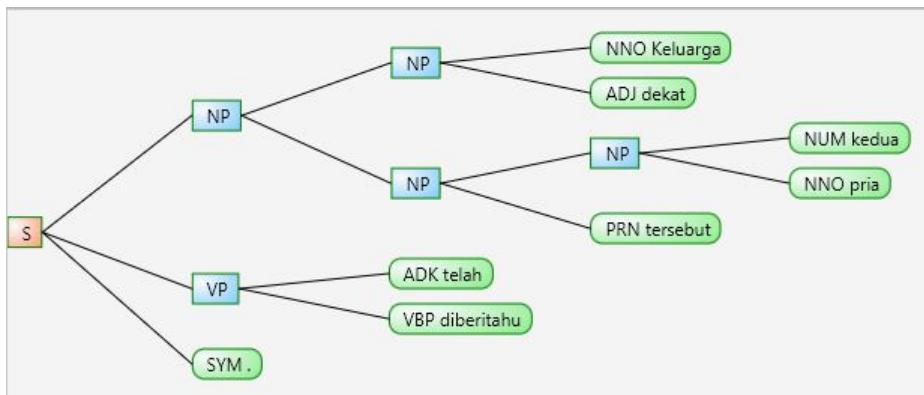
kategori frasa	Kode	struktur	contoh	kata	keterangan
Frasa Adjektiva	ADJP	(ADV)+ADJ	sangat <i>cantik</i> sekali cukup <i>lumayan</i>		Frasa yang intinya berupa adjektiva
Frasa Adverbial	ADVP	ADV ADK NEG+(PP)	kurang lebih sedikit banyak agak terlalu		Frasa yang intinya berupa adverbia yang menjelaskan selain
Frasa Nominal	NP	NNO+(DET) (DET)+NNO NNO+PP NNO+(NNO) NNP	kegagalan ini satu poin percobaan di menit ke-33 lokasi proyek andrea masih		Frasa yang intinya berupa nomina
Frasa preposisional	PP	PPN+NP	pada lokasi proyek		Frasa nomina yang didahului dengan preposisi
Frasa Verbal	VP	(ADV)+VBI+ +(ADV) (ADM)+VBI+ +(ADV) (ADV)+VBT+ +(NP)+(ADV) (ADM)+VBT+ +(NP)+(ADV) VBI+VBI VBI+VBT	diberikan akses terbatas pada lokasi proyek		Frasa yang terdiri dari frasa verbal dan keterangan(<i>adjunct</i>)
Klausa Terikat	CP	PPN+SBAR PRI+SBAR	<i>bahwa</i> gempa telah menghantam Chittagong	<i>bahwa</i> , <i>yang</i> , <i>di mana</i> , <i>bagaimana</i> , <i>mengapa</i> , <i>kapan</i> , <i>siapa</i> , <i>berapa</i> , <i>apa</i> , <i>apakah</i>	Klausa yang terdiri dari subjek, predikat, dan objek serta jika ada, yang didahului dengan preposisi <i>bahwa</i> , ataupun kata tanya yang membentuk klausa tanya tak langsung.
Klausa Relatif	RPN	PRR+VP PRR+SBAR	proyek <i>yang masih berjalan</i>	<i>yang</i> , <i>tempat</i> , <i>di mana</i>	Klausa relatif yang menjadi keterangan dari frasa nominal
Kalimat	S	SBAR+(ADV)	publik diberikan akses terbatas pada lokasi proyek		kalimat lengkap
Kalimat Tanya	SQ	PRI+S+SYM?	mengapa anda melakukannya itu?	<i>apa</i> , <i>apakah</i> , <i>siapa</i> , <i>kapan</i> , <i>dimana</i> , <i>bagaimana</i> , <i>mengapa</i> , <i>bila</i> , <i>berapa</i> , <i>kenapa</i> , <i>-kah</i>	kalimat tanya langsung yang didahului dengan kata tanya dan diakhiri dengan simbol tanda tanya.
Klausa Inti	SBAR	NP+VP	publik diberikan akses terbatas		Klausa inti yang terdiri dari subjek, predikat, dan objek jika ada, tanpa keterangan.

Label Penggabungan Kata

Tabel 20. Tabel label penggabungan kata

label penggabungan kata	Kode	contoh	keterangan
Penggabungan Kelas Kata Nomina Umum	+NNO	rumah, sakit > rumah sakit kambing, hitam > kambing hitam olah, raga > olah raga	Kata majemuk yang berupa nomina umum.
Penggabungan Kelas Kata Nomina Nama Diri	+NNP	Raisa, Andriana > Raisa Andriana Tanjung, Pinang > Tanjung Pinang TNI, AL > TNI AL	Kata majemuk yang berupa nomina nama diri, misalnya nama orang, nama geografis, nama organisasi.
Penggabungan Kelas Kata Preposisi	+PPO	di, atas > di atas di, antara > di antara	Kata majemuk yang berupa preposisi.
Penggabungan Kelas Kata Konjungsi Koordinatif	+CCN	akan, tetapi > akan tetapi selain, itu > selain itu	Kata majemuk yang berupa konjungsi koordinatif.
Penggabungan Kelas Kata Konjungsi Subordinatif	+CSN	meskipun, demikian > meskipun demikian oleh, karena, itu > oleh karena itu	Kata majemuk yang berupa konjungsi koordinatif.

Ilustrasi Teori Bahasa

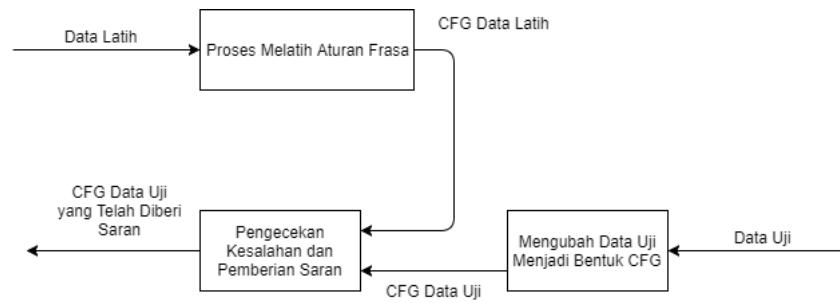


Gambar 2. Pohon Kalimat

$S \rightarrow NP \ VP \ SYM$
 $NP \rightarrow NP \ NP$
 $NP \rightarrow NNO\text{-Keluarga} \ ADJ\text{-dekat}$
 $NP \rightarrow NP \ PRN\text{-tersebut}$
 $NP \rightarrow NUM\text{-kedua} \ NNO\text{-pria}$
 $VP \rightarrow ADK\text{-telat} \ VBP\text{-diberitahu}$

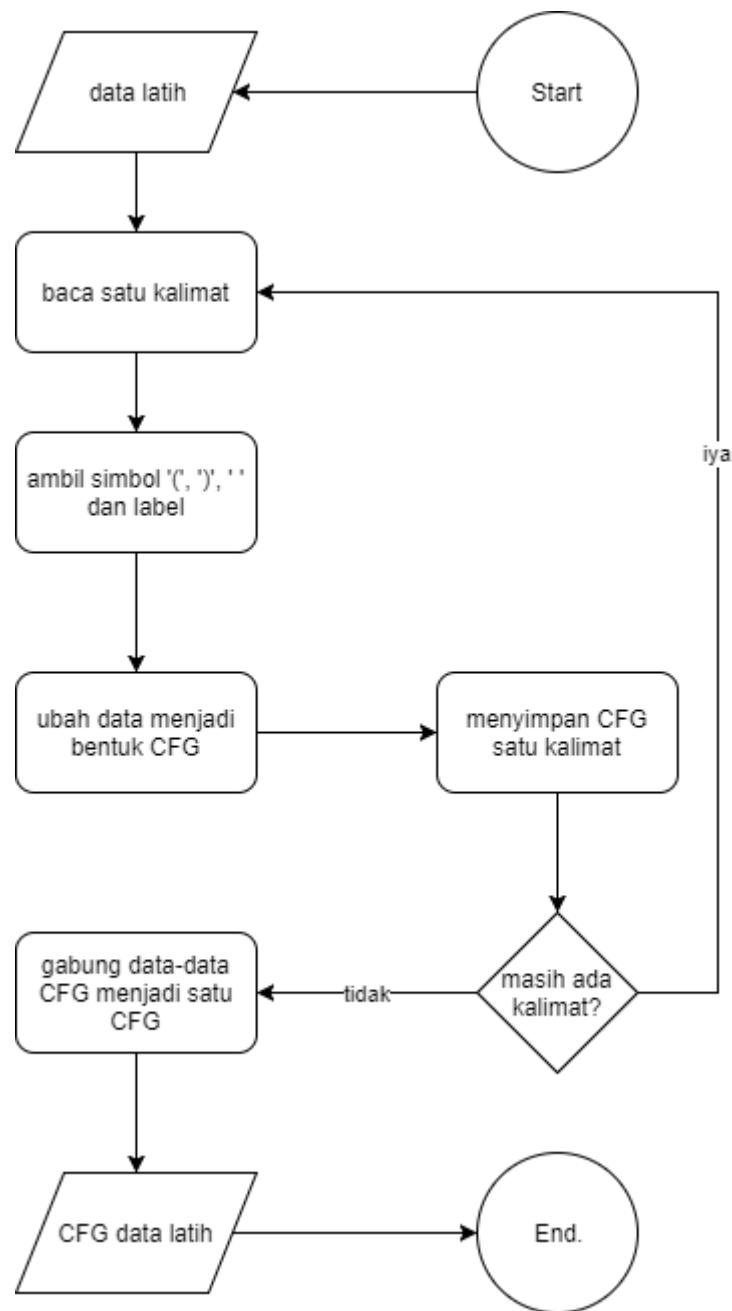
Gambar 3. Pohon Kalimat (Bentuk Lain)

Flowchart Sistem Secara Umum



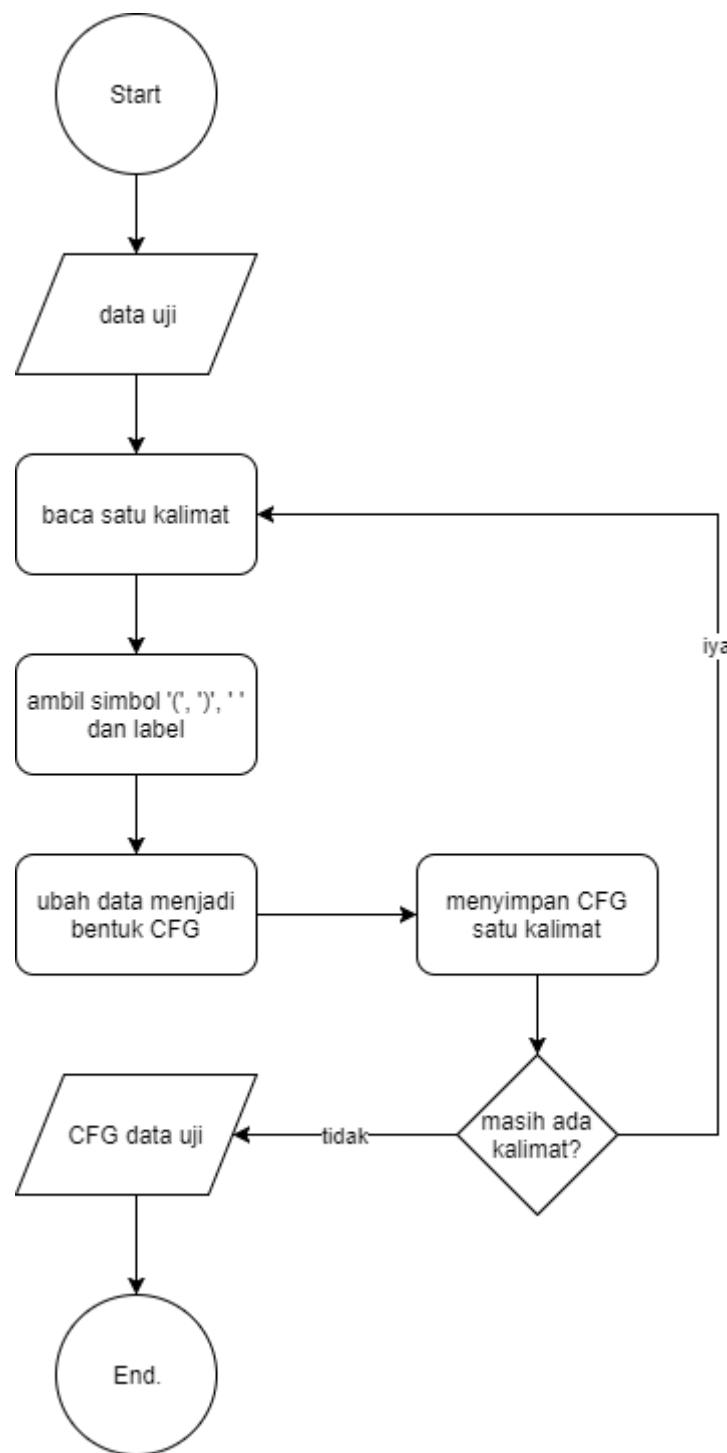
Gambar 4. Flowchart Umum Sistem

Flowchart Proses Melatih Aturan Frasa Sistem



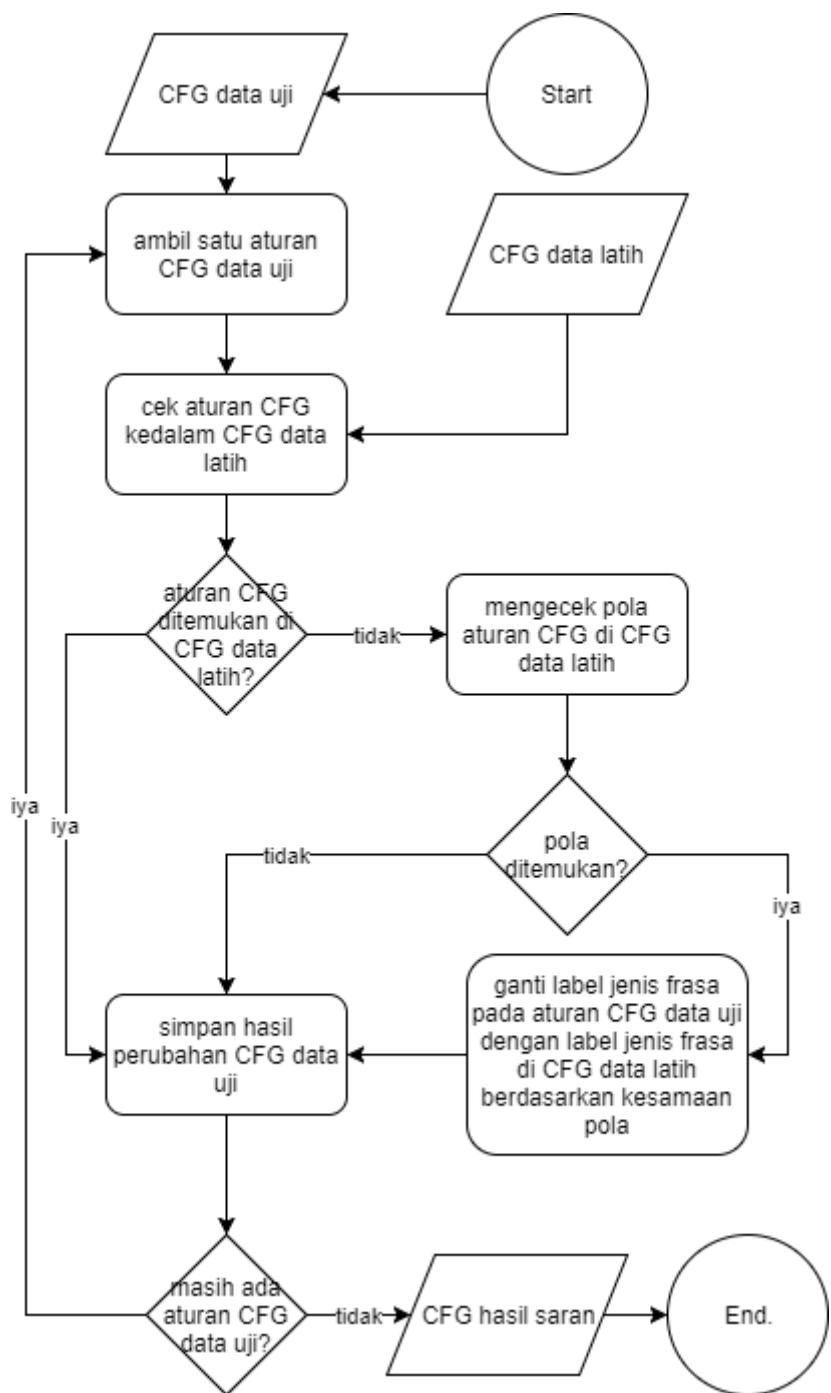
Gambar 5. Flowchart Proses Melatih Aturan Frasa Sistem

Flowchart Mengubah Data Uji Menjadi Bentuk CFG



Gambar 6. Flowchart Mengubah Data Uji Menjadi Bentuk CFG

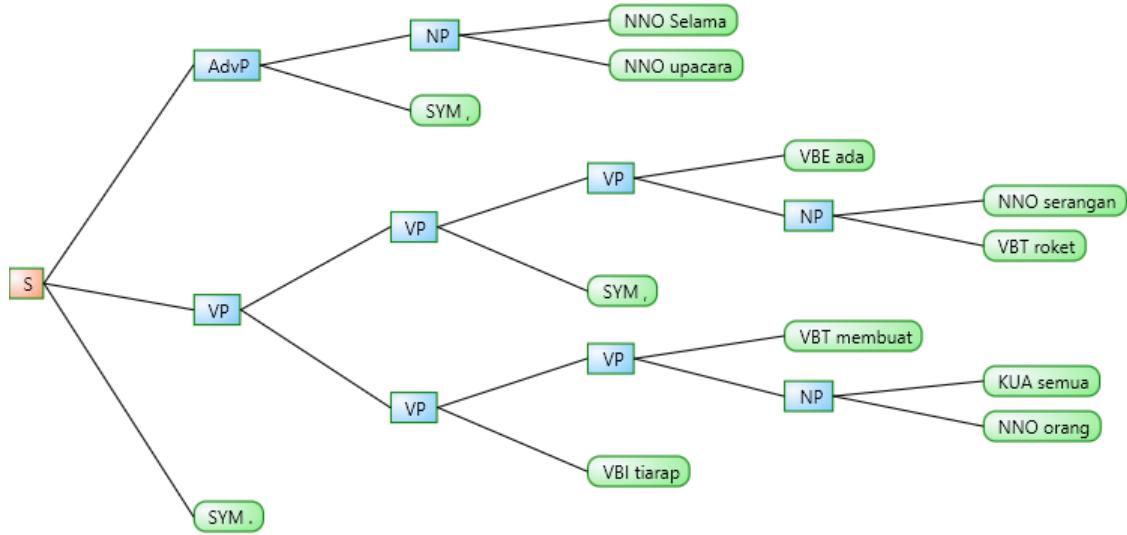
Flowchart Pengecekan dan Pemberian Saran



Gambar 7. Flowchart Pengecekan dan Pemberian Saran

Contoh Data Input Output Proses Melatih Aturan Frasa Sistem

Kalimat 1 : Selama upacara , ada serangan roket , membuat semua orang tiarap .



Gambar 8. Kalimat 1 : Bentuk pohon

```
(S (AdvP (NP (NNO Selama) (NNO upacara)) (SYM ,)) (VP (VP (VP (VBE ada) (NP (NNO serangan) (VBT roket))) (SYM ,)) (VP (VP (VBT membuat) (NP (KUA semua) (NNO orang))) (VBI tiarap))) (SYM .)))
```

Gambar 9. Kalimat 1 : Bentuk teks

```
(S (AdvP (NP (NNO) (NNO)) (SYM)) (VP (VP (VP (VBE) (NP (NNO) (VBT))) (SYM)) (VP (VP (VBT) (NP (KUA) (NNO)))) (VBI))) (SYM ))
```

Gambar 10. Kalimat 1 : Bentuk label teks

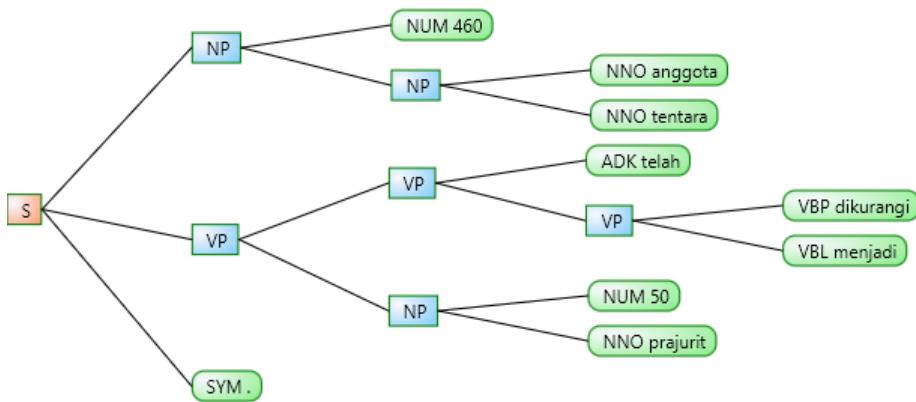
NP → NNO NNO
AdvP → NP SYM
NP → NNO VBT
VP → VBE NP
VP → VP SYM
NP → KUA NNO
VP → VBT NP
VP → VP VBI
VP → VP VP
S → AdvP VP SYM

Gambar 11. Kalimat 1 : Bentuk CFG

NP → NNO NNO
AdvP → NP
NP → NNO VBT
VP → VBE NP
VP → VP
NP → KUA NNO
VP → VBT NP
VP → VP VBI
VP → VP VP

Gambar 12. Kalimat 1 : Bentuk CFG 2

Kalimat 2 : 460 anggota tentara telah dikurangi menjadi 50 prajurit .



Gambar 13. Kalimat 2 : Bentuk pohon

(S (NP (NUM 460) (NP (NNO anggota) (NNO tentara))) (VP (VP (ADK telah) (VP (VBP dikurangi) (VBL menjadi)))) (NP (NUM 50) (NNO prajurit))) (SYM .))

Gambar 14. Kalimat 2 : Bentuk teks

(S (NP (NUM) (NP (NNO) (NNO))) (VP (VP (ADK) (VP (VBP) (VBL))) (NP (NUM) (NNO))) (SYM))

Gambar 15. Kalimat 2 : Bentuk label teks

NP → NNO NNO
NP → NUM NP
VP → VBP VBL
VP → ADK VP
NP → NUM NNO
VP → VP NP
S → NP VP SYM

Gambar 16. Kalimat 2 : Bentuk CFG

NP → NNO NNO
NP → NUM NP
VP → VBP VBL
VP → ADK VP
NP → NUM NNO
VP → VP NP

Gambar 17. Kalimat 2 : Bentuk CFG 2

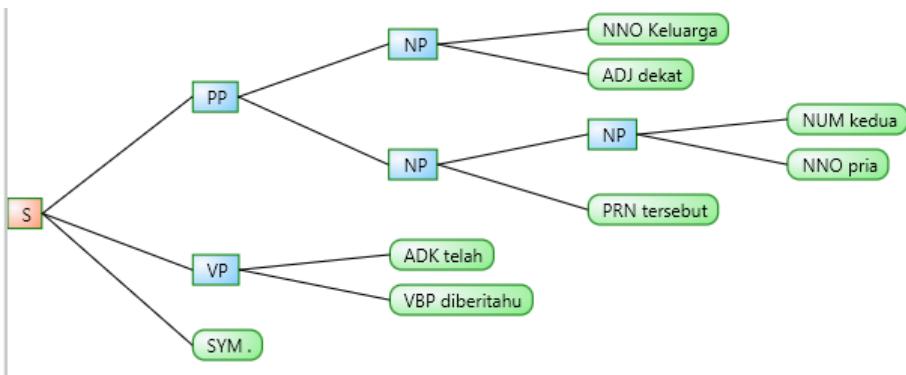
Hasil penggabungan 2 CFG :

NP → NUM NP
VP → VBP VBL
VP → ADK VP
NP → NUM NNO
VP → VP NP
NP → NNO NNO
AdvP → NP
NP → NNO VBT
VP → VBE NP
VP → VP
NP → KUA NNO
VP → VBT NP
VP → VP VBI
VP → VP VP

Gambar 18. Gabungan CFG dari 2 kalimat

Contoh Data Input Output Proses Merubah Data Uji Menjadi Bentuk CFG

Kalimat data uji : Keluarga dekat kedua pria tersebut telah diberitahu .



Gambar 19. Kalimat data uji : Bentuk pohon

(S (PP (NP (NNO Keluarga) (ADJ dekat)) (NP (NP (NUM kedua) (NNO pria)) (PRN tersebut))) (VP (ADK telah) (VBP diberitahu)) (SYM .))

Gambar 20. Kalimat data uji : Bentuk teks

(S (PP (NP (NNO) (ADJ)) (NP (NP (NUM) (NNO)) (PRN))) (VP (ADK) (VBP)) (SYM))

Gambar 21. Kalimat data uji : Bentuk label teks

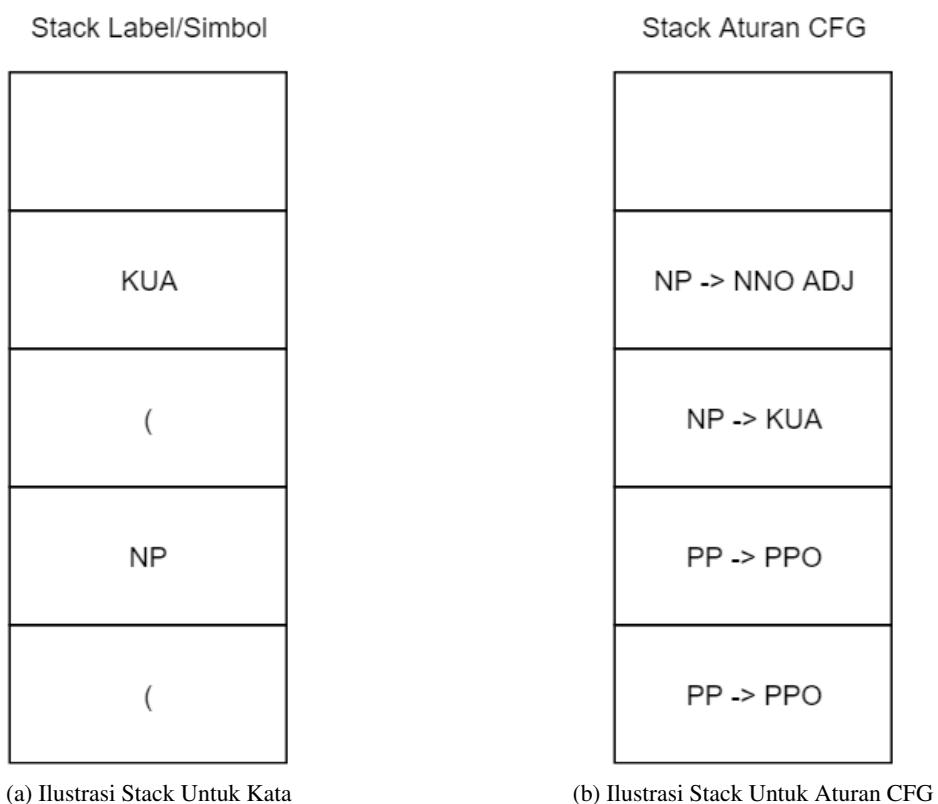
NP → NNO ADJ
NP → NUM NNO
NP → NP PRN
PP → NP NP
VP → ADK VBP
S → PP VP SYM

Gambar 22. Kalimat data uji : Bentuk CFG

NP → NNO ADJ
NP → NUM NNO
NP → NP PRN
PP → NP NP
VP → ADK VBP

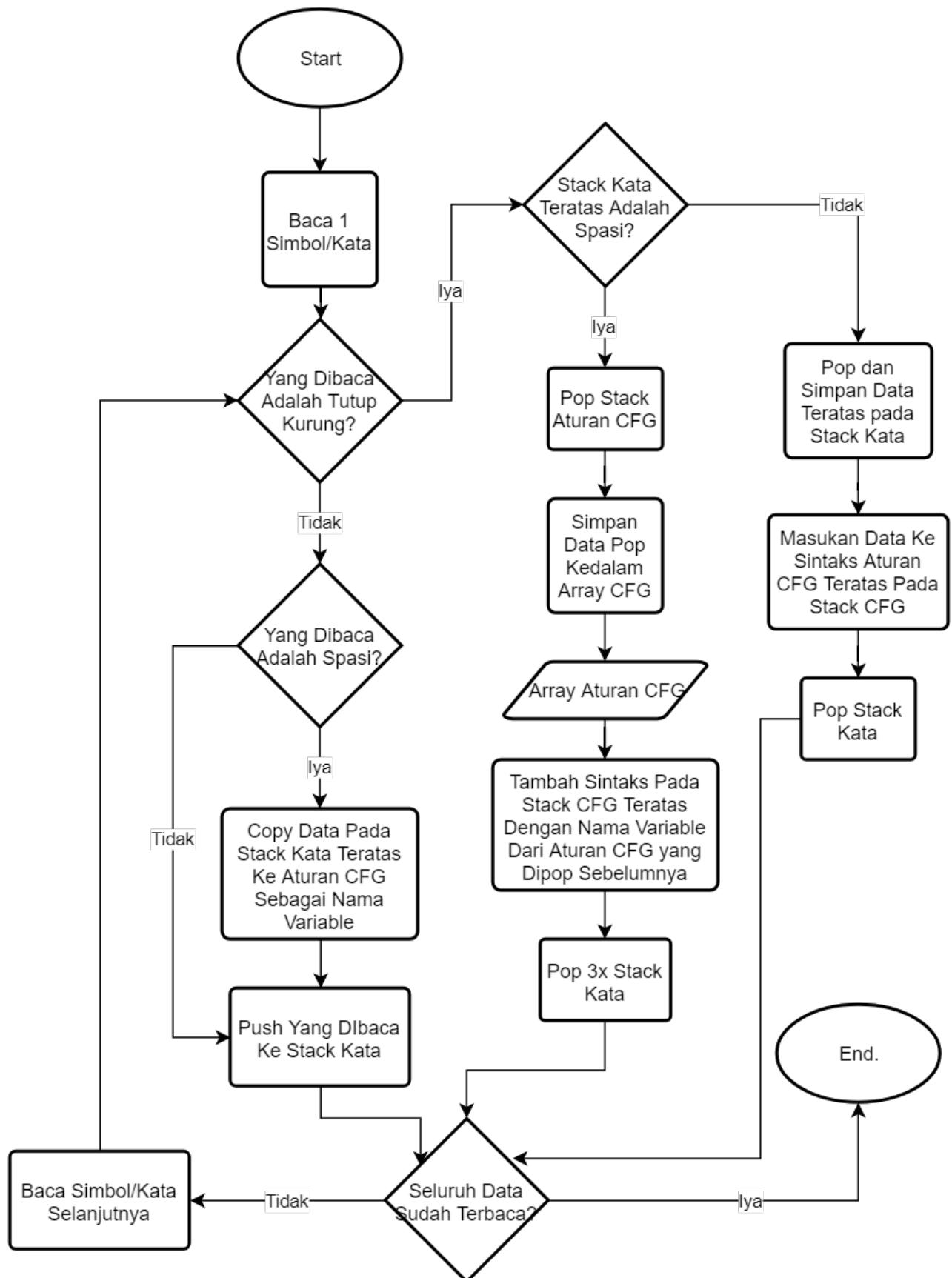
Gambar 23. Kalimat data uji : Bentuk CFG 2

Format Stack per Kata dan per Aturan CFG



Gambar 24. Ilustrasi Stack Kata dan Aturan CFG Data Latih dan Data Uji

Flowchart Cara Ubah Data ke Bentuk CFG



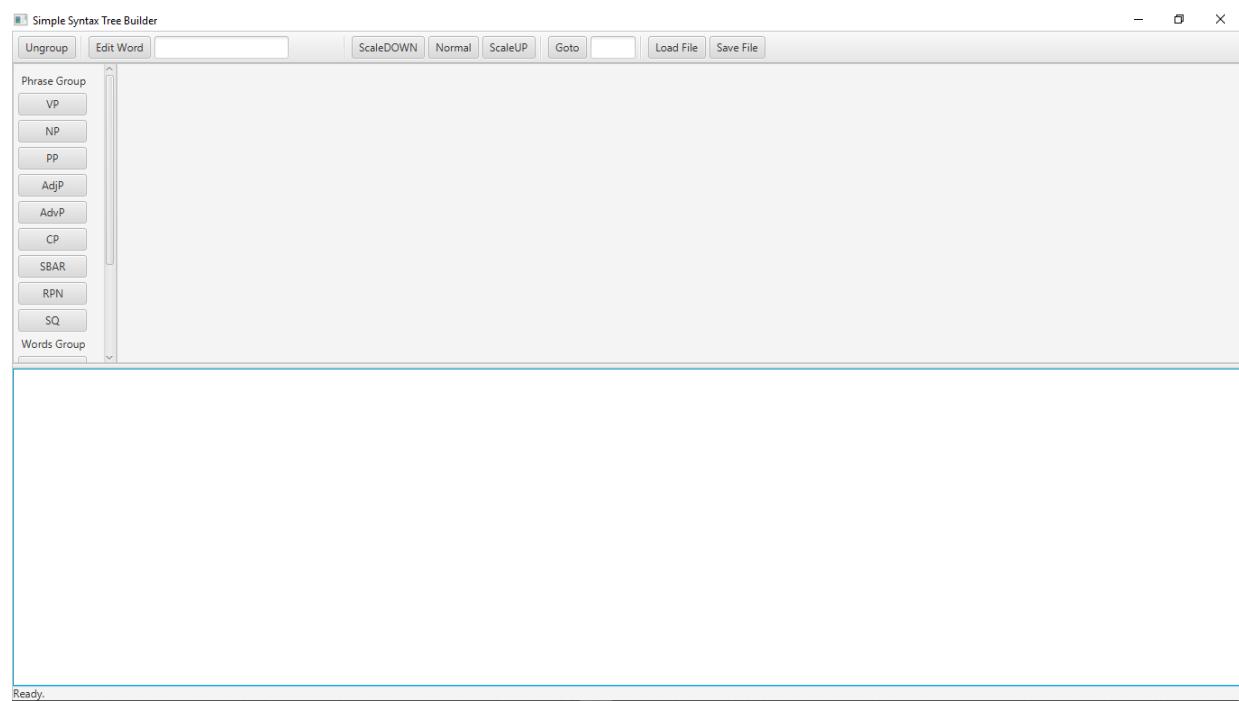
Gambar 25. Flowchart Cara Kerja Mengubah Data Ke CFG

Daftar Aturan yang Gagal Diubah oleh Sistem

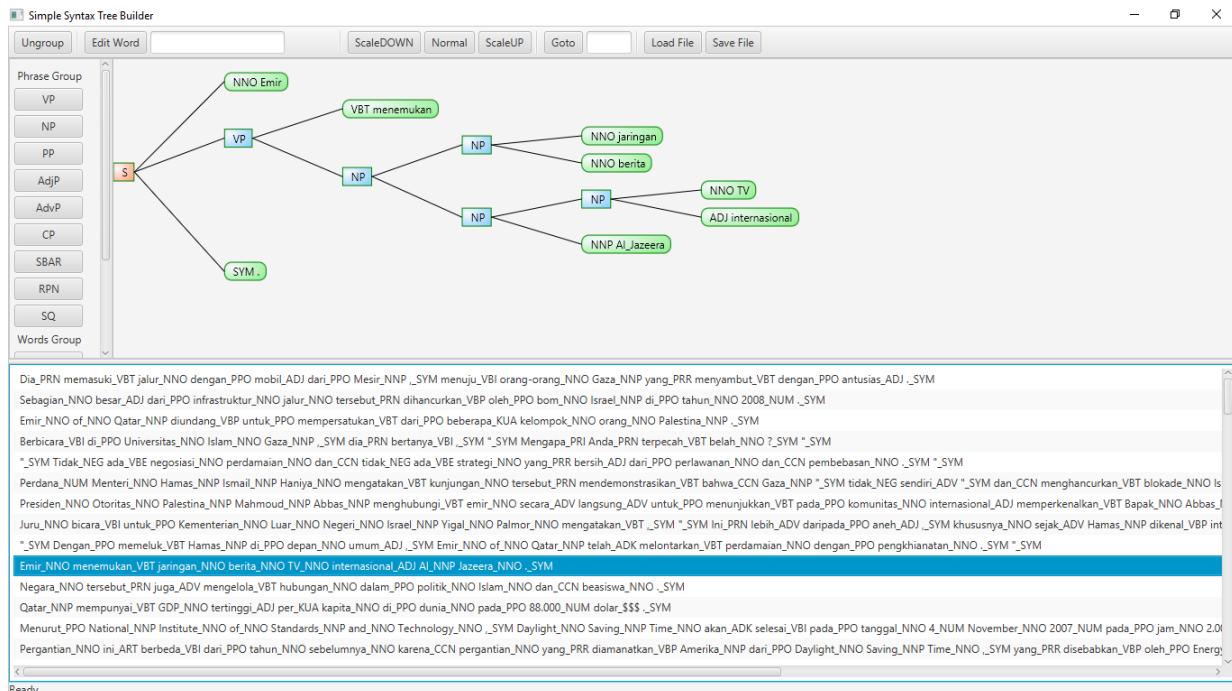
No	Nomor Kalimat	Aturan Valid	Aturan Awal	Aturan Hasil
1	Kalimat 16	VP -> VP VBP	CP -> VP VBP	CP -> VP VBP
2	Kalimat 16	VP -> VP PP	VP -> CP PP	VP -> CP PP
3	Kalimat 18	RPN -> PRR VP	VP -> PRR VP	VP -> PRR VP
4	Kalimat 18	NP -> NP RPN	NP -> NP VP	NP -> NP VP

Gambar 26. Daftar aturan yang gagal diubah oleh sistem

TIKSentence TreeMaker



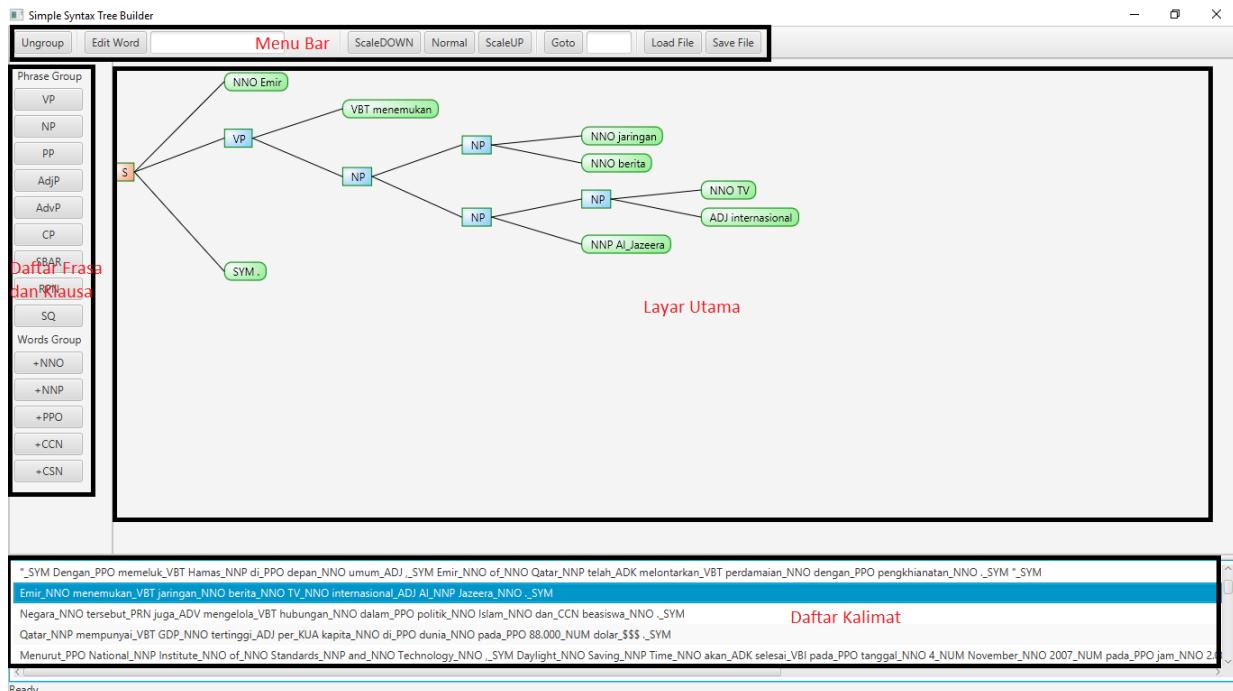
Gambar 27. Tampilan aplikasi ketika baru dibuka



Gambar 28. Tampilan aplikasi ketika melakukan labelling



Gambar 29. Tampilan aplikasi ketika kalimat belum diberi label jenis frasa



Gambar 30. Tampilan aplikasi ketika kalimat telah diberi label jenis frasa

Tabel Klasifikasi Kebenaran Recall-Precision

Tabel 22. Tabel Plot Klasifikasi Recall and Precision

Nilai Data Sebelum Dimasukkan ke Sistem	Nilai Data Setelah Dimasukkan ke Sistem	Klasifikasi
True	True	True Negative (TN)
True	False	False Negative (FN)
False	True	True Positive (TP)
False	False	False Positive (FP)

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (3)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (4)$$

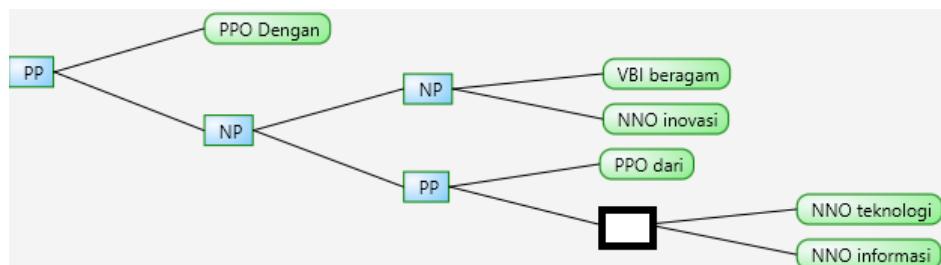
$$F1\ Score = 2x(\frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}) \times 100\% \quad (5)$$

Detail Tabel Pengujian

Tabel 23. Detail Accuracy, Recall dan Precision, F1 score table

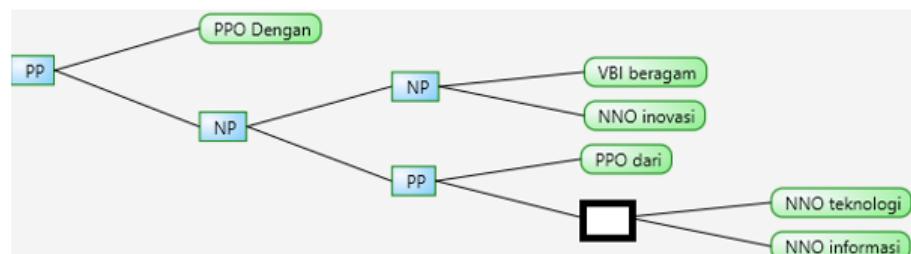
Data test	Accuracy	Recall	Precision	F1 Score
Lower Bound	80.00%	Na	Na	Na
UpperBound	100.00%	100.00%	100.00%	100.00%
Initial data test	87.88%	Na	Na	Na
Output from system	95.84%	100.00%	65.71%	79.31%
First respondent	93.77%	90.47%	54.27%	67.85%
Second respondent	94.46%	88.00%	62.85%	73.33%
Third respondent	91.34%	64.70%	62.85%	63.76%
Fourth respondent	96.88%	86.11%	88.57%	87.32%
Fifth respondent	84.42%	29.16%	20.00%	23.72%
Sixth respondent	92.38%	68.57%	68.57%	68.57%
Seventh respondent	94.80%	83.33%	71.42%	76.92%
Eighth respondent	91.34%	63.15%	68.57%	65.75%
Ninth respondent	99.30%	97.14%	97.14%	97.14%
Tenth respondent	83.39%	25.92%	20.00%	22.58%
Eleventh respondent	92.04%	70.00%	60.00%	64.61%
Twelfth respondent	86.85%	42.85%	25.71%	32.14%

Contoh Pertanyaan Survey



Gambar 31. Contoh Pohon yang Ditutup Untuk Survey

1. Tentukan Jenis frasa yang tepat pada kotak kosong



- A. VP
- B. NP
- C. PP
- D. AdjP
- E. AdvP
- F. CP
- G. SBAR
- H. RPN
- I. SQ

Gambar 32. Contoh Pertanyaan pada Survey

Riwayat Hidup Singkat

Data Pribadi

Nama: Totok Suhardijanto, M.Hum., Ph.D.
Institusi: Departemen Linguistik, Fakultas Ilmu Pengetahuan Budaya,
Universitas Indonesia
NIDN: 0321066905
NUP: 0706050101
Golongan/Ruang: III/c
NPWP: 09.472.789.8-005.000

Latar Belakang Pendidikan

1. S1: Sarjana Sastra, Program Studi Sastra Indonesia, Fakultas Sastra Universitas Indonesia
2. S2: Magister Humaniora, Program Ilmu Linguistik, Program Pascasarjana Universitas Indonesia
3. S3: Ph.D. in Information Processing, Graduate School of Media and Governance, Keio University.

Pengalaman Bekerja

1. Pengajar pada Departemen Linguistik, Fakultas Ilmu Pengetahuan Budaya, Universitas Indonesia: 1996—sekarang
2. Pengajar pada Departemen Linguistik, Fakultas Ilmu Pengetahuan Budaya, Universitas Indonesia: 2004—sekarang
3. Penyunting naskah, Oxford Global Languages, Indonesian, Oxford University Press: 2016—sekarang.
4. Project Manager, Google Speech, Google Asia Pacific: 2013—2015
5. Pengajar Tamu pada Keio University: 2002—2012
6. Pengajar Tidak Tetap pada Tokyo University of Foreign Studies: 2003—2012
7. Pengajar Tidak Tetap pada Faculty of Bioresources, Nihon University, Fujisawa, Jepang: 2010—2012

Buku, Artikel, dan Makalah Ilmiah dalam 2 Tahun Terakhir

1. “The Use of Javanese in Social Media: A structural approach on language variations in the Internet.” Makalah dipresentasikan pada APRISH 2017, Hotel Margo, Depok, 27-29 September 2017.
2. “Building A Collaborative Workspace for Lexicographic Works in Indonesia.” Makalah dipresentasikan pada Electronic Lexicography, Holiday Inn, Leiden, 19-21 September 2017.

3. "Meninjau Ulang Kamus Ungkapan Bahasa Indonesia (1997): Upaya Merekam dan Mengemas Ekspresi Masyarakat dalam Karya Leksikografis". Seminar Nasional Leksikografi, Hotel Aryaduta, Jakarta, 9-11 Agustus 2017.
4. "Sumber Daya Bahasa: Pengembangan Bank Struktur Bahasa Indonesia." Makalah pada Seminar Internasional Leksikologi dan Leksikografi, Jakarta, 2 Mei 2016
5. "Developing Rule-based POS Tagger for Javanese: The First Stage." Makalah pada International Seminar on Languages of Java, Semarang, 18-19 Mei 2016
6. "Knowledge Gaps in Contemporary Javanese Comprehensive Description: Towards the Making of Javanese Annotated Corpus." Makalah pada International Seminar on Languages of Java, Semarang, 18-19 Mei 2016.
7. "Rumpang Pengetahuan dalam Deskripsi Komprehensif Bahasa Jawa Kontemporer: Ke Arah Penyusunan Korpus Beranotasi Bahasa Jawa Periode 1945-2015." Makalah pada Seminar Nasional Bahasa Ibu, Badan Pengembangan dan Pembinaan Basa, 21 Februari 2017.
8. "Peranan Korpus dalam Penyusunan Kamus Bahasa Pergaulan Remaja." Makalah pada International Seminar on Transdisciplinary Linguistics, Fakultas Ilmu Pengetahuan Budaya Universitas Indonesia, 21 November 2016 (penulis bersama dengan Dien Rovita).
9. "Developing language resources for under-resourced languages in Indonesia." Makalah pembicara kunci pada International Conference on Knowledge Creation and Intelligence Computing 2016, Politeknik Negeri Manado, Sulawesi Utara, 15—17 November 2016.
10. "The Collocation and Grammatical Behaviour of Two Nouns Denoting Woman in Sundanese: A Corpus-based Analysis of Language and Gender Relationship." Makalah pada the Asia Pacific Research in Social Sciences and Humanities, Hotel Margo, Depok, Jawa Barat, 27—29 September 2016 (Penulis bersama dengan Prayogo Geseit Bagasworo).
11. "Metaphors of Women in Sundanese Magazine (Manglè, 1958-2013): A Corpus-Based Approach." Makalah pada the Asia Pacific Research in Social Sciences and Humanities, Hotel Margo, Depok, Jawa Barat, 27—29 September 2016 (Penulis bersama dengan Adhitya Alkautsar).
12. "Mekanisme Pengolahan dan Penyajian Kosakata Budaya dalam Kamus Pemelajar BIPA." Makalah pada Seminar Nasional Leksikografi Indonesia, Hotel Santika TMII, Jakarta, 26—29 Juli 2016. (penulis bersama dengan Atin Fitriana dan Dien Rovita)
13. "Pengembangan WordNet Bahasa-Bahasa Daerah di Indonesia: Kasus WordNet Bahasa Jawa." Makalah pada Seminar Nasional Leksikografi Indonesia, Hotel Santika TMII, Jakarta, 26—29 Juli 2016.
14. "Penelusuran Kekerabatan Bahasa-Bahasa Austronesia dengan Pendekatan Komputasional: Penerapan Metode Penajaran Rangkaian Ganda." Makalah pada Seminar Internasional Migrasi Austronesia. Hotel Mercure, Ancol, Jakarta, Indonesia, 14—16 September 2016.
15. "Wordnet Bahasa Jawa: Pangkalan Data Leksikal Semantik Elektronik dalam Bahasa Jawa." Artikel pada M. Umar Muslim, R. Niken Pramanik, Dewaki Kramadibrata, dan Silva Tenrisara Pertiwi Isma (ed.). *Mahaguru yang Bersahaja: Persembahan untuk Prof. Dr. Muhamadjir*. Fakultas Ilmu Pengetahuan Budaya, Universitas Indonesia, 2016.
16. "Creating Javanese WordNet: Between Challenge and Opportunity." Makalah pembicara undangan pada the 2nd Workshop on WordNet Bahasa, Nanyang Technological University, Singapore, January 15-16 2016.

17. "Kemampuan Membaca Anak Berdasarkan Gender dan Preferensi Gawai." Makalah pada Seminar Nasional Sosiolinguistik dan Dialektologi, Universitas Indonesia, 9-10 November 2015 (penulis artikel bersama dengan Fierenziana Getruida Junus dan Zahroh Nuriah).
18. "Pengaruh Kebiasaan Penggunaan Gawai terhadap Kualitas Membaca: Studi Kasus". Makalah pada ICLCS, LIPI, Jakarta, November 25-26, 2015 (penulis artikel bersama Zahroh Nuriah dan Fierenziana Getruida Junus).
19. "Linguistik interdisipliner di Indonesia: Selayang pandang." Makalah pembicara undangan pada PESAT, Universitas Gunadarma, October 20, 2015.
20. "Klasifikasi Bahasa, Geometri, dan Similaritas: Upaya Rekonstruksi Kekerabatan Bahasa dengan Komputasi Ruang Vektor." Makalah pembicara undangan pada the International Seminar on Austronesian Languages and Literature, Universitas Udayana, Bali, 28-29 Agustus, 2015.

```
Aturan.java
```

```
/*
```

```
* To change this license header, choose License Headers in Project Properties.
```

```
* To change this template file, choose Tools | Templates
```

```
* and open the template in the editor.
```

```
*/
```

```
package Proses.Kelas;
```

```
import java.util.ArrayList;
```

```
/**
```

```
*
```

```
* @author Mukhtar
```

```
*/
```

```
public class Aturan implements Cloneable{
```

```
    private String namaTag = "";
```

```
    private ArrayList<Aturan> Sintaks = new ArrayList<>();
```

```
    private boolean isVisited = false;
```

```
    public Aturan(String value, ArrayList<Aturan> aturan) {
```

```
        this.namaTag = value;
```

```
        this.Sintaks = aturan;
```

```
}
```

```
    public Aturan(String value) {
```

```
        this.namaTag = value;
```

```
        this.Sintaks = new ArrayList<>();
```

```
}
```

```
    public Aturan() {
```

```
        this.namaTag = "";
        this.Sintaks = new ArrayList<>();
    }

    public String getNamaTag() {
        return namaTag;
    }

    public void setNamaTag(String namaTag) {
        this.namaTag = namaTag;
    }

    public ArrayList<Aturan> getSintaks() {
        return Sintaks;
    }

    public void setSintaks(ArrayList<Aturan> Sintaks) {
        this.Sintaks = Sintaks;
    }

    public void addSintaks(Aturan data) {
        this.Sintaks.add(data);
    }

    public void setVisited(boolean value) {
        this.isVisited = value;
    }

    public boolean getVisited() {
        return this.isVisited;
    }
```

```
public Object clone() throws CloneNotSupportedException {  
    return super.clone();  
}  
}
```

```
SatuKalimat.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Proses.Kelas;

import java.util.ArrayList;

/**
 *
 * @author Mukhtar
 */
public class SatuKalimat {

    private String kalimat = "";
    private ArrayList<Aturan> listCFG = new ArrayList<>();
    private ArrayList<Integer> listNilai = new ArrayList<>();

    public SatuKalimat() {
        this.kalimat = "===";
        this.listCFG = new ArrayList<>();
        this.listNilai = new ArrayList<>();
    }

    public SatuKalimat(String kalimat, ArrayList<Aturan> list) {
        this.kalimat = kalimat;
        this.listCFG = list;
    }
}
```

```
public SatuKalimat(String kalimat, ArrayList<Aturan> list, ArrayList<Integer> listnilai) {  
    this.kalimat = kalimat;  
    this.listCFG = list;  
    this.listNilai = listnilai;  
}  
  
public String getKalimat() {  
    return kalimat;  
}  
  
public void setKalimat(String kalimat) {  
    this.kalimat = kalimat;  
}  
  
public ArrayList<Aturan> getListCFG() {  
    return listCFG;  
}  
  
public void setListCFG(ArrayList<Aturan> listCFG) {  
    this.listCFG = listCFG;  
}  
  
public void addCFG(Aturan cfg, int nilai) {  
    this.listCFG.add(cfg);  
    this.listNilai.add(nilai);  
}  
  
public ArrayList<Integer> getListNilai() {  
    return listNilai;  
}
```

```
public void setListNilai(ArrayList<Integer> listNilai) {  
    this.listNilai = listNilai;  
}  
}
```

```
SatuFile.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Proses.Kelas;

import java.util.ArrayList;

/**
 *
 * @author Mukhtar
 */
public class SatuFile implements Cloneable{

    private String namaFile = "";
    private ArrayList<SatuKalimat> listKalimat = new ArrayList<>();
    private double datahasil = 0.0;
    private int dataCount = 0;
    private ArrayList<SatuKalimat> listKalimatSalah = new ArrayList<>();

    public SatuFile() {
        this.namaFile = "";
        this.listKalimat = new ArrayList<>();
    }

    public SatuFile(String namaFile, ArrayList<SatuKalimat> list) {
        this.namaFile = namaFile;
        this.listKalimat = list;
    }
}
```

```
public String getNamaFile() {
    return namaFile;
}

public void setNamaFile(String namaFile) {
    this.namaFile = namaFile;
}

public ArrayList<SatuKalimat> getListKalimat() {
    return listKalimat;
}

public void setListKalimat(ArrayList<SatuKalimat> listKalimat) {
    this.listKalimat = listKalimat;
}

public void addKalimat(SatuKalimat value) {
    this.listKalimat.add(value);
}

public double getDatahasil() {
    return datahasil;
}

public void setDatahasil(double datahasil) {
    this.datahasil = datahasil;
}

public int getDataCount() {
    return dataCount;
}
```

```
}

public void setDataCount(int dataCount) {
    this.dataCount = dataCount;
}

public ArrayList<SatuKalimat> getListKalimatSalah() {
    return listKalimatSalah;
}

public void setListKalimatSalah(ArrayList<SatuKalimat> listKalimatSalah) {
    this.listKalimatSalah = listKalimatSalah;
}

public void addListKalimatSalah(SatuKalimat value) {
    listKalimatSalah.add(value);
}

@Override
public Object clone() throws CloneNotSupportedException {
    return super.clone();
}
```

kelasTest.java

```
import DoEverythingHere.Praproses;
import DoEverythingHere.ProsesPembuatanPohonSintaks;
import Evaluasi.EvaluasiRecallPrecision;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Mukhtar
 */
public class kelasTest {
    public static void main(String[] args) {
        Praproses Praproses = new Praproses();
        Praproses.doPraproses();
        Praproses.hapusTagS();
        Praproses.printAll();
        ProsesPembuatanPohonSintaks dataAktual = new ProsesPembuatanPohonSintaks();

        dataAktual.doProses("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\aktual");
        dataAktual.hapusTagS();
        dataAktual.printAll2();
        ProsesPembuatanPohonSintaks dataAktual2 = new ProsesPembuatanPohonSintaks();

        dataAktual2.doProses("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\aktual");
```

```
dataAktual2.hapusTagS();
dataAktual2.printAll2();
ProsesPembuatanPohonSintaks dataValid = new ProsesPembuatanPohonSintaks();

dataValid.forDataValid("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\d
ataset\\\\valid");
dataValid.hapusTagS();
dataValid.printAll2();

EvaluasiRecallPrecision eval = new EvaluasiRecallPrecision();
eval.setData(Praproses.getNormalisasi(), dataValid.getData(), dataAktual.getData(),
dataAktual2.getData());
eval.Evaluasi();
}

}
```

```
ImportExportFile.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Proses;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Mukhtar
 */
public class ImportExportFile {
    public ImportExportFile() {
        data = new ArrayList<>();
    }
    private ArrayList<String> data;
    public void ExportFile(String file) {
        try {
            FileOutputStream fos = new FileOutputStream(file);

```

```
        DataOutputStream dos = new DataOutputStream(fos);

        for (String s : data) {
            dos.writeBytes(s + System.getProperty("line.separator"));
        }

    } catch (FileNotFoundException ex) {
        Logger.getLogger(ImportExportFile.class.getName()).
            log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ImportExportFile.class.getName()).
            log(Level.SEVERE, null, ex);
    }
}

public void ImportFile(String file) {

    try {
        FileInputStream fis = new FileInputStream(file);

        DataInputStream dis = new DataInputStream(fis);

        String temp = dis.readLine();

        while (temp != null) {
            data.add(temp);

            temp = dis.readLine();
        }
    } catch (FileNotFoundException ex) {
        Logger.getLogger(ImportExportFile.class.getName()).
            log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ImportExportFile.class.getName()).
            log(Level.SEVERE, null, ex);
    }
}

public void setlsi(ArrayList<String> templsi) {

    this.data = templsi;
}
```

```
}

public ArrayList<String> getlsi() {
    return this.data;
}

}
```



```
        tempData.add(hasilCleaning);

    }

    return tempData;
}

public ArrayList<String> getDatakalimat() {
    return Datakalimat;
}

public String getFilePath() {
    return this.filePath;
}

}
```

```
BuatSatuCFG.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Proses;

import java.util.ArrayList;
import java.util.Stack;
import Proses.Kelas.Aturan;
import java.util.Arrays;

/**
 *
 * @author Mukhtar
 */
public class BuatSatuCFG {

    private ArrayList<Aturan> aturanCFG = new ArrayList<>();
    private boolean stackHabis = true;

    private ArrayList<String> kumpulanKode = new
ArrayList<>(Arrays.asList("S","VP","NP","PP","AdjP","AdvP","CP","SBAR","RPN","SQ","NNO","NNP",
"PO","CCN","CSN","SYM","PRR","PRI",
"PRK","JJ","VBI","VBT","VBL","VBE","ADV","NEG","PPN","INT","PHA","DET","PAR","NUM","$$","U
NS","PRN","ADJ","VBP","ADK","ART","LBR","RBR","KUA")) ;

    public ArrayList<String> baca(String satuKalimat) {
        ArrayList<String> temp = new ArrayList<>();
        String zzz = "";
        for(char x : satuKalimat.toCharArray()) {
```

```

        if (x=='(' || x==')' || x==' '){

            if (!zzz.isEmpty() || zzz != "") {

                temp.add(zzz);

                zzz = "";

            }

            temp.add(""+x);

        }

        if (x != '(' && x != ')' && x != ' ') {

            zzz+=""+x;

        }

    }

    return temp;
}

public void doCFG(ArrayList<String> data) {

    Stack<String> stackKata = new Stack<>();

    Stack<Aturan> stackAturan = new Stack<>();

    Aturan temp = new Aturan();

    stackKata.push("BATAS");

    stackAturan.push(new Aturan("BATAS"));



    for (String x : data) {

        if (!kumpulanKode.contains(x) && !x.equals(" ") && !x.equals("(") && !x.equals(")")) {

            stackHabis = false;

            System.out.println("yang salah :-"+x+".");

            break;

        } else {

            if (x.equals(")")) {

                if (stackKata.peek().equals(" ")) {

                    Aturan stackAturanToArrayAturan = stackAturan.pop();

                    aturanCFG.add(stackAturanToArrayAturan);

                }

            }

        }

    }

}

```

```
Aturan zzz = stackAturan.pop();
zzz.addSintaks(stackAturanToArrayAturan);
stackAturan.push(zzz);

stackKata.pop();
stackKata.pop();
stackKata.pop();

} else {
    String hasilPop = stackKata.pop();
    Aturan zzz = stackAturan.pop();

    zzz.addSintaks(new Aturan(hasilPop));
    stackAturan.push(zzz);
    stackKata.pop();
}

} else {
    if (x.equals(" ")) {
        temp.setNamaTag(stackKata.peek());
        stackAturan.push(temp);
        temp = new Aturan();
    }
    stackKata.push(x);
}

if (stackKata.peek().equals("BATAS")) {
    stackKata.pop();
    break;
}

}

}

}
```

```
if (!stackKata.isEmpty()) {  
    stackHabis = false;  
}  
  
}  
  
  
public ArrayList<Aturan> getAturan() {  
    return this.aturanCFG;  
}  
  
  
public boolean getStackHabis() {  
    return stackHabis;  
}  
}
```

```
CFG.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Proses;

//import java.util.ArrayList;
import java.util.*;
import Proses.Kelas.Atur;
import Proses.Kelas.SatuFile;
import Proses.Kelas.SatuKalimat;

/**
 *
 * @author Mukhtar
 */
public class CFG {

    private ArrayList<String> SeluruhKalimat = new ArrayList<>();
    private SatuFile satuFile = new SatuFile();
    private ArrayList<Aturan> HasilNormalisasi = new ArrayList<>();
    private boolean stackHabis = true;

    public void ProsesPembuatanCFG(EkstraksiData ed) {
        ArrayList<String> data = ed.Proses();
        SeluruhKalimat = data;
        satuFile.setNamaFile(ed.getFilePath().replaceAll("\\\\|(.tre)", ""));
        System.out.println(ed.getFilePath());
        for(String s : data) {
```

```

BuatSatuCFG bsCFG = new BuatSatuCFG();

ArrayList<String> temp = bsCFG.baca(s);

String a = "";

for(String s1 : temp) {

    a+=s1;

}

bsCFG.doCFG(temp);

if (bsCFG.getStackHabis() == false) {

    stackHabis = false;

}

SatuKalimat satukalimat = new SatuKalimat(a, bsCFG.getAturan());

satuFile.addKalimat(satukalimat);

}

}

```

```

public void ProsesNormalisasiCFG() {

    HasilNormalisasi.add(satuFile.getListKalimat().get(0).getListCFG().get(0));

    for (SatuKalimat x : satuFile.getListKalimat()) {

        for (Aturan satuProductionRule : x.getListCFG()) {

            if (!cekAturan(satuProductionRule)) {

                HasilNormalisasi.add(satuProductionRule);

            }

        }

    }

    sortDataNormalisasi();

}

```

```

public void printAll () {

    System.out.println("Jumlah Kalimat : " + SeluruhKalimat.size());

    System.out.println("Jumlah Kalimat : " + satuFile.getListKalimat().size());

    for (int i = 0; i < satuFile.getListKalimat().size(); i++) {

```

```

System.out.println("===== Kalimat "+i+"=====");
System.out.println(satuFile.getListKalimat().get(i).getKalimat());
System.out.println("Ukuran Aturan : " + satuFile.getListKalimat().get(i).getListCFG().size());
for (Aturan x : satuFile.getListKalimat().get(i).getListCFG()) {
    String temp = "";
    for (Aturan y : x.getSintaks()) {
        temp += y.getNamaTag() + " ";
    }
    System.out.println("\t " + x.getNamaTag() + " -> " + temp);
}
System.out.println("=====-----");
}

System.out.println("-----");
System.out.println("----- Hasil Normalisasi -----");
for (Aturan x : HasilNormalisasi) {
    String temp = x.getNamaTag() + " -> ";
    for (Aturan y : x.getSintaks()) {
        temp += y.getNamaTag() + " ";
    }
    System.out.println(temp);
}
}

private boolean cekAturan(Aturan value) {

String aturan = value.getNamaTag() + " -> ";
for (Aturan x : value.getSintaks()) {
    aturan += x.getNamaTag() + " ";
}

```

```

for (Aturan x : HasilNormalisasi) {
    String temp = x.getNamaTag() + " -> ";
    for (Aturan y : x.getSintaks()) {
        temp += y.getNamaTag() + " ";
    }
}

if (temp.equals(aturan)) {
    return true;
}
}

return false;
}

public void sortDataNormalisasi() {
    Comparator<Aturan> ALPHABETICAL_ORDER = new Comparator<Aturan>() {
        public int compare(Aturan str1, Aturan str2) {
            int res = 0;
            if (res == 0) {
                String aturan1 = str1.getNamaTag() + " -> ";
                for (Aturan x : str1.getSintaks()) {
                    aturan1 += x.getNamaTag() + " ";
                }
                String aturan2 = str2.getNamaTag() + " -> ";
                for (Aturan x : str2.getSintaks()) {
                    aturan2 += x.getNamaTag() + " ";
                }
                res = aturan1.compareTo(aturan2);
            }
            return res;
        }
    }
}

```

```

};

Collections.sort(HasilNormalisasi, ALPHABETICAL_ORDER);

}

public void RemoverNormalisasi() {
    ArrayList<String> tempHasil = new ArrayList<>();
    for (Aturan x : HasilNormalisasi) {
        String temp = x.getNamaTag() + " -> ";
        for (Aturan isiRule : x.getSintaks()) {
            if (!isiRule.getNamaTag().equalsIgnoreCase("SYM") &&
                !(isiRule.getNamaTag().equalsIgnoreCase("CCN") && x.getSintaks().indexOf(isiRule) != 0)) {
                temp += isiRule.getNamaTag() + " ";
            }
        }
        if (!tempHasil.contains(temp)) {
            tempHasil.add(temp);
        }
    }
}

ArrayList<Aturan> hasil = new ArrayList<>();

for (String x : tempHasil) {
    String var = x.replaceAll(" ->.*", "");
    String[] rule = x.split(".-> | ");
    Aturan temp = new Aturan(var);
    for (String y : rule) {
        if (!y.equals("")) {
            Aturan y1 = new Aturan(y);
            temp.addSintaks(y1);
        }
    }
}

```

```

        }

        hasil.add(temp);

    }

HasilNormalisasi = new ArrayList<>();

HasilNormalisasi = hasil;

}

public ArrayList<Aturan> RemoverDataUji(SatuKalimat perKalimat) {

    ArrayList<Aturan> hasil = new ArrayList<>();

    for (Aturan temp : perKalimat.getListCFG()) {

        Iterator<Aturan> xx = temp.getSintaks().iterator();

        int counter = 0;

        while(xx.hasNext()) {

            Aturan xxx = xx.next();

            if (xxx.getNamaTag().equalsIgnoreCase("SYM") ||

                (xxx.getNamaTag().equalsIgnoreCase("CCN") && counter != 0)) {

                xx.remove();

            }

            counter++;

        }

        hasil.add(temp);

    }

    return hasil;
}

public ArrayList<Aturan> getHasilNormalisasi() {

    return HasilNormalisasi;
}

```

```
public ArrayList<String> getSeluruhKalimat() {  
    return SeluruhKalimat;  
}  
  
public boolean isStackHabis() {  
    return stackHabis;  
}  
  
public SatuFile getSatuFile() {  
    return satuFile;  
}  
  
public void setSatuFile(SatuFile satuFile) {  
    this.satuFile = satuFile;  
}  
}
```

```
Praproses.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package DoEverythingHere;

import java.util.ArrayList;
import Proses.CFG;
import Proses.EkstraksiData;
import Proses.Kelas.Aturan;
import java.io.File;
import java.util.Collections;
import java.util.Comparator;

/**
 *
 * @author Mukhtar
 */
public class Praproses {

    private ArrayList<Aturan> CFGNormalisasi = new ArrayList<>();

    public void doPraproses() {

        File folder = new
File("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\training");

        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
            EkstraksiData ed = new EkstraksiData();
```

```

System.out.println("Nama File = "+listOfFiles[i].getName());

ed.Ekstrak("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\train
ing\\\\"+listOfFiles[i].getName());

CFG cfg = new CFG();
cfg.ProsesPembuatanCFG(ed);

if (cfg.isStackHabis() == false) {
    System.out.println("terdapat kesalahan dalam penyusunan sintaks data latih");
} else {
    cfg.ProsesNormalisasiCFG();
    cfg.RemoverNormalisasi();
    sort(cfg.getHasilNormalisasi());
    cfg.printAll();
}
}

}

public ArrayList<Aturan> getNormalisasi() {
    return CFGNormalisasi;
}

private void sort(ArrayList<Aturan> newData) {
    ArrayList<String> tempNorm = new ArrayList<>();
    for (Aturan x : CFGNormalisasi) {
        String temp = x.getNamaTag() + " -> ";
        for (Aturan y : x.getSintaks()) {
            temp += y.getNamaTag() + " ";
        }
        tempNorm.add(temp);
    }
}

```

```

for (Aturan x : newData) {
    String temp = x.getNamaTag() + " -> ";
    for (Aturan y : x.getSintaks()) {
        temp += y.getNamaTag() + " ";
    }
    if (!tempNorm.contains(temp)) {
        CFGNormalisasi.add(x);
        tempNorm.add(temp);
    }
}

Comparator<Aturan> ALPHABETICAL_ORDER = new Comparator<Aturan>() {
    public int compare(Aturan str1, Aturan str2) {
        int res = 0;
        if (res == 0) {
            String aturan1 = str1.getNamaTag() + " -> ";
            for (Aturan x : str1.getSintaks()) {
                aturan1 += x.getNamaTag() + " ";
            }
            String aturan2 = str2.getNamaTag() + " -> ";
            for (Aturan x : str2.getSintaks()) {
                aturan2 += x.getNamaTag() + " ";
            }
            res = aturan1.compareTo(aturan2);
        }
        return res;
    }
};

Collections.sort(CFGNormalisasi, ALPHABETICAL_ORDER);
}

```

```
public void hapusTagS() {

    ArrayList<Aturan> temp = new ArrayList<>();
    for (Aturan x : CFGNormalisasi) {
        if (!x.getNamaTag().equals("S")) {
            temp.add(x);
        }
    }

    CFGNormalisasi = temp;

}

public void printAll() {
    System.out.println("data normalisasi : ");
    for (Aturan x : CFGNormalisasi) {
        String temp = x.getNamaTag() + " -> ";
        for (Aturan y : x.getSintaks()) {
            temp += y.getNamaTag() + " ";
        }
        System.out.println(temp);
    }
}
```

```
ProsesPembuatanPohonSintaks.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package DoEverythingHere;

import java.io.File;
import java.util.ArrayList;
import Proses.CFG;
import Proses.EkstraksiData;
import Proses.Kelas.Atur;
import Proses.Kelas.SatuFile;
import Proses.Kelas.SatuKalimat;

/**
 *
 * @author Mukhtar
 */
public class ProsesPembuatanPohonSintaks {

    private ArrayList<SatuFile> data = new ArrayList<>();
    private boolean truefalse = true;

    public void doProses(String StringFolder) {
        File folder = new
File("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\dataset");
        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
            EkstraksiData ed = new EkstraksiData();

```

```

System.out.println("Nama File = "+listOfFiles[i].getName());

ed.Ekstrak("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\data
set\\\\"+listOfFiles[i].getName());

CFG cfg = new CFG();
cfg.ProsesPembuatanCFG(ed);

if (cfg.isStackHabis() == false) {
    truefalse = false;
    data.add(null);
} else {
    SatuFile temp = new SatuFile();
    temp = cfg.getSatuFile();
    ArrayList<SatuKalimat> arrayKalimat = new ArrayList<>();
    for (SatuKalimat x : temp.getListKalimat()) {
        SatuKalimat tempKalimat = new SatuKalimat();
        tempKalimat.setKalimat(x.getKalimat());
        tempKalimat.setListCFG(cfg.RemoverDataUji(x));
        arrayKalimat.add(tempKalimat);
    }
    temp.setListKalimat(arrayKalimat);
    data.add(temp);
}
//untuk memasukan nilai (tidak terlalu penting tapi harus ada)
masukanNilai(StringFolder);
}

public ArrayList<SatuFile> getData() {
    ArrayList<SatuFile> hasil = new ArrayList<>();

```

```
for (SatuFile x : data) {
    if (x == null) {
        hasil.add(null);
    } else {
        hasil.add(x);
    }
}

return hasil;
}

public boolean getTrueFalse() {
    return truefalse;
}

public void masukanNilai(String StringFolder) {
    if (!StringFolder.equals("")) {
        File folder = new File(StringFolder);
        File[] listOfFiles = folder.listFiles();

        ArrayList<SatuFile> dataNilai = new ArrayList<>();

        for (int i = 0; i < listOfFiles.length; i++) {
            EkstraksiData ed = new EkstraksiData();
            ed.Ekstrak(StringFolder+"\\"+listOfFiles[i].getName());
            ArrayList<String> satuBaris = ed.getDatakalimat();
            int indexKalimat = 0;
            ArrayList<Integer> tempNilai = new ArrayList<>();
            for (String x : satuBaris) {
                String tempNilaiAktual = x.replaceAll("(.*_)|(=.*)", "");
                tempNilai.add(Integer.parseInt(tempNilaiAktual));
            }
            dataNilai.add(new SatuFile(listOfFiles[i], tempNilai));
        }
    }
}
```

```

        if ((tempNilaiAktual.equals("") && tempNilai.size() != 0)) {
            data.get(i).getListKalimat().get(indexKalimat).setListNilai(tempNilai);
            tempNilai = new ArrayList<>();
            indexKalimat++;
        } else if (!tempNilaiAktual.equals("")){
            tempNilai.add(Integer.parseInt(tempNilaiAktual));
        }
        if (satuBaris.indexOf(x) == satuBaris.size()-1) {
            data.get(i).getListKalimat().get(indexKalimat).setListNilai(tempNilai);
            tempNilai = new ArrayList<>();
        }
    }
}

public void hapusTagS() {
    for (SatuFile satuFile : data) {
        for (SatuKalimat satuKalimat : satuFile.getListKalimat()) {
            for (int i = 0; i < satuKalimat.getListCFG().size(); i++) {
                if (satuKalimat.getListCFG().get(i).getNamaTag().equals("S")) {
                    satuKalimat.getListCFG().remove(i);
                }
            }
        }
    }
}

public void forDataValid(String stringFolder) {
    File folder = new File(stringFolder);
    File[] listOfFiles = folder.listFiles();
}

```

```

for (int i = 0; i < listOfFiles.length; i++) {

    EkstraksiData ed = new EkstraksiData();
    ed.Ekstrak(stringFolder+"\\"+listOfFiles[i].getName());
    ArrayList<String> satuBaris = ed.getDatakalimat();
    SatuFile sf = new SatuFile();
    SatuKalimat tempKalimat = new SatuKalimat();
    sf.setNamaFile(ed.getFilePath().replaceAll("\\\\|(.tre)|(.txt)", ""));
    int count = 0;
    for (String x : satuBaris) {
        count++;
        Aturan xx = new Aturan();
        String tempJudulKalimat = x.replaceAll("[A-Za-z]{1,10} ->.*","");
        String tempVariableCFG = x.replaceAll("(.*)|(=.*)","");
        String[] tempSintaksCFG = x.split("(.->)|(|(=.)|(._*))");
        String tempNilaiAktual = x.replaceAll("(.*)_|(=.*)","");
        if (!tempJudulKalimat.equals("")) {
            if (!tempKalimat.getKalimat().equals("==")) {
                sf.addKalimat(tempKalimat);
                tempKalimat = new SatuKalimat();
            }
            tempKalimat.setKalimat(tempJudulKalimat);
        } else {
            xx.setNamaTag(tempVariableCFG);
            for (String a : tempSintaksCFG) {
                if (!a.equals("")) {
                    xx.addSintaks(new Aturan(a));
                }
            }
        }
    }
}

```

```

        }

        tempKalimat.addCFG(xx, Integer.parseInt(tempNilaiAktual));

    }

    if ((count == satuBaris.size())) {

        sf.addKalimat(tempKalimat);

    }

}

data.add(sf);

}

}

public void printAll2() {

    System.out.println("print all ukuran : " + data.size());

    for (SatuFile satuFile : data) {

        System.out.println("Nama File : " + satuFile.getNamaFile());

        int count = 0;

        for (SatuKalimat satuKalimat : satuFile.getListKalimat()) {

            System.out.println("===== Kalimat "+(count+1)+"=====");

            for (int i = 0; i < satuKalimat.getListCFG().size(); i++) {

                System.out.println(satuKalimat.getListCFG().size() + " :::" +
+satuKalimat.getListNilai().size());

                String temp = satuKalimat.getListCFG().get(i).getNamaTag() + " -> ";

                for (Aturan y : satuKalimat.getListCFG().get(i).getSintaks()) {

                    temp += y.getNamaTag() + " ";

                }

            }

            count++;

        }

    }

}

```

```
EvaluasiRecallPrecision.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Evaluasi;

import DoEverythingHere.ExportToExcel;
import DoEverythingHere.SurveyXValid;
import Proses.Kelas.Aturan;
import Proses.Kelas.SatuFile;
import Proses.Kelas.SatuKalimat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 *
 * @author Mukhtar
 */
public class EvaluasiRecallPrecision {

    private ArrayList<Aturan> dataNormalisasi;
    private ArrayList<SatuFile> datasetValid;
    private ArrayList<SatuFile> datasetAktual;
    private ArrayList<SatuFile> datasetAktual2;

    private ArrayList<SatuFile> hasilValidXAktual;
    private ArrayList<SatuFile> hasilPrediksi;
    private ArrayList<SatuFile> hasilSaran;
```

```

private ArrayList<Integer> dataKebenaran = new ArrayList<>();
private int[][] dataRecallPrecision = {{0,0},{0,0}};

public void Evaluasi() {
    System.out.println("sebelum--");
    cekAkurasi(datasetAktual);

    ArrayList<SatuFile> datasetAktualTemp = new ArrayList<>();
    for (int i = 0; i < datasetAktual.size(); i++) {
        SatuFile temp1 = new SatuFile();
        for (int j = 0; j < datasetAktual.get(i).getListKalimat().size(); j++) {
            SatuKalimat temp2 = new SatuKalimat();
            for (int k = 0; k < datasetAktual.get(i).getListKalimat().get(j).getListCFG().size(); k++) {
                temp2.setKalimat(datasetAktual.get(i).getListKalimat().get(j).getKalimat());
                temp2.addCFG(datasetAktual.get(i).getListKalimat().get(j).getListCFG().get(k),
datasetAktual.get(i).getListKalimat().get(j).getListNilai().get(k));
            }
            temp1.addKalimat(temp2);
        }
        datasetAktualTemp.add(temp1);
    }
    EvaluasiValidXAktual(datasetAktualTemp);
    ArrayList<Integer> jumlahBenar = new ArrayList<>();
    ArrayList<Integer> jumlahCFG = new ArrayList<>();
    ArrayList<ArrayList<String>> daftarEdited = new ArrayList<>();
    ArrayList<ArrayList<String>> daftarValid = new ArrayList<>();
    ArrayList<ArrayList<String>> daftarSaran = new ArrayList<>();
    ArrayList<String> daftarKalimat = new ArrayList<>();
    ArrayList<SatuFile> CFGSurvey = new ArrayList<>();
    SurveyXValid ssv = new SurveyXValid();
    CFGSurvey = ssv.akurasi();
}

```

```

ExportToExcel ete = new ExportToExcel();

hasilSaran = new ArrayList<>();

for (int i = 0 ; i < datasetValid.size(); i++) {

    SatuFile tempFile = new SatuFile();

    tempFile.setNamaFile(datasetValid.get(i).getNamaFile());

    for (int j = 0; j < datasetValid.get(i).getListKalimat().size(); j++) {

        String namaKalimat = datasetValid.get(i).getListKalimat().get(j).getKalimat();

        ArrayList<String> StringEdited = new ArrayList<>();

        ArrayList<String> StringValid = new ArrayList<>();

        ArrayList<String> StringSaran = new ArrayList<>();

        SatuKalimat tempKalimat = new SatuKalimat();

        tempKalimat.setKalimat(namaKalimat);

        for (int k = 0; k < datasetValid.get(i).getListKalimat().get(j).getListCFG().size(); k++) {

            String AturanEdited =
datasetAktual.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

            for (Aturan x3 :
datasetAktual.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {

                AturanEdited += x3.getNamaTag() + " ";

            }

            String AturanValid =
datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

            for (Aturan x3 : datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks())
{

                AturanValid += x3.getNamaTag() + " ";

            }

            StringValid.add(AturanValid);

            StringEdited.add(AturanEdited);

        }

        ArrayList<Aturan> tempSaran =
Suggestion(datasetAktual.get(i).getListKalimat().get(j).getListCFG());

        tempKalimat.setListCFG(tempSaran);

    }
}

```

```

tempFile.addKalimat(tempKalimat);

int count = 0;

int totalCFG = 0;

for (int k = 0; k < tempSaran.size(); k++) {

    String ProductionRuleAktual = tempSaran.get(k).getNamaTag() + " -> ";

    for (Aturan x : tempSaran.get(k).getSintaks()) {

        ProductionRuleAktual += x.getNamaTag() + " ";

    }

}

StringSaran.add(ProductionRuleAktual);

String validRule =
datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

for (Aturan x : datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {

    validRule += x.getNamaTag() + " ";

}

if (validRule.equalsIgnoreCase(ProductionRuleAktual)) {

    count++;

}

totalCFG++;

}

jumlahBenar.add(count);

jumlahCFG.add(totalCFG);

daftarKalimat.add(namaKalimat);

daftarEdited.add(StringEdited);

daftarValid.add(StringValid);

daftarSaran.add(StringSaran);

}

hasilSaran.add(tempFile);

}

```

```

ete.setData(daftarKalimat, daftarValid, daftarEdited, daftarSaran, CFGSurvey);
ete.doExport();

for (int i = 0; i < datasetAktual.size(); i++) {
    System.out.println("=====");
    System.out.println("Nama File : " + datasetAktual.get(i).getNamaFile());

    for (int j = 0; j < datasetAktual.get(i).getListKalimat().size(); j++) {
        System.out.println("\tKalimat " + (j+1) + " : " +
datasetAktual.get(i).getListKalimat().get(j).getKalimat());

        for (int k = 0; k < datasetAktual.get(i).getListKalimat().get(j).getListCFG().size(); k++) {
            String ruleValid =
datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

            String ruleAktual =
datasetAktual2.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

            String ruleHasil = hasilSaran.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag()
+ " -> ";

            for (Aturan x : datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
                ruleValid += x.getNamaTag() + " ";
            }

            for (Aturan x :
datasetAktual2.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
                ruleAktual += x.getNamaTag() + " ";
            }

            for (Aturan x : hasilSaran.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
                ruleHasil += x.getNamaTag() + " ";
            }

            System.out.println("Aturan Valid : " + ruleValid);
            System.out.println("Aturan Awal : " + ruleAktual);
            System.out.println("Aturan Hasil : " + ruleHasil);
            System.out.println("Kebenaran : " + (ruleValid.equals(ruleAktual) + " x " +
(ruleAktual.equals(ruleHasil))) + " x kalimat " + (j+1));
        }
    }
}

```

```
        }

        System.out.println("");
    }

    System.out.println("=====");
}

cekAkurasi(datasetAktual2);

recall_precision(datasetAktual2, hasilSaran);

//baca file survey

System.out.println("untuk survey : ");

ArrayList<SatuFile> CFGSurvey1 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey2 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey3 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey4 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey5 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey6 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey7 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey8 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey9 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey10 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey11 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey12 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey13 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey14 = new ArrayList<>();

ArrayList<SatuFile> CFGSurvey15 = new ArrayList<>();

CFGSurvey1.add(CFGSurvey.get(0));

CFGSurvey2.add(CFGSurvey.get(1));

CFGSurvey3.add(CFGSurvey.get(2));
```

```
CFGSurvey4.add(CFGSurvey.get(3));  
CFGSurvey5.add(CFGSurvey.get(4));  
CFGSurvey6.add(CFGSurvey.get(5));  
CFGSurvey7.add(CFGSurvey.get(6));  
CFGSurvey8.add(CFGSurvey.get(7));  
CFGSurvey9.add(CFGSurvey.get(8));  
CFGSurvey10.add(CFGSurvey.get(9));  
CFGSurvey11.add(CFGSurvey.get(10));  
CFGSurvey12.add(CFGSurvey.get(11));  
CFGSurvey13.add(CFGSurvey.get(12));  
CFGSurvey14.add(CFGSurvey.get(13));  
CFGSurvey15.add(CFGSurvey.get(14));  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey1);  
  
System.out.println("");  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey2);  
  
System.out.println("");  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey3);  
  
System.out.println("");  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey4);  
  
System.out.println("");  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey5);  
  
System.out.println("");  
  
System.out.println("=====");  
  
recall_precision(datasetAktual2, CFGSurvey6);  
  
System.out.println("");  
  
System.out.println("=====");
```

```
recall_precision(datasetAktual2, CFGSurvey7);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey8);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey9);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey10);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey11);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey12);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey13);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey14);
System.out.println("");
System.out.println("=====");
recall_precision(datasetAktual2, CFGSurvey15);
System.out.println("");
System.out.println("=====");
System.out.println("=====");
System.out.println("Rata-Rata : ");
System.out.println("Rerata Akurasi : " + (rerataAkurasi/12));
```

```

        System.out.println("Rerata Recall : " + (rerataRecall/12));
        System.out.println("Rerata Precision : " + (rerataPrecision/12));
        System.out.println("Rerata F1 Score : " + (rerataF1/12));
        System.out.println("=====");
        cekAkurasi(CFGSurvey);
    }

    public void setData(ArrayList<Aturan> dataNormalisasi, ArrayList<SatuFile> datasetValid,
ArrayList<SatuFile> datasetAktual, ArrayList<SatuFile> datasetAktual2) {
        this.dataNormalisasi = dataNormalisasi;
        this.datasetValid = datasetValid;
        this.datasetAktual = datasetAktual;
        this.datasetAktual2 = datasetAktual2;
    }

    private void cekAkurasi(ArrayList<SatuFile> data) {
        int XjumlahBenar = 0;
        int XjumlahCFG = 0;
        for (int i = 0; i < data.size(); i++) {
            System.out.println("nama file : " + data.get(i).getNamaFile());
            for (int j = 0; j < data.get(i).getListKalimat().size(); j++) {
                for (int k = 0; k < data.get(i).getListKalimat().get(j).getListCFG().size(); k++) {
                    String CFGDataValid =
datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";
                    for (Aturan x : datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
                        CFGDataValid += x.getNamaTag() + " ";
                    }
                    String xx = data.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";
                }
            }
        }
    }
}

```

```

        for (Aturan x : data.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
            xx += x.getNamaTag() + " ";
        }

        if (CFGDataValid.equals(xx)) {
            XjumlahBenar++;
        }
        XjumlahCFG++;
    }

    XjumlahBenar = 0;
    XjumlahCFG = 0;
}

}

//dataset aktual, dataset valid
//1 = ya/true, 0 = tidak/false

private void EvaluasiValidXAktual(ArrayList<SatuFile> value) {

    hasilValidXAktual = value;

    for (int i = 0; i < datasetAktual.size(); i++) {
        ArrayList<SatuKalimat> listKalimatAktual = datasetAktual.get(i).getListKalimat();
        ArrayList<SatuKalimat> listKalimatValid = datasetValid.get(i).getListKalimat();
        for (int j = 0; j < listKalimatAktual.size(); j++) {
            ArrayList<Aturan> listAturanAktual = listKalimatAktual.get(j).getListCFG();
            ArrayList<Aturan> listAturanValid = listKalimatValid.get(j).getListCFG();
            ArrayList<Integer> dataNilai = new ArrayList<>();
            for (int k = 0; k < listAturanAktual.size(); k++) {
                String ProductionRuleAktual = listAturanAktual.get(k).getNamaTag() + " -> ";

```

```

        for (Aturan x : listAturanAktual.get(k).getSintaks()) {
            ProductionRuleAktual += x.getNamaTag() + " ";
        }

        String ProductionRuleValid = listAturanValid.get(k).getNamaTag() + " -> ";
        for (Aturan x : listAturanValid.get(k).getSintaks()) {
            ProductionRuleValid += x.getNamaTag() + " ";
        }
        if ((ProductionRuleAktual.equals(ProductionRuleValid))) {
            dataNilai.add(1);
        }else {
            dataNilai.add(0);
        }
    }
}

hasilValidXAktual.get(i).getListKalimat().get(j).setListNilai(dataNilai);
}

}

}

// System.out.println("True Negative : " + dataRecallPrecision[1][1]);
// System.out.println("True Positive : " + dataRecallPrecision[1][0]); <-
// System.out.println("False Negative : " + dataRecallPrecision[0][1]);
// System.out.println("False Positive : " + dataRecallPrecision[0][0]);

private double Recall(int[][] value) {
    double recall = 0.0;
    try {
        recall = (1.0*value[1][0] / (value[1][0]+value[0][1])) * 100.0;
    } catch (Exception E) {
        recall = 0;
    }
}

```

```

    return recall;
}

private double Precision(int[][] value) {
    double precision = 0.0;
    try {
        precision = (1.0*value[1][0] / (value[1][0]+value[0][0])) * 100.0;
    } catch (Exception E) {
        precision = 0;
    }
    return precision;
}

private double Accuracy(int[][] value) {
    double accuracy = ((1.0*value[1][0]+value[1][1]) /
(value[0][0]+value[0][1]+value[1][0]+value[1][1]))*100.0;

    return accuracy;
}

//auto saran
private ArrayList<Aturan> Suggestion(ArrayList<Aturan> dataAturan) {
    ArrayList<Aturan> hasil = new ArrayList<>();
    dataKebenaran = new ArrayList<>();
    ArrayList<String> stringDataNormalisasi = new ArrayList<>();

    for (Aturan aturan : dataNormalisasi) {
        String ruleTarget = aturan.getNamaTag() + " -> ";
        for (Aturan x : aturan.getSintaks()) {

```

```

        ruleTarget += x.getNamaTag() + " ";
    }

    stringDataNormalisasi.add(ruleTarget);

}

for (int i = 0; i < dataAturan.size(); i++) {
    try {
        Aturan x = (Aturan) dataAturan.get(i).clone();
        hasil.add((Aturan) x.clone());
        dataKebenaran.add(1);
    } catch (CloneNotSupportedException ex) {
        Logger.getLogger(EvaluasiRecallPrecision.class.getName()).log(Level.SEVERE, null, ex);
    }
}

Aturan SavedAturan = new Aturan();
for (Aturan aturanTarget : hasil) {

    String saran = "-";
    String ruleTarget = "";
    int nilaiKebenaran = 1;
    for (Aturan x : aturanTarget.getSintaks()) {
        ruleTarget += x.getNamaTag() + " ";
    }

    if (!stringDataNormalisasi.contains(aturanTarget.getNamaTag() + " -> "+ruleTarget)) {
        for (Aturan aturan : dataNormalisasi) {
            String ruleAturan = "";
            for (Aturan x : aturan.getSintaks()) {
                ruleAturan += x.getNamaTag() + " ";
            }

```

```

        if (ruleTarget.equals(ruleAturan)) {
            nilaiKebenaran = 0;
            saran = aturan.getNamaTag();
            break;
        }
    }

    if (!saran.equals("-")) {
        int index = 0;
        for (Aturan temp : hasil) {
            boolean ruleFound = false;
            for (Aturan temp2 : temp.getSintaks()) {
                if (temp2.getNamaTag().equals(aturanTarget.getNamaTag())) {
                    String ruleTemp2 = "";
                    for (Aturan x : temp2.getSintaks()) {
                        ruleTemp2 += x.getNamaTag() + " ";
                    }
                    if (ruleTarget.equals(ruleTemp2)) {
                        temp2.setNamaTag(saran);
                        ruleFound = true;
                        break;
                    }
                }
            }
            if (ruleFound) {
                dataKebenaran.set(index, nilaiKebenaran);
                break;
            }
            index++;
        }
    }
}

```

```

        aturanTarget.setNamaTag(saran);

    }

}

return hasil;
}

private double AccuracyRecomendation() {

    double accuracy = 0;

    return accuracy;
}

private double rerataAkurasi = 0.0;

private double rerataRecall = 0.0;

private double rerataPrecision = 0.0;

private double rerataF1 = 0.0;

private void recall_precision(ArrayList<SatuFile> sebelumSistem, ArrayList<SatuFile>
setelahSistem) {

    for (int i = 0; i < sebelumSistem.size(); i++) {

        for (int j = 0; j < sebelumSistem.get(i).getListKalimat().size(); j++) {

            ArrayList<Integer> temp = new ArrayList<>();

            for (int k = 0; k < sebelumSistem.get(i).getListKalimat().get(j).getListCFG().size(); k++) {

                String aturanDataUji =
sebelumSistem.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";

                for (Aturan xxx :
sebelumSistem.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {

                    aturanDataUji += xxx.getNamaTag() + " ";
                }
            }
        }
    }
}

```

```

        String aturanDataValid =
datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";
        for (Aturan xxx : datasetValid.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks())
{
    aturanDataValid += xxx.getNamaTag() + " ";
}

if (aturanDataUji.equals(aturanDataValid)) {
    temp.add(1);
} else {
    temp.add(0);
}
}

sebelumSistem.get(i).getListKalimat().get(j).setListNilai(temp);
}

}

for (int i = 0; i < setelahSistem.size(); i++) {
    for (int j = 0; j < setelahSistem.get(i).getListKalimat().size(); j++) {
        ArrayList<Integer> temp = new ArrayList<>();
        for (int k = 0; k < setelahSistem.get(i).getListKalimat().get(j).getListCFG().size(); k++) {
            String aturanDataUji =
setelahSistem.get(i).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";
            for (Aturan xxx :
setelahSistem.get(i).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {
                aturanDataUji += xxx.getNamaTag() + " ";
}
}

String aturanDataValid =
datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getNamaTag() + " -> ";
        for (Aturan xxx :
datasetValid.get(0).getListKalimat().get(j).getListCFG().get(k).getSintaks()) {

```

```

        aturanDataValid += xxx.getNamaTag() + " ";
    }

    if (aturanDataUji.equals(aturanDataValid)) {
        temp.add(1);
    } else {
        temp.add(0);
    }
}

setelahSistem.get(i).getListKalimat().get(j).setListNilai(temp);
}

}

int[][] dataRecallPrecision = {{0,0},{0,0}};
for (int i = 0; i < setelahSistem.size(); i++) {
    for (int j = 0; j < setelahSistem.get(i).getListKalimat().size(); j++) {
        for (int k = 0; k < setelahSistem.get(i).getListKalimat().get(j).getListCFG().size(); k++) {
            dataRecallPrecision[setelahSistem.get(i).getListKalimat().get(j).getListNilai().get(k)][sebelumSistem.get(0).getListKalimat().get(j).getListNilai().get(k)]++;
        }
    }
}

System.out.println("Nama File : " + setelahSistem.get(0).getNamaFile());
System.out.println("True Negative : " + dataRecallPrecision[1][1]);
System.out.println("True Positive : " + dataRecallPrecision[1][0]);
System.out.println("False Negative : " + dataRecallPrecision[0][1]);
System.out.println("False Positive : " + dataRecallPrecision[0][0]);
System.out.println("-----");
System.out.println("Accuracy : " + Accuracy(dataRecallPrecision));
System.out.println("Recall : " + Recall(dataRecallPrecision));

```

```
        System.out.println("Precision : " + Precision(dataRecallPrecision));

        System.out.println("F1 : " +
(2*((Recall(dataRecallPrecision)*Precision(dataRecallPrecision))/(Precision(dataRecallPrecision)+Recall(dataRecallPrecision)))));

if (setelahSistem.get(0).getNamaFile().contains("orang")) {

    rerataAkurasi += Accuracy(dataRecallPrecision);

    rerataRecall += Recall(dataRecallPrecision);

    rerataPrecision += Precision(dataRecallPrecision);

    rerataF1 +=

(2*((Recall(dataRecallPrecision)*Precision(dataRecallPrecision))/(Precision(dataRecallPrecision)+Recall(dataRecallPrecision))));

System.out.println("=====");

System.out.println("Rerata Akurasi : " + (rerataAkurasi));

System.out.println("Rerata Recall : " + (rerataRecall));

System.out.println("Rerata Precision : " + (rerataPrecision));

System.out.println("Rerata F1 Score : " + (rerataF1));

System.out.println("=====");

}

}

}
```

```
SurveyXValid.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package DoEverythingHere;

import Proses.CFG;
import Proses.EkstraksiData;
import Proses.Kelas.SatuFile;
import Proses.Kelas.SatuKalimat;
import java.io.File;
import java.util.ArrayList;

/**
 *
 * @author Mukhtar
 */
public class SurveyXValid {

    private ArrayList<SatuFile> data = new ArrayList<>();
    private ArrayList<SatuFile> datasetValid;
    private boolean truefalse = true;

    public ArrayList<SatuFile> akurasi() {
        File folder = new
File("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\survey\\\\");
        File[] listOfFiles = folder.listFiles();

        for (int i = 0; i < listOfFiles.length; i++) {
            EkstraksiData ed = new EkstraksiData();

```

```

ed.Ekstrak("C:\\\\Users\\\\Mukhtar\\\\Documents\\\\NetBeansProjects\\\\SkripsiTextmining\\\\dataset\\\\surv
ey\\\\"+listOfFiles[i].getName());

CFG cfg = new CFG();
cfg.ProsesPembuatanCFG(ed);

if (cfg.isStackHabis() == false) {
    truefalse = false;
    data.add(null);
} else {
    SatuFile temp = new SatuFile();
    temp = cfg.getSatuFile();
    ArrayList<SatuKalimat> arrayKalimat = new ArrayList<>();
    for (SatuKalimat x : temp.getListKalimat()) {
        SatuKalimat tempKalimat = new SatuKalimat();
        tempKalimat.setKalimat(x.getKalimat());
        tempKalimat.setListCFG(cfg.RemoverDataUji(x));
        arrayKalimat.add(tempKalimat);
    }
    temp.setListKalimat(arrayKalimat);
    data.add(temp);
}
}

for (SatuFile satuFile : data) {
    for (SatuKalimat satuKalimat : satuFile.getListKalimat()) {
        for (int i = 0; i < satuKalimat.getListCFG().size(); i++) {
            if (satuKalimat.getListCFG().get(i).getNamaTag().equals("S")) {
                satuKalimat.getListCFG().remove(i);
            }
        }
    }
}

```

```
    }  
}  
return data;  
  
}  
}
```

```
ExportToExcel.java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package DoEverythingHere;

import Proses.Kelas.Aturan;
import Proses.Kelas.SatuFile;
import Proses.Kelas.SatuKalimat;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;

/**
 *
 * @author Mukhtar
 */
public class ExportToExcel {

    private static final String FILE_NAME = "DetailHasil.xlsx";
```

```
private XSSFWorkbook workbook = null;
private XSSFSheet sheet = null;
private ArrayList<ArrayList<String>> dataTypes = new ArrayList<>();

public ExportToExcel() {
    workbook = new XSSFWorkbook();
    sheet = workbook.createSheet("Detail Hasil");
    dataTypes.add(new ArrayList<>(Arrays.asList("Nama Kalimat", "Aturan Valid", "Aturan Edited",
    "Aturan yang Disarankan")));
}

public void setData(String namaKalimat, ArrayList<String> AturanValid, ArrayList<String>
AturanEdited, ArrayList<String> AturanSaran) {

    for (int i = 0; i < AturanValid.size(); i++) {
        dataTypes.add(new ArrayList<>(Arrays.asList(namaKalimat, AturanValid.get(i),
        AturanEdited.get(i), AturanSaran.get(i)))));
    }

    int rowNum = 0;

    for (ArrayList<String> datatype : dataTypes) {
        Row row = sheet.createRow(rowNum++);
        int colNum = 0;
        for (String field : datatype) {
            Cell cell = row.createCell(colNum++);
            cell.setCellValue(field);
        }
    }
}
```

```
public void setData(ArrayList<String> namaKalimat, ArrayList<ArrayList<String>> AturanValid,
ArrayList<ArrayList<String>> AturanEdited, ArrayList<ArrayList<String>> AturanSaran,
ArrayList<SatuFile> CFGSurvey) {

    ArrayList<String> judul = dataTypes.get(0);

    for (SatuFile x : CFGSurvey) {
        judul.add(x.getNamaFile());
    }

    dataTypes.set(0, judul);

    ArrayList<ArrayList<ArrayList<String>>> tempSurvey = new ArrayList<>();

    for (SatuFile x : CFGSurvey) {
        ArrayList<ArrayList<String>> tempKalimat = new ArrayList<>();

        for (SatuKalimat xx : x.getListKalimat()) {
            ArrayList<String> tempCFG = new ArrayList<>();

            for (Aturan xxx : xx.getListCFG()) {
                String temp = xxx.getNamaTag() + " -> ";
                for (Aturan x4 : xxx.getSintaks()) {
                    temp += x4.getNamaTag() + " ";
                }
                tempCFG.add(temp);
            }
            tempKalimat.add(tempCFG);
        }
        tempSurvey.add(tempKalimat);
    }

    for (int i = 0; i < AturanValid.size(); i++) {
        ArrayList<String> isian = new ArrayList<>();
        isian.add(namaKalimat.get(i));
        for (int j = 0; j < AturanValid.get(i).size(); j++) {
//            dataTypes.add(new ArrayList<>(Arrays.asList(namaKalimat.get(i), AturanValid.get(i).get(j),
AturanEdited.get(i).get(j), AturanSaran.get(i).get(j))));
        }
    }
}
```

```
        isian.add(AturanValid.get(i).get(j));
        isian.add(AturanEdited.get(i).get(j));
        isian.add(AturanSaran.get(i).get(j));
        for (int k = 0; k < tempSurvey.size(); k++) {
            isian.add(tempSurvey.get(k).get(i).get(j));
        }

        dataTypes.add(isian);
        isian = new ArrayList<>();
        isian.add(namaKalimat.get(i));
    }

}

int rowNum = 0;

for (ArrayList<String> datatype : dataTypes) {
    Row row = sheet.createRow(rowNum++);
    int colNum = 0;
    for (String field : datatype) {
        Cell cell = row.createCell(colNum++);
        cell.setCellValue(field);
    }
}

}

public void doExport() {

    try {
        FileOutputStream outputStream = new FileOutputStream(FILE_NAME);
        workbook.write(outputStream);
    }
}
```

```
workbook.close();

} catch (FileNotFoundException e) {
    e.printStackTrace();
}

} catch (IOException e) {
    e.printStackTrace();
}

}

}
```