LAPORAN TUGAS
TEORI BAHASA DAN OTOMATA
“Parsing Sintaksis Bahasa Bali”

Disusun Oleh :

Kelompok 1

I Putu Ryan Paramaditya	**(1808561024)**
I Gede Aditya Mahardika Pratama	**(1808561028)**
I Kadek Gowinda	**(1808561033)**
Lalu Muhamad Waisul Kuroni	**(1808561037)**
Edo krishnanda Aditya	**(1808561042)**

Kelas : B

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

BADUNG

2020

KATA PENGANTAR

Puji dan syukur ke hadirat Tuhan Yang Maha Kuasa atas segala rahmat yang diberikan-Nya sehingga tugas Laporan Tugas Teori Bahasa dan Otomata yang berjudul "*Parsing Sintaksis Bahasa Bali"*ini dapat kami selesaikan. Laporan ini kami buat sebagai kewajiban untuk memenuhi tugas mata perkuliahan Tugas Teori Bahasa. Dalam kesempatan ini, kami ingin menghaturkan terimakasih yang dalam kepada semua pihak yang telah membantu menyumbangkan ide dan pikiran mereka demi terwujudnya makalah ini. Akhirnya saran dan kritik pembaca yang dimaksud untuk mewujudkan kesempurnaan laporan ini, penulis sangat hargai.

Badung, 19 Desember 2020

Tim Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Parsing merupakan bentuk representasi linier yang menghasilkan suatu pohon (Parsing Tree). Pemodelan bahasa ke dalam suatu bentuk matematis sudah di kenalkan oleh Chomsky.

Cocke, Algoritma Younger dan Kasamai (CYK) adalah suatu algoritma pemrograman dinamis dengan pendekatan bottom-up. yang merupakan salah satu pionir pengembang teknik parser yang masih sering diimplementasikan dan dikembangkan hingga saat ini. Algoritma Cocke-Younger-Kasami (CYK) memanfaatkan Context-Free Grammar (CFG) tipe 2 dari chosmsky hierarchy of grammar untuk format grammarnya dan menghindari rule string kosong,

Mempelajari serta mengimplementasikan mesin parsing akan menjadi langkah untuk menuju *text mining.* Mengerti cara kerja mesin parsing text akan menjadi dasar dari *text mining* itu sendiri. Hal ini tentunya akan sangat bermanfaat bagi programmer yang baru belajar dan ingin mengembangkan minatnya terhadap penambangan tekstual.

1.2 Tujuan

1.2.1 Mengimplementasikan CFG dalam sintaksis parsing bahasa Bali.

1.2.2 Menciptakan mesin parsing berbasis *Algoritma CKY*.

1.3 Manfaat

1.3.1 Dapat mengimplementasikan CFG dalam sintaksis parsing bahasa Bali.

1.3.2 Dapat mengimplementasikan mesin parsing *Algoritma CKY*.

BAB II

KAJIAN PUSTAKA

2.1 Context Free Grammar dan Chomsky Normal Form

Terinspirasi dari bahasa natural manusia, ilmuwan- ilmuwan ilmu komputer yang mengembangkan bahasa pemrograman, turut serta memberikan tata bahasa (pemrograman) secara formal. Tata bahasa ini diciptakan secara bebas-konteks dan disebut CFG (Context Free Grammar). Hasilnya, dengan pendekatan formal ini, kompiler suatu bahasa pemrograman dapat dibuat lebih mudah dan menghindari ambiguitas ketika parsing bahasa tersebut. Contoh desain CFG untuk parser semisal : B -> BB | (B) | e untuk mengenali bahasa dengan hanya tanda kurung {‘(’,’)’} sebagai terminal-nya. Proses parsing adalah proses pembacaan untai dalam bahasa sesuai CFG tertentu, proses ini harus mematuhi aturan produksi dalam CFG tersebut. Secara formal, CFG didefinisikan :

CFG G=(V,T,P,S)

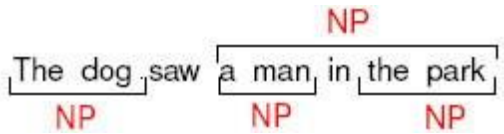
Dimana V adalah daftar variabel produksi

T, adalah daftar simbol atau terminal yang dipakai dalam CFG

P, adalah aturan produksi CFG

S, adalah variabel start aturan produksi

CFG dapat ‘dinormalkan’ dengan pola tersendiri supaya tidak ambigu dan lebih sederhana, meskipun normalisasi CFG kadang membuat aturan produksi menjadi lebih banyak dari sebelumnya. Teknik normalisasi yang digunakan dalam makalah ini adalah CNF (Chomsky Normal Form).



Gambar 1. Gambaran parsing bahasa natural (Inggris)

Contoh desain CNF dari bahasa CFG, semisal CFG berikut:

S -> aA | bB (1)

A-> Baa|ba B -> bAA|ab

CFG (1) tersebut ekivalen dengan CFG dibawah ini, dimana symbol terminal memiliki variabel produksi tersendiri:

S -> DA | EB (2)

A -> BDD | ED B -> EAA | DE D -> a

E -> b

CNF yang dihasilkan dari CFG (2) diatas ialah: S -> DA | EB (3)

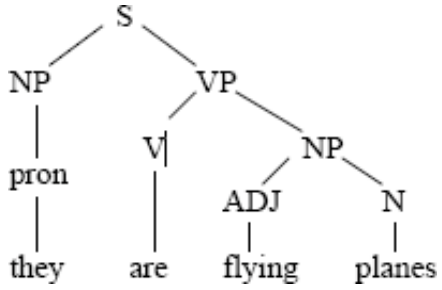
A -> BF | ED B -> EH | DE F -> DD

H -> AA D -> a

E -> b

Setelah terbentuk CFG yang telah dinormalkan secara CNF, dalam implementasi parsing, terdapat algoritma yang berguna untuk menentukan apakah suatu untai ‘valid’, atau dapat diciptakan dari aturan-aturan CFG yang ada. Salah satu algoritma yang dapat dipakai adalah Algoritma Cocke-Younger-Kasami (CYK). Algoritma ini menyelesaikan masalah analisa kembali sebuah sub-untai yang sama karena seharusnya analisa sub-untai independen terhadap parsing sub-untai yang diparsing setelahnya.

Dengan Program Dinamis, independensi yang diinginkan dapat dicapai ketika parsing. Algoritma CYK termasuk dalam bidang Program Dinamis karena algoritma ini membangun tabel status dua dimensi ketika parsing dimana penentuan parsing selanjutnya diturunkan atau dihasilkan dari parsing sebelumnya, hingga akhir untai. Selain untuk mengetahui validitas untai dalam suatu CFG, algoritma CYK yang dimodifikasi dapat dipergunakan pula untuk membangun pohon parsing.



Gambar 2. Pohon parsing yang terbentuk dari sebuah bahasa natural

2.2 Algoritma Cocke-Younger-Kasami (CYK)

Algoritma CYK menggunakan tabel dua dimensi untuk menyimpan hasil keputusan permasalahan yang lebih kecil terlebih dahulu. Sisi Program Dinamis

dari algoritma ini terletak pada pembangunan array dua dimensi atau tabel saat memparsing sebuah untai, kemudian ketika parsing untai dilakukan dalam iterasi selanjutnya, algoritma ini akan memanfaatkan array atau tabel yang telah dibangun sebelumnya. Dari tabel yang telah terbentuk, untai yang diparsing dapat diketahui apakah valid, dalam artian CFG tersebut dapat memproduksi untai tersebut melalui aturan-aturan yang ada. Berikut ini adalah persyaratan yang dibentuk dengan mengapikasikan CYK:

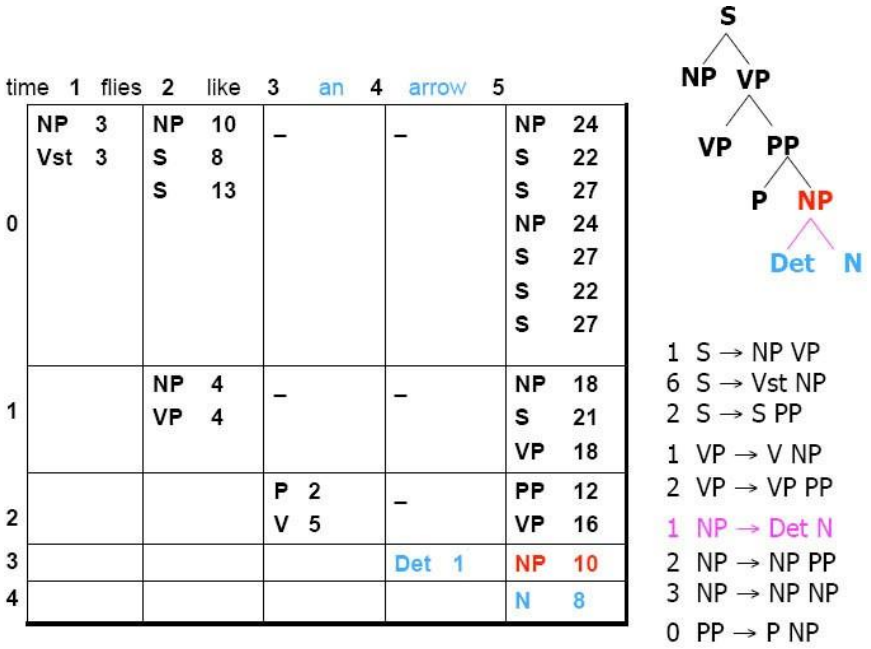
- Input: untai dengan n simbol
- Output: valid/ tak valid
- Struktur data: tabel n x n
- Baris dengan indeks 0 sampai n-1 (atau 1-n dengan modifikasi)
- Kolom dengan indeks 1 sampai n
- Sel [i,j] simbol yang yang termasuk dalam untai input

Siapkan tabel n x n dimana n adalah panjang untai yang akan dicek validitasnya, kemudian dalam diagnal tabel tersebut, isi tiap sel dengan tiap simbol dalam untai. Sel yag diisi mulai dari [i,j] awal. Setelah persyaratan pembangunan algoritma telah dipenuhi maka pseudo-code CYK seperti berikut:

```
for i := 1 to n
        Add to [i-1,i] all (part-of-speech) categories for the ith word
for width := 2 to n
for start := 0 to n-width Define end := start + width for mid := start+1 to
end-1
for every constituent X in [start,mid] for every constituent Y in [mid,end] for all
ways of combining X and Y (if any)
Add the resulting constituent to
[start,end]
```

Pseudo-code ini bermaksud untuk mengisi tiap sel setelah diagonal tengah dengan variabel produksi yang mungkin dari CFG yang diberikan oleh persoalan dengan mengecek symbol atas dan bawah. Setiap kali variabel produksi dapat dikembalikan, maka simbol dalam untai berkurang satu. Apabila tidak ada hasil produksi, maka sel diberikan penanda khusus, atau dikosongkan. Apabila untai masih berisi simbol setelah iterasi selesai sampai pada sel batas, maka untai tersebut tidak valid.

Contoh permasalahan yang dipecahkan :



Gambar 3. Algoritma CYK dalam mengecek validitas bahasa natural "time flies like an arrow"

2.3 Bahasa Pemrograman Python

Python adalah bahasa pemrograman yang ditafsirkan multiguna. Tidak seperti bahasa lain yang sulit dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah dalam memahami sintaksnya. Bagi para pemula dan yang sudah mahir dalam bahasa pemrograman lain, hal ini membuat Python sangat mudah dipelajari. Bahasa pertama kali muncul pada tahun 1991 dan dirancang oleh seorang pria bernama Guido van Rossum. Sejauh ini, Python Software Foundation masih mengembangkan Python. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distro menyertakan Python. Dengan menggunakan kode yang sederhana dan mudah, programmer dapat memprioritaskan pengembangan aplikasi yang sedang dikembangkan.

Django adalah sebuah framework full-stack untuk membuat aplikasi web dengan bahasa pemrograman python. Django juga merupakan framework python web tingkat tinggi yang dapat melakukan pengembangan aplikasi dengan cepat dan memiliki desain pragmatis yang bersih.Django dapat membuat pengembangan aplikasi menjadi lebih mudah, lebih cepat dan lebih sedikit menggunakan kode.

Kerangka kerja Django memiliki template, pustaka dan API yang dirancang untuk tumbuh dan terhubung bersama. Django cocok untuk semua proyek, dari proyek kecil hingga proyek besar. Django menggunakan Python, yang merupakan salah satu bahasa pemrograman paling populer di tahun 2015. Sekarang, merupakan bahasa pemrograman yang paling banyak dipelajari. Dibandingkan dengan framework lainnya, Django adalah yang paling kaya fitur. Django berisi semua yang dibutuhkan untuk membangun aplikasi.

BAB III

PEMBAHASAN

3.1 Sistem Mesin Parsing

Aplikasi yang kami bangun adalah aplikasi parsing sintaksis kalimat bahasa bali dengan menggunakan Algoritma CYK yang berfungsi untuk meguraikan bentuk kalimat bahasa bali berdasarkan berbagai pola kalimat (grammer) tertentu yang menyusunnya dalam bentuk parsing. Mengetahui pola kalimat (grammer) yang digunakan pada masing-masing kalimat bali yang diinpukan berbeda-beda, beserta alur dari pembentukan suatu kalimat dengan grammer tertentu dengan pola CFG yang dikonversi menjadi CNF dengan menggunakan algoritma CYK.

Fitur Utama yang terdapat pada aplikasi tersebut, yaitu: Check Validasi, apakah sintaksis pada kalimat yang diinputkan sesuai dengan tata bahasa bali. Tambah leksikon, Check Kalimat bahasa bali, Tabel CYK untuk mengetahui proses validasi pada sintaksis, dan Lihat rules untuk mengetahui pola dari sintaksis kalimat tersebut. Alur dari proses aplikasi parsing sintaksis kalimat bahasa bali tersebut yaitu: Pertama, inpukan kalimat bahasa bali pada search bar yang tersedia. Kemudian, klik “Periksa kalimat”, untuk mengecek apakah sintaksis pada kalimat bahasa bali tersebut valid atau tidak berdasalkan (pola kalimat) tata bahasa bali. Jika sintaksis pada kalimat tersebut valid, maka akan menampilkan pohon parse dari kalimat, atau struktur kalimat pada bagian bawah hasil sintaksis kalimat tersebut melalui menu “detail”. Kemudian untuk melihat akurasi pada masing-masing sintaksis kalimat bahasa bali tersebut dapat dilakukan dengan klik “Cek Akurasi”.

Aplikasi Parsing sintaksis kalimat bahasa bali dalam berbasis web programming yang dibangun dengan framework Django. Pada framework Django, didalamnya terdapat penggunaan dari framework Bootstrap yang digunakan dalam membangun front-end dari aplikasi tersebut dengan menggunkan HTML, CSS, dan Javascript. Sedangkan untuk membangun back-end pada aplikasi tersebut menggunakan bahasa pemrograman Python. Penggunaan server localhost dari

framework Django untuk menjalankan aplikasi tersebut pada web browser. Dalam proses membangun aplikasi tersebut menggunakan code editor Visual Studio Code.

3.2 Pengimplementasian Sistem

3.2.1 CYK (*table-filling algorithm*)

```
def check(sentence):
    # INPUT KEYWORD
    kalimat = sentence
    sentence = [item for item in sentence.split(' ')]
    # REMOVE EMPTY STRING IN KEYWORD
    sentence = list(filter(None, sentence))
    # ERROR IF INPUT IS EMPTY STRING
    if not sentence:
        return 'sentence tidak valid'
    else:
        # DECLARE LIST (TABLE)
        cell = [[None for i in range(len(sentence))]
                for j in range(len(sentence))]
        # RUN CYK
        return {"CYK": CYK(sentence, cell, kalimat), "cell":  cetak(cell)}
```

Pada tahap ini menggunakan 4 *function* utama untuk menjalankan algoritma CYK. *Function check( )* digunakan untuk menerima inputan kalimat yang akan diproses pada tahap selanjutnya. Pada *function* ini menggunakan parameter berupa string yang memuat kalimat yang akan diproses. Lalu kalimat tersebut akan dipecah menjadi beberapa kata yang disimpan pada suatu array (*list*). Pada *list* tersebut akan dicek apakah terdapat element yang hanya mengandung spasi saja, jika iya maka element *list* tersebut akan hapus. Selanjutnya adalah pembentukan *list* dua dimensi, dengan ordo sesuai dengan banyaknya kata yang terdapat pada kalimat tersebut. *List* dua dimensi yang nantinya digunakan untuk menyimpan hasil *terminal symbol* pada proses *table-filling.* Terakhir akan dilanjutkan dengan pemanggilan *function CYK( ).*

```
def CYK(sentence, cell, kalimat):
    global Non_valid
    for i, word in enumerate(sentence, 0):
        for j in range(i, -1, -1):
            if i is j:
                # print('sama '+str(j)+','+str(i))
                cell[j][i] = checkLexicon(word)
                if not cell[j][i]:
                    return {'validasi': 'Non-Valid', "Alasan": "tidak terdapat dalam leksikon", "kata": sentence}
            else:
                # print('beda '+str(j)+','+str(i)+' selisih : '+str(i-j))
                for k in range((i-j)):
                    # print('rumus ('+str(j)+','+str(j+k)+' , '+str(j+k+1)+','+str(i)+')')
                    if cell[j][i]:
                        cell[j][i] = list(set().union(
                            cell[j][i], checkGrammer(cell[j][j+k], cell[j+k+1][i])))
                    else:
                        cell[j][i] = checkGrammer(cell[j][j+k], cell[j+k+1][i])
```

Function CYK( ) berisi proses-proses yang digunakan pada tahapan *table-filling algorithm.* Pada proses ini terdapat 3 perulangan bersarang, dimana perulangan pertama melakukan iterasi tiap kata yang terdapat pada kalimat. Pada perulangan kedua melakukan iterasi mundur dari index pada perulangan pertama sampai index -1. Pada perulangan ini akan dilakukan pengecekan apakah index dari perulangan pertama sama dengan index dari perulangan kedua, jika iya maka akan dilakukan pemanggilan *function checkLexicon( ),* namun jika tidak maka akan dilakukan dengan perulangan ketiga. Tujuan dari perulangan ini untuk menentukan *terminal symbol* dari rule yang mengandung kata pada perulangan pertama. Perulangan ketiga melakukan iterasi sebanyak hasil pengurangan dari index perulangan pertama dengan index perulangan kedua. Perulangan ini bertujuan untuk menentukan *terminal symbol* apa saja yang dihasilkan sesuai dengan rumus yang ada pada *table-filling algorithm* dengan memanggil *function checkGrammer( ).*

```
if 'K' in cell[0][(len(sentence)-1)]:
    if kalimat in checkValid:
        Non_valid = Non_valid + 1
    return {'validasi': 'Valid', "kata": sentence}
else:
    return {'validasi': 'Non-Valid', "Alasan": "pola kalimat salah", "kata": sentence}
```

Jika perulangan diatas sudah selesai dilakukan akan dilanjutkan dengan pengecekan akhir untuk memastikan apakah kalimat yang diinputkan merupakan kalimat valid sesuai rule CNF yang telah dibuat. Dengan cara melihat apakah terdapat *symbol K* pada list dengan index [0,(jumlah kata)].

```
# FUNCTION RULE LEXICON
def checkLexicon(key):
    result = []
    for k in cnf:
        if key.lower() in [x.lower() for x in cnf[k]]:
            result.append(k)
    return result
```

Function checkLexicon( ) digunakan untuk menentukan *head of rule* dari terminal maupun kombinasi dari 2 non-terminal pada *rule* CNF yang sudah dibuat. Hasil pengecekan tersebut akan disimpan pada suatu *list*.

```
# FUNCTION RULE GRAMMER
def checkGrammer(arr1,arr2):
    result = []
    for i in arr1:
        for j in arr2:
            result = list(set().union(result,checkLexicon(i+' '+j)))
    return result
```

Function checkGrammer( ) digunakan untuk melakukan kombinasi atau *concatenation* pada beberapa *symbol non-terminal* yang nantinya hasil kombinasi tersebut akan dicek apakah terdapat pada *rule CNF* yang telah dibuat, dengan memanggil *function checkLexicon( ).*

3.2.2 AUTOCOMPLETE

Pada sistem ini memiliki fitur auto complete pada kotak search yang telah disediakan fitur ini akan mempermudah user untuk menuliskan kalimat yang diinginkan agar sesuai dengan data yang terdapat di dalam sistem yang kami bangun. Berikut potongan source code dari fitur ini yang terdapat dalam file *main.js*:

```
const input = document.querySelector("#keywords");
const perik = document.querySelector("#periksa");
const fP = document.querySelector("#formPeriksa");
fetch(url, {
  method: "GET",
})
  .then((res) => res.json())
  .then((data) => {
    kalimat = data.kalimat;
  });
let currentFocus;

perik.addEventListener("click", (e) => {
  fP.submit();
});

input.addEventListener("focus", (e) => {
  let a,
    b,
    i,
    val = input.value;
  closeAllLists();
  if (!val) {
    return false;
  }
  currentFocus = -1;

  a = document.createElement("div");
  a.setAttribute("id", a.id + "autocomplete-list");
  a.setAttribute("class", "autocomplete-items");
  input.parentNode.appendChild(a);

  for (i = 0; i < kalimat.length; i++) {
    if (kalimat[i].substr(0, val.length).toUpperCase() == val.toUpperCase()) {
      b = document.createElement("div");
      b.innerHTML = `<strong>${kalimat[i].substr(0, val.length)}</storng>`;
      b.innerHTML += kalimat[i].substr(val.length);
      b.innerHTML += `<input type='hidden' value='${kalimat[i]}'>`;
      b.addEventListener("click", (e) => {
        input.value = b.getElementsByTagName("input")[0].value;
        closeAllLists();
      });
      a.appendChild(b);
    }
  }
```

Untuk membantu dalaam proses autocomplete terdapat file pendukung yakni *kalimat.py* yang berfungsi untuk membaca setiap kalimat yang tersimpan didalam sistem sehingga akan dapat ditampilkan dalam kotak pencarian, berikut source code dari file tersebut:

```
def getKal():
    file = "parse/script/Sentence.txt"
    if (file):
        f = open(file, "r")
        data = f.read().splitlines()
        return data
```

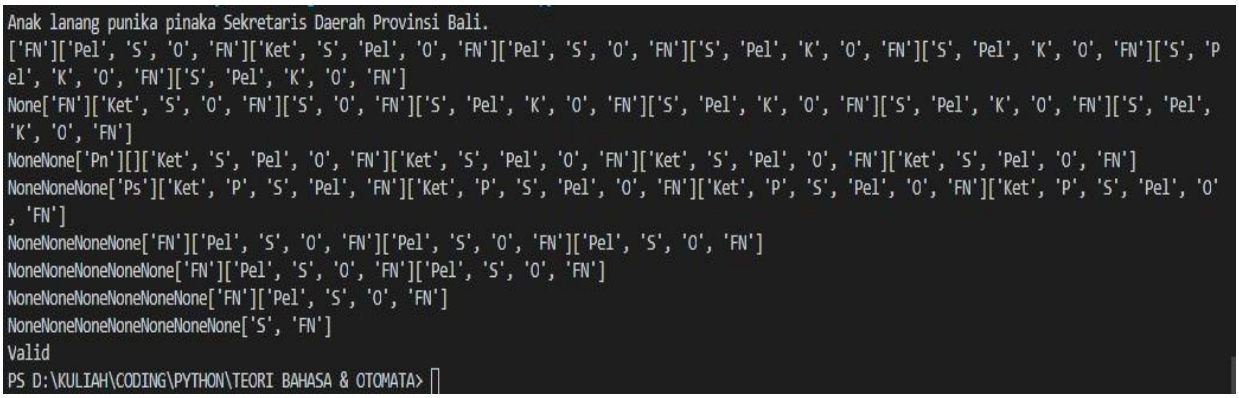
3.2.3 Tampilan

Untuk mengatur tampilan pada sistem yang ini kami mengatiurnya pada file view.py setiap fungsi didalamnya memiliki nilai kembali yang akan merender file html yang tersedia. Pada *fungtion index()* akan memanggil file index.html. selanjutnya terdapat *fungstion validasi* yang berfungsi untuk menentukan menentukan apakah kalimat yang di masukkan user valid atau tidak setelah itu akan memberikan nilai kembalian untuk memanggil file validasi.html yang berfungsi untuk menampilkan hasil validasi. Selain itu terdapat *fungstion getkalimat()* untuk menerima inputan dari user. Selain memiliki fungsi untuk mengatur kalimat pada file *view.py* ini juga terdapat fungsi yang menyimpan setiap rules pada sistem ini yakni *fungtion getRules()*:. Berikut ini potongan source code dari file *views.py :*

```
def index(request):
    Krules, Srules, Prules, Orules, Pelrules, Ketrules, FNrules, FVrules, FSrules, Gtrules, Pnrules, Psrules, Prrules, Ktrules = getRules()
    cky(Krules, Srules, Prules, Orules, Pelrules, Ketrules, FNrules,
        FVrules, FSrules, Gtrules, Pnrules, Psrules, Prrules, Ktrules)
    return render(request, 'parse/index.html')


def validasi(request):
    Krules, Srules, Prules, Orules, Pelrules, Ketrules, FNrules, FVrules, FSrules, Gtrules, Pnrules, Psrules, Prrules, Ktrules = getRules()
    if(request.GET):
        sentence = request.GET['keywords']
        context = {'data': checkSentence(Krules, Srules, Prules, Orules, Pelrules, Ketrules,
                                         FNrules, FVrules, FSrules, Gtrules, Pnrules, Psrules, Prrules, Ktrules, sentence)}
    else:
        data, akurasi = cky(Krules, Srules, Prules, Orules, Pelrules, Ketrules,
                            FNrules, FVrules, FSrules, Gtrules, Pnrules, Psrules, Prrules, Ktrules)
        context = {'data': data, "akurasi": akurasi}
    return render(request, 'parse/validasi.html', context)


def getKalimat(request):
    kalimat = getKal()
    return JsonResponse({'kalimat': kalimat}, status=200)


def getRules():
    Krules = K.objects.values_list('rules', flat=True)
    Srules = S.objects.values_list('rules', flat=True)
    Prules = P.objects.values_list('rules', flat=True)
    Orules = O.objects.values_list('rules', flat=True)
    Pelrules = Pel.objects.values_list('rules', flat=True)
    Ketrules = Ket.objects.values_list('rules', flat=True)
    FNrules = FN.objects.values_list('rules', flat=True)
    FVrules = FV.objects.values_list('rules', flat=True)
    FSrules = FS.objects.values_list('rules', flat=True)
    Gtrules = Gt.objects.values_list('rules', flat=True)
    Pnrules = Pn.objects.values_list('rules', flat=True)
    Psrules = Ps.objects.values_list('rules', flat=True)
    Prrules = Pr.objects.values_list('rules', flat=True)
    Ktrules = Kt.objects.values_list('rules', flat=True)
    # print("K :" + str(list(Krules)))
    # print("S :" + str(list(Srules)))
    # print("P :" + str(list(Prules)))
    # print("O :" + str(list(Orules)))
    # print("Pel :" + str(list(Pelrules)))
    # print("Ket :" + str(list(Ketrules)))
    # print("FN :" + str(list(FNrules)))
```

3.3 Percobaan dan Hasil

3.3.1 Evaluasi Blackbox



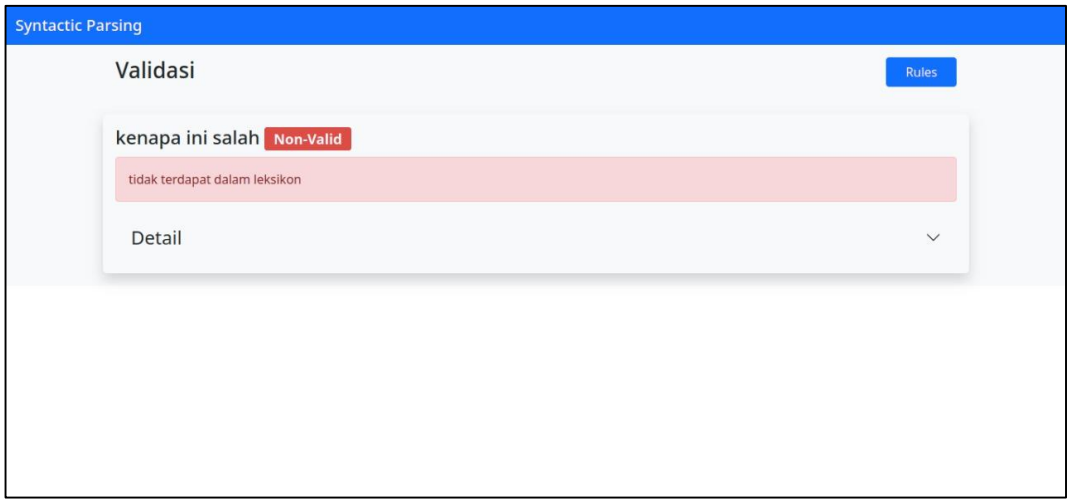
Hasil percobaan menggunakan Blackbox menunjukan hasil parsing dari kalimat "Anak lanang punika pinaka Sekretaris Daerah Provinsi Bali." Menunjukkan hasil jika kalimat tersebut valid.

Hasil blackbox juga menunjukan detail rules yang digunakan dalam proses parsing kalimat tersebut. Rules tersebut dapat menunjukan hasil yang sesuai dengan pembentukan rules leksikon dalam bentuk CNF yang dilakukan secara manual seperti berikut.

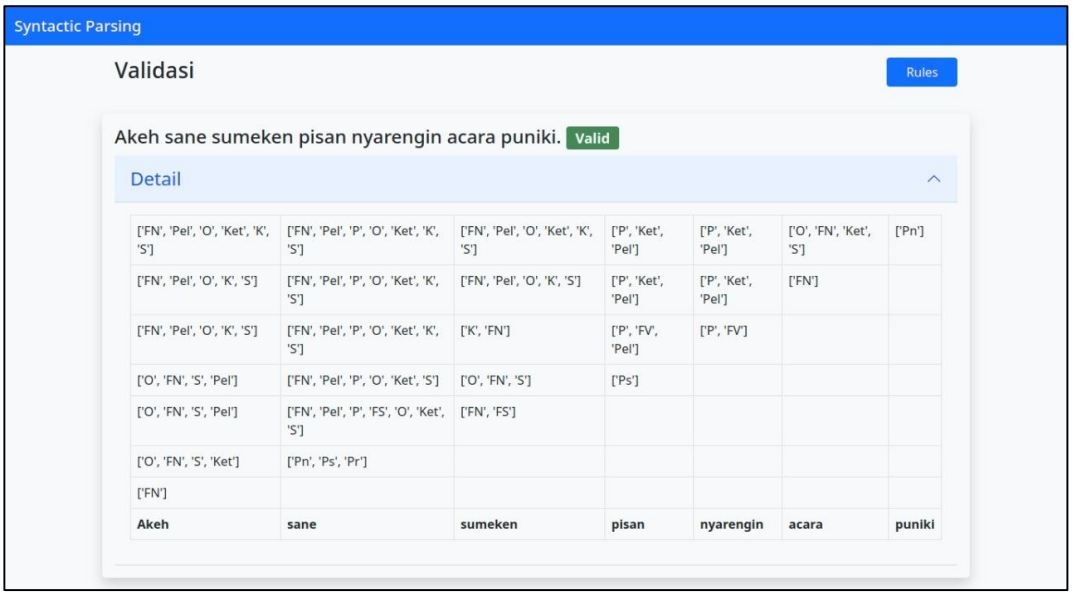
```
> S (Anak lanang punika) P (pinaka Sekretaris Daerah Provinsi Bali). Valid (S P)
K = ['S P']
S = ['FN Pn']
Pn = ['punika']
FN = ['FN FN','Anak','lanang','Sekretaris','Daerah','Provinsi','Bali']
P = ['Ps FN']
Ps = ['pinaka']
```

3.3.2 Evaluasi Dengan Masukan Langsung

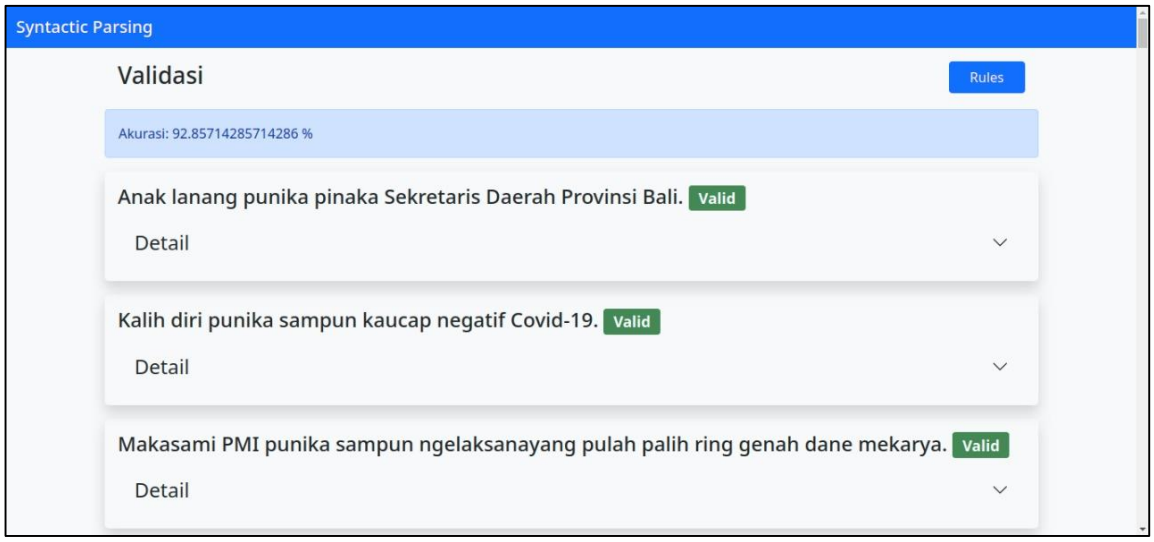
Sistem yang dibuat dapat mengetahui jika suatu kalimat masukkan termasuk valid atau tidak sesuati dengan rules yang telah ditentukan dalam sistem. Jika kalimat mengandung kata dan pola yang tidak sesuai maka sistem akan menentukan jika kalimat tidak valid.



Ini adalah contoh masukkan kalimat yang tidak valid. Kalimat “kenapa ini salah” tentunya tidak ada dalam bahasa bali dan karena itu sistem tidak dapat menemukan rules yang sesuai.



Hasil kalimat masukan yang valid dalam sistem, akan diikuti dengan rules yang digunakan untuk proses parsing terhadap kalimat tersebut. Hal ini dapat dilihat pada table detail yang menunjukkan rules yang digunakan.



Percobaan menentukan ketepatan hasil parsing kalimat dengan contoh yang diberikan menunjukan hasil seperti diatas ini. Hasil pengukuran akurasi dilakukan dengan cara jumlah kalimat yg teridentifikasi secara benar dibagi dgn jumlah seluruh kalimat dikali 100%. Sehingga sistem mendapatkan hasil akurasi sebesar 92,85714285714286%.

BAB IV

PENUTUP

4.1. Simpulan

Dengan percobaan dan implementasi mesin parsing yang dilakukan, maka dapat disimpulkan sementara bahwa mesin dapat berjalan sebagaimana mestinya sesuai dengan teori algoritma CYK. Pembentukan sintaksis parsing secara linier dapat direalisasikan ke dalam bentuk code dalam dengan platform web programming.

4.2. Saran

Dalam validasi kalimat yang diterima, ada beberapa kalimat yang tidak terdapat hasil validasinya sehingga hasil akurasi yang ditunjukkan sistem berada kurang dari angka 100%.

Diharapkan kedepannya dapat diciptakan mesin dengan leksikon dan rules yang lebih lengkap sehingga pengimplementasian mesin parsing dengan bahasa bali lebih sempurna lagi.

DAFTAR PUSTAKA

Django documentation | Django (no date). Available at: https://docs.djangoproject.com/en/3.1/ (Accessed: 16 November 2020).

Wijanarto Susanto Ajib. 2016. Generator Pohon Untuk Grammar Berbasis Algoritma Couke Younger Kasami. Seminar Nasional Aplikasi Teknologi Informasi (SNATi) 2016 ISSN: 1907 – 5022 Yogyakarta, 6 Agustus 2016, ISSN: 1907 – 5022.

Link GitHub:: https://github.com/Labenea/Parsing-Sintaksis-Bahasa-Bali-Django-