LAPORAN

TUGAS KECIL 4

IF2211 STRATEGI ALGORITMA

“Ekstraksi Informasi dari Artikel Berita dengan Algoritma Pencocokan String”

Disusun oleh:

Jovan Karuna Cahyadi 13518024

Prodi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2020

DAFTAR ISI

BAB I

Teori Singkat

1.1 Knuth-Morris-Pratt(KMP)

Algoritma KMP adalah algoritma pattern matching dari teks dengan pencarian dari kiri ke kanan. KMP melakukan preprocess untuk mencari pattern di dalam pattern itu sendiri, dengan sebutan fungsi pinggiran atau failure function yaitu ukuran tersbesar dari prefix P[0..k] yang juga merupakan suffix dari P[1..k]. Kompleksitas waktu algoritma KMP adalah O(m+n) di dapat dari fungsi pinggiran O(m) dan pencarian string O(n).

1.2 Boyer Moore

Algoritma Boyer Moore adalah algoritma pattern matching yang didasari 2 teknik yaitu looking glass dan character jump. Teknik looking glass adalah mencari P dalam T dengan berjalan mundur sesuai dengan P, dimulai dari paling belakang. Teknik character jump terjadi ketika T[i] != x dan character di P[j] tidak sama dengan T[i]. Algoritma Boyer Moore ini memiliki kompleksitas waktu O(mn + A) saat terjadi worst case yaitu ketika alphabetnya sedikit.

1.3 Regex

Regular expression (regex) adalah notasi standar yang mendeskripsikan suatu pola (pattern) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (string matching) dengan efisien.

BAB 2

Kode Program

Knuth-Morris-Pratt

```
def kmpMatch(text, pattern): #pattern matching with KMP
    text = text.lower() #lowercase
    pattern = pattern.lower() #lowercase
    text_length = len(text)
    pattern_length = len(pattern)
    fail = computeFail(pattern)
    i = 0
    j = 0
    while(i < text_length):
        if(pattern[j] == text[i]):
            if(j == pattern_length - 1):
                return i - pattern_length + 1 # match
            i += 1
            j += 1
        elif(j > 0):
            j = fail[j-1]
        else:
            i += 1
    return -1 #no match

def computeFail(pattern): #fail function for KMP
    fail = [0 for i in range (len(pattern))]
    fail[0] = 0
    pattern_length = len(pattern)
    i =1
    j =0
    while(i < pattern_length):
        if(pattern[j]== pattern[i]):
            fail[i] = j+1
            i += 1
            j += 1
        elif(j > 0):
            j = fail[j-1]
        else:
            fail[i]=0
            i += 1
    return fail
```

```
#This function used if user choose to use KMP algorithm from the web app
def KMP_Algorithm(content,pattern):
    result = []
    jumlah = []
    waktu = []
    time = findWaktu(content) #find waktu in full text
    text = get_sentences(content)
    for line in text:   #iterate every sentences
        pos = kmpMatch(line, pattern) #use kmpMatch to fine position
        if(pos != -1): #match
            num= findJumlah(line, pattern) #find jumlah from line
            if(num == -1): #nojumlah found from line
                num = "Tidak ada jumlah"
            time1 = findWaktu(line) # find waktu from line
            if(len(time1)==0): #no waktu found from line
                waktu.append(time[0]) #append the first waktu from full text, i
t supposed to be the news date
            else:
                waktu.append(time1[0])
            jumlah.append(num)
            result.append(line)
    return result,jumlah, waktu
```

Boyer Moore

```
def bmMatch(text, pattern): #Pattern matching for BM
    text = text.lower()
    pattern = pattern.lower()
    last = buildLast(pattern)
    text_length = len(text)
    pattern_length = len(pattern)
    i = pattern_length - 1
    if(i > text_length - 1):
        return -1
    j = pattern_length - 1
    while True:
        if(pattern[j] == text[i]):
            if(j == 0):
                return i  #match
            else:
                i -= 1
                j -= 1
        else:
            lo = last[ord(text[i])]
            i = i + pattern_length - min(j, 1+lo)
```

```
            j = pattern_length - 1
        if(i > text_length - 1):
            break
    return -1 #no match

def buildLast(pattern): #Initialize array for BM
    last = [-1 for i in range(128)]
    for i in range(len(pattern)):
        last[ord(pattern[i])] = i
    return last


#This function used if user choose to use BM algorithm from the web app
def BM_Algorithm(content,pattern): #this is the BM_Algorithm to give all senten
ces, jumlah, and waktu.
    result = []
    jumlah = []
    waktu = []
    time = findWaktu(content) #find waktu in full text
    text = get_sentences(content)
    for line in text: #iterate every sentences
        pos = bmMatch(line, pattern) #use bmMatch to find position
        if(pos != -1): #match
            num= findJumlah(line, pattern) #find jumlah from line
            if(num == -1): #no jumlah in line
                num = "Tidak ada jumlah"
            time1 = findWaktu(line) #find waktu from line
            if(len(time1)==0): #no waktu from line
                waktu.append(time[0]) #append the first waktu from full text, i
t supposed to be the news date
            else:
                waktu.append(time1[0])
            jumlah.append(num)
            result.append(line)
    return result,jumlah,waktu
```

Regex

```
def regexMatch(text, pattern): #Pattern matching with regex
    text = text.lower()
    pattern = pattern.lower()
    result = re.findall(str(pattern), str(text))
    return result
```

```
#This function used if user choose to use Regex algorithm from the web app
def Regex_Algorithm(content, pattern):
    result =[]
    jumlah = []
    waktu = []
    time = findWaktu(content) #find waktu in full text
    text = get_sentences(content)
    list = regexMatch(content,pattern)
    if(len(list)>0): #terdapat pattern di text
        for line in text:
            list_line = regexMatch(line, pattern)
            if(len(list_line)>0): #terdapat pattern match di line tersebut
                num= findJumlah(line, pattern)
                if(num == -1):
                    num = "Tidak Ditemukan"
                time1 = findWaktu(line)
                if(len(time1)==0):
                    waktu.append(time[0]) #append the first waktu from full tex
t, it supposed to be the news date
                else:
                    waktu.append(time1[0])
                jumlah.append(num)
                result.append(line)
    return result,jumlah, waktu
```

Find Jumlah

```
def findJumlah(text,pattern): #Find jumlah from text
    pattern_length = len(pattern)
    start_pos = kmpMatch(text,pattern)
    text = text.replace('.', '')
    list = re.findall('(?<=\s)\d+(?=\s)', text) #find all numbers with space be
tween it
    result = -1
    diff_pos = 99999999999 #initialize largest position
    for num in list:
        pos_front = abs(kmpMatch(text,num) - start_pos) #position of numbers fr
om the front of pattern
        pos_back = abs(kmpMatch(text,num) - start_pos + pattern_length) #positi
on of numbers from the back of pattern
        if(pos_back < pos_front):
            pos = pos_back
        else:
```

```
            pos = pos_front
        if(pos < diff_pos): #find the smallest difference position to pattern
            diff_pos = pos
            result = num #the number that is closer to the pattern is put into
result
    return result
```

Find Waktu

```
def findWaktu(content): #find all waktu using regex, I use 4 regex for waktu
    content = content.lower()
    list = re.findall('(?:\(\d+\/\d+\/\d+\)|\d+\/\d+\/\d+),\s*(?:pukul\s+(?:\d+
:\d+|\d+.\d+)|(?:\d+:\d+|\d+.\d+))\s*(?:wib|wita|wit|'')',content)
    temporary = re.findall('(?:senin|selasa|rabu|kamis|jumat|sabtu|minggu|'')(?
:,|'')\s\d+\s(?:jan|feb|mar|apr|mei|jun|jul|agu|sep|okt|nov|des)\s\d{4}\s\d{2}:
\d{2}\s(?:wib|wita|wit)',content)
    for temp in temporary:
        list.append(temp)
    temporary = re.findall('(?:senin|selasa|rabu|kamis|jumat|sabtu|minggu)(?:,|
'')\s+(?:\(\d+\/\d+\/\d+\)|\d+\/\d+\/\d+|\(\d+\/\d+\))\s*(?:pukul\s+(?:\d+:\d+|
\d+.\d+)|(?:\d+:\d+|\d+.\d+)|'')\s*(?:wib|wita|wit|'')',content)
    for temp in temporary:
        list.append(temp)
    temporary = re.findall('\d+\s(?:jan|feb|mar|apr|mei|jun|jul|agu|sep|okt|nov
|des|januari|februari|maret|april|mei|juni|juli|agustus|september|oktober|novem
ber|desember)\s*\d{4}',content)
    for temp in temporary:
        list.append(temp)
    return list #return list of waktu from the content
```

Utils

```
def file_read(file): #Read a file and return the content
    f= open(file, "r")
    content = f.read() #read file
    #preProcess the content before dividing into sentences
    content = content.replace('.\n', '. ')
    content = content.replace('\n\n', '. ')
    content = content.replace('\n', '. ')
    return content

def get_sentences(content):    #divide the content into sentences using nltk
```

```
    text = sent_tokenize(content) #using nltk to divide the content into lists
of sentences
    return text
```

Web App using Flask

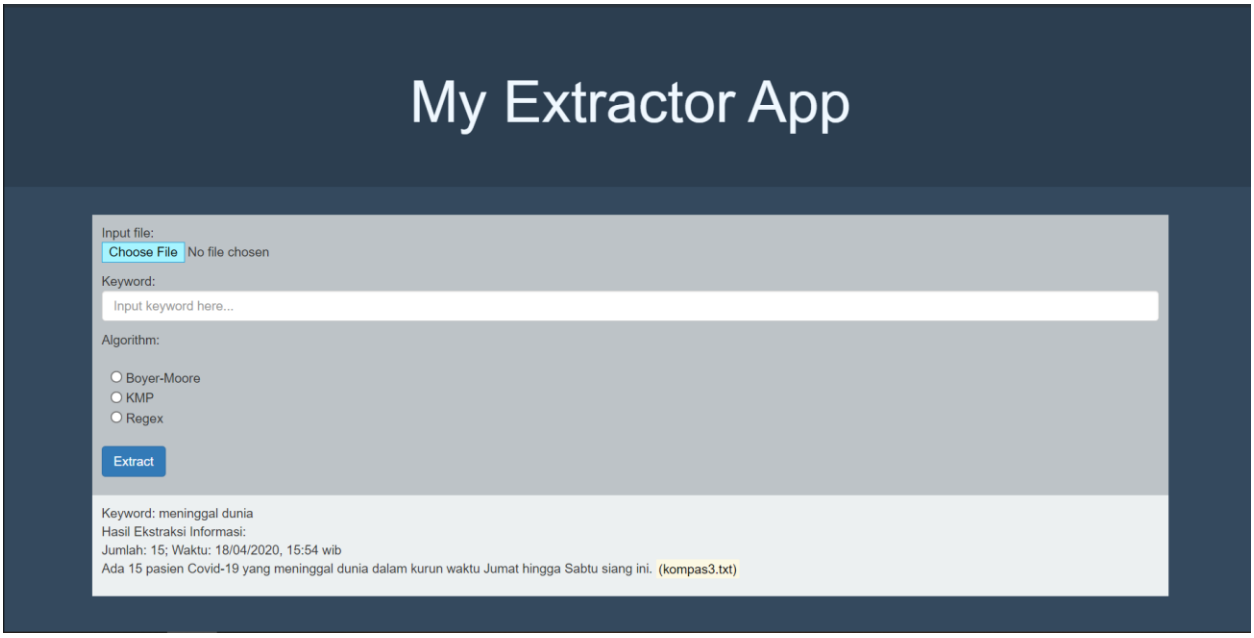
```
app = Flask(__name__)
path = "../test/"

#main page of web
@app.route('/', methods = ['POST', 'GET'])
def main():
    if request.method == 'POST':
        try:
            file = request.form['file']
            key = request.form['ky']
            option = request.form['options']
        except :
            return render_template('main.html', cek=0)
        content = file_read(path + file)
        result =[]
        jumlah =[]
        waktu = []
        if(option == 'BM'): #if user choose BM algorithm
            result, jumlah, waktu = BM_Algorithm(content,key)
        elif(option == 'KMP'): #user choose KMP algorithm
            result, jumlah, waktu = KMP_Algorithm(content,key)
        elif(option == 'Regex'): #user choose Regex algorithm
            result,jumlah, waktu = Regex_Algorithm(content,key)
        size = len(result)
        return render_template('main.html', keyword=key, size=size, result=resu
lt, jumlah=jumlah, file=file,waktu = waktu, cek =1)
    else:
        return render_template('main.html', cek=0)


if __name__ == '__main__':
   app.run(debug = True)
```

BAB II

Screenshot Input-Output Program

2.1 Screenshot Tampilan utama

2.2 Screenshot Output



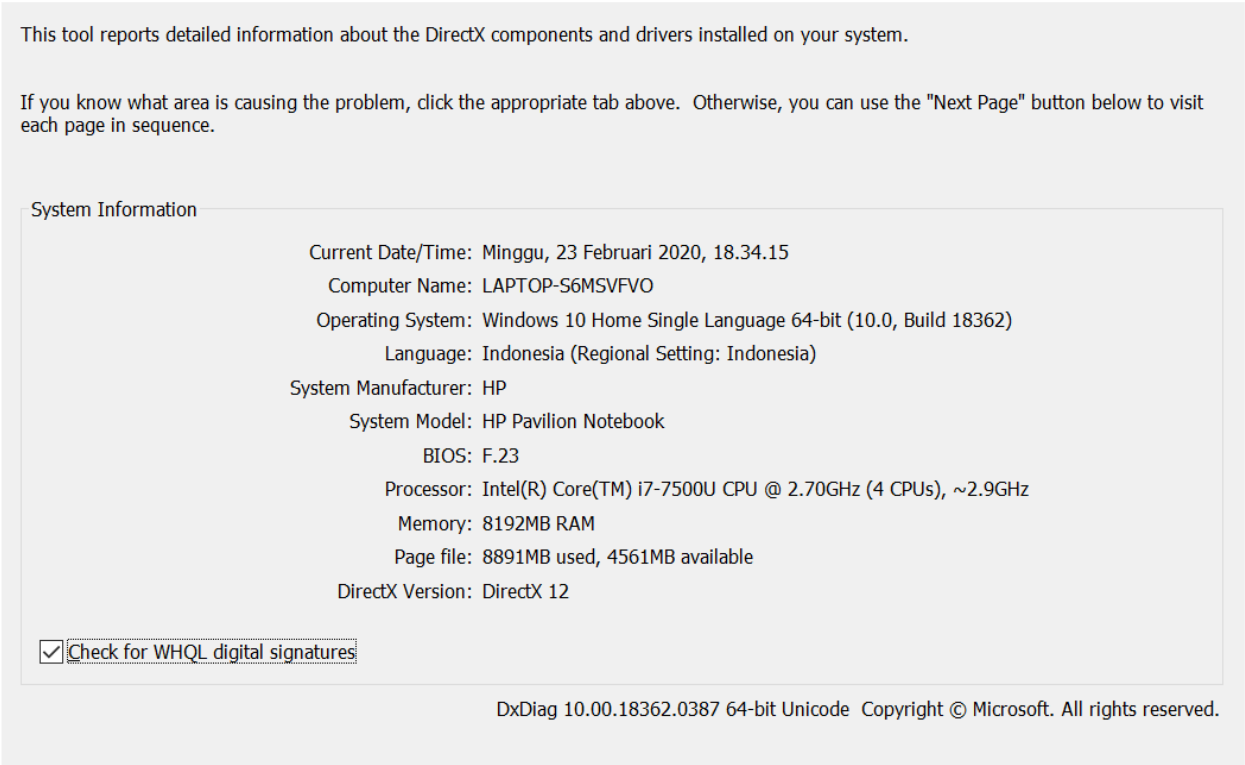
BAB IV

Tabel Nilai dan Spesifikasi Komputer

4.1 Tabel Nilai

Poin	Ya	Tidak
1. Program berhasil dikompilasi	√	
2. Program berhasil *running*	√	
3. Program dapat menerima input dan menuliskan output.	√	
4. Luaran sudah benar untuk semua semua data uji	√	

4.2 Spesifikasi Komputer



This tool reports detailed information about the DirectX components and drivers installed on your system.

If you know what area is causing the problem, click the appropriate tab above. Otherwise, you can use the "Next Page" button below to visit each page in sequence.

System Information

Current Date/Time:	Minggu, 23 Februari 2020, 18.34.15
Computer Name:	LAPTOP-S6MSVFVO
Operating System:	Windows 10 Home Single Language 64-bit (10.0, Build 18362)
Language:	Indonesia (Regional Setting: Indonesia)
System Manufacturer:	HP
System Model:	HP Pavilion Notebook
BIOS:	F.23
Processor:	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz (4 CPUs), ~2.9GHz
Memory:	8192MB RAM
Page file:	8891MB used, 4561MB available
DirectX Version:	DirectX 12

☑ Check for WHQL digital signatures

DxDiag 10.00.18362.0387 64-bit Unicode Copyright © Microsoft. All rights reserved.

DAFTAR PUSTAKA

Munir, Rinaldi. 2020. *Pattern Matching*. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf diakses pada 21 April 2020.