

**Laporan Tugas Kecil 4 - Strategi Algoritme (IF2211)**  
**“Ekstraksi Informasi dari Artikel Berita dengan Algoritme  
Pencocokan String”**



**Oleh:**

**Naufal Prima Yoriko – 13518146 – K02**

**TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**SEMESTER 2 TAHUN 2019/2020**

# **BAB I**

## **TEORI DASAR**

### **I. Algoritme Pencocokan String**

Algoritme Pencocokan String atau *Pattern Matching/Searching* adalah algoritma yang bertujuan untuk mencocokkan guna meneukan apakah suatu pola karakter/string terdapat pada text tertentu. Salah satu tujuan utama .algoritme ini adalah agar kita dapat menemukan informasi-informasi yang kita butuhkan.

### **II. Regular Expression (RegEx)**

Regular Expression adalah ekspresi baku dari suatu pola string yang biasanya penulisannya digunakan sintaks tertentu dan digunakan untuk mencari beberapa kata/ekspresi tertentu dari suatu string dasar/asli (text) yang dibutuhkan secara lebih mudah, karena penulisan dengan ekspresi baku ini sangat menyingkat penyampaian maksud dari pola yang ingin disampaikan, dan karena baku, maka dipahami secara universal di dunia komputasi.

### **III. Algoritme KMP**

Algoritme Knutt-Morris-Pratt yang digunakan untuk pencocokan string dimana menggunakan cara greedy untuk melakukan lompatan setiap ada ketidakcocokan dari string yang dicari dengan menginisialisasi Fail Array / KMP Border. Kompleksitas algoritma ini adalah  $O(m)$  untuk insialisasi fail array dan  $O(n+m)$  untuk pencocokan pattern, dengan  $m$  adalah panjang pattern dan  $n$  panjang text.

### **IV. Algoritme Boyer-Moore**

Algoritma yang menggunakan dua teknik utama yakni *looking-glass* dan *character-jump* untuk pencocokannya. Kompleksitas algoritma ini adalah  $O(m)$  untuk insialisasi fail array dan  $O(n+m)$  untuk average case dan  $O(nm)$  untuk worst case pada pencocokan pattern, dengan  $m$  adalah panjang pattern dan  $n$  panjang text.

## BAB II

### IMPLEMENTASI

#### I. Spesifikasi Komputer

Pada pengerjaan tugas kecil ini saya menggunakan sebuah laptop dengan spesifikasi sebagai berikut

Merk dan Model Laptop	HP Pavilion Laptop 14-bf1xx
BIOS	F.34
Processor	Inter Core i7-8550U CPU @1.80 GHz (8 CPUs)
Memory RAM	16384 MB
DirectX Version	DirectX 12
Display GPU	Intel UHD Graphics 620 (128 MB)
Render GPU	NVIDIA GeForce 940MX (4055 MB)
OS	Windows 10 Home

Tabel 2.1. Tabel Spesifikasi Laptop

#### II. Kode Program

Berikut ini adalah program yang bahasa python dan penjabaran akan dilakukan per file, mengingat fungsi dan struktur yang cukup banyak, sehingga tidak memungkinkan penjabaran per fungsi, disamping itu penjabaran per file masih cukup mudah dipahami.

##### 1. matcher.py

```
import re, sys

class Matcher:
    def __init__(self, text = "", pattern = "-"):
        self.text = text.lower()
        self.pattern = pattern.lower()
        self.textLength = len(self.text)
        self.patLength = len(self.pattern)
        self.resultIdx = []

    def changeText(self, text):
        self.text = text.lower()
        self.textLength = len(text)
        return self

    def changePattern(self, pattern):
        self.pattern = pattern.lower()
        self.patLength = len(pattern)
        return self
```

```

def getFirstTanggal(self):
    if(len(self.tanggal) == 0):
        return '-'
    return self.tanggal[0]

def getFirstJumlah(self):
    if(len(self.jumlah) == 0):
        return '-'
    return self.jumlah[0]

def findPattern(self, findAll = True):
    pass

def findTanggal(self, findAll = True):
    self.tanggal = []
    tanggalType = ['\d\d?[-/] \d\d[-/] \d{4}', '\d{4}[-/] \d\d[-/] \d\d?', '\d\d?[-/] \d\d?[-/] \d\d', \
        '[Kk]emarin', '\d\d[-/] \d\d[-/] \d\d?', \
        '[Hh]ari ini', '[Kk]emarin lusa', '[Ss]enin', '[Ss]elasa', '[Rr]abu', '[Kk]amis', \
        '[Jj]um\'?at', '[Ss]abtu', '[Mm]inggu']
    for pat in tanggalType:
        self.tanggal = re.findall(pat, self.text)
        if(len(self.tanggal) != 0):
            break
    if(findAll):
        return self.tanggal
    if(len(self.tanggal) == 0):
        return '-'
    return self.tanggal[0]

def findJumlah(self, findAll = True):
    self.jumlah = []
    # jumlahType = ['\d{1,3}[.],(\d{3}[.],)*\d{3} [Oo]rang', '(\d{1,3}[.],(\d{3}[.],)*\d{3}|\d+) [Kk]orban', \
    #     '(\d{1,3}[.],(\d{3}[.],)*\d{3}|\d+) [Pp]enderita', '(\d{1,3}[.],(\d{3}[.],)*\d{3}|\d+) [Jj]iwa']
    jumlahType = ['\d+ [Oo]rang', '\d+ [Kk]orban', '\d+ [Pp]enderita', '\d+ [Jj]iwa']
    for pat in jumlahType:
        self.jumlah = re.findall(pat, self.text)
        if(len(self.jumlah) != 0):
            self.jumlah = re.findall('\d+ ', self.jumlah[0])
            break
    if(findAll):
        return self.jumlah
    if(len(self.jumlah) == 0):
        return '-'
    return self.jumlah[0]

```

```

def solver(self, findAll = True):
    a = self.findPattern(findAll)
    if(self.hasPattern()):
        b = self.findJumlah(findAll)
        c = self.findTanggal(findAll)
    else:
        b, c = None, None
    return a, b, c

def showResIdx(self):
    return self.resultIdx

def hasPattern(self):
    return len(self.resultIdx) > 0

def printSolution(self):
    # for debugging
    for i in self.resultIdx:
        print(str(i) + '. ' + self.text[i:i+self.patLength])

class BoyerMooreMatcher(Matcher):
    def __init__(self, text = "", pattern = "-"):
        super().__init__(text=text, pattern=pattern)

    def initLookbackArray(self):
        # Find last occurrence of a char in the pattern, if not found then -1
        lookback = [-1] * 256
        for i in range(self.patLength):
            lookback[ord(self.pattern[i])] = i
        return lookback

    def findPattern(self, findAll = True):
        self.resultIdx = []
        if(self.patLength > self.textLength):
            return self.resultIdx
        lookback = self.initLookbackArray()
        i, j = self.patLength - 1, self.patLength - 1
        while(i < self.textLength):
            if(self.pattern[j] == self.text[i]):
                if(j == 0):
                    self.resultIdx.append(i)
                    if(not findAll):
                        break
                    i, j = i + self.patLength, self.patLength - 1
                else:
                    i, j = i - 1, j - 1
            else:
                lookback_val = lookback[ord(self.text[i])]
                i = i + self.patLength - min(j, 1 + lookback_val)
                j = self.patLength - 1
                # if(lookback_val < j and lookback_val != -1):

```

```

        #     j = lookback_val
        # else:
        #     i, j = i + self.patLength, self.patLength - 1
    return self.resultIdx

class KMPMatcher(Matcher):
    def __init__(self, text = "", pattern = "-"):
        super().__init__(text=text, pattern=pattern)

    def initKMPBorder(self):
        KMPBorder = [-1] * self.patLength
        KMPBorder[0] = 0
        i, j = 1, 0
        while(i < self.patLength):
            if(self.pattern[i] == self.pattern[j]):
                KMPBorder[i] = j + 1
            elif(j > 0):
                j = KMPBorder[j - 1]
            else:
                KMPBorder[i] = 0
                i += 1
        return KMPBorder

    def findPattern(self, findAll = True):
        self.resultIdx = []
        if(self.patLength > self.textLength):
            return self.resultIdx
        KMPBorder = self.initKMPBorder()
        i, j = 0, 0
        while(i < self.textLength):
            if(self.pattern[j] == self.text[i]):
                if(j == self.patLength - 1):
                    self.resultIdx.append(i - self.patLength + 1)
                    if(not findAll):
                        break
                    i, j = i + 1, KMPBorder[j]
                else:
                    i, j = i + 1, j + 1
            else:
                if(j > 0):
                    j = KMPBorder[j - 1]
                else:
                    i += 1
        return self.resultIdx

class RegexMatcher(Matcher):
    def __init__(self, text='', pattern='-'):
        super().__init__(text=text, pattern=pattern)

    def findPattern(self, findAll = True):
        self.resultIdx = re.findall(self.pattern, self.text)

```

```

        return self.resultIdx

if __name__ == '__main__':
    pattern = "abc"
    text = "reabcasdsabcasdaabcb"
    matcher = BoyerMooreMatcher(text = text, pattern=pattern)
    matcher.solver()
    print(matcher.resultIdx)
    matcher.printSolution()
    matcher2 = KMPMatcher(text = text, pattern=pattern)
    matcher2.solver()
    print(matcher2.resultIdx)
    matcher3 = RegexMatcher(text = text, pattern=pattern)
    matcher3.solver()
    print(matcher3.resultIdx)

```

Tabel 2.2.1 Kode matcher.py

## 2. filedata.py

```

from matcher import *
import sys, re

class FileData:
    class ResultEntry:
        def __init__(self, text, jumlah, tanggal):
            self.text = text
            self.jumlah = jumlah
            self.tanggal = tanggal

        def getText(self):
            return self.text

        def getTanggal(self):
            return self.tanggal

        def getJumlah(self):
            return self.jumlah

    def __init__(self, filename, text):
        self.filename = filename
        self.text = text
        self.parsedText = text.split(". ")

    def fetchInfo(self, pattern, method = "KMP"):
        self.result, matcher = [], None
        if(method == "KMP"):
            matcher = KMPMatcher(pattern = pattern)
        elif(method == "BM"):
            matcher = BoyerMooreMatcher(pattern = pattern)
        elif(method == "Regex"):

```

```

        matcher = RegexMatcher(pattern = pattern)

        for i in self.parsedText:
            matcher.changeText(i)
            matcher.solver()
            if(matcher.hasPattern()):
                newEntry = FileData.ResultEntry(i, matcher.getFirstJumlah(), matcher.
getFirstTanggal())
                self.result.append(newEntry)

        self.nResult = len(self.result)
        if(self.nResult != 0):
            return self

    def getResultEntry(self):
        return self.result

    def getResultNum(self):
        return self.nResult

    def getFilename(self):
        return self.filename

```

Tabel 2.2.2 Kode filedata.py

### 3. server.py

```

from filedata import *
from time import time
from flask import Flask, flash, g, redirect, render_template, request, session, url_f
or
import sys, re

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def handlerEvent():
    if request.method == 'GET':
        return render_template('mainpage.html', init = True)

    elif request.method == 'POST':
        algorithm = request.form['algorithm']
        keyword = request.form['keyword']
        filesData = request.files.getlist('files')

        files = {}
        for x in filesData:
            y = x.read(); y = y.decode("ASCII")
            y = re.sub("\r", " ", y)
            y = re.sub("\n", "", y)
            y = re.sub("\t", "", y)

```



```

        files[x.filename] = y

    result, nResult, timestamp1 = [], 0, time()
    for x, y in files.items():
        newFile = FileData(x, y)
        newFile = newFile.fetchInfo(keyword, algorithm)
        if(newFile is not None):
            result.append(newFile)
            nResult += newFile.getResultNum()

    timestamp2 = time()
    elapsedTime = (timestamp2 - timestamp1) * 1000

    return render_template('mainpage.html', keyword = keyword, init = False, result = result, nResult = nResult, elapsedTime = elapsedTime)

```

Tabel 2.2.3 Kode server.py

#### 4. mainpage.html

```

<!doctype html>
<title>{% block title %} Info Extraction App - Result{% endblock %}</title>
<link rel="stylesheet" href="{% url_for('static', filename='mainstyle.css') %}">

<main class="mainApp">
    <header>
        {% if not init %}
        <h2>Info Extraction App</h2>
        <h1>Result</h1>
        {% else %}
        <h1>Info Extraction App</h1>
        {% endif %}
    </header>

    <form method="POST" action="/" enctype="multipart/form-data">
        <p>
            <label>File(s): </label>
            <!-- <button>Browse</button> -->
            <input type="file" name="files" id="files" multiple required>

        </p>
        <p>
            <label>Keyword: </label>
            <input type="text" name="keyword" id="keyword" required>

        </p>
        <p>
            <label>Algorithm: </label>
            <p><input type="radio" name="algorithm" id="BM" value="BM" required> Boyer-Moore</p>
            <p><input type="radio" name="algorithm" id="KMP" value="KMP"> KMP</p>
            <p><input type="radio" name="algorithm" id="Regex" value="Regex"> Regex</p>

```

```

        </p>
        <p><button>Search!</button></p>
    </form>

    {% if not init %}
    <p> Keyword Pencarian: <i>{{ keyword }}</i></p>
    <main>
        {% if nResult != 0 %}
        <p> Hasil : <i>{{ nResult }}</i> | Waktu Pencarian : <i>{{ elapsedTime }} ms</i></pp>
            {% for file in result %} {% for entry in file.getResultEntry() %}
            <div>
                <p>=====</p>

                <p> Jumlah: {{ entry.getJumlah() }} | Tanggal: {{ entry.getTanggal() }} </p>
                <p> Text : {{ entry.getText() }} </p>
                <p> Nama File: <b> {{ file.getFilename() }} </b></p>
                <br>
            </div>
            {% endfor %} {% endfor %} {% else %}
            <p> Tidak ada hasil ditemukan </p>
            {% endif %}
        </main>
    {% endif %}

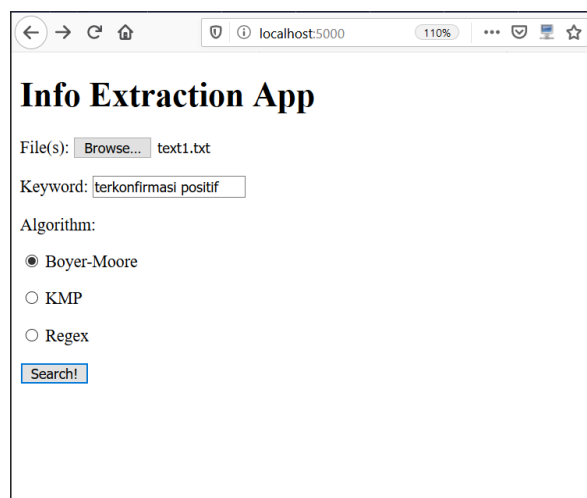
</main>

```

Tabel 2.2.4 Kode mainpage.html

### III. Uji Kasus

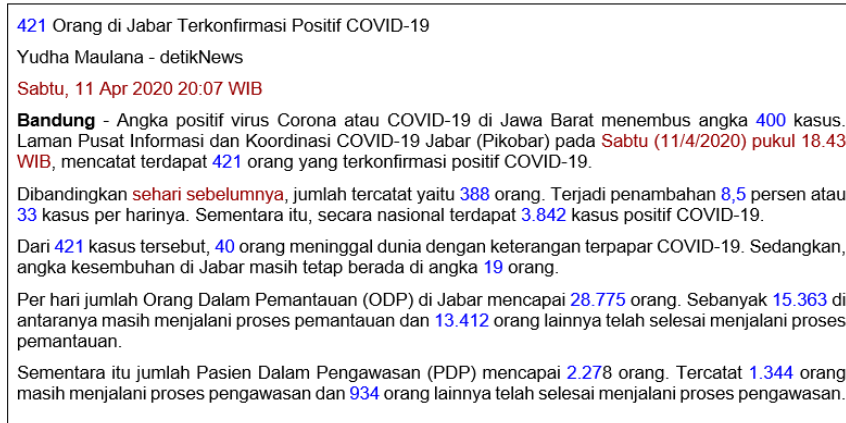
Sebelum memulai, harus membuka laman utama, dimana jika dijalankan dengan Flask (setidaknya pada apps saya) akan menggunakan `http://localhost:5000/`. Berikut ini merupakan tampilan laman utama dari web apps yang dibuat.



Gambar 2.3 Laman awal memulai  
Sumber : Dokumen Pribadi

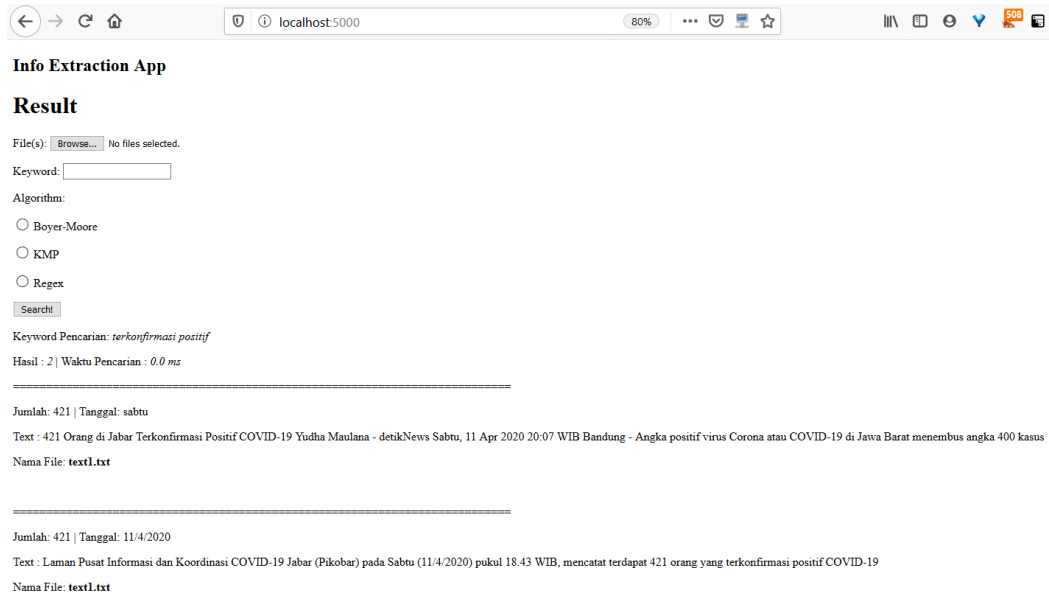
Berikut juga ditunjukkan beberapa uji coba dengan menggunakan contoh yang diberikan pada spesifikasi tucil kali ini.

## 1. Case 1



Gambar 2.3.1.1. File Text Berita  
Sumber : Spek Tugas-Kecil-4-(2020)

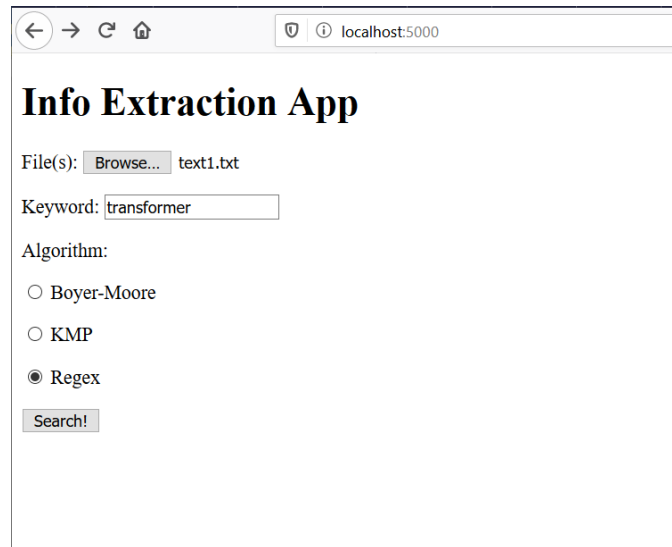
Digunakan keyword ‘terkonfirmasi positif’ dan algoritma ‘Boyer-Moore’ sehingga tampilan awal seperti pada gambar 2.3.



Gambar 2.3.1.2. Hasil Dari Pencarian Informasi ‘terkonfirmasi positif’  
Sumber : Dokumen Pribadi

## 2. Case 2

Saya akan mencoba text yang sama pada testcase 1, yakni pada gambar 2.3.1.1, namun dengan keyword yang tidak ada pada file, misal 'transformer' sehingga tampilan awal menjadi



Info Extraction App

File(s):  text1.txt

Keyword:

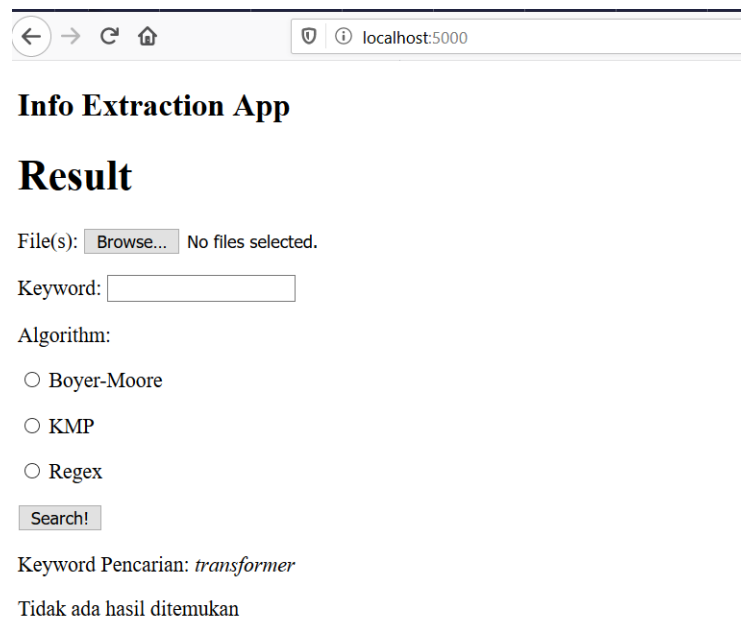
Algorithm:

☐ Boyer-Moore

☐ KMP

☒ Regex

Gambar 2.3.2.1. Tampilan Awal untuk TC 2  
*Sumber : Dokumen Pribadi*



Info Extraction App

**Result**

File(s):  No files selected.

Keyword:

Algorithm:

☐ Boyer-Moore

☐ KMP

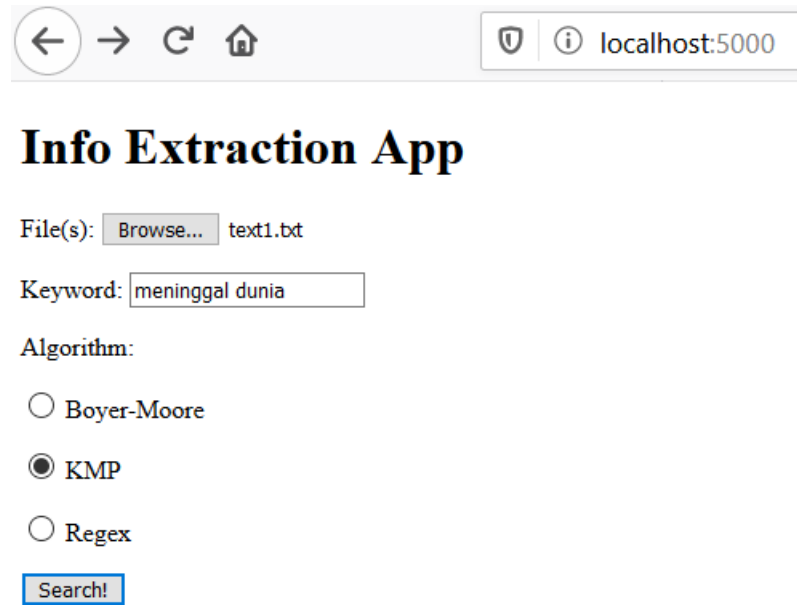
☐ Regex

Keyword Pencarian: *transformer*

Tidak ada hasil ditemukan

Gambar 2.3.2.2. Hasil Dari Pencarian Informasi 'transformer'  
*Sumber : Dokumen Pribadi*

### 3. Case 3



← → ↺ 🏠 localhost:5000

## Info Extraction App

File(s):  text1.txt

Keyword:

Algorithm:

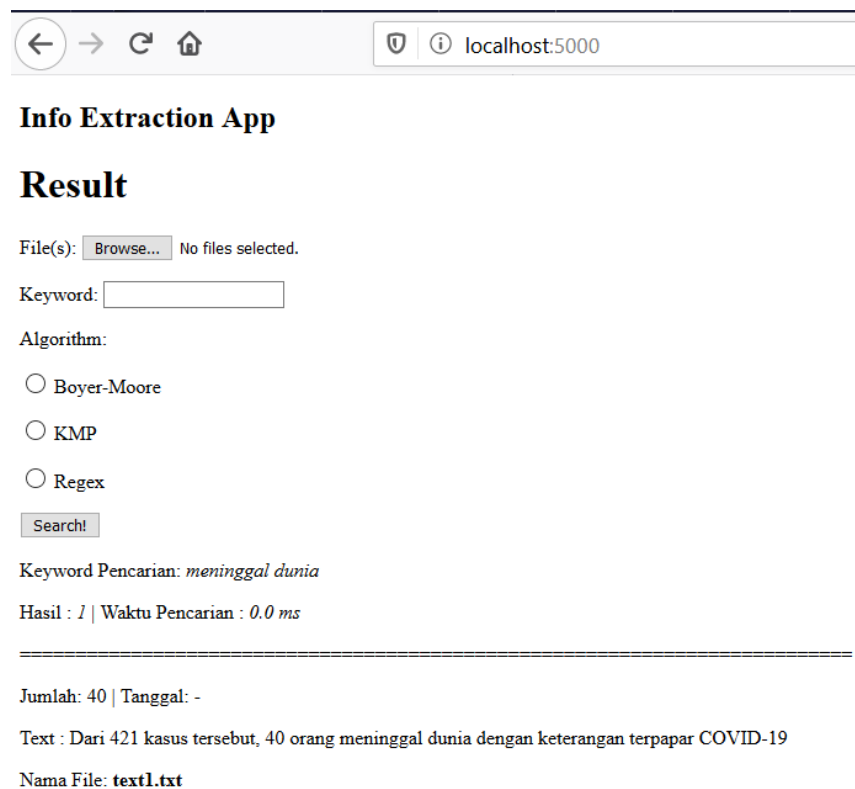
☐ Boyer-Moore

☒ KMP

☐ Regex

Gambar 2.3.3.1. Tampilan awal untuk TC 3 untuk pencarian informasi 'meninggal dunia' dengan algoritme KMP

*Sumber : Dokumen Pribadi*



← → ↺ 🏠 localhost:5000

## Info Extraction App

### Result

File(s):  No files selected.

Keyword:

Algorithm:

☐ Boyer-Moore

☒ KMP

☐ Regex

Keyword Pencarian: meninggal dunia

Hasil : 1 | Waktu Pencarian : 0.0 ms

---

Jumlah: 40 | Tanggal: -

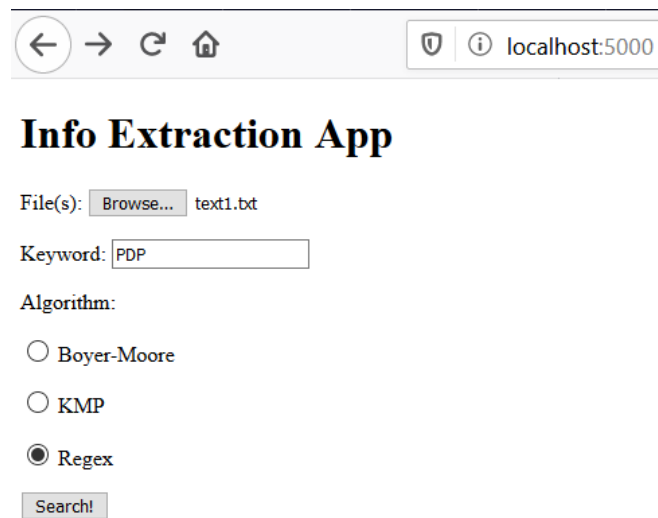
Text : Dari 421 kasus tersebut, 40 orang meninggal dunia dengan keterangan terpapar COVID-19

Nama File: text1.txt

Gambar 2.3.3.2. Hasil dari pencarian informasi 'meninggal dunia' dengan algoritme KMP

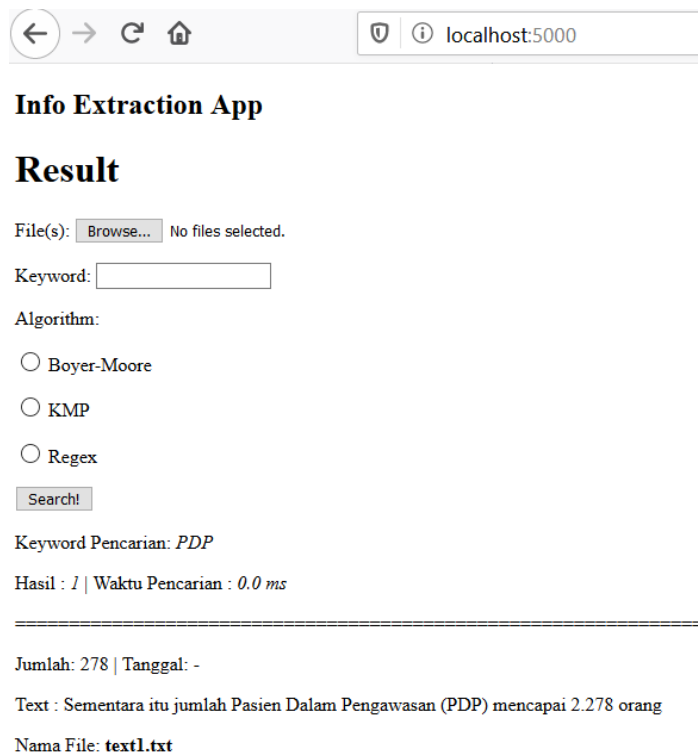
*Sumber : Dokumen Pribadi*

#### 4. Case 4



The screenshot shows a web browser window with the address bar displaying 'localhost:5000'. The page title is 'Info Extraction App'. Below the title, there is a 'File(s):' field with a 'Browse...' button and the text 'text1.bt'. A 'Keyword:' field contains the text 'PDP'. Under the 'Algorithm:' section, there are three radio buttons: 'Boyer-Moore', 'KMP', and 'Regex', with 'Regex' being selected. A 'Search!' button is located at the bottom of the form.

Gambar 2.3.4.1. Tampilan awal untuk TC 4 untuk pencarian informasi 'PDP' dengan Regex  
*Sumber : Dokumen Pribadi*



The screenshot shows the same web browser window, but the page title is now 'Result'. The 'File(s):' field shows 'No files selected.' The 'Keyword:' field is empty. The 'Algorithm:' section has three radio buttons: 'Boyer-Moore', 'KMP', and 'Regex', with 'Regex' being selected. A 'Search!' button is present. Below the form, the text 'Keyword Pencarian: PDP' is displayed. The text 'Hasil : 1 | Waktu Pencarian : 0.0 ms' is shown. A horizontal line separates the search results from the summary. The summary text includes 'Jumlah: 278 | Tanggal: -', 'Text : Sementara itu jumlah Pasien Dalam Pengawasan (PDP) mencapai 2.278 orang', and 'Nama File: text1.txt'.

Gambar 2.3.4.2. Hasil dari pencarian informasi 'PDP' dengan Regex  
*Sumber : Dokumen Pribadi*

## BAB III

### PENUTUP

#### I. Kesimpulan

Berdasarkan ringkasan hasil uji/implementasi diatas, maka dapat disimpulkan capaian dari program sebagai berikut

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi	v	
2	Program berhasil <i>running</i>	v	
3	Program dapat menerima input dan menuliskan output	v	
4	Hasil sudah benar untuk semua data uji*	parsial	

Tabel 3.1. Tabel Simpulan

**Note :** Hasil benar secara relatif, yakni untuk jumlah dan tanggal kejadian akan muncul hasil yang belum sepenuhnya **benar**, namun karena format regex **tidak persis** dengan yang dicontohkan pada spek.

#### II. Catatan Program dan Saran

Program memang berjalan sangat baik dan pada test case yang diujikan di atas, namun tetap memerlukan untuk memerhatikan beberapa hal diantaranya

1. Untuk dapat menentukan dengan 100% lengkap dan mudah dimengerti dalam ekstraksi informasi, diperlukan penggunaan regex yang banyak dan totalitas. Pada tucil ini, saya hanya menggunakan regex yang setidaknya dapat mengaproksimasi hasil berupa keluaran tanggal dan jumlah yang benar, walaupun dengan format yang masih kaku. Maka untuk pengembangannya dapat dituliskan penggunaan regex yang lebih lengkap dan menyeluruh.
2. Disini *pattern matching* yang saya gunakan kaku secara penulisan, karena berbasis algoritma dasar. Untuk kedepannya baik untuk digunakan ‘kecerdasan’ linguistik (seperti NLP), agar hasil yang ditampilkan dapat lebih luas lagi.
3. Saya hanya menggunakan struktur kaku dan tidak menggunakan style pada pembuatan interface/tampilan antarmuka. Maka, dapat dikembangkan lagi dengan memberikan rincian style (missal dengan .css, atau framework lain) untuk memperindah tampilan.

## **BAB IV**

### **REFERENSI**

Anonim. *Tugas Kecil IV IF2211 Strategi Algoritma, Ekstraksi Informasi dari Artikel Berita dengan Algoritma Pencocokan String*. Program Studi Informatika – STEI ITB. Diakses pada 18-22 April 2020.

Anonim. *Pencocokan String (String/Pattern Matching)*. Program Studi Informatika – STEI ITB. Diakses pada 18-22 April 2020.

MDN Contributors. *Sending forms through JavaScript*. Diakses pada 21-22 April 2020. [https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending\\_forms\\_through\\_JavaScript](https://developer.mozilla.org/en-US/docs/Learn/Forms/Sending_forms_through_JavaScript)

Ronacher, Armin. *Quickstart*. Diakses pada 21-22 April 2020. <https://flask-doc.readthedocs.io/en/latest/quickstart.html>