

EKSTRAKSI INFORMASI DARI ARTIKEL BERITA DENGAN ALGORITMA PENCOCOKAN STRING

LAPORAN TUGAS KECIL

Diajukan Untuk Memenuhi Tugas IF2211 Strategi Algoritma



Disusun oleh

MICHAEL HANS

13518056

**TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020**

DAFTAR ISI

DAFTAR ISI.....	1
BAB I ALGORITMA	2
1.1 Algoritma Boyers-Moore.....	2
1.2 Algoritma Knuth-Morris-Pratt	4
1.3 Regular Expression	5
BAB II PERSOALAN EKSTRAKSI INFORMASI	7
2.1 Deskripsi Persoalan.....	7
2.2 Ide Penyelesaian Masalah	7
2.3 Algoritma Pattern Matching	8
2.4 Extraction Data dengan Regular Expression	11
2.5 Algoritma Pengekstrakan Data Secara Keseluruhan	13
2.6 Web Application: Menggunakan Flask	14
BAB III PENGUJIAN PROGRAM.....	16
3.1 Spesifikasi Komputer.....	16
3.2 Pengujian Program.....	16
3.3 Checklist Program.....	21
DAFTAR PUSTAKA	22

BAB I

ALGORITMA

1.1 Algoritma Boyers-Moore

Algoritma Boyer-Moore (BM) merupakan salah satu algoritma dalam *pattern matching* yang pendekatannya berbeda dengan brute-force karena terdapat teknik-teknik tertentu dalam alur kerja algoritmanya. Berikut ini adalah mekanisme umum algoritma BM.

1) The *looking-glass technique*

Memeriksa kecocokan pattern P dengan teks T, dimulai dari indeks terakhir pada P. Pemeriksaan terhadap T tetap dimulai dari awal, dalam hal ini indeks I akan dimulai pada nilai $m-1$ bila panjang P adalah m .

2) The *character-jump technique*

Dilakukan ketika terjadi mismatch ($T[i] \neq P[j]$) dengan $T[i] = x$

Terdapat 3 kasus yang mungkin terjadi, namun pemeriksaan dilakukan secara satu per satu dengan urutan pemeriksaan sebagai berikut.

Mismatch pada $T[i]$ dan $P[j]$, dan karakter pada $T[i]$ adalah x .

a. Kasus 1:

Jika terdapat x di P dengan indeks yang lebih kecil daripada j , maka ‘seolah-olah’ geser P ke kanan agar posisi x di $T[i]$ sejajar dengan posisi **kemunculan terakhir (lo)** x di P.

Pemeriksaan berikutnya, indeks j selalu dimulai pada indeks terakhir ($m-1$) sehingga yang perlu digeser adalah posisi indeks i dengan nilai $i_{\text{new}} = i + (m-1) - lo$



Ilustrasi Kasus 1: Kemunculan terakhir x lebih kecil daripada j

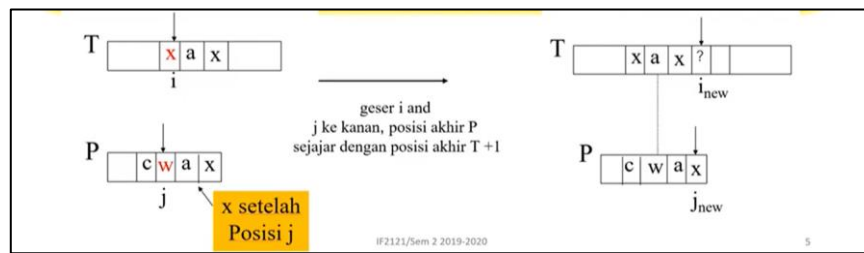
b. Kasus 2:

Mismatch pada $T[i]$ dan $P[j]$, dan karakter pada $T[i]$ adalah x .

Jika terdapat x di P, tapi pada posisi dengan indeks yang lebih besar daripada j , maka ‘seolah-olah’ geser P satu karakter ke kanan, agar posisi indeks terakhir P sejajar dengan posisi akhir T sebelumnya + 1.

Pemeriksaan berikutnya tetap dimulai dari indeks terakhir P, yaitu $j = m-1$

Yang digeser adalah nilai i , yaitu $i_{new} = i + m - j$



Ilustrasi Kasus 2: Kemunculan terakhir x lebih besar daripada j

c. Kasus 3:

Mismatch pada $T[i]$ dan $P[j]$, dan karakter pada $T[i]$ adalah x .

Jika kasus 1 dan kasus 2 tidak ditemukan saat terjadi mismatch (karakter x tidak ditemukan pada P), maka 'seolah-olah' geser P agar posisi indeks pertama P ($P[0]$) sejajar dengan indeks $i + 1$.

Pemeriksaan berikutnya tetap dimulai dari indeks terakhir P , yaitu $j = m - 1$

Yang digeser adalah nilai i , yaitu $i_{new} = i + m$



Ilustrasi Kasus 3: Karakter x tidak muncul dalam pattern P

Last Occurance Function

Untuk menangani kasus 1 dan 2, diperlukan informasi mengenai posisi kemunculan terakhir suatu karakter pada P . Oleh karena itu, akan dilakukan preproses fungsi sebelum melakukan matching P dengan T , yaitu menentukan posisi kemunculan terakhir untuk semua karakter unik pada teks T di dalam pattern P .

Jika suatu karakter pada T tidak muncul di P , maka nilai fungsi *last occurance* (lo) untuk karakter tersebut adalah -1 .

Contoh Fungsi: Last Occurance $L(x)$

Variasi karakter pada T : $A = \{a, b, c, d\}$

P	a	b	a	c	a	b
	0	1	2	3	4	5

Penjelasan:

$L(a) = 4$ karena kemunculan terakhir karakter a pada P ada di indeks 4

$L(b) = 5$ karena kemunculan terakhir karakter b pada P ada di indeks 5

$L(c) = 3$ karena kemunculan terakhir karakter c pada P ada di indeks 3

$L(d) = -1$ karena karakter d tidak muncul pada P

Semua nilai disimpan dalam tabel atau larik.

j	0	1	2	3
P[j]	a	b	a	b

1.2 Algoritma Knutt-Morris-Pratt

Algoritma Knutt-Morris-Pratt (KMP) merupakan salah satu algoritma dalam *pattern matching*. Algoritma KMP melakukan pemeriksaan dari kiri ke kanan, seperti pendekatan brute force. Namun, perbedaan dengan algoritma *brute-force* adalah pergeseran yang dilakukan lebih cerdas pada algoritma KMP.

Pergeseran algoritma KMP lebih cerdas karena algoritma ini memanfaatkan konsep prefix dan suffix. Konsep prefix dan suffix yang diterapkan adalah sebagai berikut.

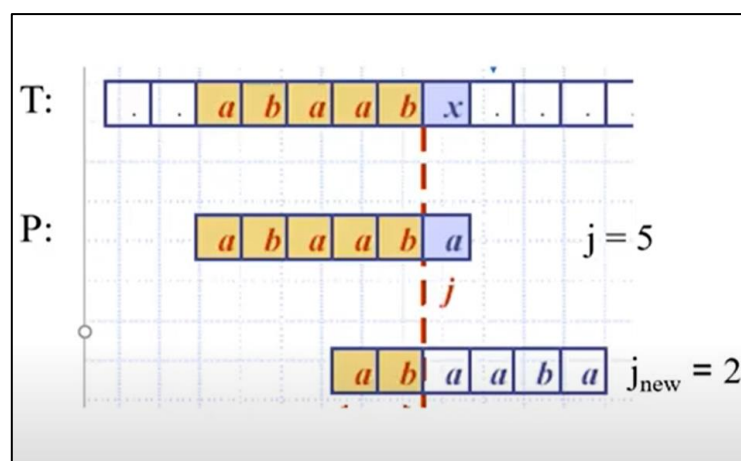
Prefix (awalan) adalah substring $S[0..k]$

Suffix (akhiran) adalah substring $S[k..m-1]$

Dengan m adalah panjang dari string pattern dan k adalah suatu index antara 0 hingga m-1

Pergeseran algoritma KMP lebih cerdas dengan mempertimbangkan sebanyak mungkin pergeseran sehingga tidak perlu mengulang pemeriksaan yang sama.

Mekanisme Umum



Ilustrasi Mekanisme Umum Algoritma KMP

Jika terjadi mismatch teks T pada $T[i]$ dan pattern P pada $P[j]$, yaitu $T[i] \neq P[j]$, pergeseran yang dilakukan adalah sebanyak:

- Nilai prefix $P[0..j-1]$ terbesar yang juga merupakan suffix $P[1..j-1]$
- Prefix $P[0..j-1]$: a, ab, aba, abaa, abaab
- Suffix $P[1..j-1]$: b, ab, aab, baab

Nilai prefix terbesar adalah 2, artinya indeks P yang diperiksa selanjutnya mulai dari indeks ke-2.

KMP Border Function

Karena nilai prefix terbesar dipengaruhi oleh posisi j yang merupakan posisi terjadi mismatch, maka diperlukan suatu preproses terhadap pattern P untuk menyimpan semua nilai prefix terbesar $P[0..j-1]$ terbesar yang merupakan suffix $P[1..j-1]$ ketika terjadi mismatch pada $P[j]$.

Preproses ini disebut sebagai KMP Border Function, yaitu sebuah fungsi pembatas untuk melakukan pergeseran berdasarkan kesamaan prefix dan suffix terbesar. KMP Border Function ini akan disimpan ke dalam sebuah senarai untuk memudahkan pengambilan nilai prefix terbesar yang sudah disimpan pada awal proses.

Berikut ini adalah contoh pemrosesan KMP Border Function.

Pattern P = ababababca

j	0	1	2	3	4	5	6	7	8	9
P[j]	a	b	a	b	a	b	a	b	c	a
k	-	0	1	2	3	4	5	6	7	8
b[k]	-	0	0	1	2	3	4	5	6	0

Penjelasan: Bila $k = 6$, maka

Prefix $P[0..6]$: a, ab, aba, abab, ababa, ababab, abababa

Suffix $P[1..6]$: a, ba, aba, baba, ababa, bababa

Perhatikan bahwa terdapat kesamaan prefix dan suffix pada a, aba, dan ababa. Karena substring ababa merupakan irisan prefix suffix terpanjang, nilai prefix terbesar adalah 5.

1.3 Regular Expression

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. Regex ini sangat bermanfaat, terutama dalam membangkitkan dan

mendefinisikan suatu bahasa yang dinyatakan dalam ekspresi tertentu. Regular expression mencocokkan string dengan suatu pola, dengan memanfaatkan state-state seperti pada mesin turing.

Pencocokan dibagi jadi 2, karakter literal dan karakter meta.

1. Karakter Literal, yaitu melakukan pencocokan seperti pada umumnya
2. Metakarakter, yaitu karakter khusus yang mencakup suatu kelompok karakter atau karakter tertentu yang tidak dapat dituliskan secara eksplisit.

Regular Expression ini dibuat sesuai dengan preferensi bahasa yang ingin diterima oleh pembuat regex. Terdapat banyak kamus data yang bisa digunakan dalam penulisan regex, seperti bila ingin mencocokkan suatu nomor yang diawali dengan angka 11, maka regex yang sesuai adalah menambahkan karakter literal berupa '11' diawal ekspresi, yang kemudian diikuti oleh bilangan integer berapapun.

Berikut ini adalah beberapa cheatsheet terkait penggunaan regex tersebut.

Character classes

`.` any character except newline
`\w \d \s` word, digit, whitespace
`\W \D \S` not word, digit, whitespace
`[abc]` any of a, b, or c
`[^abc]` not a, b, or c
`[a-g]` character between a & g

Anchors

`^abc$` start / end of the string
`\b` word boundary

Escaped characters

`\. * \` escaped special characters
`\t \n \r` tab, linefeed, carriage return
`\u00A9` unicode escaped ©

Groups & Lookaround

`(abc)` capture group
`\1` backreference to group #1
`(?:abc)` non-capturing group
`(?=abc)` positive lookahead
`(?!abc)` negative lookahead

Quantifiers & Alternation

`a* a+ a?` 0 or more, 1 or more, 0 or 1
`a{5} a{2,}` exactly five, two or more
`a{1,3}` between one & three
`a+? a{2,}? match as few as possible`
`ab|cd` match ab or cd

BAB II

PERSOALAN EKSTRAKSI INFORMASI

2.1 Deskripsi Persoalan

Algoritma *Pattern Matching* dan Regular Expression sangat bermanfaat dalam konteks kehidupan sehari-hari, terutama pada pemrosesan kata. Salah satu penerapan algoritma *Pattern Matching* dan Regular Expression adalah melakukan ekstraksi data dari suatu artikel dengan *keyword* tertentu untuk mengambil dua buah data penting, seperti jumlah dan tanggal.

Data menjadi sesuatu yang penting, terutama terkait data virus corona yang melanda saat ini sehingga dengan adanya aplikasi ekstraksi data, pengguna dapat memperoleh data-data secara mudah dari banyak sumber secara bersamaan tanpa perlu membaca secara penuh. Oleh karena itu, dibuatlah suatu ekstraktor data yang menerapkan algoritma *Pattern Matching* dan Regular Expression tersebut.

2.2 Ide Penyelesaian Masalah

Persiapan aplikasi ekstraksi data dimulai dengan menentukan parameter-parameter yang dibutuhkan untuk melakukan ekstraksi data. Berikut ini adalah parameter-parameter yang dibutuhkan sebelum proses ekstraksi data dilakukan.

1. Artikel-artikel yang ingin dibaca dalam bentuk .txt
2. Keyword yang ingin ditelusuri dalam setiap artikel
3. Algoritma pencocokkan yang diterapkan (BM, KMP, dan Regex)

Parameter-parameter tersebut akan diisi oleh pengguna yang kemudian akan diproses oleh algoritma *pattern matching* dan regular expression tersebut. Alur penyelesaian persoalan adalah sebagai berikut.

1. Untuk setiap artikel yang mau diekstrak, dilakukan pemecahan dari sebuah teks besar ke dalam kumpulan kalimat. Pemecahan menjadi kalimat-kalimat tersebut menggunakan library nltk yang telah disediakan python.
2. Dari kumpulan kalimat tersebut, akan dilakukan pemfilteran untuk memperoleh kalimat-kalimat yang mengandung *keyword* yang dimaksud. Pemfilteran ini memanfaatkan salah satu algoritma pencocokkan sesuai preferensi pengguna.
3. Untuk setiap kalimat dalam kumpulan kalimat yang sudah difilter, akan dilakukan proses ekstraksi data menggunakan regular expression yang sudah didefinisikan oleh pengembang. Oleh karena itu, pengembang perlu mendefinisikan terlebih dahulu regular expression agar bisa menerima grammar berupa tanggal dan jumlah yang diperlukan untuk ekstraksi data.

4. Ekstraksi data dilakukan dengan mencari kecocokkan antara substring dalam suatu kalimat dengan regular expression yang telah didefinisikan. Tujuan menggunakan regex karena tanggal dan jumlah mempunyai variasi penulisan yang berbeda-beda sehingga regex mampu mencocokkan dengan penulisan tersebut sesuai notasi yang didefinisikan.
5. Kembalikan hasil ekstraksi data dan kalimat yang diekstrak tersebut ke layar sehingga pengguna mampu melihat hasil ekstraksi data dari artikel.

2.3 Algoritma Pattern Matching

Berikut ini adalah tiga buah algoritma pattern matching yang diimplementasikan, yaitu algoritma Boyer-Moore, algoritma KMP, dan regular expression.

Algoritma Boyer-Moore: PatternBM.py

```
# Modul Pattern Matching Metode 1: Boyer-Moore

# Fungsi / Prosedur
# Mengembalikan posisi kemunculan terakhir character dalam pattern
def GetLastOccurance(pattern, character):
    position = -1
    n = len(pattern) - 1
    while ((n >= 0) and (position == -1)):
        if (pattern[n] == character):
            position = n
        else:
            n = n - 1
    return position

# Mengembalikan map berisi last occurance dari setiap karakter unik dari text dalam pattern
def BMLastOccurance(text, pattern):
    lastOccurance = {}
    uniqueText = list(set(text))
    for char in uniqueText:
        value = GetLastOccurance(pattern, char)
        lastOccurance.update({char : value})
    return lastOccurance

# Mengembalikan True apabila pattern ditemukan dalam text
def BMPatternMatching(text, pattern):
    lastOccurance = {}
    lastOccurance = BMLastOccurance(text, pattern)
    count = 0
    lengthText = len(text)
    lengthPattern = len(pattern)
    i = lengthPattern - 1
    while (i < lengthText):
```

```

j = lengthPattern - 1
while ((j >= 0) and (i < lengthText)):
    if (text[i] == pattern[j]):
        i -= 1
        j -= 1
    else:
        lastPosition = lastOccurance.get(text[i])
        if (lastPosition < j):
            i = i + (lengthPattern - 1) - lastPosition
        elif (lastPosition > j):
            i = i + lengthPattern - j
        else:
            i = i + 1
        j = lengthPattern - 1
i += 1
if (j < 0):
    return True
return False

```

Mengembalikan kalimat-kalimat yang mengandung pattern

```

def GetPatternBMSentence(sentence, pattern):
    sentencePattern = []
    for i in range(len(sentence)):
        if (BMPatternMatching(sentence[i], pattern)):
            sentencePattern.append(sentence[i])
    return sentencePattern

```

Algoritma Knutt-Morris-Pratt: PatternKMP.py

Modul Pattern Matching Metode 2: Algoritma KMP

Fungsi / Prosedur

Mengembalikan list of prefix dan suffix dari sebuah pattern

```

def PrefixSuffixInit(pattern, k):
    prefix = []
    suffix = []
    for i in range (k+1):
        presub = ""
        sufsub = ""
        for j in range (i+1):
            presub = presub + pattern[j]
            sufsub = pattern[k-j] + sufsub
        prefix.append(presub)
        suffix.append(sufsub)
    suffix.pop()
    return prefix, suffix

```

Mengembalikan list yang berisi border function

```

def KMPBorderFunction(pattern):

```

```

length = len(pattern)
borderFunction = []
for i in range (1, length):
    prefix = []
    suffix = []
    k = i-1
    prefix, suffix = PrefixSuffixInit(pattern, k)
    maxLength = 0
    for j in range (k):
        if (prefix[j] == suffix[j]):
            maxLength = len(prefix[j])
    borderFunction.insert(k,maxLength)
return borderFunction

# Mencari banyaknya pattern dalam sebuah text
def KMPPatternMatching(text, pattern):
    borderFunction = KMPBorderFunction(pattern)
    count = 0
    lengthText = len(text)
    lengthPattern = len(pattern)
    i = 0
    while (i < lengthText):
        j = 0
        while ((j < lengthPattern) and (i < lengthText)):
            if (text[i] == pattern[j]):
                i += 1
                j += 1
            elif (j > 0):
                j = borderFunction[j-1]
            else:
                j = 0
                i += 1
        if (j >= lengthPattern):
            return True
    return False

# Mengembalikan kalimat-kalimat yang mengandung pattern
def GetPatternKMPSentence(sentence, pattern):
    sentencePattern = []
    for i in range(len(sentence)):
        if (KMPPatternMatching(sentence[i], pattern)):
            sentencePattern.append(sentence[i])
    return sentencePattern

```

Regular Expression: PatternRegex.py

```

# Modul Pattern Matching Metode 3: Regular Expression
import re

```

```

# Fungsi / Prosedur
# Mencari pattern dalam text dengan metode regex
def RegexPatternMatching(text, pattern):
    found = re.search(pattern, text)
    return found

# Mengembalikan kalimat-kalimat yang mengandung pattern
def GetPatternRegex(sentence, pattern):
    sentencePattern = []
    for i in range(len(sentence)):
        if (RegexPatternMatching(sentence[i], pattern)):
            sentencePattern.append(sentence[i])
    return sentencePattern

```

2.4 Extraction Data dengan Regular Expression

Berikut ini adalah regular expression yang penulis definisikan dan implementasikan dalam aplikasi pengekstrak data ini.

Extraction Data: ExtractionData.py

```

# Modul Extractor Data: Memanfaatkan Regular Expression

from nltk.tokenize import sent_tokenize, word_tokenize
import re

# Kamus Regular Expression
countAble = '(:kasus|orang|manusia|korban|jiwa|pasien|meninggal|sembuh|kematian)'
numberFirstFormat = "(?:(?:\d{1,3}\.?)*(?:\d+) %s)" % (countAble)
numberEnum = '(:satu|dua|tiga|empat|lima|enam|tujuh|delapan|sembilan|sepuluh|puluh|ribu|juta)'
numberSecondFormat = "(?:(:%s )+%s %s)" % (numberEnum, numberEnum, countAble)
genericNumber = re.compile("(?:%s|%s)" % (numberFirstFormat, numberSecondFormat))

# Construct Format for Date
datenum = '(:\d{1,2})'
tahun = '(:\d{4})'
bulan = '(:januari|februari|maret|april|mei|juni|juli|agustus|september|oktober|november|desember)'
bulanSingkat = '(:jan|feb|mar|apr|mei|jun|jul|ags|sep|okt|nov|des)'
day = '(:senin|selasa|rabu|kamis|jumat|sabtu|minggu)'
dateFirstFormat = "(?:%s, %s(?:%s|%s) %s)" % (day, datenum, bulan, bulanSingkat, tahun)
dateSecondFormat = "(?:\d{1,2}/\d{1,2}/\d{4})"
dateThirdFormat = '(:kemarin|besok|hari ini|lusa)'
dateOtherFormat = "(?:%s|%s|%s|%s)" % (day, datenum, bulan, bulanSingkat, tahun)

# Construct Format for Time

```

```

timeFirstFormat = '(:\d{2}:\d{2} wib)'
timeSecondFormat = '(:pukul \d{2}\.\d{2} wib)'
timeThirdFormat = '(:pagi|siang|sore|malam)'

# Combination of Date and Time
dateTimeFull = "(?:%s,(?:%s|s))" % (dateFirstFormat, timeFirstFormat, timeSecondFormat)

# Combine all things into one regular expression
genericDate = "(?:%s|s|s|s)" % (dateFirstFormat, dateSecondFormat, dateThirdFormat, dateOtherFormat)
genericTime = "(?:%s|s|s)" % (timeFirstFormat, timeSecondFormat, timeThirdFormat)
genericDateTime = re.compile("(?:%s|s|s)" % (dateTimeFull, genericDate, genericTime))

# Fungsi / Prosedur
# Mengembalikan string dari pembacaan pada sebuah fileName
def ReadText(fileName):
    f = open("test/"+fileName, "r")
    text = f.read()
    f.close()
    return text

# Mengekstrak data angka dari sebuah kalimat
def FindNumber(sentence):
    angka = re.findall(genericNumber, sentence)
    if (len(angka) == 0):
        return "Tidak diketahui"
    else:
        return angka[0]

# Mengekstrak data tanggal dari sebuah kalimat
def FindDate(sentence):
    tanggal = re.findall(genericDateTime, sentence)
    if (len(tanggal) == 0):
        return "Tidak diketahui"
    else:
        return tanggal[0]

# Mengekstrak data artikel dari sebuah teks
def ArticleDate(text):
    tanggal = re.findall(genericDateTime, text)
    if (len(tanggal) == 0):
        return "Tidak diketahui"
    else:
        return max(tanggal, key = len)

```

2.5 Algoritma Pengekstrakan Data Secara Keseluruhan

Berikut ini adalah potongan kode program terkait pengekstrakan data dari berbagai artikel dengan keyword dan metode algoritma pencocokkan yang digunakan.

Extractor Full: ExtractorApp.py

```
# Modul ExtractorApp : Method Utama dalam Menjalankan Ekstraksi

from PatternBM import *
from PatternKMP import *
from PatternRegex import *
from ExtractorData import *

# Melakukan highlight terhadap matched word dengan tampilan bold
def BoldMatches(sentence, matches):
    return sentence.replace(matches, "<b>"+matches+"</b>")

# Memulai Proses Ekstraksi Data dari beberapa Input Files
def BeginExtraction(keyword, method, files):
    listOfResult = [] # menyimpan daftar teks dari seluruh teks yang dipilih
    # Struktur listOfResult: (namafile, teks, extractedSentence, jumlahSentence)
    # Struktur extractedSentence : (sentencePattern, jumlah, tanggal)

    # Algoritma Utama
    for i in range(len(files)):
        extractedSentence = [] # menyimpan daftar kalimat yang sudah terekstraksi
        text = [] # menyimpan string dari sebuah text yang dibaca
        sentence = [] # menyimpan daftar kalimat yang telah diparsing
        sentencePattern = [] # menyimpan daftar kalimat yang mempunyai pattern

        text = ReadText(files[i])
        text = text.lower()
        articleDate = ArticleDate(text)
        sentence = sent_tokenize(text)

        # Pattern Matching
        if (method == "Algoritma Boyers-Moore"):
            sentencePattern = GetPatternBMSentence(sentence, keyword)
        elif (method == "Algoritma Knuth-Morris-Pratt"):
            sentencePattern = GetPatternKMPSentence(sentence, keyword)
        elif (method == "Regular Expression"):
            sentencePattern = GetPatternRegex(sentence, keyword)

        # Jika tidak ditemukan kalimat yang mengandung keyword
        if (len(sentencePattern) == 0):
            print("Pencarian tidak ditemukan")
            listOfResult.append((files[i], text, sentencePattern, 0))
```

```

        # Jika ditemukan kalimat yang mengandung keyword
    else:
        # Extraction Data
        for j in range(len(sentencePattern)):
            jumlah = FindNumber(sentencePattern[j])
            tanggal = FindDate(sentencePattern[j])
            if not(jumlah == "Tidak diketahui"):
                sentencePattern[j] = BoldMatches(sentencePattern[j], jumlah)
            if not(tanggal == "Tidak diketahui"):
                sentencePattern[j] = BoldMatches(sentencePattern[j], jumlah)
            else:
                tanggal = articleDate
                sentencePattern[j] = BoldMatches(sentencePattern[j], keyword)
            extractedSentence.append((sentencePattern[j].capitalize(), jumlah, tanggal.capitalize()))

        listOfResult.append((files[i], text, extractedSentence, len(sentencePattern)))
    return listOfResult

```

2.6 Web Application: Menggunakan Flask

Web Application merupakan inti yang membuat program ini dapat dibangkitkan dengan baik. Web Application memuat segala keperluan input / output yang dibutuhkan dalam bentuk interface yang memudahkan pengguna untuk berinteraksi dengan aplikasi. Berikut ini adalah potongan kode terkait integrasi aplikasi antara Front End (HTML dan CSS tidak dilampirkan dalam laporan) dan Back End (Python) dalam bentuk aplikasi web.

Web Application: Application.py

```

# Main Program: Application.py

from flask import Flask, redirect, render_template, url_for, request
from ExtractorApp import *

# Generate HTML Based on Flask
app = Flask(__name__)

# Halaman Web Bagian Home / Menu Awal
@app.route("/", methods=["POST", "GET"])
@app.route("/home", methods=["POST", "GET"])
def home():
    if request.method == "POST":
        keyword = request.form['keyword']
        method = request.form['matchmethod']
        files = request.form.getlist('myfile')
        return redirect(url_for("result", kyw = keyword, ptr = method, fls = files))

```

```

        return render_template('home.html')

# Halaman Web Bagian Hasil Ekstraksi
@app.route("/result")
def result():
    keyword = request.args.get('kyw')
    pattern = request.args.get('ptr')
    files = request.args.getlist('fls')
    result = BeginExtraction(keyword, pattern, files)
    return render_template("result.html", keyword = keyword, pattern = pattern ,files = files, result = result)

# Halaman Web Bagian Guide Line / How To Use
@app.route("/guideline")
def guideline():
    return render_template('guideline.html', title='About')

# Menjalankan Aplikasi
if __name__ == '__main__':
    app.run(debug=True)

```


BAB III

PENGUJIAN PROGRAM

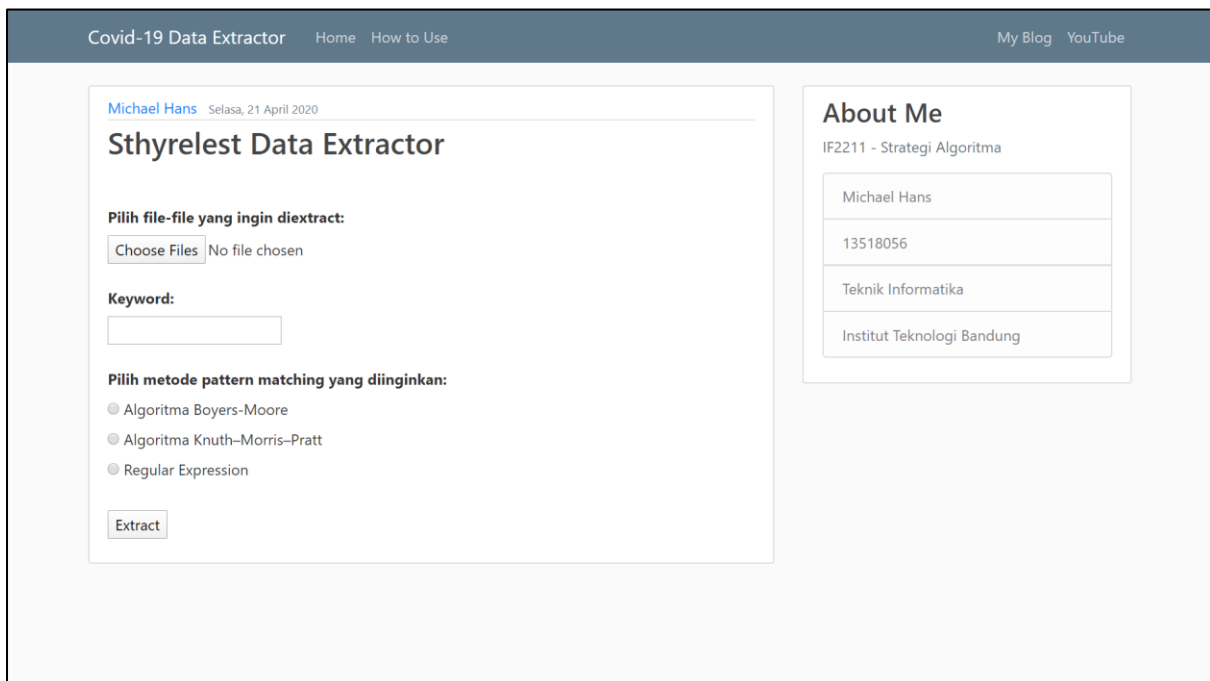
3.1 Spesifikasi Komputer

Berikut ini adalah spesifikasi dari komputer yang digunakan dalam mengimplementasikan dan mengeksekusi program polinom tersebut.

Operating System : Windows 10 Home Single Language 64-bit (10.0, Build 18362)
CPU / Processor : Intel® Core™ i7-8550U CPU @ 1.80 GHz (8 CPUs), ~2.0 GHz
RAM : 8192 MB
VGA : NVIDIA GeForce MX130 2 GB

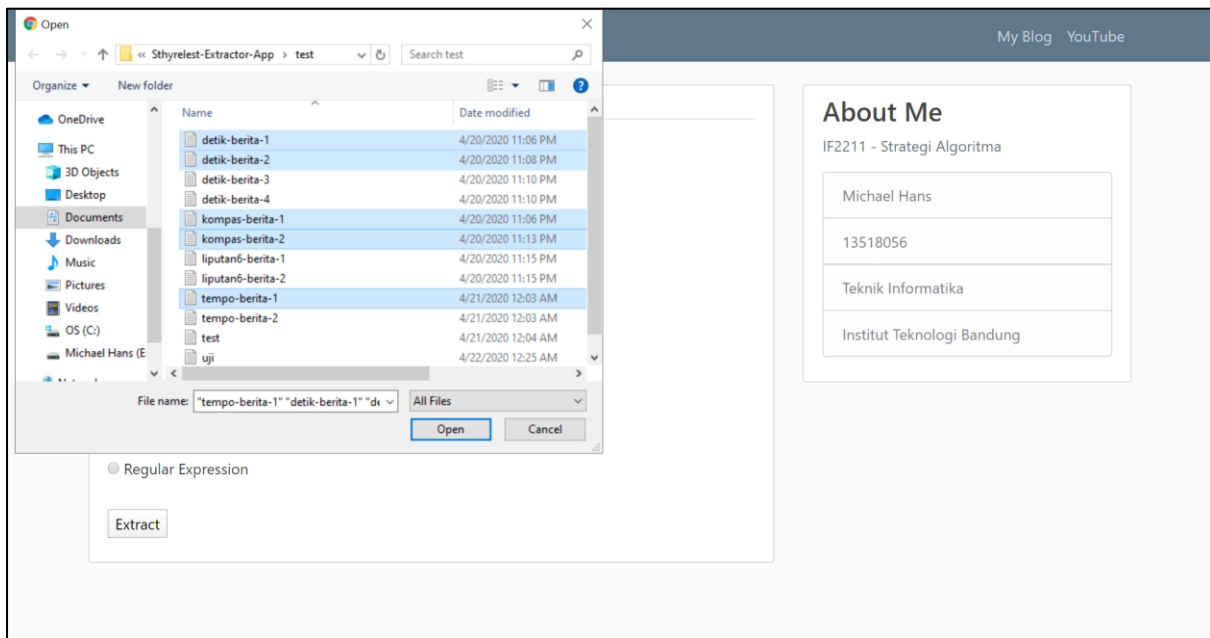
3.2 Pengujian Program

Berikut ini adalah beberapa screenshot yang dilakukan terkait penggunaan aplikasi extractor yang telah diimplementasikan.

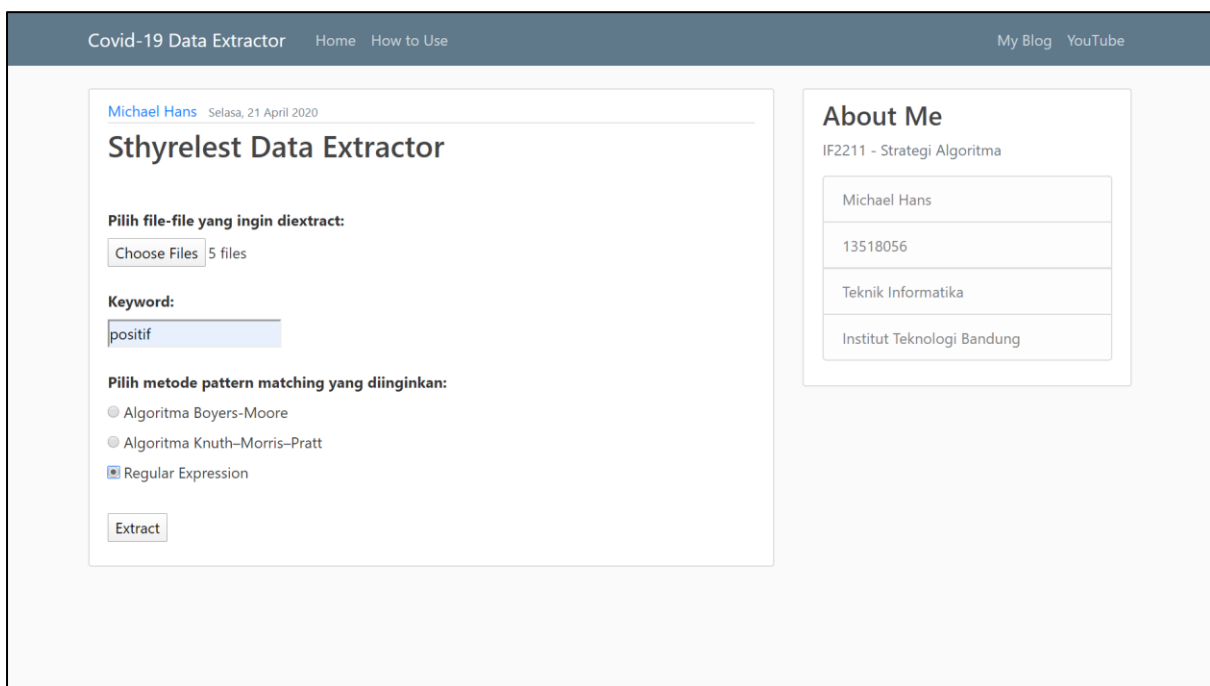


The screenshot displays the 'Covid-19 Data Extractor' web application. The header includes navigation links for 'Home' and 'How to Use', along with 'My Blog' and 'YouTube'. The main content area is titled 'Sthyrelest Data Extractor' and includes a user profile for 'Michael Hans' dated 'Selasa, 21 April 2020'. The interface features a file selection section with a 'Choose Files' button and a 'No file chosen' status. Below this is a 'Keyword:' input field. A section for 'Pilih metode pattern matching yang diinginkan:' offers three radio button options: 'Algoritma Boyers-Moore', 'Algoritma Knuth-Morris-Pratt', and 'Regular Expression'. An 'Extract' button is positioned at the bottom of the form. To the right, an 'About Me' sidebar identifies the user as 'Michael Hans' with ID '13518056', major 'Teknik Informatika', and affiliation 'Institut Teknologi Bandung'.

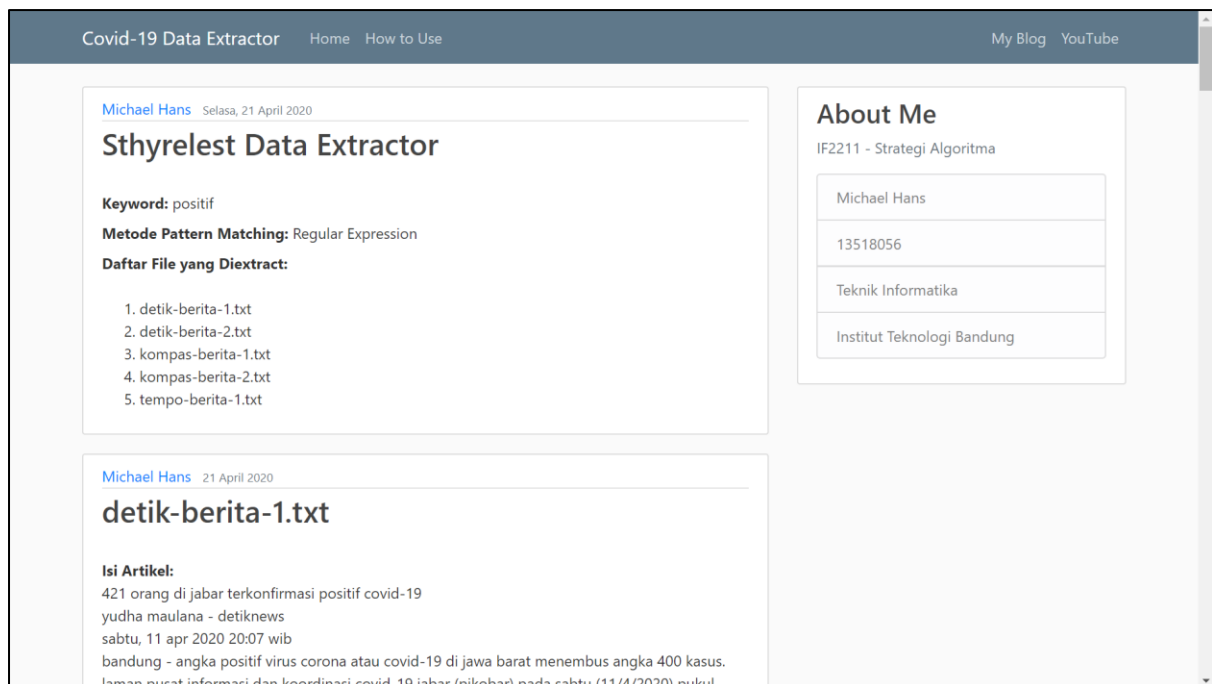
Tampilan Awal Extractor Application



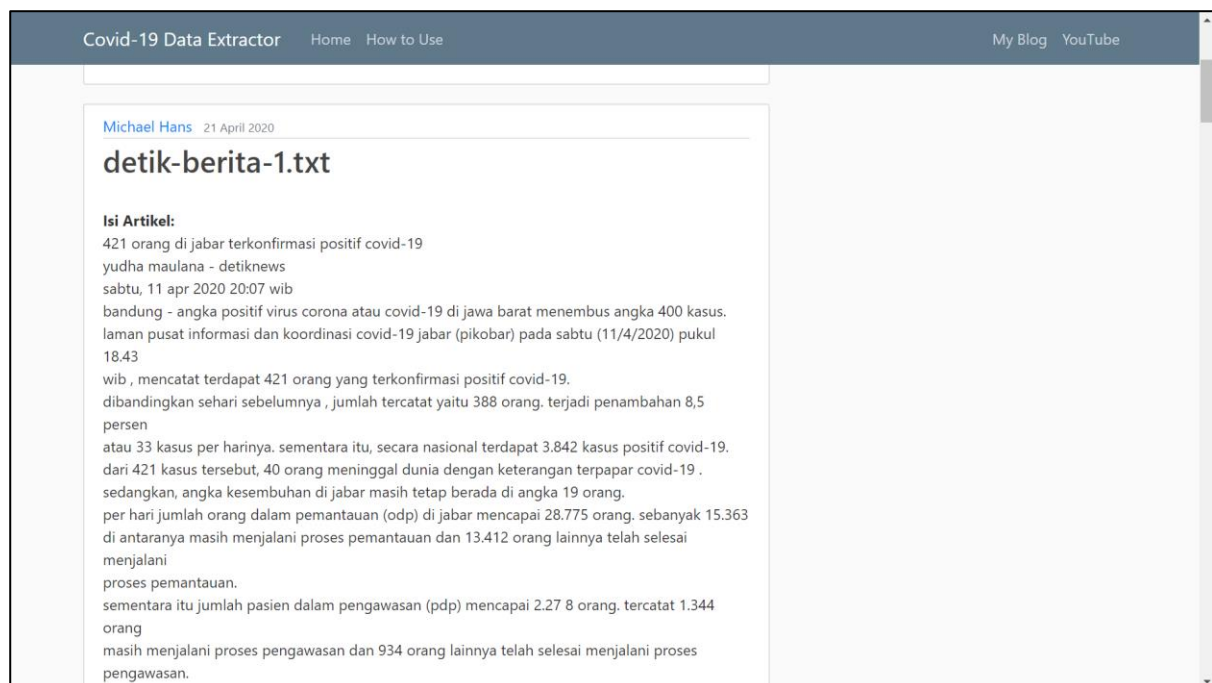
Melakukan browse file terhadap artikel-artikel yang ingin diekstraksi datanya



Tampilan sesudah tiga buah parameter diisi oleh pengguna



Tampilan Awal Setelah Ekstraksi Data Berhasil



Kasus 1: Hasil Ekstraksi Data: Menampilkan Kembali Artikel yang Diekstrak

Covid-19 Data Extractor
Home
How to Use
My Blog
YouTube

di antaranya masih menjalani proses pemantauan dan 13.412 orang lainnya telah selesai menjalani proses pemantauan. sementara itu jumlah pasien dalam pengawasan (pdp) mencapai 2.278 orang. 934 orang masih menjalani proses pengawasan dan 934 orang lainnya telah selesai menjalani proses pengawasan.

Pencarian keyword positif ditemukan dalam 3 kalimat

Kalimat-kalimat yang mengandung positif:

- 421 orang** di jabar terkonfirmasi **positif** covid-19 yudha maulana - detiknews sabtu, 11 apr 2020 20:07 wib bandung - angka **positif** virus corona atau covid-19 di jawa barat menembus angka 400 kasus. (detik-berita-1.txt)
Jumlah : **421 orang**
Waktu : **Sabtu, 11 apr 2020 20:07 wib**
- Laman pusat informasi dan koordinasi covid-19 jabar (pikobar) pada sabtu (11/4/2020) pukul 18.43 wib , mencatat terdapat **421 orang** yang terkonfirmasi **positif** covid-19. (detik-berita-1.txt)
Jumlah : **421 orang**
Waktu : **11/4/2020**
- Sementara itu, secara nasional terdapat **3.842 kasus positif** covid-19. (detik-berita-1.txt)
Jumlah : **3.842 kasus**
Waktu : **Sabtu, 11 apr 2020 20:07 wib**

Kasus 1: Menampilkan Hasil Ekstraksi Data dari Artikel pada detik-berita-1.txt

Covid-19 Data Extractor
Home
How to Use
My Blog
YouTube

Michael Hans 21 April 2020

detik-berita-2.txt

Isi Artikel:
update corona di banten: versi Pemprov 263 positif, pusat 341 positif
bahtiar rifa'i - detiknews
senin, 20 apr 2020 21:41 wib

tangerang - jumlah kasus terkonfirmasi positif di provinsi banten bertambah 11 orang sehingga total ada 262 kasus seperti yang diumumkan tim gugus tugas penanganan covid-19 provinsi banten pada senin (20/4/2020), pukul 18.00 wib.

dari total 263 kasus, 181 orang masih dirawat, 39 sembuh dan 43 meninggal. ada penambahan dua pasien meninggal dan juga dua pasien sembuh pada hari ini dibandingkan hari sebelumnya.

data ini berbeda dengan yang diumumkan pemerintah pusat, di mana total kasus positif di banten sudah mencapai 341 pasien positif, 17 orang sembuh dan 35 meninggal. ada penambahan delapan pasien yang sembuh dibandingkan hari sebelumnya.

penambahan 11 kasus versi gugus tugas penanganan covid-19 banten berada di kota tangerang sebanyak tujuh orang, tangerang selatan (tangselsel) dua orang dan kabupaten tangerang dua orang.

untuk kategori pdp corona, keseluruhan pasien tercatat adalah 1.137 orang. 842 orang masih dirawat, 179 sembuh dan 116 meninggal dunia. adapun odp yang masih dipantau ada 2.250 orang dan dinyatakan sembuh dan lepas dari pantauan ada 3.320. jumlah odp tercatat ada 5.570

Kasus 2: Menampilkan isi artikel dari detik-berita-2.txt

Covid-19 Data Extractor		Home	How to Use	My Blog	YouTube
<p>1. Update corona di banten: versi Pemprov 263 positif, pusat 341 positif bahtiar rifa'i - detiknews senin, 20 apr 2020 21:41 wib tangerang - jumlah kasus terkonfirmasi positif di provinsi banten bertambah 11 orang sehingga total ada 262 kasus seperti yang diumumkan tim gugus tugas penanganan covid-19 provinsi banten pada senin (20/4/2020), pukul 18.00 wib. (detik-berita-2.txt) Jumlah : 11 orang Waktu : Senin, 20 apr 2020 21:41 wib</p> <p>2. Data ini berbeda dengan yang diumumkan pemerintah pusat, di mana total kasus positif di banten sudah mencapai 341 pasien positif, 17 orang sembuh dan 35 meninggal. (detik-berita-2.txt) Jumlah : 341 pasien Waktu : Senin, 20 apr 2020 21:41 wib</p> <p>3. Pemprov menyampaikan data positif terjangkit corona berdasarkan domisili pasien dan disampaikan di situs resmi https://infocorona.bantenprov.go.id/. (detik-berita-2.txt) Jumlah : Tidak diketahui Waktu : Senin, 20 apr 2020 21:41 wib</p> <p>4. "kami menggunakan data asal domisili bukan berdasarkan dia positifnya di daerah banten. (detik-berita-2.txt) Jumlah : Tidak diketahui Waktu : Senin, 20 apr 2020 21:41 wib</p> <p>5. Berikut sebaran kasus pasien positif covid-19 berdasarkan versi Pemprov Banten; 1. kota tangerang: 76 dirawat, 21 sembuh 18 meninggal, total 115 kasus. (detik-berita-2.txt) Jumlah : 115 kasus Waktu : Senin, 20 apr 2020 21:41 wib</p>					

Kasus 2: Menampilkan hasil ekstraksi data dari detik-berita-2.txt

Covid-19 Data Extractor		Home	How to Use	My Blog	YouTube
<p>Michael Hans 21 April 2020</p> <h2>tempo-berita-1.txt</h2> <p>Isi Artikel: kasus baru virus corona di rusia bertambah reporter: non koresponden editor: suci sekarwati senin, 20 april 2020 21:30 wib</p> <p>tempo.co, jakarta - jumlah kasus virus corona di rusia per minggu, 19 april 2020 naik sebanyak 4.268 atau total menjadi 47.121 kasus. jumlah pasien yang meninggal karena virus ini total 405 orang, dimana 44 pasien meninggal dalam tempo 24 jam.</p> <p>tim gawat darurat rusia dalam keterangan tertulis menyebut ada 155 kasus virus corona yang sembuh minggu lalu. dengan begitu, total ada 3.446 pasien yang berhasil sembuh dari virus mematikan ini.</p> <p>dikutip dari aa.com.tr, hampir separuh pasien baru virus corona di rusia atau sekitar 43 persen adalah pasien asymptomatic atau tanpa gejala. diagnosa diperoleh melalui tes.</p> <p>virus corona menyebar pertama kali di kota wuhan, china, pada desember 2019. virus yang juga populer dengan sebutan covid-19, telah menyebar ke 185 negara dan teritorial.</p> <p>eropa dan amerika serikat saat ini menjadi wilayah paling terpukul oleh virus corona. data yang diungkap universitas johns hopkins di amerika serikat menyebut 165 ribu orang meninggal karena virus corona dan 629 ribu orang yang sembuh. total di seluruh dunia, ada 2,4 juta kasus virus corona.</p> <p>Pencarian keyword positif tidak ditemukan</p>					

Kasus 3: Menampilkan hasil ekstraksi data dari tempo-berita-1.txt: Keyword positif tidak ditemukan

3.3 Checklist Program

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 data uji		✓

DAFTAR PUSTAKA

Rinaldi Munir. “Pencocokan String”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-\(2018\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf) (diakses pada 15 April 2020)

Rinaldi Munir. “String Matching dengan

Regex”<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf> (diakses pada 20 April 2020)

Nur Ulfa Maulidevi. “Pencocokan String dengan Algoritma Knutt-Morris-Pratt”.

<https://www.youtube.com/watch?v=5oSQnywqm0M&>.

Nur Ulfa Maulidevi. “Pencocokan String dengan Algoritma Boyer-Moore”.

<https://www.youtube.com/watch?v=AvFx1vc32xY&>

Masayu Leylia Khodra, Yudi Wibisono. Regular Expression.

https://docs.google.com/document/d/1TUkkiQNL_nwbP0Vq46K5OL1_7dACrJLrC/edit
(diakses pada 20 April 2020)