Penerapan Algoritma Pencocokan String dalam Pencarian Kata Dasar Bahasa Indonesia

Muhammad Rizal Muhaimin 13519136
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519136@std.stei.itb.ac.id

***Abstract*—Kata dasar adalah satuan terkecil dalam bahasa yang memiliki arti atau makna. Di zaman sekarang, tidak jarang orang melupakan kata dasar yang membangun suatu kalimat. Kata dasar juga bisa digunakan dalam menghitung kemiripan suatu query terhadap dokumen yang sedang dicari. Maka dari itu penerapan algoritma pencocokan *string* dalam menentukan kata dasar merupakan hal yang penting. Dalam kasus ini, Hanya berfokus pada kosakata dasar dalam Bahasa Indonesia.**

Keywords—Kata Dasar; Pencocokan String; Bahasa Indonesia

I. PENDAHULUAN

Kata dasar adalah satuan terkecil dalam bahasa yang memiliki arti dan makna. Setiap kata atau kalimat yang ada pasti dibangun atau diturunkan dari kata dasar yang ada. Terkadang proses imbuhan (awalan, akhiran, atau keduanya) membuat kata turunan sulit untuk ditentukan apa kata dasar yang membangun kata tersebut.

Di zaman sekarang, tidak jarang orang melupakan kata dasar apa yang membangun suatu kalimat atau bagaimana menulis kata dasar yang benar. Kata dasar sebenarnya bisa digunakan dalam menghitung kemiripan suatu query terhadap dokumen yang dicari. Dimana query dan dokumen akan diubah dalam bentuk kumpulan kata dasar dan kemudian dihitung kemiripannya.

Sebenarnya kata dasar dari suatu kata atau teks dapat dicari dengan penerapan algoritma pencocokan string. Setiap teks inputan akan dicocokan dengan basis data kata dasar yang ada kemudian setiap kata dari teks akan diganti dengan kata dasar yang sesuai dengan kata tersebut. Dalam penerapan algoritma pencocokan string ini, hanya berfokus pada kata dasar yang ada di Bahasa Indonesia.

II. LANDASAN TEORI

A. Karakter, String, dan Substring

Karakter adalah huruf, angka, ruang, symbol khusus yang merupakan unit terkecil dari suatu kata yang mana terdaftar dalam kumpulan karakter dalam ASCII.

String merupakan salah satu tipe data yang ada karena gabungan dari beberapa karakter untuk menjadi suatu kesatuan dan biasa disebut sebagai teks. String juga bisa dikatakan sebagai kumpulan karakter yang diletakkan secara sekuensial.

Jika terdapat String S dengan panjang n, maka dapat didefinisikan sebagai berikut:

$$S = x_0\, x_1\, x_2 \ldots x_{n-2}\, x_{n-1}$$

Substring adalah himpunan bagian string yang berisikan urutan karakter yang terletak berdekatan pada suatu string. Sebagai contoh terdapat suatu string “Hari ini hari ulang tahunku” maka salah satu substring yang mungkin adalah “Hari ini” dimana setiap karakter yang adal adalah bagian dari string yang saling berdekatan.

Dalam substring sendiri ada istilah *prefix* (awalan) dan *suffix* (akhiran). Suatu substring dapat dikatakan sebagai *prefix* apabila letaknya sebagai “awal” dari suatu string yang mana setiap karakter yang ada saling berdekatan dan saling terurut dari depan ke belakang. Sedangkan sesuatu dapat dikatakan sebagai *suffix* apabila letaknya sebagai “akhir” dari suatu string yang mana setiap karakter yang ada saling berdekatan dan saling terurut dari belakang ke depan. Sebagai contoh terdapat string “Hari Senin” sebagai berikut:

H	a	r	i		S	e	n	i	n

Maka *prefix* yang mungkin dalam string diatas adalah P = {H, Ha, Har, Hari, Hari , Hari S, Hari Se, Hari Sen, Hari Seni, Hari Senin}. Sedangkan *suffix* yang mungkin dalam string diatas adalah P = {n, in, nin, enin, Senin, Senin, i Senin, ri Senin, ari Senin, Hari Senin}.

B. Pengertian Algoritma Pencocokan String

Algoritma pencocokan string adalah algoritma yang digunakan dalam mencari kemunculan suatu kata dalam rangkaian kalimat. Dalam dunia pemrograman, kata yang dicari disebut dengan *pattern sedangkan* rangkaian kalimat disebut sebagai teks. Dengan algoritma ini, kita bisa mendeteksi apakah *pettern* ada didalam suatu teks atau mencari letak indek dimana *pattern* ditemukan.

C. Jenis Algoritma Pencocokan String

a. Algoritma Brute Force

Algoritma Brute Force adalah algoritma pencocokan atau pencarian string yang melakukannya secara iterasi satu persatu dari setiap karakter yang ada di *pattern* maupun karakter yang ada di teks. Ide dari algoritma ini adalah membandingkan karakter pertama dari teks dengan karakter pertama pada *pattern*. Jika terjadi *missmatch* atau karakter tidak sama, maka karakter yang ada di teks akan digeser satu karakter ke kanan. Jika karakter di *pattern* sama pada karakter di teks maka karater yang di *pattern* lah yang akan bergerser kekanan. Langkah tersebut akan dilakukan sampai karakter di teks sudah diperiksa atau karakter *pattern* ada dalam teks sehingga iterasi berhenti.

Text:

A	B	A	C	A	D	A	A	B

Pattern:

A	B

Proses Algoritma:

Teks	A	B	A	C	A	D	A	A	B
Proses 1	A	D	-	-	-	-	-	-	-
Proses 2	-	A	D	-	-	-	-	-	-
Proses 3	-	-	A	D	-	-	-	-	-
Proses 4	-	-	-	A	D	-	-	-	-
Proses 5	-	-	-	-	A	D	-	-	-

Kompleksitas algoritma brute force pada kasus terburuk yaitu O(m*n). Kasus terburuk terjadi ketika string sering terjadi mismatch pada akhir dari string pattern. Sedangkan, kompleksitas algoritma pada kasus terbaik yaitu O(n) yang terjadi ketika mismatch sering kali ditemukan pada awal karakter dari string pattern. Kompleksitas algoritma untuk rata-rata kasus yaitu O(m+n), yang mana kompleksitas waktu tersebut juga sudah cukup cepat. Algoritma brute force sangat tidak efektif apabila string yang dicocokkan tidak bervariasi. Akan tetapi, algoritma brute force ini cukup cepat jika string yang dicocokkan cukup beragam.

b. Algoritma Knuth-Morris-Pratt

Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang akan melakukan pencocokan terhadap suatu string pattern pada string text yang dimulai dari kiri ke kanan. Saat terjadinya mismatch antara teks(T) pada T[i] dengan pattern (P) pada P[j] atau dengan kata lain T[i] ≠ P[j], maka langkah yang kita lakukan adalah dengan menggeser pattern sedemikian rupa sehingga largest prefix Pattern[0..j-1] adalah sama dengan suffix pattern dari P[1..j-1] dan sejajar. Proses ini dapat dipersingkat dengan kita membuat tabel fungsi pinggiran atau border function yang mana fungsi ini memberikan nilai pergeseran terbesar yang bisa ditempuh oleh pencarian string.

Border Function adalah suatu fungsi yang menyatakan panjang prefix dari string pattern yang juga merupakan suffix dari string tersebut. Border Function didefinisikan dengan b(k) menyatakan border function ketika terjadi mismatch pada indeks sebelum mismatch tersebut.

Langkah Algoritma Knuth-Morris-Pratt adalah sebagai berikut.

1. Kita merancang border function yang melakukan membuat list tiap prefix pattern dan menghitung nilai largest prefix pattern [0..j-1] yang sama dengan suf ix pattern dari P[i..j-1] dan sejajar dengan teks bersangkutan.
2. Selanjutnya kita memulai pencocokan string dari indeks awal teks T[0] dan pattern P[0]. Selama belum ditemukan atau indeks teks(i) belom lebih dari panjang teks(len(i)), maka kita melakukan dua hal.
3. Jika teks T[i] sama dengan pattern P[j], maka i digeser satu ke kanan begitu juga j digeser satu ke kanan (i= i+1, j= j+1).
4. Jika terjadi mismatch di indeks j >0¸ maka kita memulai mencari pattern di table Border Function.
5. Jika terjadi mismatch di indeks j =0, maka indeks i digeser 1 dan indeks j dimulai dari 0.

Text:

A	B	A	C	A	A	B	A	C	A	B

Pattern:

A	B	A	C	A	B

Perhitungan *border function* untuk string *pattern* adalah sebagai berikut:

j	0	1	2	3	4	5
P[j]	A	B	A	C	A	B
K	-	0	1	2	3	4
b(k)	-	0	0	1	0	1

Proses Algoritma:

A	B	A	C	A	A	B	A	C	A	B
A	B	A	C	A	B	-	-	-	-	-
-	-	-	-	A	B	A	C	A	B	-
-	-	-	-	-	A	B	A	C	A	B

Kompleksitas waktu untuk pencarian string dengan algoritma KMP adalah O(m+n). Pencocokan string dengan algoritma KMP jauh lebih cepat dibandingkan dengan algoritma brute force. Algoritma ini sangat cocok untuk melakukan pemrosesan pada string yang sangat panjang dan juga jarang terjadinya mismatch yang berarti string tersebut tidak cukup beragam. Namun, algoritma ini akan kurang begitu efektif apabila string yang dicocokkan kemungkinan terjadi mismatch nya cukup tinggi, atau dengan kata lain, string nya memiliki huruf yang cukup bervariasi.

c. Algoritma Boyer-Moore

Algoritma pencocokkan string Boyer-Moore adalah algoritma yang dikembangkan oleh J. Strother Moore dan Robert S. Boyer pada tahun 1977. Algoritma Boyer-Moore tergolong cukup efisien dibandingkan algoritma-algoritma lainnya pada umumnya. Hal yang membuat algoritma ini spesial karena algoritma ini menggunakan 2 teknik yang mampu membuat pencocokkan antara pattern dan teks seminimal mungkin namun tetap memberikan hasil yang optimal.

Ide algoritma ini adalah kita melakukan pencocokan dari kanan ke kiri dan melakukan 'lompatan' antara karakter sehingga lebih efisien. Dua teknik yang digunakan dalam algoritma ini adalah melakukannya dengan looking-glass teknik dimana memeriksa kecocokan pattern yang dimulai dari indeks terakhir P bergerak dari kiri ke kanan. Dan selanjutnya kita bisa melakukannya dengan teknik character jump dimana terdapat mismatch maka kita melakukan jump terhadap indeks text sehingga pemeriksaan berlangsung lebih efektif. Namun sebelum melakukannya kita membuat terlebih dahulu tabel last occurrence. Tabel ini berisikan indeks kemunculan terakhir karakter pada pattern. Untuk karakter-karakter yang tidak terdapat pada string pattern, maka kita mendefinisikan nilai karakter tersebut dengan -1, yang artinya tidak ada pada pattern.

Langkah yang kita bisa lakukan untuk membentuk algoritma Boyer-Moore adalah sebagai berikut.

1. Kita mendefinisikan terlebih dahulu table Last Occurrence mengenai informasi kemunculan terakhir suatu karakter pada pattern P. Untuk karakter yang tidak ada di pattern, kita definisikan dengan nilai -1.
2. Selanjutnya kita akan melakukan pencocokan string dari indeks Pattern dari kanan ke kiri P[len(pattern)..0]. Selama belum ditemukan atau indeks teks(i) belom lebih dari panjang teks(len(i)), maka kita akan melakukan 3 kasus.

   KASUS 1, Jika teks T[i] nilainya sama dengan Pattern P[j], maka kita melakukan 2 hal.

   a) Jika indeks Pattern j sudah 0, pemeriksaan berhenti. Pattern sudah matching
   b) Jika belum, indeks i dan j sama sama digeser sekali ke kiri

   KASUS 2, Jika indeks teks lebih kecil nilainya dengan yang di table Last Occurrence, maka lakukan i new = i + m-(io+1)

   KASUS 3, jika indeks teks lebih besar nilainya dengan yang di table Last Occurrence, maka lakukan i new = i + m- j

Text:

A	B	A	C	A	A	B	A	D

Pattern:

A	B	A	C	A	B

Tabel *last occurrence*

X	A	B	C	D
L(x)	4	5	3	-1

Proses Algoritma:

Teks	A	B	A	C	A	A	B	A	D
Proses 1	A	B	A	C	A	B	-	-	-
Proses 2	-	A	B	A	C	A	B	-	-
Proses 3	-	A	B	A	C	A	B	-	-
Proses 4	-	A	B	A	C	A	B	-	-
Proses 5	-	-	A	B	A	C	A	B	-

Kompleksitas algoritma untuk algoritma Boyer-Moore pada kasus terburuk yaitu O(nm+A). Algoritma Boyer-Moore akan sangat efektif apabila alfabet A ini cukup panjang, dengan A adalah variasi alfabet yang terdapat pada string teks. Algoritma ini akan berjalan dengan lambat apabila melakukan pencocokan string untuk *binary.* Tetapi algoritma ini akan berjalan dengan efektif untuk melakukan pencocokan pada teks yang lebih bervariasi seperti teks yang berbahasa Inggris.

D. Regex

Regulrar Expression (Regex) adalah pencocokan string yang menggunakan suatu pola tertentu yang diberikan sesuai dengan notasi yang berlaku. Regular Expression akan mencari semua posisi yang pada string teks yang cocok dengan pattern regex yang diberikan. Berikut beberapa notasi yang terdapat pada regex yang bersumber dari Modul Praktikum NLP: Regex.

Notasi	Deskripsi
[abc]	a, b, atau c
[^abc]	Semua karakter selain a,b,c
[a-zA-z]	a sampai z atau A sampai Z, inclusive
.	Semua karakter
\d	Digit [0-9]
\D	Non digit [^0-9]
\s	Whitespace character
\S	Non whitespace character
\w	Word character
\W	Non wod character
X?	X muncul satu kali atau tidak sama sekali
X*	X muncul 0 atau banyak
X+	X muncul satu atau lebih
X{n}	X muncul tepat n kali
X{n,}	X muncul setidaknya n kali
X{n,m}	X muncul antara n sampai m kali
^	Awal baris
$	Akhir baris
\b	Batas kata

E. Struktur Data

Struktur Data adalah pengorganisasian data, manajemen data dan format penyimpanan suatu data yang memungkinkan pengaksesan dan modifikasi data secara efektif. Struktur data digunakan untuk menyelesaikan berbagai macam persoalan pemrograman. Dengan menggunakan struktur data, suatu persoalan dapat diselesaikan dengan efektif dan mangkus.

F. Kata Dasar

Kata dasar merupakan satuan bahasa terkecil yang memiliki makna, kata tersebut belum mengalami penambahan atau perubahan bentuk yang mengakibatkan perubahan makna.Dengan pengertian lain bahawa kata dasar ialah kata yang belum di beri imbuhan dan kata yang menjadi dasar awal pembentukan kata yang lebih besar.

III. Pencocokan String dalam Pencarian Kata Dasar Bahasa Indonesia

Dalam melakukan pencocokan string untuk mencari kata dasar, ada hal yang sangat penting dan harus diperhatikan, yaitu pemilihan bahasa. Bahasa yang digunakan sangatlah mempengaruhi bagaimana algoritma pencocokan string bisa mengetahui kata dasar melalui aturan yang ada pada bahasa tersebut. Dalam makalah ini akan berfokus pada pencarian kata dasar dalam bahasa Indonesia.

Bahasa Indonesia sendiri memiliki banyak kosakata dasar yang sebenarnya bisa di akses dari KBBI. Dalam proses pencocokan string untuk mencari kata dasar ini, memerlukan struktur data yang sesuai untuk menyimpan semua kata dasar yang ada. Banyak macam-macam struktur data yang bisa digunakan akan tetapi dalam makalah ini akan menggunakan struktur data yang sudah disediakan dalam SQLite.

Data yang ada direpresentasikan dalam bentuk tabel sederhana dimana terdapat 28524 kata dasar. Untuk melakukan pencocokan string, setiap kosa kata dasar yang ada pada SQLite akan dibentuk menjadi satu struktur data array satu dimensi. Berikut ini adalah implementasi struktur data dalam bahasa python.

```python
def createDB():
    conn = sqlite3.connect(path +'BasisKata.sqlite')
    cur = conn.cursor()
    cur.executescript("""
        CREATE TABLE IF NOT EXISTS Kata_Dasar (
            kata        TEXT NOT NULL PRIMARY KEY UNIQUE
        )""")
    conn.commit()
    conn.close()

def Insert_KataDasar():
    df = pandas.read_csv("kata_dasar_kbbi.csv")
    createDB()
    conn = sqlite3.connect(path +'BasisKata.sqlite')
    for kata in df["a"]:
        cur = conn.cursor()
        query = '''INSERT OR IGNORE INTO Kata_Dasar(kata)
        VALUES(?)'''
        value = (kata,)
        cur.execute(query,value)
    conn.commit()
    conn.close()

def Re_KataDasar():
    conn = sqlite3.connect(path +'BasisKata.sqlite')
    cur = conn.cursor()
    data =[]
    query = '''SELECT * FROM Kata_Dasar'''
    for kata in cur.execute(query):
        kata = str(kata).replace("('","")
        kata = str(kata).replace("',)","")
        data.append(kata)
    conn.close()
    return data
```

Kemudian dilakukan untuk mencari kata dasar apa yang membangun dari suatu kalimat. Pencocokan string sendiri dilakukan dengan merubah huruf besar menjadi harus kecil. Untuk persoalan algoritma pencocokan string sendiri telah dijelaskan berdasarkan landasan teori yang ada dan berikut implementasi dalam bahasa python:

1. Algoritma Brute Force

Berikut ini implementasi algoritma Brute Force.

```python
def bruteforce(teks, pattern) :
    n = len(teks)
    m = len(pattern)
    for i in range(n - m + 1) :
        j = 0
        while j < m and teks[i+j].lower() == pattern[j].lower() :
            j += 1
        if j == m :
            return i

    return -1
```

2. Algoritma Knuth-Morris-Pratt (KMP)

Dalam algoritma KMP ini, perlu dilakukan pemrosesan menghitung *border function* terlebih dahulu.

```python
def borderFunction(pattern) :
    fail = [0 for i in range(len(pattern))]
    fail[0] = 0

    m = len(pattern)
    j = 0
    i = 1
    while i < m :
        if pattern[j].lower() == pattern[i].lower() :
            fail[i] = j + 1
            i += 1
            j += 1
        elif j > 0 :
            j = fail[j - 1]
        else :
            fail[i] = 0
            i += 1

    return fail
```

Kemudian, setelah menghitung border function, dapat langsung diteruskan pada pencocokan stringnya. Berikut implementasi algoritma KMP dalam bahasa python.

```python
def kmp(teks, pattern):
    border = borderFunction(pattern)
    n = len(teks)
    m = len(pattern)

    i = 0
    j = 0

    while i < n :
        if pattern[j].lower() == teks[i].lower() :
            if j == m - 1 :
                return i - m + 1
            i += 1
            j += 1
        elif j > 0 :
            j = border[j - 1]
        else :
            i += 1

    return -1
```

3. Algoritma Boyer-Moore

Algoritma Boyer-Moore perlu pemrosesan awal untuk menghitung *last occurrence* dari variasi karakter yang ada pada string teks. Berikut implementasinya.

```python
def lastOf(pattern):
    last = {}
    i = 0
    for char in pattern:
        last.update({char: i})
        i += 1
    return last
```

Setelah mendapatkan *last occurrence*, kita dapat melanjutkan proses pencocokan string dengan algoritma Boyer-Moore. Berikut implementasinya.

```python
def boyer_moore(text, pattern):
    # hitung panjang text & pattern
    n = len(text)
    m = len(pattern)
    if (n < m) :
        # jika panjang pattern lebih besar, return INVALID
        return -1

    last = lastOf(pattern)
    i = m - 1
    j = m - 1
    while (i <= n - 1):
        if (text[i] == pattern[j]):
            # looking glass technique
            if (j == 0):
                return i
            else :
                i -= 1
                j -= 1
        else :
            # character-jump technique
            if text[i] not in last:
                # case 3. text[i] not found in pattern
                i = i + m
                j = m - 1
            elif (last[text[i]] < j) :
                # case 1. last of text[i] found in
                #  leftside of pattern[j]
                i = i + m - last[text[i]] - 1
                j = m - 1
            else : # (last[text[j]] > j)
                # case 2. last of text[i] found,
                #  but not in lefside of pattern[j]
                i = i + m - j
                j = m - 1
    # if not match, return INVALID
    return -1
```

Secara sistematis, untuk melakukan pencarian kata dasar suatu teks dapat dilakukan beberapa langkah sebagai berikut:

1. Mendata kosa kata dasar bahasa Indonesia.

   Pendataan ini dilakukan sebagai sumber kata dasar yang digunakan dalam basis data. Kata tersebut bisa mengambilnya dalam KBBI.

2. Menyimpan kosa kata dasar dalam basis data.

   Setelah mendapatkan kata kata dasar akan dimasukkan atau disimpan dalam basis data yang telah didefinisikan. Dalam makalah ini menggunakan basis data yang telah ada dalam SQLite dan menggunakan bahasa python dalam pendefinisian.

3. Merubah kosa kata dalam basis data pada suatu struktur data.

   Data dalam basis data masih bersifat tabel yang dalam penggunaannya akan diubah dalam bentuk struktur data array satu dimensi dengan bahasa python sedemikian rupa sehingga data siap untuk dilakukan pencocokan string ke tahap selanjutnya.

4. Memasukkan input kata atau teks

   Memasukkan input kata atau teks yang akan dicari kosa kata dasarnya. Masukan dalam kapital, huruf besar akan diubah menjadi huruf kecil sehingga memudahkan dalam proses pencocokan nantinya.

5. Melakukan pencocokan string

   Setelah membentuk struktur data yang siap digunakan, proses pencocokan string dapat dilakukan. Tidak ada spesifikasi khusus algoritma apa yang harus digunakan. Semua jenis algoritma yang telah didefinisikan diatas bisa digunakan akan tetapi dalam makalah ini akan menggunakan algoritma Boyer-Moore.

6. Mengganti input dengan kata dasar

   Dalam proses penggantian input dengan kosa kata dasar dilakukan ketika kata dasar dari input telah ditemukan. Dalam makalah ini untuk mencari kata dasar hanya sebatas apakah kata dasar tersebut ada di dalam input atau tidak. Untuk kata yang mengalami inputan yang cukup rumit belum diimplementasikan.

7. Menampilkan kata dasar yang sesuai

   Setelah semua kata input diubah dalam bentuk kata dasar, selanjutnya adalah menampilkan kata dasar tersebut sebagai keluaran akhir dari proses pencocokan string dalam pencarian kata dasar bahasa Indonesia.

Berikut ini adalah proses pencarian kata dasar berdasarkan langkah-langkah yang telah didefinisikan.

1. Mendata kosa kata dasar bahasa Indonesia.

   Misal kosa kata dasar dalam bahasa Indonesia yaitu “abad”, “abadi”, “buku”, “makan”, “kandas” dan seterusnya.

2. Menyimpan kosa kata dasar dalam basis data.

   Dalam basis data SQLite, kata dasar yang ada disimpan dalam bentuk tabel sederhana sebagai berikut:

1	ab
2	aba
3	aba-aba
…	…
28523	zuriah
28524	zus

3. Merubah kosa kata dalam basis data pada suatu struktur data.

   Untuk pemrosesan dengan algoritma *brute force*, KMP, dan Boyer-Moore. Kosa kata tersebut akan disimpan pada sebuah list of string seperti berikut:

ab	aba	aba-aba	…	zurafah	zuriah	zus

4. Memasukkan input kata atau teks

   Inputan dapat berupa satu kata atau kalimat baik huruf besar, kapital, atau huruf kecil.

   Contoh: "Perekonomian Indonesia sedang dalam pertumbuhan yang membanggakan"

5. Melakukan pencocokan string

   Inputan kata atau teks akan disimpan menjadi struktur data list of string sebagai berikut.

perekonomian
indonesia
sedang
dalam
pertumbuhan
yang
membanggakan

   Selanjutnya setiap kata yang ada dalam list of string kata dasar akan dilakukan pencocokan dengan list of string dari kata inputan.

ab
aba
aba-aba
…
zuriah
zus

Proses Pencocokan String:

teks: perekonomian

pattern:

ab
….
ekonomi
…
zuriah
zus

teks: indonesia

pattern:

ab
….
indonesia
…
zuriah
zus

teks: sedang

pattern:

ab
….
sedang
…
zuriah
zus

teks: dalam

pattern:

ab
….
dalam
…
zuriah
zus

teks: pertumbuhan

pattern:

ab
….
tumbuh
…
zuriah
zus

teks: yang

pattern:

ab
….
yang
…
zuriah
zus

teks: membanggakan

pattern:

ab
….
bangga
…
zuriah
zus

6. Mengganti input dengan kata dasar

   Setelah didapatkan kata dasar yang sesuai atau dalam makalah ini hanya sebatas kata dasar yang memilikai kecocokan yang terbanyak, maka kata inputan diganti dengan kata dasar yang sesuai.

perekonomian	menjadi	ekonomi
indonesia		indonesia
sedang		sedang
dalam		dalam
pertumbuhan		tumbuh
yang		yang
membanggakan		bangga

7. Menampilkan kata dasar yang sesuai

   Setelah didapatkan list of string dari kata dasar inputan yang sesuai. Selanjutnya akan ditampilkan kata dasar yang sesuai urutan dari inputan.

   Inputan: “Perekonomian Indonesia sedang dalam pertumbuhan yang membanggakan”

   Output: “ekonomi indonesia sedang dalam tumbuh yang bangga”

   Berikut merupakan contoh dari hasil pencocokan yang telah dilakukan pada langkah-langkah sebelumnya.

```
Inputan: Perekonomian Indonesia sedang dalam pertumbuhan yang membanggakan
Output:  ekonomi indonesia sedang dalam tumbuh yang bangga
```

IV. Kesimpulan Dan Saran

1. Kesimpulan

Pada proses pencocokan string banyak algoritma yang bisa digunakan baik algoritma *brute force*, KMP, dan Boyer-Moore. Dalam proses pencarian kata dasar dapat dilakukan dengan baik dan menghasilkan kata dasar yang sesuai dengan catatan kata inputan merupakan kata yang tidak memerlukan perubahan huruf dari kata dasar ke kata turunannya.

2. Saran

Untuk memperbaiki proses pencarian kata dasar yang sesuai dan lebih akurat diperlukan kajian lebih lanjut. Bagaimana suatu kata dasar dapat membentuk kata turunan sehingga diharapkan dalam proses pencocokan didapatkan kata turunan yang sesuai dengan kata inputan atau bagaimana kata turunan bisa berubah menjadi kata dasar sehingga bisa didapatkan kata dasar yang sesuai.

Link Video di Youtube

Video untuk pemaparan makalah ini dapat diakses pada link youtube sebagai berikut.

https://youtu.be/Dk53Ld1rfpw

Video tersebut berjudul "Penerapan Algoritma Pencocokan String dalam Pencarian Kata Dasar Bahasa Indonesia" yang sesuai dengan judul makalah ini.

Ucapan Terima Kasih

Penulis ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena dengan rahmat-Nya penulis dapat menyelesaikan makalah yang berjudul "Penerapan Algoritma Pencocokan String dalam Pencarian Kata Dasar Bahasa Indonesia" ini. Penulis juga berterima kasih kepada seluruh dosen pengampu mata kuliah strategi algoritma yang telah memberikan ilmu dan pelajaran yang sangat berharga untuk penulis.

Penulis juga berterima kasih kepada keluarga penulis, terutama kepada orang tua penulis, yang selalu mensupport penulis dalam menjalani perkuliahan ini dalam kondisi apapun.

Karena support yang diberikan sangat berarti bagi penulis.

Referens

[1] https://www.dosenpendidikan.co.id/kata-dasar/ Diakses 8 Mei 2021 Pukul 09:00 WIB.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf. Diakses 8 Mei 2021 Pukul 10:00 WIB.

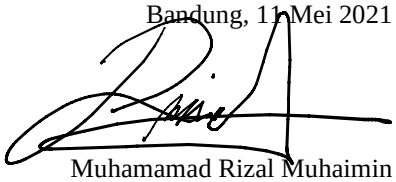
[3]https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf Diakses 8 Mei 2021 Pukul 11:00 WIB.

[4] https://kbbi.web.id/ Diakses 8 Mei 2021 Pukul 13:00 WIB.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Mei 2021



Muhamamad Rizal Muhaimin
13519136