Penerapan Algoritma String Matching dalam Implementasi Auto Complete Search menggunakan Bahasa Indonesia

Farel Winalda - 13522047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (13522047@std.stei.itb.ac.id):

***Abstract*—*Auto Complete Search* adalah fitur yang diimplementasikan pada sebuah kolom pencarian atau *address bar* baik itu dalam web browser maupun mesin pencari yang digunakan untuk membantu memberikan saran pencarian ketika kata diketikan dalam kolom pencarian. Makalah ini membahas tentang pembuatan *Auto Complete Search* yang membahas tentang masukan kata, pencarian data, serta pencarian desktipsi atau arti lain dari sebuaah kata. Pembuatan alur pencarian diimplementasikan menggunakan algoritma Pencocokan String, yaitu: *Knuth-Morris-Prath Algorithm*, *Damerau-Levenshtein Distance*, dan *Regular Expression* pada program dengan menggunakan bahasa pemrograman *Python*.**

Keywords*—*Auto Complete Search; String Matching; Knuth-Morris-Pratt Algorithm; Damerau-Levenshtein Distance; Regular Expression; Python

I. PENDAHULUAN

Dalam dunia yang semakin digital, pencarian informasi yang cepat dan akurat menjadi kebutuhan penting. *Auto Complete Search* memainkan peran besar dalam memenuhi kebutuhan ini dengan meminimalkan waktu yang diperlukan pengguna untuk menemukan informasi yang mereka cari. Fitur ini tidak hanya mempercepat proses pencarian tetapi juga membantu pengguna menemukan istilah atau frasa yang mungkin tidak terpikirkan oleh mereka sebelumnya.

Makalah ini membahas implementasi Auto Complete Search menggunakan algoritma String Matching, khususnya Knuth-Morris-Pratt (KMP) Algorithm, Damerau-Levenshtein Distance, dan Regular Expression. Algoritma-algoritma ini dipilih karena efisiensi dan kemampuannya dalam menangani berbagai macam jenis kesalahan pengetikan atau variasi input pengguna yang juga menghasilkan hasil yang paling optimal.

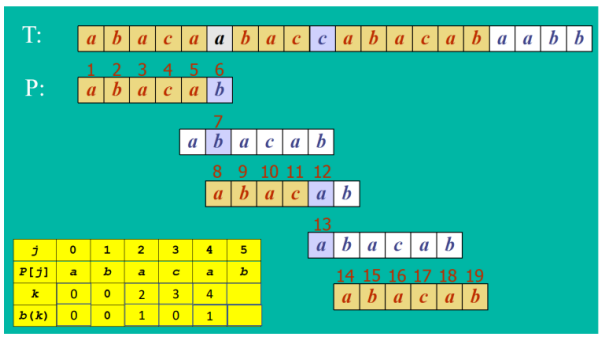
Implementasi alur pencarian ini dilakukan dengan menggunakan bahasa pemrograman Python, yang dikenal dengan kemampuannya dalam pengolahan teks dan kemudahan dalam penerapan algoritma-algoritma tersebut. Python menyediakan berbagai pustaka dan fungsi bawaan yang mendukung pengembangan fitur Auto Complete Search secara efisien.

Implementasi alur program dibuat dengan menggunakan bahasa pemrograman Python. Python menyediakan berbagai pustaka dan fungsi bawaan yang mendukung pengembangan program ini. Pembuatan kamus dilakukan dengan menggunakan program penerjemah menggunakan Bahasa Pemrograman Python yang diterjemahkan langsung dari Kamus Besar Bahasa Indonesia dalam format pdf.

II. TEORI DASAR

A. *Knuth-Morris-Pratt Algorithm*

Knuth-Morris-Pratt Algorithm adalah Algoritma pencocokan string yang dikembangkan oleh Donald E. Knuth, James H. Morris, dan juga Vaughan R. Pratt pada tahun 1967 yang dipublikasikan pada tahun 1977. Algoritma ini menggunakan metode perhitungan pergeseran bila terjadi ketidakcocokkan pada saat pola sejajar dengan teks. Pergeseran dibuat dengan menggunakan tabel Fungsi Pinggiran sehingga kita bisa mengetahui berapa banyak pergeseran terjadi. Algoritma ini menemukan semua kemunculan dari pattern dengan panjang n di dalam teks dengan panjang m dengan kompleksitas waktu O(m+n). Algoritme ini hanya membutuhkan O(n) ruang dari memory internal jika teks dibaca dari file eksternal.



Gambar 2.1 Pencocokan string pada KMP
(Sumber:[1]

Proses pencocokkan kata dengan *Knuth-Morris-Pratt Algorithm* melalui beberapa tahap berikut:

a. Menginisialisasi teks dan pola
b. Membandingkan indeks dari teks dan pola

c. Jika cocok, menggeser kedua indeks ke kanan
d. Jika tidak cocok, Menggunakan nilai pada tabel Fungsi Pinggiran untuk menentukan jumlah pergeseran pola dan mencocokkan tanpa pergeseran indeks
e. Melanjutkan hingga indeks mencapai terakhir atau pola ditemukan dalam teks.

Lalu, dalam pembuatan tabel Fungsi Pinggiran melalui beberapa tahapan berikut:

a. Menginisialisasi tabel dengan panjang yang sama dengan pola dan semua elemen diisi dengan nilai 0
b. Mulai dari posisi kedua, membandingkan karakter saat ini dengan karakter sebelumnya
c. Jika cocok, nilai saat ini ditambah 1
d. Jika tidak cocok menggunakan angka sebelumnya
e. Melanjutkan hingga akhir pola membentuk tabel

B. Damerau-Levenshtein Distance

Damerau-Levenshtein Distance adalah pengembangan dari algoritma *Levenshtein Distance*. *Levenshtein Distance* juga merupakan fungsi pengukuran jarak yang mengembalikan jarak minimumnya dimana proses deletion, insertion dan substitution dilakukan dalam proses pencocokan string. Algoritma ini memungkinkan untuk melakukan pencarian dan penyocokkan terhadap kata-kata yang mengalami kesalahan yang sedikit, seperti kesalahan satu huruf, maupun kesalahan kata dengan huruf yang terbalik.

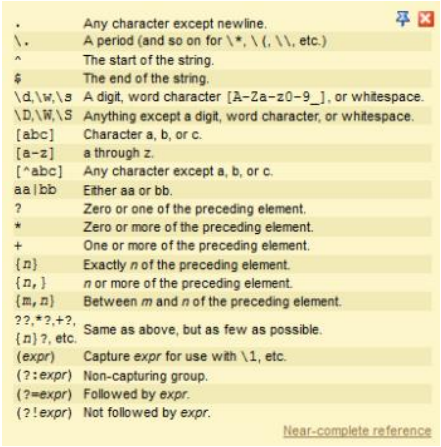
Proses pencocokkan kata dengan *Damerau-Levenshtein Distance* melalui beberapa tahap berikut:

a. Menginisialisasi matriks dengan ukuran panjang string 1 + 1 x panjang string 2 + 1 dengan mengisi baris pertama dan kolom pertama dengan nilai bertambah dari 0 hingga panjang string
b. Mengiterasi setiap karakter, jika sama biaya 0, jika tidak sama biaya 1.
c. Menghitung nilai sel saat ini sebagai minimum dari nilai sel atas + 1 (deletion), nilai sel kiri + 1 (insertion), nilai sel diagonal kiri atas + biaya (substitution)
d. Jarak antara kedua string adalah nilai di sel kanan bawah matriks

C. Regular Expression

Regular expression, atau yang sering disingkat sebagai regex, adalah konsep pencocokan pola dan manipulasi string. Pola ini terdiri dari rangkaian karakter yang bisa mencocokkan berbagai kombinasi karakter dalam teks. Konsep ini berguna untuk pencarian, penggantian, pengkombinasian, dan pemrosesan data dalam suatu teks.

Regex menggunakan berbagai simbol dan karakter dengan makna khusus. Misalnya, kelas karakter seperti [a-z] mencocokkan huruf kecil apapun, sementara * berarti nol atau lebih pengulangan dari elemen sebelumnya, dan + berarti satu atau lebih pengulangan. Simbol ^ dan $ digunakan untuk menandai awal dan akhir sebuah string. Selain itu, regex bisa menangkap bagian-bagian dari string menggunakan tanda kurung, yang membantu dalam penggantian dan penataan ulang data dalam teks.



`.`	Any character except newline.
`\.`	A period (and so on for `\*`, `\(`, `\\`, etc.)
`^`	The start of the string.
`$`	The end of the string.
`\d,\w,\s`	A digit, word character `[A-Za-z0-9_]`, or whitespace.
`\D,\W,\S`	Anything except a digit, word character, or whitespace.
`[abc]`	Character a, b, or c.
`[a-z]`	a through z.
`[^abc]`	Any character except a, b, or c.
`aa|bb`	Either aa or bb.
`?`	Zero or one of the preceding element.
`*`	Zero or more of the preceding element.
`+`	One or more of the preceding element.
`{n}`	Exactly *n* of the preceding element.
`{n,}`	*n* or more of the preceding element.
`{m,n}`	Between *m* and *n* of the preceding element.
`??,*?,+?, {n}?`, etc.	Same as above, but as few as possible.
`(expr)`	Capture *expr* for use with `\1`, etc.
`(?:expr)`	Non-capturing group.
`(?=expr)`	Followed by *expr*.
`(?!expr)`	Not followed by *expr*.

Near-complete reference

Gambar 2.2 Fungsi manipulasi Regular Expression (Sumber:[2])

Regex memiliki beberapa fungsi untuk memanipulasi teks, termasuk:

a. Pencocokan: Mengidentifikasi pola spesifik dalam teks menggunakan fungsi seperti match() atau findall().
b. Validasi: Memastikan bahwa teks sesuai dengan pola yang ditentukan menggunakan fungsi seperti search() atau match(). Validasi sangat penting dalam skenario seperti input formulir di mana kebenaran data, seperti nomor telepon atau alamat email, perlu dikonfirmasi terhadap format standar.
c. Penggantian: Mengganti pola yang cocok dengan teks baru menggunakan fungsi seperti sub() atau subn().
d. Pembagian: Membagi teks menjadi segmen-segmen berdasarkan pola tertentu menggunakan fungsi seperti split().

Pola regex diproses oleh mesin regex yang menggunakan automata deterministik (DFA) atau nondeterministik (NFA) untuk menemukan kecocokan secara efisien. Mesin ini memeriksa pola regex dan mencari di seluruh teks yang ditargetkan, menerapkan aturan pola satu per satu untuk menentukan kecocokan. Kecanggihan regex dalam mengolah teks yang kompleks, serta kemampuannya untuk digunakan dalam banyak bahasa pemrograman dan alat, membuatnya sangat berguna dalam berbagai tugas komputasi.

III. METODOLOGI

A. Pengambilan Data

Dalam pengambilan data yang digunakan sebagai kamus, digunakan Kamus Besar Berbahasa Indonesia (KBBI) tahun 2008 yang diunduh dari laman website *perpustakaan Universitas Muhammadiyah Semarang*. Data yang diambil dalam bentuk pdf dan diubah menjadi bentuk json agar lebih mudah dibaca.

Berikut label dan singkatan kata yang disediakan di dalam KBBI, antara lain:

a. Label Kelas Kata: n nomina, v verba, a adjektiva, adv adverbia, num numeralia, p partikel, pron pronomina
b. Singkatan Kata: dl dalam, dng dengan, dp daripada, dr dari, dsb dan sebagainya, kpd kepada, krn karena, msl misalnya, pd pada, sbg sebagai, spt seperti, thd terhadap, tt tentang, yg yang

1

A [1]**A**, **a** *n* huruf pertama abjad Indonesia
[2]**A** *n* Ampere; lambang satuan ukuran arus listrik
[3]**a** *n* are; nama satuan ukuran luas (= 100 m^2)
ab *ark n* tabung atau kotak candu terbuat dr tanah
aba *n* ayah; bapak; (kadang-kadang juga berarti) kakek
aba-aba *n* kata-kata perintah atau komando, spt dl baris-berbaris, senam, atau menyanyi bersama, msl *siap!, kiri! kanan!, satu! dua!, maju jalan!*
abad *n* **1** masa seratus tahun: *umurnya sudah setengah* --; **2** jangka waktu yg lamanya seratus tahun: -- *ke-20 dimulai dr tahun 1901 sampai tahun 2000*; **3** zaman (yg lamanya tidak tentu); **4** *kl* masa yg kekal, tidak berkesudahan;
-- **kekal** selama-lamanya; -- **keemasan** masa kegemilangan dan kejayaan yg dialami suatu bangsa atau negara dl sejarahnya; -- **pertengahan** zaman dl sejarah Eropa antara zaman purbakala dan zaman baru (tahun 476--1492);
berabad-abad *v* beberapa abad lamanya; beratus-ratus tahun: ~ *lamanya bangsa kita dijajah*;
seabad *n* satu abad; seratus tahun
abadi *a* kekal selama-lamanya; tetap ada sepanjang masa; tidak berkesudahan: *hanya Tuhan saja yg --; semua orang mendambakan kedamaian yg --*;
mengabadi *v* menjadi kekal (tidak berubah keadaannya, tetap selamanya);
mengabadikan *v* **1** mengekalkan: ~ *persahabatan antara kedua bangsa;* **2** menjadikan peringatan yg kekal; membuat gambar kenang-kenangan (dng dipotret, dilukis, dsb): *para wartawan foto dan televisi* ~ *upacara pembukaan Konferensi Asia-Afrika;*
pengabadian *n* proses, cara, perbuatan, mengabadikan;
keabadian *n* **1** kekekalan; **2** tempat yg abadi (alam baka): *kenanglah pahlawan kita yg telah bersemayam di* ~
abadiah *Ar n* kekekalan
abadiat → abadiah
[1]**abah** *n* arah; tuju: *tidak tentu --nya*;
mengabah *v* menuju: *berjalan* ~ *ke Timur*;
mengabahkan *v* mengarahkan; menujukan: *mereka* ~ *kapalnya ke pulau itu*
[2]**abah → aba**
abah-abah *n* **1** alat; perkakas; **2** tali-temali;
-- **kuda** pakaian kuda (tali kang, pelana, dsb); -- **perahu** tali-temali perahu; -- **tenun** perkakas tenun
abai *a* **1** tidak dihiraukan (tidak dilakukan dng sungguh-sungguh; tidak diindahkan dsb); **2** lalai: *anak-anak tidak boleh -- thd nasihat orang tua dan guru*;
mengabaikan *v* **1** memandang rendah (hina, mudah); *jangan* ~ *kemampuan musuh;* **2** tidak mengindahkan (perintah, nasihat): ~ *perintah agama;* **3** melalaikan (kewajiban, tugas, pekerjaan); **4** menyia-nyiakan; tidak menggunakan baik-baik: *ia tidak* ~ *peluang yg ada;* **5** tidak memedulikan (kritik, celaan): *mereka* ~ *kritik yg ditujukan kepadanya;* **6** membiarkan telantar (terbengkalai dsb): ~ *keluarganya;* **7** tidak memegang teguh (adat istiadat, aturan, janji): *negara itu* ~ *ketentuan-ketentuan yg telah disepakati dunia internasional;*
abaian *n Filol* penghilangan atau pengubahan bagian naskah yg tidak dipahami lagi oleh penyalin;

Gambar 3.1 Fungsi manipulasi Regular Expression (Sumber:[4])

Pengubahan data dilakukan dengan mengambil semua kata yang di-bold sebagai *'words'* dan penjelasan nya sebagai *'description'* yang dihasilkan dalam bentuk json.


```
[
    {
        "words": "A,",
        "description": [
            "huruf pertama abjad Indonesia ukuran arus listrik"
        ]
    },
    {
        "words": "ab",
        "description": [
            "tabung atau kotak candu terbuat dr tanah"
        ]
    },
    {
        "words": "aba",
        "description": [
            "ayah",
            "bapak",
            "kakek"
        ]
    },
    {
        "words": "aba-aba",
        "description": [
            "kata-kata perintah atau komando, spt dl baris-berbaris, senam, atau menyanyi bersama
        ]
    },
    {
        "words": "abad",
        "description": [
            "masa seratus tahun",
            "jangka waktu yg lamanya seratus tahun",
            "zaman",
            "masa yg kekal, tidak berkesudahan"
        ]
    },
    {
        "words": "berabad-abad",
        "description": [
            "beberapa abad lamanya",
            "beratus-ratus tahun"
        ]
    },
    {
        "words": "seabad",
        "description": [
            "satu abad",
            "seratus tahun"
        ]
    },
    {
        "words": "abadi",
        "description": [
            "kekal selama-lamanya",
            "tetap ada sepanjang masa",
```


Gambar 3.2 Bentuk data yang dibuat dari KBBI (Sumber: Dokumen Pribadi)

Data yang telah diolah berjumlah sebanyak 33093 kata beserta deskripsi dari masing-masing kata tersebut. Dari data tersebut deskripsi melambangkan arti lain atau maksud dari kata tersebut. Semakin banyak deskripsi kata tersebut maka penggunaan nya dalam kalimat semakin sering pula

B. Penggambaran Permasalahan

Dalam penyelesaian masalah ini, digunakan beberapa algoritma pencocokan string, yaitu *Knuth-Morris-Pratt* (KMP), *Damerau-Levenshtein Distance*, dan *Regular Expression*. KMP digunakan untuk menyocokkan pola dengan menghitung kecocokkan substring / pola dengan string / teks. *Damerau-Levenshtein Distance* digunakan untuk menangani kesalahan ketik yang umum seperti penghapusan, penyisipan, dan transposisi antara 2 karakter. Sedangkan, *Regular Expression* digunakan untuk menyederhanakan teks dari huruf besar menjadi huruf kecil, penyederhanaan spasi dan '_' pada pola.

Awalnya, string input / pola disederhanakan menjadi bentuk normal dengan mengolah huruf besar, spasi, serta '_' menggunakan *Regular Expression*. Lalu, kamus diiterasi dan dicocokkan dengan menggunakan KMP yang dibandingkan dengan teks yang telah dinormalisasi. Lalu, jika ditemukan akan dimasukkan ke daftar rekomendasi. Batas panjang kata yang dicocokkan menggunakan kmp adalah kata yang memiliki panjang 3 lebih dari panjang kata awal / pola.

Lalu, untuk mengatasi masalah kesalahan dapat dengan menggunakan *Damerau-Levenshtein Distance* sehingga jika

ada kata yang memiliki panjang sama yang memiliki perbedaan yang tidak terlalu signifikan, maka akan dimasukkan ke dalam daftar rekomendasi.

C. Implementasi Program Utama

Dari permasalahan tersebut, berikut adalah program yang dibuat menjadi beberapa fungsi, yaitu:

- Fungsi kmp_search(text,pattern): fungsi ini membangun terlebih dahulu fungsi pinggiran lalu mengiterasi dan menyocokkan antara teks dengan pola yang mengembalikan indeks awal yang cocok

```python
def kmp_search(text, pattern):
    def build_lps(pattern):
        lps = [0] * len(pattern)
        length = 0
        i = 1
        while i < len(pattern):
            if pattern[i] == pattern[length]:
                length += 1
                lps[i] = length
                i += 1
            else:
                if length != 0:
                    length = lps[length - 1]
                else:
                    lps[i] = 0
                    i += 1
        return lps

    lps = build_lps(pattern)
    i = j = 0
    while i < len(text):
        if pattern[j] == text[i]:
            i += 1
            j += 1
        if j == len(pattern):
            return i - j
        elif i < len(text) and pattern[j] != text[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1
    return -1
```

Gambar 3.3 Implementasi Pencarian KMP dalam bahasa Python
(Sumber: Dokumen Pribadi)

- Fungsi demerau_levenshtein_distance(s1, s2): fungsi ini mengimplementasikan levenshtein dengan tambahan pencocokkan dari Demerau.. Fungsi ini menyimpan nilai-nilai matriks lalu diloop sebanyak 2 kali dan mengembalikan hasil matriks yang berada di kanan bawah

```python
def damerau_levenshtein_distance(s1, s2):
    d = {}
    lenstr1 = len(s1)
    lenstr2 = len(s2)
    for i in range(-1, lenstr1 + 1):
        d[(i, -1)] = i + 1
    for j in range(-1, lenstr2 + 1):
        d[(-1, j)] = j + 1

    for i in range(lenstr1):
        for j in range(lenstr2):
            if s1[i] == s2[j]:
                cost = 0
            else:
                cost = 1
            d[(i, j)] = min(
                d[(i - 1, j)] + 1,
                d[(i, j - 1)] + 1,
                d[(i - 1, j - 1)] + cost,
            )
            if i and j and s1[i] == s2[j - 1] and s1[i - 1] == s2[j]:
                d[(i, j)] = min(d[(i, j)], d[i - 2, j - 2] + cost)

    return d[lenstr1 - 1, lenstr2 - 1]
```

Gambar 3.4 Implementasi Pencarian Damerau-Levenshtein Distance dalam bahasa Python
(Sumber: Dokumen Pribadi)

- Fungsi auto_complete(dictionary, input): fungsi ini mencari semua kata yang mungkin dengan maksimal sebanyak 3 kata lebih untuk awalan dan 2 kata lebih untuk akhiran. Lalu, untuk kesalahan penamaan akan digunakan levenshtein sehingga semua kata yang memiliki panjang yang sama dan terdapat sedikit perubahan akan masuk ke dalam daftar rekomendasi. Lalu, dimasukkan juga kata yang memiliki lebih dari 5 deskripsi jika kata tersebut merupakan substring dari kata yang dicari. Pada akhir, diurutkan berdasarkan prioritas untuk awalan lalu akhiran, dan prioritas terakhir yaitu kesalahan penamaan.

```python
def auto_complete(dictionary, input):
    normalized_input = normalize_word(input)
    suggestions = []

    input_length = len(normalized_input)
    max_input_offset = input_length + 3
    max_suffix_offset = input_length + 2

    for entry in dictionary:
        word = entry['normalized']
        word_length = len(word)
        description_count = len(entry['description'])

        if word_length >= 2 and word_length <= max_input_offset:
            if word.startswith(normalized_input) and word_length <= max_input_offset:
                suggestions.append((entry['words'], entry['description'], 0))
            elif kmp_search(word, normalized_input) != -1 and word_length <= max_suffix_offset:
                suggestions.append((entry['words'], entry['description'], 1))
        if description_count > 5 and kmp_search(word, normalized_input) != -1:
            suggestions.append((entry['words'], entry['description'], 2))

        if len(word) >= 2 and kmp_search(normalized_input, word) != -1 and description_count > 5:
            suggestions.append((entry['words'], entry['description'], 3))

        if word_length == input_length and damerau_levenshtein_distance(word, normalized_input) == 1:
            suggestions.append((entry['words'], entry['description'], 4))

    suggestions.sort(key=lambda x: (x[2], len(x[0]), x[0]))
    return suggestions
```

Gambar 3.5 Implementasi Program Utama Pencarian Auto Complete dalam bahasa Python
(Sumber: Dokumen Pribadi)

IV. ANALISIS DAN PENGUJIAN

A. Pengujian

```
PS D:\Coding\MAKALAH-STIMA> python main.py
Masukan kata: ibunad
['ibu', 'ibunda']
PS D:\Coding\MAKALAH-STIMA>
```

Gambar 4.1 Pengujian 1 Kata *'ibunad'*
(Sumber: Dokumen Pribadi)

```
PS D:\Coding\MAKALAH-STIMA> python main.py
Masukan kata: ayah
['ayah', 'ayahan', 'dayah', 'daayah', 'kuayah', 'abah', 'alah', 'amah', 'arah', 'asah', 'awah', 'ayak', 'ayal', 'ayam',
'ayan', 'ayat', 'ayuh', 'syah']
PS D:\Coding\MAKALAH-STIMA>
```

Gambar 4.2 Pengujian 2 Kata *'ayah'*
(Sumber: Dokumen Pribadi)

```
PS D:\Coding\MAKALAH-STIMA> python main.py
Masukan kata: kemana
['kemanakan', 'mana', 'kelana', 'kemala', 'kemang']
PS D:\Coding\MAKALAH-STIMA>
```

Gambar 4.3 Pengujian 3 Kata *'kemana'*
(Sumber: Dokumen Pribadi)

```
PS D:\Coding\MAKALAH-STIMA> python main.py
Masukan kata: makalahstima2024
['mak', 'akal', 'kalah']
PS D:\Coding\MAKALAH-STIMA>
```

Gambar 4.4 Pengujian 4 Kata *'makalahstima2024'*
(Sumber: Dokumen Pribadi)

```
PS D:\Coding\MAKALAH-STIMA> python main.py
Masukan kata: kemdnasdla
[]
PS D:\Coding\MAKALAH-STIMA>
```

Gambar 4.5 Pengujian 5 Kata *'kemdnasdla'*
(Sumber: Dokumen Pribadi)

B. Analisis Pembahasan

Pada Gambar **4.1** ditemukan kata 'ibu' dan 'ibunda' sebab ibunad merupakan kesalahan dalam penulisan sehingga pendekatan dengan Damerau-Levenshtein akan digunakan dalam penyocokkan string. Substring juga digunakan karena ibu memiliki lebih dari 5 deskripsi yang berarti kata tersebut umum digunakan

```
{
    "words": "ibu",
    "description": [
        "sebutan untuk perempuan yg telah melahirkan kita",
        "mak",
        "wanita yg sudah bersuami",
        "panggilan yg takzim kpd I ibuk x idealisme wanita",
        "bagian yg pokok",
        "ayah dan ibu",
        "jempol kaki",
        "beribu 1 mempunyai ibu",
        "menganggap  ibu"
    ]
},
```

Gambar 4.1 Data dari kata *'ibu'*
(Sumber: Dokumen Pribadi)

Pada Gambar **4.2** ditemukan kata ['ayah', 'ayahan', 'dayah', 'daayah', 'kuayah', 'abah', 'alah', 'amah', 'arah', 'asah', 'awah', 'ayak', 'ayal', 'ayam', 'ayan', 'ayat', 'ayuh', 'syah']. Semua ini menyangkut awalan yaitu 'ayahan', daftar kata yang memiliki akhiran ayah seperti 'dayah', 'daayah' dan 'kuayah'. Lalu semua kata yang memiliki kesalahan 1 kata dengan ayah.

Pada Gambar **4.3** ditemukan banyak kata seperti yang ada pada gambar. Semua kata tersebut memilki awalan yang cocok seperti 'kemanakan', substring 'mana' dan kesalahan kata yang merupakan sisanya

Pada Gambar **4.4** ditemukan kata 'mak', 'akal', dan 'akal' karena tidak ditemukan kata yang cocok, namun memiliki substring yang bagus karena 'mak', 'akal', dan 'kalah' memiliki deskripsi yang lebih dari 5 yang berarti kara tersebut umum digunakan

Pada Gambar **4.5** tidak ditemukan kata-kata apapun karena kata tersebut juga kacau sehingga tidak bisa ditemukan kata-kata yang cocok baik awalan, akhiran, substring maupun pengoreksian

V. KESIMPULAN

Dalam penerapan algoritma Auto Complete Search, penggunaan berbagai teknik pencocokan string terbukti sangat efektif untuk menangani berbagai jenis masukan pengguna. Pada **Gambar 4.1**, ditemukan bahwa pendekatan dengan Damerau-Levenshtein dapat mendeteksi kesalahan penulisan seperti 'ibunad' dan mengoreksinya menjadi kata yang benar seperti 'ibunda'. Hal ini menunjukkan kemampuan algoritma Damerau-Levenshtein yang tepat untuk menangani kesalahan ketik dengan efisien. Selanjutnya, penggunaan substring dalam algoritma juga efektif seperti pada **Gambar 4.4**, terutama untuk kata-kata yang memiliki banyak deskripsi seperti 'ibu', yang menunjukkan bahwa kata tersebut sering digunakan dan memiliki banyak arti.

Pada **Gambar 4.2** dan **Gambar 4.3**, pendekatan ini memungkinkan algoritma untuk menemukan kata-kata dengan awalan yang sama, akhiran yang mirip, dan kata-kata yang memiliki kesalahan penulisan satu karakter dari kata 'ayah'. Ini memperlihatkan bahwa algoritma mampu memberikan saran yang relevan berdasarkan awalan, akhiran, dan kesalahan ketik. Lalu juga ditemukan banyak kata yang cocok dengan awalan tertentu, substring, dan kesalahan penulisan satu karakter, seperti 'kemanakan' yang cocok dengan awalan 'ke', substring 'mana', dan berbagai kesalahan ketik lainnya. Hal ini menunjukkan bahwa kombinasi berbagai metode pencocokan string dalam algoritma mampu menangani berbagai variasi masukan pengguna dengan baik.

Secara keseluruhan, implementasi algoritma Auto Complete Search yang menggunakan *Knuth-Morris-Pratt*, *Damerau-Levenshtein Distance*, dan *Regular Expression* memberikan hasil yang akurat dan relevan. Algoritma ini mampu meningkatkan efisiensi dan akurasi pencarian, serta memberikan pengalaman pengguna yang lebih baik dalam menemukan kata atau frasa yang tepat, meskipun terdapat kesalahan penulisan atau variasi masukan.

VI. UCAPAN TERIMA KASIH

Penulis menyampaikan ucapan terima kasih kepada:

1. Tuhan Yang Maha Esa atas berkat-Nya sehingga penulis dapat menyelesaikan makalah ini dengan baik,
2. Bapak Dr. Ir. Rinaldi Munir, M.T., selaku pembimbing kelas IF2211 Strategi Algoritma kelas 1.

Penulis juga menyampaikan terima kasih banyak kepada semua orang yang belum disebutkan yang telah memberikan dukungan kepada saya. Tanpa bantuan dari Anda, makalah ini tidak dapat terselesaikan dengan baik. Akhir kata, penulis mengharapkan makalah ini dapat membawa berkah dan manfaat kepada pihak yang membacanya..

VII. LAMPIRAN

Source code:
https://drive.google.com/drive/u/1/folders/1ugwRD0PS02bWNde4tLfXkS6kUYSyDl2e

VIII. REFERENSI

[1] Munir, Rinaldi (2024). Pencocokan String (String/Pattern Matching) https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf (diakses tanggal 11 Juni 2024)

[2] Munir, Rinaldi (2024). String Matching dengan Regular Expression https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf (diakses tanggal 11 Juni 2024)

[3] Erisanda, Rika Dwi (2008). Analisis dan Implementasi Algoritma Damerau Levenshtein Distance untuk Content Based Music Retrieval https://repository.telkomuniversity.ac.id/pustaka/94690/analisis-dan-implementasi-algoritma-damerau-levenshtein-distance-untuk-content-based-music-retrieval.html (diakses tanggal 11 Juni 2024)

[4] PUSAT BAHASA DEPARTEMEN PENDIDIKAN NASIONAL JAKARTA, (2008). KAMUS BAHASA INDONESIA https://perpus.unimus.ac.id/wp-content/uploads/2012/05/Kamus-Besar-Bahasa-Indonesia.pdf (diakses tanggal 11 Juni 2024)

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 Juni 2024

Farel Winalda 13522047
Nama dan NIM