Algoritma Pencocokan String pada Rekomendasi Query dalam Mesin Pencari

Arjuna Marcelino - 13519021
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519021@std.stei.itb.ac.id

Abstrak*—Mesin pencari merupakan aplikasi yang sering digunakan untuk membantu mendapatkan informasi dari internet. Mesin pencari memberi kemudahan pada pengguna dalam menjalani kehidupan sehari-hari. Salah satu fitur yang didapatkan oleh pengguna dalam menggunakan mesin pencari adalah rekomendasi *query* atau *autocomplete suggestion* berdasarkan query masukan yang diketikkan pengguna dalam kolom pencairan. Rekomendasi *query* ini memberi kemudahan dan keefisienan dalam melakukan pencarian. Pada makalah ini, penulis akan menjelaskan bagaimana algoritma pencocokan string dapat dimanfaatkan dalam mencari rekomendasi *query* berdasarkan *query* masukkan pengguna di dalam mesin pencari. Metode yang digunakan dalam penulisan makalah ini adalah dengan studi literatur dan uji coba implementasi dengan program yang telah dibuat oleh penulis. Penulis berhasil menunjukkan dan mengimplementasikan penerapan pencocokan string untuk mencari rekomendasi *query* dengan algoritma Knuth-Morris-Pratt (KMP).

Kata-kata kunci—pencocokan string, rekomendasi query, mesin pencari, algoritma Knuth-Morris-Pratt (KMP).

I. PENDAHULUAN

Perkembangan teknologi yang berjalan dengan sangat pesat, khususnya beberapa tahun belakangan ini, memberikan banyak kemudahan bagi manusia dalam menjalani kehidupan sehari-harinya. Hampir semua kegiatan yang dilakukan oleh manusia tidak dapat dilepaskan lagi dengan teknologi. Mulai dari membuka mata di pagi hari hingga menutup mata di malam hari, teknologi menjadi sesuatu yang sangat penting untuk digunakannya. Salah satu bentuk kemudahan yang dapat dirasakan oleh manusia adalah dalam memperoleh informasi melalui internet. Internet memberikan fasilitas yang luar biasa yang dapat menggantikan banyak barang lainnya dalam memperoleh informasi seiringnya terdapat berbagai kanal informasi elektronik yang dapat digantikan oleh internet. Ada kanal media berita elektronik yang mulai menggantikan koran, buku elektronik yang menggantikan buku fisik, dan platform media lain seperti youtube yang mulai menggantikan CD, kaset, atau bahkan televisi.

Kemudahan dalam mencari informasi melalui internet dapat digunakan hanya dengan memasukkan satu buah frasa atau kalimat pada mesin pencari. Mesin pencari adalah program yang mencari dan mengidentifikasi informasi di dalam basis data yang sesuai dengan kata kunci yang dimasukkan oleh pengguna. Setelah menerima masukan *query*, mesin pencari akan menampilkan hasilnya pada halaman hasil mesin pencari dimulai dari informasi yang paling relevan dengan masukan anda. Ada berbagai mesin pencari yang sering digunakan, yaitu Google, Bing, Baidu, Yahoo!, dan Yandex. Nama pertama adalah mesin pencari yang paling sering dan awam digunakan di masyarakat umum. *Query* masukan pengguna akan dicocokkan pada basis data yang ada dan diambil yang memiliki kemiripan yang tinggi untuk ditampilkan pada hasil pencarian.

Pada saat memasukkan kata atau frasa atau kalimat pada kolom pencarian baik pada mesin pencari ataupun tempat lain, kita akan diberi beberapa rekomendasi kata selanjutnya dalam pencarian kita. Hal ini dapat kita lihat seperti yang ditunjukkan pada gambar di bawah ini.

Gambar 1. Contoh rekomendasi query pada mesin pencari
(Sumber : Dokumentasi Pribadi)

Pada mesin pencari tersebut, pengguna memasukkan kata "kapal" dan mesin pencari memberi beberapa rekomendasi kata selanjutnya yang mungkin menjadi pilihan pengguna yang akan mengarahkan pengguna mencari informasi yang lebih spesifik dari apa yang diinginkan. Pada makalah ini, penulis akan mencoba menjelaskan bagaimana penerapan algoritma *string matching* (pencocokan string) dalam memberi rekomendasi *query* kepada pengguna dalam mesin

pencari. Penulis akan menjelaskan mengenai mesin pencari, algoritma *string matching*, rekomendasi *query*, yang akan dibantu juga dalam bentuk implementasi dan pengujian sederhana yang akan dilakukan dan dibuktikan.

II. LANDASAN TEORI

A. Mesin Pencari

Mesin pencari adalah program komputer yang dirancang untuk melakukan pencarian dan pengidentifikasian informasi di dalam berkas-berkas yang tersimpan di dalam basis data yang sesuai dengan kata kunci yang dimasukkan oleh pengguna. Hasil pencarian umumnya ditampilkan dalam bentuk daftar yang sering kali diurutkan menurut tingkat akurasi ataupun rasio pengunjung atas suatu berkas. Fungsi mesin pencari adalah menyediakan informasi berdasarkan kata kunci yang dimasukkan oleh pengguna. Fungsi ini sesuai dengan cara kerjanya yaitu mendaftar/mengindeks/mendata situs yang ada di internet.

Mesin pencari bekerja dalam tiga tahapan utama, yaitu *crawling*, *indexing*, dan *ranking.*

1) Crawling

Pada tahap ini, mesin pencari memroses pengumpulan data untuk disimpan di dalam basis data. Robot dari mesin pencari akan menjelajahi situs web di seluruh dunia dan mencari konten baru untuk dikumpulkan dalam basis data. Robot juga akan mencari konten yang diperbarui dan menyimpan versi terbaru dari konten tersebut. Mesin pencari tidak akan menyimpan versi asli dari konten tersebut. Robot hanya akan mengikuti tautan yang ada di situs web.

2) Indexing

Proses ini akan mengelompokkan informasi sesuai dengan kategori yang sama dengan tujuan mempermudah penampilan hasil ketika diminta oleh pengguna. Kategori yang digunakan dalam proses *indexing* sangat beragam, seperti bahasa yang digunakan, lokasi pencarian, dan lainnya.

3) Ranking

Tahap ini adalah tahap untuk menampilkan hasil pencarian sesuai dengan urutan. Mesin pencari akan menampilkan banyak sekali hasil pencarian yang terbagi dalam banyak halaman hasil pencarian. Mesin pencari berusaha menampilkan hasil terbaik yang paling relevan dengan masukan pengguna di halaman pertama hasil pencarian.

B. Algoritma Pencocokan String (String Matching)

Algoritma pencocokan string (*string matching*) adalah algoritma yang memproses pencarian semua kemunculan string pendek P[0..n-1] yang biasa disebut sebagai pattern di string yang lebih panjang T[0..m-1] yang biasa disebut sebagai teks. Contoh implementasi pencocokan string adalah pencarian di dalam *editor text*, *web search engine*, analisis citra, dan bioinformatika, seperti pencocokan string asam amino pada rantai DNA.

Proses pencocokan string dapat dilakukan dengan beberapa algoritma berikut ini.

- Algoritma *Brute Force*
- Algoritma *Knuth Morris Pratt (KMP)*
- Algoritma *Boyer-Moore (BM)*
- *Regular Expression*

Pada permasalahan rekomendasi *query* pada mesin pencari ini, kita akan menggunakan algoritma *Knuth Morris Pratt (KMP)*. Hal ini dikarenakan algoritma ini merupakan salah satu algoritma yang efisien.

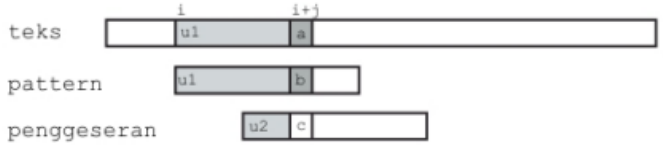
C. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977 (Manikandan dan Ramyachitra,2018).

Jika kita melihat algoritma KMP lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian.

Perhitungan pergeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokkan pada saat pattern sejajar dengan teks[i..i+n-1], dapat menganggap ketidakcocokan pertama terjadi di antara teks[i+j] dan pattern[j], dengan 0<j.

Dengan kata lain, pencocokkan string akan berjalan secara efisien bila kita mempunyai tabel yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokkan di karakter ke- j dari pattern. Tabel itu harus memuat next[j] yang merupakan posisi karakter pattern[j] setelah digeser, sehingga kita bisa menggeser pattern sebesar j-next[j] relatif terhadap teks. (Sunarto,2018;Jimale et al,2018).



Gambar 2. Penyejajaran *u2* dengan akhiran dari *u1*
(Sumber : Dokumentasi Pribadi)

Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat pencocokan string:

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.

2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:

1. Karakter di-pattern dan di teks yang dibandingkan tidak cocok (mismatch).

2. Semua karakter di-pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.

3. Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

```
procedure borderFunction (
    input P : array[0..n-1] of char,
    input n : integer,
    input/output b : array[0..n] of integer
)

Kamus
    I,j : integer

Algoritma
    I ← 0
    j ← b[0] = -1
    while (i<n) do
        while (j > -1 and not (P[i] = P[j])) do
            j ← b[j]
        I ← i+1
        j ← j+1
        if (P[i] = P[j]) then
            b[i] ← b[j]
        else
            b[i] = j
        endif
    endwhile
```

Gambar 3. *Pseudocode* penghitungan nilai *border function*
(Sumber : Dokumentasi Pribadi)

```
procedure KMP (
    input m,n : integer
    input P : array[0..n-1] of char,
    input T : array[0..m-1] of char,
    output idx : array[0..m-1] of integer
)

Kamus
    I,j,k,next : integer
    bdrFunc : array [0..n] of integer

Algoritma
    borderFunction(P, n, bdrFunc)
    I ← 0
    j ← 0
    k ← 0
    while (i≤ m-n) do
        while (j < n and (T[i+j] = P[j])) do
            j ← j+1
        endwhile
        if (j ≥ n) then
            idx[k] ← i
            k ← k + 1
        endif
        next ← j - bdrFunc[j]
        if(bdrFunc[j]>0)then
            j ← bdrFunc[j]
        else
            j ← 0
        endif
        I ← i+next
    endwhile
```

Gambar 4. *Pseudocode* Algoritma *Knuth-Morris-Pratt*
(Sumber : Dokumentasi Pribadi)

Kompleksitas waktu KMP :

- Menghitung fungsi pinggiran : O(m),
- Pencarian string : O(n)
- Kompleksitas waktu algoritma KMP adalah O(m+n).

Kompleksitas ini sangat cepat dibandingkan brute force.

Algoritma KMP tidak perlu bergerak mundur dalam masukan teks, T. Hal ini membuat algoritma KMP baik untuk pemrosesan file sangat besar yang dibaca dari perangkat eksternal atau melalui sebuah aliran jaringan. Algoritma KMP tidak berfungsi sebaik ukuran alfabet meningkat karena pada kasus ini akan lebih banyak kemungkinan ketidakcocokan (lebih banyak kemungkinan ketidakcocokan) dan ketidaksesuaian cenderung terjadi di awal pola, tetapi KMP lebih cepat ketika ketidakcocokan terjadi nanti.

D. Rekomendasi Query

Pada saat kita melakukan pencarian di mesin pencari, kita perlu memasukkan *query* pada kolom pencarian. Pada hampir semua mesin pencari, saat kita memasukkan *query*, akan muncul beberapa *query* lanjutan yang disarankan yang mungkin menjadi pilihan pengguna. Rekomendasi query ini dapat berupa kata/frasa kata setelah kata yang kita masukkan ataupun sebelum kata yang kita masukkan.

Gambar 5. Contoh rekomendasi query pada mesin pencari
(Sumber : Dokumentasi Pribadi)

Pada gambar 5, kita bisa melihat mesin pencari memberi beberapa rekomendasi *query* yang lebih lengkap di bawah kolom pencarian. Query yang lebih lengkap ini biasanya berupa satu sampai dua kata setelah maupun sebelum kata yang sudah kita masukkan dalam kolom pencarian. Seringkali rekomendasi *query* ini membantu pengguna dalam melakukan pencarian. Pengguna jadi dapat memilih pencarian yang lebih spesifik yang sesuai dia inginkan ataupun pengguna jadi dapat mengetahui sesuatu yang ingin mau dicari hanya dengan memasukkan beberapa kata kunci.

Tujuan awal dari rekomendasi/saran query ini adalah untuk membantu meningkatkan kecepatan mengetik dan

membantu mengurangi jumlah penekanan tombol keyboard untuk menyelesaikan sebuah kata atau kalimat.

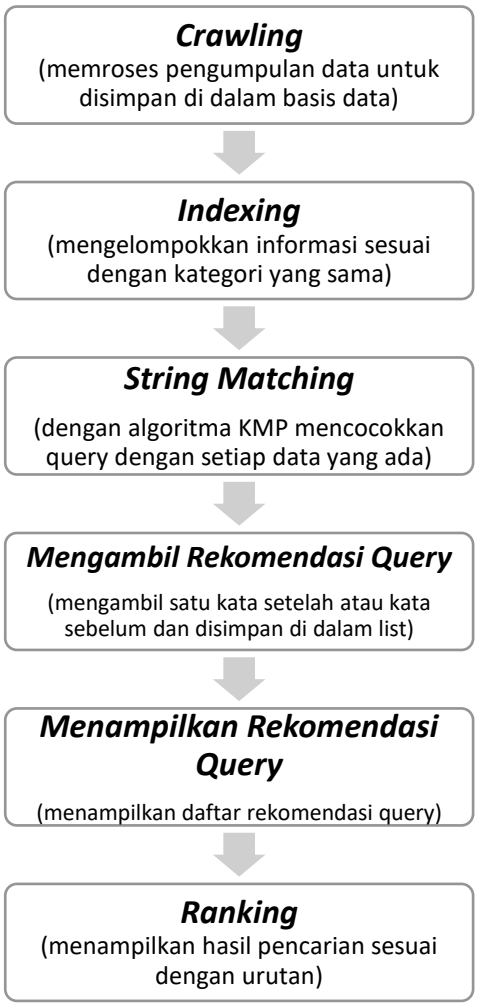
Rekomendasi *query* ini ditentukan oleh banyak faktor, seperti popularitas dari kata kunci yang diketik pada kolom pencarian, ataupun jumlah klik pada pencarian serupa yang telah dilakukan sebelumnya. Namun, secara sederhana, kita dapat mengambil beberapa *query* yang muncul paling sering dari seluruh data yang kita miliki.

Ada beberapa kemungkinan apabila rekomendasi *query* tidak muncul ketika pengguna mengetikkan kata kunci pada kolom pencarian, seperti :

1. Kata kunci yang dicari tidak popular.
2. Kata kunci yang dicari masih baru atau tidak ada data yang mengacu pada kata kunci yang dimasukkan.
3. Kata kunci yang dicari dianggap sebagai pelanggaran kebijakan.

III. Analisis dan Pembahasan

Proses untuk mencari rekomendasi *query* dapat dilihat dari bagan di bawah ini.



Gambar 6. Bagan alir proses pencarian rekomendasi *query*
(Sumber : Dokumentasi Pribadi)

A. Mencari Rekomendasi Query

Pengguna memasukkan *query* pada kolom pencarian. Setiap masukan pengguna dan perubahan pada masukan pengguna akan dilakukan pencocokan string dengan KMP dengan setiap judul yang telah kita punya dan simpan di dalam basis data. Untuk setiap *query* yang telah dicocokkan dan terdapat di dalam judul yang ada pada basis data, maka kata setelah atau sebelumnya sebanyak satu sampai dua kata akan disimpan ke dalam list. Sebelum dimasukkan ke dalam list, akan dicek terlebih dahulu, apakah rekomendasi *query* sudah ada tersimpan ke dalam list atau belum. Jika sudah ada, maka rekomendasi *query* akan dilewatkan. Jika belum ada tersimpan di list, maka rekomendasi *query* akan disimpan ke dalam list.

Rekomendasi *query* akan diurutkan berdasarkan rekomendasi *query* paling banyak kemunculan pada data yang kita miliki sampai yang paling sedikit. Rekomendasi *query* hanya akan diambil sebanyak 10 data.

B. Pencocokan String (String Matching)

Suatu *query* pencarian : stra

Daftar judul yang kita miliki :

[“strategi algoritma brute force”,
“strategi algoritma”,
“strategi perang”,
“aku suka belajar stima”,
“strategi algoritma kmp”,
“strategi bermain bola”,
“pencocokan string pada search engine”,
“du stra sutra”]

Akan dilakukan pencocokan *query* dengan salah satu daftar judul yang kita miliki, dalam hal ini judul yang terakhir, yaitu “du stra sutra”.

Langkah pertama :

Menghitung *border function*

P = stra

J = 0123

k = j-1

j	0	1	2	3
P[j]	s	t	r	a
k	-	0	1	2
b[k]	-	0	0	0

Langkah kedua :

Pencocokan karakter awal pada *query* T[0] dengan karakter awal pada judul terakhir P[0].

```
Du stra sutra
stra
```

Karakter tidak ada yang sama, dilakukan pergeseran pola satu karakter ke kanan judul.

```
Du stra sutra
 stra
```

Langkah ketiga :

Pencocokan karakter awal pada *query* T[0] dengan karakter awal pada judul terakhir P[1].

```
Du stra sutra
 stra
```

Karakter tidak ada yang sama, dilakukan pergeseran pola satu karakter ke kanan judul.

Langkah keempat :

Pencocokan karakter dilakukan berulang seperti pada Langkah kedua dan Langkah ketiga sampai menemukan string yang bersesuaian.

```
Du stra sutra
    stra
```

Didapat sebuah string yang sama di judul sesuai dengan *query*, maka fungsi KMP akan mengembalikan indeks awal letak string *query* yang bersesuaian dengan judul. Dalam hal ini, fungsi akan mengembalikan nilai 3.

Pencocokan string seperti inilah yang akan dilakukan pada *query* dengan semua judul yang sudah disimpan di basis data.

C. Hasil Implementasi Kode

Setelah melakukan pencocokan string *query* dengan judul yang telah disimpan. Apabila fungsi KMP mengembalikan sebuah indeks posisi, maka program akan mengambil kata selanjutnya dari judul yang kita punya dengan algoritma sebagai berikut.

```
Procedure recom (
    input data : array [0..p] of string,
    input query : string,
    output rec : array[0..q] of string,
)

Kamus
    I : string
    idx : array [0..n] of integer
    n,m,j,curr : integer
    found : boolean

Algoritma
    for I in (data) :
        n ← len(query)
        m ← len(i)
        KMP(m,n,query,I,idx)
        str ← query

        if (len(idx)!= 0) then
            curr ← idx[0] + n
            spasi ← 0

            while (curr<m and spasi<=1) do
                str ← str + i[curr]
                if (i[curr]=" "):
                    spasi ← spasi + 1
                endif
                curr ← curr + 1
            endwhile

            j ← 0
            found ← False

            if (len(rec)>0) then
                while(j<=len(rec) and ! (found)):
                    if(rec[j-1] = str) then
                        found ← True
                    endif
                    j ← j+1
             endwhile
        endif

    if not (found) :
        rec.append(str)
    endif
endfor
```

Gambar 7. *Pseudocode* pencarian rekomendasi query
(Sumber : Dokumentasi Pribadi)

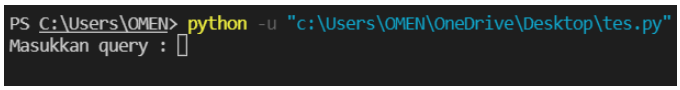
Rekomendasi *query* sudah kita dapatkan dalam variabel rec pada procedure recom. Apabila rec memiliki isi, maka akan dipilih sepuluh indeks teratas untuk ditampilkan. Apabila tidak ditemukan rekomendasi, makan tidak akan menampilkan apapun. Apabila memiliki rekomendasi kurang dari 10, akan ditampilkan semua rekomendasi yang mungkin.

Contoh hasil implementasi yang telah dibuat.:

Berikut adalah judul data yang sudah disimpan di dalam database :

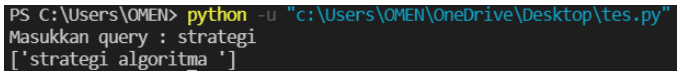
```
data = ["strategi algoritma brute force",
"aku suka belajar",
"aku suka stima",
"belajar stima menyenangkan",
"strategi algoritma greddy",
"strategi algoritma kmp",
"strategi algoritma binary",
"strategi algoritma forex"]
```

Ketika kita menjalankan program yang telah dibuat, maka program akan meminta *query* yang ingin kita masukkan.



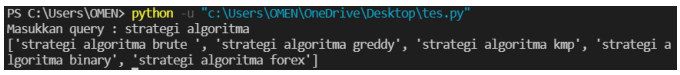
Gambar 8. Tampilan awal kode yang dijalankan
(Sumber : Dokumentasi Pribadi)

Setelah itu, kita dapat memasukkan *query* yang kita inginkan seperti berikut ini.



Gambar 9. Hasil pengujian 1
(Sumber : Dokumentasi Pribadi)

Setelah memasukkan *query*, kita mendapatkan rekomendasi *query*. Dalam permasalahan ini, kita mendapatkan satu buah rekomendasi *query*, yaitu "strategi algoritma".



Gambar 10. Hasil pengujian 2
(Sumber : Dokumentasi Pribadi)

Apabila kita memasukkan *query* lain, dalam hal ini "strategi algoritma", kita mendapatkan rekomendasi *query* dalam bentuk list seperti yang ditunjukkan pada gambar di atas (gambar 10).

Gambar 11. Hasil pengujian 3
(Sumber : Dokumentasi Pribadi)

Saat kita memasukkan sebuah *query* yang tidak cocok dengan semua judul yang kita miliki di basis data, program akan menampilkan list kosong, yang artinya program tidak dapat memberi rekomendasi *query* pada pengguna.

Dari hasil pengujian implementasi yang telah dibuat, didapatkan hasil rekomendasi *query* yang diberikan pengguna sebagai respons yang bersesuaian dengan masukan *query* pengguna. Hasil implementasi yang telah dilakukan masih dalam bentuk yang sangat sederhana.

IV. Simpulan

Algoritma pencocokan string (*string matching*) dapat digunakan dalam kehidupan sehari-hari dan memberi kemudahan bagi manusia dalam menjalani kehidupannya. Salah satu implementasi pencocokan string yang sering kita temukan dan gunakan adalah pada mesin pencari yaitu pada saat memberi rekomendasi *query* pada pengguna sebagai respons sistem atas masukan *query* pengguna. Pada saat memasukkan *query*, sistem akan menampilkan beberapa rekomendasi *query* atau *autocomplete query* pada kolom di bawah kolom pencairan. Pemilihan rekomendasi *query* disebabkan oleh beberapa faktor seperti banyaknya dokumen pada basis data dan jumlah klik pengguna dengan *query* yang sama. Algoritma pencocokan string dilakukan untuk menemukan *query* di *pattern* yang ada pada *text*. Setelah menemukan *query* di *text* yaitu judul yang ada di basis data, program akan mengambil satu sampai dua kata sebelum atau sesudah kata yang ditemukan. Setelah itu dilakukan pengurutan berdasarkan jumlah kemunculan yang paling banyak. Hasil dari proses yang dilakukan adalah rekomendasi *query* yang akan ditampilkan oleh sistem kepada pengguna. Hasil penelitian dari makalah ini tentu masih banyak memiliki kekurangan, karena pada kenyataan pun, rekomendasi *query* tidak hanya ditampilkan berdasarkan dokumen yang dimiliki oleh basis data, namun memiliki beberapa faktor lain yang mempengaruhi. Penelitian ini masih dapat mengalami perkembangan dan perbaikan lagi untuk menuju hasil yang diperoleh dapat menjadi lebih baik lagi.

Pranala Video di Youtube

https://youtu.be/nZKWpvOqcUQ

Ucapan Terima Kasih

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena dengan rahmat dan berkat-Nya, penulis dapat menyelesaikan penulisan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih kepada para Bapak/Ibu Dosen pengampu mata kuliah IF2211 Strategi Algoritma : Bapak Dr.Ir. Rila Mandala, M.Eng., Bapak Dr. Ir. Rinaldi Munir, M.T., Ibu Dr. Nur Ulfa Maulidevi, ST., M.Sc., dan Bapak Prof. Dwi Hendratmo W., Ph.D., yang telah mengajar mata kuliah IF2211 selama satu semester ini dan telah membantu penulis untuk memahami materi yang dijadikan sebagai bahan acuan dalam pembuatan makalah ini. Tak lupa pula, penulis mengucapkan terima kasih kepada semua pihak yang tidak dapat disebutkan satu persatu yang telah berkontribusi baik secara langsung maupun tidak langsung dalam membantu penyelesaian makalah ini.

Referensi

[1] GeeksforGeeks. 2020. *Applications of String Matching Algorithms*. Diambil dari https://www.geeksforgeeks.org/applications-of-string-matching-algorithms/. Diakses tanggal 10 Mei 2021.

[2] GeeksforGeeks. 2021. *KMP Algorithm for Pattern Searching*. Diambil dari https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/. Diakses tanggal 10 Mei 2021.

[3] H, Jeannie. 2015. *Predictive Search Can Match Search Queries to User Intent*. Diambil dari https://www.hillwebcreations.com/predictive-search-matches-search-queries-user-intent/. Diakses tanggal 9 Mei 2021.

[4] K, Yasin. 2018. *Apa Itu Search Engine & Apa Saja Fungsinya?*. Diambil dari https://www.niagahoster.co.id/blog/pengertian-search-engine/. Diakses tanggal 8 Mei 2021.

[5] M, Megan. 2015. *Predictive Search: Is This the Future or the End of Search*. Diambil dari https://www.wordstream.com/blog/ws/2013/06/24/predictive-search#:~:text=Google's%20predictive%20search%20feature%20uses,characters%20to%20the%20search%20input.. Diakses tanggal 9 Mei 2021.

[6] Program Studi Teknik Informatika. 2021. *Pencocokan String (String/Pattern Matching)*. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses tanggal 8 Mei 2021.

[7] Y.B. Lintang, dkk. 2014. *Analisis pada Fitur Autocomplete Suggestion dan Semantik pada Pencarian di Mesin Pencari Google*. Diambil dari

https://media.neliti.com/media/publications/170515-ID-analisis-pada-fitur-autocomplete-suggest.pdf. Diakses tanggal 9 Mei 2021.

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.

Medan, 10 Mei 2021

Arjuna Marcelino
13519021