Menentukan Tingkat Kecocokan Teks Menggunakan Algoritma KMP

Cut Meurah Rudi - 13514057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung – Jalan Ganesha No. 10 Bandung 40132, Indonesia
13514057@std.stei.itb.ac.id

***Abstract*—Dewasa ini berbagai kebutuhan dokumen yang dulu bersifat cetak telah berpindah menjadi dalam format digital. Perpindahan format dokumen dari fisik menjadi digital ini mendatangkan manfaat sekaligus terdapat pula kekurangan. Dari segi manfaat, penyebaran data menjadi lebih mudah tetapi kekurangannya mempermudah tindak plagiarisme. Untuk mendeteksi plagiarisme ini dibutuhkan pula sebuah program yang dapat mendeteksi kecocokan teks. Selain itu masih banyak lagi kebutuhan untuk mengetahui tingkat kecocokan dua buah teks, misalnya untuk pencarian menggunakan teks yang cukup panjang. Pada pencarian seperti ini string matching saja tidak cukup karena belum tentu teks yang ingin dicari memiliki kesamaan seratus persen dengan pattern.**

Keywords—text, string, match.

I. PENDAHULUAN

Digitilisasi di berbagai bidang saat ini memunculkan istilah baru yaitu paperless office. Paperless office ini bertujuan mengurangi penggunaan kertas pada kantor, sekolah maupun institusi-institusi lainnya. Tujuannya yaitu untuk menyelamatkan bumi, penggunaan kertas dalam jumlah besar menyebabkan meningkatnya penebangan pohon pada hutan-hutan, alhasil hutan menjadi gundul dan mengancam keselamatan bumi.

Arus digitalisasi ini membuat dokumen-dokumen yang dahulu dalam bentuk kertas, disimpan dalam lemari-lemari besar kini dapat disimpan dalam memory dengan ukuran yang jauh lebih kecil dibanding lemari-lemari tersebut tetapi dengan kapasitas yang jauh lebih besar. Saat ini rata-rata hard disk memiliki kapasitas 500 GB (Gigabyte). Jika setiap halaman dokumen berukuran 100kb, maka hard disk yang berdimensi 7 cm x 5 cm x 1,5 cm dapat menyimpan lima juta lembar kertas.

Penggunaan dokumen dalam bentuk digital ini memberikan dampak positif maupun dampak negatif. Salah satu dampak positifnya yaitu penyimpanan yang lebih efisien. Jika kita melihat dari dampak negatifnya, salah satu yang menjadi trend saat ini yaitu plagiarisme atau pembajakan. Data-data dalam bentuk digital ini dapat dengan mudah berpindah dari suatu tempat ke tempat lain. Terlebih dengan bantuan internet, seseorang dapat membobol jaringan komputer suatu kantor misalnya, kemudian dengan mudah mencuri data yang tersimpan pada penyimpanan jaringan tersebut.

Salah satu kebutuhan yang muncul akibat digitalisasi ini yaitu kebutuhan untuk menentukan kecocokan dua buah dokumen, atau kebutuhan untuk mencari dokumen mana yang cocok suatu dokumen dalam tumpukan data dalam jumlah besar pada memori penyimpanan.

Jika dahulu pencocokan dilakukan secara manual dengan cara membaca dokumen satu persatu, kini sudah seharusnya pencocokan dokumen ini dapat dilakukan secara otomatis untuk mempermudah dan mempersingkat waktu pencocokan dokumen.

II. PENCOCOKAN STRING

Dalam pencocokan string, dikenal dua buah istilah yaitu teks (T = t1t2....tm) dan pola (P = p1p2....pn), kemudian kita akan selalu mengasumsikan bahwa n < m. Dalam string matching, akan dicocokkan pola terhadap teks. Secara umum string matching ini terbagi dua, yaitu pencocokan pasti dan pencocokan perkiraan.

A. Pencocokan Pasti

Pada pencocokan pasti, pola harus menjadi upa himpunan dari teks. Misal kita memiliki T = “Informatika ada di Labtek V” dan P = “Labtek”. Pada contoh tersebut terdapat kecocokan karena “Labtek” merupakan upa himpunan dari “Informatika ada di Labtek V”. Namun jika kita mengubah sedikit saja pola menjadi P = “Labtex”, maka metode pencocokan pasti akan menyatakan bahwa pola dan teks tidak cocok.

Ada banyak algoritma yang dapat digunakan dalam pencocokan pasti ini, mulai dari algoritma brute force yang memiliki ongkos mahal dan kurang efisien, hingga ada pula algoritma knuth-morris-prat atau lebih dikenal KMP dan algoritma bayer moore. Bedanya kebanyakan algoritma memulai pengecekan dari p1 pada pola terhadap teks, tetapi pada algoritma bayer-moore, pencocokan dimulai dari pn pada

teks. Berikut disajikan algoritma pencocokan teks yang menggunakan pola pikir brute force untuk lebih memahami pencocokan pasti ini.

Brute Force

Algorithm

BruteForceStringMatch(*T*[0...*n*-1], *P*[0...*m*-1])

for *i* ← 0 **to** *n*-*m* **do**

j ← 0

while *j* < *m* **and** *P*[*j*] = *T*[*i*+*j*] **do**

j++

if *j* = *m* **then return** *i*

return -1

Algoritma brute force cukup sederhana, hanya menggunakan nested loop. Loop paling luar menggerakkan posisi karakter yang sedang digunakan pada teks, sedangkan loop terdalam menggerakkan posisi karakter pada pola.

Misalkan T = "aaatgt" dan P = "atgt"

		1	2	3	4	5	6
T	=	*a*	*a*	*a*	*t*	*g*	*t*
P	=	*a*	*t*	*g*	*t*		

Maka proses pencocokan pada brute force sebagai berikut:

1	2	3	4	5	6
a	*a*	*a*	*t*	*g*	*t*
a	*t*				
	a	*t*			
		a	*t*	*g*	*t*

Seperti pada gambar diatas, tulisan yang berwarna hijau menandakan bahwa huruf yang sedang aktif pada teks sesuai dengan huruf yang sedang aktif pada pola. Pencocokan dilakukan satu persatu pada pada teks, jika ada ketidakcocokan maka pencocokan kembali diulang mulai dari karakter pertama dari pola dan karakter selanjutnya dari teks, karakter selanjutnya yang dimaksud adalah karakter setelah karakter pertama yang dicocokkan dengan pola.

Brute force menjadi jauh lebih tidak efektif ketika teks yang digunakan misalnya T = "aaaaac" dan pola yang digunakan adalah P = "aaac", maka pencocokannya akan menjadi.

1	2	3	4	5	6
a	*a*	*a*	*a*	*a*	*c*
	a	*a*	*a*	*c*	
		a	*a*	*a*	*c*

B. Pencocokan Perkiraan

Pencocokan perkiraan jauh lebih rumit dibandingkan pencocokan pasti, karena juga dipertimbangkan kemiripan pola dan teks. Untuk pencocokan tidak pasti, dikenal sebuah fungsi jarak yang akan menghitung kesamaan antara teks dan pola.

Mari kita asumsikan teks (T) menjadi S1 dan pola (P) menjadi S2. Kemudian kita akan mengubah S1 dan S2 dengan tiga buah operator, yaitu operator hapus, operator sisip dan operator ganti. Untuk mempermudah penjelasana mengenai pencocokan perkiriaan, kita hanya akan mengubah S2 dengan tiga operator tadi.

Operator Hapus

Misalkan S1 = aacgt dan S2 = aaccggt. Kita akan menghapus c pada pada posisi ke-4 dan g pada posisi ke-5 dari S2. Kemudian setelah operasi penghapusan tersebut, kita dapat menyimpulkan bahwa kini S1 sudah sama dengan S2.

		1	2	3	4	5	6	7	8
S_1	=	*a*	*a*	*c*	-	-	*g*	*t*	
S_2	=	*a*	*a*	*c*	*c*	*g*	*g*	*t*	

Operator Sisip

Misalkan S1 = aactgt dan S2 = act. Kita akan menyisipkan a pada posisi ke-2 dan g pada peosisi ke-5 pada S2. Kemudian dapat kita lihat kembali, S1 telah sama dengan S2.

		1	2	3	4	5	6
S_1	=	*a*	*a*	*c*	*t*	*g*	*t*
S_2	=	*a*	-	*c*	*t*	-	*t*
			a			*g*	

Operator Ganti

Misalkan S1 = aactgtt dan S2 = abctatt. Kemudian kita akan mengubah kembali S2 menjadi S1 dengan mengganti karakter yang berada pada S2. Kita akan melakukkan operasi mengganti pada posisi ke-2 dan ke-5 dari S2. Pada posisi ke-2 kita menggantikan b dengan a dan paada posisi ke 5 kita menggantikan a dengan g. Kini dapat dilihat kembali S1 telah sama dengan S2.

		1	2	3	4	5	6	7	8
S_1	=	*a*	*a*	*c*	*t*	*g*	*t*	*t*	
S_2	=	*a*	*b*	*c*	*t*	*a*	*t*	*t*	
			a			*g*			

Contoh penggunaan operator pada pencocokan perkiraan sebagai berikut. Misalkan S1 = "aagttcgtta" dan S2 =

“aattacaa”. Kemudian kita akan mengubah S2 menjadi S1 dengan ketiga operator yang dimiliki pencocokan perkiraan.

		1	2	3	4	5	6	7	8	9	10
S_1	=	*a*	*a*	*g*	*t*	*t*	-	*c*	*g*	*t*	*a*
S_2	=	*a*	*a*	-	*t*	*t*	*a*	*c*	*a*	-	*a*
				g					*g*	*t*	

Seperti yang dapat kita lihat, operasi yang dilakukan sebagai berikut:

1. menyisipkan g pada posisi ke-3,
2. menghapus a pada posisi ke-6,
3. mengganti a dengan g pada posisi ke-8
4. menyisipkan t pada posisi ke-9

Pada kasus ini kita melakukan empat kali operasi dan itu merupakan jumlah operasi minimal yang dapat dilakukan untuk membuat S1 dan S2 menjadi sama. Dengan begitu maka fungsi jarak antara S1 dan S2 bernilai 4.

Fungsi jarak dapat dilihat sebagai ukuran kemiripan antara teks dan pola. Jika fungsi jarak nilainya semakin kecil, berarti teks dan pola memiliki kemiripan yang semakin tinggi, sebaliknya jika fungsi jarak nilaiya semakin besar berarti teks dan pola memiliki kemiripan yang semakin jauh.

Jika fungsi jarak bernilai nol, maka dapat kita katakan pola dan teks memiliki kesamaan seratus persen. Fungsi jarak ini biasanya dicari menggunakan pemrograman dinamis.

III. Algoritma Knuth-Morris-Pratt (KMP)

Sebelumnya kita telah membahas algoritma brute-force untuk pencocokan pasti. Sama seperti algoritma tersebut, algoritma Knuth-Morris-Pratt yang ditemukan oleh Donald E. Knuth, James H. Morris dan Vaughn. R. Pratt pada tahun 1977 pada sebuah paper dengan judul “Fast Pattern Matching” yang dipublikasikan pada journal SIAM Journal on Computing 6(2):323-350.

Selanjutnya kita akan mengilustrasikan bagaimana algoritma KMP ini bekerja, pertama kita misalkan

T = xyxxyxyxyyxyxyxyyyxyxyxxy

dan

P = xyxyyxyxyxx

Pada beberapa kasus, algoritma KMP ini sama saja dengan algoritma naïf atau algoritma brute force. Algoritma ini melakukan pergesaran pola dari posisi ke-1 pada teks hingga posisi ke n-m pada teks dan menentukan jika pola sesuai dengan teks.

Perbedaan antara KMP dan algoritma naïf yaitu, KMP menggunakan informasi yang dikumpulkan dari pencocokan sebagian pola terhadap teks sehingga dapat melakukan beberapa kali pergesaran secara sekaligus dan menjamin tidak akan terjadi kecocokan jika pegeseran hanya dilakukan sekali saja.

Misalkan kita memulai dari mensejajarkan teks dengan pola dengan sejajar kiri. Kemudian kita menggeser pattern perlahan ke kana sesuai dengan fungsi pinggiran yang dihasilkan, fungsi pinggiran menentukan berapa kali kita dapat menggeser pattern ke kanan, pencocokan berakhir terdapat upa himpunan dari teks yang sama dengan pola atau pergeseran telah membuat posisi pertama pada pola sama dengan n – m.

Mari kita lihat langkah-langkah yang dapat dilakukan untuk menerapkan algoritma kmp:

1. Misalkan situasinya P[1,….,3] berhasil disamakan dengan T[1,…..,3]. Kemudian dtitemukan ketidaksamaan pada karakter selanjutnya P[4] ≠ T[4]. Saat ini kita mengetahui bahwa P[1,….,3] = T[1,…..,3], kemudian kita dapat mengabaikan karakter pada pola dan teks setelah posisi ke-3.Lalu apa yang dapat kita hasilkan mengenai dimana kesamaan yang mungkin akan terjadi? Pada kasus ini algoritma KMP akan menggeser pola dua posisi ke kanan sehingga P[1] sejajar dengan T[3] dan perbandingan selanjutnya adalah antara P[2] dan T[4]
2. Kemudian kita ketahui bahwa P[2] ≠ T[4] dan pola bergeser ke kanan kembali sehingga P[1] sejajar dengan T[4] dan selanjutnya kita akan membandingkan P[1] dengan T[4].
3. Pada point selanjutnya, P[1,…..,10] telah berhasil disamakan dengan T[6,…..,15]. Kemdian ditemukan ketidaksamaan pada P[11] ≠ T[16]. Berdasarkan fakta sebelumnya, kita mengetahui bahwa T[6,…,15] = P[1,….,10]. Kemudian kita dapat mengabaikan karakter setelah posisi 10 pada pola dan karakter setelah posisi 15 pada teks. Kita dapat menduga bahwa pergesaran yang potensial untuk terjadi kesamaan yaitu 12. Kemudian, kita akan menggeser pola ke kanan, dimaa P[1,….,11] = T[13,….,23] dan perbandingan selanjutnya antara P[4] dan T[16] hasilnya sama, kemudian antara P[5] dan T[17] hasilnya juga sama dan perbandingan selanjutnya antara P[6] dan T[18] dan hasilnya juga sama. Karena semua pola telah disamakan dan hasilnya sama maka diperoleh lah upa himpunan dari teks yang sama dengan pola.

Aturan Penggeseran

Aturan penggeseran atau sering juga disebut fungsi pinggiran berguna untuk mempermudah kita dalam menggunakan algoritma KMP sehingga tiap kali ditemukan ketidakcocokan, dapat langsung diketahui kita harus menggeser pola ke kanan sebanyak berapa posisi.

Untuk menjelaskan mengenai aturan penggesera, pertama kita akan mendefinisikan terlebih dahulu beberapa notasi yang akan kita gunakan. Misalkan S adalah sebuah teks dan S = S1S2….Sk. Setiap upa himpunan dari teks akan berbentuk S1…Si, dimana 1 ≤ i ≤ k dan disebut suffiks dari S. Sebuah prefiks S’ dari S dan merupakan proper prefiks jika S’ tidak

sama dengan S, S' ≠ S. Sama seperti sebelumnya, setiap teks daam bentuk S1…Sk dimana 1 ≤ i ≤ k disebut suffiks dari S. Juga teks kosong (tidak mengandung karakter apapun) adalah suffiks dari S. Sebuah suffiks S' dari S merupakan proper suffiks jika S' tidak sama dengan S, S' ≠ S.

Sekarang misakan sebuah kondisi P[1,….,1] telah sama dengan teks T[i-q+1,…..,i] dan ditemukan ketidak cocokan pada karakter selanjutnya yaitu P[q+1] ≠ T[i+1]. Kemudian geser pola ke kanan sepanjang proper prefix dari P[1,….,q] yang juga merupakan suffix dari P[1,….,q] yang sekarang sejajar dengn teks dimana karakter terakhir dari prefiks ini sejajar dengan T[i].

Jika π(q) adalah sebuah nilai dimana P[1,….,π(q)] adalah proper prefiks terpanjang yang merupakan suffiks dari P[1,….,q], kemudian pola digeser kekanan sehingga P[1,….,π(q)] sejajar dengan T[i – π(q) + 1,…,i].

P:	x	y	x	Y	y	x	y	x	y	x	x
q:	1	2	3	4	5	6	7	8	9	10	11
Π(q):	0	0	1	2	0	1	2	3	4	3	0

Table 1 : Table of π values for pattern P

Algoritma KMP akan menghitung nilai dari π(q) kemudian menyimpan nilai-nilainya dalam sebuah table π[1,….,m]. Kita akan menjelaskan bagaimana tabel ini nanti.

Ringkasnya , sebuah langkah-lagkah dalam KMP dapat dibagi menjadi dua. Tapi sebelumnya, kita misalkan sebuah kondisi dimana P[1,…,q] sudah sama dengan T[I – q + 1,….,i].

1. Jika P[q + 1] = T[I + 1], panjang dari karakter yang sudah sama dapat bertamah, kecuali q + 1 = m, dimana pada kasus ini kita telah menemukan kesamaan keselurahan pola dengan upa himpunan dari teks.
2. Jika P[q + 1] ≠ T[I + 1], maka kita harus menggeser pola ke kanan.

Running Time

Setiap kali kita melakukan looping, ada dua kemungkinan yang terjadi, yang pertama yaitu kita menambah nilai i menjadi i + 1 atau kemungkinan kedua kita menggeser pola ke kanan sejauh π(q).

Kedua aksi tersebut mungkin dilakukan sebanyak n kali, dan jika loop diulang eksekusi dapat dilakukan hingga 2n kali. Cost untuk setiap kali iterasi pada loop yaitu O(1). Oleh karena itu waktu eksekusi adalah O(n), kemudian waktu untuk menghitung fungsi pinggiran π yaitu sepanjang m, maka waktu eksekusi total adalan O(n + m).

Menghitung nilai π(q)

Sekarang kita akan masuk lebih dalam mengenai bagaimana cara memperoleh table π. Tabel π ini juga biasa disebut fungsi jarak atau fungsi pinggiran yang berguna untuk memnentukan berapa banyak pergeseran ke kanan yang harus dilakukan jika terdapat ketidakcocokan antara sebuah karakter yang dibandingkan pada pola dengan sebuah karakter yang dibandingkan pada teks.

Misalkan table tersebut adalah π[1,….,m] yang berarti berukuran m buah, m adalah jumlah karakter yang terdapat pada pola. Kemudian kita anggap kondisinya kita telah menghitung π[1,…,i] dan kita ingin menghitung π[i + 1]. Pertama-tama kita telah ketahui bahwa P[1,….,π(i)] adalah proper prefix terpanjang dari P[1,…,i] dan juga merupakan suffix dari P[1,….,i].

Kemudian, misalkan q = π(i). Jika P[i + 1] = P[q + 1] kemudian berarti π(i + 1) = q + 1. Jika tidak, kita ubah nilai q menjadi π(q) dan proses diulangi kembali dan dilanjutkan hingga mencapai nilai q = 0, jika telah sampai pada q = 0, kita akan mengisi nilai π[i + 1] = 0.

Pseudocode Algoritma KMP

Algorithm KMP(P[1,…..,m],T[1…….,n])

input: pola P dan teks T dengan panjang m dan n

prekondisi : 1 ≤ m ≤ n

output : daftar semua nilai s, dimana P muncul pada pergeseran S pada T

```
q ← 0;
i ← 0;
while (i < n) //P[1,….q] == T[i – q + 1, …., i]
{
    if (P[q + 1] == T[I + 1])
    {
        q ← q + 1;
        i ← i + 1;
        if (q == m)
        {
        output i – q;
        q ← π(q);
        }
    }
    else // a mismatch occurred
    {
        if (q == 0) { i ← i + 1}
```

```
        else {q ← π(q)}
    }
}
```

IV. Menentukan Tingkat Kecocokan Dua Buah Teks

Sebelumnya kita telah mempelajari mengenai pencocokan string yang terdiri dari dua macam, yaitu pencocokan string dengan pendekatan pasti dan pencocokan string atau teks dengan pendekatan perkiraan. Pada bagian ketiga juga dijelaskan mengenai algoritma Knuth-Morris-Pratt, salah satu contoh algoritma pencocokan string dengan pendekatan pasti atau lebih familiar dengan istilah exact string matching.

Pada algoritma Knuth-Morris-Pratt yang ditemukan oleh tiga sekawan ini dengan nama "Fast String Matching" telah memperlihatkan kepada kita bagaimana menentukan kesamaan string dengan lebih cepat dan efisien dibandingkan dengan menggunakan algoritma brute force atau lebih dikenal dengan algoritma naif.

Sekarang kita akan masuk ke bagian inti dari publikasi ini, yaitu menentukan tingkat kecocokan dua buah teks, kita akan menggunakan bantukan salah satu algoritma dari pencocokan string dengan pendekatan pasti, yaitu algoritma Knuth-Morris-Pratt.

Lalu kita memilih menggunakan algoritma Knuth-Morris-Pratt karena ini adalah algortima yang tidak terlalu rumit untuk diaplikasikan tetapi mangkus dalam memecahkan permasalahan kecocokan antara teks dan pola. Dalam menentukan tingkat kecocokan antara dua buah teks ini kita tetap akan menggunakan istilah teks dan pola ketika kita mengaplikasikan algoritma Knuth-Morris-Pratt.

Perlu diketahui juga kita akan menggunakan istilah-istilah tambahan, yaitu D yang berarti Document. Document yang dimaksud adalah teks yang berukuran panjang dan hanya boleh mengandung kumpulan huruf-huruf, tidak boleh mengandung gambar karena kita tidak sedang ingin menentukan kesamaan dua buah gambar.

Document yang dimaksud juga harus merupakan kumpulan huruf yang membentuk rangkaian kata, kemudian rangkaian kata membentuk rangkaian kalimat. Hal ini perlu disampaikan karena kita tidak bermaksud untuk menyamakan dua buah document yang isinya bilangan biner seluruhnya, yang dimaksud dengan dokumen sama seperti tulisan ini.

Lalu untuk penjelasan mengenai bagaimana algoritma ini bekerja, pertama-tama misalkan ada dua buah Dokumen, yaitu D1 dan D2.

Misalkan D1 sebagai berikut:

"Digitilisasi di berbagai bidang saat ini memunculkan istilah baru yaitu paperless office. Paperless office ini bertujuan mengurangi penggunaan kertas pada kantor, sekolah maupun institusi-institusi lainnya. Tujuannya yaitu untuk menyelamatkan bumi, penggunaan kertas dalam jumlah besar menyebabkan meningkatnya penebangan pohon pada hutan-hutan, alhasil hutan menjadi gundul dan mengancam keselamatan bumi.. Arus digitalisasi ini membuat dokumen-dokumen yang dahulu dalam bentuk kertas, disimpan dalam lemari-lemari besar kini dapat disimpan dalam memory dengan ukuran yang jauh lebih kecil dibanding lemari-lemari tersebut tetapi dengan kapasitas yang jauh lebih besar. Saat ini rata-rata hard disk memiliki kapasitas 500 GB (Gigabyte). Jika setiap halaman dokumen berukuran 100kb, maka hard disk yang berdimensi 7 cm x 5 cm x 1,5 cm dapat menyimpan lima juta lembar kertas. Penggunaan dokumen dalam bentuk digital ini memberikan dampak positif maupun dampak negatif. Salah satu dampak positifnya yaitu penyimpanan yang lebih efisien. Jika kita melihat dari dampak negatifnya, salah satu yang menjadi trend saat ini yaitu plagiarisme atau pembajakan. Data-data dalam bentuk digital ini dapat dengan mudah berpindah dari suatu tempat ke tempat lain. Terlebih dengan bantuan internet"

dan D2 sebagai berikut:

"Paperless office ini bertujuan mengurangi penggunaan kertas pada kantor, sekolah maupun institusi-institusi lainnya. Tujuannya yaitu untuk menyelamatkan bumi, penggunaan kertas dalam jumlah besar menyebabkan meningkatnya penebangan pohon pada hutan-hutan, alhasil hutan menjadi gundul dan mengancam keselamatan bumi.. Arus digitalisasi ini membuat dokumen-dokumen yang dahulu dalam bentuk kertas, disimpan dalam lemari-lemari besar kini dapat disimpan dalam memory dengan ukuran yang jauh lebih kecil dibanding lemari-lemari tersebut tetapi dengan kapasitas yang jauh lebih besar. Saat ini rata-rata hard disk memiliki kapasitas 500 GB (Gigabyte). Jika setiap halaman dokumen berukuran 100kb, maka hard disk yang berdimensi 7 cm x 5 cm x 1,5 cm dapat menyimpan lima juta lembar kertas. Penggunaan dokumen dalam bentuk digital ini memberikan dampak positif maupun dampak negatif. Salah satu dampak positifnya yaitu penyimpanan yang lebih efisien. Jika kita melihat dari dampak negatifnya, salah satu yang menjadi trend saat ini yaitu plagiarisme atau pembajakan."

Lalu kita akan melakukan pencocokan terhadap D1 dan D2. Pertama-tama yang harus kita lakukan adalah membagi D1 menjadi potongan-potongan kata, pada contoh D1 terdiri dari 176 kata dan D2 terdiri dari 145 kata.

Kemudian tiap kata diberi indeks, untuk D1 indeksnya mulai dari 1 hingga 176, sehingga membentuk tabel D1[1,…,176], dan untuk D2 indeksnya mulai dari 1 hingga 145 sehingga membentuk tabel D2[1,…,145].

Kemudian menggunakan algoritma Knuth-Morris-Pratt kita akan menjadikan D1[1] sebagai pola dan D2[1,….,145] sebagai teks. Misalkan kecocokan ditemukan pada D2[i] dimana $1 \leq i \leq 145$ sehingga selanjutnya menggunakan algoritma KMP juga kita akan melakukan pencocokan dimana pola yang digunakan adalah D1[2] dan teks yang digunakan D2[i+1,…,145].

Namun bagaimana jika pada pencocokan dimana pola adalah D1[1] dan teks yang digunakan D2[1,….,145] tidak

juga ditemukan hingga indeks terakhir pada D2. Jika kasus ini terjadi maka pencarian tetap dilanjutkan dengan pola yang digunakan adalah D[2] tetapi teks yang digunakan tetap menggunakan teks awal yaitu D2[1…145].

Kemudian kita akan menghitung tingkat kecocokan antara D1 dan D2. Namun sebelum kita mulai menghitung tingkat kecocokan kedua document, perlu diketahui beberapa variabel berikut.

1. Kesamaan, tingkat kesamaan antara D1 dan D2 dalam persen.
2. hitungSama, variabel ini berisi jumlah perbandingan KMP yang tejadi yang menghasilkan nilai ≥ 0, atau sama dengan perbandingan KMP menemukan kesamaan antara pola dan teks.
3. m, merupakan jumlah kata yang terdapat pada D1, jika kita lihat pada contoh sebelumnya maka m bernilai 176.
4. n, variabel ini adalah jumlah kata yang terdapat pada D2, jika kita lihat pada contoh sebelumnya maka n bernilai 145.

Kemudian rumus yang akan kita gunakan untuk menghitung tingkat kesamaan antara D1 dan D2 adalah sebagai berikut:

$$\%Kesamaan = \frac{hitungSama}{(n+m)/2} x100$$

Jika kita meliht dari contoh sebelumnya, hitungSama = 145, n = 145 dan n = 176, maka %Kesamaan pada contoh sebelumnya yaitu:

$$\%Kesamaan = \frac{145}{(145+176)/2} x100 = 90{,}3\%$$

Dengan diperolehnya tingkat kesamaan dua buah teks tersebut, maka kita telah memperoleh tujuan akhir dari makalah ini, yaitu menentukan tingkat kesamaan dua buah teks yang direpresentasikan dalam persen (per seratus) oleh variabel kesamaan.

V. KESIMPULAN DAN SARAN

Pembahasan pada bagian VI menjelaskan bagaimana untuk menentukan tingkat kesamaan dua buah teks panjang yang dalam makalah ini disebut document D merupakan salah satu alternatif yang dapat digunakan untuk menentukan kesamaan antara dua buah teks.

Selain itu perlu juga diketahui bahwa algoritma ini belum sempurna karena kompleksitasnya bisa saja sama dengan algoritma naif jika pada kasus terburuknya oleh karena itu algoritma ini masih sangat mungkin untuk ditingkatkan.

TERIMA KASIH

Penulis mengucupakan terima kasih kepada Tuhan Yang Maha Esa yang berkat-Nya pula tulisan ini dapat diselesaikan dengan baik dan dapat dan sampai pada pembaca. Selain itu juga penulis ingin mengucapkan terima kasih kepada Pak Rinaldi Munir atas ilmu yang telah diajarkan kepada penulis, khususnya mengenai mata kuliah Strategi Algoritma yang menjadi dasar teori penulisan makalah ini.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. Strategi Algoritma. Bandung: STEI-ITB. 2009

[2] R. C. T. Lee, K. H. Chen, C. W. Lu, Y. K. Shieh, Exact and Approximate String Matching.Taiwan:2006

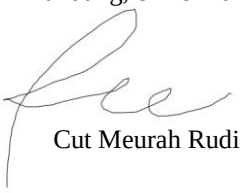
[3] Christian-Charras, Thierry Lacroq. Exact String Matching Algorithm.Prancis:Universite de Rouen. 1997

[4] *Baeza-Yates R, Navarro G (June 1996). "A faster algorithm for approximate string matching". In Dan Hirchsberg, Gene Myers. Combinatorial Pattern Matching (CPM'96), LNCS 1075. Irvine, CA. pp. 1–23.*R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

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, 8 Mei 2016



Cut Meurah Rudi