Implementasi Algoritma KMP dan Boyer-Moore dalam Aplikasi *Search Engine* Sederhana

Moch. Yusup Soleh/13507051
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If17051@students.if.itb.ac.id

***Abstrak*—KMP dan Boyer-Moore merupakan algoritma pattern matching yang memiliki kinerja bagus. Penerapan algoritma ini dalam *search engine* akan membantu meningkatkan performansi pencarian mesin tersebut menjadi lebih baik dari pada hanya dengan menggunakan algoritma Brute force. Goole'i merupakan *search engine* sederhana yang menerapkan kedua algoritma tersebut dalam pencarian string query-nya.**

***Kata Kunci—KMP, Boyer-Moore, Search engine*.**

I. PENDAHULUAN

Algoritma pencocokan string (pattern) yang mempunyai kinerja bagus adalah Knuth-Morris-Pratt (KMP) dan Algoritma Boyer-Moore. Kedua algoritma ini popular digunakan pada editor teks (menu find), search engine, analisis citra, dan sebagainya.

Search engine atau mesin pencari adalah program komputer yang dirancang untuk membantu seseorang menemukan file-file yang disimpan dalam komputer, misalnya dalam sebuah server umum di web (WWW) atau dalam komputer sendiri. Mesin pencari memungkinkan kita untuk meminta content media dengan kriteria yang spesifik (biasanya yang berisi kata atau frasa yang kita tentukan) dan memperoleh daftar file yang memenuhi kriteria tersebut. Mesin pencari biasanya menggunakan indeks (yang sudah dibuat sebelumnya dan dimutakhirkan secara teratur) untuk mencari file setelah pengguna memasukkan kriteria pencarian.

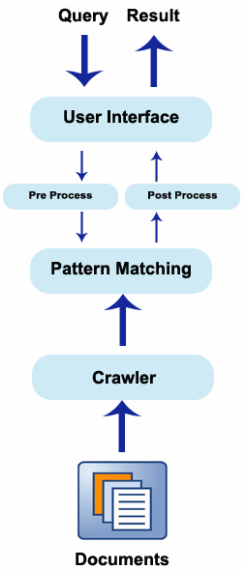
Pada makalah ini akan dijelaskan mengenai implementasi algoritma Knuth-Morris-Pratt dan Boyer-Moore dalam aplikasi s*earch engine* yaitu Goole'i. Goole'i ini bekerja layaknya *search engine* pada umumnya, seperti google, namun lingkup pencarian Goole'i lebih sederhana, yaitu dokumen yang terdapat dalam file-file pada komputer.

II. GOOLE'I SEARCH ENGINE

Search engine yang dibuat untuk mengimplementasikan algoritma KMP dan Boyer-Moore ini diberi nama "Goole'i", yang dibuat penulis bersama dengan dua anggota tim lainnya, yaitu Didik Haryadi dan Adventus Wijaya.

A. Arsitektur dan Desain

Arsitektur Goole'i terdiri dari empat komponen utama, yaitu crawler, pre-process, pattern matching dan post-process. Seperti terlihat pada gambar 1.



Gambar 1. Arsitektur Goole'i

Crawler berperan dalam pengambilan dokumen-dokumen yang ada dalam direktori dan sub-sub direktori dari *path* yang telah ditentukan.

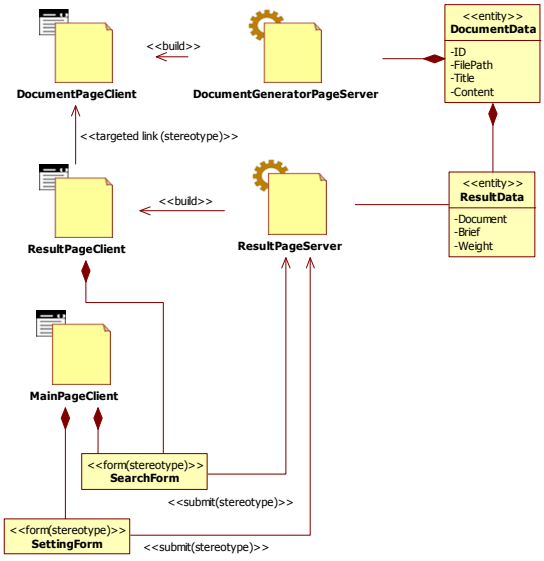
Pre-process berperan dalam memisahkan query menjadi list kata atau list pattern yang nantinya akan dicari padanannya dengan dokumen-dokumen yang ada. Pada proses ini juga dilakukan penghilangan *stopwords* sehingga didapatkan kata-kata atau *query* yang relevan. Dengan penghilangan *stopwords* ini maka kemungkinan akan dokumen yang tidak relevan terambil menjadi kecil. *Stopwords* ini berupa kata-kata yang sering terdapat dalam dokumen yang bukan merupakan inti isi dari dokument tersebut, seperti kata *yang, dan, kemudian, lagi* dll.

Pattern matching merupakan inti dari *search engine*

dimana disini terjadi proses pencarian string yang cocok dari query-query yang dimasukan dengan dokumen-dokumen yang ada.

Post-process berperan dalam mengekstrak dokumen untuk mendapatkan ringkasan serta penghitungan bobot dari kemunculan untuk perankingan dokumen dan siap ditampilkan dalam bentuk **Result**.

Adapun kelas perancangan Goole'i dapat dilihat pada gambar 2.



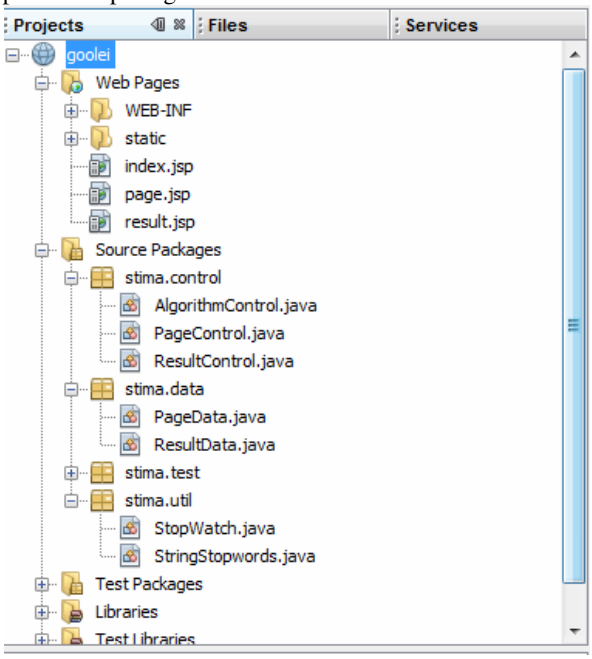
Gambar 2. Kelas Perancangan Goole'i

Dalam kelas perancangan tersebut terdapat tiga interface utama, yaitu MainPageClient, ResultPageClient dan DocumentPageClient yang mana ResultPageClient dan DocumentPageClient masing-masing merupakan instance dari *server page* ResultPageServer dan DocumentGeneratorPageServer. MainPageClient berisi tampilan utama yang berisi form untuk menseting path dan algoritma yang akan digunakan serta form untuk melakukan pencarian. ResultPageClient menampilkan hasil dari pencarian yang telah dilakukan oleh *search engine*, selain itu dalam antarmuka ini ditampilkan juga atribut dari hasil pencarian seperti, bobot dari dokumen terkait, serta ditampilkan juga waktu pencarian secara keseluruhan. Hasil-hasil yang ditampilkan dalam ResultPageClient me-*link* ke halaman yang menampilkan isi dari dokumen yang bersangkutan yang digenerate oleh DocumentGeneratorPageServer.

Server page ResultPageServer merupakan elemen *boundary* atau *view* sekaligus *control* utama dari *search engine* Goole'i, dimana terjadinya pre-process, pattern-matching dan post-process. Result atau hasil dari pencarian akan ditampilkan dengan membangun instance (build) page yaitu ResultPageClient.

Search engine Goole'i ini diimplementasikan dalam lingkungan java dengan menggunakan teknologi java web. Pada tahap implementasinya sendiri, Goole'i di bagi menjadi 3 paket (*package*), yaitu paket data, control dan util. paket data berisi kelas PageData.java dan ResultData.java yang masing-masing merupakan struktur data dari dokumen dan hasil pencarian (Result). Dalam kelas control terdapat 3 kelas, yaitu AlgorithmControl.java, PageControl.java dan ResultControl.java. PageControl.java merupakan Crawler *search engine* Goole'i, dimana kelas ini menyimpan semua informasi dari dokumen yang ada. Sedangkan ResultControl.java merupakan implementasi dari control utama yang berperan dalam proses pencarian utama. Dalam paket util terdapat kelas StringStopwords.java dan StopWatch.java yang merupakan kelas tambahan dalam pembangunan *search engine* Goole'i. Implementasi dari interface dalam Goole'i ini yaitu index.jsp, result.jsp dan page.jsp yang masing masing merupakan implementasi dari kelas perancangan MainPageClient, ResultPageServer dan DocumentGeneratorPageServer.

Struktur project yang merupakan implementasi Goole'i dapat dilihat pada gambar 3.



Gambar 3. Struktur project Goole'i.

B. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth Morris Pratt (KMP) dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R. Pratt. Untuk pencarian string dengan menggunakan algoritma Brute Force , setiap kali ditemukan ketidakcocokan pattern dengan teks, maka pattern akan digeser satu karakter ke kanan. Sedangkan pada algoritma KMP, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti halnya pada

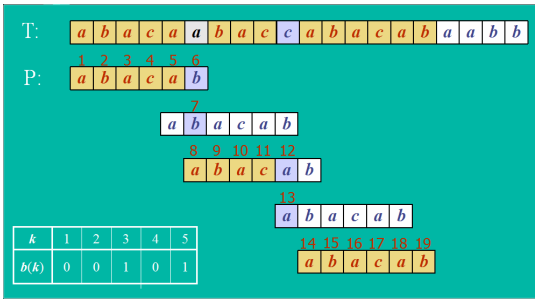
algoritma brute force.

Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan 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, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Algoritma ini menemukan semua kemunculan dari pattern dengan panjang *n* di dalam teks dengan panjang *m* dengan kompleksitas waktu O(*m*+*n*). Algoritma ini hanya membutuhkan O(*n*) ruang dari memory internal jika teks dibaca dari file eksternal. Semua besaran O tersebut tidak tergantung pada besarnya ruang alphabet.

Berikut contoh pencocokan pattern dengan menggunakan algoritma KMP.



Gambar 4. Contoh pattern matching dengan KMP.

Boyer-Moore

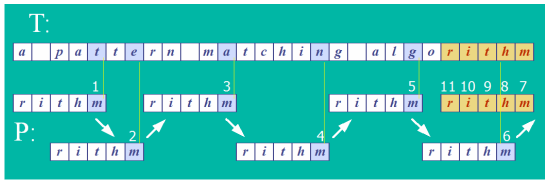
Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokkan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat mencocokkan string adalah:

1. Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
2. Dari kanan ke kiri, 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 dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Tabel untuk penggeseran bad-character dan good-suffix dapat dihitung dengan kompleksitas waktu dan ruang sebesar O(n + σ) dengan σ adalah besar ruang alfabet. Sedangkan pada fase pencarian, algoritma ini membutuhkan waktu sebesar O(mn), pada kasus terburuk, algoritma ini akan melakukan 3n pencocokkan karakter, namun pada performa terbaiknya algoritma ini hanya akan melakukan O(m/n) pencocokkan.

Berikut contoh pencocokan pattern dengan menggunakan algoritma Boyer-Moore.



Gambar 5. Contoh pattern matching dengan Boyer-Moore.

C. Cara Kerja

Search engine Goole'i memiliki cara kerja yang cukup sederhana.

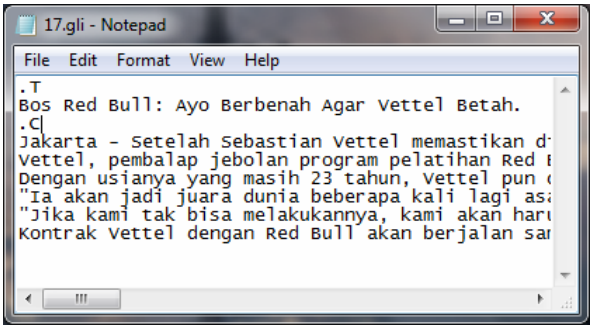
1. Input query dimasukan oleh user, misalnya: "Facebook atau Youtube, mana yang lebih baik?".
2. Query di-*split* menjadi list kata dan setiap *stopwords* dihilangkan, misalnya: {"Facebook","Youtube", "baik"}. (pre-process)
3. Setiap kata yang merupakan pattern dicari padanannya (dilakukan pattern matching) dengan isi (judul dan/atau konten) dari dokumen-dokumen yang ada.
4. Setiap dokumen yang memiliki padanan dilakukan pengecekan ulang apakah merupakan padanan satu kata utuh atau tidak. Setiap padanan yang bukan merupakan padanan 1 kata utuh akan dihilangkan, misalnya: padanan "baikan" untuk pattern "baik" akan dihapus. (post-process)
5. Dokumen yang berisi padanan kata akan diekstrak ringkasannya dan dilakukan pembobotan. (post-process)

6. Hasil dari pencarian ditampilkan.

III. METODE DAN PENGUJIAN

A. Data Test

Data test yang digunakan berupa file eksternal dengan ekstensi *.gli yang merupakan dokumen yang berisi judul dan konten. Untuk membedakan antara judul dan konten diletakan ".T" dan ".C" yang masing-masing merepresentasikan judul dan konten dari dokumen.



Gambar 6. Contoh file dokumen.

Dari contoh diatas, kalimat "Bos Red Bull: ..." merupakan judul, dan "Jakarta – setelah sebastian ..." merupakan konten dokumen.

Dalam pengujian yang dilakukan disediakan data test sebanyak 55 buah file .gli.

B. Pengujian dengan KMP

Pengujian dilakukan dengan terlebih dahulu menseting algoritma dengan algoritma KMP pada halaman utama. Pelaksanaan pengujian dilakukan oleh pengguna dengan cara memasukan query kedalam form pencarian yang tersedia.

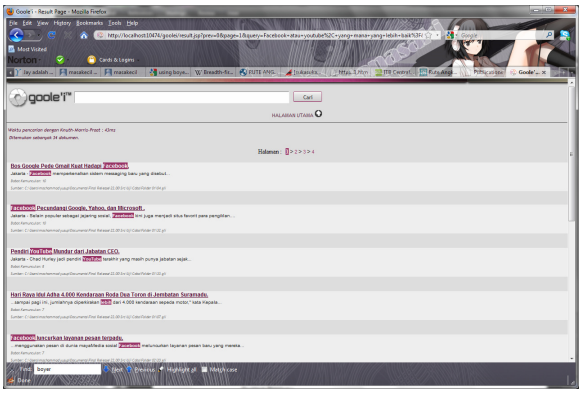
C. Pengujian dengan Boyer-Moore

Pengujian dilakukan dengan terlebih dahulu menseting algoritma dengan algoritma Boyer-Moore pada halaman utama. Pelaksanaan pengujian dilakukan oleh pengguna dengan cara memasukan query kedalam form pencarian yang tersedia.

IV. HASIL PENGUJIAN DAN ANALISIS

A. KMP

Hasil pencarian oleh Goole'i untuk query user "Facebook atau youtube, mana yang lebih baik?", dengan menggunakan algoritma Knuth-Morris-Pratt (KMP) mendapatkan hasil pencarian 34 dokumen relevan (terdapat padanan kata), dengan waktu pencarian 43ms.

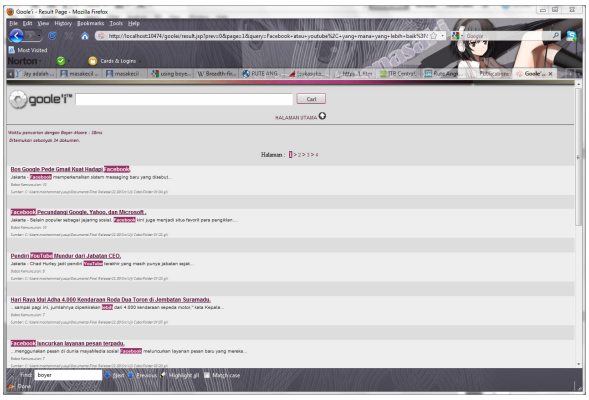


Gambar 7. Hasil pengujian algoritma KMP.

.

B. Boyer-Moore

Hasil pencarian oleh Goole'i untuk query user "Facebook atau youtube, mana yang lebih baik?", dengan menggunakan algoritma "Boyer-Moore" mendapatkan hasil 34 dokumen relevan (terdapat padanan kata), dengan waktu pencarian 38ms.



Gambar 8. Hasil pengujian algoritma Boyer-Moore

C. Analisis

Dari hasil kedua algoritma tersebut didapatkan pencarian dengan menggunakan algoritma Boyer-Moore lebih cepat dari pencarian dengan menggunakan algoritma KMP. Hal ini dikarenakan dokumen-dokumen dalam data test yang dipakai merupakan teks dengan alpabet yang banyak yang tentunya algoritma Boyer-Moore akan lebih cocok, karena kemungkinan terjadinya *missmatch* dalam proses komparasi besar.

Algoritma KMP sendiri sangat cocok untuk pattern matching dengan teks yang menggunakan alpabet sedikit, seperti biner. Karena kemungkinan terjadinya kesamaan antara prefiks dan postfiks pada pattern semakin besar sehingga setiap kali terjadi missmatch proses komparasi tidak *redundant*.

Bila dilihat pada hasil yang diberikan memang perbedaan rentan waktu pencarian hanyalah sedikit yaitu 6ms, karena pattern matching dilakukan dengan data test dokumen yang sedikit yaitu 55 buah. Jika menggunakan

data test yang jauh lebih banyak, misalnya jutaan, maka perbedaan rentan waktu akan semakin mencolok.

V. KESIMPULAN DAN SARAN

Implementasi algoritma pencocokan string seperti KMP dan Boyer-Moore akan mampu meningkatkan performansi dari *search engine* dalam pencarian query dalam dokumen.

Algoritma Knuth-Morris-Pratt lebih cocok untuk pencarian string dengan alpabet sedikit seperti biner. Dan algoritma Boyer-Moore lebih cocok untuk pencarian string dengan alpabet banyak.

Untuk penelitian selanjutnya sebaiknya pencarian dilakukan untuk kata dasar dari suatu kata. Hal ini dapat dilakukan dengan menambahkan morphological analiser kedalam *search engine* yang dibangun. Karena pada Goole'i ini tidak bisa mencocokan kata dengan kata imbuhannya, misalnya: "facebook", dengan "Facebookku" atau "tabrak" dengan "menabrak"/"tabrakan", padahal keduanya merupakan kata yang saling terkait.

Untuk pengembangan lebih lanjut *search engine* bisa mengatasi permasalahan homonim dan sinonim yang sering terjadi dalam pencarian. Hal ini dapat diatasi dengan menambahkan Thesaurus kedalam *search engine*.

DAFTAR PUSTAKA

[1] Boyer R.S., Moore J.S.,.1977. A fast string searching algorithm. Communications of the ACM. 20:762-772.

[2] Donald Knuth, James H. Morris, Jr. Vaughan Pratt. 1977. *Fast pattern matching in strings*. SIAM Journal on Computing.

[3] J.S. Moore, Matt Martinez. 2008. *A Mechanically Checked Proof of the Correctness of the Boyer-Moore Fast String Searching Algorithm*. University of Texas.

[4] M Yusup Soleh, Didik Haryadi, Adventus W.L.. 2010. *Tugas Besar IF3051 Strategi Algoritma:Aplikasi Algoritma KMP dan Boyer-Moore dalam Search Engine*. Institut Teknologi Bandung.

[5] Munir, Rinaldi. 2007. *Diktat Kuliah IF2251 Strategi Algoritma*. Bandung: Penerbit ITB.

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, 09 Desember 2010

Moch. Yusup Soleh / 13507051