PENCARIAN KATA MENGGUNAKAN ALGORITMA TUNED BOYER MOORE YANG DIIMPLEMENTASIKAN PADA WEB CRAWLER

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

oleh:
MUHAMMAD CHANDRA SAPUTRA
0410963032-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

PENCARIAN KATA MENGGUNAKAN ALGORITMA TUNED BOYER MOORE YANG DIIMPLEMENTASIKAN PADA WEB CRAWLER

Oleh:
MUHAMMAD CHANDRA SAPUTRA
0410963032-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 5 Agustus 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Bayu Rahayudi, ST., MT	**Candra Dewi, S.Kom, M.Sc**
NIP. 197407122006041001	**NIP. 197711142003122001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya
Ketua,

Dr. Abdul Rouf Alghofari, M.Sc.
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama **:** Muhammad Chandra Saputra
NIM **:** 0410963032-96
Jurusan **:** Matematika
Penulis skripsi berjudul **:** Pencarian Kata Menggunakan Algoritma Tuned Boyer Moore yang Diimplementasikan Pada Web Crawler.

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 5 Agustus 2011
Yang menyatakan,

(Muhammad Chandra Sapurta)
NIM. 0410963032

PENCARIAN KATA MENGGUNAKAN ALGORITMA TUNED BOYER MOORE YANG DIIMPLEMENTASIKAN PADA WEB CRAWLER

ABSTRAK

Sebuah pencarian kata merupakan proses pencocokan antara teks dan *pattern*. Jumlah kemiripan teks dengan *pattern* yang dicari pada dokumen dan posisi rinci kata tersebut juga diperlukan. Algoritma pencocokan *string* (*string matching*) yang merupakan bagian utama dalam proses pencarian kata memegang peranan penting untuk mendapatkan informasi yang sesuai dengan kebutuhan.

Aplikasi yang dibangun menggunakan web *crawler* ini digunakan untuk melakukan pencarian ke halaman-halaman situs yang terdapat di internet untuk mencari kata yang ada di situs tersebut. Tidak semua situs akan dikunjungi oleh situs *crawler* ini, hanya situs-situs yang telah ditentukan sebelumnya yang akan menjadi tujuan *crawling*.

Untuk data *precision* dari analisa tersebut dapat dijelaskan pula, rata-rata nilai *precision* dari percoban ini adalah 0,59 dengan kata lain bahwa sistem berjalan dengan baik untuk menemukan kata yang di cari. Perhitungan waktu yang dilakukan didapatkan rata-rata waktu yang diperlukan untuk menemukan satu proses pencocokan *tuned boyer moore* adalah 5.97 detik.

Faktor yang berpengaruh pada percoban ini adalah perangkat keras. Pada percobaan ini digunakan komputer *desktop* sebagai perangkat keras pengujian dengan spesifikasi terbatas, kondisi tersebut menyebabkan pada saat proses *crawling*, *browser* sering *hang* akibat kehabisan *memory* dan *CPU Usage* yang mencapai 100%.

Kata Kunci : *string matching*, *tuned boyer moore*, *web crawler*.

WORD SEARCH USING TUNED BOYER MOORE ALGORITHM IMPLEMENTED ON WEB CRAWLER

ABSTRACT

A word search is a process to match text and pattern. The amount of similarity between the text and pattern searched in the document and the detailed position of the word is also needed. The string matching process algorythm, which is the main part in string searching process, has an important role in getting the needed information.

The application, which is built using crawler site, is used to perform searchings to sitepages in the internet to find the words in the sites. Not every site will be visited by the crawler site, only those that has been predetermined as the object of *crawling*.

For the precision from the analysis, it can also be explained that the average precision value from the test is 0,59, this means that the crawler application using tuned boyer moore has works well in finding the looked up word. For the time recording, the average time needed to perform crawling process on for every tuned boyer moore algorithm proces is 5.97s.

Other factor that affects the test is the hardware. The test uses desktop computer with limited specifications as the testing hardware, this often causes browser to hang when performing crawling, due to the computer running out of memory and the CPU usage which reaches 100%.

Keyword : *string matching*, *tuned boyer moore*, *web crawler*.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahNya, penulis masih dapat belajar dan mengerjakan skripsi yang berjudul "**PENCARIAN KATA MENGGUNAKAN ALGORITMA TUNED BOYER MOORE YANG DIIMPLEMENTASIKAN PADA WEB CRAWLER**". Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bayu Rahayudi, ST., MT sebagai Pembimbing I dan Candra Dewi, S.Kom, M.Sc selaku pembimbing II. Terima kasih atas semua waktu dan bimbingan yang telah diberikan.
2. Segenap bapak dan ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
3. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya
4. Ayah, ibu, kakak dan adik dan seluruh keluarga. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangatnya.
5. Putri Damayanti, Herry Satrio, Eko Nugroho, Widhy Hayuhardika, Farid Jauhari, teman-teman PPTI-UB dan Asrama Banjarbaru atas bantuan, semangat dan doanya.
6. Sahabat- sahabat Ilkomers angkatan 2004, seluruh angkatan dan keluarga besar Ilmu Komputer Universitas Brawijaya.
7. Pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini disebabkan oleh keterbatasan kemampuan dan pengalaman. Oleh karena itu Penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi skripsi ini untuk kelanjutan penelitian serupa di masa mendatang.

Penulis berharap semoga skripsi ini dapat memberikan manfaat kepada pembaca dan bisa diambil manfaatnya, baik oleh Penulis selaku mahasiswa maupun pihak-pihak lain yang tertarik untuk menekuni pengembangan algoritma *string matching*.

Malang, 5 Agustus 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL


UNIVERSITAS BRAWIJAYA

DAFTAR SOURCE CODE

DAFTAR RUMUS


UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sebuah pencarian kata merupakan proses pencocokan antara teks dan *pattern*. Hasil pencocokan tersebut merupakan kombinasi huruf yang menjadi kata kunci dengan data kata yang ada, sehingga jumlah kemiripan dengan kata yang dicari pada tiap teks dan posisi rinci kata tersebut juga juga didapatkan. Algoritma pencocokan *string* (*string matching*) yang merupakan bagian utama dalam proses pencocokan kata dan memegang peranan penting untuk mendapatkan informasi yang sesuai dengan kebutuhan.

Perkembangan teknologi yang cukup pesat saat ini memungkinkan untuk mendapatkan data, informasi dengan mudah dan cepat. Data dan informasi tersebut semakin dibutuhkan, maka manusia berusaha untuk mengembangkan teknologi yang mendukung agar pertukaran data semakin cepat dan akurat. Berbagai teknologi telah tersedia, sedikit banyak telah membantu dalam memenuhi kebutuhan akan informasi.

Web crawler merupakan penyederhanaan dari proses pengolahan informasi web yang membuat web lebih ramah dan lebih berguna. Mengoperasikan web dengan menggunakan kata kunci pada media pencarian sering lebih intuitif daripada mencoba untuk menggunakan *Uniform Resource Locator* (URL) untuk mengakses halaman Web secara langsung. Jika pengguna memiliki pengalaman yang baik, mereka lebih cenderung untuk terus menggunakan pencarian pada web. web crawler telah memberi kontribusi pada kemudahan, kesederhanaan pencarian dan pertumbuhan web. Seorang pencari informasi dapat menemukan luasnya informasi tentang topik tertentu dan dapat menggunakan informasi tersebut untuk lebih menyempurnakan tujuannya (Brian, 2000).

Proses *crawling* dalam banyak cara yang sama seperti seseorang akan jika dia berusaha untuk membangun koleksi halaman web secara manual. web crawler memulai dengan URL tunggal, mengunduh halaman, mengambil link dari halaman, dan mengulangi proses dengan masing-masing halaman tersebut. Dalam prosesnya, web crawler akan menemukan link ke sebagian besar halaman di

web, walaupun membutuhkan beberapa waktu untuk benar-benar mengunjungi masing-masing halaman (Brian, 2000).

Aplikasi yang akan dibuat ini menggunakan *web crawler* untuk melakukan pencarian ke halaman-halaman situs yang terdapat di internet untuk mencari frekuensi kemunculan kata yang ada di situs tersebut. Tidak semua situs akan dikunjungi oleh *web crawler* ini, hanya situs-situs yang telah ditentukan sebelumnya yang akan menjadi tujuan *crawling*. Setelah data yang diinginkan telah ditemukan, maka data tersebut akan disimpan di basis data dan akan digunakan untuk proses selanjutnya.

Pencocokan string adalah operasi dasar dalam ilmu komputer. Untuk prosesnya dibutuhkan teks string (T) dan pola string (P) untuk menemukan semua kejadian P di T. Pencocokan String tidak hanya menyediakan masalah yang menantang para ilmuwan komputer teoritis, tetapi hanya memiliki banyak aplikasi termasuk query database, pengolahan teks, DNA dan analisis urutan protein, pencari internet, crawler, dan sebagainya (Cao, 2004).

Banyak algoritma dan metode pencarian yang sudah dikenal oleh *programmer*. Algoritma-algoritma tersebut antara lain algoritma *Brute Force*, algoritma dari *Morris* dan *Pratt*, yang kemudian biasa disebut oleh algoritma *Knuth*, *Morris*, dan *Pratt*. Algoritma *Colussi*, Algoritma *Crochemore-Perrin*, algoritma *Boyer* dan *Moore*, algoritma *turbo boyer-moore*, algoritma *tuned boyer-moore*, dan algoritma *zhu-takaoka*.

Metode Tuned Boyer Moore merupakan pengembangan dari metode Knuth Morris Pratt, pada metode Boyer Moore proses dimulai membuat perbandingan dari akhir string pencarian daripada awalnya. Hal ini juga menggunakan dua aturan pergeseran aturan *bad character* dan *good suffix*, aturan ini juga digunakan untuk pengembangan algoritma *turbo boyer moore* dan *tuned boyer moore* (Cao, 2004).

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah:

1. Bagaimana mengimplementasikan algoritma *Tuned Boyer Moore* pada *crawler* situs?
2. Bagaimana analisa terhadap penerapan algoritma *Tuned Boyer Moore* pada *crawler* situs dengan menggunakan parameter

perbandingan jumlah frekuensi kemunculan kata dan lama waktu pencarian?

1.3 Tujuan

Tujuan yang ingin dicapai dari pembuatan skripsi ini adalah:

1. Mengimplementasikan algoritma *Tuned Boyer Moore* pada *crawle* situs.
2. Mengukur jumlah frekuensi kemunculan kata, lama waktu pencarian dengan menggunakan algoritma *Tuned Boyer Moore* pada *web crawler*.

1.4 Batasan Masalah

Batasan masalah dalam penulisan skripsi ini adalah:

1. Hanya membahas mengenai pencocokan kata menggunakan algoritma *Tuned Boyer Moore*.
2. Data yang digunakan berupa bagian *body* dan *title* pada halaman HTML.
3. Parameter yang dianalisis terbatas pada perbandingan frekuensi kemunculan kata dan lama waktu pencarian.

1.5 Manfaat

Manfaat yang diperoleh dari penulisan skripsi ini adalah memberikan sebuah solusi pencarian kata pada teks menggunakan algoritma *Tuned Boyer Moore* yang di implementasikan pada *crawler* situs.

1.6 Metode Penyelesaian Masalah

Metode penyelesaian masalah yang dilakukan pada penelitian ini, yaitu :

1. Studi Literatur
   Membaca dan mempelajari beberapa literatur (jurnal, buku dan artikel dari situs) mengenai *crawler* situs dan algoritma *Tuned Boyer Moore*.
2. Perancangan dan implementasi perangkat lunak

Merancang dan membangun sebuah perangkat lunak (*crawler* situs) yang mengimplementasikan pencarian kata menggunakan algoritma *Tuned Boyer Moore*.

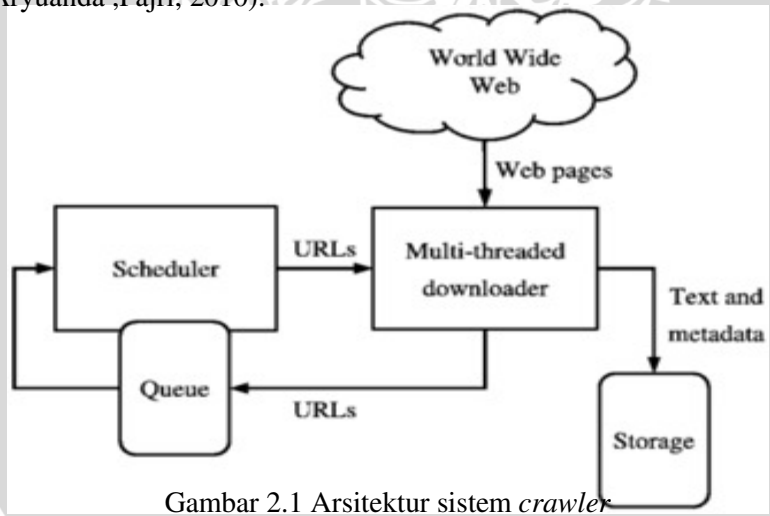
3. Uji coba dan analisa hasil implementasi
   Menganalisa hasil implementasi, yaitu perbandingan jumlah frekuensi kemunculan kata dan lama waktu pencarian.

BAB II
TINJAUAN PUSTAKA

2.1 Web Crawler (*Crawler* Situs)

Web Crawler merupakan program pengumpul informasi yang hasilnya akan disimpan pada sebuah basis data. Sebuah *web crawler* akan berjalan menelusuri halaman situs dan mengumpulkan dokumen-dokumen atau data-data di dalamnya. Selanjutnya *web crawler* akan mengurutkan dokumen-dokumen tersebut dan membangun sebuah daftar indeks untuk memudahkan proses pencarian.

Tujuan pengurutan adalah untuk menentukan seberapa penting suatu dokumen dan untuk memperkirakan halaman-halaman situs yang mungkin dan berhubungan sehingga halaman situs tersebut perlu ditelusuri lebih dahulu. Proses terpenting dari sebuah *crawler* adalah estimasi *link*. Proses ini menentukan *link* yang harus dijelajahi terlebih dahulu, sehingga jika halaman yang dipilih sesuai dengan topik yang diinginkan maka akan mendapatkan hasil yang maksimal (Aryuanda ,Fajri, 2010).



Gambar 2.1 Arsitektur sistem *crawler*
(Dwi Utami, Pawestri dkk, 2009)

Gambar 2.1 merupakan arsitektur dari sistem *crawler*. *Crawler* diawali dengan adanya daftar halaman situs yang akan dikunjungi, disebut dengan *seeds*. Setelah *crawler* mengunjugi halaman situs

tersebut, kemudian mengidentifikasi semua *hyperlink* dari halaman itu dan menambahkan kembali ke dalam *seeds*. Hal ini dinamakan *crawl frontier*.

Setelah *web crawler* mengunjungi halaman-halaman situs yang ditentukan di dalam *seeds*, maka *web crawler* membawa data-data yang dicari oleh user kemudian menyimpanya ke sebuah *storage*. *Web crawler* dapat dibuat untuk mencari informasi yang berhubungan dengan topik tertentu saja. Web crawler yang hanya mengumpulkan topik tertentu saja disebut dengan *topical web crawler*.

Proses *crawling* merupakan proses dimana *web crawler* mengumpulkan data-data dari halaman sebuah situs. *Web crawler* dimulai dengan sekumpulan halaman situs, kemudian mengunduh setiap halamannya, mendapatkan *link* dari setiap halaman yang dikunjungi kemudian mengulangi kembali proses *crawling* pada setiap link halaman tersebut. (Dwi Utami, Pawestri dkk, 2009)

2.2 Indexing

Indexing system bertugas untuk menganalisa halaman situs yang telah tersimpan sebelumnya dengan cara mengindeks setiap kemungkinan term yang terdapat di dalamnnya. Data yang ditemukan disimpan dalam sebuah basis data untuk digunakan dalam pencarian selanjutnya.

Indexing system mengumpulkan, memilah dan menyimpan data untuk memberikan kemudahan dalam pengaksesan informasi secara tepat dan akurat. Proses pengolahan halaman situs agar dapat digunakan untuk proses pencarian berikutnya biasa disebut *web indexing*. Dalam implementasinya *index system* dirancang dari penggabungan beberapa cabang ilmu antara lain ilmu bahasa, psikologi, matematika, informatika, fisika, dan ilmu komputer.

Tujuan dari penyimpanan data berupa indeks adalah untuk performansi dan kecepatan dalam menemukan informasi yang relevan berdasarkan inputan user. Tanpa adanya indeks, search engine harus melakukan pengecekan terhadap setiap dokumen yang ada didalam basis data. Hal ini tentu saja akan membutuhkan proses sumber daya yang sangat besar dalam proses komputasi. Sebagai contoh, indeks dari 10.000 dokumen dapat diproses dalam waktu beberapa detik saja, sedangkan penulusuran secara berurutan setiap

kata yang terdapat di dalam 10.000 dokumen akan membutuhkan waktu yang berjam lamanya. Tempat tambahan mungkin akan dibutuhkan di dalam computer untuk penyimpanan indeks, tapi hal ini akan terbayar dengan penghematan waktu pada saat pemrosesan pencarian dokumen yang dibutuhkan (Aqwam Rosadi K, 2010).

2.3 *String Matching*

String matching merupakan istilah dari sebuah pemrosesan pencocokan kata, sistem yang menyediakan berbagai kemampuan untuk memanipulasi teks. Seperti sistem proses teks *string*, yang mungkin merubah pendefinisian urutan dari huruf, angka, dan karakter khusus. Permasalahan ini bisa cukup besar (misalnya, dalam buku ini berisi lebih dari satu juta karakter), dan efisiensi algoritma mempunyai peranan penting dalam memanipulasi. Operasi yang mendasari pada operasi string adalah pola pencocokan, misal diberikan teks string panjang N dan pola panjang M, menemukan kemunculan pola dalam teks. Kebanyakan algoritma untuk masalah ini dapat dengan mudah diperluas untuk mencari semua kemungkinan terjadi pada pola dalam teks, karena mereka menandai teks secara berurutan, teks dapat kembali pada titik awal jika tidak ditemukan kata yang cocok (Sedgewick, 1983).

Algoritma pengolahan string mencakup berbagai metode untuk menangani dengan panjang karakter. Proses pencarian string mengarah ke pola pencocokan yang juga memanfaatkan metode *parsing* (Sedgewick, 1983).

2.3.1 Algoritma *Boyer Moore*

Algoritma *Boyer-Moore* adalah salah satu algoritma untuk mencari suatu string di dalam teks, dibuat oleh R.M boyer dan J.S Moore. Ide utama algoritma ini adalah mencari string dengan melakukan pembandingan karakter mulai dari karakter yang paling kanan dari string yang di cari. Algoritma ini dianggap sebagai lagoritma yang paling efisien pada aplikasi umum (yugianus, 2011).

2.3.2 Cara kerja umum Algoritma *Boyer-Moore*

Secara sistematis, langkah-langkah yang dilakukan algoritma *boyer moore* pada saat mencocokan string adalah :

1. Algoritma *Boyer moore* mulai mencocokan *pattern* pada awal teks
2. Dari kanan ke kiri, algoritma ini akan mencocokan karakter per karakter dengan karakter di teks yang bersesuaian, sampai salah satu kondisi terpenuhi :
    a. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (*missmatch*).
    b. Semua karakter di pattern cocok, kemudian algoritma akan memberitahukan penemuan 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.

Algoritma *boyer moore* memiliki karakteristik sendiri dibandingkan dengan algortima yang lain, karakteristik tersebut antara lain :

1. Melakukan perbandingan dari kanan ke kiri
2. Fase persiapan / *preprocessing* membutuhkan waktu O(m+σ)
3. Fase pencarian membutuhkan kompleksitas waktu O(n/m)
4. Pada kasus terburuk, sebanyak 3n karakter teks yang dibandingkan untuk *pattern* yang tidak berulang.

(yugianus, 2011)

2.3.3 Algoritma Tuned Boyer Moore

Algoritma *tuned boyer moore* adalah jenis pengembangan algoritma *boyer moore* yang lebih cepat dibandingkan dengan algoritma *boyer moore*. *Tuned boyer moore* ini hanya menggunakan fungsi perpindahan saat terjadinya ketidakcocokan untuk melakukan pergeseran.

Algoritma *tuned boyer moore* mempunyai keuntungan lebih efisien melakukan perpindahan ke karakter dan kondisi pada teks, algortima tunde boyer moore pada saat perpindahan akan memulai pengecekan dari akhir karakter pattern yang terdapat pada teks. Setelah kondisi teks dan pattern di dapatkan, maka algortima tuned boyer moore akan melakukan pengecekan terhadap pola yang ditemukan tersebut. Sama seperti boyer moore, algoritma tuned

boyer moore juga terdapat kondisi *preprocessing* dan kondisi pencarian

Bagian yang paling penting dari algoritma string matching adalah saat kondisi pengecekan kondisi karakter pada saat pencocockan. Untuk menghindari pengecekan yang terlalu sering, maka dimungkinkan untuk memberikan nilai pada karakter yang akan di cocockan sebelum melakukan pencocokan. Algoritma tuned boyermoore menggunakan bad-character shift untuk melakukan pergeseran karakter pattern pada teks *x*[*m*-1] pergeseran terus dilakukan sampai kondisi benar ditemukan. Kondisi ini akan disimpan dalam nilai dari sebuah kondisi *bmBc*[*x*[*m*-1]] pada variabel perpindahan (shift) dan kemudian digunakan untuk memberikan nilai pada kondisi *bmBc*[x[m-1]] sampai nilai bilangan 0. Hal ini diperlukan juga untuk memberikan nilai pada kejadian lain (*m*) yang mana nilai *m* akan dimasukkan pada kondisi *x*[*m*-1] sampai dengan akhir teks (*y*). Ketika kondisi *x*[*m*-1] ditemukan, maka perpindahan dilakukan dengan menngunakan kondisi *m*-1 menuju karakter yang lain. Perbandingan antara pola dan karakter teks dapat dilakukan dalam urutan apapun. Algoritma ini memiliki nilai *quadratic worst-case time* yang kurang baik tetapi memiliki perilaku praktis yang sangat baik. Hal ini dapat di cntohkan pada pseudocode berikut: (Charras ,Christian. Lecroq,Thierry)

```
void preBmBc(char *x, int m, int bmBc[]) {
   int i;

   for (i = 0; i < ASIZE; ++i)
      bmBc[i] = m;
   for (i = 0; i < m - 1; ++i)
      bmBc[x[i]] = m - i - 1;
}

void TUNEDBM (char *x, int m, char *y, int n)
{
int j,k,shift, bmBc[ASIZE]

/*Preprocessing*/
preBmBc (x, m, bmBc)
shift = bmBc[x[m-1]];
bmBc[x[m-1]]=0;
memset(y+n,x[m-1],m);
```

Gambar 2.2 Contoh pseudocode algortima *tuned boyer moore* (Charras ,Christian. Lecroq,Thierry)

```
/*Searching*/
j=0;
while (j < n){
   k=bmBc[y[j+m-1]];
   while (k! = 0){
      j +=k; k = bmBc[y[j+m-1]];
      j +=k; k = bmBc[y[j+m-1]];
      j +=k; k = bmBc[y[j+m-1]];
   }
   If(memcmp(x, y+j, m-1) == 0 && j < n)
   OUTPUT(j);
   j+= shift; /*shift*/
 }
}
```

Lanjutan Gambar 2.2 Contoh pseudocode
algortima *tuned boyer moore*(Charras ,Christian. Lecroq,Thierry)

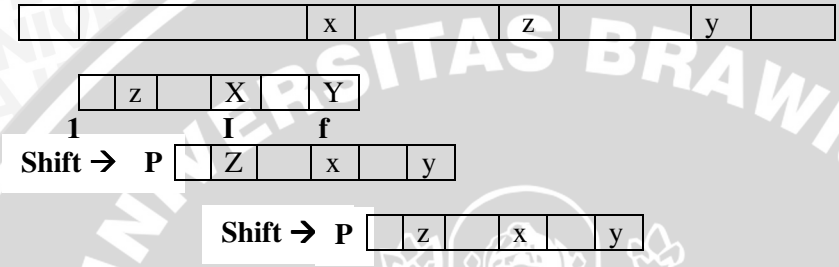
Pada algoritma ini, yang menjadi fokus adalah pada karakter terakhir dari sebuah teks dan mencoba untuk pola pergeseran dari karakter karakter teks (Adviser: R. C. T. Lee).

Algoritma *Tuned Boyer-Moore* dapat dianggap sebagai pengembangan yang efisien dari Horspool algoritma. Setiap perulangan dari *Tuned Boyer-Moore* algoritma dapat dibagi menjadi dua tahapan: lokasi terakhir karakter dan pencocokan kalimat atau karakter. Tahap pertama mencari kecocokan dengan menerapkan tiga *blind shift* (berdasarkan *classical bad character rule*) sampai diperlukan. Jika terdapat yang cocok pada tahap ini kemudian mencoba untuk mencocokkan sisa pola yang sesuai dengan karakter dari teks, melanjutkan dari kanan ke kiri. Model algoritma *boyer moore* dapat digambarkan seperti berikut, misalkan kita mempunyai sebuah teks (T) dengan panjang *n* dan *pattern* (P) dengan panjang m, maka model proses pencocokan dapat diilustrasikan sebagai berikut :

- Pendefinisian
  - Ts : karakter pertama dari sebuah string T.
  - Pl : karakter pertama dari pattern P.
  - Tj : karakter pada posisi *ith* dari string T.
  - Pi : karakter pada posisi *ith* dari *pattern* P.

- Pf : karakter terakhir dari *pattern* P.
- n : panjang teks (T).
- m : panjang *pattern* (P).

Pada $T_{S+m-1} \neq P_f$, pemindahan pola P ke kanan, P_i sama dengan T_{S+m}. Kita dapat beralih pola setidaknya (*m-i*) posisi kanan sampai $T_{S+m-1} = P_f$.



Gambar 2.3 Ilustrasi algortima tuned boyer moore
(Adviser: R. C. T. Lee)

Kecepatan algoritma *tuned boyer moore* lebih cepat dibandingkan *algortima boyer moore* dan *turbo boyer moore* jika pada karakter yang dicari tidak berulang-ulang, karena algoritma *tuned boyer moore* hanya membangun *table bad-character shift* dan tidak membangun *good-suffix shift* (Hermanto, 2007).

2.4 *Precision*

Performa dari identifikasi topik sebuah dokumen biasanya diukur menggunakan *Precision and Recall scores* (Hovy, 2003). Jika terdapat dua perbandingan hasil, hasil ringkasan sistem dan hasil ringkasan manusia, penilaian dilakukan pada seberapa dekat hasil ekstraksi topik yang dilakukan sistem dengan hasil ekstraksi yang dilakukan manusia. Pada skripsi ini akan dibandingkan hasil ekstraksi kata yang dilakukan sistem terhadap masing-masing atau jumlah frekuensi kemunculan kata. *Precision* adalah nilai yang menandakan seberapa besar sistem berhasil mengekstrak kalimat topik yang sesuai (Hovy, 2003).

Jika *correct* adalah jumlah kata yang berhasil diekstrak sistem dan sesuai dengan hasil ekstrak manusia, *wrong* adalah kata yang berhasil diekstrak sistem tetapi tidak diekstrak oleh manusia, dan

missed adalah kata yang diekstrak manusia tetapi tidak diekstrak oleh sistem, maka rumus *Precision* ditunjukkan pada persamaan 2.1 berikut (Hovy, 2003).

$$Precision = \frac{correct}{(correct + wrong)} \quad \textbf{2.1}$$

2.5 HyperText Markup Language (HTML)

HTML atau *HyperText Markup Language* merupakan salah satu format yang digunakan dalam pembuatan dokumen dan aplikasi yang berjalan dihalaman *web*. HTML adalah file text murni yang dapat dibuat dengan editor text apapun. Dokumen ini dikenal sebagai *web page*, dokumen HTML merupakan dokumen yang disajikan pada *web browser*. Ada dua cara dalam menulis sebuah dokumen HTML yang nantinya mejadi sebuah halaman browser. Yang pertama menggunakan HTML editor atau Web editor, dan yang kedua menggunakan teks editor biasa seperti menggunakan notepad.

Saat ini sudah banyak sekali paket aplikasi yang dapat digunakan untuk membuat halaman web secara WYSIWYG (What You See Is What You Get) seperti *FrontPage*,*Dreamweaver* dan lain sebagainya. Dalam penamaan sebuah dokumen yang akan ditampilkan pada web browser maka nama yang digunakan harus diakhiri dengan ektensi (.html) atau (.htm). Ekstensi dokumen HTML awalnya 3 karakter, adalah untuk mengakomodasi sistem penamaan dalam DOS. Dalam pemberian nama sebuah dokumen bersifat *case sensitive* sehingga dokumen dengan nama a.html akan berbeda dengan dokumen A.html.

Sebuah dokumen HTML disusun oleh beberapa Elemen atau lebih dikenal dengan komponen-komponen dasar. Elemen dapat berupa teks murni, atau bukan teks, atau keduanya. Elemen atau komponen tersebut misal head, body, paragraf, list dll.

Untuk menandai sebuah elemen dalam suatu dokumen HTML digunakan tag. Tag HTML terdiri dari sebuah kurung sudut kiri (<,lebih kecil), nama tag, kurung sudut kanan(>,lebih besar), contoh <H1>, tag pada umumnya berpasangan (misalnya <H1> dengan </H1>), tanda / pada tag pasangan memberikan tanda bahwa tag tersebut merupakan pembatas akhir elemen yang dibuka oleh tag awal (Masruro, Ahlihi,S.Kom, 2011).

Contoh dokumen HTML sederhana

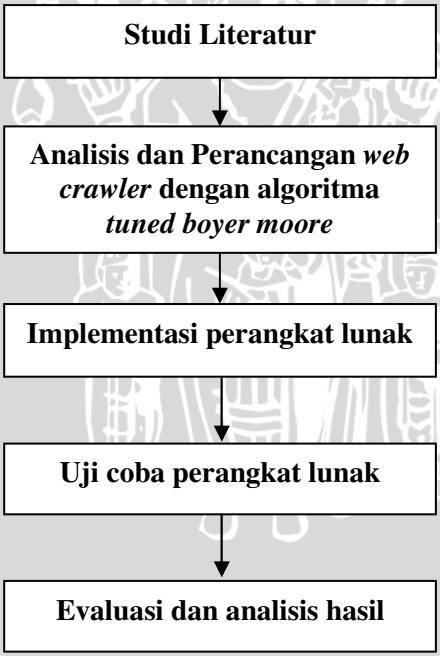
```
<!DOCTYPE html>
<html>
  <head>
    <title>'''Selamat Malam''' HTML</title>
  </head>
  <body>
    <h1>Selamat Belajar</h1>
    <p>Nama saya chandra!</p>
  </body>
</html>
```

Gambar 2.4 contoh pseudocode dokumen html sederhana

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metode dan tahap-tahap yang digunakan dalam pencarian kata menggunakan algoritma *tuned boyer moore* yang diimplementasikan pada *web crawler*. Diagram alir pembuatan perangkat lunak dapat dilihat pada gambar 3.1. Adapun tahapan pembuatannya adalah sebagai berikut:

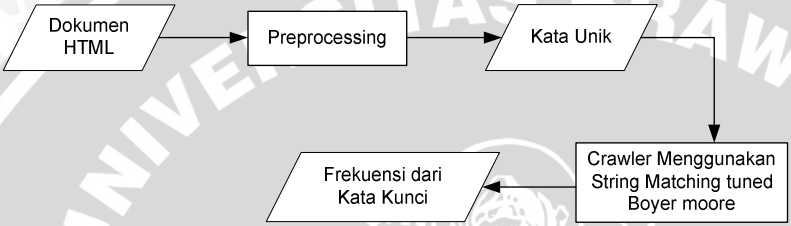
1. Melakukan studi literatur mengenai algoritma *tuned boyer moore* dan *web crawler*.
2. Menganalisis dan merancang implementasi *web crawler* dengan algoritma *tuned boyer moore* sebagai algoritma pencarian kata.
3. Implementasi perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Melakukan uji coba terhadap perangkat lunak.
5. Melakukan evaluasi hasil yang diperoleh dari uji coba perangkat lunak.



Gambar 3.1 Diagram alir pembuatan perangkat lunak

3.1 Perancangan Sistem Secara Keseluruhan

Secara umum, sistem akan memiliki fungsi untuk mencari kata pada dokumen HTML, kemudian dengan menggunakan algoritma *tuned boyer moore* sistem akan melakukan pencocokan *string* sehingga dapat di hitung frekuensi kemunculan string tersebut pada dokumen. Gambar 3.2 berikut akan menjelaskan skema aliran data dalam sistem.



Gambar 3.2 Aliran Data

3.2 Perancangan Proses

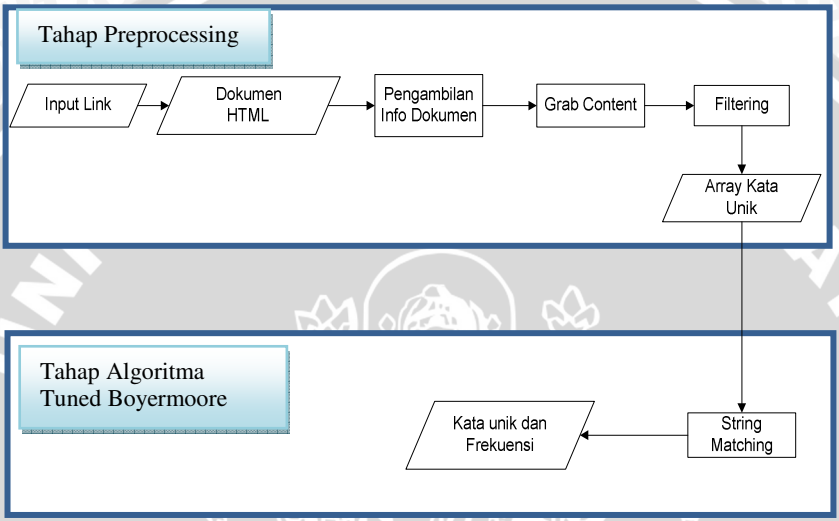
Perangkat lunak yang akan dibuat adalah aplikasi *crawler* dokumen HTML. Dokumen dalam hal ini merupakan *input* atau masukan dari situs berupa dokumen dalam bentuk HTML yang berekstensi *.html*. Kemudian setelah dokumen berhasil ditemukan, maka sistem akan mulai melakukan proses pengambilan kata kunci. Proses pengambilan kata kunci dilakukan oleh sistem diawali dengan membaca dokumen HTML mengambil semua kata yang unik pada dokumen tersebut dan menyimpannya di basis data.

Pertama kali proses yang dilakukan oleh sistem adalah membaca file HTML. Dari dokumen tersebut, sistem akan melakukan pengecekan terhadap dokumen tersebut sehingga akan didapatkan informasi berupa jumlah karakter, jumlah kata tersebut.

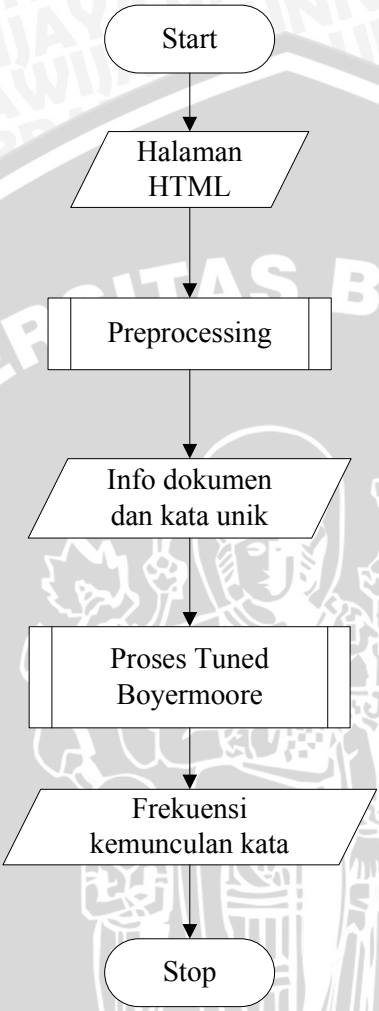
Setelah sistem mendapatkan informasi dari dokumen, sistem akan masuk ke tahap *preprocessing*. Pada tahap ini akan dilakukan beberapa proses, yaitu *filtering* (penghilangan karakter yang tidak penting.

Proses *filtering* adalah proses penghilangan tanda baca dan karakter yang kurang penting seperti spasi, koma dan sebagainya. Proses *Filtering* yang digunakan dalam sistem ini adalah

menggunakan fungsi yang ada pada *PHP* yaitu *regular expression*. Jika terdapat dalam fungsi PHP tersebut, karakter tersebut akan dihilangkan didapatkan daftar kata unik sebagai kata kunci dan disimpan di basis data.



Gambar 3.3 Gambar Arsitektur Sistem

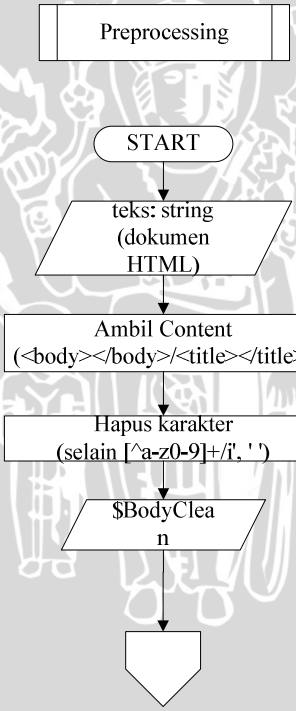


Gambar 3.4 *Flowchart* Proses Sistem

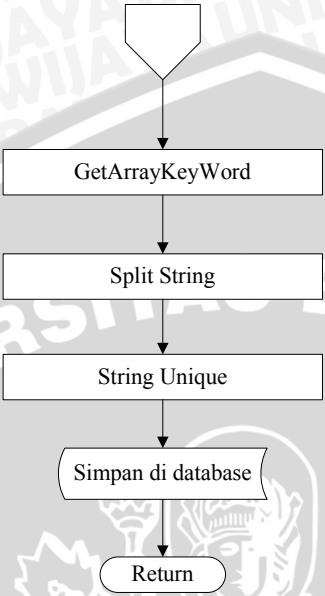
3.2.1 Algoritma Tuned Boyer Moore

3.2.1.1 Tahap Preprocessing

Pada tahap *preprocessing* terdapat beberapa proses yang dilakukan oleh sistem terhadap dokumen HTML. Proses tersebut adalah *filtering*, *case folding, tokenizing*. Proses *tokenizing* adalah proses memecah kalimat menjadi potongan kata. Sedangkan proses *case folding* adalah proses merubah huruf menjadi huruf kecil semua (*lowercase*). Proses *filtering* adalah proses penghilangan partikel-partikel karakter yang tidak penting sehingga didapatkan kata yang unik atau kata kunci. Proses *filtering* menggunakan fungsi yang terdapat di PHP. Pada program ini *case folding* dijadikan satu dengan proses penghilangan karakter yang tidak penting.



Gambar 3.5
Flowchart Proses *filtering*, *case folding, tokenizing*



Sambungan gambar 3.5
Flowchart Proses *filtering*, *case folding, tokenizing*

Proses berikutnya, kata kunci tersebut akan di lakukan proses pencocokkan dengan kata yang ada pada dokumen HTML tersebut untuk mengetahui frekuensi kemunculan dari kata tersebut pada sebuah dokumen HTML. Proses pencocokkan menggunakan algoritma *tuned boyer moore*, jika cocok maka akan ditambahkan frekuensi kemunculan kata tersebut dan menyimpannya di basis data.

Proses *indexing* dilakukan di *basis data* dengan mengurutkan kata kunci sesuai dengan urutan abjad. Hal ini dilakukan untuk memudahkan proses pencarian kata berikutnya.

3.2.1.2 Tahap Tuned Boyer Moore

Proses berikutnya pada halaman HTML yang lain disediakan sebuah halaman input untuk menginputkan kata yang akan dicari, hasil pencarian tersebut akan ditampilkan sesuai urutan berdasarkan frekuensi kemunculan kata yang paling sering. Kata yang ditemukan akan ditampilkan berdasarkan frekuensi kemunculan yang paling banyak.

Sebagaimana yang telah dituliskan sebelumnya, algoritma *tuned boyer moore* bertujuan untuk mencari kata pada teks sumber yang akan di *crawler*, untuk itu dalam pengerjaan skripsi ini dilakukan uji coba yakni:

- Teks dicoba dengan menggunakan beberapa pattern yang berasal dari *keyword.*

Pada uji coba tersebut, pencarian akan *filtering* teks guna memudahkan pencarian, setelah teks di filter barulah pencarian dilakukan sesuai dengan kata kunci dan hasil pencarian akan ditambahkan frekuensi kemunculan ke kata kunci. Dilakukan pula perhitungan terhadap lama waktu yang digunakan dan peringkat berdasarkan hasil kemiripan teks dengan *pattern*. Hal ini bertujuan untuk melihat seberapa besarkah waktu pencarian yang diperlukan dan peringkat kemiripan antara teks dengan *pattern* untuk pencarian menggunakan metode ini.

Kata kunci didapatkan dengan cara mengambil semua karakter unik yang terdapat pada teks menggunakan fungsi pada PHP, kemudian menyimpan kata kunci tersebut di *basis data* untuk dicocokkan dengan teks yang ada guna mencari frekuensinya.
Sebagai contoh:

- Teks = Optimalisasi Algoritma Pencarian Data Memanfaatkan Pohon Biner Terurut
- untuk *pattern*nya kita dapatkan dari kata kunci .
  Pattern = data

Pencocokan String terdiri atas langkah-langkah menemukan satu string atau lebih atas seluruh kejadian pada sebuah string (secara umum disebut pola string) di dalam teks. Pencarian dimulai dari kanan (akhir) teks.

Untuk melakukan pencarian dengan menggunakan algoritma *tuned boyer-moore* yang diimplemantasikan pada *web crawler*, tahap-tahapnya adalah:

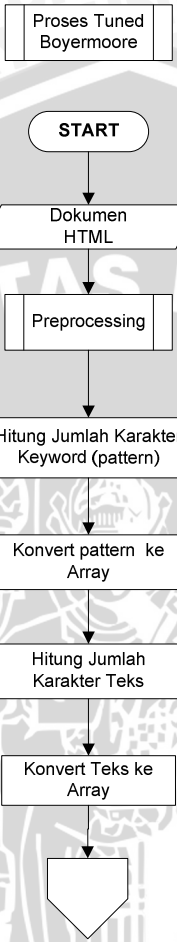
1) Proses pengumpulan informasi dari dokumen yang ada.
2) Sistem mencari kata kunci (kata unik) yang ada pada dokumen HTML.
3) Kata kunci dimasukkan ke dalam basis data untuk memudahkan proses pencarian.

4) Proses pencocokan teks dengan dengan menggunakan algoritma *tuned boyer moore*
5) Proses penambahan frekuensi kemunculan kata kunci pada teks dan disimpan di *basis data*.

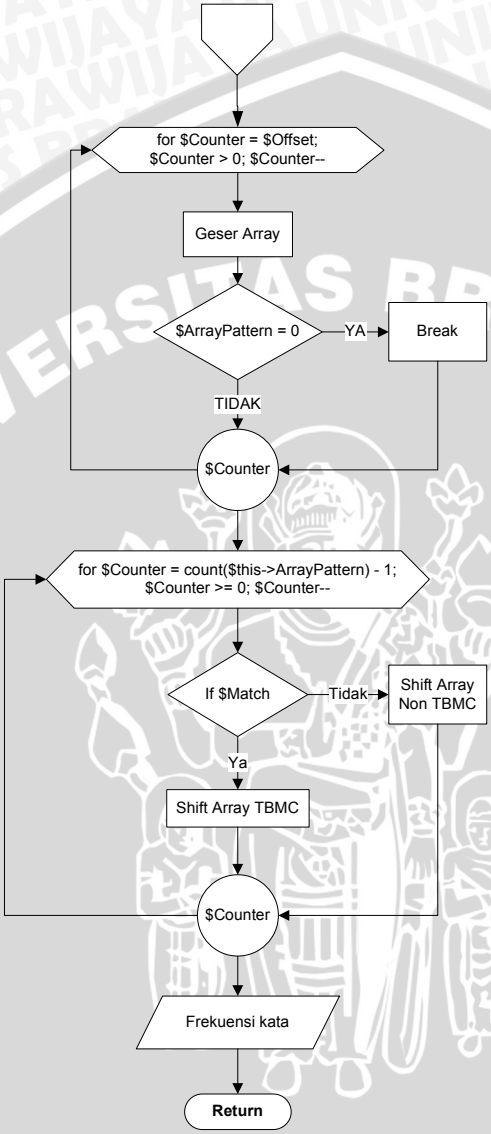
Pencarian dengan menggunakan algoritma *tuned boyer moore*, tahap-tahapnya adalah:

1) Proses dimulai dari sebuah data berupa teks dan *pattern*.
2) Proses inisialisasi teks dan *pattern*.
3) Proses pemberian *indeks* pada *pattern*.
4) Proses pencocokan teks dengan *pattern*, perbandingan pattern dengan teks dilakukan dari arah kanan ke kiri.
5) Jika terjadi kecocokkan, maka perbandingan akan dilanjutkan dengan karakter yang di sebelah kiri dari yang dibandingkan sampai ke karakter pertama dari pattern.
6) Jika terjadi ketidakcocokkan maka akan dilakukan pergeseran yang ditentukan oleh 2 fungsi pergeseran yaitu *bad character shift* dan *good suffix shift*.
7) Hasil dari pencarian akan ditampilkan sebagai informasi terhadap inputan *user*.

Adapun diagram alir proses menujukkan mekanisme proses dari aplikasi pencarian kata pada *web crawler* dengan *algoritma tuned boyer moore*.



Gambar 3.6
Flowchart Proses *Tuned Boyer Moore*



Sambungan gambar 3.6
Flowchart Proses *Tuned Boyer Moore*

3.3 Perancangan Uji Coba dan Evaluasi Hasil

3.3.1 Bahan Pengujian

Data yang diuji berupa dokumen HTML yang mempunyai ekstensi .html. Data diambil dari halaman tersebut ada lah pada bagian `<title></title>` dan `<body></body>`. Dokumen yang diuji akan dirubah sedemikian rupa untuk menguji berapa banyak kata dan frekuensi kemunculan kata pada dokumen tersebut menggunakan Algoritma *Tuned Boyer Moore*.

3.3.2 Tujuan Pengujian

Tujuan dari pengujian sistem untuk mengetahui hasil implementasi *algoritma tuned boyer more* untuk *web crawler* adalah sebagai berikut:

1. Memerikasa kesesuaian hasil implementasi sistem terhadap perancangan sistem apakah telah berjalan dengan baik atau tidak.
2. Menganalisa dan mengevaluasi hasil oleh sistem yang menggunakan algoritma tuned boyer moore.
3. Menganalisa hasil percobaan yaitu jumlah frekuensi kemunculan kata dengan waktu yang diperlukan untuk melakukan pencockan tersebut menggunakan algoritma tuned boyer moore.

3.3.3 Pengujian Implementasi Sistem

Dengan membandingkan jumlah karakter dan waktu proses algoritma tuned boyer moore pada masing-masing kasus. Berikut ini adalah rancangan tabel perhitungan manual:

Tabel 3.1 Data uji coba sistem

Link	Jumlah Frekuensi Sebenarnya	Jumlah Frekuensi Kata dengan Tuned Boyermoore

Tabel 3.2 Data uji coba terhadap Sistem

Link	Jumlah Frekuensi Sebenarnya	Rata-rata lama waktu pencocokan dengan tuned Boyermoore (dalam detik)

Tabel 3.3 Data uji coba terhadap Sistem

Link	Jumlah Frekuensi Kata dengan Tuned Boyer moore	Rata-rata lama waktu pencocokan dengan tuned Boyermoore (dalam detik)

3.3.4 Perancangan *Input* pada Sistem

Pada sistem crawler ini inputan yang diperlukan , yaitu:

1. Halaman HTML.
2. Bagian <title></title> dan <body></body> dari halaman HTML tersebut.
3. Algoritma yang akan digunakan untuk proses pencocokan kata.
4. Teks.
5. *Pattern* (Kata Kunci).

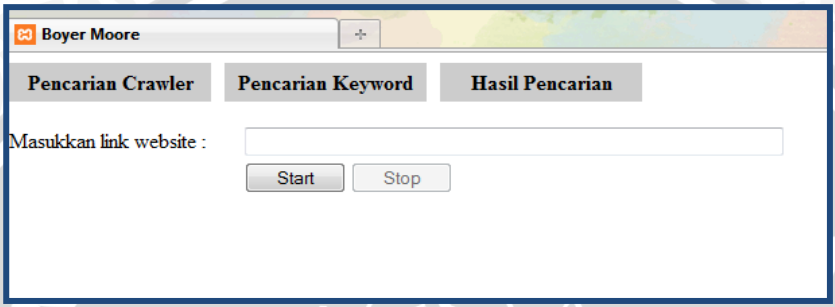
3.3.5 Perancangan *User Interface*

Sistem untuk pencocokankata ini berbasis web dengan menggunakan bahasa PHP dan *Javascript*. Sistem ini akan membaca inputan yang diberikan oleh user, yaitu link yang ingin di crawler untuk pengujian. Algoritma yang digunakan adalah algoritma *tuned boyer more*. Setelah data yang diiputkan selesai, sistem akan memproses dokumen tersebut sehingga akan diperoleh informasi-informasi dari dokumen tersebut. Kemudian sistem akan masuk ke tahap preprocessing yang terdiri dari *filtering*, *case folding, tokenizing*.

Gambar 3.7 adalah gambar rancangan *user interface* sistem. Menu-menu yang terdapat didalam sistem, antara lain:

1. Terdapat menu *Pencarian Crawler* yang digunakan untuk memasukkan *link website* yang ingin di *crawling*.

2. Terdapat menu pencarian *keyword* yang digunakan untuk mencari kata yang sudah di *crawling* oleh sistem
3. Terdapat menu hasil pencarian untuk melihat hasil crawling terhadap site yang di *crawl*



Gambar 3.7 *prototype user interface* sistem

3.3.6 Pengukuran Tingkat Kesesuaian

Setelah perangkat lunak berhasil dibuat dengan baik dan sesuai dengan implementasi yang ada. Maka perangkat lunak akan diuji seberapa besar perbandingan tingkat kesesuainnya antara frekuensi sebenarnya dengan frekuensi menggunakan algoritma *tuned boyer moore* juga waktu yang di perlukan untuk melakukan proses algoritma tersebut. Pengujian dilakukan dengan mencoba menggunakan data yang ada dan akan dibandingkan dan dideteksi seberapa besar lama waktu pencarian yang ada. Semakin sedikit waktu yang dibutuhkan untuk melakukan pencarian, maka perangkat lunak yang dihasilkan semakin baik. Semakin banyak frekuensi penemuan teks yang sesuai, maka perangkat lunak yang dihasilkan semakin baik tingkat keberhasilannya. Semakin sama frekuensi yang dikasilkan, maka perangkat lunak yang dihasilkan semakin baik tingkat keberhasilannya.

3.3.7 Contoh Penghitungan Manual

Berikut ini akan dicontohkan bagaimana cara kerja sistem dalam melakukan pencocokan kata antara *pattern* dengan teks. Hal pertama yang akan dilakukan sistem adalah menganalisa panjang teks, setelah

itu diambil kata kuci dari dokumen HTML, dilakukan pencockan *pattern* dengan teks yang berasal dari dokumen HTML.

Setelah itu sistem akan mencocokkan *pattern* dan teks tersebut berdasarkan urutan karakter pattern dan karakter teks yang terdapat di dalamnya. Hasil dari pencocokkan pattern dan teks tersebut dapat kita lihat pada berikut:

Tabel 3.4 Hasil Pencarian

No	Teks	Pattern	Banyak Ditemukan
1	A	B	2
2	C	D	5

Setelah didapatkan hasil dari pencocokan *pattern* dengan teks, maka sistem akan melakukan pengecekan terhadap hasil tersebut akan setiap elemen yang sama.

Skema dari pengecekan tersebut dapat dilihat dalam tabel dibawah ini:

Misal :

- Teks : ibu dan adik membeli kue di pasar
- Pattern : membeli kue

Tabel 3.5 Teks dan *Pattern*

T	i	b	u	d	a	n	a	d	i	k	m	e	m	b	E	l	i	k	u	e	d	i	p	a	s	a	r
P	m	e	m	b	e	l	i	k	u	e																	

Tabel 3.6 Pemberian nilai awal

shift	9	8	7	6	5	4	3	2	1	0
P	m	e	m	b	e	l	i	k	u	e

Tabel 3.7 nilai TBMC

Pattern TBMC	u	k	i	l	e	b	m	T
tbmc	1	2	3	4	5	6	7	10

Tabel 3.8 Proses Pencocokan

i	b	u	d	a	n	a	d	i	k	m	e	m	b	e	l	i	k	u	e	d	i	p	a	s	a	r
												1	2	3	4	5	6	7		1	2	3	4	5		
m	e	m	b	e	l	i	k	u	e	1	2								1							
		m	e	m	b	e	l	i	k	u	e															
									m	e	m	b	e	l	i	k	u	e								
										m	e	m	b	e	l	i	k	u	e							
											m	e	m	b	e	l	i	k	u	e						

Dalam tahap pengecekan pencocokan pattern dan teks ini, sistem akan mengabaikan teks yang tidak cocok dengan pattern.

Pada perhitungan berikutnya, di hitung pula nilai *precision* dari dokumen tersebut.sesuai dengan teori dan rumus pada bab sebelumnya, maka dapat dibuat contoh perhitungan manual sebagai berikut :

Tabel 3.9 Contoh data yang di perlukan untuk perhitungan *precision*

Content	Frekuensi Sebenarnya	Frekuensi Tuned Boyermoore	Correct	Wrong	Miss
ada	2	10	2	8	0
adalah	4	2	2	0	2
anjuran	6	3	3	0	3
anwar	1	0	0	0	1
bagi	1	1	1	0	0
baik	1	0	0	0	1
banyak	1	0	0	0	1
batas	4	3	3	0	1
beda	2	0	0	0	2

Dari data diatas di dapatkan :

- Jumlah correct = 11
- Jumlah wrong = 8
- Jumlah miss = 10

Dengan menggunakan rumus 2.1 dapat diketahui nilai *precision* dan seperti berikut.

$$Precision = \frac{11}{(11+8)} = \frac{11}{19} = 0{,}58$$


UNIVERSITAS BRAWIJAYA

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

Sebelum melakukan implementasi sistem, beberapa syarat harus disiapkan untuk memenuhi kebutuhan dari program yang akan implementasikan baik dari segi perangkat keras (*hardware*) maupun perangkat lunak (*software*) komputer.

3.1 *Lingkungan Implementasi*

Lingkungan implementasi meliputi lingkungan perangkat keras serta lingkungan perangkat lunak.

3.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem pembuatan perangkat lunak *web crawler* adalah laptop dengan spesifikasi:

1. Prosesor Intel(R) Core i3 – 2.4 GHz.
2. RAM 2048 MB.
3. *Harddisk* dengan kapasitas 500 GB.
4. Monitor.
5. Keyboard.

3.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem *web crawler* ini adalah :

1. Sistem Operasi *Microsoft Windows 7 Home Basic*
2. *PHP* sebagai bahasa pemrograman.
3. *Mozilla* sebagai *Browser*.
4. MySql sebagai basis data.
5. XAMPP 1.6.3
6. PHP Version 5.2.3
7. Apache 2.0
8. Heidi SQL 4.0
9. PHP Designer 7.0

3.2 *Implementasi Program*

Berdasarkan perancangan perangkat lunak pada subbab 3.2 maka pada subbab ini akan dibahas mengenai implementasi dari perancangan tersebut.

3.2.1 Pengumpulan Data

Tahapan pertama adalah pengumpulan data. Contoh data yang didapatkan adalah sebagai berikut **:**

Tabel 4.1 Contoh data uji

Judul	**Penulis**	**Jenis Tulisan**	**Deskripsi**
Penerapan Kalori Konveksi Alami Dan Konveksi Paksa Pada Sistem pengeringan Dan Penyimpanan JagungSkala Komersial Di Pedesaan DI PIEBESUN	Moch. Muslech Moestadjab, Nur Komar,Usman Efendi	Minor Thesis	Tujuan dari kegiatanadalah untuk menentukan perawatan yang murah efisien, serta yang mudah penyimpanm jagung di harapkan dengan tehik ini dapat mencegah kerusakan jagung selama penyimpanan sebagai akibat dari iklim dan hama.

Hubungan antara Obesitas dengan Tingkat Perkembangan Anak Usia Prasekolah (4 - 6 Tahun) di TK Plus Al Kautsar Malang.	Hani Riska Ariyanti		Obesitas merupakan permasalahan yang akhir-akhir ini muncul di dunia, bahkan World Health Organisation (WHO) telah mendeklarasikannya sebagai epidemik global.

Total data yang diperoleh adalah : 12208. Terdiri dari 7 kelompok,, yaitu jurnal, *minor thesis, thesis,* disertasi, *course material, researh report, undefined*. Data merupakan data *dummy* dari basis data perpustakaan Universitas Brawijaya.

Tabel 4.2 Daftar rincian jumlah data uji

Kategori	**Jumlah**
Minor Thesis	2000
Research Report	1500
Course Material	48
Disertasi	3
Journal	876
Thesis	7
Undefined	151

Setelah data dikumpulkan data dimasukkan kedalam basis data untuk dibuat dokumen HTML. Proses berikunya adalah *crawling*, proses *crawling* dilakukan oleh *search engine* guna mengumpulkan data yang ada pada HTML tersebut.

3.2.2 Struktur Data

Dalam penerapan sistem ini, dibentuk struktur data berupa kelas-kelas utama yang menerapkan tiap proses dalam sistem. Kelas-

kelas tersebut antara lain kelas *crawler*, kelas *boyermoore*, kelas *helper*, dan kelas *site*.

1. Kelas Crawler

Kelas *crawler* digunakan untuk memperoleh informasi dokumen HTML yang nantinya akan digunakan untuk mengolah data pada proses inti. Berikut diagram UML kelas crawler yang disajikan dalam bentuk tabel :

Tabel 4.3 Tabel *Class Crawler*

CLASS	FUNCTION
class Crawler	
	function Crawler($Data)
	function InsertSite($LinkSource)
	function SaveLink()
	function GetSiteByLinkSource($LinkSource)
	function GetLinkPath()
	function GetContent($Link)
	function GetTitleFromContent($Content)
	function GetArrayLink($Content)
	function IsInternalLink($LinkSource, $Link)
	function SaveSubLink($ArraySubLink)
	function SaveLinkContent()
	function GetLinkByUrl($Url)
	function CleanInvalidChar($Content)
	function GetArrayKeyWord($Content)
	function SaveKeyWord($ArrayKeyWord)
	function GetKeyWordIdByKeyWord($KeyWord)
	function GetArrayBoyerMoore($Content, $ArrayKeyWord)
	function GetNextLink()
	function Write($FileLocation, $FileContent)

Pada kelas *crawler* ini terdapat beberapa fungsi, yaitu konstruktor kelas crawler yang berisi deklarasi *time, link source*, dan *sub link.* Fungsi *InsertSite* merupakan fungsi yang digunakan untuk menyimpan *site* yang berasal dari *link source*, setelah itu dicari kembali *link/sub link* yang terdapat pada situs tersebut dan menyimpannya di basis data, menggunakan fungsi *SaveLink.*

Pengecekan *sub link* menggunakan fungsi *GetArrayLink* untuk mengetahui apakah terdapat *sub link* pada halaman situs tersebut. Setelah informasi di dapatkan, digunakan fungsi *SaveLinkContent* untuk menyimpan sub link yang terdapat pada halaman HTML tersebut, lalu *content* dari *link source* dipisahkan menggunakan *GetArrayKeyWord* untuk mendapatkan *unique keyword* yang kemudian di simpan ke basis data menggunakan *SaveKeyWord.*

2. Kelas BoyerMoore

Kelas BoyerMoore digunakan untuk menjalankan algoritma *Tuned Boyer moore* pada pengujian ini. Berikut tabel kelas BoyerMoore yang disajikan dalam bentuk tabel :

Tabel 4.4 Tabel *Class BoyerMoore*

Class	**FUNCTION**
class BoyerMoore	
	function BoyerMoore($Text, $Pattern)
	function ConvertStringToArray($String)
	function ArrayShift($Array)
	function ArrayTbmc($Array)
	function Calculate($Offset)

Pada kelas *BoyerMoore* ini terdapat beberapa fungsi, yaitu konstruktor kelas *BoyerMoore* yang berisi deklarasi *teks, pattern, array teks, array pattern, teks shift, pattern shift* dan *array TBMC*. Fungsi *ConvertStringToArray* merupakan fungsi yang digunakan untuk merubah *string* yang berasal dari halaman HTML menjadi

array, setelah itu *array* tersebut digunakan untuk proses pencocokan tapi terlebih dahulu *array* tersebut dilakukan proses *indexing* menggunakan fungsi *ArrayShift* untuk mempermudah pencocokan. Pada kelas ini terdapat pula fungsi *ArrayTbmc* yang berfungsi untuk memberikan *indeks* pada kata kunci yang kemudian di seleksi sesuai dengan aturan tuned boyer moore. Setelah fungsi *ArrayTbmc* mendapatkan *array* yang sesuai dengan aturan algoritma *tuned boyermoore*, maka dijalankan fungsi *Calculate* untuk menjalankan algoritma *tuned boyermoore* dalam hal pencocokan kata untuk menemukan frekuensi kemunculan kata tersebut.

3. Kelas Helper

Kelas *Helper* digunakan untuk memanipulasi dan memodifikasi string yang didapatkan dari halaman HTML. Berikut diagram UML kelas *helper* yang disajikan dalam bentuk tabel :

Tabel 4.5 Tabel *Class helper*

CLASS	**FUNCTION**
class Helper	
	function StripArray($Array)
	function EscapeString
	function GetOption($OptAll, $ArrayOption, $Selected)
	function MoneyFormat($Value)

Terdapat fungsi *StripArray* yang berguna untuk menghapus tanda baca "/" yang kemudian dirubah menjadi *array* untuk mempermudah pengolahan data. Pada kelas *helper* terdapat pula fungsi *EscapeString* untuk memberikan eksepsi string pada array sebelum melakukan *query* pada basis data,eksepsi ini pada karakter tertentu yang mempunyai kemungkinan menyebabkan kegagalan pengeksekusian *query*. Fungsi *GetOption* untuk membuat option html dari *array input*, fungsi *ArrayToJSON* untuk mengubah *array* ke bentuk *json* yang nanti akan digunakan oleh aplikasi.

4. Kelas Site

Kelas *site* digunakan untuk manajemen menu dan hasil pencarian. Berikut diagram UML kelas *site* yang disajikan dalam bentuk tabel :

Tabel 4.6 Tabel *Class site*

CLASS	FUNCTION
class site	
	function GetMenu()
	function GetArray()
	function GetSubLinkBySiteID($Param)
	function GetContentByArrayPage($ArraySubLink, $PageActive, $PageTotal)
	function GetKeyWordByLinkContentID($Param)
	function GetKeyWordByContent($KeyWord)
	function GetSubLinkByKeyWord($Param)

Terdapat fungsi *GetMenu* yang berguna untuk menampilkan menu yang ada pada halam ini, antara *crawler*, pencarian *keyword* (kata kunci) dan hasil pencarian keseluruhan . Pada kelas *site* terdapat pula fungsi *GetArray* untuk menampilkan daftar site pada array yang ada pada basis data, seterlah itu dengan menggunakan fungsi *GetSubLinkBySiteID* akan menampilkan *sub link* dari *link* yang kita inginkan. Fungsi *GetContentByArrayPage* mengatur halaman yang akan ditampilkan beserta infonya, untuk menampilkan prekuensi kemunculan kata kunci serta iterasinya digunakan fungsi *GetKeyWordByLinkContentID*, fungsi *GetKeyWordByContent* dan *GetSubLinkByKeyWord* digunakan untuk menyertakan asal content dan sublink dari kata kunci yang ditampilkan.

3.2.3 Tahap Preprocessing

Pada tahap preprocessing ini akan dilakukan pendeklarasian konstruktor kelas *crawler* yang berisi deklarasi *time, link source*, dan *sub link* yang terdapat pada *souce code* 4.1.

```
function Crawler($Data) {
   $this->Time = 0;
   $this->Title = '';
   $this->LinkNext = '';

   $this->Link = $Data['Link'];
```

```
    $this->LinkSource = $Data['LinkSource'];
    $this->IsStartSite = $Data['IsStartSite'];

     if (empty($Data['Link'])) {
     return;
      }

      if ($this->IsStartSite == 1) {
        $this->InsertSite($this->LinkSource);
        }
   $this->Site=$this->
         GetSiteByLinkSource($this->LinkSource);
   $this->GetLinkPath();
   $this->Content = $this->GetContent($this->Link);
   $this->ContentClean = $this->
         CleanInvalidChar($this->Content);
   $this->Title=$this->
         GetTitleFromContent($this->Content);

   // Get / Save Current Link
   $this->SaveLink();

   // Get & Save SubLink
   $this->ArraySubLink=$this->
         GetArrayLink($this->Content);
   $this->SaveSubLink($this->ArraySubLink);
   $this->SaveLinkContent();

   // Get & Save KeyWord
   $this->ArrayKeyWord=$this->
         GetArrayKeyWord($this->Content);
   $this->SaveKeyWord($this->ArrayKeyWord);
   $this->ArrayBoyerMoore=$this->GetArrayBoyerMoore
          ($this->ContentClean,$this->ArrayKeyWord);

   // Get Next Link
   $this->LinkNext = $this->GetNextLink();
     }
```

Source code 4.1 deklarasi Fungsi class *Crawler()*

Pada proses berikutnya situs yang berasal dari *link source* disimpan ke basis data menggunakan fungsi *InsertSite($LinkSource)*, *SaveLink()* akang mengurai *link*, *title*, dan *full text* yang ada seperti terdapat pada *source code* 4.2 dan *source code* 4.3.

```
function InsertSite($LinkSource) {
   $Content = $this->GetContent($LinkSource);
   $InsertQuery = "INSERT INTO ".SITE."
      (SiteID, Url, Judul, Deskripsi, Tgl_Index)
      VALUES (NULL, '$LinkSource', '
      ".$this->Title."', '', '".date('Y-m-d')."')";
    $InsertResult = mysql_query($InsertQuery);
    }
```

Source Code 4.2 Fungsi *InsertSite($LinkSource)*

```
function SaveLink() {
   $SubLink = array(
   'Link' => $this->Link,
   'Title' => $this->Title,
   'FullText' => $this->Content
   );
     $ArraySubLink[] = $SubLink;
     $this->SaveSubLink($ArraySubLink);
    }
```

Source Code 4.3 Fungsi *SaveLink()*

Selanjutnya ditelusuri untuk mendapatkan *link/sub link* dan juga isi yang ada pada halaman yang ditemukan tersebut menggunakan fungsi *GetContent()* yang terdapat pada *source code* 4.4 yang kemudian pada fungsi ini terjadi pula fungsi *case folding* yaitu merubah seluruh karakter menjadi huruf kecil, bukan kapital.

```
function GetContent($Link) {
   $Content = file_get_contents($Link);
   $Content = trim(strtolower($Content));
   return $Content;
    }
```

Source Code 4.4 Fungsi *GetContent()*

Content yang didapatkan tersebut kemudian di proses untuk mendapatkan *title* atau judul dari *content* tersebut dan juga merubah *link* tersebut menjadi *array link*, proses ini menggunakan fungsi

GetTitleFromContent() pada *source code* 4.5 dan Fungsi *GetArrayLink()* pada *source code* 4.6.

```
function GetTitleFromContent($Content) {
   preg_match('/title>([^<]+)
              <\/title>/i',$Content, $Match);
   $Title = (isset($Match[1])) ?
            mysql_escape_string($Match[1]) : '';
   $Title = ucwords($Title);
   return $Title;
    }
```

Source Code 4.5 Fungsi *GetTitleFromContent()*

```
function GetArrayLink($Content) {
   preg_match_all('/a href=[\'\"]?
       ([a-z0-9\.\,\?\=\&\\/\_ ]+)[\'\"]?>/i',
       $Content, $Match);
   $MatchLink = (isset($Match[1])) ?
                $Match[1] : array();
  foreach ($MatchLink as $Key => $Element) {
   $ParseLink = parse_url($Element);

     // Direct Link with relative path
   if ($Element[0] == '/') {
     $MatchLink[$Key] = $this->LinkHost.$Element;
     }

     // Direct Link without protocol HTTP
      else if (!isset($ParseLink['scheme'])) {
        $MatchLink[$Key] = $this->LinkPath.$Element;
            }
        }

  foreach ($MatchLink as $Key => $Link) {
    $LinkResult = preg_replace('/\&phpsessid\=
                 [a-z0-9]+/i', '', $Link);
    $MatchLink[$Key] = preg_replace
              ('/localhost\/berita_edit\i',
              'localhost/berita_edit/', $LinkResult);
        }
    return $MatchLink;
    }
```

Source Code 4.6 Fungsi *GetArrayLink()*

Proses berikutnya pada content dicari apakah terdapat *link external*, jika terdapat *link external* maka akan diberikan tanda di basis data pada *link* yang ada pada content tersebut tetapi tidak ditelusuri, menggunakan fungsi *IsInternalLink()* seperti pada *source code* 4.7.

```
function IsInternalLink($LinkSource, $Link) {
 $IsInternalLink = '1';
 $Link = parse_url($Link);

    }
```

```
  if (!empty($LinkSource)) {
   $ArrayLinkSource = parse_url($LinkSource);
   $IsInternalLink = ($ArrayLinkSource['host']
         == $Link['host']) ? '1' : '0';
       }
  return $IsInternalLink;
```

Source Code 4.7 Fungsi *IsInternalLink()*

Setelah isi didapatkan maka terdapat kemungkinan adanya karakter-karakter yang tidak *valid*, maka karakter-karakter tersebut dihilangkan guna mempermudah proses pengindeksan kata, penghilangan *invalid* karakter ini menggunakan fungsi *CleanInvalidChar()* fungsi ini sering disebut proses *filtering* seperti pada *source code* 4.8.

```
function CleanInvalidChar($Content) {
  $BodyStart = strpos($Content, '<body');
  $BodyEnd = strpos($Content, '</body') - $BodyStart;
  $BodyContent = substr($Content,
                $BodyStart, $BodyEnd);
  $BodyClean = trim(strip_tags($BodyContent));
  $BodyClean = preg_replace('/[^a-z0-9]+/i', ' ',
              $BodyClean);
  return $BodyClean;
   }
```

Source Code 4.8 Fungsi *CleanInvalidChar()*

Setelah di dapatkan karekter yang *valid* maka berikutnya dengan menggunakan fungsi *GetArrayKeyWord()* seperti pada *source code* 4.9 maka akan dicari kata unik pada teks untuk mendapatkan dan dipergunakan proses berikutnya yang biasa disebut proses *tokenezing*.

```
function GetArrayKeyWord($Content) {
  $ContentClean = $this->CleanInvalidChar($Content);
  $ArrayKeyWord = explode(' ', $ContentClean);
  $ArrayKeyWord = array_unique($ArrayKeyWord);
  return $ArrayKeyWord;
    }
```

Source Code 4.9 *Fungsi GetArrayKeyWord()*

Selanjutnya data kata unik itu di masukkan ke basis data menggunakan fungsi *SaveKeyWord()* seperti pada *source code* 4.10. untuk selanjutnya digunakan untuk perhitungan frekuensi kemunculan kata tersebut.

```
function SaveKeyWord($ArrayKeyWord) {
  foreach ($ArrayKeyWord as $KeyWord) {
   $KeyWord = trim($KeyWord);
     if (!empty($KeyWord)) {
      $InsertQuery = "INSERT INTO ".KEYWORD."
                     (Content)VALUES ('$KeyWord') ";
      $InsertResult = mysql_query($InsertQuery);
            }
       }
    }
```

Source code 4.10 Fungsi SaveKeyWord()

Setelah didapatkan kata tersebut maka berikutnya akan dirubah kedalam bentuk *array* yang sesuai dengan struktur *tuned boyermoore* menggunakan fungsi *GetArrayBoyerMoore()* seperti pada *source code* 4.11. Pada fungsi ini hanya menguraikan kata dalam bentuk *array*, kemudian *array* ini akan digunakan algoritma *tuned boyermoore* untuk dilakukan operasi algoritma tersebut yang kemudian hasil dari operasi algoritma tersebut dimasukkan ke basis data sebagai frekuensi kemunculan kata tersebut.

```
function GetArrayBoyerMoore($Content, $ArrayKeyWord)
{
 $Counter = MAX_KEYWORD;
 $ArrayResult = array();
 $CurrentDate = date("Y-m-d");
 // Insert all keyword before proccess
 foreach ($ArrayKeyWord as $KeyWord) {
  $KeyWordID = $this->
               GetKeyWordIdByKeyWord($KeyWord);
  $SerialIteration = serialize('');
  $InsertQuery = "INSERT INTO ".LINK_KEYWORD."
       (KeywordID, LinkID, Tgl_Index, Time,
        Iteration, Frekuensi) VALUES
       ('$KeyWordID', '".$this->LinkID."','"
       .$CurrentDate."', '0',
        '$SerialIteration', '0')";
  $InsertResult = mysql_query($InsertQuery);
        }
$TimeStart = microtime(true);
  foreach ($ArrayKeyWord as $KeyWord) {
    $KeyWordID = $this->
                 GetKeyWordIdByKeyWord($KeyWord);
    $Content = trim($Content);
    $KeyWord = trim($KeyWord);
    $KeyWordTimeStart = microtime(true);

    if (!empty($Content) && !empty($KeyWord)) {
     $BoyerMoore = new BoyerMoore($Content,
                   $KeyWord);
     $ArrayResult[$KeyWord]['ArrayIteration'] =
                $BoyerMoore->ArrayIteration;
     $ArrayResult[$KeyWord]['MatchValue'] =
                $BoyerMoore->MatchValue;
     } else {
     $ArrayResult[$KeyWord]['ArrayIteration'] = '';
     $ArrayResult[$KeyWord]['MatchValue'] = 0;
           }
     $KeyWordTimeEnd = microtime(true);
     $KeyWordTime =
                $KeyWordTimeEnd - $KeyWordTimeStart;
     $SerialIteration =
                  serialize($ArrayResult[$KeyWord]
                  ['ArrayIteration']);
     $UpdateQuery = "UPDATE ".LINK_KEYWORD."
       SET Time = '$KeyWordTime',
       Iteration = '$SerialIteration',
```

```
        Frekuensi = '".$ArrayResult[$KeyWord]
         ['MatchValue']."
         'WHERE KeywordID = '$KeyWordID'
         AND LinkID = '".$this->LinkID."'
         AND Tgl_Index = '$CurrentDate'";
      $UpdateResult = mysql_query($UpdateQuery)
         or die(mysql_query());
      $Counter--;
      if ($Counter <= 0) {
        break;
            }
        }

    $TimeEnd = microtime(true);
    $this->Time = $TimeEnd - $TimeStart;
     return $ArrayResult;
    }
```

Source Code 4.11 Fungsi *GetArrayBoyerMoore*()

3.2.4 Tahap Tuned Boyer moore

Tahap selanjutnya adalah melakukan proses pengolahan teks menggunakan algoritma tuned boyermoore. Untuk melakukan proses perhitungan ini, digunakan kelas *boyermoore* yang terdapat pada file *boyermoore.php*. Penggunaannya adalah dengan melakukan deklarasi *variabel* beserta fungsi-fungsi yang akan digunakan untuk proses algoritma tuned boyermoore. Deklarasi variabel digambarkan pada *source code* 4. 12.

```
class BoyerMoore {
    var $ArrayText = null;
    var $ArrayPattern = null;
    var $TextCount = null;
    var $PatternCount = null;
    var $MatchValue = null;
    var $Iteration = null;
    var $ArrayIteration = null;
    var $Shift = 0;
```

Source Code 4.12 deklarasi array

Pada tahap ini terdapat *function BoyerMoore($Text, $Pattern)* yang digunakan untuk menghitung panjang teks, panjang *pattern*, merubah teks dan *pattern* menjadi *array*, *array shift*, *pattern shift*, dan *pattern tuned boyermoore* seperti pada *source code* 4.13.

```
function BoyerMoore($Text, $Pattern) {
 $Text = preg_replace('/ /i', '', $Text);
 $Pattern = preg_replace('/ /i', '', $Pattern);
 $this->TextCount = strlen($Text);
 $this->PatternCount = strlen($Pattern);
 $this->ArrayText =
   $this->ConvertStringToArray(strtolower($Text));
 $this->ArrayPattern =
   $this->ConvertStringToArray(strtolower($Pattern));
 $this->ArrayTextShift =
   $this->ArrayShift($this->ArrayText);
 $this->ArrayPatternShift =
   $this->ArrayShift($this->ArrayPattern);
 $this->ArrayPatternTbmc =
   $this->ArrayTbmc($this->ArrayPatternShift);
 // set default shift
 $this->Shift =
   $this->ArrayPatternTbmc
   [$this->ArrayPattern[count($this->ArrayPattern)
   -1]] == 0 ?
 $this->ArrayPatternTbmc["T"] :
   $this->ArrayPatternTbmc[$this->
   ArrayPattern[count($this->ArrayPattern)-1]];

$this->ArrayPatternTbmc[$this->
   ArrayPattern[count($this->ArrayPattern)-1]] = 0;
          $this->MatchValue = 0;
        $this->Iteration = 1;
        $this->ArrayIteration = array();
        $this->Calculate(0);
    }
```

Source Code 4.13 fungsi *boyermoore()*

Pada sebuah array dimungkin kan juga terdapat duplikasi *array* pada saat pemrosesan, maka perlu dihilangkan duplikasi tersebut menggunakan *function ArrayTbmc($Array)* seperti pada *source code* 4.14.

```
function ArrayTbmc($Array) {
 $ArraySource = $Array;
 $nArr = count($ArraySource);
 // Remove First Index if there is a duplicate value
   $Temp = $Array;
   unset($Temp[0]);
   $FirstIndex = $Array[0];

    if (in_array($FirstIndex, $Temp)) {
      unset($Array[0]);
       }

       // Reverse Order of Array
      $Temp = $Array;
      ksort($Temp);
      $Counter = 0;
       $Result = array();
       foreach ($Temp as $Key => $Value) {
           if (!isset($Result[$Value])) {
               $Result[$Value] = $Key;
               $Counter++;
           }
       }
       $Result['T'] = $nArr;
            return $Result;
   }
```

Source Code 4.14 fungsi *arraytbmc()*

Proses selanjutnya, dari data yang ada akan di proses menggunakan algoritma tuned boyermoore sesuai dengan perhitungan manual yang ada, untuk proses algoritma tuned boyermoore ini terdapat pada *function Calculate($Offset)* yang ada pada *source code* 4.15.

```
function Calculate($Offset) {
 //echo $Offset.'<br />';
        $Match = true;
        $ArrayText = $this->ArrayText;
        $ArrayPattern = $this->ArrayPattern;

        // Insert Array to Pattern
 for ($Counter = $Offset; $Counter > 0; $Counter--)
   {
    array_unshift($ArrayPattern, '');
    }
```

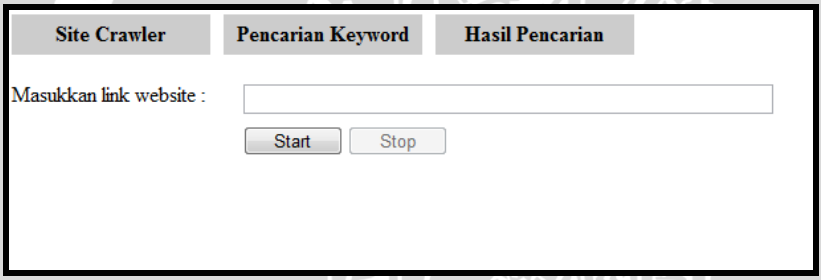
```
        $ShiftTbmc = 0;
        $MatchKey = count($ArrayPattern) - 1;

        if (!isset($ArrayText[$MatchKey])
       ||$ArrayText[$MatchKey]!=
       $ArrayPattern[$MatchKey]) {
         $ShiftTbmc =
          (isset($this->ArrayPatternTbmc
            [$ArrayText[$MatchKey]])) ?
            $this->ArrayPatternTbmc
            [$ArrayText[$MatchKey]] :
            $this->ArrayPatternTbmc['T'];
               $Match = false;
               }
               else {
       for ($Counter =
         count($this->ArrayPattern) - 1;
         $Counter >= 0;
         $Counter--)
         {
       Echo $ArrayText[$MatchKey].' != '
      .$ArrayPattern[$MatchKey].'<br />';
       if (!isset($ArrayText[$MatchKey]) ||
         $ArrayText[$MatchKey] !=
         $ArrayPattern[$MatchKey]) {
               $Match = false;
               break;
                              }
               $MatchKey--;
                     }
               }

        if ($Match) {
$KeyTbmc = $ArrayText[count($ArrayPattern) - 1];


$this->MatchValue++;
$this->ArrayIteration[] = $this->Iteration;
               echo '++<br />';
        } else {
               echo '--<br />';
        }
$ShiftTbmc = $ShiftTbmc==0?$this->Shift:$ShiftTbmc;
//echo "Shift =". $ShiftTbmc."<br/>";
$NextOffset = $Offset + $ShiftTbmc;
$NextOffestPattern =
```

```
   $NextOffset + count($this->ArrayPattern);
//if ($NextOffestPattern <=
    $this->TextCount && $this->LimitRecursion > 0) {
    if ($NextOffestPattern <= $this->TextCount) {
        $this->LimitRecursion--;
        $this->Iteration++;
        $this->Calculate($NextOffset);
       }
    }
```

Source Code *4.15 fungsi* calculate()

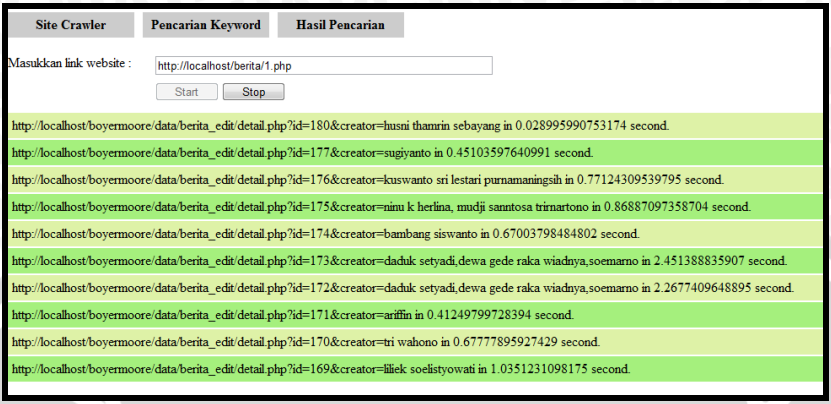
3.3 *Implementasi Antarmuka*

Berdasarkan rancangan antarmuka pada sub bab 3.3 maka dihasilkan antarmuka seperti berikut ini. Pada form utama terdapat beberapa menu antara lain "*site crawler„ "Pencarian keyword", "hasil pencarian"* seperti pada gambar 4.1.



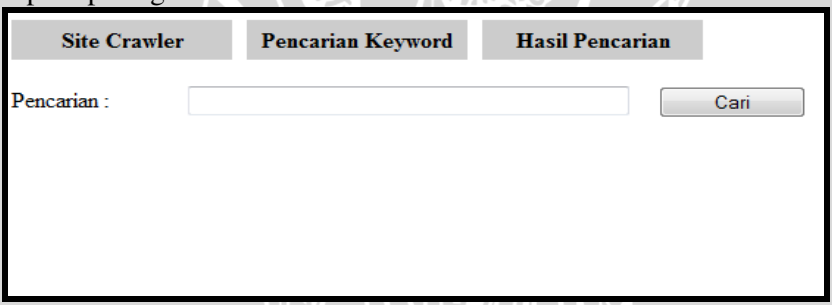
Gambar 4.1 Form Utama

Menu pada bagian *"Site Crawler*" merupakan menu yang digunakan untuk mengisikan alamat situs yang ingin kita *crawling*, tombol "*start*" digunakan untuk memulai proses *crawling* pada situs, sedangkan tombol "*stop*" digunakan untuk menghentikan proses *crawling* pada situs.



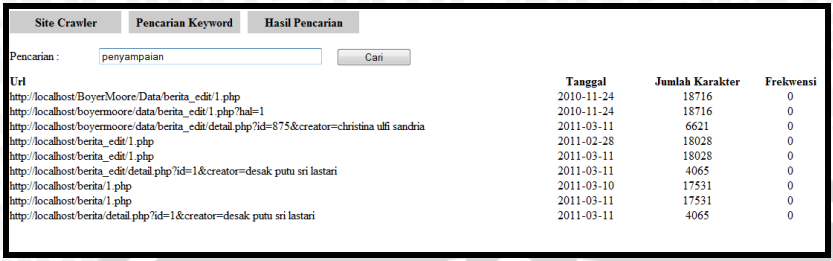
Gambar 4.2 *Site Crawler*

Pertama user akan memasukkan link situs yang akan di lakukan proses *crawling*, selanjutnya klik tombol start untuk memulai dan tombol *stop* untuk menghentikan proses crawling tersebut. Proses crawling terdapat pada gambar 4.2 tersebut diatas.
Untuk menu pencarian keyword, kolom pencarian digunakan untuk mengisi kata yang ingin dicari pada dokumen yang telah di *crawler* seperti pada gambar 4.3.



Gambar 4.3 Menu Pencarian kata

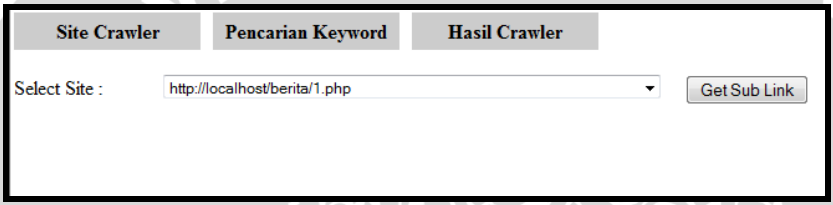
Setelah kata dimasukkan, jika tombol pilih di tekan maka akan tampil hasil pencarian berserta keterangan URL, tanggal *crawling*, jumlah karakter, dan frekuensi, seperti pada gambar 4.4.



Url	Tanggal	Jumlah Karakter	Frekwensi
http://localhost/BoyerMoore/Data/berita_edit/1.php	2010-11-24	18716	0
http://localhost/boyermoore/data/berita_edit/1.php?hal=1	2010-11-24	18716	0
http://localhost/boyermoore/data/berita_edit/detail.php?id=875&creator=christina ulfi sandria	2011-03-11	6621	0
http://localhost/berita_edit/1.php	2011-02-28	18028	0
http://localhost/berita_edit/1.php	2011-03-11	18028	0
http://localhost/berita_edit/detail.php?id=1&creator=desak putu sri lastari	2011-03-11	4065	0
http://localhost/berita/1.php	2011-03-10	17531	0
http://localhost/berita/1.php	2011-03-11	17531	0
http://localhost/berita/detail.php?id=1&creator=desak putu sri lastari	2011-03-11	4065	0

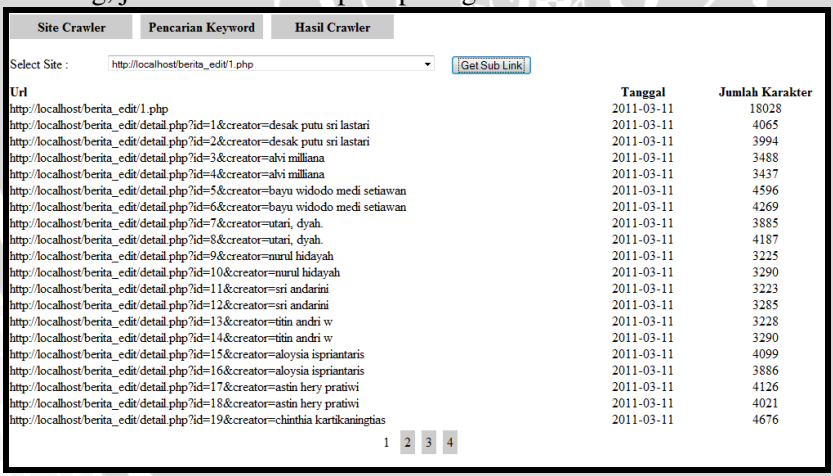
Gambar 4.4 Hasil Pencarian kata

Untuk menu hasil crawler, kolom *select site* digunakan melihat hasil crawler dari situs yang telah di crawler seperti pada gambar 4.5.



Gambar 4.5 Menu Hasil Crawler site

Setelah situs kita pilih , maka jika tombol "*get sub link*" di tekan maka akan tampil sub link dari alamat situs yang di *crawling*, tanggal crawling, jumlah karakter seperti pada gambar 4.6.

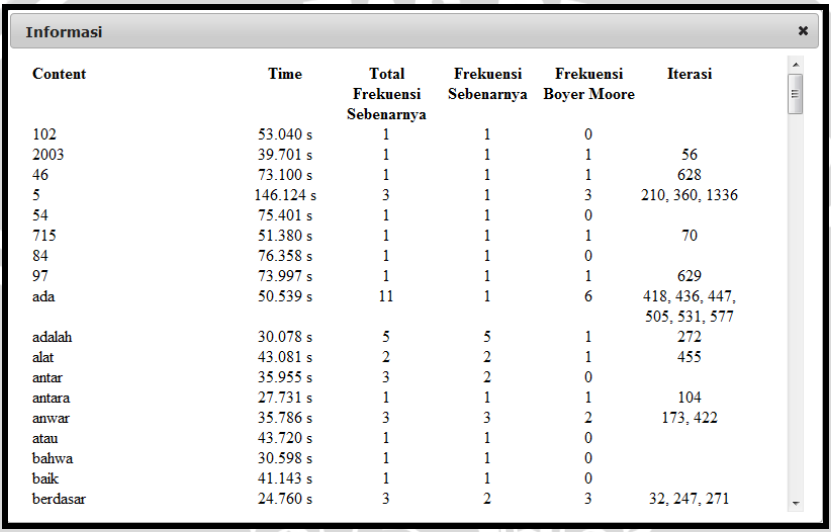


Url	Tanggal	Jumlah Karakter
http://localhost/berita_edit/1.php	2011-03-11	18028
http://localhost/berita_edit/detail.php?id=1&creator=desak putu sri lastari	2011-03-11	4065
http://localhost/berita_edit/detail.php?id=2&creator=desak putu sri lastari	2011-03-11	3994
http://localhost/berita_edit/detail.php?id=3&creator=alvi milliana	2011-03-11	3488
http://localhost/berita_edit/detail.php?id=4&creator=alvi milliana	2011-03-11	3437
http://localhost/berita_edit/detail.php?id=5&creator=bayu widodo medi setiawan	2011-03-11	4596
http://localhost/berita_edit/detail.php?id=6&creator=bayu widodo medi setiawan	2011-03-11	4269
http://localhost/berita_edit/detail.php?id=7&creator=utari, dyah.	2011-03-11	3885
http://localhost/berita_edit/detail.php?id=8&creator=utari, dyah.	2011-03-11	4187
http://localhost/berita_edit/detail.php?id=9&creator=murul hidayah	2011-03-11	3225
http://localhost/berita_edit/detail.php?id=10&creator=murul hidayah	2011-03-11	3290
http://localhost/berita_edit/detail.php?id=11&creator=sri andarini	2011-03-11	3223
http://localhost/berita_edit/detail.php?id=12&creator=sri andarini	2011-03-11	3285
http://localhost/berita_edit/detail.php?id=13&creator=titin andri w	2011-03-11	3228
http://localhost/berita_edit/detail.php?id=14&creator=titin andri w	2011-03-11	3290
http://localhost/berita_edit/detail.php?id=15&creator=aloysia ispriantaris	2011-03-11	4099
http://localhost/berita_edit/detail.php?id=16&creator=aloysia ispriantaris	2011-03-11	3886
http://localhost/berita_edit/detail.php?id=17&creator=astin hery pratiwi	2011-03-11	4126
http://localhost/berita_edit/detail.php?id=18&creator=astin hery pratiwi	2011-03-11	4021
http://localhost/berita_edit/detail.php?id=19&creator=chinthia kartikaningtias	2011-03-11	4676

1 2 3 4

Gambar 4.6 Hasil Crawler site

Informasi lebih lengkap akan di dapatkan ketika *link* pada *sub link*, informasi yang ditampilkan antara lain *content*, *time, frekuensi* dan *iterasi*. Content adalah kata yang ada pada situs yang di crawler tersebut, *time* adalah waktu yang di perlukan untuk menemukan kata tersebut, frekuensi adalah jumlah kemunculan data pada content sebuah situs dan iterasi adalah keterangan kata itu ditemukan pada iterasi keberapa pada isi situs tersebut seperti pada gambar 4.7.


Informasi

Content	Time	Total Frekuensi Sebenarnya	Frekuensi Sebenarnya	Frekuensi Boyer Moore	Iterasi
102	53.040 s	1	1	0	
2003	39.701 s	1	1	1	56
46	73.100 s	1	1	1	628
5	146.124 s	3	1	3	210, 360, 1336
54	75.401 s	1	1	0	
715	51.380 s	1	1	1	70
84	76.358 s	1	1	0	
97	73.997 s	1	1	1	629
ada	50.539 s	11	1	6	418, 436, 447, 505, 531, 577
adalah	30.078 s	5	5	1	272
alat	43.081 s	2	2	1	455
antar	35.955 s	3	2	0	
antara	27.731 s	1	1	1	104
anwar	35.786 s	3	3	2	173, 422
atau	43.720 s	1	1	0	
bahwa	30.598 s	1	1	0	
baik	41.143 s	1	1	0	
berdasar	24.760 s	3	2	3	32, 247, 271

Gambar 4.7 info sub link

3.4 Analisa Hasil

Berdasarkan data yang ada, dilakukan beberapa kali percobaan yaitu dengan memasukkan beberapa alamat situs yang akan dilakukan pengecekan, dari beberapa percobaan tersebut akan didapatkan informasi waktu *crawling* setiap kata, total waktu *crawling*, frekuensi sebenarnya, frekuensi *tuned boyermoore*. Data hasil percobaan ini dapat digunakan untuk mengetahui waktu rata-rata yang diperlukan setiap kata serta *precision* dan *recall* dari algoritma *tuned boyermoore*.

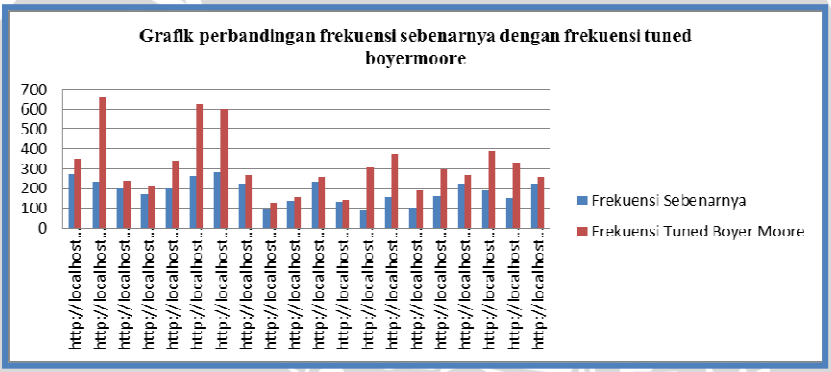
Tabel 4.7 Perbandingan Jumlah Frekuensi Sebenarnya dengan Frekuensi Tuned Boyermoore.

No	Link	Jumlah Frekuensi Sebenarnya	Frekuensi Tuned Boyer Moore
1	http://localhost/berita_edit/detail.php?id=1&creator=Desak%20Putu%20Sri%20Lastari	272	349
2	http://localhost/berita_edit/detail.php?id=112&creator=Dian%20Faza	236	659
3	http://localhost/berita_edit/detail.php?id=116&creator=Aditya%20Sigit	206	239
4	http://localhost/berita_edit/detail.php?id=127&creator=Tulle,%20Andrew%20William	171	211
5	http://localhost/berita_edit/detail.php?id=141&creator=Andi%20Yuliana%20Agnetha	200	341
6	http://localhost/berita_edit/detail.php?id=15&creator=Aloysia Ispriantaris	265	625
7	http://localhost/berita_edit/detail.php?id=157&creator=Desi%20Ariwinanti	282	601
8	http://localhost/berita_edit/detail.php?id=159&creator=Devi%20Yolanda	223	268
9	http://localhost/berita_edit/detail.php?id=174&creator=Bambang%20Siswanto	101	123

10	http://localhost/berita_edit/detail.php?id=176&creator=Kuswanto%20%20Sri%20Lestari%20Purnamaningsih	138	155
11	http://localhost/berita_edit/detail.php?id=206&creator=Dini%20Agustina	235	259
12	http://localhost/berita_edit/detail.php?id=262&creator=Welmin%20Suharto	133	141
13	http://localhost/berita_edit/detail.php?id=267&creator=Waego%20Hadi%20Nugroho,%20l%29%20Bambang%20Guritno,%20%27%29%20clan%20Ludji%20Panca%20Astuti	94	305
14	http://localhost/berita_edit/detail.php?id=273&creator=Salyo%20Sutrisno%20*%29%20dan%20Trise%20sulisiyaningrurn	159	373
15	http://localhost/berita_edit/detail.php?id=285&creator=AR%20Faqik,%20EUana%20S.,%20Yahya,%20Maftuh%20dun%20Anton%20E.	103	187
16	http://localhost/berita_edit/detail.php?id=359&creator=Hari%20Purnomo	162	299
17	http://localhost/berita_edit/detail.php?id=382&creator=Bambang%20Sidharta	227	269
18	http://localhost/berita_edit/detail.php?id=426&creator=Ni%20ayu%20Wulandari	188	392
19	http://localhost/berita_edit/d	149	329

	etail.php?id=45&creator=Dewi%20Kurniawati		
20	http://localhost/berita_edit/detail.php?id=472&creator=Permata%20Penalar.	221	260

Data diatas didapatkan algoritma *tuned boyer moore* tidak sepenuhnya mempunyai kesesuaian frekuensi dengan frekuensi sebenarnya. Pada gambar 4.8 tersebut kolom biru jumlah frekuensi sebenarnya, merah menyatakan jumlah frekuensi dengan *tuned boyer moore*.



Gambar 4.8 Grafik Perbandingan data frekuensi *tuned boyer moore* dengan data sebenarnya.

Pada tabel dibawah ini terdapat nilai *precision* dari hasil percobaan.

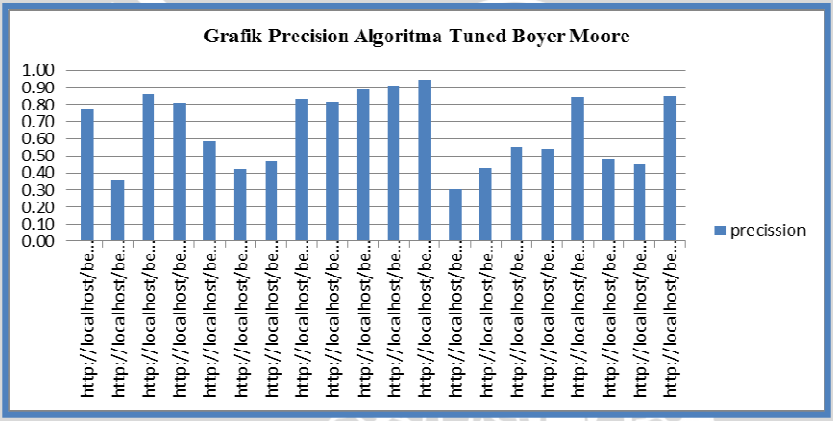
Tabel 4.8 Nilai *precission*.

Link	Precision
http://localhost/berita_edit/detail.php?id=1&creator=Desak%20Putu%20Sri%20Lastari	0.78
http://localhost/berita_edit/detail.php?id=112&creator=Dian%20Faza	0.36

http://localhost/berita_edit/detail.php?id=116&creator=Aditya%20Sigit	0.86
http://localhost/berita_edit/detail.php?id=127&creator=Tulle,%20Andrew%20William	0.81
http://localhost/berita_edit/detail.php?id=141&creator=Andi%20Yuliana%20Agnetha	0.59
http://localhost/berita_edit/detail.php?id=15&creator=Aloysia Ispriantaris	0.42
http://localhost/berita_edit/detail.php?id=157&creator=Desi%20Ariwinanti	0.47
http://localhost/berita_edit/detail.php?id=159&creator=Devi%20Yolanda	0.83
http://localhost/berita_edit/detail.php?id=174&creator=Bambang%20Siswanto	0.82
http://localhost/berita_edit/detail.php?id=176&creator=Kuswanto%20%20Sri%20Lestari%20Purnamaningsih	0.89
http://localhost/berita_edit/detail.php?id=206&creator=Dini%20Agustina	0.91
http://localhost/berita_edit/detail.php?id=262&creator=Welmin%20Suharto	0.94
http://localhost/berita_edit/detail.php?id=267&creator=Waego%20Hadi%20Nugroho,%20l%29%20Bambang%20Guritno,%20%27%29%20clan%20Ludji%20Panca%20Astuti	0.31
http://localhost/berita_edit/detail.php?id=273&creator=Salyo%20Sutrisno%20*%29%20dan%20Trise%20sulisiyaningrurn	0.43
http://localhost/berita_edit/detail.php?id=285&creator=AR%20Faqik,%20EUana%20S.,%20Yahya,%20Maftuh%20dun%20Anton%20E.	0.55
http://localhost/berita_edit/detail.php?id=359&creator=Hari%20Purnomo	0.54

http://localhost/berita_edit/detail.php?id=382&creator=Bambang%20Sidharta	0.84
http://localhost/berita_edit/detail.php?id=426&creator=Ni%20ayu%20Wulandari	0.48
http://localhost/berita_edit/detail.php?id=45&creator=Dewi%20Kurniawati	0.45
http://localhost/berita_edit/detail.php?id=472&creator=Permata%20Penalar.	0.85

Data diatas bisa di dapatkan bahwa data *precision* yang paling tinggi adalah 0,94, sedangkan yang paling rendah adalah 0,42, dan rata-rata nilai precision adalah 0,59.



Gambar 4.9 Grafik *Precision* data frekuensi tuned boyer moore.

Tabel 4.9 Perbandingan jumlah frekuensi kata dengan *tuned boyer moore* dan rata-rata lama waktu pencarian.

Link	Frekuensi Tuned Boyer Moore	Rata-rata waktu tuned boyermoore dalam detik
http://localhost/berita_edit/detail.php?id=1&creator=Desak%20Putu%20Sri%20Lastari	349	29.96

http://localhost/berita_edit/detail.php?id=112&creator=Dian%20Faza	659	23.88
http://localhost/berita_edit/detail.php?id=116&creator=Aditya%20Sigit	239	17.68
http://localhost/berita_edit/detail.php?id=127&creator=Tulle,%20Andrew%20William	211	14.83
http://localhost/berita_edit/detail.php?id=141&creator=Andi%20Yuliana%20Agnetha	341	16.72
http://localhost/berita_edit/detail.php?id=15&creator=Aloysia Ispriantaris	625	30.12
http://localhost/berita_edit/detail.php?id=157&creator=Desi%20Ariwinanti	601	41.53
http://localhost/berita_edit/detail.php?id=159&creator=Devi%20Yolanda	268	15.41
http://localhost/berita_edit/detail.php?id=174&creator=Bambang%20Siswanto	123	2.28
http://localhost/berita_edit/detail.php?id=176&creator=Kuswanto%20%20Sri%20Lestari%20Purnamaningsih	155	5.06
http://localhost/berita_edit/detail.php?id=206&creator=Dini%20Agustina	259	18.88
http://localhost/berita_edit/detail.php?id=262&creator=Welmin%20Suharto	141	5.40

http://localhost/berita_edit/detail.php?id=267&creator=Waego%20Hadi%20Nugroho,%20l%29%20Bambang%20Guritno,%20%27%29%20clan%20Ludji%20Panca%20Astuti	305	3.12
http://localhost/berita_edit/detail.php?id=273&creator=Salyo%20Sutrisno%20*%29%20dan%20Trise%20sulisiyaningrurn	373	6.38
http://localhost/berita_edit/detail.php?id=285&creator=AR%20Faqik,%20EUana%20S.,%20Yahya,%20Maftuh%20dun%20Anton%20E.	187	2.40
http://localhost/berita_edit/detail.php?id=359&creator=Hari%20Purnomo	299	9.55
http://localhost/berita_edit/detail.php?id=382&creator=Bambang%20Sidharta	269	20.38
http://localhost/berita_edit/detail.php?id=426&creator=Ni%20ayu%20Wulandari	392	13.31
http://localhost/berita_edit/detail.php?id=45&creator=Dewi%20Kurniawati	329	9.28
http://localhost/berita_edit/detail.php?id=472&creator=Permata%20Penalar.	117	24.98

Dari data tersebut diatas, dapat diketahui bahwa banyaknya frekuensi *tuned boyer moore* yang ditemukan, tidak berbanding lurus dengan lama waktu pencarian, terdapat beberapa data yang mempunyai frekuensi *tuned boyer moore* besar memiliki waktu rata-rata waktu pencarian yang lebih kecil dibandingkan dengan frekuensi tuned boyer moore yang lebih kecil. Rata-rata waktu yang di

perlukan untuk menemukan satu pencocockan *tuned boyer moore* adalah 5.97 detik.

3.5 Analisa Hasil Secara Keseluruhan

Secara keseluruhan, algoritma *tuned boyer moore* dari data percobaan didapatkan waktu yang di perlukan untuk menemukan satu pencocockan *tuned boyer moore* adalah 5.97 detik. Percobaan tersebut didapatkan perbandingan waktu,yang diketahui bahwa banyaknya frekuensi *tuned boyer moore* yang ditemukan, tidak berbanding lurus dengan lama waktu pencarian, terdapat beberapa data yang mempunyai frekuensi *tuned boyer moore* besar memiliki rata-rata waktu pencarian yang lebih kecil dibandingkan dengan frekuensi *tuned boyer moore* yang lebih kecil.

Untuk data precision dari data hasil percobaan tersebut dapat dijelaskan rata-rata nilai precision dari percoban ini adalah 0,59. Didapatkan data rata-rata nilai *precision* seperti tersebut diatas yang berarti bahwa nilai *wrong* dari aplikasi *crawler* ini cukup besar, hal ini menunjukan bahwa sistem telah bekerja dengan baik yaitu telah berhasil menemukan sejumlah frekuensi kata yang tidak didapat secara manual, tetapi sistem dapat menemukan. Untuk contoh perhitungan data terdapat pada lampiran 3.

Algoritma *tuned boyer moore* ini terdapat perbedaan antara jumlah frekuensi sebenarnya dengan jumlah frekuensi dengan menggunakan algoritma *tuned boyer moore*, hal ini disebabkan terdapat karakter tunggal yang dicocokan menggunakan algoritma *tuned boyer moore* yang karakter tersebut terdapat pada kalimat-kalimat yang ada pada teks, sebagai contoh ketika terdapat sebuah rumus persamaan f=m.a, pengguna hanya ingin mendapatkan karaket f saja yang terdapat pada rumus, karena karakter f juga terdapat pada kalimat atau teks lain maka semakin banyak karakter tersebut yang ditemukan sehingga semakin besar pula frekuensi *tuned boyer moore* yang didapatkan.

Hal lain yang berpengaruh pada percoban ini adalah perangkat keras. Pada percobaan ini hanya menggunakan PC sebagai perangkat keras pengujian dengan spesifikasi terbatas, hal ini menyebabkan pada saan proses *crawling browser* sering *hang* akibat kehabisan *memory*, hal lain yaitu proscessor yang digunakan juga terbatas. Sering kali ketika program ini dijalankan CPU Usage mencapai 100% yang menyebabkan proses tidak bisa berjalan dengan baik.


UNIVERSITAS BRAWIJAYA

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan yang dihasilkan dalam skripsi ini adalah :

1. Aplikasi *web crawler* yang mengimplementasikan algoritma *tuned boyer moore* dengan cara menjadikan algoritma *tuned boyer moore* ini sebagai penghitung frekuensi kemunculan kata pada web tersebut yang kemudian digunakan untuk menentukan peringkat link pada saat pencarian kata oleh pengguna, akan dimunculkan alamat website yang memiliki frekuensi kemunculan kata paling besar sesuai dengan kata yang dicari. Telah dilakukan pengujian terhadap sistem yang dibuat dengan mengujikan dokumen HTML.
2. Sistem berhasil melakukan *crawler* HTML dengan rata-rata waktu yang di perlukan untuk menemukan satu proses pencocokan *tuned boyer moore* adalah 5.97 detik
3. Untuk data *precission* didapatkan bahwa data rata-rata nilai *precision* dari percoban ini adalah 0,59.

5.2 Saran

Untuk pengembangan lebih lanjut disarankan :

1. Menggabungkan algoritma ini dengan algoritma pencocokan kata yang lain guna mensiasati pergeseran karakter agar hasi frekuensi yang didapatkan lebih akurat, dan juga metode lain yang digunakan untuk optimasi algoritma ini lebih baik.
2. Disarankan untuk menggunakan perangkat keras yang mempunyai spesifikasi yang lebih baik dari pada yang digunakan pada penelitian ini.


UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Aryuanda,Fajri, dan Suadi ,Wahyu, S.Kom, Royyana, dan T.C, Henning.2010. *Informatics Media Board News Reader Via Bluetooth Dengan Penerapan Algoritma Web Crawler*. ITS. Surabaya

Atmopawiro, Alsasian. 2010. *Pengkajian Dan Analisis Tiga Algoritma Efisien Rabin-Karp, Knuth-Morris-Pratt, Dan Boyer-Moore Dalam Pencarian Pola Dalam Suatu Teks*. ITB. Bandung.

Cao, Feng. 2004. *Pama: A Fast String Matching Algorithm and Its Application in DNA Sequence Search*. Wayne State University. USA

Charras, Christian and Lecroq, Thierry. *Handbook of Exact String Matching Algorithms*. L'Institut d'électronique et d'informatique Gaspard-Monge (IGM), Paris.

Dwi Utami, Pawestri, dan Muslim I , Royyana, dan T.C, Henning.2009. *Perancangan dan Pembuatan Web Crawler Aplikasi Panduan Pembelian Spesifikasi Komputer Rakitan Online dengan Memanfaatkan Google Gears*. ITS. Surabaya.

Hermanto, Endy. 2007. *Perbandingan Algoritma Boyer-Moore, Tuned Boyer-Moore Dan Turbo Boyer-Moore Untuk Pencarian Kata Pada Berkas Teks*. UKDW. Yogyakarta.

Hovy, E. (2003). Text *Summarization*. Dalam R. Mitkov, *The Oxford Handbook of Computational Linguistics* (hal. 583-589). Oxford: Oxford University Press.

Lee, R. C. T. 1991. *Tuned Boyer Moore Algorithm Fast string searching , HUME A. and SUNDAY D.M., Software - Practice & Experience 21(11), 1991, pp. 1221-1248*. National Chi Nan University, Taiwan.

Masruro, Ahlihi,S.Kom.2011. Pengenalan HTML. Amikom. Yogyakarta.

Pinkerton, Brian. 2000. *WebCrawler: Finding What People Want*. University of Washington. USA

Rosadi K, Aqwam. 2010. *Pengertian Search Engine Optimization*. Gunadarma. Jakarta.

Sedgewick, Robert. 1983. *Algorithms*. *Brown University*. *France*
Yugianus, Pausta. 2011. *Implementasi Algoritma Boyer-Moore Dalam Sistem Penelusuran Katalog Perpustakaan Sekolah*. UPI. Bandung.

LAMPIRAN 1. Contoh Dokumen HTML

Dokumen :
http://localhost/berita_edit/detail.php?id=116&creator=Aditya%20Sigit

Journal
116
Faktor-faktor risiko yang berpengaruh terhadap frekuensi kejadian ISPA pada bayi berumur kurang dari 1 tahun di RB/BKIA/BP "Siti Miriam" Lawang
Aditya Sigit
Penyakit infeksi dan kurang gizi merupakan penyebab terbanyak kesakitan dan kematian pada bayi dan anak-anak di Indonesia. Selama tahun 2004, di RB/BKIA/BP "Siti Miriam" Lawang mencatat bahwa pada bayi berumur kurang dari 1 tahun yang datang berobat, ISPA merupakan penyakit yang paling banyak diderita yakni sebesar 76,24%. Tujuan peneliti adalah mengetahui faktor-faktor risiko yang berpengaruh terhadap frekuensi kejadian ISPA pada bayi berumur kurang dari 1 tahun. Penelitian ini dilakukan secara observasional analitik. Teknik pengambilan sampel secara consecutive sampling. Sampel terdiri dari 123 bayi dengan 56 bayi laki-laki dan 67 bayi perempuan, dibedakan menjadi 2 kelompok yaitu yang sering terkena ISPA (66 sampel) dan yang jarang terkena ISPA (57 sampel). Variabel yang diukur adalah jenis kelamin, pemberian ASI, status imunisasi, kepadatan penghuni rumah, dan riwayat kontak. Hasil penelitian menunjukkan terdapat perbedaan bermakna antara kelompok sering terkena ISPA dengan yang jarang terkena ISPA dalam kaitannya dengan variabel pemberian ASI dan riwayat kontak. Kesimpulan penelitian ini adalah adanya hubungan yang signifikan antara pemberian ASI dan riwayat kontak dalam kaitannya dengan frekuensi terjadinya ISPA pada bayi berumur kurang dari 1 tahun.

LAMPIRAN 2. Data Hasil *Crawler*

Content	Time	Frekuensi Sebenarnya	Frekuensi Tuned Boyer Moore
1	71.739	4	7
116	20.181	1	1
123	25.232	1	1
2	74.222	1	4
2004	18.124	1	1
24	40.213	1	1
56	36.399	1	1
57	40.039	1	1
66	35.254	1	1
67	35.728	1	1
76	35.429	1	1
adalah	14.57	3	3
adanya	13.317	1	1
aditya	15.436	1	1
anak	22.632	1	2
analitik	12.636	1	1
antara	13.996	2	2
asi	25.514	3	7
bahwa	16.704	1	1
banyak	14.935	1	2
bayi	22.27	8	8
bermakna	11.105	1	1
berobat	18.594	1	1
berpengaruh	14.264	2	2
berumur	11.195	4	4
bkia	19.26	2	2
bp	35.639	2	2

consecutive	8.606	1	1
dalam	14.538	2	2
dan	26.901	8	9
dari	22.63	5	5
datang	16.365	1	1
dengan	19.159	4	4
di	36.3	3	13
dibedakan	11.235	1	1
diderita	11.159	1	1
dilakukan	10.1	1	1
diukur	13.884	1	1
faktor	15.25	2	4
frekuensi	13.402	3	3
gizi	18.691	1	1
hasil	15.175	1	1
hubungan	10.888	1	1
imunisasi	9.815	1	1
indonesia	10.764	1	1
infeksi	12.225	1	1
ini	26.875	2	2
ispa	20.974	8	8
jarang	16.724	2	2
jenis	18.238	1	1
journal	11.149	1	1
kaitannya	9.107	2	2
kejadian	15.507	2	2
kelamin	16.673	1	1
kelompok	13.027	2	2
kematian	12.174	1	1
kepadatan	9.027	1	1

kesakitan	10.617	1	1
kesimpulan	9.342	1	1
kontak	13.001	3	3
kurang	15.944	5	5
laki	26.465	1	2
lawang	16.186	2	2
mencatat	12.033	1	1
mengetahui	10.461	1	1
menjadi	18.23	1	1
menunjukkan	7.711	1	1
merupakan	10.739	2	2
miriam	15.163	2	2
observasional	8.265	1	1
pada	21.065	5	7
paling	17.281	1	1
pemberian	11.937	3	3
peneliti	11.467	1	4
penelitian	11.456	3	3
pengambilan	10.103	1	1
penghuni	8.566	1	1
penyakit	13.105	2	2
penyebab	12.887	1	1
perbedaan	9.998	1	1
perempuan	10.025	1	1
rb	37.706	2	4
risiko	13.284	2	2
riwayat	11.285	3	3
rumah	15.268	1	1
sampel	14.56	4	4
sampling	12.601	1	1

sebesar	13.462	1	1
secara	15.293	2	2
selama	14.853	1	1
sering	16.307	2	2
sigit	19.063	1	1
signifikan	9.534	1	1
siti	19.758	2	2
status	12.677	1	1
tahun	16.026	5	5
teknik	16.455	1	1
terbanyak	12.464	1	1
terdapat	9.902	1	1
terdiri	12.476	1	1
terhadap	13.446	2	2
terjadinya	10.424	1	1
terkena	13.138	4	4
tujuan	14.51	1	1
variabel	13.891	2	2
yaitu	16.96	1	1
yakni	17.531	1	1
yang	23.284	9	9
Jumlah Keyword : 108	**1909.393**	**206**	**239**

LAMPIRAN 3. Data Perhitungan Precision

Content	Frekuensi Sebenarnya	Frekuensi Tuned Boyer Moore	**correct**	**wrong**	**Miss**
1	4	7	4	3	0
116	1	1	1	0	0
123	1	1	1	0	0
2	1	4	1	3	0
2004	1	1	1	0	0
24	1	1	1	0	0
56	1	1	1	0	0
57	1	1	1	0	0
66	1	1	1	0	0
67	1	1	1	0	0
76	1	1	1	0	0
adalah	3	3	3	0	0
adanya	1	1	1	0	0
aditya	1	1	1	0	0
anak	1	2	1	1	0
analitik	1	1	1	0	0
antara	2	2	2	0	0
asi	3	7	3	4	0
bahwa	1	1	1	0	0
banyak	1	2	1	1	0
bayi	8	8	8	0	0
bermakna	1	1	1	0	0
berobat	1	1	1	0	0
berpengaruh	2	2	2	0	0
berumur	4	4	4	0	0

bkia	2	2	2	0	0
bp	2	2	2	0	0
consecutive	1	1	1	0	0
dalam	2	2	2	0	0
dan	8	9	8	1	0
dari	5	5	5	0	0
datang	1	1	1	0	0
dengan	4	4	4	0	0
di	3	13	3	10	0
dibedakan	1	1	1	0	0
diderita	1	1	1	0	0
dilakukan	1	1	1	0	0
diukur	1	1	1	0	0
faktor	2	4	2	2	0
frekuensi	3	3	3	0	0
gizi	1	1	1	0	0
hasil	1	1	1	0	0
hubungan	1	1	1	0	0
imunisasi	1	1	1	0	0
indonesia	1	1	1	0	0
infeksi	1	1	1	0	0
ini	2	2	2	0	0
ispa	8	8	8	0	0
jarang	2	2	2	0	0
jenis	1	1	1	0	0
journal	1	1	1	0	0
kaitannya	2	2	2	0	0
kejadian	2	2	2	0	0
kelamin	1	1	1	0	0
kelompok	2	2	2	0	0

kematian	1	1	1	0	0
kepadatan	1	1	1	0	0
kesakitan	1	1	1	0	0
kesimpulan	1	1	1	0	0
kontak	3	3	3	0	0
kurang	5	5	5	0	0
laki	1	2	1	1	0
lawang	2	2	2	0	0
mencatat	1	1	1	0	0
mengetahui	1	1	1	0	0
menjadi	1	1	1	0	0
menunjukkan	1	1	1	0	0
merupakan	2	2	2	0	0
miriam	2	2	2	0	0
observasional	1	1	1	0	0
pada	5	7	5	2	0
paling	1	1	1	0	0
pemberian	3	3	3	0	0
peneliti	1	4	1	3	0
penelitian	3	3	3	0	0
pengambilan	1	1	1	0	0
penghuni	1	1	1	0	0
penyakit	2	2	2	0	0
penyebab	1	1	1	0	0
perbedaan	1	1	1	0	0
perempuan	1	1	1	0	0
rb	2	4	2	2	0
risiko	2	2	2	0	0
riwayat	3	3	3	0	0
rumah	1	1	1	0	0

sampel	4	4	4	0	0
sampling	1	1	1	0	0
sebesar	1	1	1	0	0
secara	2	2	2	0	0
selama	1	1	1	0	0
sering	2	2	2	0	0
sigit	1	1	1	0	0
signifikan	1	1	1	0	0
siti	2	2	2	0	0
status	1	1	1	0	0
tahun	5	5	5	0	0
teknik	1	1	1	0	0
terbanyak	1	1	1	0	0
terdapat	1	1	1	0	0
terdiri	1	1	1	0	0
terhadap	2	2	2	0	0
terjadinya	1	1	1	0	0
terkena	4	4	4	0	0
tujuan	1	1	1	0	0
variabel	2	2	2	0	0
yaitu	1	1	1	0	0
yakni	1	1	1	0	0
yang	9	9	9	0	0
	206	239	206	33	0