Penerapan Algoritma String Matching dalam Penghapusan Stopword

Muhammad Bintang Pananjung 13519004
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519004@std.stei.itb.ac.id

***Abstract*—Dalam pencarian kata, sering kali pengguna memasukkan input pattern yang tidak sesuai dengan keinginan kita. Banyak sekali kata-kata yang seharusnya tidak ada bahkan mengganggu pencarian pattern. Kata-kata itu disebut juga dengan stopwords. Stopwords ini dapat dideteksi menggunakan beberapa pola atau cara. Pencarian stopwords ini dapat menggunakan algoritma string matching, sehingga pencarian kata dapat lebih efisien.**

Keywords—String matching; stopword; pencarian

I. PENDAHULUAN

Hampir setiap orang sudah tidak asing dengan istilah search engine. Search engine adalah mesin pencari kata atau pattern dalam suatu database. Sering kali digunakan pada website search. Pengguna akan memasukkan pencarian berupa atau kata atau kalimat singkat yang berkaitan dengan yang ingin dicari, kemudian pattern tersebut akan diolah oleh algoritma didalam search engine tersebut sehingga dapat ditemukan hasil yang serupa atau mirip dengan data yang berada pada database. Dalam memasukkan pencarian, pengguna sering kali memasukkan kata-kata yang sebenarnya mengganggu pencarian atau bahkan tidak berhubungan sama sekali dengan yang dicari. Sebagai contoh stopword misalnya kata ganti, konjungsi, kata tanya, dan lain-lain. Contoh dalam patternnya adalah "makanan yang disukai oleh kucing". Kata "oleh" dan "yang" merupakan stopword karena tanpa kedua kata tersebut, pencarian tetap akan berjalan dengan baik.

Setiap stopword yang terdapat pada kata dapat mengganggu pencarian hasil yang sesuai dengan database yang ada. Semakin banyak stopword yang ada dari masukkan pengguna, semakin jauh kesesuaian dari pencarian yang diinginkan dengan hasil yang ditemukan. Dalam melakukan pencarian, algoritma biasanya akan menjadikan tiap kata dari kalimat tersebut menjadi token-token yang memiliki nilai tertentu sesuai dengan banyaknya kata tersebut pada kalimat tersebut. Token-token ini kemudian akan dibandingkan dengan token pada data yang berada di database. Misalnya pengguna memasukkan kalimat "makanan yang disukai oleh kucing". Dari total 5 kata pada kalimat tersebut, terdapat 2 stopword sehingga persentase stopword dalam kalimat adalah 40%. Maka, ketidaksesuaian pencarian akan bertambah cukup besar dari seharusnya. Data yang dicari seharusnya mengandung kata-kata "makanan", "disukai", "kucing". Sementara data-data yang mengandung kata-kata "oleh" dan "yang" juga akan masuk kedalam hasil pencarian. Terlebih lagi kata konjungsi sangatlah umum dipakai dalam penulisan sebuah kalimat. Dengan kata konjungsi yang digunakan lebih banyak dalam suatu data saja dapat merusak kesesuaian hasil pencarian.

Oleh karena itu, stopword sangat mengganggu dalam pencarian suatu data. Maka dari itu, stopword dalam masukkan pencarian harus dihapuskan agar pencarian dapat menemukan hasil dengan kesesuaian yang optimal.

Dalam hal ini, penulis akan mencoba menerapkan algoritma string matching dalam penghapusan stopword dari masukan pencarian. Algoritma string matching adalah algoritma yang berguna dalam mencari suatu kata atau kalimat terdapat pada suatu text. Dengan mencari stopword dari suatu kalimat atau masukan, maka diharapkan algoritma string matching akan dapat menemukan stopwordnya.

II. LANDASAN TEORI

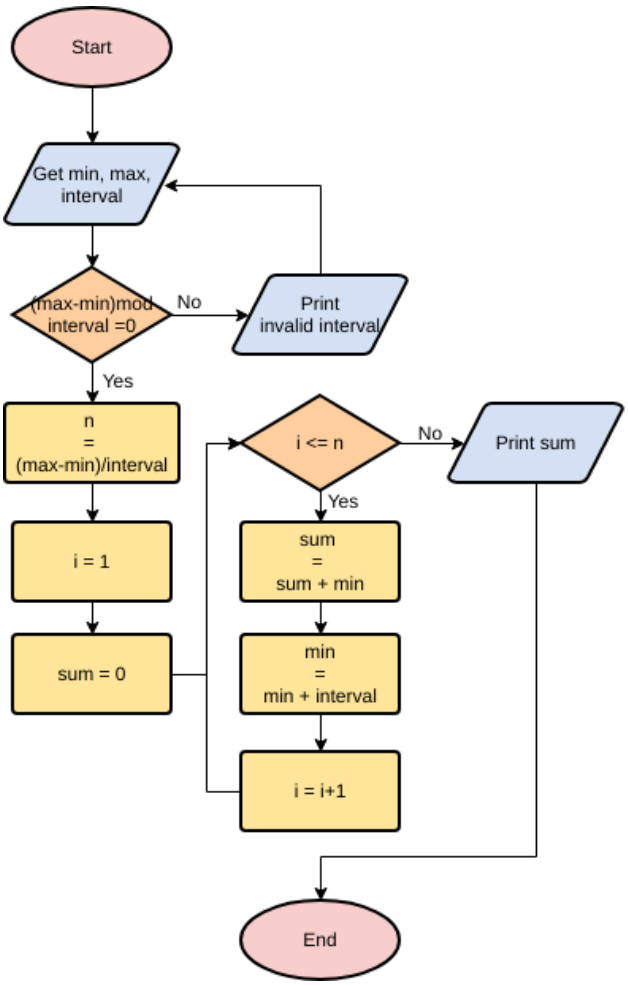
A. Algoritma

Algoritma adalah suatu metode yang terdiri dari rangkaian tahapan sistematis dan terstruktur yang digunakan untuk mencari solusi dari suatu permasalahan. Kata algoritma berasal dari nama ahli matematika yang menemukan tentang algoritma itu sendiri. Orang itu adalah Abu Ja'far Muhammad Ibnu Musa Al Khawarizmi, lebih tepatnya dari kata "Al Khawarizmi" kata algoritma itu dilatinasi menjadi algorism.

Pada awalnya, kata algoritma adalah istilah yang merujuk hanya ke aturan aritmetis dalam penyelesaian persoalan dengan menggunakan bilangan numerik arab saja. Pada abad ke-18, definisinya berkembang menjadi lebih luas dengan mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan dalam penyelesaian persoalan.

Algoritma biasa digunakan dalam pemecahan suatu masalah melalui program komputer. Hal ini karena komputer membutuhkan suatu intruksi berupa solusi prosedural dengan alur yang pasti.

Sebelum direalisasikan dalam program komputer, algoritma biasanya digambarkan terlebih dahulu menggunakan flowchart/diagram alur. Dari diagram alur ini, runtutan penyelesaian sudah jelas dan pasti, tidak ada yang lompat ataupun terlewat. Selanjutnya barulah diagram tadi diimplementasikan dalam bentuk program komputer.



Gambar 1. Contoh flowchart dari algoritma

(Sumber :https://online.visual-paradigm.com/repository/images/9e081ad3-2035-48fa-a226-06b0238a4a19/flowchart-design/simple-mathematics-algorithm.png)

Penggambaran dari suatu ide algoritma tidak selalu dalam bentuk flowchart, banyak metode lain yang bisa digunakan.

Kejelasan dan kepastian dari algoritma sangatlah penting. Setiap langkah dalam algoritma haruslah pasti dan jelas maknanya. Algoritma juga harus efektif dan efisien. Setiap langkah harus memiliki progress yang mengarah kepada solusi dan juga tidak bertele-tele (dalam hal ini, mengulang langkah lain yang tidak perlu). Tanpa memenuhi hal-hal di atas, maka algoritma tidaklah hanya sebuah ide yang fana dan tidak bisa direalisasikan.

B. String Matching Algorithm

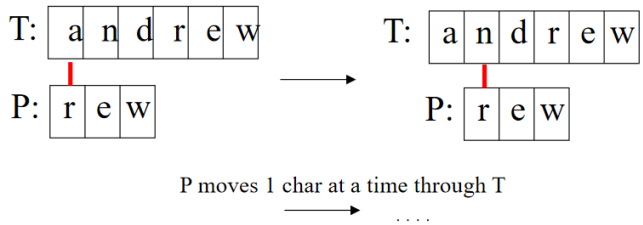
String Matching Algorithm adalah algoritma yang menerima input berupa pattern dari suatu string, kemudian akan dicocokkan dengan yang ada pada text tertentu. Pattern string bisa berupa kata ataupun kalimat, tetapi tetap lebih pendek dari text yang akan dicocokkan.

Sebagai contoh dalam search engine, pattern adalah masukan pencarian dari pengguna sementara text adalah keseluruhan string dari tiap data di dalam database. Penerapan lain yang menggunakan algoritma ini diantaranya adalah pendetexti plagiarisme, pengurutan bioinformatika dan DNA, forensik digital, cek pengejaan, filter spam, dan lain-lain.

C. String Matching dengan Algoritma Brute Force

Misalkan sebuah pattern dan sebuah text bertipe String atau array of character. Pattern disimbolkan dengan P dan text disimbolkan dengan T. Maka, algoritma brute force untuk string matching adalah sebagai berikut:

1. Setiap karakter dari pattern akan dicocokkan dengan setiap karakter dari text secara terurut (dari kiri ke kanan) sampai karakter pada pattern telah dicocokkan semua atau ada karakter yang tidak sesuai/mismatch.
2. Jika karakter dari pattern ada yang mismatch, langsung menggeser pencocokan karakater pertama pada pattern dengan karakter selanjutnya dari indeks awal seblumnya pada text (P[x] dicocokkan dengan T[x+1]).
   Jika tidak ada yang mismatch, maka pencarian berhasil.
3. Selama terjadi mismatch, ulangi langkah ke 1 begitu juga selanjutnya langkah 2 hingga akhir text.



Gambar 2. Contoh string matching dengan algoritma brute force

(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Algoritma ini memiliki kelebihan yaitu sangat mudah diimplementasikan. Tidak ada kondisi khusus atau metode khusus dalam algoritmanya. Namun, algoritma ini memiliki kekurangan yaitu buruknya kompleksitas jika pattern dan text yang dicocokkan sangat panjang. Kompleksitas rata-rata dari algoritma ini adalah O(m+n) dengan kompleksitas terbaik yaitu O(n) dengan n adalah panjang text dan m adalah panjang pattern. Meskipun cukup buruk, kompleksitas ini masih cukup cepat dalam melakukan pencarian. Kasus terburuk dari algoritma ini adalah ketika pattern ditemukan diakhir text dan pattern tersebut sangat panjang. Contohnya text “aaaaaaaaaaaaaaaaaaaaaaaaaaaaac” dengan pattern “aaac”, diperlukan pergeseran maksimal sebanyak panjang text dan pengecekan yang maksimal juga sebanyak panjang pattern. Sehingga, kompleksitas terburuk dari algoritma ini adalah O(mn).

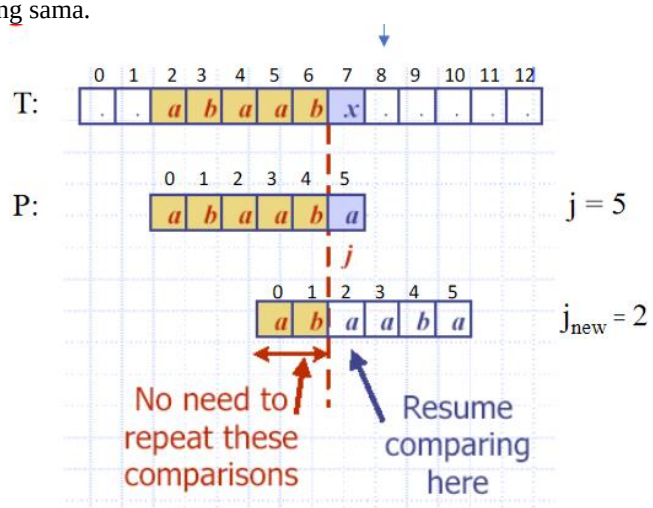
D. Algoritma Knuth-Morris-Pratt

Pada algoritma brute force, setiap kali ada mismatch dari pencocokan pattern dengan text, pergeseran selalu 1 karakter dari kiri ke kanan. Berbeda dengan algoritma brute force, Knuth-Morris-Pratt menggunakan metode yang berbeda dalam

melakukan pergeserannya. Setiap kali mismatch terjadi antara pattern dengan text, pergeseran yang dilakukan akan berdasarkan kepada suatu fungsi tertentu. Fungsi inilah yang membuat pencarian menggunakan algoritma ini lebih cepat dibanding menggunakan algoritma brute force.

Algoritma ini ditemukan oleh D.E. Knuth, kemudian dikembangkan juga bersama dengan J.H. Morris dan V.R. Pratt.

Misalnya sebuah pattern bertipe string dengan simbol P dan text bertipe string dengan simbol T. Sebelum memulai melakukan pencocokan antara pattern dengan text, dibuat terlebih dahulu array yang berisi nilai border dari pattern. Fungsi untuk menentukan nilai border ini disebut sebagai KMP Border Function. Setiap pattern akan dicari suffix dan prefixnya. Prefix adalah awalan sementara suffix adalah akhiran dari pattern. Misalnya sebuah pattern "bagus", maka prefixnya adalah "b", "ba", "bag", "bagu", dan "bagus". Sementara untuk suffixnya adalah "s", "us", "gus", "agus", dan "bagus". Fungsi KMP Border ini akan mencari kesamaan tiap suffix dengan prefixnya dengan nilai yang dikembalikan adalah panjang dari string maksimal dari prefix dan suffix yang sama.



Gambar 3. Contoh ilustrasi penentuan nilai KMP border function
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Prefix dan suffix yang diambil adalah tiap range tertentu (bukan keseluruhan dari pattern). Range ini adalah dari (0..x) untuk prefix dan (1..x) untuk suffix dengan x adalah indeks dari pattern (0..m) dikurangi 1. Misalnya pattern "kapak" untuk indeks 3, maka range untuk prefixnya adalah (0..2) dengan prefix "k","ka", dan "kap" sememtara suffixnya adalah (1..2) dengan suffix "k","ak", dan "pak". Karena panjang string maksimal antara suffix dan prefix yang sama adalah 1 (yaitu "k"), maka nilai KMP bordernya adalah 1. Fungsi itu dilakukan tiap indeks dari pattern (0..m).

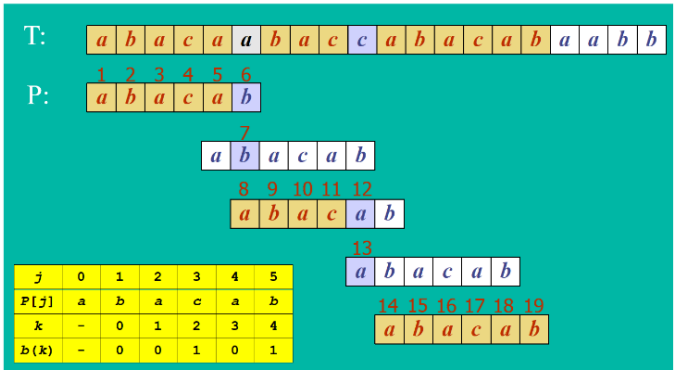
(k = j-1)

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k	-	0	1	2	3	4
b(k)	-	0	0	1	1	2

b(k) is the size of the largest border.

Gambar 4. Contoh KMP Border Function (b(k))
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Setelah KMP Border Function dibuat, pencocokan string dilakukan dengan tiap mismatch, index text saat itu bergeser ke index pattern yang sesuai dengan nilai indeks yang ada pada KMP Border Function. Misalnya, pattern "abaaba" terjadi mismatch pada indeks 4. Maka pencocokan dimulai pada indeks 1 pattern.



Gambar 5. Contoh ilustrasi algoritma KMP
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Dapat disimpulkan prosedur algoritma KMP sebagai berikut:

1. Pencocokan karakter antara pattern dengan text dimulai dari kiri ke kanan.
2. Jika mismatch terjadi, maka pergeseran sesuai dengan KMP Border function pada indeks tersebut. Dimana pencocokan dimulai pada indeks b(k) dan k adalah indeks pattern – 1.
   Jika tidak ada mismatch, maka pencarian berhasil.
3. Ulangi langkah 1 begitu pula langkah 2 hingga akhir text.

Algoritma ini memiliki kompleksitas O(m) saat mencari KMP Border Function dan O(n) saat mencari hasil dari pencarian string, sehingga totalnya adalah O(m+n). jauh lebih cepat jika dibandingkan dengan brute force.

Algoritma ini memiliki keuntungan berupa tidak pernah melakukan pemunduran indeks dari textnya sehingga sangat baik dalam pemrosesan file besar yang dibaca dari luar melalui stream network.

Kelemahan dari algoritma ini adalah saat text dan pattern memiliki karakter yang bervariasi (tidak ada yang sama).

Mismatch sering terjadi dan pergeserannya jarang ke indeks > 0. Hal ini berarti bahwa pergeserannya sama saja dengan brute force.

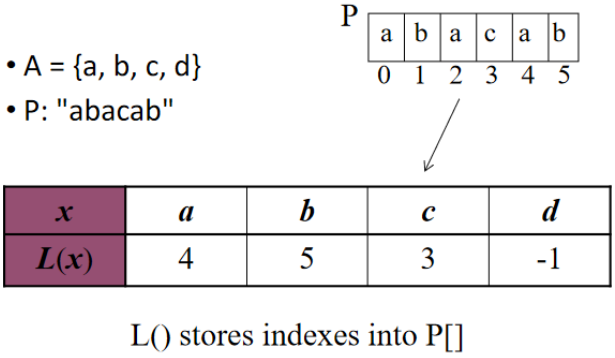
E. Algoritma Boyer-Moore

Pada algoritma brute force dan KMP pergeseran terjadi dari kiri ke kanan untuk patternnya. Berbeda dengan keduanya algoritma Booyer-Moore melakukan pergeseran dari kanan ke kiri untuk pencocokan karakter patternya. Pencocokan antara pattern dengan text tetap dari kiri ke kanan.

Algoritma ini dibuat oleh R.M. Booyer dan J.S. Moore. Algoritma ini sebenarnya mirip dengan KMP dalam melakukan pergeseran, sama-sama memiliki fungsi untuk bergeser indeks. Akan tetapi, fungsi dalam algoritma BM mempunyai kondisi-kondisi yang lebih spesifik. Jika dalam algoritma KMP hanya perlu dicari nilai bordernya dengan fungsi yang sama, algoritma BM memiliki beberapa kondisi yang berbeda yang harus dicek ketika terjadi mismatch sebelum melakukan pergeseran.

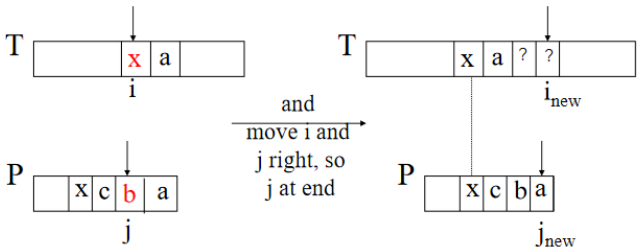
Algoritma BM memiliki 2 teknik, yaitu looking-glass technique dan character-jump technique. looking-glass technique yaitu pengecekan yang terbalik (dari kanan ke kiri) dan character-jump technique yaitu indeks akan lompat ketika mismatch dengan kondisi tertentu.

Ada 3 kondisi yang berbeda yang perlu dicek saat terjadi mismatch. Kondisi pertama adalah ketika mismatch terjadi pada karakter tertentu (misalnya x) maka perlu dicek apakah ada last occurrence dari x di kiri indeks pattern tersebut. Last occurrence adalah kemunculuan terakhir (indeks paling akhir) dari suatu karakter pada pattern. Jika ada beberapa karakter yang sama muncul, maka yang dilihat hanya yang berada pada indeks paling terakhir



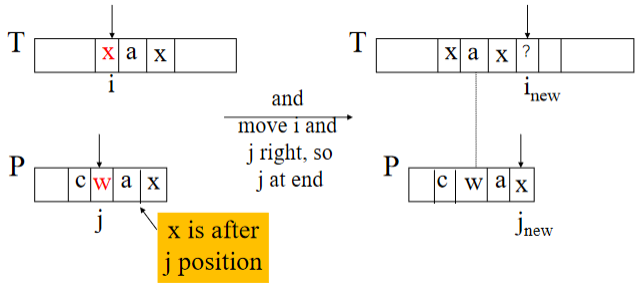
Gambar 6. Contoh last occurrence dari suatu pattern (Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Jika kondisi ini memenuhi, maka indeks pattern akan diganti menjadi indeks akhir pattern yaitu (m-1) dan indeks text akan lompat sebanyak (i+ (m-1)-lo) dengan m adalah panjang pattern, i adalah indeks text saat mismatch, lo adalah last occurrence dari karakter yang mismatch.



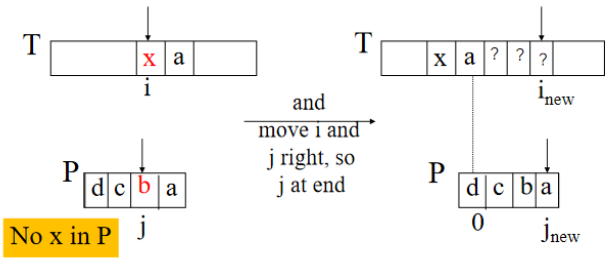
Gambar 7. Contoh kondisi 1 dari character-jump technique (Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Kondisi kedua adalah ketika mismatch terjadi pada karakter tertentu (x), maka perlu dicek apakah last occurencenya berada pada kanan dari indeks pattern saat ini. Jika ada, maka lakukan pergeseran untuk indeks pattern menjadi indeks akhir pattern (m-1) dan indeks text lompat sejauh (i+m-j) yang berarti akan lompat sampai sebanyak indeks pattern yang tersisa.



Gambar 8. Contoh kondisi 2 dari character-jump technique (Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Kondisi ketiga terjadi ketika mismatch, tidak ada kemunculan dari karakter yang mismatch baik di kanan maupun di kiri indeks pattern saat ini. Jika ini terjadi, maka indeks dari pattern akan mulai dari indeks akhir pattern lagi (m-1) kemudian indeks text lompat sebanyak jumlah karakter yang ada pada pattern (i+m).



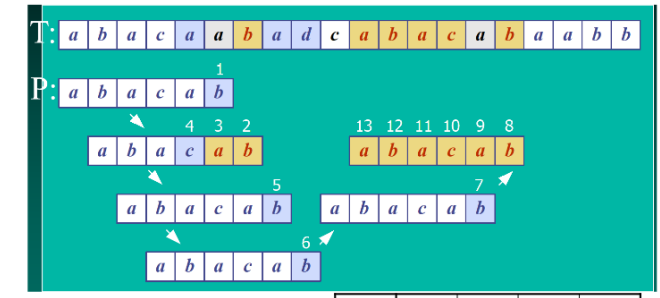
Gambar 9. Contoh kondisi 3 dari character-jump technique (Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Secara umum, algoritma BM dapat disimpulkan sebagai berikut:

1. Cari nilai last occurrence dari masing-masing karakter yang ada pada pattern.
2. Letakkan pattern pada awal text sehingga indeks awal pattern = indeks awal text

3. Lakukan pencocokan mulai dari indeks paling terakhir pattern (m-1) hingga paling awal (dari kanan ke kiri).
4. Jika terjadi mismatch, cek apakah kondisi tersebut masuk ke dalam kondisi yang mana (kondisi 1 atau 2 atau 3). Jika sudah, lakukan character-jump sesuai dengan kondisinya.
5. Jika tidak ada mismatch, maka pencarian berhasil.
6. Ulangi langkah 3 dan 4 sampai akhir text.

Boyer-Moore Example (2)



x	a	b	c	d
L(x)	4	5	3	-1

Gambar 10. Contoh pencocokan string dengan algoritma Booyer-Moore
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Algoritma ini memiliki kompleksitas terburuk O(nm+A). A ini bergantung kepada banyaknya variasi dari karakter yang ada pada pattern dan text. Makin bervariasi maka makin cepat algoritma ini berjalan, sebaliknya makin lambat juga jika variasi karakter sedikit. Algoritma ini sangat cocok untuk text dalam bahasa inggris (karakter variatif) dan buruk untuk biner. Algoritma ini juga secara signifikan lebih cepat daripada algoritma brute force.

III. Implementasi dan Pengujian

Dalam penghapusan stopword, tentunya perlu diketahui terlebih dahulu kata-kata yang termasuk dalam stopword. Dengan merujuk kepada http://static.hikaruyuuki.com/wp-content/uploads/stopword_list_tala.txt , terdapat sekitar 700 stopword yang ada dalam Bahasa Indonesia. Namun, jumlah ini tidak pasti karena banyaknya variasi kata yang mungkin belum diketahui sebagai stopword.

Dalam pengolahan masukan pencarian dalam search engine, penghapusan stopword akan dilakukan sebelum pengolahan terjadi. Sehingga, masukan yang dicari sudah tidak mengandung stopword lagi. Penghapusan stopword dilakukan dengan cara mencocokkan string antara pattern yang berada pada list stopword dengan text. Setiap stopword dalam list stopword akan dilakukan string matching dengan text masukan pencarian sehingga diperlukan string matching sebanyak jumlah stopword dalam list. Jika rata-rata kompleksitas tiap string matching adalah O(m+n), maka kompleksitas untuk pengecekan ini adalah O(N(m+n)) dengan N adalah jumlah stopword yang ada pada list. Kompleksitas ini tergolong besar sehingga perlu difilter kembali stopword yang berada dalam list supaya program dapat berjalan dengan lebih efisien.

Dalam algoritma yang ingin penulis buat, semua stopword akan dimasukkan ke dalam database berupa json file untuk memudahkan penambahan, pengurangan dan pengaksesan dari stopword itu sendiri. Kemudian, algoritma untuk pencocokan string akan dibagi menjadi dua, yaitu algoritma KMP dan algoritma BM.

Gambar 11. Contoh list dari stopword dalam file JSON

(Sumber : Penulis)

Penulis mengasumsikan search engine akan mengolah data berupa hasil dari tokenisasi masukan pencarian yang mana berisi parsing dari tiap kata pada masukan dan juga jumlah kata tersebut dalam masukan pencarian tersebut.

Pada makalah ini, akan diterapkan algoritma penghapusan stopword menggunakan algoritma KMP dalam Bahasa javascript.

Sebelum membuat algoritma, ada beberapa hal yang perlu diperhatikan dalam penghapusan stopword, diantaranya adalah:

1. Stopword selalu dalam bentuk per kata atau gabungan kata (bukan merupakan bagian dari suatu kata).
2. Oleh karena poin 1, perlu dicek 2 hal saat melakukan pencocokan string:
    a. Mengecek indeks sebelum dan sesudah kata dalam text apakah berupa spasi atau bukan. Jika spasi, maka huruf awal dari kata tersebut cocok dengan stopword. Jika bukan, maka

huruf awal kata tersebut tidak cocok dengan stopword, bisa jadi kata tersebut mengandung stopword.

b. Mengecek apakah kata yang sedang dicek merupakan kata pertama atau kata terakhir dari text. Jika iya, maka bisa dipastikan kata tersebut cocok dengan stopword. Jika tidak, maka perlu dicek poin a.

Dari hal-hal di atas, algoritma KMP perlu dimodifikasi agar output dapat sesuai dengan yang diinginkan.

Algoritma KMP lebih efektif digunakan untuk tipe pattern atau text yang variasi karakternya tidak terlalu banyak (cenderung berulang karakternya). Oleh karena itu, algoritma KMP lebih efektif dan efisien jika digunakan untuk melakukan penghapusan stopword dalam Bahasa Indonesia.

Hal pertama yang perlu dilakukan adalah membuat KMP Border Function yang menerima masukan berupa pattern stopword dan return berupa array.

```
function kmpBorder(pattern){
    var borderArr = [0,0];
    for(var i=0; i<pattern.length;i++){
        var prefix = "";
        var suffix = "";
        if(i>1){
            var added = false;
            for (let j=0; j<i-1;j++){
                prefix+=pattern[j];
                suffix=pattern[(i-j-1)]+suffix;
                if(prefix==suffix){
                    if(added){
                        borderArr.pop();
                    }
                    else{
                        added=true;
                    }
                    borderArr.push(prefix.length);
                }
            }
            if(!added){
                borderArr.push(0);
            }
        }
    }
    //console.log(borderArr);
    return borderArr;
}
```

Gambar 12. Fungsi Border KMP dalam javascript

(Sumber : Penulis)

Hal yang perlu dilakukan selanjutnya membuat fungsi untuk algoritma pencocokan string KMP itu sendiri. Perlu diperhatikan ada beberapa modifikasi dalam algoritma ini yang sudah dijelaskan sebelumnya. Fungsi ini menerima masukan berupa pattern, text serta array yang berisi nilai border KMP.

```
function kmpMatching(pattern,text,borderKMP){
    var i=0;
    var match="";
    var idxfound=[];
    var lastidx=-1;
    //console.log(text.length);
    while(i<text.length){
        for(var j=0;j<pattern.length;j++){
            if(pattern[j]==text[i]){
                if(j==0){
                    if(i!=0){
                        if(text[i-1]!=" "){
                            j=0;
                        }
                    }
                }
                if(j==pattern.length-1){
                    if(i==text.length-1){
                        idxfound.push(i);
                        j=0;
                    }
                    else{
                        if(text[i+1]==" "){
                            idxfound.push(i);
                            j=0;
                        }
                    }
                }
                i++;
            }
            else{
                if(j==0){
                    i++;
                }
                else{
                    j=borderKMP[j];
                }
                j--;
            }
            if(i>=text.length){
                break;
            }
        }
    }
    return idxfound;
}
```

Gambar 13. Fungsi algoritma pencocokan string KMP yang termodifikasi dalam javascript

(Sumber : Penulis)

Hal selanjutnya adalah melakukan pencarian dan penghapusan stopword dengan pattern merupakan stopword dari list stopword dan text merupakan masukan pencarian dari pengguna. Dalam melakukan penghapusan, stopword tidak perlu benar-benar dihapus dari text masukan pencarian.

Stopword-stopword tersebut dapat diganti dengan spasi karena spasi sendiri dalam pencarian di search engine tidak diperhitungkan.

```
function replaceCharRange(input,idxawal,idxakhir){
    let str=input.split('');
    for(let i=idxawal;i<idxakhir+1;i++){
        str[i]=" ";
    }
    str = str.join('');
    return str;
}
```

Gambar 14. Fungsi untuk mengubah char menjadi spasi dalam javascript

(Sumber : Penulis)

```
const stopwordList = require("./stopwordList.json");
var input = "berita politik yang paling terbaru";
for (let i=0; i<stopwordList["stopword"].length;i++){
    var
borderArr=kmpBorder(stopwordList["stopword"][i]);
    var idxfound =
kmpMatching(stopwordList["stopword"][i],input,borderA
rr)
    if(idxfound.length!=0){
        for(let j=0;j<idxfound.length;j++){

input=replaceCharRange(input,idxfound[j]+1-
stopwordList["stopword"][i].length,idxfound[j]);
        }
    }
}
```

Gambar 15. Contoh algoritma penghapusan stopword menggunakan KMP string matching

(Sumber : Penulis)

Dari semua fungsi di atas, maka akan menghasilkan penghapusan seperti di bawah ini:

```
berita politik      paling terbaru
```

Gambar 16. Output dari potongan kode penghapusan stopword

Setelah itu, input baru bisa ditokenisasi sehingga akan didapat hasil yang lebih sesuai dengan data.

IV. Kesimpulan

Algoritma string matching dapat digunakan dalam menyelesaikan banyak persoalan, seperti plagiarism checker, spell checker, spam filter, dan lain-lain. Algoritma ini mencari kesesuaian antara pattern dengan text. Banyak strategi yang dapat digunakan untuk pencocokan string, seperti brute force, KMP, BM, dan lain-lain. Algoritma ini juga bisa dimodifikasi sesuai kebutuhan, salah satunya pada penghapusan stopword.

V. Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Allah Swt. Karena berkat rahmat dan anugerahnya, penulis dapat menyelesaikan makalah berjudul “Penerapan Algoritma String Matching dalam Penghapusan Stopword” dengan sebaik-baiknya. Penulis juga mengucapkan banyak terima kasih kepada Bapak Rila Mandala selaku dosen mata kuliah IF2211 Strategi Algoritma untuk kelas K01, yang telah membimbing penulis dalam mempelajari mata kuliah ini. Penulis juga mengucapkan banyak terima kasih kepada keluarga dan teman-teman yang telah membantu penulis selama perkuliahan.

Referensi

[1] Munir, Rinaldi, Pencocokan-String-2021 https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf Diakses pada 11 Mei 2021, pukul 13.55 WIB

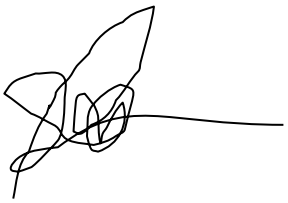
[2] https://www.dosenpendidikan.co.id/contoh-algoritma/ Diakses pada 10 Mei 2021, pukul 15.55 WIB

[3] http://hikaruyuuki.lecture.ub.ac.id/kamus-kata-dasar-dan-stopword-list-bahasa-indonesia/ Diakses pada 11 Mei 2021, pukul 13.55 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Tangerang Selatan, 11 Mei 2021



Muhammad Bintang Pananjung
13519004