Analisisa Pengguna Social Media dengan *Pattern Matching*

Luthfi Ahmad Mujahid Hadiana / 13516051
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung , Indonesia
13516051@stei.itb.ac.id , luthfihadiana@gmail.com

***Abstract*—Saat ini perkembangan dunia maya di dunia sudah berkembang , salah satunya juga media sosial . Di media sosial seseorang dapat lebih mengekspresikan diri karena disini penuh kebebasan . Dengan bantuan algoritma – algoritma pemecahan kasus masalah *pattern matching* , kita dapat melakukan analisis seorang user tergantung dari apa yang kita analisa di postingannya .**

Keywords—Pattern matching; posting ; KMP;Regex; Boyer-moore ;

I. PENDAHULUAN

Pada era ini , Dunia telah mengalami perkembangan yang sangat pesat . Salah satu yang mengalami perkembangan adalah teknologi . Seperti yang kita lihat saat ini , Sudah banyak teknologi yang dahulunya kita anggap itu adalah suatu hal yang mustahil dan sekarang dapat kita jumpai teknologi itu , salah satu teknologi itu adalah Internet . Dengan internet , kita dapat menyambungkan seseorang yang secara geografi tidak berada di dekat kita . Perkembangan internet ini juga muali berkembang pesat semenjak munculnya wifi dan juga smartphone.

Dunia saat ini , terutama untuk orang-orang yang hidup di daerah perkotaan , Internet merupakan bahan dasar pokok yang harus dimiliki terlebih lagi jika orang tersebut menggunakan smartphone . Mengapa terjadi demikian ? Ini dikarenakan pada dunia saat ini , hampir seluruh aspek terhubung dengan Internet . Untuk pengumpulan tugas atau bahkan PR , beberapa dosen atau guru dapat melakukan pengumpulan melalui internet . Untuk bermain game yang *multiplayer* , kita memerlukan internet agar dapat bermain agme tersebut . Kita ingin mencari referensi buku untuk belajar , maka terlintas pertama kali dalam pikiran adalah mencari referensi di internet . Jika kita ingin pergi suatu tempat , saat ini , kita dengan mudah cukup membuka aplikasi transportasi online dan memesannya yang dimana aplikasi ini juga terhubung dengan internet . Saat ini , Hampir seluruhnya terhubung dengan internet begitu pun dunia sosial.

Manusia ini sesungguhnya adalah mahluk sosial . Manusia tidak bisa hidup sendiri , mereka harus bersosialisasi dengan sesama . Seiring dengan berkembang nya internet , Mulai muncul berbagai sebuah media yang dimana media tersebut sebagai tempat bersosialisasi antar pengguna di media tersebut , media tersebut kita kenal sebagai sosial media. Sosial media ini muncul tidak lepas dari manusia sebagai mahluk sosial . Media sosial ini bahkan dapat membuat kita dapat bersosialisasi dengan seseorang yang secara fisik dan geografi berada jauh dari tempat kita berada. Dengan media sosial , kita dapat menghilangkan kendala ruang dan waktu yang dahulunya merupakan halangan bagi seseorang ketika ingin bersosialisasi dengan yang lainnya.

Dengan media sosial , kita dapat mengekspresikan apapun baik itu pemikiran , pendapat , karya kita mengenai apapun dengan secara bebas tanpa mengalami kendala . Dikarenakan tidak ada kendala , disana mereka terlalu bebas bahkan terlampau bebas . Kita mungkin bisa melihat seseorang itu sangat pendiam ketika kita bertemu di dunia asli , namun di media sosial dia adalah seorang yang sangat aktif . Mungkin orang itu mengalami kendala malu tampil di publik umum , tetapi dengan media sosial dia dapat menghilangkan kendala tersebur . Oleh karena itu , kita mungkin dapat melihat apa sisi asli dari seseorang itu melalui kegiatannya di media sosial .

Berdasarkan peristiwa – peristiwa diatas , kita mungkin dapat menganalisis bagaimana seseorang itu mungkin sebenarnya berperilaku dengan melihat apa yang dia lakukan ketika di media sosial . Kita dapat menentukan seseorang itu pecinta bola melalui postingannya di media sosial baik itu apa yang posting , apa yang searching , apa yang dia tonton , siapa yang dia ikuti atau jadikan teman , grup apa saja yang dia ikuti. Kita dapat menganalisis perilaku , kepribadian bahkan kita dapat menentukan dia adalah seorang yang mengganggu kita berdasarkan dari apa yang dia lakukan di media sosial.

Makalah ini mencoba mengaplikasi algoritma *pattern matching* yang merupakan salah satu topik di mata kuliah “Strategi Algoritma” dengan cara analisis user di media sosial dari isi dari konten apa yang dia posting. Masalah ini dapat digolongkan kedalam masalah *pattern matching* dikarenakan , kita ingin melakukan beberapa pencocokan atau pencarian sesuatu di dalam posting dan menentukan bagaimana analisa dari pengguna media sosial tersebut.

II. DASAR TEORI

A. *Social Media*

Sosial media adalah sebuah media untuk bersosialisasi satu sama lain dan dilakukan secara online yang memungkinkan manusia untuk saling berinteraksi tanpa dibatasi ruang dan waktu.

Gambar 1.1 Contoh-contoh Social Media
Sumber: https://makeawebsitehub.com/social-media-sites/

Sosial media dapat dikelompokkan menjadi beberapa bagian besar yaitu :

1. *Social Networks*, media sosial untuk bersosialisasi dan berinteraksi (Facebook, myspace, hi5, Linked in, bebo).
2. *Discuss*, media sosial yang memfasilitasi sekelompok orang untuk melakukan obrolan dan diskusi (google talk, yahoo! M, skype, phorum).
3. *Share*, media sosial yang memfasilitasi kita untuk saling berbagi file, video, music, dll (youtube, slideshare, feedback, flickr, crowdstorm).
4. *Publish*, (wordpress, wikipedia, blog, wikia, dig).
5. *Social game*, media sosial berupa game yang dapat dilakukan atau dimainkan bersama-sama (koongregate, doof, pogo, cafe.com).
6. *MMO* (kartrider, warcraft, neopets, conan, DOTA).
7. *Virtual worlds* (habbo, imvu, starday).
8. *Livecast* (y! Live, blog tv, justin tv, listream tv, livecastr).
9. *Livestream* (socializr, froendsfreed, socialthings!).
10. *Micro blog* (twitter, plurk, pownce, twirxr, plazes, tweetpeek).[1]

B. Twitter

Twitter adalah sebuah media sosial yang tergolong kedalam kategori *Microblog* yang dimana para pengguna dapat melakukan pengiriman dan membaca pesan dengan basis teks yang dimana panjang dari teks ini tidak lebih dari 140 karakter , atau yang biasa kita kenal saat ini sebagai kicauan atau *tweet* .

Twitter didirikan oleh Jack Dorsey pada maret 2006 dan situs twitter ini diluncurkan secara resmi pada bulan Juli 2006. Sejak diluncurkan twitter ini , Twitter menjadi salah satu dari 10 nsitus di Internet yang paling sering dikunjungi bahkan Twitter ini mendapat julukan “pesan singkat Intrernet”.

C. *Pattern Matching*

1) Definisi

Diberikan data sebagai berikut:

a. T : *text* , yaitu sebuah (long) *string* yang panjangnya diketahui yaitu *n*
b. P : *pattern* , yaitu sebuah *string* dengan panjang *m* karakter dan panjang *m* ini diasumsikan selalu lebih kecil dari *m* (*m* <<< *n*) yang dimana *pattern* ini akan dicari didalam *text* .

Carilah sebuah lokasi didalam *text* yang dimana lokasi yang dicari adalah sebuah lokasi *text* yang bersusaian dengan pattern .

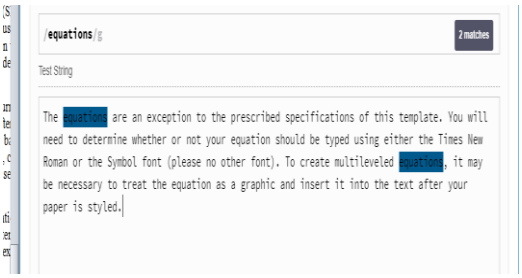
2) Algoritma *Pattern Matching*

Didalam Algoritma Pattern Matching ini , dibagi menjadi 2 jenis algoritma:

a. Exact Matching

Di dalam algoritma *exact matching* , algoritma hanya dapat mencari sebuah lokasi di teks yang dimana lokasi ini harus sama persis dengan dengan *pattern* yang diinginkan. Yang termasuk algoritma tipe ini adalah algoritma Brute-force , KMP dan Boyer-Moore.

Contoh:

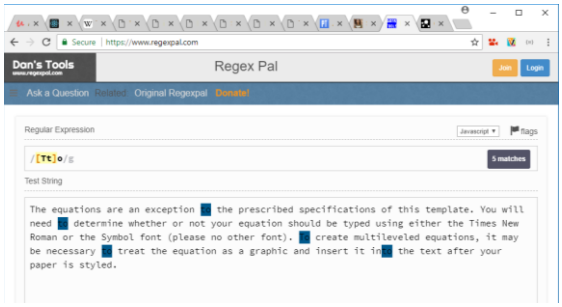


Gambar 1.4 Contoh Exact Matching
Sumber: berkas penulis

b. Regex Matching

Di dalam algoritma *Regex matching* , algoritma mencari sebuah lokasi di teks yang dimana lokasi ini memiliki sebuah pola regex atau maksud dari sebuah kata yang sama dengan *pattern* yang diinginkan.

Contoh:

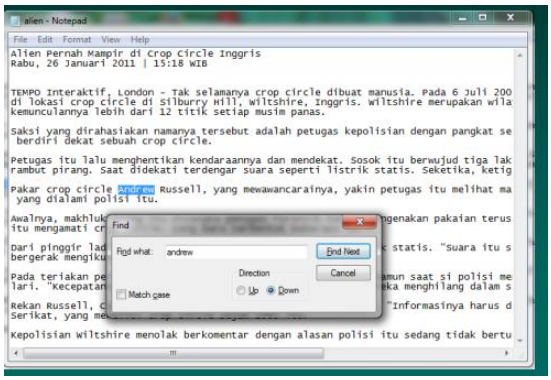


Gambar 1.5 Contoh Regex Matching
Sumber: berkas penulis

Algoritma *Pattern Matching* ini diaplikasikan ke dalam beberapa Aplikasi , contohnya :

a. Pencarian di dalam *Editor Text*

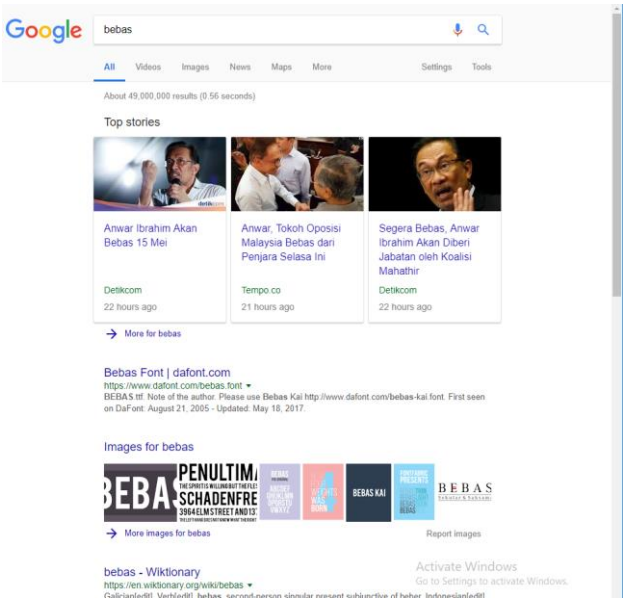
Di dalam *Editor Text* , Kita terkadang ingin melakukan sebuah pencarian kata di dalam Text kita. Agak menyulitkan jika kita melakukan pencarian semua kata itu dengan secara manual . Oleh karena itu , Hampir semua *Editor Text* yang ada saat ini menyediakan fitur *Find*. Fitur *Find* ini menggunakan *Pattern Matching* untuk menemukan sebuah kata di dalam teks ini.



Gambar 1.6 Pencarian di dalam Editor Text
Sumber: berkas penulis

b. *Web Search engine*

Dalam mencari lama page di search engine , cara kita mencari web page yang kita inginkan adalah dengan cara memasukkan sebuah kata di dalam sebuah input text . *Search engine* akan melakukan pencarian yang dimana konten dari sebuah laman tersebut memiliki kesesuaian dengan kata yang sudah diinputkan oleh pengguna . Pencarian di dalam *Search engine* ini dilakukan dengan cara *pattern matching*.



Gambar 1.7 Web Search Engine
Sumber: berkas penulis

c. *Plagiarism Detection*

Plagiarism adalah sebuah perbuatan seseorang dengan mencopy kerjaan dari seseorang dan mengaku pekerjaaan itu merupakan kerjaan dirinya. Cara untuk melakukan *Plagiarism Detection* adalah melakukan *pattern matching* antara satu file dengan sebuah file yang dicurigai sebagai plagiat. Jika kesamaan dari kedua file ini ada di luar batas wajar , file yang diperiksa tadi kita *suspect* sebagai file yang *plagiarism.*

d. *Bionformatics*

Bionformatics adalah sebuah aplikasi informasi teknologi dan komputer science untuk menyelesaikan permasalahan di cabang ilmu Biologi. Salah satu masalah yang dipecahkan dengan *pattern matching* adalah pencocokan sebuah DNA. Untuk melakukan sebuah pencocokan antara satu DNA dengan DNA lain adalah mengambil salah satu *sequence* di dalam DNA dan dicocokan dengan DNA. Jika ditemukan kecocokan , Kita bisa sebutkan bahwa dua induk ini memiliki kemiripan.

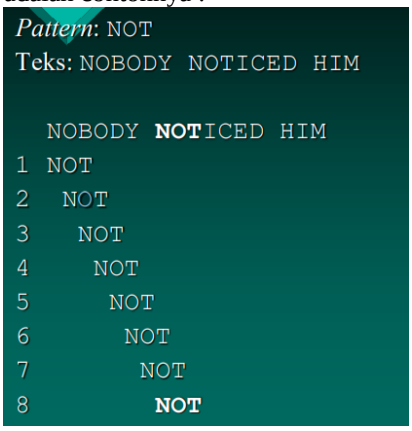
Gambar 1.8 Program Pencocokan DNA
Sumber: screenshot dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

Berikut adalah penjelasan – penjelasan algoritma yang digunakan untuk menyelesaikan sebuah permasalahan *pattern matching :*

a) Brute-Force

Dengan algoritma ini , kita akan melakukan pengecekan di setiap karakter di *pattern* dari index terkecil sampai akhir . Jika semua cocok dengan kata dari *pattern* , *pattern* ditemukan di dalam teks tersebut.Pengecekan di mulai dari index terkecil dari teks tersebut. Jika tidak cocok , index pengecekan akan digeser . Proses ini akan terus dilakukan hingga menemukan kecocokan atau hingga indeks pengecekan mencapai *panjang(pattern)-1*.

Berikut adalah contohnya :



Gambar 1.10 Brute Force Pattern Matching
Sumber: screenshot dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

Berikut adalah implementasi kode dari algoritma *brute-force* dengan bahasa java:

```
public static int brute(String text,String
pattern){
   int n = text.length(); // n is length of
text
   int m = pattern.length(); // m is length
of pattern
   int j;
   for(int i=0; i <= (n-m); i++) {
        j = 0;
        while ((j < m) && (text.charAt(i+j)==
        pattern.charAt(j)) ) {
                j++;
        }
        if (j == m)  return i; // match at i
  }
 return -1; // no match
}
```

Performansi dari algoritma ini adalah :

- *Worst case*

  Case ini terjadi jika suatu pattern ditemukan ketidakcocokannya di akhir pattern dan hampir semua kasus tersebut terjadi di setiap pengecekan dengan teks. Jumlah perbandingan dari kasus ini adalah m(n-m+1). Jadi , kompleksitas dari algoritma ini adalah O(mn).
- *Best case*

  Terjadi bila karakter pertama pattern P tidak pernah sama dengan karakter teks T yang dicocokkan. Kompleksitas kasus terbaik adalah O(n).
- *Average case*

  Kasus umum ini rata-rata memiliki kompleksitas O(m+n).

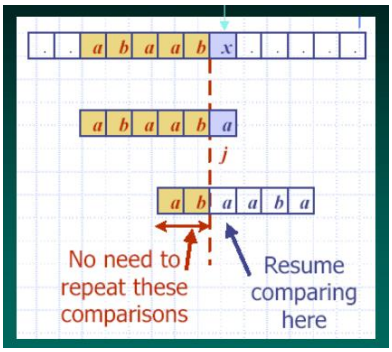
b) Knuth-Morris-Pratt (KMP)

Algoritma ini konsep nya adalah sama seperti algoritma *brute-force* , yaitu mencocokan *pattern* dengan text dengan melakukan pergeseran. Perbedaannya terletak di bagaimana algoritma ini untuk melakukan pergeseran dan pergeseran dilakukan dengan cara yang lebih cerdas.

Pergeseran pada algoritma ini dilakukan berdasarkan *Border-Function*. *Border Function* (b(k))adalah panjang terbesar dari prefix pattern (P[0..k]) juga dari suffix pattern (P[1..k]). Jika kita bilang kita menemukan ketidakcocokan di indeks *j* , maka *k* adalah *j-1* . Border function ini akan dibuat sebelum kita melakukan algoritma. Untuk pengisian dari *BorderFunction* adalah kita ambil contoh ambil j dari pattern di gambar 1.12 nya adalah 5. Nilai k-nya adalah 4. Kita akan mencari panjang terbesar yang dimana suffix dan prefixnya sama dari pattern[0..k](“abaab”). Berikut adalah pencariannya :

- Suffix : “a” prefix:”b” panjang : 1 (tidak sama)
- Suffix : “ab” prefix:”ab” panjang : 2 (sama)
- Suffix : “aba” prefix:”aab” panjang : 3 (tidak sama)
- Suffix : “abaa” prefix:”baab” panjang : 4 (tidak sama)

Maka b[k] akan diisi dengan nilai 2.



Gambar 1.11 KMP Algorithm
Sumber: screenshot dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

						(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

Gambar 1.12 Border Function
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

Berikut ini adalah implementasi dari algoritma KMP dengan bahasa Java :

```
public static int kmpMatch(String text, String
pattern){
        int n = text.length();
         int m = pattern.length();
        int fail[] = computeFail(pattern);
        int i=0;
        int j=0;
        while (i < n) {
           if (pattern.charAt(j) ==
text.charAt(i)) {
                if (j == m - 1) return i - m + 1;
        // match
                i++;
                j++;
           } else if (j > 0) j = fail[j-1];
           else i++;
        }
        return -1; // no match
} // end of kmpMatch()
public static int[] computeFail( String pattern) {
        int fail[] = new int[pattern.length()];
        fail[0] = 0;
        int m = pattern.length();
        int j = 0;
        int i = 1;
        while (i < m) {
        if (pattern.charAt(j) ==
pattern.charAt(i)) { //j+1
                chars match fail[i] = j + 1;
                i++;
                j++;
        } else if (j > 0) // j follows matching
prefix
                j = fail[j-1];
        else { // no match
                fail[i] = 0; i++;
        }
        }
        return fail;
```

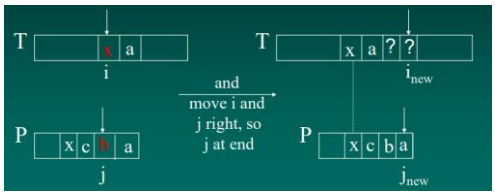
Performansi KMP :

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

c) Boyer-Moore

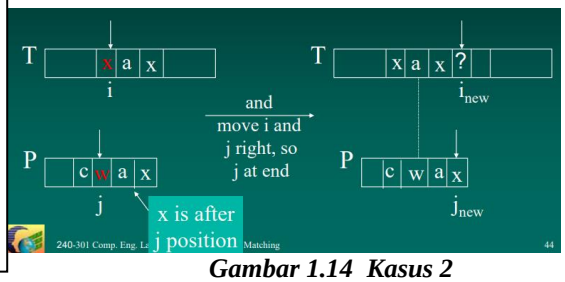
Algoritma ini memiliki konsep yang tidak sama dengan algoritma *brute-force* maupun *KMP*. Jika kedua algoritma sebelumnya caranya adalah dengan mencocokan *pattern* dengan text dengan melakukan pergeseran , Algoritma ini menggunakan dua 2 teknik untuk *pattern matching*:

- *Looking-glass technique*
  Menemukan Pattern di teks dengan cara *Backwards* melaui Pattern dari mulai hingga akhir.
- *Character-jump technique*
  Jika ada ketidakcocokan antara karakter di pattern dan teks (T[i] != P[j] and T[i]==x), maka nantinya akan dibagi menjadi 3 kasus :
  - Kasus 1
    Jika suatu pattern mengandung karakter x di suatu tempat , lakukan pergeseran pattern menuju ke tempat dimana elemen x ini terakhir ditemukan di pattern.



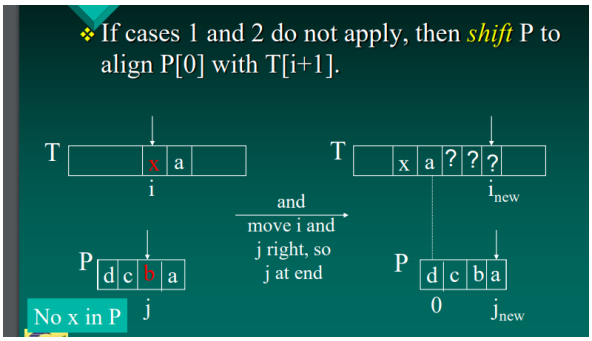
Gambar 1.13 Kasus 1
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

  - Kasus 2
    Jika suatu pattern mengandung karakter x di suatu tempat , tetapi melakukan pergeseran menuju ke tempat dimana elemen itu ditemukan tidak memungkinkan , lakukan pergeseran ke kanan sejauh 1 karakter.

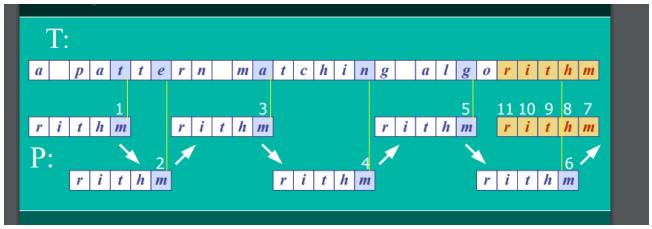


Gambar 1.14 Kasus 2
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf

  - Kasus 3
    Jika tidak memenuhi kasus 1 dan 2 maka , lakukan pergeseran pattern menuju i+1.



Gambar 1.14 Kasus 2
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/
2017-2018/Pencocokan-String-(2018).pdf



Gambar 1.13 Kasus 3
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/
2017-2018/Pencocokan-String-(2018).pdf

Berikut ini adalah implementasi dari algoritma Bayer-Moore dengan bahasa Java :

```
public static int bmMatch(String text, String pattern) {
        int last[] = buildLast(pattern);
        int n = text.length();
        int m = pattern.length();
        int i = m-1;
        if (i > n-1) return -1; // no match if pattern
is
                                // longer than text :
        int j = m-1;
        do {
                if (pattern.charAt(j) == text.charAt(i))
                    if (j == 0) return i; // match
                    else { // looking-glass technique
                            i--;
                            j--;
                    }
                else { // character jump technique
                int lo = last[text.charAt(i)]; //last
                occ
                i = i + m - Math.min(j, 1+lo);
                j = m - 1;
                }
        } while (i <= n-1); return -1; // no match
} // end of bmMatch()
public static int[] buildLast(String pattern){
        int last[] = new int[128]; // ASCII char set
        for(int i=0; i < 128; i++) last[i] = -1; //
initialize
        for (int i = 0; i < pattern.length(); i++)
                last[pattern.charAt(i)] = i;
        return last;
} // end of buildLast()
```

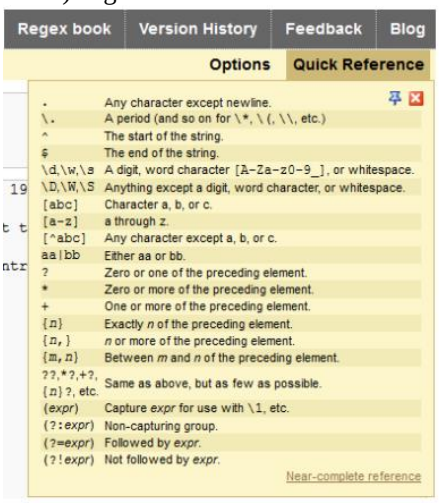
Performansi Bayer-Moore :

- Kasus Boyer-Moore terburuk dari algoritma ini adalah O(nm + A).

d) Regular Expressions

Regular Expression atau yang lebih sering disebut regex merupakan sebuah teknik yang digunakan untuk mencocokan string teks, seperti karakter tertentu, kata-kata, atau pola karakter.

Engine RegEx terdiri dari 2 jenis Text-directed engine dan regex-directed engine atau ada juga yang mengatakan DFA (Deterministic Finite Automaton) dan NFA (Nondeterministic Finite Automaton) engines



Gambar 1.14 Kasus 2
Sumber: screenshot dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/
2017-2018/Pencocokan-String-(2018).pdf

III. PEMBAHASAN

Analisis pengguna media sosial akan kita analisis dari hasil *pattern matching* dari apa yang pengguna postingkan . Dengan *pattern matching* , kita dapat menentukan apa saja yang terdapat di dalam postingan orang tersebut . Sebuah postingan ini nantinya akan di-*pattern-matching*-kan dengan sebuah *keyword*. *Keyword* ini merupakan sesuatuyang sangat penting karena ini akan menjadi penentu dari analisis kita . Semisalnya , kita memiliki sebuah *keyword* "liverpool" , kita akan melakukan *pattern matching* dengan *keyword* tersebut terhadap postingan seseorang. Jika mayoritas dari semua postingan yang kita analisis ini menunjukkan tanda kecocokan yang tinggi (hampir seluruh postingannya mengandung keyword tersebut) , kita dapat menyimpulkan bahwa orang ini adalah seorang fans dari klub liverpool atau juga kita dapat mengatakan bahwa orang ini bertempat tinggal di liverpool atau mungkin juga kita dapat analisis bahwa orang itu sedang melakukan liburan di liverpool .

Dalam analisis kali ini , saya tertarik untuk meneliti seseorang itu adalah apakah seorang spammer atau bukan.Apa itu spammer ? Spam adalah sebuah terminologi baru yang tercipta di belantara dunia maya. Kamus Oxford mengartikannya sebagai sebuah pesan yang tidak relevan, tidak berguna atau tidak diminta, yang dikirim lewat internet, biasanya langsung ke sejumlah besar pengguna, untuk kepentingan iklan, pengelabuan (penipuan), untuk

menyebarkan perusak perangkat dan sebagainya. Orang yang sering menyebar spam disebut spammer .

Dalam kasus ini , saya mencoba melakukan analisis seorang spammer yang memiliki tujuan untuk promosi atau komnersil. Pesan yang terkandung dalam spam cenderung berbentuk penawaran produk berupa barang, jasa atau layanan. Pelakunya sendiri memang tak terbatas pihak-pihak yang tidak bertanggung jawab saja, pihak operator layanan misalnya operator sesluler pun terkadang mengirimkan pesan-pesan yang menurut saya kurang bermanfaat dengan frekuansi yang diluar nalar. Bayangkan saja jika sebuah layanan operator seluler mengirimkan pesan dengan embel-embel pelayanan lebih dari lima kali dalam tempo 24 jam .

Untuk mementukan apakah orang ini adalah seorang spammer atau bukan , mencari sebuah keyword yang cocok dengan yang kita bisa kategori kan sebagai spam . Dalam analisa saya saat ini , saya mencoba mengambil beberapa tweet dari seseorang tersebut (dalam hal ini saya mencoba mengambil sekitar 20 postingan di media sosial) . Jika kita lakukan pattern matching dan dominan isi tweet dari dia adalah sebuah spam , maka kita patut mencurigai bahwa user itu adalah spammer . Sesuai yang kita sering ketahui , untuk mengetahui sesuatu itu adalah sesuatu yang dominan , saya mencoba meniru metode pemilihan seseorang . Di dalam metode pemilihan seseorang ketika menjadi seorang pemimpin , kita sering mengatakan bahwa calon pemimpin itu memenangkan sebuah pemilihan jika suara para pemilih lebih dominan dibandingkan calon lainnya . Biasanya , jumlah suara yang ditentukan adalah ketika suatu suara mencapai 50% +1. Dengan metode yang sama saya akan menghitung jumlah postingan dari seorang pengguna yang dimana jika terhitung didalam posting ini isinya spam dan terhitung melebihi 50% isi tweetnya , maka kita bisa kategorikan bahwa orang yang kita sedang analisis ini adalah seorang spammer.

Dalam kasus ini juga saya mencoba meneliti user di media sosial Twitter . Kenapa menggunakan twitter ? alasan pertama adalah karena penggunaan API untuk twitter yang terlampau tidak menyulitkan , lalu alasan kedua saya menggunakan twitter untuk diteliti adalah karena di twitter ini , isi postingan atau tweet isinya cukup sederhana dan memilki batasan jumlah kata di postnya yaitu sekitar 145 karakter / tweet . Hal itu tentu membantu karena jika suatu posting itu terlalu banyak kata-katanya , maka pemrosesan terhadap suatu data yang ada di dalam program akan menjadi lebih rumit dan kompelksitasnya pun menaik. Dengan pembatasan ini , saya dapat meneliti dengan lebih mudah.

Untuk melihat suatu tweet itu adalah spam atau bukan maka kita dapat melihat bagaimana jenis kalimat yang digunakan di dalam postingan . Untuk spam yang bertujuan promosi , tweet tersebut harus memiliki sebuah kalimat ajakan. Kalimat ajakan tentu saja bertujuan untuk mengajak orang lain , berdasarkan https://brainly.co.id/tugas/2316503 , kalimat ajakan ini singkat , jelas dan mudah di mengerti dan ada beberapa kata yang sering digunakan contohnya “Ayo bla bla” , “Ayo Klik link ini” , dan lain –lainnya . Selain dari kita lihat dari jenis kalimatnya , kita juga dapat mengatakan itu adalah spam biasanya mengadung link-link . Dengan begitu kita dapat melihat bahwa ciri spam adalah memiliki keyword kata-kata untuk link seperti “https” atau mungkin memiliki kata- kata yang bersifat mengajak seperti “Ayo” . Tetapi spam tidak sebatas itu , terkadang kita juga bisa menganggap suatu yang tweet yang di posting seseorang itu *meaningless.* Ambil contoh bahwa kita mengganggap bahwa jika kata “JKT48” atau “AKB48” adalah sebuah spam .

Berikut adalah implementasi dengan bahasa python untuk decision bahwa dia adalah spammer atau bukan :

```
import re
import sys

def keyword_spam(text):
# mengembalikan 1 jika ditemukan keyword spam pada postingan
# mengembalikan -1 jika tidak
    patterns = [r'^[hH][tT]{2}[Pp][Ss]', r'[wW][aA][dD][uU][hH]',
r'[hH][eE][yY]', r'[Ww][eEiI][bBeE][bBUu]', r'[A-Z][A-Z][A-
Z]48', r'[Mm][Aa][Rr][Ii][Kk][Aa]']
    for pattern in patterns :
        if (bool(re.search(pattern,text))) :
            return 1
    return -1

# main program
print(keyword_spam(sys.argv[1]))
```

Program ini mengganggap bahwa untuk sebuah tweet itu mengandung pola kata “https” , “waduh” , apapun kata akhiran “48” , “hey” , “wibu” adalah sebuah spam , untuk *pattern matching* menggunakan *Regular expression.* Selain itu kita juga membuat pendetksian dengan *excact matching* berikut adalah implementasi dengan python algoritma *Boyer-Moore* dan *KMP* :

Boyer-Moore

```
import sys
def generate_d_vector(text,pattern):
    d = {}
    for char in text:
        founded = pattern.rfind(char)
        if char not in d:
            d[char] = len(pattern)-1-pattern.rfind(char) if
founded != -1 else len(pattern)
    return d

def boyer_moore(text,pattern,d):
    j = len(pattern)-1
    while j<len(text):
        i = len(pattern)-1
        while i>0 and pattern[i]==text[j]:
            i,j = i-1,j-1
        if i==0: return True
        else:
            if len(pattern)-1-i>d[text[j]]: j = j + len(pattern)-1- i
+ 1
            else: j = j + d[text[j]]
    return False
```

KMP

```
import sys
def generate_d_vector(text,pattern):
    d = {}
    for char in text:
        founded = pattern.rfind(char)
        if char not in d:
            d[char] = len(pattern)-1-pattern.rfind(char) if
founded != -1 else len(pattern)
    return d

def boyer_moore(text,pattern,d):
    j = len(pattern)-1
    while j<len(text):
        i = len(pattern)-1
        while i>0 and pattern[i]==text[j]:
            i,j = i-1,j-1
        if i==0: return True
        else:
            if len(pattern)-1-i>d[text[j]]: j = j + len(pattern)-1- i
+ 1
            else: j = j + d[text[j]]
    return False
```

Algoritma untuk pendeteksian :

```
def keyword_spam(text):
# mengembalikan 1 jika ditemukan keyword spam
pada postingan
# mengembalikan -1 jika tidak
    patterns = ['https', 'waduh', 'hey', 'wibu', 'AKB48']
    for pattern in patterns :
        if
(boyer_moore(text,pattern,generate_d_vector(text,p
attern))) :
            return 1
    return -1

def keyword_spam(text):
# mengembalikan 1 jika ditemukan keyword spam
pada postingan
# mengembalikan -1 jika tidak
    patterns = ['https', 'waduh', 'hey', 'wibu', 'AKB48']
    for pattern in patterns :
        if (KMP(text,pattern)) :
            return 1
    return -1
```

Berikut adalah implementasi perhitungan *tweet* spam dengan bahasa PHP :

```
    $statuses = $connection->get("statuses/user_timeline",
["screen_name"=>$_POST["userName"],"count" =>
$numTweet, "exclude_replies" => true]);

foreach ($statuses as $iter){
        $flagSpam = exec("python RE_spammerDetector.py
\"$iter->text\"");
        array_push($arSpam,$flagSpam);
        if($flagSpam != "-1"){
            $spamCount ++;
        }
    }
    $numTweet = count($arSpam);
    if($numTweet % 2 == 0){
        if($spamCount >= ($numTweet/2)){
            $category = "Spammer";
            $colorCount = "Red";
        }else{
            $category = "Clean";
            $colorCount = "#008CBA";
        }
    }else{
        if($spamCount >= (($numTweet+1)/2)){
            $category = "Spammer";
            $colorCount = "Red";
        }else{
            $category = "Clean";
            $colorCount = "#008CBA";
        }
    }
```

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

IV. KESIMPULAN

Beberapa algoritma yang sudah ada dapat membantu kita untuk menyelesaikan permasalahan *pattern matching* ini . Setiap algoritma memiliki kelebihan dan keurangannya masing-masing . Penggunaan algoritma yang tepat tentu akan membuat performansi program kita lebih mangkus .

Salah satu persoalan mengenai *pattern matching* ini adalah permasalahan analisa user media sosial . Dalam makalah ini kita dapat menyimpulkan bahwa salah seorang user itu adalah seorang spammer jika dominansi posting yang dilakukan oleh user adalah sebuah spam . Namun , untuk makalah ini kita baru bisa mencurigai bukan menetapkan hasil analisa user . Untuk perhitungan dominansi ini kita dibantu oleh algoritma – algoritma mengenai *pattern matching*.

UCAPAN TERIMA KASIH

Pertama penulis ingin mengucapkan puji syukur kepada Allah SWT karena dengan rahmat dan karunai-Nya saya sebagai penulis dapat menyelesaikan makalah dengan judul "Analisis Pengguna Media Sosial dengan *Pattern Matching*" ini dengan baik. Penulis juga berterima kasih kepada para dosen pengajar mata kuliah IF2211 Strategi Algoritma, Dr. Ir. Rinaldi Munir, M.T.,Masayu Leylia Khodra, S.T., M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M. Sc., atas bimbingan mereka selama ini dalam mengajar dan memberikan ilmu sehingga penulis mampu membuat makalah ini. Penulis juga berterima kasih kepada rekan-rekan yang telah memberikan semangat dan dorongan kepada penulis..

REFERENSI

[1] https://www.unpas.ac.id/apa-itu-sosial-media/ diakses 13 Mei 2018 , Jam 13.23 .

[2] http://web.archive.org/web/20110715062407/www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf. diakses 13 Mei 2018 , Jam 13.58 .

[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf. Diakses pada 13 Mei 2018 pukul 17.00.

[4] http://arcanerain7.blogspot.co.id/2010/04/apa-itu-regular-expression-regex.html. Diakses pada 13 Mei 2018 pukul 18.44.

PERNYATAAN

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

Bandung, 29 April 2012

Luthfi Ahmad MH / 13516051