Laporan Tugas Besar III IF2211 Strategi Algoritma

Deteksi *Spam* pada Media Sosial atau *Chat-Messenger* dengan Algoritma Pencocokan String

SEMESTER II 2017-2018

Kelompok Sahupili :

Yuly Haruka Berliana Gunawan 13516031

Ahmad Faiz Sahupala

Yasya Rusyda Aslina 13516091

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

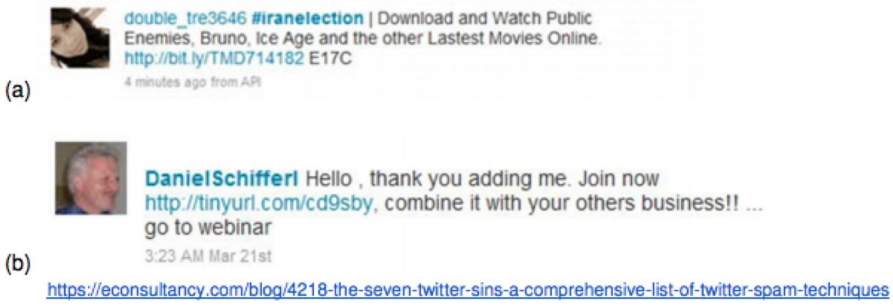
PROGRAM STUDI INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2018

BAB I
DESKRIPSI MASALAH

Electronic-spam merupakan pesan elektronik yang tidak diinginkan penerimanya, bisa dalam bentuk surat elektronik, SMS, posting atau komentar di media sosial yang muncul di timeline kita, ataupun pesan pada chatmessenger. Spammer melakukan spamming untuk tujuan tertentu, paling banyak untuk menyebarkan iklan. Penentuan spam sangat bersifat subjektif, artinya spam untuk kita, belum tentu spam untuk pengguna lain. Gambar 1-2 merupakan contoh spam pada twitter, facebook, dan line.



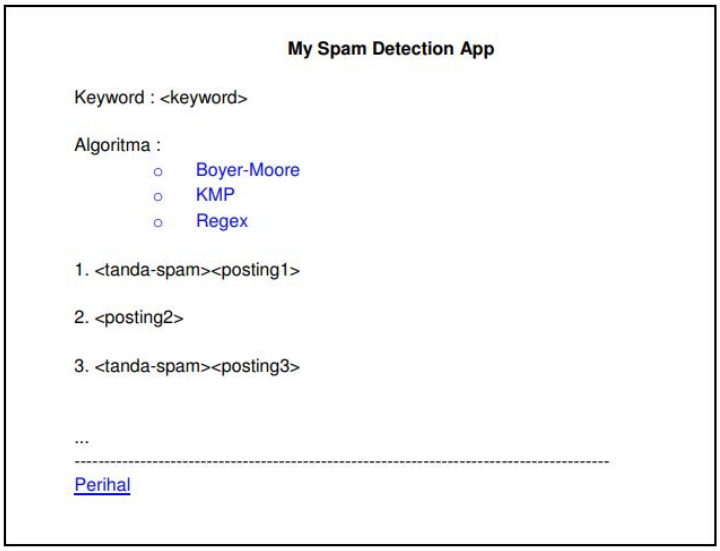
Gambar 1. Contoh spam pada media sosial *twitter*:
(a) hashtag/trending topic spam; (b) autoresponder spam

Gambar 2. Contoh *spam* pada media sosial *facebook*

Algoritma pencocokan string (pattern) Knuth-Morris-Pratt (KMP) dan Algoritma BoyerMoore merupakan algoritma yang lebih baik daripada brute force. Pada Tugas Besar III kali ini Anda diminta membuat aplikasi sederhana deteksi spam pada media sosial dengan kedua algoritma tersebut, plus menggunakan regular expression (regex). Teks yang akan Anda proses adalah posting berbahasa Indonesia. Pengguna aplikasi ini akan memberikan masukan berupa keyword spam, dan menandai daftar posting yang dikategorikan spam terurut berdasarkan tanggal. Pencocokan string yang anda buat adalah exact matching (untuk KMP dan BM) jadi posting yang diproses mengandung string yang tepat sama dengan keyword spam dari pengguna. Sedangkan bila menggunakan regex maka tidak selalu exact matching. Pencarian juga tidak bersifat case sensitive, jadi huruf besar dan huruf kecil dianggap sama (hal ini dapat dilakukan dengan mengganggap seluruh karakter di dalam pattern dan teks sebagai huruf kecil semua atau huruf kapital semua). Kumpulan posting diambil secara otomatis menggunakan Facebook API atau Twitter API (https://developer.twitter.com/en/docs/tweets/search/overview) atau Line API (https://developers.line.me/en/services/messaging-api/) atau api dari media sosial lainnya.

Spesifikasi program:

1. Aplikasi deteksi spam yang anda buat merupakan aplikasi berbasis web yang menerima keyword pencarian, misalnya “klik link ini”. Tampilan antarmuka pengguna-komputer kira-kira seperti Gambar 3 di bawah ini:



Perihal: pranala ke halaman tentang program dan pembuatnya. Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya

2. Deteksi spam menggunakan hasil implementasi algoritma KMP, Boyer-Moore, dan Regex dengan menggunakan Bahasa C#, Java, Python, Golang. Pencocokan string dilakukan pada konten posting.

BAB II
DASAR TEORI

2. 1 Algoritma Knutt Morris-Pratt

Algoritma Knutt Morris-Pratt merupakan salah satu algoritma yang digunakan untuk pencarian string (*pattern*). Algoritma ini baik dalam segi kompleksitas maupun segi kecepatan lebih baik dibandingkan dengan algoritma pencarian string (pattern) *bruteforce.* Algoritma ini melakukan pencarian string dari kiri ke kanan seperti *bruteforce*, tetapi pergeseran string nya yang berbeda dengan *bruteforce*. Pergeseran string pada algoritma KMP berguna untuk menghindari pencocokan karakter yang tidak diperlukan pada algoritma *bruteforce*.

Jadi pada algoritma KMP ini, terdapat 3 *array* karakter yaitu, *array* yang menyimpan string *pattern*, *array* yang menyimpan teks tempat pencarian string *pattern,* dan *array* yang menyimpan pergeseran string pada saat dicocokan. *Array* pergeseran string tersebut dibuat dengan mencocokan *prefix* dengan karakter-karakter yang ada pada *string pattern.* Pada saat ada urutan karakter yang sama dengan urutan karakter pada *prefix*, maka *array* pergeseran string akan menginput dengan nilai indeks karakter pada *prefix.*

Karena kompleksitas KMP sangatlah cepat dan tidak ada proses *backwards* pada KMP, maka KMP sangatlah cocok untuk digunakan dalam memproses file yang ukurannya besar. Kelemahan dari algoritma KMP ini sendiri ialah ketika jangkauan alfabetnya cukup besar. Hal itu menyebabkan semakin besarnya kemungkinan karakter yang tidak sama dengan prefix sehingga *array* pergeseran string menjadi tidak optimal dalam menginput nilai pergeserannya.

2.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan salah satu algoritma yang digunakan utnuk pencarian stirng (*pattern*). Algoritma ini tentunya lebih baik dibandingkan dengan algoritma *bruteforce* dalam segi kompleksitas. Algoritma ini sendiri mempunyai 2 teknik penyelesaian yaitu *looking-glass* dan *character-jump.*

Teknik *looking-glass* merupakan teknik yang menelusuri string *pattern* dari belakang dan mencocokannya dengan teks. Secara sistematis, langkah-langkah yang dilakukan algoritma *Boyer-Moore* pada saat mencocokkan string adalah:

1. Algoritma *Boyer-Moore* mulai mencocokkan pattern pada awal teks.
2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
    a. Karakter di pattern dan di teks yang dibandingkan tidak cocok (*mismatch*).
    b. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser *pattern* dengan memaksimalkan nilai penggeseran *good-suffix* dan penggeseran *bad-character*, lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks.

2.3 Algoritma Regex

Regular Expression, sering ditulis/disebut juga *Regex / Regexp*, adalah deretan karakter spesial yang mendefinisikan sebuah pola dalam pencarian teks. Regex dapat menyederhanakan pencarian teks yang rumit. Pencarian teks akan dilakukan berdasarkan pola yang kita tentukan. Pemrosesan pola regex menggunakan konsep *finite state machines.* Transisi state dapat berjalan dengan masukan string teks. Teks tersebut akan dicocokkan dengan pola yang telah didefinisikan dengan *finite state*.

BAB III

ANALISIS PEMECAHAN MASALAH

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4. 1 Spesifikasi Teknis Program

Program ini terdiri dari beberapa file, seperti :

a. Data.txt

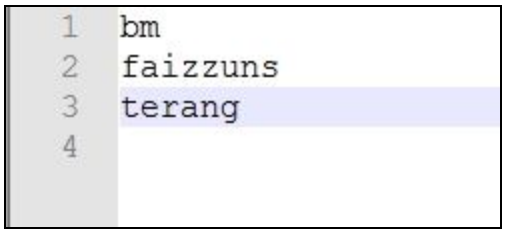
File ini berisikan data JSON yang berisi list dari konten konten yang akan dibuat ketika kode twitter.py dieksekusi. File ini akan menjadi acuan pada fle spam_detector.php untuk menangkap data data yang telah kompilasi pada file twitter.py, seperti username, tweet, dan penanda bahwa ia sebuah tweet spam atau bukan.

```
{
    "text": [
        {
            "test": "habis gelap, terbitlah ngoding",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "@rindanrh azka bersisik",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "\"Idea is cheap, execution everything\" #TIAchapterBDG",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "Stlh kita buat startup,Bagaimana menghadapi orang indonesia yang tidak percaya dengan made in indonesia?\n@1000startupID #1000startupdigital",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "@Rifkihizbullah udah kesana aja secepatnya",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "@Rifkihizbullah  ayo bar",
            "spam": false,
            "username": "faizzunS"
        },
        {
            "test": "@FriscilliaS hati hati ya yun :)",
            "spam": false,
```


b. input.txt

File ini berisi tiga baris. File ini akan terbentuk ketika button search pada *web* di klik (spam_detector.php). File ini berisi tiga baris, dengan baris pertama merupakan kategori pencarian algoritma (kmp/bm/regex). Baris selanjutnya merupakan username twitter yang tweet tweetnya akan dicek. Baris Terakhir, merupakan sebuah keyword yang

menyatakan spam yang akan dicari. Lalu file inilah yang akan menjadi acuan bagi program twitter.py untuk mengeksekusi algoritma yang telah dipilih dan mengambil data tweet yang di inginkan.



c. Twitter.py

File ini berisi potongan program untuk mengakses algoritma KMP, BM, dan Regex. Pada File ini, pertama-tama, file akan membaca data dari input.txt dan mengakses API Twitter dari data yang sudah di ambil.

```
if __name__ == "__main__":
    f=open("input.txt", "r")
    f1 = f.read().split('\n')
    print (f1)
    category = f1[0]
    searchKey = f1[1]
    spam = f1[2]
    query = api.user_timeline(screen_name=searchKey, count = 1000)
```

Lalu, mengecek algoritma mana yang akan dipakai mengacu pada variabel *category*. Setelah itu melakukan iterasi dan pengecekan apakah tweet tersebut spam atau tidak dari semua data query yang telah diambil sebelumnya. Lalu menuliskan hasilnya ke data.txt

```
if (category == "kmp") :
    for tweet in query:
        sp = False
        if (KMP(spam.lower(), tweet.text.lower())) :
            sp = True
        data['text'].append({'test':tweet.text, 'spam':sp, 'username':tweet.user.screen_name})
        with open('data.txt', "w+") as outfile:
            json.dump(data, outfile)
elif (category == "bm") :
    for tweet in query:
        sp = False
        if (boyer(tweet.text, spam) != -1) :
            sp = True
        data['text'].append({'test':tweet.text, 'spam':sp, 'username':tweet.user.screen_name})
        with open('data.txt', "w+") as outfile:
            json.dump(data, outfile)
elif (category == "regex") :
    for tweet in query:
        sp = False
        if (re.search( spam, tweet.text, re.I)) :
            sp = True
        data['text'].append({'test':tweet.text, 'spam':sp, 'username':tweet.user.screen_name})
        with open('data.txt', "w+") as outfile:
            json.dump(data, outfile)
```

Berikut merupakan fungsi yang mengandung algoritma utama pencarian string. Pencarian string dapat dilakukan dengan mengimplementasi metode *boyer, kmp,* atau *regex.*

Algoritma KMP dijalankan dengan melihat prefix terpanjang yang sama dengan suffix sebuah pattern. Algoritma ini akan mengembalikan *true* jika ditemukan kata-kata spam, sedangkan *false* jika tidak ditemukan.

```
def arrayFail(pattern) :
  length = len(pattern)
  fail = [0 for x in range(length)]
  fail[0] = 0
  j = 0
  i = 1
  for i in range(length) :
            if (pattern[j]==pattern[i]) :
            fail[i] = j+1
            i += 1
            j += 1
            elif (j > 0) :
            j = fail[j-1]
            else :
            fail[i] = 0
            i += 1
  return fail
```

```
def KMP(pattern, text) :
  patternLength = len(pattern)
  textLength = len(text)

  fail = []
  fail = fail + arrayFail(pattern)
  i = 0
  j = 0

  for i in range(textLength) :
            if (pattern[j] == text[i]) :
            if (j == (patternLength-1)) :
            return True  # match the pattern
            i += 1
            j += 1
            elif (j > 0) :
            j = fail[j-1]
            else :
            i += 1
  return False  # no match at all
```

Pilihan kedua, kita bisa menggunakan *Boyer-Moore*. Pada algoritma ini akan dilihat indeks pada teks yang menyebabkan *pattern* dan teks *mismatch.* Karakter pada indeks tersebut akan dicari kesamaannya dengan karakter pada indeks *pattern* yang belum dibandingkan. Jika ada kesamaan huruf, perbandingan dilakukan dengan menggeser *pattern* hingga huruf yang memiliki kesamaan bisa dibandingkan.

Selain itu, jika tidak ditemukan huruf yang memiliki kesamaan, perbandingan dilakukan dengan menggeser *pattern* sejauh banyaknya indeks pada *pattern*. Pergeseran ini dilakukan dengan menggunakan fungsi *last occurence*. Algoritma ini akan menghasilkan indeks pertama ditemukannya kata-kata spam dan -1 jika tidak ditemukan.

```
def buildLast(text,pattern):
        last = {}

        for i in range(len(text)):
        last[text[i]]=-1

        for i in range(len(pattern)):
        last[pattern[i]] = i
        return last
```

```
def boyer(teks,pattern):
        text= teks.lower()
        last = buildLast(text,pattern)
        n = len(text)
        m = len(pattern)
        i = m-1

        if ( i> n-1):
        return -1 # gaada pattern yang cocok karena
panjang pattern > text

        j = m-1
        if (pattern[j] == text[i]):
        if (j==0):
        return i
        else:
        i= i-1
        j= j-1
        else:
        lo = last[text[i]]
        i = i+m - min(j, 1 + lo)
        j = m-1
        while (i<= n-1):
        if (pattern[j] == text[i]):
        if (j==0):
        return i
        else:
        i= i-1
        j= j-1
        else:
        lo = last[text[i]]
        i = i+m - min(j, 1 + lo)
        j = m-1
        return -1
```

Untuk implementasi pencarian string dengan *regex*, kami menggunakan *library re* yang telah disediakan oleh *Python*. Kami memanfaatkan *method search* untuk mencari kata-kata spam. *Method* ini akan mengembalikan nilai *true* jika terdapat kata-kata spam pada sebuah *tweet*, dan *false* jika tidak ditemukan.

```
if (re.search( spam, tweet.text, re.I)) :
    sp = True
```

d. Spam_detector.php

File ini merupakan sebuah file yang akan menjadi tempat untuk membuat tampilan pada web. Selain itu pada file ini kami menginisiasikan form untuk *user* mengisi username dan spam word yang ingin di cari dengan hanya mengklik sebuah tombol. Alur dari file ini adalah, user mengisi form dan mengklik tombol yang telah di sediakan, lalu program akan membuat file input.txt dari form dan pilihan user. Lalu, akan mengeksekusi python yang di dalamnya akan membuat file data.txt, setelah itu me-*refresh* halaman agar mendapatkan data spam yang terkini.

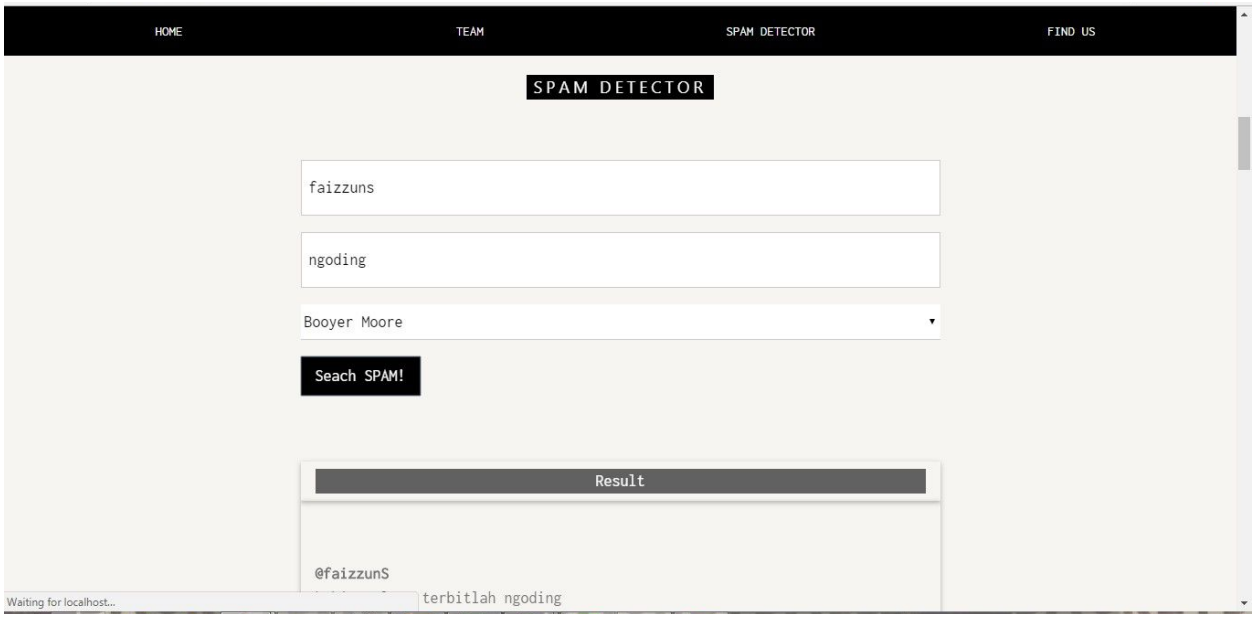
```php
<?php
        //exec("python twitter.py kitty", $output);

  $data = json_decode(file_get_contents("data.txt"), true);
  $text = $data['text'];
  //print_r($text);
  $i = 0;
  while($i<40):
      if ($text[$i]['spam']){
          echo "<p class="."\""."w3-red"."\""."">";
      }else{
          echo "<p class="."\""."w3-text-grey"."\""."">";
      }
      echo "<b>";
      echo "@";
      echo $text[$i]['username'];
      echo "</b>";

      echo "<br>";

      echo $text[$i]['test'];
      $i += 1;
      echo "</p>";
  endwhile;
?>
```

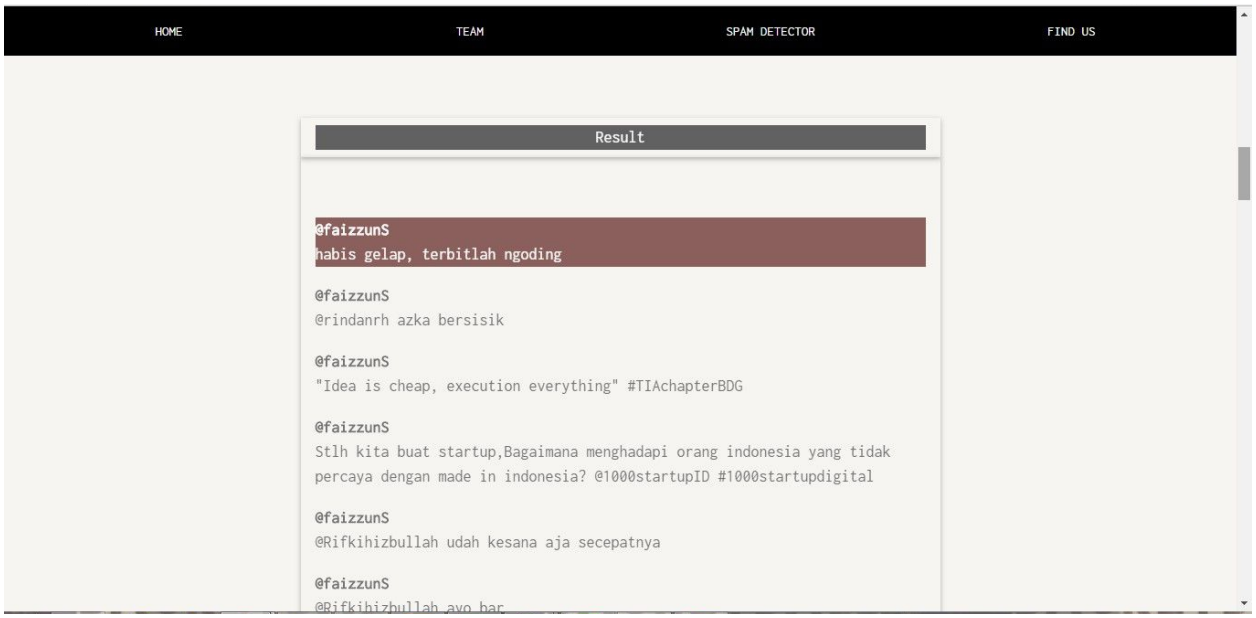
```html
<form method="post">
  <p><input class="w3-input w3-padding-16 w3-border" type="text" placeholder="Username" required name="searchKey"></p>
  <p><input class="w3-input w3-padding-16 w3-border" type="text" placeholder="Spam word" required name="spamKey"></p>
  <p>
    <select class="w3-select" name="option">
    <option value="" disabled selected>Choose your option</option>
    <option value="1">Booyer Moore</option>
    <option value="2">KMP</option>
    <option value="3">Regex</option>
    </select>
  </p>
  <p><button class="w3-button w3-black" type="submit">Seach SPAM!</button></p>
</form>
```

```php
<?php
    if (isset($_POST['option']) == "1")
    {
        $category = "bm\n";
        $searchKey = $_POST['searchKey']."\n";
        $spamKey = $_POST['spamKey']."\n";
        $myfile = fopen("input.txt", "w+") or die("Unable to open file!");
        fwrite($myfile, $category);
        fwrite($myfile, $searchKey);
        fwrite($myfile, $spamKey);
        fclose($myfile);
        exec("python twitter.py", $output);
        header("Refresh:0");
    }else if (isset($_POST['option']) == "2"){
        $category = "kmp\n";
        $searchKey = $_POST['searchKey']."\n";
        $spamKey = $_POST['spamKey']."\n";
        $myfile = fopen("input.txt", "w+") or die("Unable to open file!");
        fwrite($myfile, $category);
        fwrite($myfile, $searchKey);
        fwrite($myfile, $spamKey);
        fclose($myfile);
        exec("python twitter.py", $output);
        header("Refresh:0");
    }else if (isset($_POST['option']) == "3"){
        $category = "regex\n";
        $searchKey = $_POST['searchKey']."\n";
        $spamKey = $_POST['spamKey']."\n";
        $myfile = fopen("input.txt", "w+") or die("Unable to open file!");
```

4.2 Contoh Eksekusi

4.2.1 Kasus Normal Boyer Moore





4.2.2 Kasus Pengujian Case Insensitive Boyer-Moore

4.3 Analisis Hasil Pengujian

Pada pengujian diatas (poin b). Kami mencoba memasukan username twitter “faizzuns” dan membuat “ngoding” adalah kata kata spam dan memilih boyer moore untuk proses algoritmanya. Setelah mengeksekusi prgram, didapatkan bahwa tweet yang terindikasi SPAM akan di warnai warna merah pada seluruh boxnya.

Dapat disimpulkan bahwa percobaan kali ini berhasil.

BAB V
KESIMPULAN

Pencarian string (*pattern*) merupakan salah satu strategi yang banyak dipakai di kehidupan sehari-hari. Hal itu membuat pemilihan algoritma pencarian string (*pattern*) sangatlah penting, karena untuk algoritma yang dipilih akan mempengaruhi kecepatan program pencarian string tersebut. Pada tugas besar ini, kami melakukan pencarian string (*pattern*) dengan menggunakan algoritma Knutt Morris-Pratt (KMP), Boyce-Moore, dan *Regex.*

REFERENSI

https://id.wikipedia.org/

masputih.com

Munir Rinaldi, Diktat Kuliah Strategi Algoritmik, 193, 2005.