**TUGAS BESAR 3**

**Aplikasi *String Matching* untuk Disposisi *Tweets* ke Dinas-Dinas dan Instansi di Bawah Pemerintah Kota Bandung**

**LAPORAN**

Diajukan untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma

oleh :

**Wiega Sonora 13514019**

**Scarletta Julia Yapfrine 13514074**

**I Dewa Putu Deny Krisna Amrita 13514096**

****

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

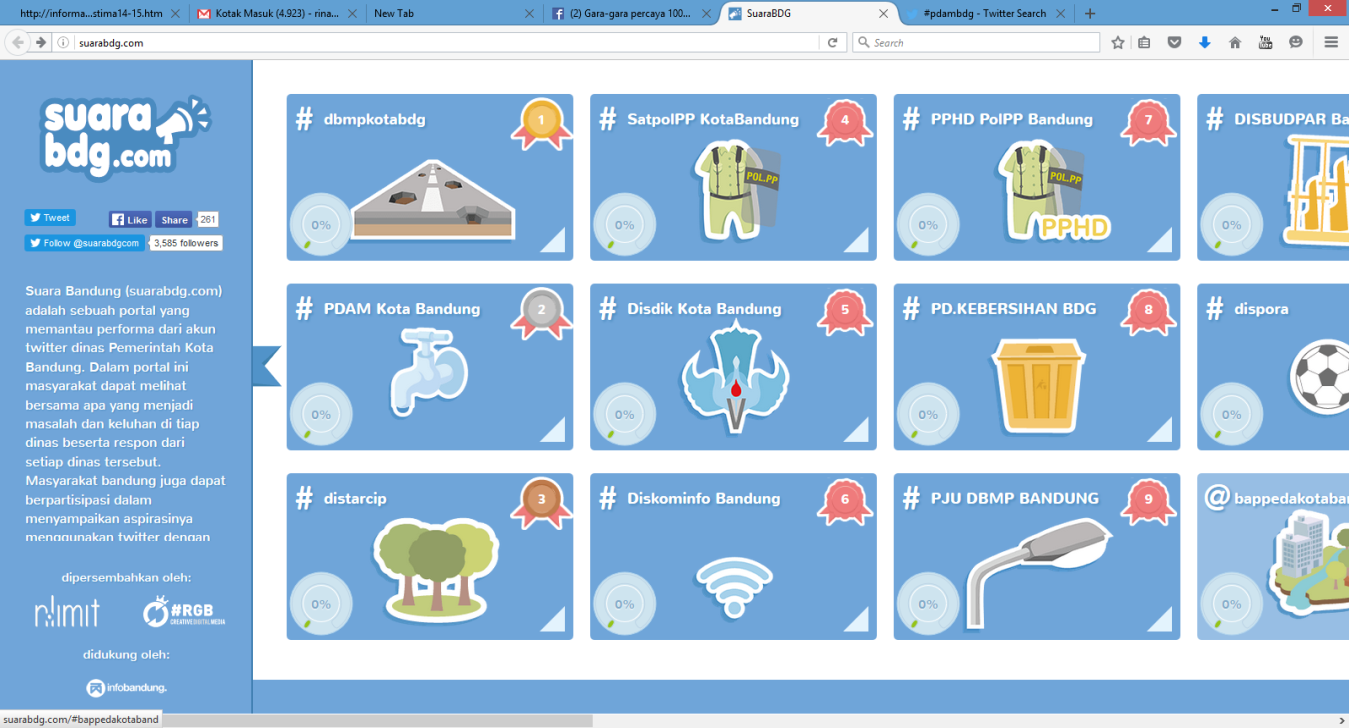
**INSTITUT TEKNOLOGI BANDUNG**

**2016**

**BAB I**

**DESKRIPSI MASALAH**

*Twitter* (<https://twitter.com/> ) merupakan media sosial yang banyak digunakan sebagai sumber informasi instans yang berasal dari masyarakat. Analisis *twitter* sudah sering menjadi topik riset di Teknik Informatika ITB. Berbagai aplikasi analisis twitter yang telah dihasilkan sebagai Tugas Akhir/Tesis, seperti *Blackberry AntiMacTweet* (Hasby angkatan 2009), peringkasan untuk menjelaskan *trending topic* Indonesia (Yosef angkatan 2009), analisis sentimen review film (Novan angkatan 2009), disposisi keluhan (Fauzan angkatan 2010), prediksi Big Five personality (Agnes angkatan 2010), *buzzer matcher* (Afif angkatan 2010; Elisafina & Irfan S2 angkatan 2012), dst. Tidak hanya sebatas riset, beberapa alumni mengembangkan startup analisis tweets seperti *noLimit* (Aqsath angkatan 2005) dengan produk <http://suarabdg.com/> .



Gambar 1. Contoh aplikasi analisis tweets: suarabdg.com

Pemerintah Kota Bandung menggunakan *Twitter* untuk menyerap keluhan dari masyarakat yang berkaitan dengan masalah yang menjadi wewenang dan tanggung jawab dinas-dinas dan instansi di dalam Pemkot (Dinas Kebersihan, Dinas Pendidikan, PDAM, Dinas Sosial, Satpol PP, Damkar, dll). Untuk selengkapnya silakan cari informasi tentang dinas/lembaga di bawah Pemkot Bandung, atau lihat contoh di <http://suarabdg.com>.

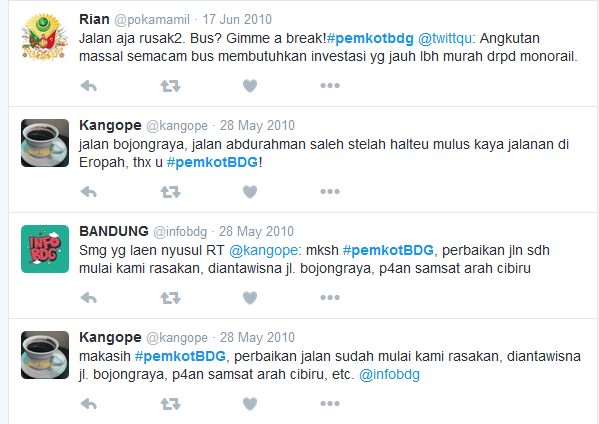
Pada Tugas Besar III kali ini Anda diminta membuat aplikasi sederhana analisis *tweet* berbasis kata kunci. *Tweet* yang berhasil diunduh dari *Twitter* dianalisis untuk selanjutnya didisposisikan ke dinas atau instansi yang terkait. Salah satu fungsi dasar yang digunakan dalam sistem analisis teks tersebut adalah pencocokan string. Algoritma pencocokan *string* (*pattern*) yang digunakan adalah *Knuth-Morris-Pratt* (KMP), dan *Boyer-Moore* (BM).

Terdapat banyak dinas maupun instansi di bawah Pemkot Bandung. **Anda pilih lima dinas/lembaga** saja. Pengguna aplikasi analisis *tweet* sederhana ini akan memberikan dua jenis masukan yaitu: (1) *keyword* pencarian *tweet* yang berasal dari tagar (*hashhtag*) atau *mention* #pemkotbdg, @pemkotbdg, #pemkotbandung, @ridwan kamil, @infobdg, dan yang sejenis dengan itu, dan (2) *keyword*. Tabel berikut menunjukkan contoh masukan dari pengguna.

|  |  |  |
| --- | --- | --- |
| *Keyword* Pencarian *Tweet* | *Keyword* | Output ke Dinas/Lembaga |
| #pemkotbdg  @pemkotbdg  @ridwan kamil  @infobdg | Pipa bocor; air ga ngalir; pdam mati; ga ngocor  ... | PDAM Bandung |
| PJU; Penerangan mati; lampu padam; terang; jalan gelap;  … | Dinas Bina Marga |
| sampah; bau busuk;  … | Dinas kebersihan |



Gambar 2. Contoh tweets @pemkotbdg



Gambar 3. Contoh tweets #pemkotbdg

Berdasarkan *keyword* pencarian *tweet* yang dimasukkan pengguna, aplikasi akan menggunakan *Twitter API* untuk mendapatkan minimal 100 *tweets* terbaru (*recent tweets*). Lalu, aplikasi menentukan kategori dinas/instansi dari setiap *tweet* dengan mengecek apakah *tweet* tersebut mengandung *keyword* yang diberikan untuk kategori dinas/lembaga tersebut.

* Jika *tweet* mengandung lebih dari satu *keyword* dari kategori dinas yang berbeda, *tweet* tersebut masuk dalam kategori dengan *keyword* yang muncul lebih dulu. Sebagai contoh, jika terdapat keyword “sampah” (kategori “dinas kebersihan”) dan “taman kotor” (kategori “dinas pertamanan”), *tweet* masuk kategori dinas kebersihan jika “sampah” muncul lebih dulu daripada kata “taman”.
* Jika tidak mengandung *keyword* apapun, *tweet* tersebut masuk dalam kategori dinas “unknown”.

Pencocokan *string* yang anda buat adalah *exact matching*, jadi *tweet* yang diproses mengandung string yang tepat sama dengan *keyword* yang telah ditentukan oleh pengguna. Di sini Algoritma KMP dan Boyer-Moore dapat digunakan. Untuk *keyword* berupa frase, maka perlu dilakukan penanganan secara khusus untuk menemukan tweet yang mengandung kata-kata di dalam frasa. 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).

Jika *tweet* mengandung nama tempat (dimulai dengan kata depan “di”), maka sebagai **bonus**, Anda dapat menampilkan lokasi dari setiap kategori dengan *GoogleMaps API*.

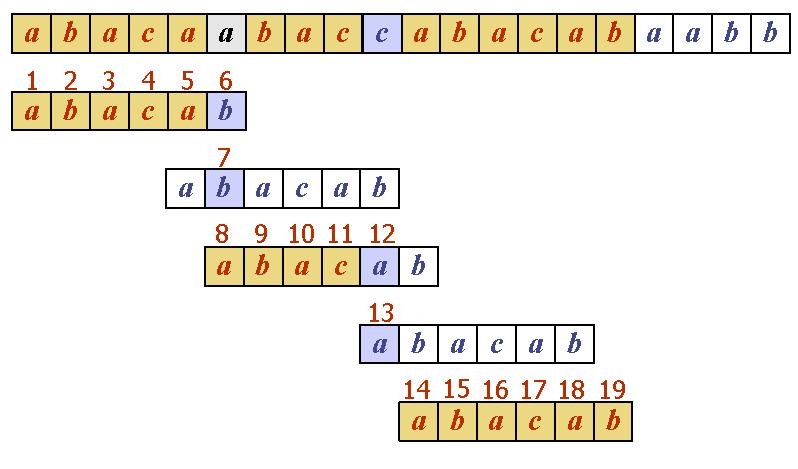
|  |  |
| --- | --- |
| Tweet | Lokasi Hasil Ekstraksi |
| [**#pemkotbdg**](https://twitter.com/search?q=%23lalinBDG&src=hash) 09.16 : lampu di jln tamansari jika malam padam | tamansari |
| [@**infobdg**](https://twitter.com/infobdg): 08.23 : Macet (lagi) di jln rancaekek - cicalengka maju dikit2 ga terlalu [pic.twitter.com/gyK7PaCTVM](http://t.co/gyK7PaCTVM) | [@**Latifaa\_Aulia**](https://twitter.com/Latifaa_Aulia): | rancaekek - cicalengka |
| [#**suaraBDG**](https://twitter.com/search?q=%23suaraBDG&src=hash) via [@**dionmudjenan**](https://twitter.com/dionmudjenan): Hatihati di jembatan perbatasan komp GBI td ada yg kena begal pelaku mabok bawa golok nyandra yg bawa motor | jembatan perbatasan komp GBI |
| [@**ridwan kamil**](https://twitter.com/quinsymegamira): di terminal leuwipanjang banyak anak jalanan yg ngemis2 sementara ibunya diem di pos polisi sambil bbman X\_X [#**Suarabdg**](https://twitter.com/search?q=%23Suarabdg&src=hash) [@**dinsos\_bdg**](https://twitter.com/Dinsos_BDG) | terminal leuwipanjang |

**BAB II**

**DASAR TEORI**

1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP dikembangkan oleh D.E. Knuth, J. H. Morris, dan V. R. Pratt. Pada algoritma KMP, informasi dipelihara untuk melakukan jumlah pergeseran. Algoritma KMP memelihara informasi tersebut untuk membuat pergeseran yang banyak, tidak hanya per satu karakter seperti pada algoritma *brute force*. Waktu pencarian karakter dengan menggunakan algoritma KMP berkurang secara signifikan jika dibandingkan dengan *brute force*. Algoritma KMP memroses teks dari kiri ke kanan.



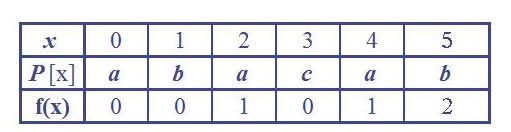
**Gambar 4. Pemrosesan dengan KMP**

Sumber : <http://3.bp.blogspot.com/-VYM2tFCZAYY/VmmBlGy5WSI/AAAAAAAAAbs/RxuXJUcG1pw/s1600/kmpexample.jpg>

Pada gambar 4, T[6] = a ≠ P[6] = b. Karena T[1..5] sama dengan P[1..5] dan T[6] ≠ P[6] maka pola digeser dengan pergeseran yang mungkin berhasil, yaitu sebanyak 4. Perbandingan T[5] dan P[1] tidak diperlukan karena sudah dilakukan pada pencocokan sebelumnya) sehingga perbandingan dimulai dari T[6] dan P[2].

**Fungsi Pinggiran (*Border Function)***

Fungsi pinggiran digunakan untuk melakukan proses awal terhadap *pattern* P. Fungsi pinggiran menandakan pergeseran *s* terbesar yang mungkin dengna menggunakan perbandingan yang dibentuk sebelum pencarian *string*. Fungsi pinggiran ini mencegah pergeseran yang tidak berguna, seperti pada *brute force*. Perhitungan fungsi pinggiran dilakukan sebelum pencarian *string* dilakukan karena hanya bergantung pada karakter-karakter di dalam pola, dan bukan pada karakter-karakter di dalam teks. Fungsi pinggiran pada gambar 4 adalah

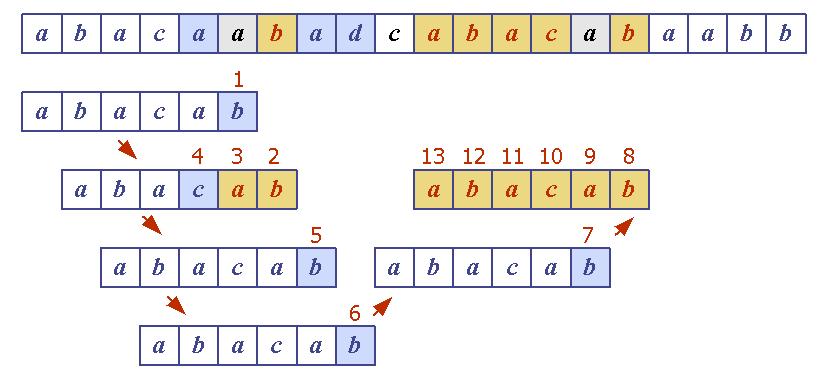


**Gambar 5. Fungsi Pinggiran**

Sumber : <http://3.bp.blogspot.com/-VYM2tFCZAYY/VmmBlGy5WSI/AAAAAAAAAbs/RxuXJUcG1pw/s1600/kmpexample.jpg>

1. Algoritma Boyer-Moore

Algoritma Boyer-Moore mencocokkan pola dengan teks dimulai dari akhir pola (dari kanan ke kiri). Jika terjadi ketidakcocokkan karakter, maka ada tiga kemungkinan pergeseran yang dilakukan. Misalkan ketidakcocokkan terjadi pada T[i] = x, dengan T adalah teks. Karakter pada pola P[j] tidak sama dengan T[i]. Kasus yang pertama adalah jika P mengandung x pada indeks yang lebih kecil daripada j, maka P digeser ke kanan untuk menyesuaikan indeks x tersebut dengan T[i]. Kasus yang kedua adalah jika P mengandung x pada ideks yang lebih besar daripada j, maka P digeser ke kanan sebanyak 1 karakter. Kasus ketiga adalah jika P tidak mengandung x, maka P digeser ke kanan untuk menyesuaikan P[1] dengan T[i+1].



**Gambar 5. Pemrosesan dengan algoritma Boyer-Moore**

Sumber : <https://koding4fun.files.wordpress.com/2010/05/complete_example.jpg>

**Fungsi Kemunculan Terakhir**

Algoritma Boyer-Moore melakukan pemrosesan awal terhadap pola P untuk membuat fungsi kemunculan terakhir, L(x). L(x) menghasilkan indeks kemunculan terakhir karakter pada pola. Jika suatu karakter x tidak ada pada pola P, maka L(x) = -1. Tabel fungsi kemunculan terakhir untuk karakter a, b, c, d pada gambar 5 adalah

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | a | b | c | d |
| L(x) | 5 | 6 | 4 | -1 |

**BAB III**

**ANALISIS PEMECAHAN MASALAH**

Untuk melakukan analisis *tweet* warga Bandung, kami menggunakan *Twitter* *API* untuk. *Twitter API* digunakan untuk mendapatkan *tweet-tweet* terbaru dari <https://twitter.com/>. *Tweet-tweet* yang didapat kemudian dikategorikan ke dinas atau instansi yang ada di Bandung dengan memeriksa apakah *tweet* tersebut mengandung kata kunci yang diberikan untuk masing-masing kategori dinas atau instansi tersebut. Banyaknya *tweet* yang kami ambil dengan *Twitter API* adalah 200 *tweet*.

Pengecekan apakah *tweet* tersebut mengandung kata kunci dilakukan dengan pencocokan *string* (*string matching*) dalam bentuk *exact matching* (*tweet* mengandung *string* yang tepat sama dengan kata kunci yang telah ditentukan). Pencocokan *string* dilakukan dengan algoritma Knuth-Morris-Pratt dan algoritma Boyer-Moore. Pencarian bersifat *case insensitive* sehingga dalam pemrosesannya, semua karakter yang berbentuk huruf dijadikan *lower case letter*. Pengguna bisa memilih pencarian apakah ingin menggunakan Boyer-Moore atau Knuth-Morris-Pratt. Masing-masing algoritma pencarian dibuat dalam bahasa pemrograman C#.

Jika *tweet* mengandung nama tempat, maka tempat tersebut akan ditampilkan lokasinya pada peta dengan menggunakan *GoogleMaps API*. Berikut adalah ilustrasi sederhana

**My Tweet Analysis**

#pemkotbdg

Pencarian *twitter*:

Pipa bocor; ngalir; air mati; , bioskop, cinema, blitzmegaplex,

*Keyword* PDAM:

Tiang roboh, padam pampakbola, gol

*Keyword* Dinas PJU:

pengemis, orang gila

*Keyword* Dinas Sosial:

Analisis

Algoritma :

* Boyer-Moore
* KMP

Perihal

**Kategori PDAM:**

Jumlah: 2

Daftar tweets:

TL simp Malabar-Gatsu sering padam & banyak lampunya yg mati, mhn diperhatikan **@PemkotBdg** [~~@~~**tmc\_restabesbdg**](https://twitter.com/tmc_restabesbdg) cc [~~@~~**ridwankamil**](https://twitter.com/ridwankamil) [~~@~~**PRFMnews**](https://twitter.com/PRFMnews)

PJU Jl.Veteran mulai simp Sunda hingga A.Yani padam [~~@~~**dbmpkotabdg**](https://twitter.com/dbmpkotabdg) [~~@~~**PRFMnews**](https://twitter.com/PRFMnews) **@PemkotBdg**

**Kategori Dinas Kebersihan:**

Jumlah: 1

Daftar tweets:

[~~#~~**pemkotBDG**](https://twitter.com/search?q=%23lalinBDG&src=hash) 13.06 : Sampah di pertigaan Veteran-Sunda menumpuk, luber ke jalan. [pic.twitter.com/Sm0Ua0Vn54](http://t.co/Sm0Ua0Vn54) | [@**BemiCuteMan**](https://twitter.com/BemiCuteMan)

**Kategori UNKNOWN:**

Jumlah: 1;

Daftar tweets:

...

**BAB IV**

**IMPLEMENTASI DAN PENGUJIAN**

1. Spesifikasi Teknis Program

Program dibuat menggunakan empat bahasa serta digunakan satu kakas. Bahasa tersebut antara lain, HTML, javascript, CSS, dan C#. Bahasa pemrograman HTML, Javascript, dan CSS digunakan untuk mengatur bagian front-end atau tampilan antar-muka. Ketiga bahasa pemrograman tersebut diaplikasikan dengan ASP.NET yang disediakan oleh Microsoft. Bahasa pemrograman C# digunakan untuk mengatur bagian back-end atau pemrosesan data.

Pada tampilan antar muka, kami menggunakan template Bootstrap yang tersedia gratis di startbootstrap.com. Halaman web tersebut terbagi menjadi tiga bagian utama, yaitu header, bod, dan footer. Header berisi halaman awal pengenalan. Body berisi konten utama yang terdiri dari form input (area pencarian berupa username atau hashtag, keyword per dinas, dan pilihan algoritma baik Boyer-Moore atau KMP) dan hasil pencarian yang dipetakan pada panel-panel sesuai Dinas masing-masing. Lalu pada bagian footer terdapat informasi mengenai pembuat project tersebut.

Pada bagian back-end, kami menggunakan C#.

Sruktur Teknis Program

Pada file Default.aspx.cs yang mana merupakan program berbasis C# untuk mengatur bagian back-end terdiri dari beberapa kelas dan method. Kelas-kelas tersebut antara lain: kelas Default dan kelas TwitterExtensions. Kelas Default berisi method-method berupa void dan beberapa fungsi. Beberapa method tersebut antara lain :

Struktur Data:

* Daftar tweet yang diretrieve dari Twitter API disimpan dalam List of Status.
* Dalam mengategorikan tweet sesuai dinas yang terkait disimpan dalam array sesuai sesuai dinas yang berhubungan

Algoritma KMP:

* Fungsi public static int[] computeFail(String pattern)

Fungsi ini berguna untuk menghitung jumlah kegagalan dalam pencocokan string sehingga dapat dijadikan parameter untuk menggeser pattern.

* Fungsi public static int kmpMatch(String text, String pattern)

Fungsi ini berguna untuk mencocokan pattern ke text dengan algoritma Knuth-Morris-Pratt dan mengembalikan indeks ditemukannya string yang sesuai, jika tidak diitemukan fungsi akan mengembalikan nilai -1.

Algoritma Boyer-Moore:

* Fungsi public static int[] buildLast(String pattern)

Fungsi ini mengembalikan array integer berupa indeks terakhir dimana ditemukannya kesamaan antara text dan pattern sehingga dapat digunakan untuk pencocokan selanjutnya, akan mengembalikan nilai -1 jika tidak ditemukan kesamaan

* Fungsi public static int bmMatch(String text, String pattern)

Fungsi ini berguna untuk mencocokan pattern ke text dengan algoritma Boyer-Moore dan mengembalikan indeks ditemukannya string yang sesuai, jika tidak diitemukan fungsi akan mengembalikan nilai -1.

Twitter API:

* Method private SingleUserAuthorizer authorizer = new SingleUserAuthorizer

Method ini berfungsi untuk autorisasi aplikasi ke API Twitter

* Fungsi private List<Status> SearchTwitter(string searchTerm)

Fungsi ini berguna untuk menghasilkan List dengan elemen status (tweet) yang dicari di daftar tweet dengan kata kunci sesuai dengan searchTerm

* Prosedur public void seekTwit(object sender, EventArgs e)

Prosedur ini berfungsi untuk memisahkan tweet-tweet ke dinas-dinas terkait sesuai dengan kata kunci yang dimasukkan pengguna pada form yang terdapat pada file Default.aspx dan sesuai pilihan algoritma yang dipilih. Selain itu, prosedur ini menghitung jumlah tweet dari masing-masing kategori (Dinas)

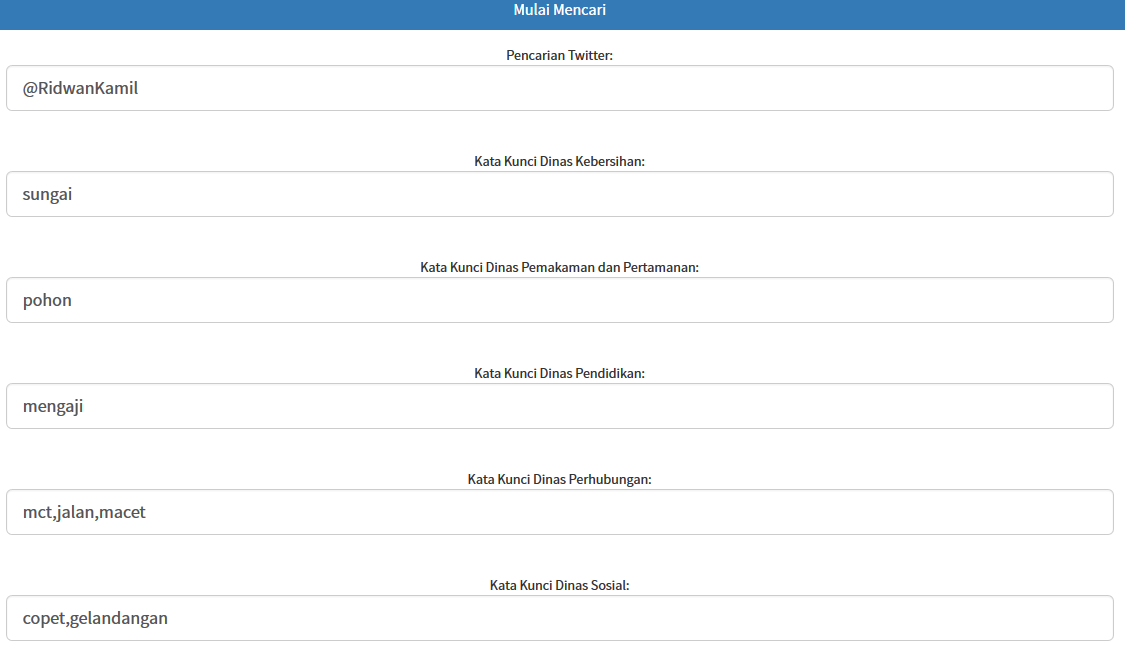
Twitter Extensions:

* Fungsi public static string TextAsHtml(this String temp)

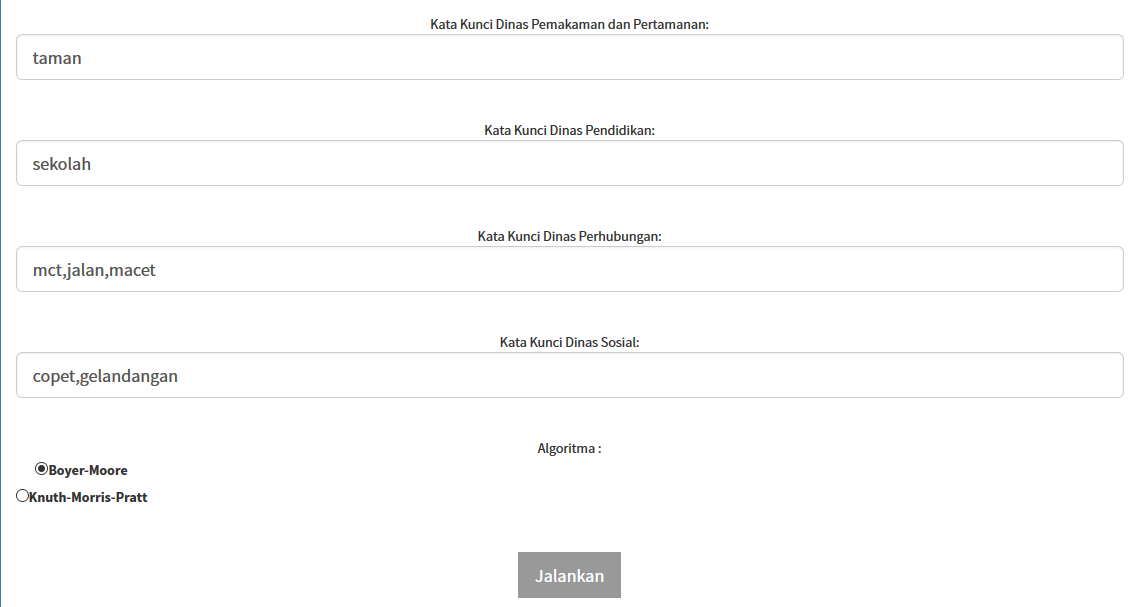
Fungsi ini berguna untuk mengubah string tweet yang biasa menjadi string yang mempunyai link sesuai dengan jenisnya (mention, hashtag, link, dan lain-lain).

1. Eksperimen / Pengujian

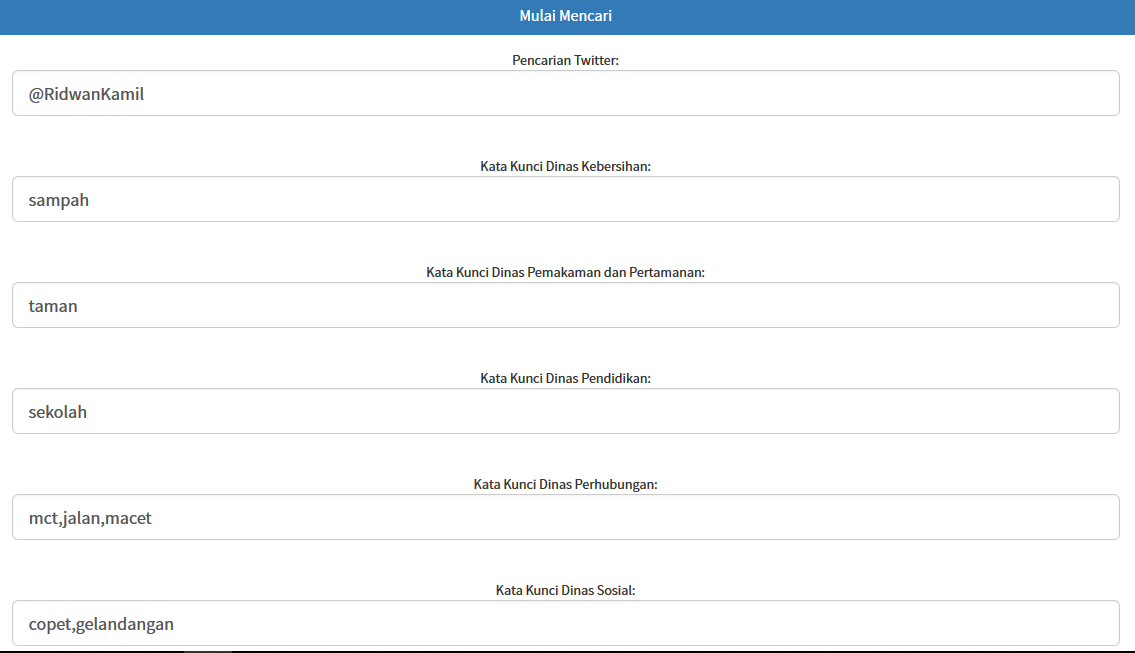
Pengujian aplikasi analisa *tweet* dapat dilihat pada *screenshot* di bawah :



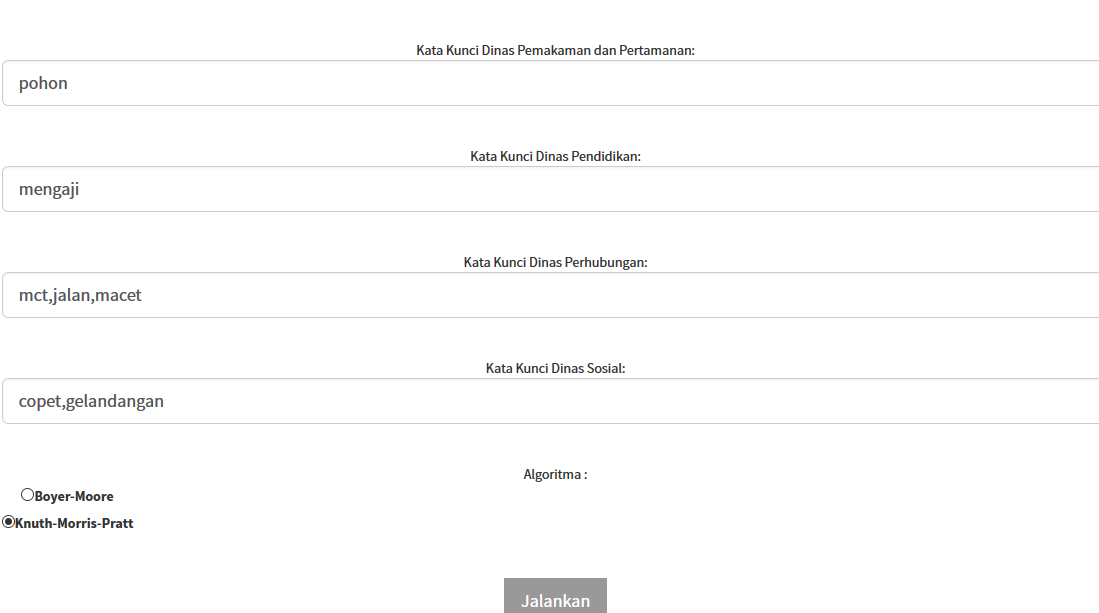
**Gambar 6. Hasil *Screenshot* I**



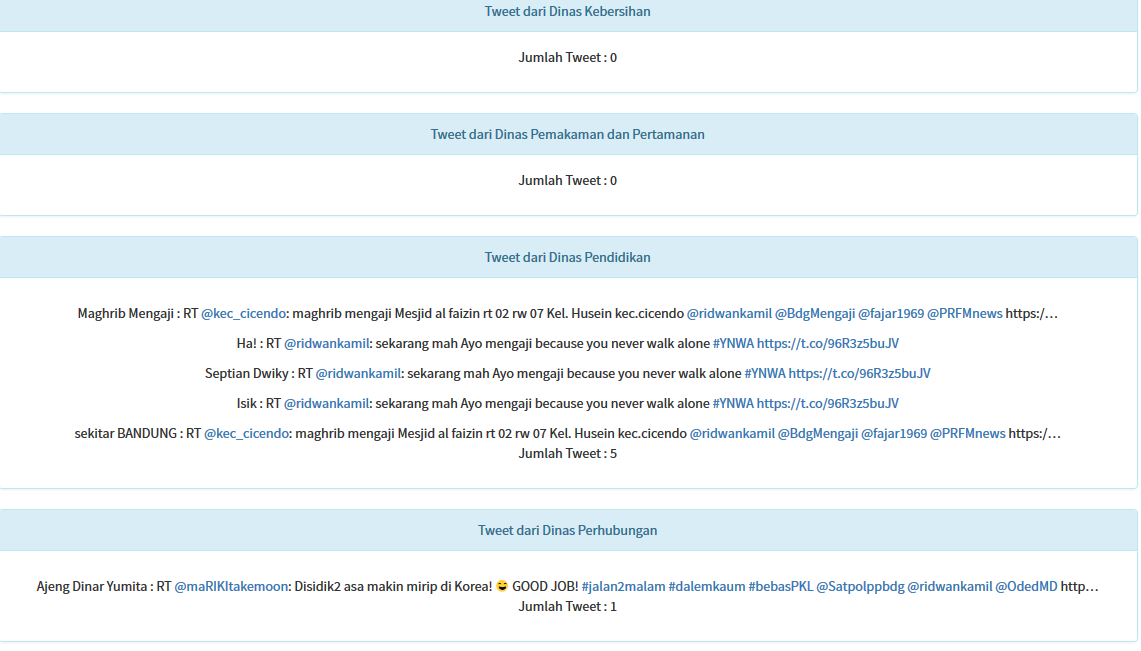
**Gambar 7. Hasil *Screenshot* 2**



**Gambar 8. Hasil *Screenshot* 3**



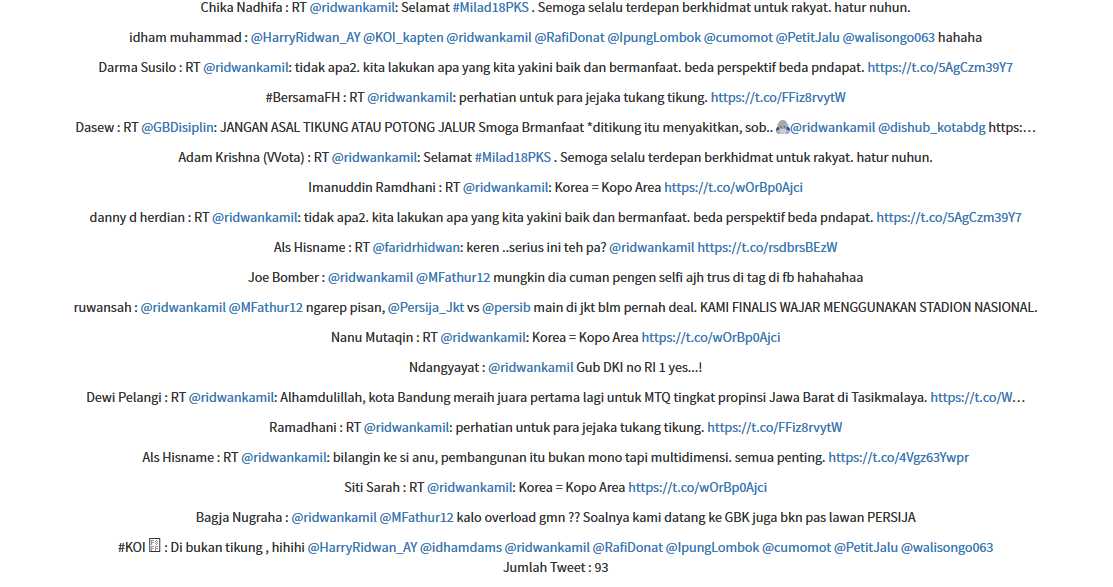
**Gambar 9. Hasil *Screenshot* 4**



**Gambar 10. Hasil *Screenshot* 5**



**Gambar 11. Hasil *Screenshot* 6**



**Gambar 12. Hasil *Screenshot* 7**

Penjelasan :

1. Pengguna dapat memberikan masukan berupa pengguna lain atau tagar yang diinginkan, kata kunci untuk 5 dinas yang disediakan, yaitu dinas kebersihan, dinas pertamanan dan pemakaman, dinas pendidikan, dinas perhubungan, dan dinas sosial. Dalam eksperimen kali ini, masukan pengguna dapat dilihat pada *screenshot* 1,2,3, dan 4.
2. Setelah pengguna mengisi *text box* dengan masukan yang diinginkan, pengguna dapat menggunakan dua option algoritma pencarian, yaitu *Knuth-Morris-Pratt* dan *Boyer-Moore*. Setelah memilih algoritma, pengguna dapat melakukan analisis dengan menekan tombol ‘Jalankan!’
3. Setelah tombol ditekan, program akan melakukan analisis sesuai dengan algoritma yang dipilih. Pada eksperimen ini, pengguna mendapatkan 0 *tweet* dari dinas kebersihan, 0 *tweet* dari dinas pertamanan, 5 *tweet* dari dinas pendidikan, 1 *tweet* dari dinas perhubungan, 1 *tweet* dari dinas sosial, dan 93 *tweet* yang tidak masuk ke-5 kategori dinas tersebut.
4. Analisis Hasil Pengujian

Dari pengujian yang dilakukan dengan *testcase* masukan seperti pada bagian 4.2., didapatkan hasil sebagai berikut :

|  |  |
| --- | --- |
| Dinas Tujuan | *Tweet* yang dihasilkan |
| Dinas Kebersihan | 0 |
| Dinas Pertamanan | 0 |
| Dinas Pendidikan | 5 |
| Dinas Perhubungan | 1 |
| Dinas Sosial | 1 |
| Tidak terkategorisasi | 93 |

Dari hasil pengujian di atas, dapat diberikan analisis sebagai berikut :

* Dari prediksi 200 *tweet* yang dapat diambil, hanya 100 *tweet* yang dihasilkan oleh program. Dari 100 *tweet*  yang didapat, 93 diantaranya tidak masuk dalam kategori dinas manapun. Hasil ini menunjukkan bahwa masukan pengguna akan menentukan hasil kategorisasi sesuai dengan masukan yang diberikan. Selain itu, hasil yang didapat juga ditentukan oleh faktor lain, seperti kuantitas *tweet* yang diberikan pengguna lain.
* Performansi kedua algoritma yang digunakan, yaitu *Knuth-Morris-Pratt* dan *Boyer-Moore* tergolong sama, terlihat dari kedua masukan pencarian yang sama dengan algoritma yang berbeda memberikan hasil yang sama. Perbedaan yang terlihat hanya di algoritmanya saja, dimana algoritma KNP menitikberatkan pada penghitungan *border function* atau fungsi perbedaan string, sedangkan algoritma *Boyer-Moore* bergantung pada string yang dicocokkan sesuai dengan kasus yang dihadapi.

**BAB V**

**KESIMPULAN DAN SARAN**

1. Kesimpulan

Pencocokan *string* dapat dilakukan untuk disposisi *tweet* untuk dinas atau instansi di Bandung, sehingga dapat dijadikan bahan analisis. Algoritma *string matching* yang digunakan adalah algoritma Knuth-Morris-Pratt dan algoritma Boyer-Moore. Algoritma Knuth-Morris-Pratt melakukan pencocokan pola dimulai dari depan pola, sementara algoritma Boyer-Moore melakukan pencocokan pola dimulai dari belakang pola. Pencocokan *string* dilakukan berdasarkan *keyword* yang dimasukkan pengguna pada masing-masing kategori. Pencocokan dilakukan terhadap *tweet* yang telah di-*filter*.

1. Saran

Untuk melakukan analisis dari media sosial, lebih baik tidak menggunakan *Twitter*. Hal ini disebabkan karena media sosial *Twitter* mulai ditinggalkan, sehingga pengguna *Twitter* tidak terlalu banyak. Karena hal itu, *tweet* yang bisa diambil tidak banyak, sehingga tidak terlalu menggambarkan pendapat seluruh masyarakat kota Bandung.

**DAFTAR PUSTAKA**

Munir, Rinaldi. “Diktat Kuliah IF2211 Strategi Algoritma”. 2009. Bandung : Institut Teknologi Bandung

Davidson, Andrew. 240-301, Computer Engineering Lab III (Software), Semester 1, 2006-2007

http://www.w3schools.com/googleapi/