Deteksi Spam Pada Twitter Menggunakan Algoritma Pencocokan String

Priagung Satyagama 13516089[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516089@std.stei.itb.ac.id

***Abstrak*—Spam bisa disebut sebagai pengganggu atau parasit ketika sedang berselancar atau menggunakan sosial media. Spam tersebut bisa datang dari bot, akun palsu, atau yang lainnya. Spam bisa dideteksi dengan menggunakan algoritma pencocokan string seperti algoritma KMP, Booyer-Moore ataupun dengan menggunakan Regular Expression. Twitter merupakan salah satu sosial media yang juga tidak lepas dari yang namanya spam. Oleh karena itu pada kesempatan ini saya akan mencoba membuat implementasi algoritma pencocokan string tersebut untuk mendeteksi spam yang ada pada twitter.**

***Kata Kunci*— Spam, Algoritma, Twitter, Social Media**

I. PENDAHULUAN

Dewasa ini, zaman sudah berganti menjadi zaman digital moderen. Di zaman moderen seperti ini, manusia sangat bergantung pada teknologi digital. Teknologi digital sangat mempengaruhi kehidupan manusia di segala aspek. Salah satu nya adalah pada spek sosial dimana media digital yang memfasilitasi itu disebut dengan media sosial. Media sosial menjadi media bagi manusia terutama manusia generasi milenial untuk mencari jati diri, mengumbar eksistensi diri, juga sebagai media belajar atau sekedar media komunikasi saja.

Ketergantungan manusia pada media sosial selain dampak negatif yang menyebabkan manusia itu kurang bersosialisasi dengan lingkungannya secara langsung tapi juga membawa dampak positif seperti mudahnya berkomunikasi jarak jauh dan mudahnya mengakses konten-konten pembelajaran. Potensi yang dimiliki media sosial itu tidak hanya dimanfaatkan oleh orang-orang yang bertanggung jawab saja. Terkadang, potensi yang sangat besar tersebut digunakan oleh orang-orang yang tidak bertanggung jawab untuk melakukan perbuatan-perbuatan tidak etis yang merugikan orang lain hanya untuk kesenangan atau kemaslahatan pribadi.

Spam adalah salah satu bentuk dari penggunaan media sosial yang tidak bertanggung jawab tersebut. Latar belakang orang melakukan spam tentu berbeda-beda. Ada yang mengejar popularitas, berdagang, bahkan sampai penipuan. Spam ini beredar diberbagai macam sosial media yang ada saat ini, baik itu yang cenderung formal seperti email, atau sosial media yang informal seperti twitter.

Pada makalah ini, penulis akan mencoba membuat aplikasi berbasis web untuk mendeteksi spam yang beredar pada twitter dengan menggunakan keyword spam tertentu. data tweet penulis dapatkan dari tweet-tweet yang melakukan mention terhadap akun twitter yang sedang di analisis. Penulis akan menggunakan algoritma pencocokan string untuk mencocokan apakah suatu tweet mengandung spam atau tidak, tentu jika mengandung spam, pemilik akun twitter bisa menentukan keputusan nya apakah akan memblok akun dengan tweet spam tersebut ataukah tidak. Aplikasi ini merupakan aplikasi berbasis web yang dibuat dengan bahasa pemrograman PHP framework Laravel, sedangkan untuk program algoritma pencocokan string nya akan ditulis dengan bahasa Python.

Semoga aplikasi yang akan kami jelaskan pada makalah ini dapat berguna bagi pengguna twitter untuk mendeteksi spam dan mengurangi ketidaknyamanan dalam berselancar di sosial media yang disebabkan oleh pengguna-pengguna yang tidak bertanggung jawab yang melakukan *spamming*.

II. DASAR TEORI

A. Media Sosial

Media sosial adalah media berbasiskan internet yang biasanya digunakan untuk berkomunikasi atau berinteraksi dengan siapapun tanpa dibatasi ruang dan waktu. Bentuk dari sosial media sangat bermacam-macam antara lain blog, jaringan bisnis, forum, *microblogs*, *photo sharing*, *products/services review*, *social bookmarking*, *social gaming*, *social networks*, *video sharing*, dll. Pelopor dari sosial media antara lain platform-platform seperti sixdegrees, ICQ, iChat, dsb. Sayangnya, pelopor sosial media itu umur nya tidak panjang karena pada masa itu, sosial media belum terlalu diminati oleh banyak orang.

Sosial media muncul ketika *World Wide Web* bisa memfasilitasi interaksi sosial. Perkembangan fungsionalitas dari *Web 2.0* belakangan ini menyebabkan generasi millenial melakukan terobosan untuk menciptakan media yang dapat mereka gunakan untuk Web sebagai media untuk melakukan interaksi sosial. Ditunjang dengan *online data storage* yang sudah memungkinkan untuk menaruh data yang sangat besar membuat perkembangan media sosial terjadi dengan sangat cepat. Menanggapi permintaan tersebut, sosial media berkembang sangat cepat yang bisa dilihat sebagai potensi bisnis atau juga fenomena sosial. Sebagai contoh, Facebook yang diluncurkan pada tahun 2004, sekarang pengguna nya sudah mencapai 1,4 Milyar, begitu juga dengan twitter dan lain-lain. Berikut adalah tabel mengenai jumlah pengguna media sosial yang *leading* saat ini.

Tabel 1

Layanan sosial media terbesar di dunia

Layanan	Jumlah Akun (dalam juta)
Facebook	1415
QQ	829
WhatsApp	700
QZone	629
WeChat	468
LinkedIn	347
Skype	300
Google+	300
Instagram	300
Baidu Tieba	300
Twitter	288
Viber	236
Tumblr	230
Snapchat	200
LINE	181
Sina Weibo	167

Sumber : Statisa.com

B. Social Spam

Social Spam adalah konten spam yang muncul pada jaringan layanan media sosial dan website lain yang terdapat konten yang dapat di buat oleh user seperti komen, chat, dsb. Lebih dari 90% pengguna jejaring sosial pernah mendapatkan *Social Spam* dalam berbagai bentuk. Mereka yang melakukan spam bisa berupa bot, akun palsu, atau bahkan manusia sungguhan. *Social Spammer* biasanya menyebarkan konten-konten berupa berita atau link yang berbahaya yang mendominasi bagian komentar dari suatu website atau sosial mdia.

Social Spam saat ini juga sedang berkembang. Sekitar 40% dari seluruh akun sosial media yang ada saat ini adalah akun palsu. Pada bulan Agustus 2012, Facebook menyatakan melalui regulasi terbaru nya bahwa 8,7% dari pengguna nya saat itu yaitu 955 juta pengguna aktif adalah akun palsu. Terdapat beberapa karakteristik dari spam antara lain berjumlah banyak, tidak senonoh, cenderung menghina, mengancam, menyinggung SARA, penipuan, dsb.

C. Pencocokan String

Pencocokan String adalah ketika diberikan sebuah text long string T yang panjang sebut saja panjangnya n karakter dan sebuah *pattern* P yaitu string dengan panjang m karakter dimana m jauh lebih kecil dari n (m <<< n). Kemudian kita diminta untuk mencari (*find* atau *locate*) lokasi pertama kemunculan substring dari text T yang bersesuaian dengan *pattern* P. Berikut adalah contohnya.

T : "*semoga uas stima tidak susah seperti uts*"

P : "*uas*"

maka pattern "uas" ditemukan pada karakter ke 8 pada text T

Terdapat beberapa aplikasi pencocokan string pada aplikasi yang lumayan populer saat ini seperti editor text, search engine, analisis citra, dan *bioinformatics*.

Ada beberapa terminologi dasar dari sebuah string yang perlu diketahui yaitu suffix dan prefix. Prefix dari sebuah string S adalah sebuah substring S[0..k], sedangkan Suffix dari sebuah string S adalah sebuah substring S[k..m-1], dimana m adalah panjang string S dan k adalah index yang berada antara 0 dan m-1.

D. Algoritma Bruteforce Untuk Pencocokan String

Prinsip dari algoritma bruteforce untuk pencocokan string sangat sederhana. Pengecekan dilakukan secara satu persatu untuk setiap karakter pada text dan mencocokannya dengan setiap karakter pada pattern. Jika terjadi *mismatch*, maka posisi *pattern* digeser satu persatu sampai tidak terjadi *mismatch* yang menandakan *pattern* ditemukan atau ketika *pattern* sudah sampai posisi akhir dari teks tetapi masih terdapat *mismatch* yang berarti *pattern* tidak terdapat pada teks. sebagai contoh.

Pattern : UAS

Teks : SEMOGA UAS STIMA TIDAK SULIT

SEMOGA **UAS** STIMA TIDAK SULIT

1. UAS mismatch
2. UAS mismatch
3. UAS mismatch
4. UAS mismatch
5. UAS mismatch
6. UAS mismatch
7. UAS mismatch
8. **UAS** **match!**

Pada kasus terburuk, Algoritma bruteforce untuk pencocokan string memiliki kompleksitas O(mn). Contoh kasus terburuk adalah sebagai berikut.

T : "uuuuuuuuuuuuuuuuuuuuuuuuus"

P : "uuus"

Sedangkan untuk kasus terbaik, algoritma bruteforce untuk pencocokan string memiliki kompleksitas O(n). Kasus terbaik terjadi ketika karakter pertama *pattern* P tidak pernah sama dengan karakter teks T yang dicocokan atau sekalinya sama, maka karakter-karakter selanjutnya akan terus sama sampai menunjukan bahwa *pattern* match dengan karakter-karakter pada teks tersebut. Berikut adalah contoh dari kasus terbaik algoritma bruteforce untuk pencocokan string.

T : “semoga besok stima mudah uas nya”

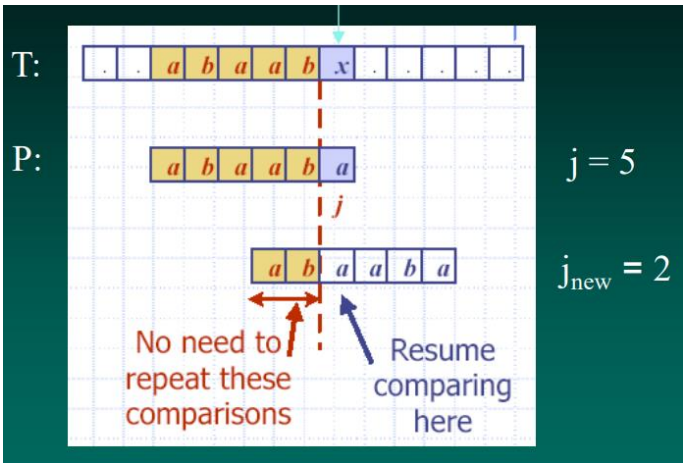
P : “uas”

Untuk kasus rata-rata, algoritma bruteforce untuk pencocokan string memiliki kompleksitas O(m+n). Algoritma ini cepat ketika alfabet yang digunakan dalam text itu variasi nya banyak (contoh : decimal), dan algoritma ini lambat ketika variasi alfabet yang digunakan itu sedikit (contoh : binary).

E. Algoritma KMP Untuk Pencocokan String

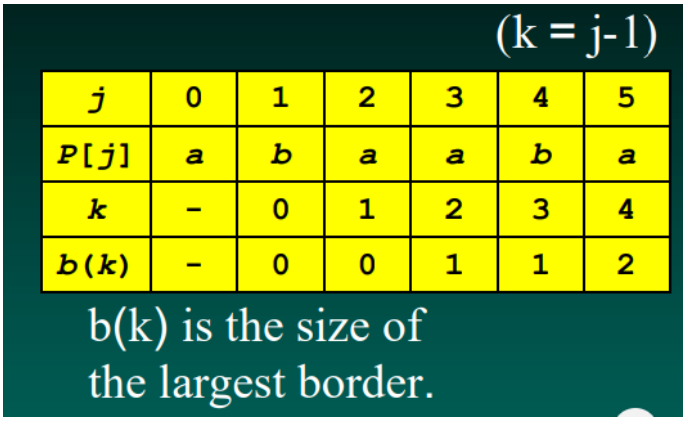
Algoritma KMP pertama kali oleh seorang profesor dari stanford bernama Donald Ervin Knuth. Knuth biasa disebut bapak dari anlisis algoritma. Pada prinsipnya, algoritma KMP mirip seperti bruteforce, yaitu melakukan pengecekan pola dari kiri ke kanan. Hanya saja, pergeseran pola nya lebih cerdas. Tidak hanya satu-satu ke kanan, tapi bergesernya bisa lebih dari itu menyesuaikan dengan hasil dari evaluasi *border function* atau fungsi pembatas.

Jika *mismatch* muncul diantara teks dan *pattern* pada P[j], maka sejauh-jauhnya kita dapat menggeser *pattern* untuk mencegah komparasi yang berlebihan adalah sejauh panjang *pattern* dikurang dengan panjang prefix dari P[0..j-1] yang juga suffix dari P[1..j-1].



Gambar 1 Ilustrasi Algoritma KMP[4]

Mengenai fungsi pembatas/fungsi pinggiran, algoritma KMP memproses *pattern* untuk menentukan kecocokan antara prefix dari *pattern* dengan *suffix* dari pattern itu sendiri. Keluaran dari fungsi ini adalah berupa tabel. misalkan j adalah posisi mismatch didalam *pattern*, k adalah posisi sebelum mismatch yang merupakan j-1. fungsi pinggiran b(k) didefinisikan sebagai panjang terbesar dari prefix P[0..k] yang juga merupakan suffix dari P[1..k]. Berikut adalah contoh pemrosesan *pattern* dengan fungsi pinggiran.

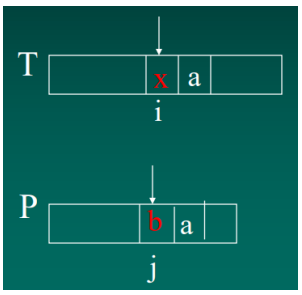


Gambar 2 Contoh Evaluasi Fungsi Pinggiran

Untuk menghitung fungsi pinggiran, kompleksitas waktunya adalah O(m), sedangkan untuk pencarian string kompleksitas waktu nya adalah O(n), sehingga kompleksitas waktu algoritma KMP adalah O(m+n). Algoritma ini sangat cepat dibandingkan bruteforce. Algoritma ini sangat cocok untuk melakukan pencarian pada teks yang berukuran besar. Namun algoritma KMP sedikit lebih lambat ketika jumlah dari alfabet bertambah karena kemungkinan untuk mismatch nya lebih tinggi. Mismatch yang terjadi pada awal-awal karakter pada *pattern* membuat performa KMP kurang baik, karena KMP akan cepat jika mismatch nya terjadi pada akhir-akhir karakter pada *pattern*.

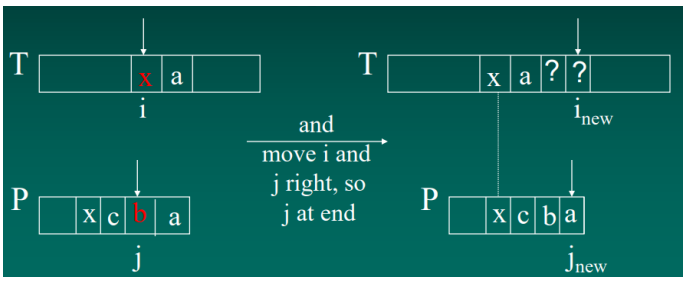
F. Algoritma Boyer-Moore Untuk Pencocokan String

Algoritma Boyer-Moore untuk pencocokan string dibangun berdasarkan 2 teknik. Pertama adalah teknik *looking-glass* dimana ketika kita mencari *pattern* P dalam sebuah teks T, pencarian dilakukan dengan mundur mulai dari karakter akhir *pattern* P menuju karakter awal *pattern* P. Kedua adalah teknik *character-jump* yaitu adalah ketika mismatch terjadi pada T[i] == x atau dengan kata lain karakter pada *pattern* P[j] tidak sama dengan T[i] maka ada 3 kemungkinan kasus.



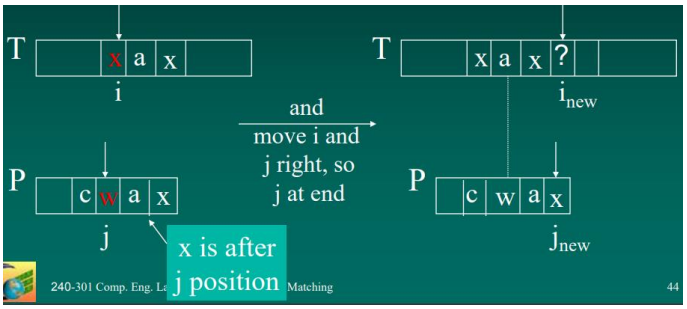
Gambar 3 Kondisi awal mismatch[4]

Kasus pertama, jika pada *pattern* P terdapat x disuatu tempat, maka geser P ke kanan sampai ketemu x dan pada posisi yang sama dengan x pada T[i].



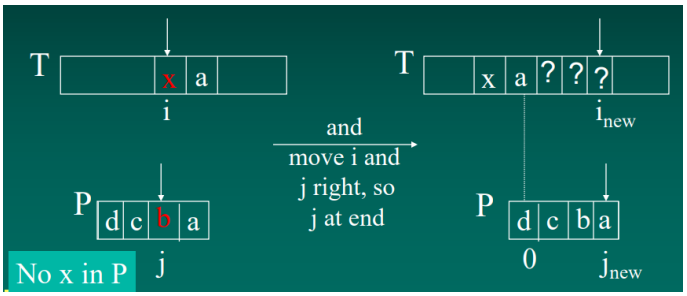
Gambar 4 Ilustrasi Kasus Pertama dari teknik *character-jump*[4]

Kasus kedua adalah jika x terdapat pada *pattern* P, tetapi tidak dimungkinkan untuk geser *pattern* kekanan karena x pada *pattern* P berada disebelah kanan posisi mismatch. Sehingga jika didapati kondisi seperti ini, maka geser *pattern* P kekanan sejauh 1 karakter saja yaitu ke T[i+1].



Gambar 5 Ilustrasi Kasus Kedua dari teknik *character-jump*[4]

Jika kasus 1 maupun kasus 2 tidak terpenuhi, maka geser kekanan hingga karakter pertama pada *pattern* P dibandingkan dengan karakter T[i+1].



Gambar 6 Ilustrasi Kasus Ketiga dari teknik *character-jump*[4]

Untuk menentukan kemunculan terakhir dari dari karakter pada T[i] yang mismatch dalam *pattern* p adalah dengan menggunakan fungsi kemunculan terakhir (*last occurrence function*). *last occurrence function* L(x) terdefinisi sebagai indeks terbesar sedemikian sehingga P[i] == x atau mengembalikan -1 jika tidak ada x dalam *pattern* P. Sebagai contoh.

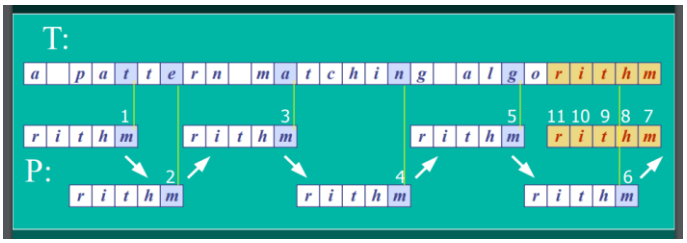
A = {a,b,c,d}

P : “abacab”

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

Gambar 7 Hasil Evaluasi Fungsi L(x)[4]

Berikut adalah contoh penggunaan algoritma Boyer-Moore dalam melakukan pencocokan suatu *pattern* pada suatu teks.



Gambar 8 Contoh Penggunaan Algoritma Boyer-Moore[4]

Kasus terburuk dari penggunaan algoritma Boyer-Moore untuk melakukan pencocokan string adalah O(nm+A). Terlihat lambat, namun ternyata algoritma Booyer-Moore sangat cepat ketika jumlah alfabet banyak, dan pelan ketika jumlah alfabet sedikit. Contohnya algoritma ini akan bagus untuk menelusuri teks-teks berbahasa inggris dan akan kurang cepat ketika menelusuri binary. Algoritma Booyer-Moore secara signifikan jauh lebih cepat dibandingkan bruteforce ketika melakukan pencarian pada teks berbahasa inggris.

G. Pencocokan String dengan Regular Expression

Regular expression adalah sebuah sekuens karakter yang dapat mendefinisikan pola dalam pencarian. Konsep ini muncul pertama kali pada tahun 1950an yang mana penggagas nya adalah Stephen Cole Kleene, matematikawan asal amerika. Regular expression banyak digunakan dalam mesin pencari, fitur *find and replace* pada mesin pengolah kata, dll. Berikut adalah pola-pola dasar dari regular expression

Symbol	Example	Definition
.	. , A.C	match any character
[]	[ABCabc] , [A-Ca-c]	match any char in the list
[^]	[^Z] , [^XYZ] , [^x-z]	match all except the chars in the list
^	^C , ^A.*	next token must be the first part of string
$	[CD]G$	prev token must be the last part of string
*	C* , [ab]*	match 0 or more copies of prev char or regular expression token
+	C+ , [ab]+	match 1 or more copies of the prev token
|	C|D	match either the 1st token or the 2nd
\(\)	\(CA\)+	combines multiple tokens into one

Gambar 9 Pola Dasar Regular Expression[5]

III. IMPLEMENTASI DAN PENGUJIAN

A. Implementasi Program

Pada percobaan ini saya dibantu dengan 2 orang teman saya mencoba untuk merealisasikan aplikasi berbasis web ini menggunakan *framework* dengan bahasa PHP yaitu Laravel versi 5.6 untuk membuat *platform* berbasis web, sedangkan algoritma penyelesaian masalah menggunakan bahasa Python. Secara umum, program kami memiliki alur sebagai berikut:

1. Tampilan beranda web yang berisi tentang program, dan formulir pencarian.
2. Pengguna memasukkan informasi yang ingin dicari, dilanjutkan dengan melakukan klik pada tombol *submit.*
3. Isian yang telah diisi tersebut akan diproses pada sebuah *controller* dengan bahasa PHP. *Controller* akan menuliskan informasi-informasi yang dibutuhkan algoritma pada sebuah file berformat JSON.

4. *Controller* juga akan mendapatkan data dari Twitter melalui API Twitter dengan otentifikasi yang telah disediakan oleh sistem secara *default* atau pun dengan pengguna. Data yang didapatkan tersebut, selanjutnya akan ditulis ke dalam file berformat JSON.
5. Selanjutnya, *controller* memanggil file berbahasa Python yang berisikan algoritma penyelesaian masalah, dan menunggu program Python menyelesaikan tugasnya.
6. Program Python akan melakukan penyelesaian masalah sesuai dengan algoritma yang dipilih oleh pengguna. Jika telah selesai, maka program ini akan menuliskan *file* dengan format JSON, dilanjutkan dengan mengirimkan sinyal bahwa proses telah selesai ke *controller*.
7. Setelah program Python selesai diproses, *controller* akan membaca *file* JSON yang telah dibuat oleh program Python sebelumnya.
8. Kemudian, *controller* akan mengirimkan data tersebut ke *view* dengan bahasa PHP, dan menampilkan hasil ke layar.

B. Implementasi PHP

Untuk bagian implementasi PHP, saya membuat 2 *HTTP Routing* yaitu url('/') dan url('/hasil'). Kedua *route* tersebut akan mengarahkan ke sebuah *controller* yaitu *HomeController.php* . Untuk *route* url('/'), *controller* akan memanggil fungsi show() dan mengirimkan sebuah *view dashboard.php* ke *browser*. Sedangkan, url('/hasil') akan memanggil fungsi process() yang nantinya akan mengirimkan *view result.php* dengan data yang dihasilkan oleh algoritma yang dipilih. Fungsi process() memiliki algoritma sebagai berikut:

1. Menerima *request* dari *form* yang memiliki *method POST*.
2. Jika isian profil diisi dengan lengkap, maka akan melakukan *overwrite* terhadap otentifikasi *default* yang telah dimiliki oleh sistem.
3. Fungsi membuat sebuah *file* JSON yang dan menulikan informasi yang berisikan algoritma yang dipilih dan banyaknya tweets yang akan diproses oleh algoritma tersebut.
4. Kemudian, fungsi ini akan memanggil API Twitter untuk mendapatkan informasi data yang dibutuhkan oleh algoritma dengan menggunakan otentifikasi yang berlaku.
5. Data yang didapatkan dari API Twitter, selanjutkan akan ditulis kedalam *file* JSON.
6. Langkah berikutnya, fungsi akan memanggil program Python untuk memproses masalah tersebut.
7. Setelah program Python menyelesaikan programnya, fungsi ini kemudian akan melakukan pembacaan terhadap file JSON yang telah ditulis oleh program Python sebelumnya.
8. Langkah terakhir yaitu fungsi akan melakukan *return* sebuah *view result.php* dengan mengirimkan data yang telah diproses tersebut dan menampilkannya ke *browser*.

C. Implementasi Python

Saya membagi file python menjadi 2 modul, yaitu Progam.py dan Algo.py.

Program.py : file python ini merupakan main program dari python yang nanti akan dibangkitkan proses nya melalui PHP. File ini berfungsi untuk membaca file JSON yang sudah ditulis oleh PHP, dan kemudian memparsing JSON tersebut ke struktur data internal python. Kemudian data-data yang didapat diproses menggunakan algoritma yang sesuai yang terdapat pada file Algo.py. Setelah itu, data-data yang sudah di proses di simpan kembali ke file JSON yang nantinya akan dibaca kembali oleh PHP dan akan ditampilkan ke web browser. Method yang terdapat pada file ini antara lain :

1. ParseTimeFromStatus(timestr) : Fungsi yang mengembalikan struct_time dengan input string yang menggambarkan waktu yang didapat dari data_twitter.json
2. TimeToStr(t) : Fungsi yang mengembalikan string yang menggambarkan waktu dengan input struct_time

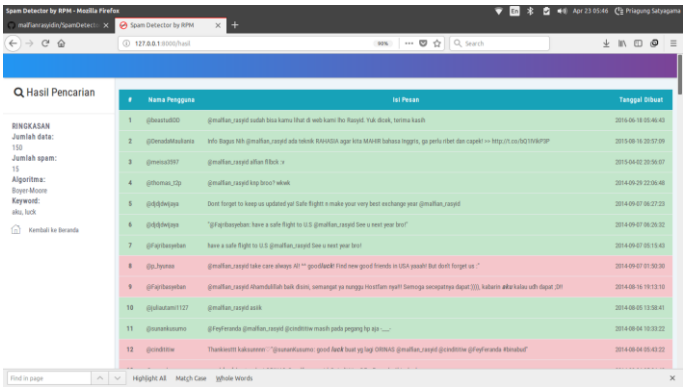
Algo.py : file python ini berisikan pendefinisian fungsi-fungsi untuk melakukan Algoritma *Pattern Matching*. Fungsi yang terdapat pada file ini di antaranya :

1. computeFail(pattern) : Fungsi yang mengembalikkan tabel fungsi pinggiran KMP dari suatu tabel
2. matchKMP(text,pattern) : Fungsi yang mengembalikan nilai True apabila suatu pattern terdapat dalam suatu teks, jika tidak, akan mengembalikan nilai False. matchKMP menggunakan pendekatan algoritma KMP dan memanggil fungsi computeFail
3. buildLast(pattern) : Fungsi yang mengembalikan tabel kemunculan terakhir kaarakter-karakter ASCII di pattern
4. matchBM(text,pattern) : Fungsi yang mengembalikan nilai True apabila suatu pattern terdapat dalam suatu teks, jika tidak, akan mengembalikan nilai False. matchKMP menggunakan pendekatan algoritma Boyer-Moore dan memanggil fungsi buildLast.
5. matchRE(text,pattern) : Fungsi yang mengembalikan nilai True apabila suatu pattern terdapat dalam suatu teks, jika tidak, akan mengembalikan nilai False. matchKMP menggunakan pendekatan algoritma Regex search dan memanfaat *library* re dari python

D. Pengujian

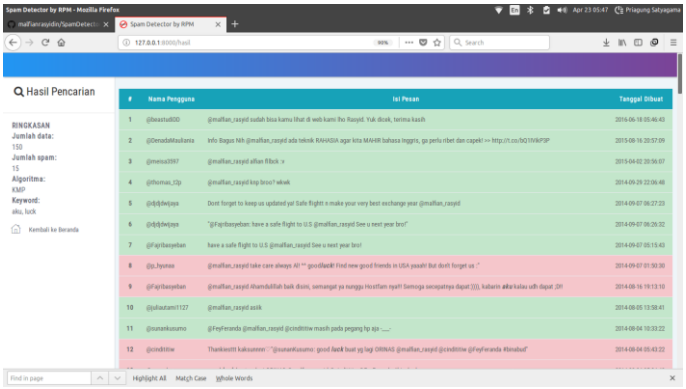
Pada pengujian kali ini, kami mencoba untuk memfilter spam dengan menggunakan kata kunci "aku" dan "luck" dengan jumlah tweet yang di filter adalah 150 tweet.

1. Algoritma Booyer-Moore



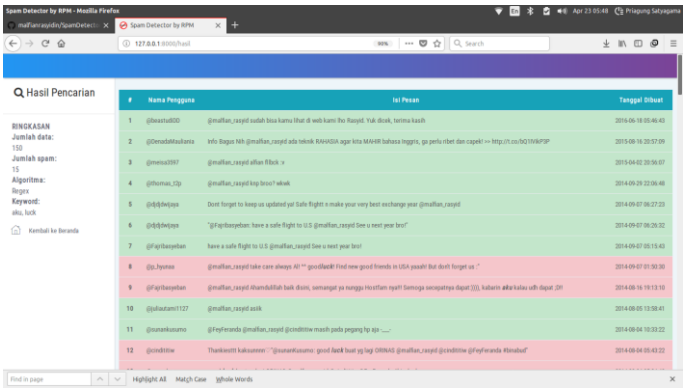
Gambar 10 Pengujian Algoritma Booyer-Moore

2. Algoritma KMP



Gambar 11 Pengujian Algoritma KMP

3. Regular Expression



Gambar 12 Pengujian Regular Expression

IV. Kesimpulan dan Saran

A. Kesimpulan

Berdasarkan pengujian yang sudah kami lakukan, kami simpulkan bahwa program kami berjalan dengan baik. Terdapat beberapa kekurangan seperti keterbatasan tweet hanya maksimal sejumlah 200 tweet. Hal itu dikarenakan limitasi dari API Twitter nya.

B. Saran

Untuk pengembangan selanjutnya, bisa ditambahkan beberapa fitur seperti pencarian spam mention untuk username tertentu, fitur pencarian spam dari timeline, dsb.

Ucapan Terima Kasih

Pertama-tama, saya ucapkan syukur kepada Allah SWT karena berkat rahmat dan karunia nya, saya diberi kesehatan dan kemampuan untuk menyelesaikan makalah ini dengan baik. Kemudian saya juga mengucapkan terima kasih sebesarbesarnya kepada tim dosen pengampu mata kuliah IF 2211 Strategi Algoritma khususnya kepada ibu Dr. Nur Ulfa Maulidevi yang telah membimbing dan menyampaikan materi terkait Strategi Algoritma. Semoga makalah ini dapat memberikan manfaat sebesar-besarnya kepada para pembaca.

Referensi

[1] Obar, Jonathan A.; Wildman, Steve (2015). "Social media definition and the governance challenge: An introduction to the special issue". *Telecommunications policy*.

[2] Kirkpatrick, David (2011). *The Facebook effect: the real inside story of Mark Zuckerberg and the world's fastest-growing company*. London: Virgin.B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.

[3] Gyöngyi, Zoltan; Garcia-Molina, Hector (2005). "Web spam taxonomy". *Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), 2005 in The 14th International World Wide Web Conference (WWW 2005) May 10, (Tue) – 14 (Sat), 2005, Nippon Convention Center (Makuhari Messe), Chiba, Japan.*

[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf, diakses pada 13 Mei 2018 pukul 13.28

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/String-Matching-dengan-Regex-2018.pdf, diakses pada 13 Mei 2018 pukul 13.28

[6] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung:Penerbit Informatika.

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, 13 Mei 2018

Priagung Satyagama
13516089