Penerapan Pencocokan String dalam Klasifikasi SMS melalui Deteksi Spam

Aloysius Gilang Pramudya - 13520147
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520147@std.stei.itb.ac.id

***Abstract*— Akhir-akhir ini Penggunaan SMS seringkali disalahgunakan oleh pihak-pihak yang tidak bertanggung jawab seperti menyebarkan SMS spam yang tidak diinginkan oleh penerima pesan. Spam dapat berupa iklan, pesan porno, penipuan, dan lain sebagainya. Deteksi spam dapat dilakukan dengan beberapa metode salah satunya dengan filter kata kunci. Paper ini membahas penerapan algoritma *String Matching* yaitu Knuth–Morris–Pratt dan Boyer-Moore dalam deteksi SMS spam.**

Keywords—SMS; spam; algoritma; string-matching.

I. PENDAHULUAN

SMS atau Layanan Pesan Singkat merupakan layanan pengiriman pesan berbasis teks yang ada pada telepon seluler. SMS masih banyak digunakan masyarakat karena akses yang mudah dan harganya yang terjangkau. Namun, seiring berkembangnya layanan ini, semakin banyak pula pihak-pihak yang tidak bertanggung jawab yang memanfaatkan layanan ini untuk menyebarkan spam seperti iklan, pesan porno, penipuan dan lain sebagainya. Spam akan menimbulkan ketidaknyamanan kepada penerima pesan karena spam dapat memenuhi kotak masuk sehingga mengurangi ruang simpan. Selain itu spam juga dapat mejadi media virus atau *malware* untuk masuk pada perangkat pengguna.

Berdasarkan laporan tahunan Global Spam Report 2021 dari Truecaller, Indonesia menepati posisi enam teratas dengan peningkatan jumlah spam yang signifikan. Indonesia juga menjadi salah satu dari 20 negara dengan jumlah spam SMS terbanyak pada tahun 2021. CEO dan Co-Founder Truecaller mengatakan bahwa masyarakat Indonesia menjadi target penipuan dengan semakin tingginya tingkat teror komunikasi yang tidak diinginkan, menyebabkan jutaan pengguna *smartphone* di Indonesia berisiko menjadi korban penipuan. Berdasarkan statistik Truecaller, sebagian besar spam berasal dari layanan keuangan mencapai 80%, iklan sales sebesar 19%, dan sisanya merupakan penipuan (*scam*) [2].

Penanggulangan spam dapat dilakukan dengan beberapa cara seperti filter kata kunci spam dan *blacklist* IPS atau nomor tertentu. Pada paper ini akan membahas dibahas penerapan algoritma *String Matching* yaitu Knuth–Morris–Pratt dan Boyer-Moore pada filter kata kunci spam. Program ini akan mencari kata pemicu spam pada teks SMS kemudian menggolongkannya menjadi spam jika terdapat kata pemicu spam.

II. DASAR TEORI

A. *Pencocokan String* [1]

Pencocokan String (*String Matching)* merupakan algoritma yang mencoba mencari satu atau beberapa *string* (disebut juga *pattern*) pada teks yang dicari. Persoalan pencocokan *string* didefinisikan sebagai berikut : 1. Diberikan teks, yaitu (*long*) *string* yang panjangnya *n* karakter. 2. Diberikan *pattern*, yaitu *string* dengan panjang *m* karakter ($m < n$) yang akan dicari di dalam teks. Akan dicari lokasi pertama di dalam teks yang bersesuaian dengan pattern. Beberapa implementasi dari algoritma pencocokan string antara lain pencarian dalam teks editor, *web search engine*, analisis citra, pengecekan plagiarisme, filter spam, dan *bioinformatics*.

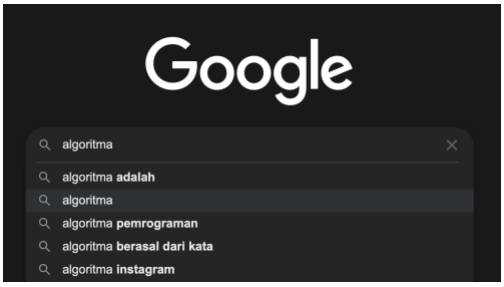


Fig. 1. Search Engine Google

Konsep *string* yang perlu diperhatikan pada algoritma pencocokan *string* adalah *suffix* dan *prefix*. Jika diberikan *string S* dengan panjang m karakter, *prefix* dari *S* adalah sebagian string (*substring*) mulai dari index 0 sampai *k* dan *suffix* dari *S* adalah sebagian string (*substring*) mulai dari indeks *k* sampai *m-1* dengan *k* suatu index antara 0 sampai *m-1*. Contoh: jika diberikan *string S* yaitu "pesan", semua kemungkinan *prefix* dari *S* adalah "p", "pe", "pes", "pesa", dan "pesan". Semua kemungkinan *suffix* dari *S* adalah "n", "an", "san", "esan", dan "pesan".

B. Pencocokan String dengan Algoritma Brute Force [1]

Algoritma yang paling simpel untuk pencocokan *string* adalah Algoritma Brute Force. Proses pencocokan *string* adalah sebagai berikut :

1. Cocokan karakter pertama dari *pattern* dengan karakter pertama dari teks.
2. Jika karakter cocok, bandingkan karakter kedua dan seterusnya hingga akhir teks.
3. Jika karakter tidak cocok, geser *pattern* ke kanan sebanyak satu karakter dan ulangi langah ke-1.
4. Jika *pattern* cocok, catat lokasi awalnya.

Contoh kasus :

Teks : ACTGCTGAGA

Pattern : AGA

	A	C	T	G	C	T	G	A	G	A
1	A	G	A							
2		A	G	A						
3			A	G	A					
4				A	G	A				
5					A	G	A			
6						A	G	A		
7							A	G	A	
8								**A**	**G**	**A**

Hasil pencarian menunjukan kecocokan dimulai pada indeks ke-7 dengan total perbandingan sebanyak 24. Satu-satunya kelebihan dari Algoritma Brute Force adalah mudah dan simpel untuk diimplementasikan.

Kasus terburuk (*worst case*) terjadi ketika kecocokan ditemukan pada akhir teks. Jumlah perbandingan pada kasus ini adalah *m(n-m+1)= O(m)* dengan *m* adalah panjang *pattern* dan *n* adalah panjang teks. Kasus terbaik (*best case*) terjadi bila karakter pertama *pattern* tidak pernah sama dengan karakter teks yang dicocokan sehingga. Jumlah perbandingan pada kasus ini maksimal sebesar panjang teks sehingga kompleksitasnya adalah O(n). Namun, pada kasus rata-rata pencarian (*average case*) algoritma ini memiliki kompleksitas O(m+n).

C. Pencocokan String dengan Algoritma Knuth-Morris-Pratt [1]

Algoritma Knuth-Morris-Pratt atau biasa disingkat KMP mencari pattern pada teks dari kiri ke kanan. Hal yang membedakan algoritma KMP dengan Brute Force adalah pergeseran *pattern* setelah menemukan ketidakcocokan yang jauh lebih efisien dari algoritma Brute Force. Algoritma KMP dapat menentukan posisi pencocokan berikutnya berdasarkan *pattern* itu sendiri. Algoritma KMP tidak perlu untuk bergerak mundur dalam pencocokan teks masukan sehingga algoritma ini bagus untuk memproses file yang berukuran sangat besar yang dibaca dari perangkat eksternal atau jaringan internet. Algoritma ini ditemukan pada tahun 1977 oleh D.E Knuth dan V.R. Pratt dan disecara independen oleh Morris, tetapi ketiganya menerbitkannya bersama-sama.

Jika diberikan sebuah *pattern* P dan teks T dan terjadi ketidakcocokan antara teks dan *pattern* pada P[j], maka pada peninjauan berikutnya *pattern* akan bergeser sebanyak *prefix* terbesar dari P[0..j-1] yang nilainya sama dengan *suffix* dari P[1..j-1] dengan j adalah indeks saat terjadi ketidakcocokan pada *pattern* dengan teks.

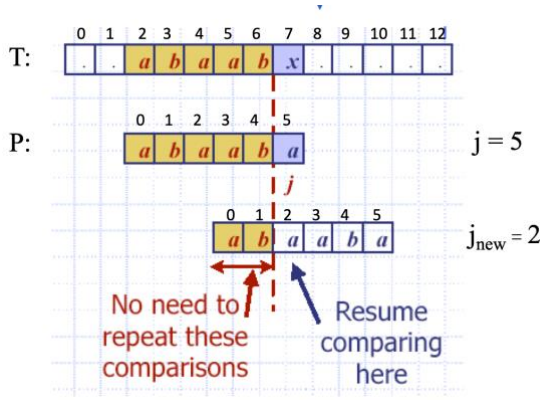


Fig. 2. Visualisasi Algoritma KMP
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Pencarian *prefix* yang sekaligus menjadi *suffix* pada *pattern* dapat direpresentasikan dengan menggunakan KMP *border function* atau bisa juga disebut *failure function*. Misal j adalah posisi ketidakcocokan dari pattern P dan k adalah posisi tepat sebelum terjadi ketidakcocokan (k = j - 1), *border function* b(k) didefinisikan sebagai panjang *prefix* terbesar dari P[0..k] yang juga merupakan *suffix* dari P[1..k]. *border function* b(k) dapat direpresentasikan dengan array pada kode program. Misalkan terdapat pattern P yaitu "abaaba", tabel *border function* akan terlihat sebagai berikut. Nilai b(4) = 1 diperoleh dengan mencari panjang *prefix* terbesar dari "abacab" yang juga merupakan *suffix* dari "baca"

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

Fig. 3. Tabel *Border Function*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Jika m adalah panjang karakter dari teks dan n adalah panjang karakter dari *pattern*, kompleksitas waktu algoritma KMP adalah sebagai berikut.

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

Contoh penyelesaian dengan algotrima KMP adalah sebagai berikut.

Fig. 4. *Contoh penyelesaian dengan KMP.*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

D. *Pencocokan String dengan Algoritma Boyer Moore* [1]

Berbeda dengan algoritma *brute force* dan KMP, algoritma Boyer Moore melakukan pergeseran *pattern* dari kanan ke kiri. Namun perbandingan karakter teks dan *pattern* tetap dari kiri ke kanan. Kunci dari algoritma ini adalah tidak memeriksa setiap karakter dalam teks berdasarkan informasi percobaan yang gagal sebelumnya. Hal ini dilakukan menggunakan tabel yang sudah dihitung sebelumnya seperti halnya *border function* pada algoritma KMP. Algoritma ini dikembangkan pada tahun 1977 oleh Robert S. Boyer dan J Strother Moore.

Algoritma ini didasarkan pada dua teknik, yaitu :

1) *The looking-glass technique*
Pencocokan *pattern* pada teks dilakukan secara mundur dari kanan ke kiri dimulai dari indeks paling akhir.

2) *The character jump technique*
Ketika terjadi ketidakcocokan pada suatu indeks, geser paterrn untuk menghindari pencocokan yang sia-sia.

Terdapat tiga kasus yang mungkin terjadi ketika ditemukan ketidakcocokan, yaitu

1) *Kasus 1*
Jika terjadi ketidakcocokan pada karakter x pada teks, maka cek apakah terdapat kemunculan terakhir (*last occurence*) dari karakter x tersebut pada *pattern*. Jika ada, maka *pattern* akan digeser ke kanan sehingga sejajar dengan *last occurence* dari karakter x pada teks.

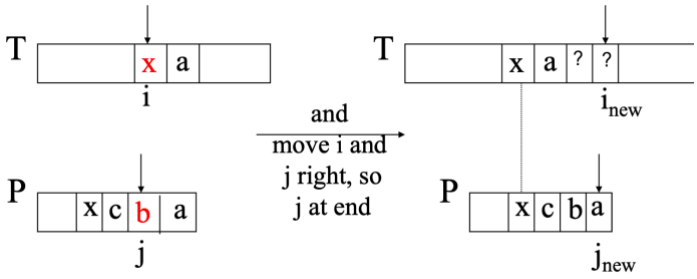


Fig. 5. *Kasus 1 Boyer Moore*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

2) *Kasus 2*

Jika terjadi ketidakcocokan pada karakter x pada teks, maka cek apakah terdapat kemunculan terakhir (*last occurence*) dari karakter x tersebut pada *pattern*. Jika pergeseran *pattern* sampai ke *last occurence* tidak mungkin, maka geser pattern ke kanan sebanyak satu karakter.

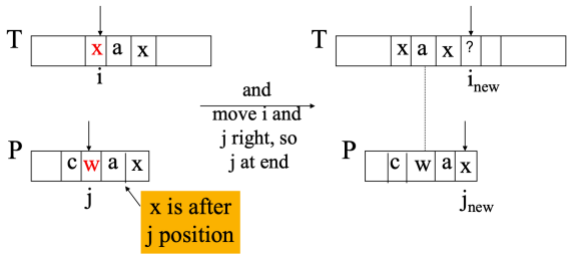


Fig. 6. *Kasus 2 Boyer Moore*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

3) *Kasus 3*

Jika tidak terdapat karakter x pada *pattern* saat terjadi ketidakcocokan, maka geser *pattern* sehingga indeks nol *pattern* sejajar dengan indeks teks satu karakter dari perbandingan sebelumnya.

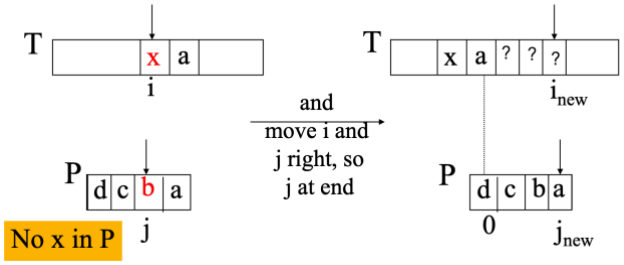


Fig. 7. *Kasus 3 Boyer Moore*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Kemunculan terakhir (*last occurence*) suatu karakter bisa didapatkan dengan *Last Occurence Function*. *Last occurence* l(x) didefinisikan sebagai indeks terbesar i ketika nilai P[i] sama dengan x. Misalkan terdapat pattern P yaitu "abacab" maka tabel *last occurence* adalah sebagai berikut.

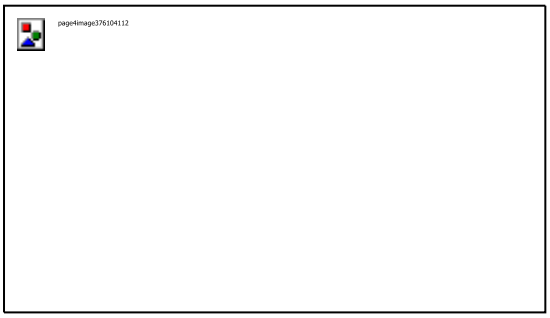


Fig. 8. Table *last occurence.*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

Contoh penyelesaian dengan algotrima Boyer Moore adalah sebagai berikut.

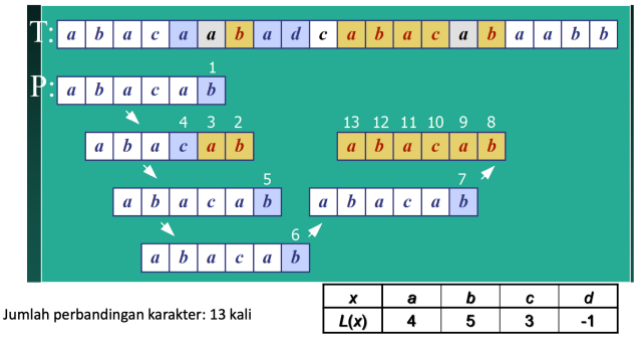


Fig. 9. *Contoh penyelesaian dengan BM.*
(Sumber : Diktat Ir. Rinaldi Munir, M.T.)

E. Spam[4]

Spam merupakan pesan yang tidak diminta atau tidak diinginkan yang mungkin mempunyai tujuan jahat atau tidak. Seiring waktu, spam hanya akan menggangu pengguna serta berpotensi menjadi ancaman untuk perangkat dan ekonomi pengguna. Beberapa jenis spam yang paling umun antara lain.

1) *Iklan*

Iklan adalah salah satu jenis spam yang paling banyak ditemukan. Beberapa contoh spam iklan yaitu iklan penurun berat badan, jasa asuransi, dan pinjaman uang. Dalam banyak kasus, spam ini bisa jadi penipuan, tetapi bisa juga nyata.

2) *Surat berantai*

Biasanya surat berantai akan berisi cerita-cerita yang menarik dan mendebarkan lalu membujuk penerima untuk menyebarkan pesan itu dengan hukuman tertentu jika penerima tidak menyebarkan pesan itu.

3) *Hoaks*

Jenis spam ini menawarkan penawaran yang ajaib dan sering kali tidak masuk akal seperti produk penurun berat badan dalam semalam. Spam ini bertujuan menarik perhatian penerima untuk masuk ke situs web yang tidak aman.

4) *Penipuan uang*

Jenis spam ini berisi janji agar mendapatkan uang dengan mudah dan cepat. Penerima disyaratkan untuk membayar sejumlah kecil uang untuk menerima hadiah besar di masa depan. Spam ini juga mungkin melibatkan kegiatan donasi palsu.

5) *Peringatan virus*

Penerima biasanya akan menerima pesan yang memperingatkan tentang infeksi virus atau *malware* pada perangkat penerima. Dalam kasus ini pengirim spam mengakatakn bahwa mereka memiliki solusi untuk masalah ini dan mensyaratkan penerima untuk memberikan beberapa informasi atau mengunduh sesuatu.

6) *Spam pornografi*

Spam pornografi banyak ditemukan karena pasarnya yang sangat luas dan memikat minat masyarakat. Spam ini bertujuan menarik perhatian penerima untuk masuk ke situs web pornografi.

F. Spam Filter[3]

Beberapa teknik untuk membantu proses filter spam adalah sebagai berikut.

1) *Filter berdasarkan konten spam*

Teknik yang paling umum digunakan untuk memblokir spam adalah dengan menyaring beberapa kata yang sering digunakan dalam pesan spam. Beberapa kata spam yang paling umum adalah bonus, kredit, hosting, dan klaim hadiah.

2) *Blacklisting ips/nomor tertentu*

Teknik lainnya yang umum digunakan adalah memasukan ips/nomor ke daftar hitam. Dengan cara ini, pengguna dapat mencegah menerima pesan lagi dari ips/nomor tersebu.

3) *Hindari memberikan informasi kontak ke publik*

Hindari memposting informasi kontak dan *form online* jika tidak perlu. Pengirim spam biasanya mendapatkan nomor hp atau email secara online.

III. IMPLEMENTASI DAN PENGUJIAN

A. Alur Program

Program yang dibuat akan mengimplementasikan algoritma Knuth-Morris-Pratt dan Boyer-Moore dan ditulis dalam bahasa pemrograman python. Program akan menerima masukan teks pesan kemudian menentukan apakah pesan tersebut merupakan spam atau tidak serta kategori spam tersebut. Alur program yang telah dibuat adalah sebagai berikut.

1) *Pengumpulan kata kunci pemicu spam*

Pada tahap ini penulis mengumpulkan kata kunci yang paling sering digunakan pada pesan spam berjenis iklan, peringatan virus dan spam pornografi. Kata kunci disimpan per kategori dalam format *text* (.txt).

TABLE I. CONTOH KATA KUNCI SPAM IKLAN

#1	slot	poker
deposit	pinjaman	jackpot
investasi	promo	100%
miliar	hosting	kredit

TABLE II. CONTOH KATA KUNCI SPAM PERINGATAN VIRUS

virus	malware	trojan
proteksi	antivirus	bahaya
android	bersihkan	deteksi
warning	peringatan	sekuriti

TABLE III. CONTOH KATA KUNCI SPAM PORNOGRAFI

viagra	dewasa	adult
erotis	perawan	hot
hottest	janda	babes
ladies	bikini	pacar

2) *Tokenisasi masukan pesan*

Masukan teks dari pengguna kemudian ditokenisasi dan dimasukan kedalam *list*. Tokenisasi merupakan proses pemecahan kalimat menjadi kumpulan kata. Kalimat "Anda mendapatkan uang

sebesar 100 juta" ketika melalui proses tokenisasi menjadi "Anda", "mendapatkan", "uang", "sebesar", "100", "juta".

3) *Case folding*

Pada tahap ini semua kata yang telah ditokenisasi akan diubah menjadi huruf kecil. Hal ini bertujuan untuk mempermudah proses pencocokan string.

4) *Penghapusan stop words*

Stop words adalah kata umum yang biasanya banyak muncul dalam teks namun dianggap tidak berguna atau tidak memiliki makna [5].

TABLE IV. CONTOH STOP WORDS INDONESIA

yang	untuk	pada
sudah	ada	ke
agar	lain	namun
jika	harus	pun

5) *Pencocokan string*

Setelah melewati tahap tokenisasi, *case folding*, dan penghapusan *stop words*, teks siap untuk diproses dalam tahap pencocokan string. Pengguna dapat memilih algoritma yang akan digunakan dalam program. Setiap kata/token dari pesan akan dicocokan dengan *pattern* dari kata kunci spam yang sudah disiapkan sebelumnya. Jika suatu kata cocok dengan *pattern* spam atau *mengandung* pattern spam, maka kemunculannya akan dihitung. Jika pesan mengandung kata kunci spam maka pesan tersebut diklasifikasikan sebagai pesan spam. Kategori spam ditentukan dari jumlah kata kunci.

B. *Pengujian*

1) *Kasus 1*

Kasus pertama masukan pesan yang bukan merupakan spam dengan algoritma KMP.

Input : "Maaf aku harus berangkat ke Bandung lebih awal, dan aku tidak ingin membangunkanmu. Aku tidak ingin membangunkanmu, jadi aku naik taksi. Akan kuhubungi kamu ketika aku sudah sampai rumah."

Fig. 10. *Kasus uji 1.*

2) *Kasus 2*

Kasus kedua memeriksa masukan pesan yang merupakan spam iklan dengan algoritma BM.

Input : Hari terakhir! Bonus pulsa Telkomsel Rp75rb/1hr (Telp & SMS sesama Tsel) dgn isi ulang pulsa min Rp100rb via MyTelkomsel App tsel.me/pulsa Promo sd 25 Feb

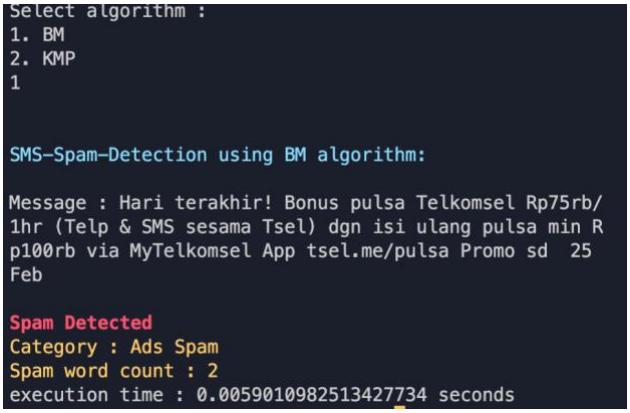


Fig. 11. *Kasus uji 2.*

3) *Kasus 3*

Kasus ketiga memeriksa masukan pesan yang merupakan spam peringatan virus dengan algoritma BM.

Input : Peringatan ! pada perangkat Vivo Y36 anda terdeteksi 7 Virus. Segera bersihkan dan proteksi perangkat anda dengan antivirus. download di www.bitly/antivirus

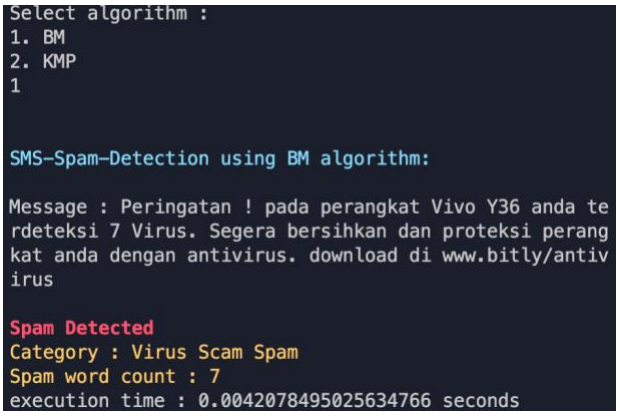


Fig. 12. *Kasus uji 2.*

IV. SIMPULAN

Pesan spam pada akhirnya hanya akan menggangu pengalaman pengguna bahkan berpotensi menyebabkan kerugian. Salah satu teknik untuk menanggulangi pesan spam adalah memfilter berdasarkan konten pesan tersebut. Algoritma pencocokan *string* seperti KMP dan BM terbukti dapat digunakan dalam proses klasifikasi pesan ini. Namun program ini masih sangat sederhana dan diperlukan penyesuaian dan pengembangan lagi jika ingin dilanjutkan ke tahap berikutnya.

GITHUB LINK

https://github.com/Aloysiusgilang/SMS-Spam-

REFERENCES

[1] Munir, Rinaldi, Pencocokan-String-2021 https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada 23 Mei 2022, pukul 14.00 WIB

[2] Industry.co.id, "Global spam report 2021: Indonesia Alami Peningkatan Jumlah telepon spam Dua Kali Lipat sepanjang 2021," *Industry.co.id*, https://www.industry.co.id/read/99509/global-spam-report-2021-indonesia-alami-peningkatan-jumlah-telepon-spam-dua-kali-lipat-sepanjang-2021. Diakses pada 21 Mei 2022, pukul 19.00 WIB

[3] Comodo.com, "Comodo News and Internet Security Information", https://blog.comodo.com/email-security/anti-spam-filtering-techniques/, Diakses pada 21 Mei 2022, pukul 19.00 WIB

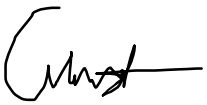
[4] Gatefy, "7 most common types of email spam and how to identify", https://gatefy.com/blog/most-common-types-email-spam/. Diakses pada 23 Mei 2022, pukul 14.00 WIB

[5] Wikipedia, "Stop word", https://en.wikipedia.org/wiki/Stop_word. Diakses pada 23 Mei 2022, pukul 18.00 WIB

PERNYATAAN

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

Bandung, 23 Mei 2022



Aloysius Gilang Pramudya
13520147