Penerapan *String Matching* dalam Kategorisasi *Email*

Muhammad Fikri Ranjabi - 13520002
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520002@std.stei.itb.ac.id

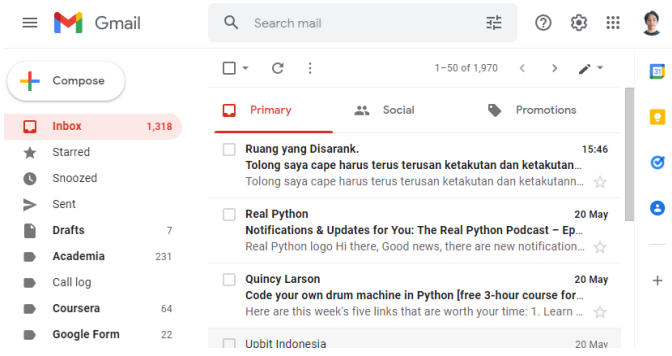
***Abstract*—*Email* merupakan sebuah alat komunikasi yang sering digunakan dalam berbagai aspek kehidupan. Setiap harinya terdapat berbagai macam jenis *email* yang masuk ke kotak masuk tanpa melalui *filter* kategori sehingga berbagai macam kategori *email* akan tergabung dalam satu kotak masuk. Saat ini *Gmail* memiliki fitur klasifikasi kategori otomatis, tetapi hanya terbatas dengan 5 kategori yang sudah disediakan. Dalam makalah ini, penulis ingin menunjukkan bagaimana implementasi klasifikasi kategori *email* memanfaatkan *string matching*.**

Keywords—string matching; email; klasifikasi kategori; brute force algorithm;

I. PENDAHULUAN

Electronic mail (email) merupakan sebuah media yang digunakan untuk bertukar pesan antara dua orang menggunakan alat elektronik. *Email* merupakan versi digital dari pengiriman pesan yang saat ini banyak digunakan sebagai sarana komunikasi utama. Setiap pengguna *email* memiliki alamat *email* unik yang digunakan untuk berkomunikasi dengan pengguna lain. Penggunaan *email* sebagai media komunikasi banyak digunakan dalam berbagai aspek kehidupan seperti bisnis, pemerintahan, edukasi, hiburan, berita, dan lain-lain.

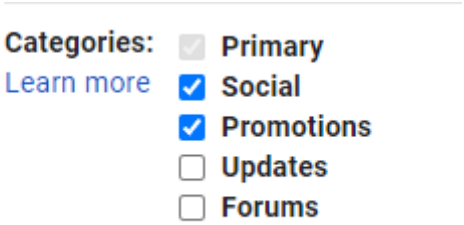
Gmail merupakan salah satu penyedia layanan *email* dari Google. Layanan ini merupakan sebuah antar muka *mail server* yang digunakan pengguna untuk mengirim, menerima, dan mengunduh *email*. Setiap pengguna harus terhubung melalui internet agar dapat terkoneksi dengan *mail server* dari Gmail.



Gambar 1.1 Antarmuka Gmail

Gmail memiliki fitur *tab* pada kotak masuk *email* yang memungkinkan pengguna untuk memisahkan *email* yang masuk berdasarkan kategori yang sudah disediakan oleh Gmail. Setiap *email* yang masuk akan disortir terlebih dahulu secara otomatis sebelum masuk ke dalam kotak masuk. Secara bawaan, kotak masuk dari Gmail akan terbagi menjadi 3 *tab* yaitu *Primary, Social, Updates, Forums, dan Promotions*.

Algoritma sortir *email* dijalankan secara otomatis oleh Gmail. Kategori *primary* akan berisikan konten utama yang sesuai dengan ketertarikan pengguna dan biasanya berisikan *email* penting seperti pesan dari teman dan keluarga. Kateogri *social* berisikan kumpulan *email* yang berasal dari berbagai media sosial seperti Facebook, Twitter, Quora, dan lain-lain. Kategori *updates* berisikan *email* yang berhubungan dengan tagihan pembayaran dan surat pernyataan. Kategori *forums* berisikan *email* yang pengguna ikuti dari berbagai forum. Kategori *promotions* berisikan *email* yang berhubungan dengan tawaran-tawaran promosi suatu produk atau surat kabar yang pengguna ikuti.



Gambar 1.2 Pilihan Kategori Email

Pengguna Gmail bisa memilih kategori apa saja yang ingin ditampilkan pada *tab* kotak masuk. Akan tetapi, pengguna tidak bisa membuat kustomisasi *tab* sendiri. Jika ingin mengatur *email* yang masuk dengan lebih spesifik, pengguna dapat memanfaatkan fitur *folder* dan *label*.

Pada makalah ini akan ditunjukkan bagaimana kategorisasi *email* dapat bekerja. Konsep yang digunakan adalah *pattern matching* dengan kumpulan kata kunci sebagai *pattern* untuk mengenali suatu kategori. Implementasi yang ditunjukkan adalah simplifikasi dari sistem kategorisasi *email* dengan memuat *email* ke dalam CSV dan dijalankan sebuah program yang dibuat dengan bahasa pemrograman Python untuk mengenali kategori dari setiap *email* yang ada berdasarkan kata kunci yang berhubungan setiap kategori.

II. LANDASAN TEORI

A. *Pattern Matching*

Pattern matching adalah proses untuk mengecek apakah suatu pola karakter terdapat dalam data yang diberikan. Secara definisi, diberikan sebuah teks yang merupakan *string* dengan panjang n karakter dan sebuah *pattern* dengan panjang m karakter beserta asumsi bahwa panjang m lebih kecil dari n. Maka masalah yang ingin diselesaikan adalah mencari lokasi pertama di dalam teks yang ada yang bersesuaian dengan *pattern*. Sebagai contoh sebuah teks "the rain in spain stays mainly on the plain" memiliki *pattern* "main" pada kata "mainly" dalam teks tersebut. Penggunaan *pattern matching* dapat ditemukan penerapannya dalam pencarian *text editor, google search engine,* analisis citra, dan *bioinformatics* (pencocokan protein pada rantai DNA).

B. *Konsep String*

Asumsikan terdapat sebuah *string* S dengan ukuran m.

$$S = x_0 x_1 \dots x_{m-1}$$

Maka *prefix* dari S adalah *substring* S[0 .. k] dan *suffix* dari S adalah *substring* S[k .. m – 1] dengan k merupakan sebuah indeks dengan nilai 0 sampai m – 1. Sebagai contoh terdapat *string* S = "andrew". Maka semua kemungkinan *prefixes* dari S adalah "a", "an", "and", "andr", "andre", "andrew" dan semua kemungkinan *suffixes* dari S adalah "w", "ew", "rew", "drew", "ndrew" , "andrew".

C. *Algoritma Brute Force*

Penerapan algoritma *brute force* dalam *string matching* adalah sebagai berikut. Akan dicek untuk setiap posisi di suatu teks T untuk dilihat apakah sebuah *pattern* P bermulai pada posisi tersebut, jika tidak maka geser 1 karakter dari *pattern* P terhadap teks T. Jika karakter pada indeks pertama dari teks T sama dengan *pattern* P, maka bandingan karakter kedua dari *pattern* P dengan teks T. Hal ini dilakukan sampai karakter terakhir dari *pattern* P. Berikut adalah simulasi pergeseran *pattern* terhadap teks yang dibandingkan.

Teks: NOBODY NOTICED HIM
Pattern: NOT

```
  NOBODY NOTICED HIM
1 NOT
2  NOT
3   NOT
4    NOT
5     NOT
6      NOT
7       NOT
8        NOT
```

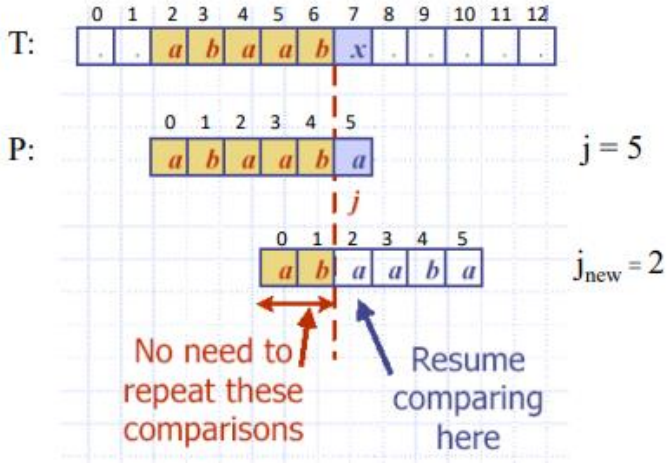
Gambar 2.1 Pergeseran Pattern Algoritma Brute Force

Pada kasus di atas, dibandingkan *pattern* "NOT" dengan teks "NOBODY NOTICED HIM". *Pattern* "NOT" cocok dengan tiga karakter pertama dari kata "NOTICED" dalam teks pada pergeseran ke-8. Dengan ini maka pada kompleksitas kasus terburuk adalah O(mn). Kasus ini dapat ditemukan jika dilakukan banyak pergeseran sampai indeks dari kata untuk mencocokan *pattern* yang mana pergeseran tersebut dilakukan saat pencocokan kata dengan *pattern* menunjukkan bahwa karakter akhir dari *pattern* tidak sama dengan yang dibandingkan pada teks. Kompleksitas kasus terbaik adalah O(n) yang terjadi jika karakter pertama dari *pattern* tidak pernah sama dengan karakter teks yang sedang dicocokkan, hal ini menghasilkan perbandingan dengan jumlah maksimal *n* kali. Kompleksitas kasus rata-rata adalah O(m+n).

Algoritma *brute force* sangat cepat jika alfabet dari teks berjumlah banyak dan lambat jika alfabet dari teks sedikit, misalnya angka *binary* karena banyak kasus yang menyebabkan pergeseran ketika sudah melakukan beberapa pencocokan.

D. *Algoritma KMP*

Algoritma KMP (Knuth-Morris-Pratt) adalah algoritma yang mencari *pattern* dari teks dengan urutan dari kiri ke kanan seperti algoritma *brute force*. Tetapi perbedaannya dengan algoritma *brute force* adalah pergeseran *pattern* dilakukan dengan lebih baik. Dengan mengimplementasikan algoritma ini, jika terjadi ketidakcocokan pada teks T dengan *pattern* P saat T[i] != P[j] maka jumlah pergeseran pattern yang dilakukan paling banyak sejumlah *prefix* dari P[0 .. j-1] yang merupakan *suffix* dari P[1 .. j-1]. Hal ini menyebabkan penghematan pada jumlah pergeseran *pattern* yang dilakukan.



Gambar 2.2 Pergeseran Pattern dari Algoritma KMP

Keuntungan KMP dibandingkan dengan algoritma *brute force* dapat dilihat pada gambar di atas. Saat terjadi ketidakcocokan pada indeks ke-5 di *pattern* dengan indeks ke-7 di teks, maka perbandingan dapat dilanjutkan dari indeks ke-2 di *pattern* dan tidak perlu membandingan dua indeks pertama *pattern* karena sudah pasti cocok dengan karakter pada teks.

Terdapat fungsi pinggiran KMP untuk mengimplementasikan hal ini. Fungsi pinggiran b(k) didefiniskan sebagai ukuran dari *prefix* terbesar P[0..k] yang

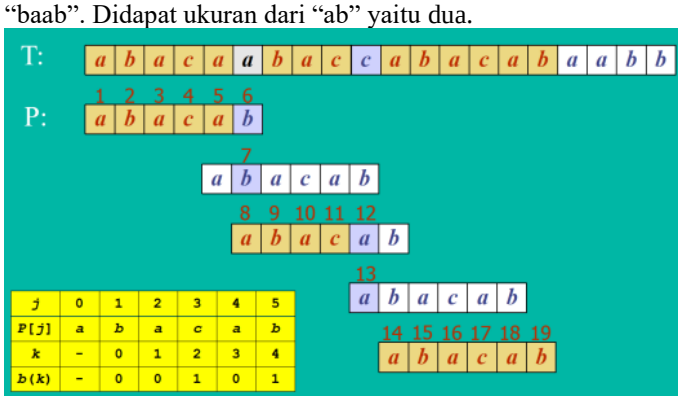
juga merupakan *suffix* dari P[1..k]. Contoh dari penerapan fungsi pinggiran KMP dapat dilihat pada gambar di bawah.

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 2.3 Tabel Fungsi Pinggiran KMP

Fungsi pinggiran direpresentasikan dalam sebuah *array*. Dapat dilihat b(k) adalah ukuran terbesar dari sebuah pinggiran. Misal pada k = 4 memiliki b(k) = 2, hal ini berarti bahwa fungsi pinggiran KMP mencari ukuran terbesar dari *prefix* P[0..4] yang juga merupakan *suffix* dari P[1..4], akan dicari *prefix* terbesar dari "abaaba" yang juga merupakan *suffix* dari "baab". Didapat ukuran dari "ab" yaitu dua.



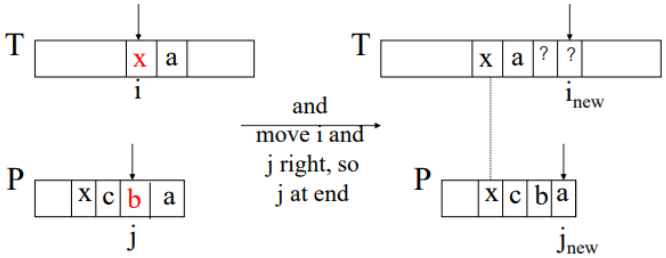
Gambar 2.4 Pergeseran Pattern Algoritma KMP dengan Fungsi Pinggiran

Dari gambar 2.4 dapat dilihat bahwa jumlah perbandingan karakter sebanyak 19 kali. KMP memiliki kompleksitas waktu untuk menghitung fungsi pinggiran yaitu O(m) dan untuk pencarian string yaitu O(n).

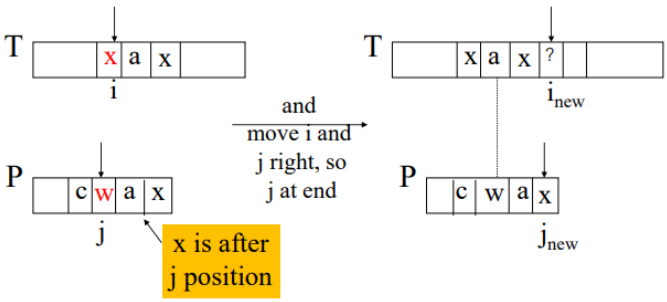
E. Algoritma Boyer-Moore

Algoritma *Boyer-Moore* adalah algoritma *pattern matching* yang didasarkan dari dua teknik. Teknik pertama adalah *the looking-glass*, akan dicari *pattern* P dalam teks T dengan bergerak mundur pada P, dimulai dari akhir P. Teknik kedua yaitu *character-jump*, yang mana ketika terdapat ketidakcocokan pada teks T[i] == x dan karakter pada *pattern* P[j] tidak sama dengan karakter pada teks T[i].

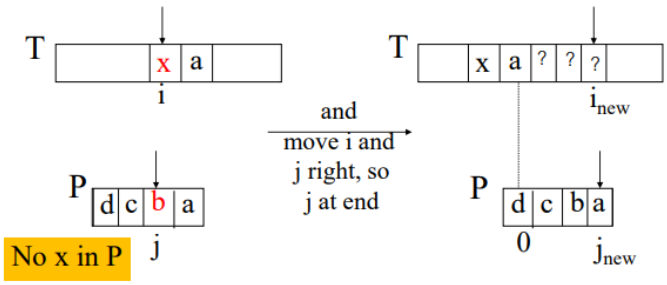
Terdapat tiga kasus dalam algoritma ini. Kasus pertama adalah ketika P mengandung x, maka geser P untuk menyesuaikan kemunculan terakhir dari x dalam P dengan T[i]. Kasus kedua terjadi jika P mengadung x pada suatu indeks, tetapi pergeseran ke kanan dari karakter terakhir teks tidak memungkinkan, maka geser P ke kanan sebanyak 1 karakter ke T[i+1].



Gambar 2.5 Kasus Pertama Algoritma Boyer-Moore



Gambar 2.6 Kasus Kedua Algoritma Boyer-Moore



Gambar 2.7 Kasus Ketiga Algoritma Boyer-Moore

Kasus ketiga dari algoritma ini terjadi ketika kasus pertama dan kasus kedua tidak dapat diterapkan, maka geser P untuk menyesuaikan P[0] dengan T[i+1]. Algoritma *Boyer-Moore* memiliki fungsi *Last Occurence* yang memroses pattern P dan alfabet A. Fungsi ini memetakan seluruh huruf dalam A menjadi *integer*. L(x) dengan x merupakan sebuah huruf dalam A didefinisikan dengan indeks terbesar i yang mana P[i] == x atau -1 jika indeks tidak ada. Fungsi *Last Occurence* dikalkulasikan ketika *pattern* P dibaca, fungsi ini disimpan ke dalam sebuah *array*.

Algoritma *Boyer-Moore* memiliki kasus terburuk dengan kompleksitas waktu O(nm + A). Keunggulan algoritma ini adalah ketika ditemukan dengan alfabet yang besar, misalkan sebuah teks berbahasa Inggris. Algoritma ini lambat ketika jumlah alfabetnya kecil, contohnya ketika ditemukan dengan *binary*. Algoritma *Boyer-Moore* lebih cepat secara signifikan daripada algoritma *Brute Force* untuk mencari *pattern* dalam teks bahasa Inggris.

III. Analisis dan Pembahasan

A. *Data Email*

Dalam melakukan implementasi, terdapat data *dummy* kumpulan *email* yang belum diklasifikasikan kategorinya. Untuk tujuan kemudahan dalam visualisasi hasil pengujian, *database email* dimuat dalam file dengan format CSV (Comma-separated values). CSV digunakan untuk menyimpan *database email* dengan setiap objek merupakan sebuah email yang memiliki atribut judul, nama pengirim, dan isi konten. Berikut adalah contoh data *email* yang digunakan:

	title	sender	content	category
2	Your Grab E-Receipt	no-reply@grab.com	GrabBike Grab Hope you enjoyed your ride! Picked up on 2 November 2021 Booking ID: A-2MEBVNDW Total Paid RP 16.000 Receipt Issued by the driver Nuryanto. 5.0 stars Compliments for driver Excellent Service Breakdown Metered fare 14.000 Platform	
3	New login to Twitter from ChromeDesktop on Windows	verify@twitter.com	We noticed a login to your account @fansperkedel from a new device. Was this you? New login Location* Penjaringan, Indonesia Device ChromeDesktop on Windows *Location is approximate based on the login's IP address. If this was you You can ignore	
4	Listing APE di Pasar BTC	id_info@upbit.com	Listing APE di Pasar BTC Trader yang terhormat, Upbit Indonesia akan menambahkan APE di pasar BTC hari ini (20 May). [Dukungan Setoran Untuk Aset Digital Baru] • Dukungan setoran untuk APE akan dibuka 30 menit setelah pemberitahuan ini. •	
5	NFT Terjual Seharga Rp 1 Triliun!	do_not_reply@mailer3.binance.com	Apa Itu NFT? Mengapa Ada Yang Terjual Senilai Rp 1 Triliun? Non-fungible Token (NFT) Merupakan Sebuah Tipe Token Kriptografi Yang Melambangkan Sebuah Aset Unik Dan Tanpa Ada Salinannya. NFT Berfungsi Sebagai Bukti Yang Terverifikasi	
6	Kami punya playlist yang pas buatmu	no-reply@spotify.com	Satu playlist untuk segala mood-mu. desain Kamu ingin menambah semangat atau justru menenangkan diri? Pasti ada playlist yang pas buatmu. Kembali ke Spotify dan dengarkan semuanya, gratis. DENGARKAN GRATIS Dapatkan Spotify untuk:	
7	Your Daily Mix dibuat khusus untukmu.	no-reply@spotify.com	Semua lagu favoritmu di satu tempat. daily mix gif Your Daily Mix adalah playlist yang dibuat khusus untukmu, berisi lagu-lagi favoritmu ❤ dan diperbarui setiap hari. Siap menjajal playlist khususmu? Kembalilah ke Spotify dan dengarkan semuanya.	
8	Buat playlist bersama-sama!	no-reply@spotify.com	Ciptakan dunia musik sendiri. Ciptakan dunia musik sendiri. Butuh mix yang pas untuk acara ulang tahun? Undang keluarga untuk membuatnya bersama-sama! Cukup buat playlist, ketuk 'Buat Kolaborasi', lalu semua bisa mulai menambahkan lagu favorit	
9	Notifications & Updates for You: Python News: What's New From April 2022	info@realpython.com	Hi there, Good news, there are new notifications & updates waiting for you in your Real Python account. Here's what you may have missed: New Tutorial Python News: What's New From April 2022 In April 2022, the PyCon US conference happened in Salt	
10	[PENTING] Mulai 1 Mei 2022, Transaksi Aset Kripto Akan Dikenakan Pajak	do_not_reply@mailer.tokocrypto.com	Hai, Tokonauts! Seperti yang kamu ketahui bahwa pemerintah akan menerapkan Pajak Kripto secara efektif mulai 1 Mei 2022. Penerapan pajak ini berdasarkan Peraturan Menteri Keuangan No. 68/PMK.03/2022 (PMK68) tentang Peraturan Pajak	
11	Career Supplement #140: How to get head-hunted + more	team@cs.resumeworded.com	Career Supplement: Your secret weapon to getting the most out of your career. If you've been forwarded this and want my next one, add your email here. Rohan from Resume Worded here. Spend 2 minutes & 40 seconds reading this and your career will be better	

Gambar 3.1 Daftar Judul, Nama Pengirim, serta Isi Konten Email yang Digunakan dalam Pengujian

B. *Klasifikasi Kategori*

Untuk mengenali kategori dari setiap email, dibutuhkan klasifikasi kategori. Klasifikasi ini membuat daftar kata-kata yang berhubungan dengan suatu kategori. Sebagai contoh, kategori *crypto* memiliki daftar kata: btc, kripto, crypto, binance, upbit, tokocrypto, nft, trading. Nantinya setiap kata yang ada pada klasifikasi kategori akan digunakan sebagai *pattern* untuk mengenali frasa yang ada dalam konten *email*.

Setiap kategori merupakan sebuah objek yang disimpan dalam tipe data *dictionary*. *Key* dari *dictionary* merupakan nama kategori dan *value*-nya adalah daftar kata yang berhubungan dengan kategori tersebut. Terdapat 6 kategori yang tersedia dengan setiap *pattern* sebagai kata kunci kategori tersebut. Berikut adalah cuplikan dari klasifikasi kategori:

```
kategori = {
  "transportation":["grab","driver","passenger","ride"]
  ,
  "social media":["twitter","instagram","facebook",
  "follower"],
  "crypto":["btc","kripto","crypto","binance",
  "tokocrypto","upbit","nft","trading","btc","eth",
  "bnb"],
  "music":["spotify","playlist","musik","lagu","album"]
  ,
  "programming":["python","github","html","css",
  "javascript","heroku"],
  "career":["linkedin","career","job","resume",
  "recruitment"]
}
```

Gambar 3.2 Klasifikasi Kategori yang Disimpan dalam Dictionary

C. *Implementasi Program*

Program klasifikasi kategori otomatis dibuat menggunakan bahasa pemrograman Python. Program ini akan membaca file CSV yang berisikan daftar *email*. Setelah dibaca, kemudian kolom kategori dari setiap *email* akan diisi dengan kategori yang bersesuaian. Daftar kategori beserta kata kunci setiap kategori diambil dari klasifikasi kategori yang sudah dibuat. Berikut adalah tampilan kode sumber dari program.

```
class Catergory_Found(Exception): pass

def get_category(email_content):
  try:
    for category, keywords in kategori.items():
      for keyword in keywords:
        if (string_matching(email_content,
        keyword)):
          raise Catergory_Found
  except Catergory_Found:
    return category, keyword
```

Gambar 3.3 Fungsi get_category

```
with open('database_email.csv',
encoding='utf8') as csv_file:
  with open('database_email_categorized.csv',
  mode='w', newline='', encoding='utf8') as
  csv_file_categorized:
    writer = csv.writer(csv_file_categorized,
    delimiter=',')
    header = ['title','sender','content',
    'category']
    writer.writerow(header)

    csv_reader = csv.reader(csv_file,
    delimiter=',')
    next(csv_reader, None)

    for row in csv_reader:
      print("Sender: " + row[1])
      email_content = row[2]
      category, found_keyword = get_category
      (email_content)
      print("Found keyword: " + found_keyword)
      print("Category: " + category)
      print
      ("-----------------------------------")

      row[3] = category
      writer.writerow(row)
```

Gambar 3.4 Kode Program Utama

Terdapat fungsi string_matching yang diimplementasikan menggunakan algoritma *brute force.* Fungsi ini akan mengembalikan nilai *True* jika pattern terdapat dalam teks. Pada fungsi *get_category* dengan parameter *email_content*, akan dikembalikan kategori yang ditemukan dalam konten *email* beserta kata kunci yang berhubungan dengan kategori tersebut.

Bagian utama program akan membuka file *database email*, setelah itu untuk setiap *email* yang ada, dijalankan fungsi *get_category* untuk mendapatkan kategori dan kata kunci yang bersesuaian. Kemudian program akan menuliskan kembali kategori yang ditemukan di file *database email* yang baru. Berikut adalah tampilan yang menunjukkan kategori beserta kata kunci yang ditemukan di setiap *email.*

```
Sender: no-reply@spotify.com
Found keyword: spotify
Category: music
-----------------------------------
Sender: info@realpython.com
Found keyword: python
Category: programming
-----------------------------------
Sender: do_not_reply@mailer.tokocrypto.com
Found keyword: twitter
Category: social media
-----------------------------------
```

Gambar 3.5 Hasil Kategori Email dengan Kata Kuncinya

Setelah program dijalankan, setiap kategori *email* akan terisi sesuai dengan kata kuncinya masing-masing. Dari gambar di atas dapat dilihat bahwa *email* yang dikirim oleh “no-reply@spotify.com” memiliki kata kunci *spotify* sehingga kategori yang sesuai adalah musik. Kemudian *email* dari “info@realpython.com” memiliki kata kunci *python* pada konten dari *email* sehingga termasuk ke dalam kategori *programming.*

Dalam salah satu *email* yang berasal dari “do_not_reply@mailer.tokocrypto.com”, walaupun *email* ini termasuk kategori *crypto*, tetapi program mendeteksi bahwa *email* ini memiliki kategori *twitter*, hal ini terjadi karena kata kunci pertama yang ditemukan dalam konten *email* adalah *twitter*, padahal kata kunci tersebut hanya menunjukkan media sosial dari *email* yang bersangkutan, bukan merupakan konteks utama dari *email* yaitu tentang *crypto.*

Kesalahan klasifikasi kategori terjadi karena urutan kata kunci dari kategori yang sesuai, dalam kasus ini kategori urutan kata kunci dari kategori yang sesuai, dalam kasus ini kategori *crypto* berada setelah kategori *social media* sehingga kategori yang dibaca oleh program adalah *social media.*

Dari percobaan yang sudah dilakukan, 3 dari 15 *email* memiliki klasifikasi kategori yang tidak sesuai. Hal ini dapat diminimalisasi dengan klasifikasi kategori yang lebih pintar, salah satu implementasinya adalah menggunakan *pool of category* yang mana kata kunci yang ada tidak langsung diklasifikasikan terhadap kategori tertentu, tetapi dikumpulkan dan kategori yang sesuai diambil dari kemunculan yang paling banyak terhadap kata kunci yang sesuai. 12 *Email* lainnya sudah memiliki kategori yang sesuai.

IV. Kesimpulan

Dari implementasi program yang dilakukan, klasifikasi kategori *email* sesuai pada 12 dari 15 *email* yang diujikan. Implementasi *pattern matching* menggunakan algoritma *brute force*, algoritma ini dipilih karena implementasi program saat ini masih merupakan skala kecil sehingga kompleksitas waktu tidak terlalu memengaruhi hasil dari pengujian.

Klasifikasi kategori tidak sesuai terjadi pada 3 *email*, kategori yang terbaca adalah *social media*, sedangkan kategori yang seharusnya dibaca adalah *crypto*. Hal ini terjadi karena untuk setiap kata kunci pertama yang ditemukan, kategori yang sesuai pada kata kunci tersebut langsung dijadikan kategori *email.*

Untuk perkembangan dari program, dapat dibuat sebuah *pool of category* yang menentukan kategori berdasarkan kemunculan kata kunci terbanyak sehingga klasifikasi kategori dapat menjadi lebih akurat. Selain itu, kata kunci beserta kategori yang ada dapat ditambah agar dapat menentukan variasi kategori *email* yang lebih beragam.

V. Ucapan Terima Kasih

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas rahmatnya, makalah dengan judul “Penerapan *String Matching* dalam Kategorisasi *Email*” ini

dapat selesai tepat waktu. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada teman-teman dan segala pihak yang telah membantu dan memberi dukungan dalam penyusunan makalah ini, terutama Ibu Dr. Nul Ulfa Maulidevi, S.T., M.Sc, Ibu Dr. Masayu Leylia Khodra, S.T., M.T, serta Bapak Dr. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma 2021/2022.

VI. REFERENSI

[1] Munir, Rinaldi. Pencocokan String (*String/Pattern Matching)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. Diakses pada 20 Mei 2022.

[2] Bonnington, Christina. Gmail's New Inbox Sorts Emails Into Tabbed Categories. https://www.wired.com/2013/05/gmail-update/. Diakses pada 20 Mei 2022.

[3] Brain, Marshall. How E-mail Works. https://computer.howstuffworks.com/e-mail-messaging/email.htm. Diakses pada 20 Mei 2022.

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, 20 Mei 2022

Muhammad Fikri Ranjabi 13520002