Penerapan Algoritma Pencocokan String untuk Pemblokiran Kata Umpatan pada *Next Word Prediction* dalam Google Keyboard

Valentinus Devin Setiadi / 13518116
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518116@std.stei.itb.ac.id

***Abstrak* - Pada era kemajuan teknologi ini, manusia lebih sering menggunakan ponsel pintar-nya dalam berkomunikasi dengan sesamanya baik melalui aplikasi pesan instan ataupun media sosial. Kita menuliskan pesan atau informasi dengan mengetiknya pada media aplikasi keyboard seperti contohnya Google Keyboard. Pada aplikasi keyboard terdapat fitur - fitur yang membantu kita dalam menuliskan kalimat demi kalimat yang ingin dituangkan, salah satunya adalah fitur *next word prediction* yang memunculkan kemungkinan kata selanjutnya yang ingin kita tuliskan. Pada fitur tersebut terdapat opsi untuk memblokir kata yang berkonotasi negatif / kata umpatan untuk dimunculkan. Hal tersebut dapat diterapkan dengan menggunakan algoritma pencocokan string seperti Bruteforce, Knuth-Morris-Pratt, Boyer Moore, dan regular expression.**

Keywords— Pencocokan string; Kata umpatan; Keyboard

I. Pendahuluan

Teknologi pada era digital seperti sekarang telah mengalami lompatan yang luar biasa, dengan pesatnya perkembangan memudahkan manusia berkomunikasi dengan sesamanya melalui aplikasi pesan instan atau media sosial. Dalam berkomunikasi dengan smartphone yang kita miliki, kita memanfaatkan aplikasi keyboard untuk menuliskan pesan yang diinginkan sehingga kita mencari aplikasi keyboard yang mana yang paling nyaman untuk digunakan. Kebanyakan orang sudah merasa cukup / nyaman dalam menggunakan aplikasi keyboard bawaan dari *operating system* smartphonenya, tetapi ada juga orang seperti saya yang lebih pemilih dalam menggunakan aplikasi keyboard tertentu dalam keseharian. Dalam hal ini saya menggunakan aplikasi yaitu *google keyboard* karena penggunaannya yang nyaman dan fitur - fitur yang terdapat di dalamnya. Salah satu fiturnya adalah *next word prediction* yang memunculkan kemungkinan kata selanjutnya yang ingin kita tuliskan, pada fitur ini terdapat pengaturan untuk memblokir kata yang berkonotasi negatif / kata umpatan untuk dimunculkan.

Manusia dalam menilai manusia lainnya cenderung melihat dari sikap maupun perkataan yang dilontarkannya. Dengan berkata kasar, manusia dapat menyakiti manusia lainnya walaupun secara tidak langsung. Mirisnya pada zaman sekarang banyak orang yang menganggap berkata kasar merupakan hal yang lumrah dalam berkomunikasi dengan sesama.

Lingkungan yang lebih sehat dapat dilihat dari kata - kata yang dilontarkan dalam berkomunikasi, untuk menciptakan hal tersebut kita harus mengurangi atau bahkan tidak menggunakan kata - kata yang berkonotasi negatif dalam dunia nyata maupun dalam dunia maya. Pada dunia maya kita bisa memanfaatkan fitur pemblokiran kata dalam *next word prediction* di google keyboard. Sebagai mahasiswa informatika, saya tertarik untuk melakukan penelitian tentang algoritma yang digunakan dalam melakukan pemblokiran kata umpatan pada google keyboard tersebut.

II. Dasar Teori

A. String

String merupakan suatu tipe data yang digunakan dalam pemrograman seperti integer dan float yang merepresentasikan angka sedangkan string merepresentasikan teks. String ini terdiri dari serangkaian character yang berupa huruf, spasi, tanda baca, maupun angka. Apabila terdapat angka dalam string, ia tidak dapat dioperasikan seperti halnya angka dalam integer dan float.

Asumsikan sebuah string dengan panjang m, string didefinisikan sebagai berikut : $S = xox1 \ldots xm-1$. Prefix dari string tersebut adalah sebuah bagian dari string atau substring dengan $S = [0. . k]$. Suffix dari string tersebut adalah sebuah bagian dari string atau substring dengan $S = [k. . m-1]$. K menunjukkan indeks apapun yang bernilai 0 sampai m-1.

Berikut merupakan contoh dari suffix dan prefix dalam suatu string:

- S (string) : “DEVIN”
- Prefix : “D” , “DE”, “DEV”, “DEVI”, “DEVIN”
- Sufix : “N”, “IN”, “VIN”, “EVIN”, “DEVIN”

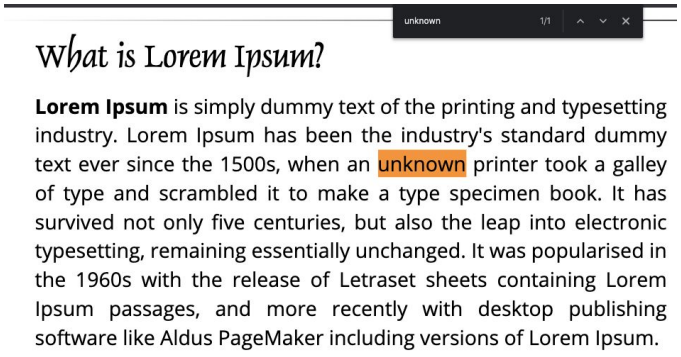
B. Pencocokan String

Pencocokan string merupakan suatu algoritma untuk menemukan kemunculan string yang disebut pola / pattern dalam sebuah string lain atau disebut teks. Dalam pencocokan string digunakan beberapa simbol yaitu :

- T : teks
- P : pola
- m : panjang pola
- n : panjang teks

Panjang pola pasti lebih lebih kecil atau sama dengan panjang teks.

String matching / pencocokan string biasanya digunakan pada fitur *search / find* yang terdapat dalam suatu aplikasi. Salah satu contoh aplikasi yang menggunakan fitur *search / find* adalah google chrome.



Gambar 1. Fitur find pada google chrome

Pada sepotong teks tersebut fitur find mencari string yang menjadi pola yaitu “unknown” pada string yang menjadi teks. Hasilnya ditemukan 1 buah match pola yang sama pada teks tersebut.

C. Algoritma Brute Force

Algoritma brute force melakukan pencocokan string pada teks dengan pattern di setiap karakternya berurutan dari awal sampe akhir / sampe menemui match pada teks.

Berikut merupakan tahapan penerapan algoritma brute force pada pencocokan string dengan asumsi bahwa teks berada di dalam array T[0..n-1] dan pattern berada di dalam array P[0..m-1] :

1. Pertama - tama T[0] akan dicocokkan dengan P[0]. Pencocokkan akan dilakukan dari kiri ke kanan.
2. Apabila karakter keduanya sama maka indeks keduanya bertambah lalu dicocokkan kembali untuk indeks selanjutnya tetapi apabila berbeda indeks pada teks bertambah dan dicocokkan dengan pattern dengan indeks 0.
3. Hal tersebut dilakukan berulang sampai menemui pattern yang cocok dengan teks atau indeks yang melebihi panjang dari teks.

Berikut merupakan contoh penerapannya :

- T (teks) :

D	E	V	I	N

n = 5 (panjang teks)

- P (pola / pattern) :

V	I	N

m = 3 (panjang pattern)

- Iterasi 1

D	E	V	I	N

↑

V	I	N

Karakter pertama pada teks tidak cocok dengan karakter pertama pada pattern sehingga indeks keduanya bertambah 1. Pada iterasi ke-2 juga sama tetapi indeks yang bertambah hanya pada teks dan indeks pada pattern tetap dari 0, dan terus melanjutkan ke iterasi ke-3.

- Iterasi 3

D	E	V	I	N

↑

V	I	N

Pada iterasi ke-3 karakter pada teks dan karakter pattern cocok sehingga indeks keduanya bertambah. Pada iterasi ke-5 telah mencapai besar dari pattern yang ingin dicari dan juga besar dari teks sehingga pencarian dihentikan serta dalam persoalan ini pattern ditemukan dalam teks.

Pada algoritma brute-force, terdapat kasus terburuk dalam pencarian yang menyebabkan jumlah perbandingan yaitu sebanyak $m * (n - m + 1) = O(mn)$. Terdapat juga kasus terbaik yang terjadi ketika karakter pertama pada pola P selalu sama dengan karakter teks T yang dicocokan, kompleksitas yang terjadi dalam notasi big-O adalah $O(n)$. Sedangkan, pada kasus rata-rata, kompleksitas yang terjadi dalam notasi big-O adalah $O(m + n)$.

D. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt atau biasa disebut KMP merupakan algoritma pencocokan string yang melakukan pencocokan string dari kiri ke kanan mirip seperti algoritma pada pencarian string dengan brute force. Terdapat sedikit

perbedaan pada algoritma Knuth-Morris-Pratt, algoritma ini dapat melakukan pergeseran lebih dari 1 karakter dengan memanfaatkan suffix dan prefix sehingga algoritma ini lebih efisien daripada algoritma pada bruteforce. Pada algoritma Knuth-Morris-Pratt ini menyimpan informasi yang digunakan nantinya untuk pergeseran dengan jumlah tertentu sehingga pola dapat bergeser lebih dari 1 karakter pada teks.

Proses awal pada pencocokan string dengan algoritma KMP adalah dengan melakukan proses kalkulasi fungsi pinggiran (The border function). Fungsi pinggiran ini mengindikasikan pergeseran terbesar yang mungkin terjadi dengan menggunakan perbandingan yang dibentuk sebelum pencarian string. Fungsi pinggiran $b(k)$ didefinisikan sebagai prefix terpanjang dari $P[0. . k]$ yang merupakan akhiran dari suffix $P[1. . k]$.Fungsi ini bergantung pada karakter dalam pattern saja, bukan pada karakter dalam teks.

Setelah membuat fungsi pinggiran, proses pencocokan string sama dengan brute force. Perbedaan terjadi disaat iterasi menemukan *mismatch* atau ketidakcocokan antara pattern dan teks dengan melakukan pergeseran sejumlah yang sesuai pada fungsi pinggiran yang telah dibuat. Berikut merupakan contoh penerapan algoritma KMP :

- Pattern : "ABABABC"
- Fungsi pinggiran :

j	0	1	2	3	4	5	6
P(j)	A	B	A	B	A	B	C
k	-	0	1	2	3	4	5
b(k)	-	0	0	0	2	3	0

- Teks : "ABABXABABABC"

A	B	A	B	X	A	B	A	B	A	B	C

↑ Mismatch!

A	B	A	B	A	B	C

Saat bertemu mismatch / ketidakcocokan pada j=4, algoritma KMP melihat fungsi pinggiran pada j=4 lalu mengambil b(k) dimana b(k) saat j=4 adalah 2, sehingga pencarian dilanjutkan dengan pattern dengan j=2.

A	B	A	B	X	A	B	A	B	A	B	C

↑

A	B	A	B	A	B	C

Dalam menghitung fungsi pinggiran, algoritma KMP memiliki kompleksitas yaitu O(m). Pada pencarian string yang dilakukan oleh KMP kompleksitasnya adalah O(n) Sehingga kompleksitas waktu algoritma KMP adalah O(m+n), hal ini menunjukkan algoritma KMP sangat cepat dibandingkan dengan algoritma brute force.

E. Algoritma Boyer Moore

Algoritma Boyer Moore merupakan algoritma pencocokan string yang melakukan pencocokan string dari kanan ke kiri tidak seperti algoritma KMP dan brute force. Algoritma Boyer Moore ini memiliki variasi lain dari pencarian stringnya yaitu dengan melompati karakter maju sejauh mungkin.

Terdapat 2 teknik dalam pencocokan string dengan algoritma Boyer Moore yaitu :

1. The Looking-glass Technique

   Teknik ini mencari pattern pada teks dengan pencarian yang dimulai dari karakter terakhir / paling kanan pada pattern.

2. The character-jump technique

   Teknik ini terjadi ketika ketidakcocokan terjadi, dengan menggeser pattern pada text sebanyak suatu nilai untuk mencegah pencocokan yang sia-sia.

Terdapat 3 jenis kemungkinan kasus sebagai berikut:

1. Kasus I

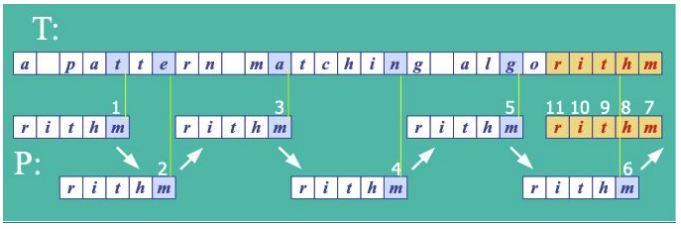
   Jika P mengandung karakter 'x' di sebelah kiri, maka geser P sehingga karakter di T[i] saat ini, yaitu 'x', sejajar dengan karakter 'x' dipola P (kemunculan terakhir karakter 'x' di pola).

2. Kasus II

   Jika P mengandung karakter 'x', tetapi sudah berada di sebelah kanan dari karakter di P yang saat ini diperiksa, maka geser P sebanyak satu karakter sehingga sejajar dengan T[i+1] dengan T[i] adalah karakter di teks yang sedang diperiksa.

3. Kasus III

   Jika kasus satu dan dua tidak berlaku, maka geser pola sehingga P[0] (karakter pertama pola) sejajar dengan T[i+1] dengan T[i] adalah karakter di teks yang sedang diperiksa



Gambar 2. Contoh penerapan algoritma Boyer Moore
Source:
http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

F. Regular Expression

Regular Expression atau yang biasa disingkat regex merupakan sebuah algoritma dan alat untuk melakukan pencarian pola yang cocok pada sebuah atau lebih karakter pada string. Kelebihan dari Regular Expression adalah pencocokan yang dilakukan tidak hanya pada sebuah pola yang statis dan spesifik, dia dapat mencocokan berbagai pola yang mirip ataupun satu himpunan kata atau karakter.

Terdapat beberapa syntax yang perlu diketahui dalam penggunaan Regular Expression :

- *Characters*

  Semua karakter, kecuali yang memiliki arti khusus dalam regex cocok dengan dirinya sendiri. Contohnya : pada regex x cocok dengan substring "x", regex 9 cocok dengan "9"; regex = cocok "="; dan regex @ cocok dengan "@".

- *Special Regex Characters*

  Karakter-karakter ini memiliki arti khusus dalam regex contohnya : ., +, *,?, ^, $, (,), [,], {,}, |, \.

- *Escape sequences*

  Dalam pencocokkan karakter yang memiliki arti khusus dalam regex, kita perlu menggunakan awalan urutan escape dengan garis miring terbalik (\). Contohnya : \. cocok dengan ".", regex \+ cocok dengan "+"; dan regex \ (cocok dengan "(". Kita juga perlu menggunakan regex \\ untuk mencocokkan "\" (garis miring).

  Regex mengenali urutan pelarian umum seperti \n untuk baris baru, \t untuk tab, \r untuk carriage-return, \nnn untuk angka oktal hingga 3 digit, \xhh untuk kode hex dua digit, \uhhhh untuk Unicode 4 digit, \uhhhhhhhh untuk Unicode 8 digit.

- *A Sequence of Characters (or String)*

  String dapat dicocokkan dengan menggabungkan urutan karakter (disebut sub-expression). Contohnya regex Saturday cocok dengan "Saturday". Pencocokan dilakukan secara default sensitif / peka terhadap huruf besar dan kecil, tetapi dapat diatur kepekaannya terhadap huruf besar dan kecil melalui pengubah.

- *OR Operator (|)*

  Operator or (|) ini dipergunakan ketika regex dapat memilih salah satu dari kemungkinan yang ada. Contohnya regex 9|nine akan menerima “9” atau “nine”.

- *Character class (or Bracket List)*

  [...]: Menerima setiap satu karakter dalam kurung siku, contohnya : [Aeiou] akan cocok dengan "a", "e", "i", "o" atau "u".

  [.-.] (Ekspresi Rentang): Menerima setiap satu karakter dalam rentang tersebut, contohnya : [0-9] cocok dengan angka apa pun dan pada [A-Za-z] akan cocok dengan huruf besar atau kecil.

  [^ ...]: Tidak termasuk karakter dalam kurung siku, contohnya : [^ 0-9] cocok dengan yang bukan digit.

- *Occurrence Indicators (or Repetition Operators)*
  - +: satu atau lebih. Contohnya : [0-9]+ akan cocok dengan angka yang berdigit satu atau lebih dari satu seperti '123', '000' dan ‘5’.
  - *: nol atau lebih. Contohnya : [0-9]* akan cocok dengan nol atau lebih banyak digit. Ia menerima semua yang ada di [0-9]+ dan string kosong.
  - ?: nol atau satu (opsional). Contohnya : [+-]? akan cocok dengan "+", "-" opsional atau string kosong.
  - {m, n}: m to n (keduanya termasuk)
  - {m}: tepat m kali
  - {m,}: m atau lebih (m +)

- *Metacharacters*
  - . (titik): karakter apapun kecuali baris baru, sama seperti [^ \ n]
  - \d, \D: satu karakter berupa digit / non-digit. Digit yang dimaksud adalah [0-9]
  - \w, \W: satu kata / karakter non-kata. Pada ASCII, karakter kata adalah [a-zA-Z0-9_]
  - \s, \S: satu karakter spasi / non-spasi. Untuk ASCII, karakter spasi putih adalah [\n \r \t \f].

G. Umpatan

Dikutip dari kbbi, umpat merupakan perkataan yang keji (kotor dan sebagainya) yang diucapkan karena marah (jengkel, kecewa, dan sebagainya) dan mengumpat adalah mengeluarkan umpat(an) atau mengeluarkan kata-kata keji (kotor) karena marah (jengkel, kecewa, dan sebagainya) serta mengutuk orang karena merasa diperlakukan kurang baik. Dari definisi kbbi dapat disimpulkan bahwa umpatan merupakan perkataan kotor yang diucapkan. Umpatan biasanya menggunakan nama - nama hewan, nama anggota tubuh, dan kata sifat yang bermakna buruk.

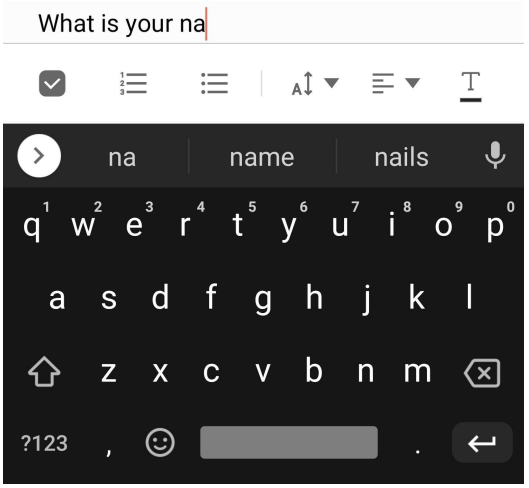
Salah satu contoh kata umpatan asing yang sering kita didengar adalah *fuck* yang memiliki arti dalam bahasa Indonesia yaitu bersetubuh. Kata ini biasanya digunakan oleh orang-orang luar negeri dalam meluapkan emosinya kepada lawan tutur sambil mengacungkan jari tengah sebagai tanda kemarahan mereka. Selain itu ada juga kata umpatan asing yaitu *bitch*, *bitch* memiliki arti pelacur dalam bahasa Indonesia, kata ini dipakai untuk mencibir perempuan yang berkelakuan tidak senonoh.

H. Google Keyboard

Gboard adalah aplikasi papan ketik virtual yang dikembangkan oleh Google untuk perangkat Android dan iOS. Keyboard virtual adalah komponen perangkat lunak yang memungkinkan input karakter tanpa perlu tombol fisik. Interaksi dengan keyboard virtual sebagian besar terjadi

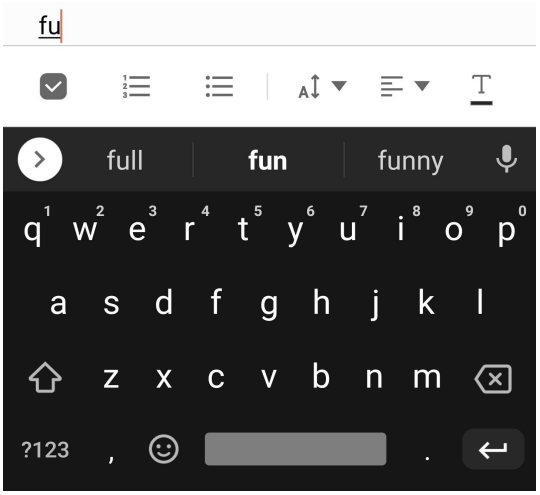
melalui antarmuka layar sentuh, tetapi juga dapat terjadi dalam bentuk yang berbeda dalam virtual atau augmented reality.

Pada google keyboard terdapat bermacam - macam fitur yang dapat kita pergunakan seperti mengatur tampilan keyboard sesuka kita mulai dari ukuran, huruf, dan warna temanya, fitur *voice-dictation,* fitur *next word prediction*, dan lainnya. Fitur *next word prediction* adalah fitur yang menampilkan kemungkinan kata selanjutnya yang ingin kita ketik untuk membantu apabila kita bingung dengan penulisan kata yang ingin ditulis. Fitur tersebut terletak di atas keyboard tempat kita mengetik, berikut merupakan gambar penggunaan fitur *next word prediction* :

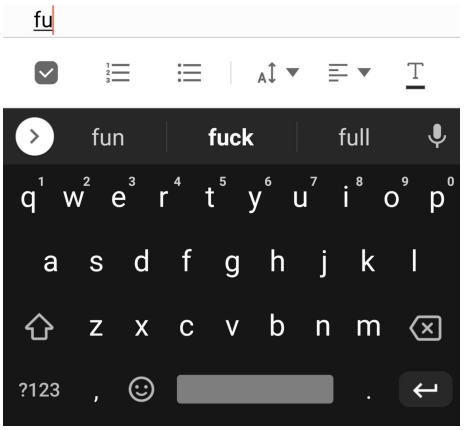


Gambar 3. Tampilan google keyboard

Pada gambar 3 dapat dilihat bahwa user sedang mengetik "na", lalu pada bagian *next word suggestion* memberikan beberapa opsi yaitu "name" dan "nails". Pada fitur tersebut terdapat pengaturan untuk mematikan *offensive word* untuk *next word prediction.* Agar dapat lebih memahami, berikut merupakan contoh penggunaannya :



Gambar 4. *Next word prediction* with *block offensive words*



Gambar 5. *Next word prediction* without *block offensive words*

III. Implementasi dan Uji kasus

Dalam bab ini akan dibahas implementasi dan uji kasus dari fitur *next word prediction* yang terdapat pada google keyboard. Pertama-tama, harus dibuat dulu 2 buah kamus / daftar yang pertama menyimpan semua kata dan yang kedua untuk menyimpan kata - kata umpatan, sehingga apabila pengaturan untuk blokir kata umpatan dinyalakan, program akan mencari apakah kata - kata prediksi yang dapat dimunculkan terdapat pada daftar kata - kata umpatan. Setelah itu, untuk mengimplementasikan pencarian tersebut penulis melakukan penerapan algoritma - algoritma pencocokan string yang telah dijelaskan di bab dasar teori dengan menggunakan bahasa pemrograman python, untuk kode pengimplementasian algoritma - algoritma beserta programnya dapat dilihat lebih lanjut pada laman berikut ini.

https://github.com/dvnvln/SwearingStringMatch

Pada kamus data pertama, penulis mengisi beberapa kata - kata yang cukup umum dan mencampurnya dengan kata - kata umpatan asing yang sering penulis dengar dalam bentuk file .txt. Pada main program, kedua file teks tersebut di-import untuk diolah dengan algoritma pencocokan string yang telah dibuat dan karena sudah adanya algoritma regex built-in di dalam python itu sendiri maka yang diperlukan hanyalah meng-import library tersebut.

Pertama, program akan meminta input-an dari user yang berupa string untuk dicari semua kemungkinan kata selanjutnya yang dapat dimunculkan. Dalam implementasi persoalan tersebut, penulis memanfaatkan algoritma regex yaitu dengan memanfaatkan re.findall.

```
def genPattern(word):
   return r"{}[a-z]+".format(word)

def findWord(masukan,kamus):
   result = re.findall
(genPattern(masukan),kamus)
   return result
```

Setelah itu, program telah memiliki list *next word prediction* yang dapat dimunculkan, tetapi pada list tersebut mungkin saja terdapat kata umpatan didalamnya, maka program memberikan opsi apakah *next word prediction* yang ingin ditampilkan mengandung kata umpatan atau tidak. Apabila user tidak ingin kata umpatan ditampilkan, maka program akan menyaring pada list *next word prediction* yang telah didapatkan dengan regex mengandung kata umpatan atau tidak. Hal tersebut dapat dilakukan dengan memanfaatkan algoritma brute force, kmp (knuth -morris-pratt), boyer moore, dan regex.

Setiap algoritma akan mengembalikan sebuah boolean yang berarti terdapat atau tidaknya dari keberadaan kata umpatan pada list *next word prediction* sebelumnya dengan cara mengecek setiap kata pada list tersebut dengan kamus data kata - kata umpatan. Pada algoritma regex, penulis memanfaatkan re.search yang akan mencari keberadaan string pada string lainnya serta mengeluarkan boolean tentang keberadaannya. Apabila method algoritma tersebut mengembalikan true maka kata yang dicek tersebut dihapus dari list. Setelah selesai pengecekan, program akan mengeluarkan semua kemungkinan kata yang dapat dituliskan.

A. Uji coba pertama : kata umpatan tidak diblokir

```
Type here : da
Block the offensive word? (y/n) : n
what algo do you want to use?
(bf(bruteforce)/kmp/bm(boyer moore)/regex) : regex
These are the suggestion words :
damn
date
day
Execution time : 0.0004842281 sec
```

Gambar 6. uji coba pertama

Pada uji coba pertama user memasukkan string “da” dengan pengaturan pemblokiran kata umpatan tidak dinyalakan. Program mengembalikan list beberapa kata yang dapat dimunculkan, yaitu : “damn”, “date” dan “day”, pada list tersebut terdapat kata umpatan yaitu “damn”. Eksekusi programnya sendiri membutuhkan waktu sekitar 0.484 ms.

B. Uji coba kedua : kata umpatan diblokir memanfaatkan algoritma brute force

```
Type here : sh
Block the offensive word? (y/n) : y
what algo do you want to use?
(bf(bruteforce)/kmp/bm(boyer moore)/regex) : bf
These are the suggestion words :
shirt
shift
Execution time : 0.0008838177 sec
```

Gambar 7. uji coba kedua

Pada uji coba kedua user memasukkan string “sh” dengan pengaturan pemblokiran kata umpatan dinyalakan. Program mengembalikan list beberapa kata yang dapat dimunculkan, yaitu : “shirt” dan “shift”. Pada list tersebut seharusnya terdapat kata umpatan yaitu “shit” ,tetapi karena pemblokiran kata umpatan dinyalakan maka kata umpatan tersebut tidak ditampilkan. Eksekusi programnya sendiri membutuhkan waktu sekitar 0.884 ms.

C. Uji coba kedua : kata umpatan diblokir memanfaatkan algoritma knuth morris pratt

```
Type here : sh
Block the offensive word? (y/n) : y
what algo do you want to use?
(bf(bruteforce)/kmp/bm(boyer moore)/regex) : kmp
These are the suggestion words :
shirt
shift
Execution time : 0.0006170273 sec
```

Gambar 8. uji coba ketiga

Pada uji coba ketiga user memasukkan string “sh” dengan pengaturan pemblokiran kata umpatan dinyalakan. Program mengembalikan list beberapa kata yang dapat dimunculkan, yaitu : “shirt” dan “shift”. Pada list tersebut seharusnya terdapat kata umpatan yaitu “shit” ,tetapi karena pemblokiran kata umpatan dinyalakan maka kata umpatan tersebut tidak ditampilkan. Eksekusi programnya sendiri membutuhkan waktu sekitar 0.617 ms.

D. Uji coba kedua : kata umpatan diblokir memanfaatkan algoritma boyer moore

```
Type here : sh
Block the offensive word? (y/n) : y
what algo do you want to use?
(bf(bruteforce)/kmp/bm(boyer moore)/regex) : bm
These are the suggestion words :
shirt
shift
Execution time : 0.0004987717 sec
```

Gambar 9. uji coba keempat

Pada uji coba keempat user memasukkan string “sh” dengan pengaturan pemblokiran kata umpatan dinyalakan. Program mengembalikan list beberapa kata yang dapat dimunculkan, yaitu : “shirt” dan “shift”. Pada list tersebut seharusnya terdapat kata umpatan yaitu “shit” ,tetapi karena pemblokiran kata umpatan dinyalakan maka kata umpatan tersebut tidak ditampilkan. Eksekusi programnya sendiri membutuhkan waktu sekitar 0.499 ms.

E. Uji coba kelima : kata umpatan diblokir memanfaatkan algoritma regular expression

```
Type here : sh
Block the offensive word? (y/n) : y
what algo do you want to use?
(bf(bruteforce)/kmp/bm(boyer moore)/regex) : regex
These are the suggestion words :
shirt
shift
Execution time : 0.0006067753 sec
```

Gambar 10. uji coba kelima

Pada uji coba kelima user memasukkan string “sh” dengan pengaturan pemblokiran kata umpatan dinyalakan. Program mengembalikan list beberapa kata yang dapat dimunculkan, yaitu : “shirt” dan “shift”. Pada list tersebut seharusnya terdapat kata umpatan yaitu “shit” ,tetapi karena pemblokiran kata umpatan dinyalakan maka kata umpatan tersebut tidak ditampilkan. Eksekusi programnya sendiri membutuhkan waktu sekitar 0.607 ms.

IV. Kesimpulan

Algoritma - algoritma pencocokan string yaitu brute force, Knuth Morris Pratt dan Boyer Moore serta Regular Expression dapat diterapkan di banyak permasalahan, salah satunya pada pemblokiran kata umpatan di papan ketik virtual google keyboard. Walaupun pada contoh implementasi penulis hanya menggunakan beberapa kata dalam kamus datanya, tetapi hal tersebut sudah dapat merepresentasikan cara kerja *next word prediction* pada google keyboard dengan menggunakan algoritma pencocokan string yang penulis pakai.

V. Video Link pada Youtube

https://youtu.be/SiHoCXbBK8I

VI. Ucapan Terima Kasih

Pertama dan yang paling utama, penulis mengucapkan syukur dan terima kasih kepada Tuhan Yang Maha Esa, dengan berkatnya yang melimpah saya mampu menyelesaikan makalah ini dengan baik. Penulis juga berterima kasih kepada Dr. Ir. Rinaldi Munir, M. T., Dr. Nur Ulfa Maulidevi, ST., M.Sc., dan Dr. Masayu Leylia Khodra, ST., MT. sebagai dosen dan pengajar dalam mata kuliah Strategi Algoritma IF2211. Tidak lupa dan tidak boleh ketinggalan, penulis juga sangat mengucapkan terima kasih kepada semua teman yang telah membantu dan memberi wawasan yang membantu dalam pembuatan makalah ini

References

[1] Munir, Rinaldi.Diktat Kuliah IF2251 Strategi Algoritmik. Institut Teknologi Bandung. 2007.

[2] Cormen, Thomas H, Charles E., Leiserson Ronald L. and Rivest Clifford Stein. Introduction to Algorithms. 3rd ed. 1989.

[3] Munir, Rinaldi. Pencocokan String. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Pencocokan-String-(2018).pdf. Diakses tanggal 30 April 2020, pukul 16.46 GMT+7.

[4] Munir, Rinaldi. Pencocokan String dengan Regular Expression (Regex). http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf. Diakses tanggal 30 April 2020, pukul 16.48 GMT+7.

[5] https://kbbi.web.id/umpat
Diakses tanggal 30 April 2020, pukul 17.00 GMT+7.

[6] https://techterms.com/definition/string
Diakses tanggal 30 April 2020, pukul 18.05 GMT+7

[7] https://www.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html
Diakses tanggal 30 April, pukul 21.00 GMT+7

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, 02 May 2020

Valentinus Devin Setiadi
13518116