Penerapan String Matching pada Fitur Auto Correct dan Fitur Auto Text di Smart Phones

Fandi Pradhana/13510049
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*author@itb.ac.id*

ABSTRAK

Auto Correct atau disebut juga Auto Text adalah suatu fitur yang biasanya dapat ditemukan di berbagai macam aplikasi seperti pemroses kata dan program untuk produk Apple, seperti iPod,iPhone dan iPad. Auto Correct ini berfungsi untuk mempersingkat waktu pengetikan sebuah kata, hanya dengan beberapa huruf maka akan keluar kemungkinan kata yang diinginkan. Dalam penggunaannya sendiri masih banyak kekurangan yang dimiliki oleh fitur Auto Correct ini sehingga malah sering mengesalkan para penggunanya.

Makalah ini akan membahas mengenai fitur Auto Correct atau Auto Text ini, bagaimana cara pencocokan huruf yang dimasukkan sehingga bisa dicari kata yang sesuai, serta bagaimana jika diterapkan algoritma-algoritma tentang string matching dalam cara kerja Auto Correct ini.

Kata Kunci : string matching, auto text,auto correct

I. PENDAHULUAN

Seiring dengan perkembangan zaman di dunia, para manusia semakin sering menciptakan berbagai macam teknologi yang mencengangkan dunia. Semakin pesatnya teknologi berkembang manusia semakin ingin membuat teknologi yang mempermudah pekerjaan-pekerjaan manusia, mulai dari pekerjaan berat sampai dengan pekerjaan yang ringan. Selain mempermudah pekerjaan, teknologi juga dapat menggantikan manusia dalam mengerjakan sesuatu (otomatisasi). Salah satu hal yang mendapatkan dampak otomatisasi adalah pengetikan kata-kata di gadget-gadget terkini.

Fitur untuk mendukung otomatisasi penulisan kata-kata ini sering disebut sebagai fitur Auto Text atau Auto Correct. Seperti yang tergambarkan oleh namanya, fitur ini berfungsi untuk membantu pengguna dalam mengetik kata-kata di suatu gadget. Kegunaan standar dari fitur ini adalah harus dapat memperbaiki kata-kata yang dianggap salah oleh sistem di dalam fitur Auto Correct ini. Dengan memeriksa huruf-huruf yang telah diketik maka sistem akan mencari dan mencocokkan huruf-huruf itu dengan huruf-huruf suatu kata yang berada di dalam database.

Input	Output
yuor	your
(r)	®
aboutit	about it
where;s	where's
bold	**bold**
italic	*italic*

Gambar 1 : Tabel pembenaran kata

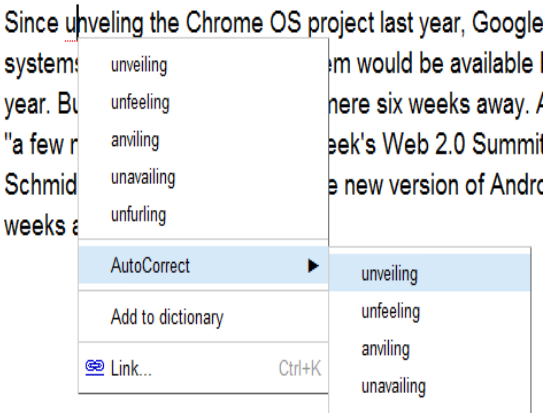
Selain itu fitur Auto Text dan Auto Correct ini juga memiliki fungsionalitas lainnya , yaitu fitur untuk memberikan sugesti kata (suggestion). Dengan mengetikkan beberapa huruf atau seluruh huruf maka sistem akan mencari ke dalam database apakah ada sebuah kata yang memenuhi kriteria dari huruf-huruf yang dimasukkan. Jika setelah dicek dan ternyata di dalam database ada maka akan muncul gambar yang menunjukkan list dari suggestion yang ada. Pengecekan itu sendiri berdasarkan dari kecocokan huruf dari kata yang dimasukkan user dengan list kata-kata yang ada di database.

Gambar 2 : Contoh Auto Correct di iPhone

Untuk semakin mempersempurnakan fungsionalitas yang dimilki oleh fitur Auto Text dan Auto Correct ini, maka disediakan fungsionalitas lainnya. Fungsionalitas tersebut adalah fungsionalitas untuk menambahkan kosa kata sesuai dengan keinginan kita. Dengan adanya fungsionalitas penambahan kata ini maka fungsionalitas lainnya yaitu fungsionalitas penulisan otomatis dan pengkoreksian otomatis akan bisa bekerja dengan lebih tepat dan baik.

II. MASALAH YANG DIHADAPI

Pada awalnya fitur yang serupa dengan auto text dan auto correct ini masih dapat bekerja dengan baik dan masih belum mendapatkan permasalahan yang cukup berarti. Bahkan karena kegunaannya yang sangat mempermudah para pengguna, teknologi ini terus dikembangkan mulai yang tadinya hanya berupa spellchecker biasa hingga berkembang sehingga dapat memeriksa contextnya juga atau lebih dikenal dengan Grammar Checker. Akan tetapi ketika teknologi ini mulai merambah ke dunia smart phones, mulailah teknologi ini mengarah ke permasalahan yang sangat serius. Terjadilah banyak sekali kasus-kasus dimana teknologi ini malah mengubah kata-kata si pengguna menjadi kata-kata aneh yang kadang-kadang bahkan dapat menimbulkan masalah yang sangat serius.



Gambar 3 : Penggunaan autocorrect di awal perkembangannya

Walaupun kasus-kasus yang sering terjadi akibat auto text ini mungkin dapat kita tertawakan karena terlihat lucu, akan tetapi pada kenyataannya ini bisa menimbulkan hal-hal yang tidak diinginkan. Kenapa masalah ini bisa terjadi di teknologi autocorrect yang sebenarnya sudah bertahun-tahun lalu diimplementasikan. Salah satu alasan yang ditemukan adalah karena keyboard smart phone yang sangat kecil dan letaknya sangat-sangat berdekatan, hal ini membuat autocorrect menyarankan kata-kata alternatifyang mengandung huruf yang berdekatan di pengetikan kata yang salah. Ini artinya lingkup kesalahan menjadi sangat-sangat besar dibandingkan autocorrect yang terdapat di device-device lainnya dimana lebih bergantung dengan grammarnya yang diperiksa, berbeda dengan smartphone yang menggunakan permeriksaan berdasarkan huruf.

Gambar 4 : Kasus yang timbul akibat Auto Correct

Selain karena faktor teknis hardware tersebut saya juga dapat menyimpulkan bahwa penggunaan algoritma dalam pengembangan fitur auotcorrect di smart phone masih kurang efektif. Kesalahan yang sering timbul sekarang bisa terjadi mungkin karena algoritma yang dirancang untuk membentuk fitur autocorrect ini pada saat pembuatannya tidak ikut mempertimbangkan faktor keyboard smartphone yang kecil, sehingga walaupun algoritmanya sebenarnya berfungsi dengan baik tetapi karena faktor keyboard inilah sehingga muncul suatu masalah. Penggunaan algoritma dalam fitur ini masih bisa disesuaikan dengan kondisi hardwarenya jadi kesalahan-kesalahan fatal akibat ulah autocorrect ini bisa diminimalisir kemunculannya.

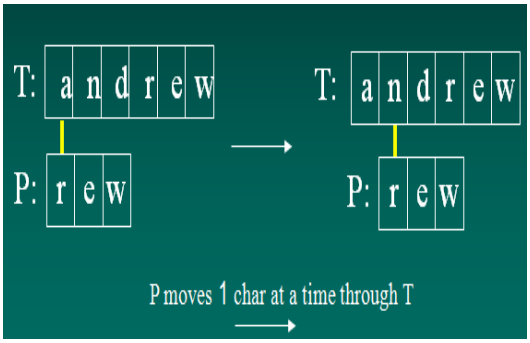
III. METODE

A. Brute Force

Pencocokan string ini dapat dilakukan dengan algoritma Brute Force , dengan menggunakan brute force amaka akan dilakukan pencarian secara bertahap di seluruh rangkaian string yang ada hingga dapat ditemukan rangkaian string yang cocok dengan pattern masukan user. Saat pemeriksaan antara pattern dengan text tidak cocok maka pemeriksaan di teks akan bergeser satu lalu diperiksa lagi apakah pattern mulai dari titik itu atau tidak. Apabila sampai string telah habis belum juga ditemukan rangkaian karakter yang sesuai dengan pattern masukan dari user, maka masukan dari user tersebut dianggaptidak ditemukan. Jika melihat mekanisme penccocokan string yang dimulai dari posisi awal karakter string, maka jika rangkaian karakter yang sesuai dengan pattern terdapat di akhir string maka algoritma ini akan menjadi worst case.

Contoh Algoritma brute-force :

```
Public static int brute (string text, string pattern)
{
    Int n = text.length(); //cari panjangnya text
    Int m = pattern.length(); //cari panjangnya pattern
    Int j;
    For (int i = 0; i <= (n-m); i++) {
         J = 0;
        While ((j< m) && (text.charAt(i+j) == pattern.charAt(j)))
        J++;
        If(j == m)
            Return i; //kecocokan ditemukan di i
        }
         Return -1;  //tidak ada yang cocok
}
```



Gambar 5 : mekanisme pencocokan string dengan brute force

B. Algoritma Knuth-Morris Prat

Berbeda dengan brute force dimana teknik iterasi yang dilakukan benar-benar pengecekan biasa yang diulang-ulang terus-menerus hingga ditemukan bagian yang cocok dengan pattern. Pada algoritma Knuth-Morris Prat tekniknya masih mirip dengan burte force dimana pergesaran dilakukan dari kiri ke kanan , akan tetapi pada algoritma Knuth-Morris Prat pergeseran dilakukan dengan cara yang lebih pintar dibandingkan dengan algoritma brute force. Pergeseran dilakukan dengan mencari sebuah sub string yang sama di awal dan di akhir pattern dan saat terjadi ketidak cocokan maka akan dilakukan pergeseran dengan cara yang unik sehingga tidak perlu terjadi lagi pengecekan ulang yang tidak perlu seperti di dalam algoritma brute force.

Algoritma Knuth-Morris Pratt ini juga menggunakan fungsi pembatas (border function) yang digunakan untuk menghitungletak suatu urutan karakter dimana perbandingan harus dilakukan. Border function ini dihitung dengan cara menghitung panjang prefix yang ada di sebuah pattern yang sama dengan suffixnya. Secara singkatnya algoritma kmp bekerja seperti ini :

a. Pencocokan karakter dari kiri ke kanan
b. Mencari prefix terpanjang dari P[0..j-1] yang juga merupakan suffix dari P[1..j-1], untuk menghindari pergeseran yang tidak perlu
c. Hasil dari pencarian prefix terpanjang disimpan dalam tabel yang disebut juga sebagai Failure Function.
d. Misalkan panjang string yang telah diperiksa dan cocok = N dan nilai dari Failure Function adalah M, maka dilakukan pergeseran sebanyak (N-M).

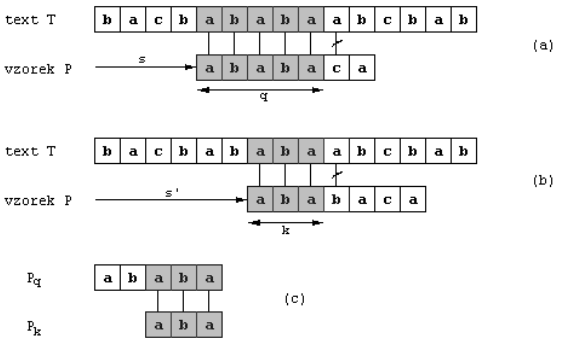
Contoh Algoritma Knuth-Morris Pratt :

```
public static int kmpMatch (string text, string pattern)
{
    Int n = text.length();
    Int m = pattern.length();

    Int fail[] =  computeFail (pattern);

    Int i = 0 ;
    Int j = 0 ;

    While (i < n) {
      If (pattern.charAt(j) == text.charAt(i)) {
           Return i - m + 1; //match
      i++;
      j++;
    }else if (j > 0)
      J = fail[j-i];
        else
          i++;
    }
   Return -1; // no match
} // end of kmpMatch()
```



Gambar 6 : mekanisme pencocokan string dengan Knuth-Morris Prat

C. Algoritma Boyer-Moore

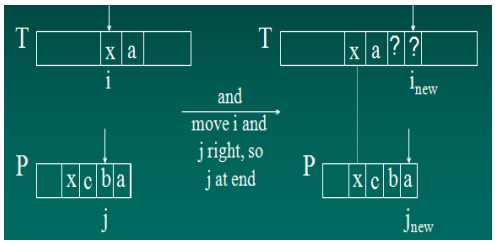
Algoritma Boyer-Moore adalah algoritma pencocokan string yang dapat digunakan dalam permasalahan autocorrect ini. Algoritma Boyer-Moore adalah algoritma pencocokan string yang dilakukan dengan menerapkan 2 teknik yaitu :

1. Teknik looking-glass
2. Character-jump

Teknik looking-glass adalah mencari suatu P di dalam kata T dengan cara mengecek mundur melalui P, mulai dari yang paling akhir.Teknik character-jump adalah

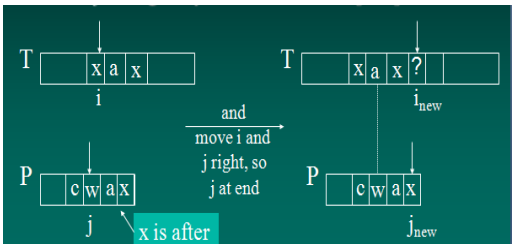
teknik pelompatan ke target pengecekan berikutnya , terdapat 3 kasus yang mungkin :

1. Jika pattern P mengandung x , maka pindahkan P ke kanan sehingga sebaris dengan kemunculan terkahir dari x di P dengan T[i]



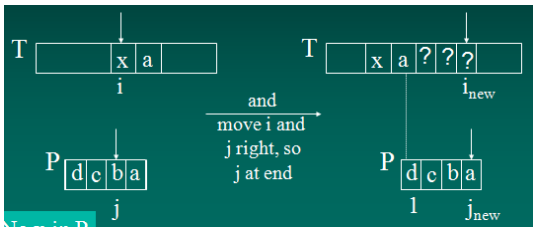
Gambar 7 : Kasus 1 Boyer-Moore

2. Jika P mengandung x di suatu tempat, tetapi pergeseran tidak dimungkinkan maka geser P ke kanan sebanyak 1 karakter ke T[i+1].

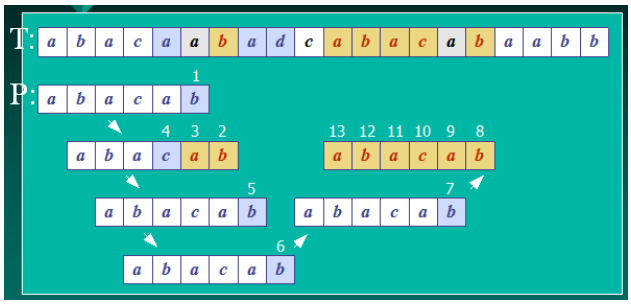


Gambar 8 : Kasus 2 Boyer-Moore

3. Jika kasus 1 dan 2 tidak di lakukan maka geser P agar membuat p[1] sebaris dengan T[i+1]



Gambar 9 : Kasus 3 Boyer-Moore



Gambar 10 : Contoh dari Boyer-Moore

Algoritma Booyer-Moore :

```
public static int bmMatch(String text,
                          String pattern)
 {
  int last[] = buildLast(pattern); //mengembalikan index
//dari last occurence tiap ASCII char di pattern

  int n = text.length();
  int m = pattern.length();
  int i = m-1;

  if (i > n-1)
   return -1; // no match if pattern is
              // longer than text

  int j = m-1;
  do {
   if (pattern.charAt(j) == text.charAt(i))
    if (j == 0)
     return i; // match
    else { // looking-glass technique
     i--;
     j--;
    }
   else { // character jump technique
    int lo = last[text.charAt(i)];  //last occ
    i = i + m - Math.min(j, 1+lo);
    j = m - 1;
   }
  } while (i <= n-1);

  return -1; // no match
 } // end of bmMatch()
```

IV. PENGGUNAAN STRING MATCHING PADA FITUR AUTOCORRECCT DI SMARTPHONE.

Dalam menjawab permasalahan dimana sering sekali terjadi kesalahan pengetikan yang disebabkan oleh fitur autocorrect. Dilakukanlah analisis permasalahan terlebih dahulu, pertama kesalahan yang timbul seringkali dikarenakan kesalahan pengetikn satu huruf lalu dilakukan autocorrect oleh program. Yang kedua kita juga mengetahui bahwa banyak kesalahan yang terjadi karena database yang digunakan sebagai acuan(pattern) tidak sesuai dengan kebiasaan mengetik si pengguna. Yang ketiga adalah karena untuk memasukkan kata ke dalam database cukup repot maka seringkali user menjadi malas menggunakan fitur ini dikarenakan masih sering terjadi kesalahan karena database yang ada adalah kata baku dan untuk mengeditnya cukup merepotkan user.

Untuk menjawab hal-hal diatas dapat dibuatlah suatu metode yang dapat membantu fitur autocorrect ini untuk menjadi fitur yang lebih baik lagi. Untuk menjawab permasalahan pertama mengenai kesalahan yang timbul

karena masalah kesalahan pengetikan lalu diperparah oleh fitur autocorrect ini. Solusinya adalah dengan membuat suatu aturan baru dimana harus dilakukan string matching menggunakan algoritma Brute Force saat melalukan pengecekan dan menggunakan algoritma Booyer-Moore saat mengeluarkan suggestion kata. Mengapa menggunakan algoritma Brute Force saat melakukan pengecekan? hal ini dimaksudkan agar ketika dilakukan pengecekan kita dapat mendapatkan hasil yang benar-benar akurat dengan mengecek tiap karakter satu-persatu, dengan mengecek satu persatu begitu terdapat suatu karakter yang tidak sesuai dengan kata-kata yang ada di database maka akan langsung keluar kemungkinan kata-kata yang benar, dengan begitu pengkoreksian kata dilakukan ditengah-tengah pengetikan bukan di akhir pengetikan kata. Dengan begitu kesalahan yang timbul ketika kita selesai mengetik sebuah kata lalu terauto koreksi dapat diminimalisir.

Kenapa dalam mengeluarkan suggestion kata harus menggunakan Boyer-Moore ? hal ini dilakukan dengan mengingat cara kerja Boyer-Moore dimana pengecekan dilakukan dari karakter paling belakang. Dengan begitu tiap selesai mengetik satu karakter akan dilakukan pengecekan dengan boyer-moore lalu dicari kata-kata yang mendekati dengan bentuk kata yang ingin diketik di dalam database. Dengan metode ini maka suggestion yang keluar diharapkan dapat memenuhi harapan si user.

Permasalahan yang terakhir adalah permasalahan memasukkan kata-kata baru dalam database. Hal ini dapat dilakukan dengan memeriksa setiap ada kata yang ditulis oleh user dilakukan pengecekan string dengan algoritma apapun. Jika didapatkan presentasi yang tidak tinggi (dibawah 70%) maka kata tersebut akan dimasukkan ke database sementara yang isinya adalah kata-kata dan suatu angka yang menunjukkan berapa kali kata tersebut ditulis oleh user. Saat angka menunjukkan lebih dari 25 kali pengetikan maka secara otomatis kata tersebut akan dipindahkan kedalam database utama.

V. KESIMPULAN

Pada suatu fitur autocorrect yang ada dan telah beredar sekarang sebenarnya sudah dapat dianggap bagus. Walaupun begitu dengan mengotak-atik penggunaan string matching dalam fitur tersebut maka kita dapat mengoptimalkan penggunaan dari fitur ini. Dari ketiga algoritma yang ada saya sudah membahas mana yang lebih baik menurut saya walaupun begitu algoritma lainnya sebenarnya bisa digunakan juga dalam membangun fitur string matching . Solusi yang saya kemukakan di makalah ini sebenarnya efektif untuk menjawab permasalahan akan tetapi waktu yang dibutuhkan untuk mendapatkan solusi sepertinya masih sangat banyak karena saya berfokus untuk mengatasi masalah kesalahan pengkoreksian dan kesalahan pemberian suggestion.

REFERENSI

[1] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma"
[2] hotword.dictionary.com/autocorrect
[3] www.switched.com/2010/07/16/cell-phone-autocorrect-how-it-works-and-why-you'll-never-avoid-embarrasing texts
[4] answers.yahoo.com
[5] apple.stackexchange.com

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, 21 Desember 2012

Fandi Pradhana