Penerapan Algoritma Boyer-Moore untuk Pengecekan Plagiatisme Source Code

Redya Febriyanto[1], Sherry Bayu Zulfianto[2], Hanindyo Wirotomo[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14014@students.if.itb.ac.id[1], if14032@students.if.itb.ac.id[2], if14047@students.if.itb.ac.id[3],

Abstrak

Metode pencocokan string dapat diterapkan dalam berbagai macam hal. Salah satu contoh penerapannya adalah untuk membandingkan dua source code untuk mengetahui apakah terdapat kesamaan diantara kedua source code tersebut. Namun, berbeda dari teks biasa, source code memiliki sifat-sifat khusus. Tingkat kesamaan antar source code tidak dilihat dari karakter per karakter pada teks tersebut, namun lebih ke kesamaan alur program. Oleh karena itu strategi untuk menentukan tingkat kecocokannya agak berbeda dibandingkan pencocokan string biasa. Dalam makalah ini penulis berusaha menerapkan metode pencocokan string (Boyer-Moore) untuk mencari kata kunci dalam sebuah source dan kemudian melakukan penghitungan sederhana untuk menentukan tingkat kesamaan dua buah source.

***Kata kunci**: plagiatisme, source code, Boyer-Moore,Pencocokan String*

1. Pendahuluan

Praktik plagiatisme (penjiplakan) dalam pembuatan sebuah program cukup sering terjadi terutama di kalangan akedimisi, khususnya mahasiswa. Hal ini sangat tidak pantas dilakukan. Oleh karena itu penulis berusaha memaparkan strategi pendeteksian praktik plagiat dalam source code dengan menggunakan algoritma pencocokan string Boyer-Moore.

Makalah ini bertujuan untuk memaparkan sebuah strategi sederhana pendeteksian plagiatisme source code. Penulis masih membuka kemungkinan-kemungkinan lain penyelesaian masalah tersebut dengan metode yang jauh lebih mangkus. Penelitian lebih lanjut masih sangat diperlukan.

2. Ruang Lingkup

Penulis sengaja memilih menggunakan algoritma Boyer-Moore karena lebih efektif dibandingkan algoritma lainnya dalam penyelesaian masalah ini. Pembuktiannya dapat dilihat di http://www.cs.nyu.edu/cs/faculty/cole/papers/CHPZ95.ps yang telah dibuktikan oleh Richard Cole yang telah membandingkan dengan kekompleksitasan algoritma lain.

Dalam makalah ini teknik pendeteksian sengaja dikhususkan pada source code program yang memiliki sifat yang berbeda dengan teks pada umumnya. Oleh karena itu, strategi pendeteksian berdasar pada alur program bukannya kesamaan tekstual program.

3. Prinsip Kerja Algoritma Boyer-Moore

Satu hal yang menarik pada algoritma Boyer-Moore ini adalah pembandingan karakter dalam sebuah string yang dilakukan dari belakang ke depan. Jika dia membandingkan teks “MAKALAH” misalnya, algoritma ini melakukan pengecekan apakah karakter ke tujuh dari teks yang dibandingkan adalah karakter ‘H’. Jika karakter ke tujuh adalah ‘H’ maka ia akan melakukan pengecekan apakah karakter sebelumnya (ke-6) adalah ‘A’. Demikian seterusnya hingga menemukan bahwa karakter pertama adalah ‘M’.

Alasan kenapa Boyer-Moore melakukan pengecekan dari belakang akan lebih jelas jika kita mengamati apa yang terjadi jika pengecekan menghasilkan nilai yang tidak sama. Misalnya, algoritma mendapati karakter ke-8 dengan karakter ‘X’ bukannya ‘H’. ‘H’ tidak muncul sama sekali pada “MAKALAH”, dan ini berarti tidak ada kesamaan sama sekali karakter ‘X’ dengan semua karakter dalam “MAKALAH”. Sehingga, dengan hanya melakukan sekali pengecekan pada karakter ke-8 kita dapat mengabaikan pengecekan karakter ke-1 sampai ke-7 dan langsung melanjutkan pengecekan karakter dimulai dari karakter ke-9, tepat setelah ‘X’.

Algoritma ini pada awalnya melakukan perhitungan sebuah tabel untuk menentukan banyaknya ‘loncatan’ karakter yang akan dilakukan

setelah mendapati sebuah perbandingan yang tidak cocok.

Tabel dibentuk dengan mengisi masing-masing karakter yang ada dengan posisinya yang terbesar pada string yang dicari. Sementara karakter yang tidak ditemukan pada string diisi dengan nilai nol. Sebagai contoh berikut ini tabel yang menggambarkan hasil pengisian untuk string "MAKALAH"

Tabel "last":

a	1
b	0
c	0
d	0
e	0
f	0
g	0
h	7
i	0
j	0
k	4
l	2
m	7
n	0
o	0
p	0
q	0
r	0
s	0
t	0
u	0
v	0
w	0
x	0
y	0
z	0

Proses pengisian tabel tersebut dapat dijelasakan sebagai berikut. Karakter pertama dalam string "MAKALAH" adalah 'M' maka dalam tabel pada karakter M diisi dengan nilai 1 (indeksnya dari awal string) demikian seterusnya hingga karakter terakhir. Proses pengisian tabel yang terjadi adalah proses penyimpaan sehingga hanya nilai terbesarlah yang pada akhirya tersipan dalam tabel. Berikut ini algoritmanya dalam pseudo-code.

```
{init:  last table}
{last adalah tabel, pat=string}
i <- 1
while i <= m do
        last[pat[i]] = i
        i := i + 1
end
```

Setelah tabel selesai diisi maka langkah selanjutnya adalah membandingkan karakter demi karakter dalam teks yang ada. Pembandingan dimulai pada karakter ke-N dimana N adalah panjang string yang dicari. Ketika pembandingan menemukan kecocokan maka ia akan melanjutkan ke karakter sebelumnya. Demikian seterusnya hingga ditemukan bahwa karakter pertama dalam string cocok (yang berarti string ditemukan).

Ketika pembandingan menemukan ketidakcocokan, maka proses akan dilanjutkan dengan meloncat ke beberapa karakter sesudahnya berdasarkan tabel yang ada. Jumlah loncatan dapat dihitung sesuai dengan algoritma sebagai berikut.

```
{teks=teks      dimana      string      dicari
kemunculannya}
while i <=n-m+1 do
        i <- 1
        while i <= n-m+1 do
                j <- j-1
        end
        if j=0 then return true
        i <- i + max(1,j-last[text[i+j-1]])
end
return false
```

4. Prinsip Kerja Pendeteksian Plagiatisme

Pada dasarnya proses pendeteksian plagiatisme menggunakan penghitungan jumlah prosedur/fungsi, loop, if dan variable dengan pembobotan sederhana. Aplikasi melakukan scanning keyword-keyword tertentu (menggunakan Boyer-Moore) untuk menentukan bagian dari source yang memenuhi kondisi tertentu.

Aplikasi pada awalnya akan mencari keyword yang mendeklarasikan sebuah prosedur atau fungsi (syntax tergantung bahasa yang digunakan). Kemudian menghitung nilai loop, if, dan variabel dengan menghitung jumlah masing-masing tipe alur program.

Setelah melakukan penghitungan untuk satu buah prosedur/fungsi maka aplikasi akan melanjutkan menghitung untuk seluruh prosedur/fungsi yang ada pada kedua source

program dan menyimpannya masing-masing ke dalam sebuah array.

Langkah terakhir adalah membandingkan elemen array dari source yang dicurigai melakukan plagiatisme dengan source yang asli. Setiap elemen dicari kesamaan jumlah elemen-elemennya dengan elemen dari array dari source asli. Jika didapati ada prosedur/fungsi yang sama maka ia akan mencatatnya sebagai suatu kesamaan. Setelah selesai maka hasil akhir kemungkinan tingkat kesamaan source code dapat ditampilkan dalam bentuk persentase yang dihitung dari: Jumlah kesamaan/total prosedur source yang dicurigai.

Contoh penghitungannya adalah sebagai berikut.

elemen 1 array I(objek)
jumlah loop=6
jumlah conditional=5

elemen ke-N array II (pembanding)
jumlah loop=6
jumlah conditional=5

Karena didapati ada elemen yang sama dengan elemen pertama array yang diambil dari source yang dicurigai. maka akan dihitung sebagai sebuah kesamaan. Algoritma akan terus melakukan pencarian masing-masing elemen dari array objek ke dalam array pembanding. dan jika telah selesai maka akan dapat dihitung kemungkinan plagiatisme yang terjadi.

Misalnya didapati 7 dari sepuluh elemen dalam array objek ada kesamaannya dengan elemen pada array pembanding. Maka tingkat plagiatismenya dapat dihitung berdasarkan rumus:

$$\frac{\sum kesamaan}{\sum elemen\_objek} \times 100\% \qquad (1)$$

sehingga didapatkan nilai 70%. Nilai ini menggambarkan tingkat kesamaan sebuah source dengan source lain. Batasan penentuan sebuah source merupakan produk plagiatisme atau bukan sangat relatif dan dapat ditentukan sendiri sesuai kebutuhan.

Secara sederhana berikut coba kami rumuskan pseudo-code nya:

```
function BoyMoore(I:String S1):String
{fungsi pencarian kata kunci prosedur/fungsi
dalam S1, mereturn String sebuah
procedure/fungsi dengan lengkap dimana
procedure/fungsi ditemukan, mereturn NULL
jika tidak menemukan prosedur/fungsi}

procedure Count(I:String S, O:
Nloop:integer, Nif:integer,)
{menghitung kemunculan masing-masing loop,
if dalam string S}

procedure Calculate(I: A1, A2: array of
proc, O: hasil:integer){
 melakukan pembandingan dan kalkulasi elemen
dari kedua array dan menghitung prosentase
akhir
}


ALGORITMA UTAMA
found : boolean
S1, S2 : string
AX, AY : array of proc
i : integer
hasil : real

found <- true
i <- 1
while found
       if (BoyMoor(proc/func,teks)=0) then
              found <- false
       else
              S<-BoyMoor(proc/func,teks)
              rember (S,teks)
              Count(S,AX[i].Nloop,
              AX[i].Nif)
              i <- i+1
end
i <- 1
found <- true
while found
       if (BoyMoor(proc/func,teks)=0) then
              found <- false
       else
              S<-BoyMoor(proc/func,teks)
              rember (S,teks)
              Count(S,AY[i].Nloop,
              AY[i].Nif)
              i <- i+1
end
Calculate(AX, AY, hasil)
```

4. Kesimpulan

Pada dasarnya dua proses utama yang terdapat dalam strategi pendeteksian plagiatisme source code disini adalah String Matching, dan Counting (penghitungan). Khusus untuk kasus pencocokan string, algoritma yang digunakan adalah algoritma Boyer-Moore yang termasuk algoritma pencocokan String yang paling mangkus terutama untuk kasus String yang cukup panjang. Sementara proses counting dan pencocokan algoritma mengunakan algoritma yang paling sederhana.

Srategi pendeteksian praktik plagiatisme yang telah dijelaskan diatas menghasilkan nilai-nilai relatif yang dapat diinterpretasikan dengan bebas dan sesuai tingkat kebutuhan pengguna. Seorang pemakai dapat menentukan tingkat yang diangap batas praktik plagiatisme.

Praktik plagiatisme source code dapat dideteksi dengan menggunakan metode-metode yang cukup sederhana namun efektif. Tetapi metode ini akan mudah diantisipasi jika semua orang mengetahui metode yang digunakan dalam pendeteksian.

5. Referensi

1. Munir, Rinaldi.2005. *"Strategi Algoritmik"*. Departemen Teknik Informatika, Institut Teknologi Bandung.
2. http://en.wikipedia.org
3. http://www-sr.informatik.uni-tuebingen.de/~buehler/BM/BM1.html.Tanggal 18 Mei 2006.
4. http://www.cs.nyu.edu/cs/faculty/cole/papers/CHPZ95.ps
5. Paris, Maeve. *"Source Code And Text Plagiarism Detection Strategies "*. www.ics.ltsn.ac.uk/pub/conf2003/Maeve%20Paris.PPT .Diakses tanggal 17 Mei 2006 pukul 19.00 WIB.