Penerapan Algoritma Knuth-Morris-Pratt dan Boyer-Moore dalam Mendeteksi Plagiarisme pada Artikel Blog

Muhammad Ayyub Abdurrahman 13518076
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (13518076@std.stei.itb.ac.id):

***Abstract*— Plagiarisme adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri Plagiarisme sangat sering ditemukan, baik oleh individu maupun kelompok. Salah satu konten di Internet yang paling banyak diduplikasi adalah blog. Sebagai salah satu algoritma string matching, algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore dapat digunakan sebagai dasar utama untuk mendeteksi terjadinya plagiarisme pada artikel blog.**

Keywords—Plagiarisme, String Matching, Knuth-Morris-Pratt,Boyer-Moore

I. PENDAHULUAN

Plagiarisme atau biasa disebut sebagai plagiat menurut Kamus Besar Bahasa Indonesia adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri. Sedangkan menurut Peraturan Menteri Pendidikan Nasional Republik Indonesia No. 17 Tahun 2010, plagiat adalah perbuatan secara sengaja maupun tidak sengaja dalam memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu karya ilmiah dengan mengutip sebagian atau seluruh karya dan/atau karya ilmiah pihak lain yang diakui sebagai karya ilmiahnya tanpa menyatakan sumber secara tepat dan memadai

Plagiarisme sangat sering ditemukan, baik oleh individu maupun kelompok. Internet merupakan sarana utama yang biasa digunakan pelaku plagiarisme untuk melakukan tindakan tersebut. Pada tahun 2013, Matt Cutts mengatakan bahwa 25-30% konten di internet bersifat duplikatif. Duplikasi konten di Internet memang sangat mudah dan menggiurkan, sehingga menimbulkan rasa candu untuk melakukannya. Salah satu konten di Internet yang paling banyak diduplikasi adalah blog.

Walaupun menplagiarisasi konten cukup mudah, namun plagiarism memberikan efek buruk kepada banyak pihak. Banyak blogger khawatir tentang duplikat konten dan bagaimana hal itu dapat mempengaruhi peringkat situs mereka. Meskipun tidak semua konten yang digandakan buruk, jika sebuah situs mengungguli Anda dan menggunakan konten yang sama, maka itu bisa menjadi situasi sulit yang perlu diperbaiki. Plagiarisme juga dinilai mampu menurunkan tingkat kreativitas seseorang untuk berkarya. Oleh karena itu, beberapa pihak telah melarang dan mengatur hal ini agar tidak dilakukan dengan menerapkan peraturan tertulis sebagai ketentuan pembuatan suatu tugas atau karya.

Sayangnya, untuk mendeteksi konten diduplikasi atau tidak lumayan sulit, karena artikel tersebut harus dipindai seluruhnya dan cukup memakan waktu jika dibandingkan kalimat per kalimat. Salah satu cara terbaik untuk memerangi konten yang disalin adalah menggunakan teknologi. Kita dapat mendeteksi kehadiran bagian dari sebuah artikel pada artikel lainnya dengan menggunakan metode string matching. Sebagai salah satu algoritma string matching, algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore dapat digunakan sebagai dasar utama untuk mendeteksi terjadinya plagiarisme pada artikel blog.

II. LANDASAN TEORI

A. Pencocokan String

String merupakan deretan simbol yang tidak tertentu panjangnya, yang dianggap sebagai panjang satu unit. String dapat tersusun atas beberapa hal, baik itu berupa huruf, angka, karakter khusus, maupun karakter unicode. String matching atau bisa disebut pencocokan string merupakan langkah pencarian kemunculan string pendek dengan ukuran n yang disebut sebagai pattern dalam suatu string panjang dengan ukuran m yang dinamakan text, dengan n bernilai lebih kecil atau sama dengan m.

Dalam pencarian string tersebut, kita dapat menggunakan berbagai jenis algoritma yang dapat disesuaikan berdasarkan pendekatan tertentu. Berikut ini adalah beberapa algoritma yang biasa digunakan pada string matching.

- Algoritma Brute force
- Algoritma Boyer-Moore
- Algoritma Knuth-Morris-Pratt (KMP)
- Algoritma Aho-Corasick
- Algoritma Rabin-Karp

B. Algoritma Knuth-Morriss-Pratt

Algoritma KMP memanfaatkan informasi yang didapatkan dari partial match sepanjang j karakter saat

pencocokan pattern dimulai dari posisi i sehingga kita mengetahui karakter apa saja yang ada di T[i]..T[i + j - 1]. Dari informasi ini, terdapat dua kemungkinan untuk menghemat jumlah iterasi yang dilakukan.

Pertama, kita dapat melangkahi beberapa iterasi yang tidak memiliki solusi yang mungkin dengan cara mencoba mencocokkan partial match yang sebelumnya telah ditemukan pada iterasi selanjutnya.

i=2: w a w

i=3: w a w o

Pada kasus di atas, dua buah pattern saling berlawanan. Kita mengetahui bahwa pada i = 2, T[3] dan T[4] adalah “a” dan “w”, sehingga tidak mungkin “w” dan “a” pada i = 3 adalah pattern yang kita cari. Kita dapat melangkahi satu atau lebih iterasi sampai kita menemukan sepasang pattern yang tidak saling berlawanan

i=2: w a w

i=3: w a w o

Pada kasus di atas, terdapat sepasang “w” yang beririsan. Kita definisikan bahwa irisan dari dua buah string x dan y adalah urutan karakter terpanjang yang merupakan akhiran dari x dan awalan dari y. Pada kasus ini, irisan dari “waw” dan “wawo” adalah “w”. Secara umum, nilai i yang ingin kita langkahi adalah nilai i yang berkorespondensi terhadap irisan terpanjang dari partial match saat ini.

Selain optimasi di atas (yang merupakan optimasi untuk melangkahi iterasi luar yang tidak perlu), kita juga dapat melakukan optimasi untuk melangkahi iterasi dalam yang tidak perlu.

i=2: w a w

i=4: w a w o

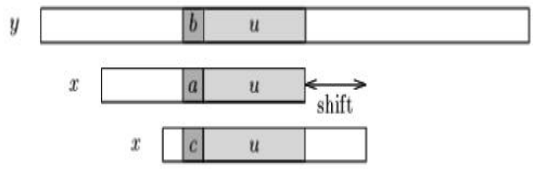
Misal, pada kasus di atas, terdapat pasangan karakter “w” yang beririsan dan diuji oleh iterasi i = 2. Seharusnya, kita tidak perlu lagi melakukan pengujian pada i = 4. Jika kita sudah memiliki pattern irisan dengan partial match sebelumnya dalam suatu iterasi, kita dapat mengabaikan pengujian sepanjang karakter pattern yang beririsan sebelumnya.

Algoritma KMP ini melakukan paling banyak 2n perbandingan, dengan kompleksitas waktu O(n).

C. Algoritma Boyer-Moore

Algoritma Boyer-Moore dinilai sebagai algoritma yang paling efisien. Algoritma ini bekerja dengan memindai karakter-karakter dari pattern yang dimiliki dari ujung paling kanan teks hingga ujung paling kiri, dimulai dari potongan string yang paling kanan dari teks tersebut. Jika string tidak cocok atau string cocok seluruhnya (bukan partial match), algoritma ini akan menggunakan dua buah fungsi untuk menggeser pattern pengujian string matching: good-suffix shift (matching shift) dan bad-character shift (occurence shift).

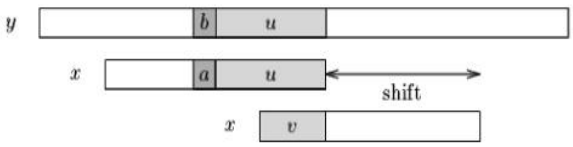
Misal, kita memiliki string x yang merupakan pattern yang ingin kita cari dan string y yang merupakan string teks. Kita asumsikan terjadi ketidakcocokan string pada karakter x[i] = a pada pattern dan karakter y[i+j] = b pada string teks pada sebuah pengujian di posisi j.



GAMBAR 1. GOOD-SUFFIX SHIFT

Sumber : https://www-igm.univ-mlv.fr/~lecroq/string/node14.html diakses 2204-2020 pukul 3.27

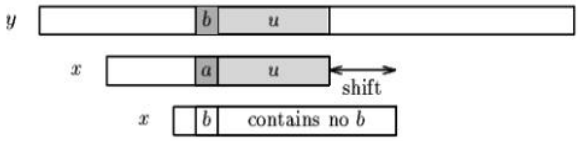
Lalu, x[i + 1 ... m - 1] = y[i + j + 1 ... j + m - 1] = u dan x[i] ≠ y[i + j]. Good-suffix shift menyejajarkan y[i + j + 1 ... j + m - 1] = x [i + 1 ... m - 1] dengan partial match pada x yang didahului oleh karakter yang berbeda dari x[i] (lihat gambar di atas). Jika partial match seperti contoh di atas tidak ada, pergeseran yang terjadi adalah pergeseran pattern agar urutan karakter akhir v terpanjang dari y[i + j + 1 ... j + m - 1] sejajar dengan awalan x yang cocok (lihat gambar 2.2).



GAMBAR 2. GOOD-SUFFIX SHIFT

Sumber: https://www-igm.univ-mlv.fr/~lecroq/string/node14.html diakses 22-04-2020 pukul 3.27

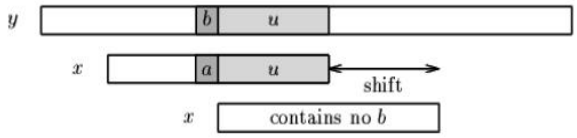
Bad-character shift terdiri dari proses menyejajarkan string teks y[i + j] dengan urutan karakter paling kanan yang cocok dari x[0 ... m-2], seperti yang dapat dilihat di gambar di bawah ini:



GAMBAR 3. BAD-CHARACTER SHIFT

Sumber : https://www-igm.univ-mlv.fr/~lecroq/string/node14.html diakses 22-04-2020 pukul 3.27

Jika y[i + j] tidak ada pada pattern x, tidak ada urutan karakter pada x yang ada di y dan dapat memenuhi y[i + j], dan ujung kiri dari posisi pengujian disejajarkan dengan y[i + j + 1].



GAMBAR 4. BAD-CHARACTER SHIFT

Sumber : https://www-igm.univ-mlv.fr/~lecroq/string/node14.html diakses 22-04-2020 pukul 3.27

Algoritma BM ini melakukan paling banyak 3n perbandingan, dengan kompleksitas waktu O(n/m).

D. Plagiarisme

1. Definisi Plagiat

Berdasarkan Peraturan Mentri Pendidikan RI Nomor 17 Tahun 2017 bahwa Plagiat adalah perbuatan sengaja atau tidak sengaja dalam memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu karya ilmiah, dengan mengutip sebagian atau seluruh karya dan atau karya ilmiah pihak lain yang diakui sebagai karya ilmiahnya, tanpa menyatakan sumber secara tepat dan memadai. Jika dilihat dari Kamus Besar Bahasa Indonesia atau KBBI plagiat adalah pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan (pendapat) sendiri.

2. Ruang Lingkup Plagiarisme

Berdasarkan kedua definisi diatas dapat ditarik kesimpulan bahwa ruang lingkup plagiarisme yaitu :

a. Mengutip kata-kata atau kalimat orang lain tanpa menggunakan tanda kutip dan tanpa menyebutkan identitas sumbernya.

b. Menggunakan gagasan, pandangan atau teori orang lain tanpa menyebutkan identitas sumbernya.

c. Menggunakan fakta (data, informasi) milik orang lain tanpa menyebutkan identitas sumbernya.

d. Mengakui tulisan orang lain sebagai tulisan sendiri.

e. Melakukan parafrase (mengubah kalimat orang lain ke dalam susunan kalimat sendiri tanpa mengubah idenya) tanpa menyebutkan identitas sumbernya.

f. Menyerahkan suatu karya ilmiah yang dihasilkan dan /atau telah dipublikasikan oleh pihak lain seolah-olah sebagai karya sendiri.

3. Tipe Plagiarisme

Plagiarisme memiliki beberapa tipe. Menurut Soelistyo[6] tipe-tipe tersebut yaitu :

a. **Plagiarisme Kata demi Kata (Word for word Plagiarism)**. Penulis menggunakan kata-kata penulis lain (persis) tanpa menyebutkan sumbernya.

b. **Plagiarisme atas sumber (Plagiarism of Source**). Penulis menggunakan gagasan orang lain tanpa memberikan pengakuan yang cukup (tanpa menyebutkan sumbernya secara jelas).

c. **Plagiarisme Kepengarangan (Plagiarism of Authorship).** Penulis mengakui sebagai pengarang karya tulis karya orang lain.

d. **Self Plagiarism**. Termasuk dalam tipe ini adalah penulis mempublikasikan satu artikel pada lebih dari satu redaksi publikasi. Dan mendaur ulang karya tulis/ karya ilmiah. Yang penting dalam self plagiarism adalah bahwa ketika mengambil karya sendiri, maka ciptaan karya baru yang dihasilkan harus memiliki perubahan yang berarti. Artinya Karya lama merupakan bagian kecil dari karya baru yang dihasilkan. Sehingga pembaca akan memperoleh hal baru, yang benar-benar penulis tuangkan pada karya tulis yang menggunakan karya lama.

E. Artikel Blog

Artikel menurut Kamus Besar Bahasa Indonesia atau KBBI merupakan karya tulis lengkap, misalnya laporan berita atau esai dalam majalah, surat kabar dan sebagainya. Tujuan umum dari sebuah artikel dibuat adalah untuk mendidik, meyakinkan, memberitahu hingga menghibur pembacanya dan dipublikasikan melalui media cetak atau pun media online.

Sedangkan Blog adalah website yang mengandung konten personal dalam bentuk artikel, video, foto, dan link ke website lain yang disediakan oleh penulis blog. Blog adalah salah satu jenis website yang kontennya berisi pemikiran satu atau beberapa penulis dan memiliki urutan posting secara kronologis (dari konten terbaru ke konten terlama).

III. IMPLEMENTASI DAN PEMBAHASAN

Implementasi aplikasi pendeteksi plagiarism dilakukan dengan menggunakan bahasa pemrograman python. Secara umum, aplikasi akan menerima inputan 2 buah text, lalu memisahkan kalimat perkalimat, lalu mencari kalimat tersebut di artikel yang lain pada blog yang lain. Sebelumnya, blog dipindahkan dulu menjadi format text(.txt).

Algoritma KMP dan Boyer-Moore akan digunakan untuk menemukan pola pada source code yang diberikan untuk mendeteksi plagiarisme. Pola yang digunakan pada pemeriksaan menggunakan algoritma ini adalah potongan paragrafatau keseluruhan artikel yang dicurigai sama dengan atau terdapat pada sebuah artikel blog tertentu. Tingkat kesamaan bisa diatur untuk menyesuaikan ketentuan plagiarisme, namun saat ini kita akan mengasumsikan pola yang sama persis dengan yang terdapat pada artikel berita akan mengindikasikan plagiarisme.

Algoritma KMP dan Boyer-Moore yang digunakan pada makalah ini merupakan modifikasi dan pengembangan dari algoritma yang telah dibuat pada tugas sebelumnya (Tugas Kecil 4). Algoritma dibuat dengan paradigma pemrograman berorientasi objek dengan menggunakan bahasa Python. Berikut ini adalah rincian terkait implementasi algoritma KMP dan Boyer-Moore dalam mendeteksi plagiarism:

A. *Algoritma Knutt-Morris-Pratt*

Algoritma KMP memanfaatkan informasi yang didapatkan dari partial match sepanjang j karakter saat pencocokan pattern dimulai dari posisi i sehingga kita mengetahui karakter apa saja yang ada di T[i]..T[i + j - 1]. Algoritma KMP memiliki 3 buah parameter, yakni pat, text dan p. pat adalah pattern yang ingin dicari pada text, dan p adalah jumlah kecocokan kalimat pada text tersebut.

Di awal program, diinisialisasi beberapa variable seperti m yang menghitung panjang pattern, n untuk menghitung panjang text, I dan j sebagai iterator, lps sebagai pendukung dalam optimasi pergeseran. Setelah variable diinisiasi, text diiterasi perkarakter, dan akan ada 2 kemungkinan untuk tiap iterasi.

Kemungkinan pertama pattern sama dengan kalimat. Jika kasus ini terpenuhi, nilai I dan j sebagai iterator akan ditambah. Jika nilai m sama dengan j, maka p akanditambahsatu lalu nilai j direset sesuai dengan lps. Kemungkinan kedua adalah pattern berbeda dengan kalimat atau tidak menemui kecocokan, maka nilai j akan direset jika nilai j tidak nol, atau nilai I akan ditambah satu

Berikut adalah potongan kode utama pada Algoritma Knutt-Morris-Pratt.

```
def searchWithKMP(Pattern, File):

    #Instansiasi Variabel
    PatternLength = len(Pattern)
    FileLength = len(File)
    cntPattern = 0
    cntFile = 0
    Ada = 0

    Prefix = [0]*PatternLength

    borderFunction(Pattern, Prefix)
    while ( cntFile < FileLength ):
        if (Pattern[cntPattern] == File[cntFile] ):
            cntFile += 1
            cntPattern += 1
        if ( cntPattern == PatternLength ):
            Ada += 1
            cntPattern = Prefix[cntPattern-1]
        elif cntFile < FileLength and Pattern[cntPattern]
 != File[cntFile]:
            if cntPattern != 0:
                cntPattern = Prefix[cntPattern-1]
            else:
                cntFile += 1
    return Ada
```

GAMBAR 5. ALGORITMA UTAMA KMP

Untuk mendukung optimasi pergeseran pada algoritma, pada saat program pertama berjalan dihitung dulu border functionnya. Border functionmenghitung nilai suffix yang palingbesar dn sama denganprefixnya. Algoritma untuk mencari border function adalah sebagai berikut.

```
def borderFunction(Pattern, Prefix):
    # Fungsi yang mengembalikan nilai border Function dar
i suatu Pattern
    # Masukan : String Pattern
    # Keluaran : List of Integer yang berisi
    PatternLength = len(Pattern)
    cntPrefix = 0
    Prefix[0]
    cntSuffix = 1
    while ( cntSuffix < PatternLength):
        if (Pattern[cntSuffix] == Pattern[cntPrefix]):
            cntPrefix +=1
            Prefix[cntSuffix] = cntPrefix
            cntSuffix += 1
        else:
            Prefix[cntSuffix] = 0
            cntSuffix += 1
```

GAMBAR 6. ALGORITMA PENDUKUNG KMP

B. *Algoritma Boyer-Moore*

Algoritma ini bekerja dengan memindai karakter-karakter dari pattern yang dimiliki dari ujung paling kanan teks hingga ujung paling kiri, dimulai dari potongan string yang paling kanan dari teks tersebut. Jika string tidak cocok atau string cocok seluruhnya (bukan partial match), algoritma ini akan menggunakan dua buah fungsi untuk menggeser pattern pengujian string matching: good-suffix shift (matching shift) dan bad-character shift (occurence shift).

Di awal algoritma, ada 2 variabel yang menjadi inputan yakni pattern dan file. Diinisiasi juga patternlength dan filelength sebagai representasi panjang patterndan panjang file. Ada juga variable ada sebagai jumlah dari kecocokan pattern pada kalimat dan badchar sebagai pendukung optimasi pergeseran.

File akan diiterasi perkarakter, dan akan menghasilkan 2 kemungkinan, jika pattern sama, iterasi akan dilanjutkan dan jika pattern sama persis dengan bagia file, nilai ada akan ditambah satu. Jika pattern tidak cocok, file akan digeser sesuai dengan nilai maksimal dari 1 atau badcharnya.

Berikut adalah potongan kode utama pada Algoritma Boyer-Moore.

```
def searchWithBM(Pattern, File):
    PatternLength = len(Pattern)
    FileLength = len(File)
    Ada = 0

    badChar = patternRecognizer(Pattern, PatternLength)

    itrFile = 0
    while(itrFile <= FileLength - PatternLength):
        itrPattern = PatternLength-1
        while itrPattern >= 0 and Pattern[itrPattern] ==
File[itrFile + itrPattern]:
            itrPattern -= 1
        if itrPattern < 0:
            Ada = Ada + 1
            if itrFile + PatternLength < FileLength:
                itrFile += (PatternLength -
 badChar[ord(File[itrFile + PatternLength])])
            else:
                itrFile += 1
        else:
            itrFile += max(1, itrPattern -
 badChar[ord(File[itrFile + itrPattern])])
    return Ada
```

GAMBAR 7. ALGORITMA UTAMA BOYER MOORE

Untuk mendukung optimasi pergeseran pada algoritma, pada saat program pertama berjalan dihitung dulubad characternya. Bad character mencari kemunculan terakhir dari suatu karakter pada pattern. Algoritma untuk mencari bad character adalah sebagai berikut.

```
def patternRecognizer(Pattern, PatternLength):
    badChar = [-1]*256

    for i in range(PatternLength):
        badChar[ord(Pattern[i])] = i;

    return badChar
```

GAMBAR 8. ALGORITMA PENDUKUNG BOYER MOORE

C. Algoritma Utama

Secara umum, aplikasi akan menerima inputan 2 buah text, lalu memisahkan kalimat perkalimat, lalu mencari kalimat tersebut di artikel yang lain pada blog yang lain. Sebelumnya, blog dipindahkan dulu menjadi format text(.txt). Masukan text disimpan dalam variable text dan pattern_file. Pattern file diparse dengan nltk dalam variable sentences menjadi list of sentences. Diinisialisasi juga variable counter-matched dan counter-total dan p sebagai penghitung.

Kemudian, kalimat2 tersebut diiterasi dan dicocokkan dengan text yang lain, apabila ditemukan kalimat yang sama nilai counter akan ditambah. Persentase akan dihitung dari nilai counter matched dibagi counter total. Algoritma utama dapat dilihat pada gambar 9.

```
import re
import nltk
import bm
import kmp

root_file=input("Enter The Name of root file ")
plagiarised_file=input("Enter The Name of plagiarised fil
e ")

Text = open(root_file,"r")
text = Text.read()

pattern_file = open(plagiarised_file,"r").read()
#sentences = re.split(r'[\.\?!\r\n]', pattern_file)
sentences = nltk.sent_tokenize(pattern_file)
counter_matched = 0
counter_total = 1

for pattern in sentences:
  pattern = pattern.strip()

  if len(pattern) > 0:
    counter_total +=1
    counter_matched += kmp.searchWithKMP(pattern, text)

print ("Match percentage = %s%%" % (counter_matched*100/c
ounter_total))

if(counter_matched*100/counter_total) >= 70 :
  print ("The input file appears to be plagiarised. %s%%
of its content matches with the file %s." % ((counter_mat
ched*100/counter_total), root_file))
```

Gambar 9. Algoritma Utama

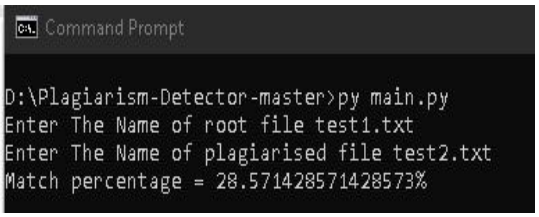
D. Pembahasan

Dalam melakukan pengujian, saya memilih beberapa artikel blog hasil pencarian beberapa kata kunci dalam laman pencarian google. Pada kasus uji pertama saya mencari dengan kata kunci "pancasila adalah" dan mengambil 2 hasil pada halaman pertama laman pencarian. Dalam hasil pencarian, penulis mengambil beberapa paragraph awal yang dicurigai sebagai hasi duplikasi.

Paragraf dipindahkan dalam file teks dan diperbaiki penulisannya agar tidak ada karakter Unicode yang tidak dapat dibaca oleh program. Lalu, program dijalankan dan akan meminta masukan sebagai berikut.

GAMBAR 10. ILUSTRASI PROGRAM DIAWAL

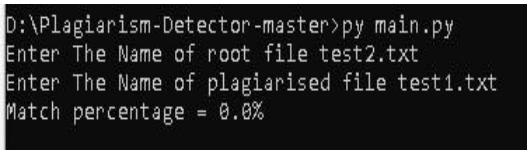
Kemudian, program akan menjalankan algoritma untuk mencari seberapa mirip potongan artikel tersebut, dan diperoleh hasil sebagai berikut.



GAMBAR 11. ILUSTRASI HASIL PROGRAM

Setelah dicek pada artikel aslinya, ternyata ada satu paragraph yang sama persis dengan paragraph yang lain pada artikel blog yang dipilih. Namun, susunan kalimat pada blog ini berbeda dengan blog yang lain sehingga cukup menyulitkan penulis untuk mendeteksi dimana letak plagiarism pada blog sebelumnya.

Untuk memvalidasi pengujian, penulis mencari 2 artikel yang dicari dengan 2 kata kunci berbeda pada kasus uji berikutnya. Hasil pengujian pada program menghasilkan 0% match, seperti pada gambar 12 berikut.



GAMBAR 12. ILUSTRASI HASIL PROGRAM

Hal ini menunjukkan bahwa pengujian berhasil,karena memang tidak ada plagiarism yang bisa dilakukan jika artikel memang membahas topic yang berbeda.

Tidak ada bug pada program yang menyebabkan artikel ditandai sebagai plagiarism namun dalam realitasnya tidak.

IV. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, kita dapat menyimpulkan bahwa algoritma Boyer-Moore dan KMP dapat digunakan untuk mendeteksi plagiarisme dalam artikel blog. Dengan memanfaatkan algoritma Boyer-Moore dan KMP dan beberapa tahap untuk menyaring informasi dari sebuah artikel, dapat ditentukan tingkat kecocokan konten dari dua buah artikel. Akan tetapi algoritma ini tentu bukan algoritma yang sempurna. Untuk pengembangan algoritma ini dapat menggunakan Artificial Intellegence atau kecerdasan buat agar pencocokan semakin akurat. Terutama dengan perkembangan Machine Learning sehingga algoritma ini dapat belajar dan meningkatkan keakuratan seiring keberjalanan waktu.

VIDEO LINK AT YOUTUBE

Hasil pengujian pada makalah ini dapat ditonton di https://youtu.be/-TCxrFYKJx4.

ACKNOWLEDGMENT

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah mendukung penulis sehingga makalah ini dapat selesai dengan baik. Terima kasih kepada Tuhan YME yang telah memberkati penulis sehingga dapat menyelesaikan makalah ini. Terima kasih kepada kedua orang tua penulis atas segala dukungan baik lahir maupun batin.

Terima kasih kepada ibu Dr. Masayu Leylia Khodra yang mengajar di kelas penulis sehingga penulis dapat memahami materi-materi yang diberikan dan dapat menulis makalah ini. Terima kasih juga kepada para penulis dan pencipta dari semua referensi yang digunakan oleh penulis sebagai sumber data dari makalah ini.

Terimakasih juga kepada teman-teman teknik informatika angkatan 2018 terutama yang sekelas di kelas K1 atas dukungannya dalam membuat makalah ini. Tanpa dukungan dari pihak-pihak diatas, saya ragu saya mampu meyelesaikan makalah ini dengan baik.

REFERENCES

[1] Munir, Rinaldi, Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Program Studi Teknik Informatika Institut Teknologi Bandung, 2009

[2] Charras Christian, Thierry Lecroq. “Booyer-Moore Algorithm”. diakses 22-04-2020 pukul 3.27 dari https://www-igm.univ-mlv.fr/~lecroq/string/node14.html

[3] Eppstein, Knuth-Morris-Pratt String Matching, Irvine, CA: University of California, 1996

[4] Naufal Zhafran Latif,Algoritma Pencocokan KontenArtikel Dengan Menerapkan Regular Ekspression untuk mendeteksi plagiarisme. Bandung, Program Studi Teknik Informatika Institut Teknlogi Bandung, 2019

[5] https://www.youtube.com/watch?v=mQZY7EmjbMA Waktu Akses : 2 Mei 2020 pukul 1.40 WIB.

[6] https://kbbi.web.id/plagiat. Waktu Akses : 2 Mei 2020 pukul 0.59 WIB.

[7] https://mahasiswa.ut.ac.id/images/stories/artikel/osmb/Permendiknas_Pencegahan_Plagiat_2010.pdf Waktu Akses : 2 Mei 2020 pukul 1.20 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, 2 Mei 2020
Ttd

Muhammad Ayyub Abdurrahman
13518076