Penerapan Algoritma String Matchig Untuk Mendeteksi Plagiarisme

Thareq Muhammad Yusuf Hasnul Aziz
Program Studi Teknik Informatika
Ekolah Teknik Elektro dan Informatika
Bandung, Indonesia
thareqmyha@gmail.com

Zaman sekarang akses informasi menjadi sangatlah mudah dan cepat. Banyak sekali paper maupun referensi-referensi yang dapat dilihat untuk digunakan kita bila mengerjakan sebuat tugas maupun sekedar melihat reerensi. Dengan adanya internet berbagai paper dari seluruh dunia ini bisa kita akses dengan keyword yang tepat menggunakan berbagai *search engine* yang tersedia gratis online. Namun hal ini juga menciptakan dilemanya sendiri. Banyak sekali paper bertebaran, namun juga banyak yang dipertanyakan orisinalitasnya. Dengan menggunakan algoritma string matching kita akan mencocokan beberapa keyword penting yang ada di paper tersebut agar dapat menentukan apakah sebuah paper itu plagiat dari sebuah paper.

Keywords—String Matching, Plagiarisme, Paper, Search Engine

I. PENDAHULUAN

Perkembangan internet marak dimana-mana. Hal tersebut menyebabkan semakin cepat dan banyak arus informasi yang ada di dunia ini. Hal tersebut memudahkan seseorang dalam melakukan penyalinan suatu karya, khususnya karya tulis. Dapat kita definisikan bahwa penjiplakan dalam Kamus Besar Bahasa Indonesia (KBBI) dapat diartikan menggambar atau menulis garis-garis gambaran atau tulisan yang telah tersedia (dengan menempelkan kertas kosong pada gambar atau tulisan yang akan ditiru), mencontoh atau meniru tulisan atau pekerjaan orang lain, mencuri karangan orang lain dan mengakui sebagai karangan sendiri, mengutip karangan orang lain tanpa seizin penulisnya.

Maraknya kasus penjiplakan oleh golongan intelektual menjadi suatu tragedi dalam dunia pendidikan Indonesia seperti kasus profesor termuda bidang hubungan internasional yang diberhentikan secara tidak hormat di tahun 2010. Akibat banyaknya informasi tersedia secara online maka kebiasaan copy–paste tanpa menyebutkan referensi menjadi mudah dilakukan. Sehingga karya ilmiah yang dibuat menjadi hasil plagiat dari karya ilmiah lain. Namun dikarenakan sebagian besar karya ilmiah belum dilindungi Undang-Undang Hak atas Kekayaan Intelektual (HaKI) maka plagiarisme digolongkan sebagai kejahatan akademik yang termasuk sebagai pelanggaran etika dan sulit untuk dipidanakan. Sebagai langkah awal untuk mencegah kasus serupa diperlukan cara mendeteksi kemungkinan terjadinya penjiplakan seperti di lingkungan perguruan tinggi yaitu utamanya pada hasil tugas akhir calon sarjana S1 maupun tesis calon sarjana S2 dan disertasi calon sarjana S3 yang rawan penjiplakan.

Oleh karena itu diperlukannya sebuah aplikasi ataupun algoritma yang dapat melihat tinggkat plagiarism suatu karya ilmiah dengan suatu karya ilmiah tertentu. Algoritma ini bisa memanfaatkan berbagai macam jenis algoritma *strong matching* seperti *Knutt-Morris-Payne, Booyer Moore,* maupun dengan *briteforce*.

II. DASAR TEORI

A. Definisi Plagiat

Plagiat adalah teknik penyalinan atau meniru karya orang lain yang diklaim menjadi hasil karya sendiri. Tidak adanya motivasi ataupun kemudahan dalam proses penyalinan dengan harapan tidak diketahui orang lain menjadi alasan utama terjadinya praktik plagiat. Beberapa jenis plagiat yang dikenal selama ini, yaitu:

a. *Word-for-word plagiarism* : menyalin setiap kata secara langsung tanpa diubah sedikitpun

b. *Plagiarism of the form of a source* : menyalin dan atau menulis ulang kode-kode program tanpa mengubah struktur dan jalannya program

c. *Plagiarism of authorship*: mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang sebenarnya

B. Plagiarisme di Source Code

Beberapa contoh praktik plagiat pada sebuah program, yaitu:

a. Leksikal: perubahan pada kode (source code) program, misalnya:

1. Komentar diubah (ditambah, dikurangi, atau diganti)

2. Format penulisan diubah

3. Nama variabel diubah

b) Struktural : perubahan struktur program

1. Perubahan urutan algoritma yang tidak mengubah jalannya program

2. Prosedur diubah menjadi fungsi atau sebaliknya

3. Pemanggilan prosedur diganti dengan isi prosedur itu sendiri.

C. Undang-Undang yang Mengatur Plagiarisme

Undang-undang yang mengatur hak cipta di Indonesia adalah UU nomor 19 tahun 2002 tentang hak cipta :

Pasal 2 ayat (1)
Pasal 3 ayat (1),(2)
Pasal 12
Pasal 15
Pasal 26 ayat (1)

Sanksi Plagiat (UU No. 20/2003)

Lulusan PT yang karya ilmiahnya digunakan untuk memperoleh gelar akademik, profesi, atau vokasi terbukti merupakan jiplakan:

- Pencabutan gelar (Pasal 25 ayat 2).
- dipidana penjara paling lama dua tahun dan/atau pidana denda paling banyak 200 juta rupiah (Pasal 70).

Hak Cipta

- Memberi perlindungan terhadap karya cipta di bidang seni, sastra, dan ilmu pengetahuan.
- Timbul secara otomatis setelah suatu ciptaan dilahirkan.
- Dianggap sebagai benda bergerak.
- Dapat beralih atau dialihkan.
- Ciptaan yang tidak diketahui penciptanya, hak ciptanya adalah pada Negara.

Sanksi Pidana Pelanggaran Hak Cipta

- Pidana penjara paling singkat satu bulan dan/atau denda paling sedikit 1 juta rupiah, atau pidana penjara paling lama 7 tahun dan/atau denda paling banyak 5 milyar rupiah.

Bentuk Ciptaan yang Dilindungi

- Buku, program komputer, pamflet, perwajahan (layout) terbitan, dan karya tulis lain;
- Ceramah, kuliah, pidato dan sejenis;
- Alat peraga pendidikan dan ilmu pengetahuan;
- Lagu atau musik dgn atau tanpa teks;
- Drama atau drama musikal, tari, koregografi, pewayangan dan pantomim;
- Seni rupa (seni lukis, gambar, ukir, kaligrafi, pahat, patung, kolase, dan seni terapan;
- Arsitektur;
- Peta;
- Seni batik;
- Fotografi;
- Sinematografi; dan
- Terjemahan, tafsir, saduran, bunga rampai, database, dan karya lain hasil pegalihwujudan.

Ruang Lingkup HAKI

- Hak cipta dan hak-hak berkaitan dgn hak cipta;
- Merek;
- Indikasi geografis;
- Rancangan industri;
- Paten;
- Desain layout dari rangkaian elektronik terpadu;
- Perlindungan thd rahasia dagang; dan
- Pengendalian praktek-praktek persaingan tdk sehat dalam perjanjian lisensi.

D. Cara Menghindari Plagiarisme

Cara Menghindari Plagiat

- Cantumkan dua tanda petik (“) pada pernyataan yang berasal langsung dari naskah asli dan cantumkan sumbernya dengan benar;
- Tulis ulang (paraphrase); dan
- Cantumkan sumbernya dengan benar. Cara Melakukan Paraphrase
- Baca ulang secara cermat, singkirkan naskah aslinya;
- Gunakan kata-kata dan ide anda sendiri dalam merangkai kalimat;
- Urutkan pemikiran anda dan utarakan ide tersebut; dan
- Periksa ulang parapharase anda, bandingkan dengan naskah asli, pastikan tidak menggunakan kata/istilah yang sama, dan informasi yang akan disampaikan tepat.

Yang Termasuk Kutipan

- Ringkasan (summary);
- Ungkapan dengan kata-kata sendiri (paraphrase);
- Kutipan (quotation); dan
- Statistik dan grafik. Yang Bukan Kutipan
- Pengetahuan umum (common sense);
- Kesimpulan anda sendiri;
- Fakta-fakta yang dapat ditemukan pada berbagai sumber

- Istilah standar (standard term)

E. Algoritma String Matching

Pencocokan pola (*pattern matching*) dalam ilmu komputer adalah pengecekan dan penempatan urutan data spesifik dari beberapa pola di antara data mentah atau urutan pola. Tidak seperti pengenalan pola (*pattern recognition*), kecocokan harus tepat berada di dalam pencocokan pola. Pencocokan pola adalah salah satu paradigma paling mendasar dan penting dalam beberapa bahasa pemrograman.

Pattern matching melibatkan penggunaan pencocokan string satu dimensi. Pola adalah struktur atau urutan pohon. Ada berbagai kelas bahasa pemrograman dan mesin yang menggunakan pencocokan pola.

Misalkan diberikan T (teks), yakni string yang panjangnya n karakter dan P (pattern), yakni string dengan panjang m karakter (asumsi m <<< n) yang akan dicari di dalam teks. Lalu cari lokasi pertama di dalam teks yang bersesuaian dengan pattern.

Contoh:

T = “I will tell you how to gain money in one day!”

P = “gain”

Terdapat beberapa algoritma yang dapat digunakan untuk menyelesaikan permasalahan ini, di antaranya adalah sebagai berikut.

1. Algoritma KMP (Knuth-Morris-Pratt)

Algoritma pencocokan KMP menggunakan properti degenerasi (pola yang memiliki sub-pola yang sama muncul lebih dari sekali dalam pola) dari pola dan meningkatkan kompleksitas kasus terburuk ke O(n). Algoritma ini mencari pattern di dalam teks dengan *left-to-right order*. Jika sebuah ketidakcocokan terdapat di antara teks T dan pattern P pada P[j], misalkan T[i] $\neq$ P[j] maka kita menggeser pattern sejauh prefix terbesar dari P[0..j-1] yang merupakan suffix dari P[1..j-1]. Hal ini dilakukan untuk menghindari perbandingan yang sia-sia.

Contoh algoritma KMP menggunakan Java:

```
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)) {
                        if (j==m-1)
                                return i-m+1;
                        i++;
                        j++;
                }
                else if (j>0)
                        j = fail[j-1];
                else
                        i++;
        }
        return -1; //no match
} // end of kmpMatch

public static int[] computeFail(String pattern) {
        int fail[] = new int[pattern.length()];
        fail[0] = 0;

        int m = pattern.length();
        int j = 0;
        int i = 1;

        while (i<m) {
                if (pattern.charAt(j) ==
              pattern.charAt(i)) { // j+1 chars match
                        fail[i] = j+1;
                        i++;
                        j++;
                } else if (j>0) {
                // j follows
                matching prefix
                        j = fail[j-1];
                } else {
                        fail[i] = 0;
                        i++;
                }
                return fail;
        } // end of computeFail()
}
```

2. Algoritma BM (Booyer Moore)

Algoritma Boyer-Moore dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Algoritma Boyer-Moore memiliki dua teknik pencocokan string, yaitu:

a. The looking-glass technique

Cari pattern P di dalam T dengan bergerak mundur melalui ujung P

b. The character-jump technique

-Saat sebuah ketidakcocokan terdapat pada T[i] == x

-Karakter pada pattern P[j] tidak sama dengan T[i]

Terdapat tiga kemungkinan kasus, di antaranya:

1. Jika P mengandung x di suatu tempat, maka cobalah geser P ke kanan untuk menyelaraskan kejadian terakhir x dalam P dengan T[i].
2. Jika P mengandung x di suatu tempat, namun sebuah pergeseran ke kanan untuk kejadian terakhir tidak

mungkin, maka geser P tepat sebanyak satu karakter ke T[i+1].

3. Jika kasus 1 dan 2 tidak berlaku, maka geser P untuk menyejajarkan P[0] dengan T[i+1].

Contoh algoritma Boyer-Moore menggunakan Java:

```
public static int bmMatch(String text,
String pattern) {
        int last[] = buildLast(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;
                        else {
                                i--;
                                j--;
                        }
                else {
                        int lo =
last[text.charAt(i)];
                         //last occur
                        i = i+m-Math.min(j, 1+lo);
                        j = m-1;
                }
        }       while (i<=n-1);
        return -1; //no match
}

public static int[] buildLast(String pattern)
/* Return array storing index of last occurrence of
        each ASCII char in pattern. */
{
        int last[] = new int[128]; // ASCII char set


        for (int i=0; i<128; i++)
                last[i] = -1; //initialize array
        for (int i=0; i<pattern.length(); i++)
                last[pattern.charAt(i)] = i;
        return last;
} // end of buildLast()
```

F. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. PEMBAHASAN

Untuk mengetahui tingkat persentase sebuah dokumen itu plagiat atau tidak kita dapat menggunakan rumus sebagai berikut ini

P = Jumlah Baris yg Cocok / Jumlah Baris Total * 100%

1. Menentukan jenis algoritma *String Matching*

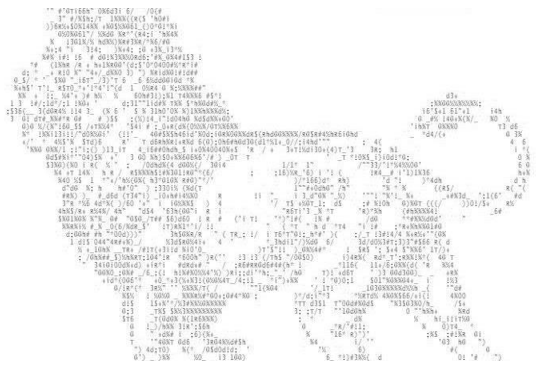
   Sebelum beralih ke proses yang ada, kita akan menggunakan sebuah algoritma untuk pattern matching, dalam konteks ni kita akan emnggunakan KMP. Mengapa KMP? Karena KMP lebih efisien untuk ukuran dokumen besar ketimbang Booyer Moore. Beriku algoritmanya

```
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)) {
                        if (j==m-1)
                                return i-m+1;
                        i++;
                        j++;
                }
                else if (j>0)
                        j = fail[j-1];
                else
                        i++;
        }
        return -1; //no match
} // end of kmpMatch

public static int[] computeFail(String pattern) {
        int fail[] = new int[pattern.length()];
        fail[0] = 0;


        int m = pattern.length();
        int j = 0;
        int i = 1;


        while (i<m) {
                if (pattern.charAt(j) ==
             pattern.charAt(i)) { // j+1 chars match
                        fail[i] = j+1;
                        i++;
                        j++;
```

```
                } else if (j>0) {
               // j follows
               matching prefix
                        j = fail[j-1];
                } else {
                        fail[i] = 0;
                        i++;
                }
                return fail;
        } // end of computeFail()
}
```

2. Mengubah dokumen menjadi array of string

   Langkah pertama adalah mengubah sebuah dokumen menjadi array of string yang tiap stringnya berisikan konten dokumen per baris tersebut. Ambil contoh sebuah gambar ascii dibawah ini :



Source http://www.materidosen.com/2016/10/pengertian-dan-fungsi-kode-ascii-lengkap.html

Fig 3.1 A Lion in ASCII

3. Memasukkan array of string dan mencocokannya ke dalam dokumen yang ingin menjadi pengujinya

   Lalu setelah mengubah menjadi array of string, maka array of string tersebut dimisalkan S, dimasukkan ke dalam algoritma yang sudah kita siapkan. Dalam hal ini kita menggunakan KMP,

   Berikut Pseudo Code Nya

```
public static int main(string[100] S, dokumen D) {
   //CEK BERAPA BARIS YG SAMA
   int barisSama = 0;
   foreach (String baris : S) {
     int result;
     result = kmpMatch(D,
     if (result != -1) {
       barisSama++;
     ]


}
```

4. Menghitung Persentase keplagiatan Dokumen

   Setelah mendapatkan jumlah baris yang cocok dengan dokumen D, maka dapat disimpulkan persentase kesimilaritasan dokumen ini dengan dokumen D

IV. KESIMPULAN DAN SARAN

Setelah dilakukan percobaan ternyata tidak secara akurat dapat menentukan tingkat keplagiatan suatu dokumen terhadap suatu hal. DIbutuhkannya sebuah cara mencocokkan suatu dokumen yang lebih baik lagi karena dengan metode seperti ini hanya akan mendapati dokumen yang secara eksak mengcopy sebuah dokumen atau *word to word copy*.

Untuk aplikasi jenis sama kedepannya diharapkan dapat menggunakan cosine similarities yang memanfaatkan sebuah vector untuk menentukan kesamaan sebuah dokumen.

V. UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan ucapan terima kasih kepada Allah SWT yang dengan izin dan rahmatnya mengizinkan penulis untuk membuat makalah ini dengan keadaan sehat wal'afiat.

Terima kasih pula dengan kedua orang tua penulis Bapak Hasnul Aziz dan Ibu Nurvita Trianasari dengan segala dorongan dari segi materil maupun moril yang mendorong penulis untuk terus giat dalam menuntut ilmu dan menjadi seorang yang bertanggung-jawab dan berguna untuk khalayak ramai kedepannya.

Penulis juga ucapkan terima kasih kepada teman-teman seperjuangan penulis sekalian yang tanpa kehebatan-kehebatan dan segala jenis inspirasi mereka bagi penulis tidak akan mendorong penulis menjadi seorang yang ingin berkompetisi secara sehat dengan teman-teman penulis.

Dan tidak dilupakan juga kepada Ibu Dr. Masayu Leylia Khodra ST.MT selaku dosen pengajar penulis dalam mata kuliah Strategi Algoritma ini yang selalu bersemangat dan menginspirasi dalam berbagi ilmu-ilmunya yang sangatlah bermanfaat untuk penulis dan juga memberikan pengalaman-pengalaman unik yang penulis rasakan selama pengajaran beliau ini.

REFERENCES

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

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/. akses: 13 Mei 2018.

[3] Noynaert, J.Evan, "Plagiarsim Detection Software", http://www.cs.uwec.edu/mics/CD%20image/papers/paper9 7.pdf, waktu akses: 13 Mei 2018.

[4] http://en.wikipedia.org/wiki/, waktu akses: 16 Mei 2008 pukul 14.00.

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.

Thareq Muhammad Yusuf Hasnul Aziz
13516004

Bandung, 13 Mei 2018