ISSN 2085-4552

Implementasi Algoritma *Hamming Distance* dan *Brute Force* dalam Mendeteksi Kemiripan *Source Code* Bahasa Pemrograman C

Andreas Budiman, Dennis Gunawan, Seng Hansun
Program Studi Teknik Informatika, Universitas Multimedia Nusantara, Tangerang, Indonesia
andreasbudiman93@gmail.com, dennis.gunawan@umn.ac.id, hansun@umn.ac.id

Diterima 30 September 2016
Disetujui 28 Oktober 2016

***Abstract*—Plagiarism is a behavior that causes violence of copyrights. Survey shows 55% of college presidents say that plagiarism in students' papers has increased over the past 10 years. Therefore, an application for detecting plagiarism is needed, especially for teachers. This plagiarism checker application is made by using Visual C# 2010. The plagiarism checker uses hamming distance algorithm for matching line code of the source code. This algorithm works by matching the same length string of the code programs. Thus, it needs brute force algorithm to filter the length of line code that will be matched with hamming distance. Another important thing for detecting plagiarism is the preprocessing, which is used to help the algorithm for detecting plagiarized source code. This paper shows that the application works good in detecting plagiarism, the hamming distance algorithm and brute force algorithm works better than levenstein distance algorithm for detecting structural type of plagiarism and this thesis also shows that the preprocessing could help the application to increase its percentage and its accuracy.**

***Index Terms*—Brute Force, Hamming Distance, Plagiarisme, Preprocessing.**

I. PENDAHULUAN

Plagiarisme menurut KBBI memiliki arti penjiplakan yang melanggar hak cipta. Menurut Sulianta [1] dalam bukunya yang berjudul "Seri Praktis Konten Internet", menjelaskan bahwa kegiatan plagiarisme marak terjadi di kalangan mahasiswa. Hal ini dipertegas dengan hasil survei yang dilakukan oleh Pew Research Center di tahun 2011 yang menunjukkan bahwa sekitar 55% rektor dari berbagai kampus di Amerika, menyatakan bahwa tindakan plagiarisme telah meningkat selama 10 tahun ini.

Penelitian berupa pemeriksaan plagiarisme pada sebuah *source code* berbahasa C pernah dilakukan sebelumnya oleh Andreas Arifianto dari Universitas Multimedia Nusantara pada tahun 2011 [2] yang berjudul "Rancang Bangun Aplikasi Pendeteksi Plagiarisme Kode Program Bahasa C Menggunakan algoritma Levenshtein Distance dan Brute Force". Dalam penelitian tersebut dilakukan perbandingan kinerja antara algoritma *levenshtein distance* dan algoritma *brute force* dalam mendeteksi plagiarisme. Penelitian tersebut menunjukkan bahwa algoritma *levenshtein* menghasilkan bobot persentase kecenderungan plagiarisme lebih tinggi pada saat membandingkan kode program yang mengalami perubahan yang bersifat non-struktural, sedangkan algoritma *brute force* menghasilkan bobot persentase kecenderungan plagiarisme lebih tinggi pada saat membandingkan dokumen kode program struktural.

Pada dasarnya algoritma *levenshtein distance* dan algoritma *hamming distance* memiliki persamaan, sebab keduanya termasuk algoritma *edit distance*. Menurut Lovrencic [3], algoritma *edit distance* merupakan algoritma yang digunakan untuk mencari perbedaan atau selisih antar dua objek. Objek tersebut bisa berupa biner, *string* dan lainnya. Perbedaan algoritma *levenshtein distance* dan algoritma *hamming distance* terletak pada proses penyelesaian masalahnya. Berbeda dengan algoritma *levenshtein,* algoritma *hamming distance* hanya dapat bekerja pada dua buah objek yang memiliki ukuran atau panjang yang sama. Menurut Abdeen [4], algoritma brute force adalah algoritma paling sederhana yang dapat digunakan pada pencarian pola. Sedangkan menurut Sinapova [5], beberapa algoritma brute force yang terkenal pada kasus *search and sort* adalah *sequential search, selection sort,* dan *bubble sort*. Penggunaan algoritma *hamming distance* dan *brute force* untuk mendeteksi tindakan plagiat memungkinkan terbentuknya aplikasi deteksi

kemiripan *source code* dengan konsep yang sederhana.

Adapun dalam penelitian ini, terdapat beberapa batasan masalah yang ditentukan, antara lain:

- *Source code* yang dikenali dalam bahasa C.
- Input berupa dua buah *file* berekstensi ".c".
- Aplikasi berbasis *desktop* dengan bahasa C#.

Dalam hal pemrograman, terdapat dua jenis plagiat yang umum dilakukan, yaitu sebagai berikut.

1. Leksikal

Teknik plagiat leksikal merupakan teknik plagiat dengan melakukan pengubahan kode program tanpa mengubah strukturnya.

2. Struktural

Teknik plagiat struktural merupakan teknik plagiat dengan mengubah struktur kode program tanpa mengubah fungsi program.

II. ALGORITMA HAMMING DISTANCE DAN BRUTE FORCE

A. Hamming Distance

Algoritma hamming distance merupakan salah satu dari algoritma *approximate string matching* yang ditemukan oleh Richar Hamming, pada tahun 1950. Algoritma hamming distance pertama kali digunakan untuk mendeteksi dan memperbaiki telekomunikasi sebagai estimasi *error* [6].

Algoritma hamming distance dapat digunakan untuk menghitung jumlah perbedaan dua buah biner yang mempunyai panjang yang sama. Menurut Murti [7], rumus yang digunakan pada algoritma ini dapat dilihat pada Rumus (1).

(1)

= nilai Hamming Distance

= jumlah variabel dengan nilai 1 pada objek ke-i, tapi bernilai 0 pada objek ke-k

= jumlah variabel dengan nilai 0 pada objek ke-i, tapi bernilai 1 pada objek ke-k

B. Brute Force

Algoritma brute force menurut Abdeen [4] adalah algoritma paling sederhana yang dapat digunakan pada pencarian pola. Meskipun algoritma brute force merupakan algoritma yang dapat diimplementasikan pada hampir semua permasalahan, namun algoritma ini sebenarnya bukanlah algoritma yang cerdas dan efisien, sebab tidak jarang dalam prosesnya algoritma ini memakan banyak sumber daya untuk menyelesaikan masalah.

Menurut Sinapova [5], salah satu algoritma yang termasuk ke dalam brute force adalah algoritma *sequential search*. Algoritma ini bekerja dengan cara membandingkan elemen-elemen dalam *list* dengan sebuah kata kunci sampai menemukan kata kunci yang dicari atau tidak ditemukan sama sekali.

C. Penerapan Hamming Distance dan Brute Force

Algoritma Hamming Distance digunakan untuk membantu dalam memeriksa dua buah *string* pada dua buah *source code*, sedangkan algoritma Brute Force digunakan untuk menyaring dua buah *string* yang akan dicocokkan oleh algoritma Hamming Distance. Kemudian, program akan memeriksa hasil pencocokkan dari algoritma Hamming Distance. Jika nilai Hamming Distance mencapai 0, maka dipastikan kedua buah *string* yang dicocokkan adalah identik. Jumlah baris yang identik tadi akan dikalkulasikan dan akan dibagi dengan jumlah total baris *source code* terpanjang dari input program untuk didapatkan persentase kemiripannya.

Source code 1	N	Brute force	N	Source Code 2	Hamming Distance	Hamming Value
Intfunc-function	16		16	Intfunc-function	Intfunc-function Intfunc-function	0
Intvar-int=10,var-int;	22		14	Intvar-int=10;		Null
While1<=var-int	15		11	intvar-int;		Null
Var-int=1;	10		15	While1<=var-int	While1<=var-int While1<=var-int	0
Whilevar-int<=10	16		10	Var-int=1	Var-int=1 Var-int=1	0
Printf, var-int%2?:;	19		16	Whilevar-int<=10	Whilevar-int<=10 Whilevar-int<=10	0
Var-int++;	10		19	Printf,var-int%2?:;	Printf,var-int%2?:; Printf,var-int%2?:;	0
Var-int--;	10		10	Var-int--;	Var-int++; Var-int--;	2
					Var-int--; Var-int--;	0
Printf;	7		10	Var-int++;	Var-int++; Var-int++;	0
Return0;	8		7	Printf;	Printf; Printf;	0
-	-		8	Return0;	Return0; Return0;	0

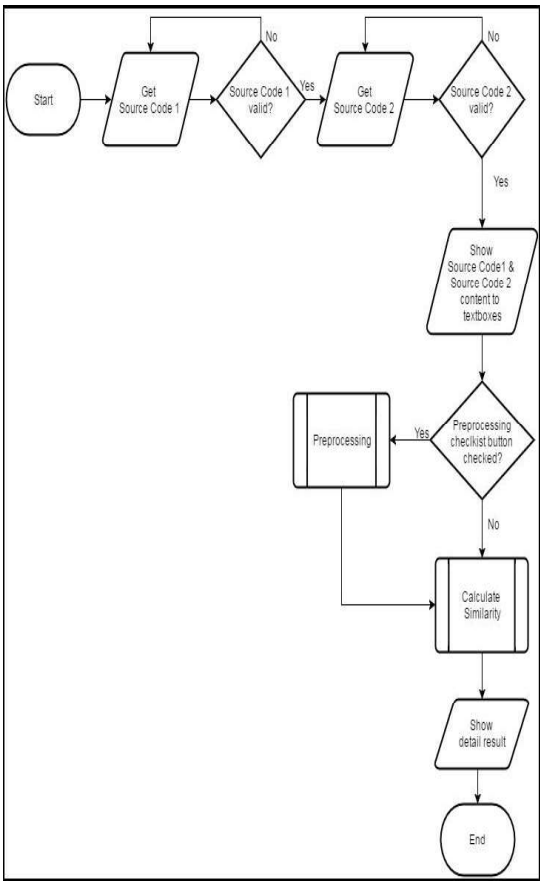
Gambar 1. Proses kerja algoritma Hamming Distance dan algoritma Brute Force dalam mendeteksi plagiarisme dua buah *source code*

Terlihat pada Gambar 1, kode baris kedua dengan panjang karakter 22, tidak menemui pasangannya. Sehingga baris ini tidak dapat dihitung menggunakan algoritma Hamming Distance. Berbeda dengan kode baris ketujuh, dimana pada baris ini meskipun memiliki panjang

baris yang sama, namun isi dari baris keduanya berbeda. Hal ini ditunjukkan pada kolom algoritma Hamming Distance, dimana perbedaan keduanya terpaut pada simbol "++" dan "--". Dengan hasil yang didapat di atas, program hanya perlu menghitung jumlah nilai nol yang didapat pada kolom "Hamming Value" dan membaginya dengan jumlah baris kode terpanjang. Contohnya saja, pada gambar di atas terdapat sembilan baris yang memiliki nilai Hamming Value-nya nol, dan jumlah baris kode terpanjang ada pada *source code* 2 dengan total baris sebesar 11. Maka persentase kemiripan kedua buah *source code* di atas adalah (9/11)*100% = 81.82%.

III. Diagram Alur Aplikasi

Hal pertama yang dilakukan program adalah menerima input berupa dua buah *source code* dan memvalidasinya. Setelah kedua *source code* dinyatakan valid, maka isi dari kedua *source code* akan ditampilkan. Selanjutnya, jika pengguna ingin menggunakan fitur *preprocessing*, maka isi dua buah *source code* akan di-*preprocessing* sebelum akhirnya dihitung tingkat kemiripannya pada proses *calculate similarity*. Hasil yang ditunjukkan berupa persentase kemiripannya.



Gambar 2. *Flowchart* program deteksi kemiripan menggunakan algoritma Hamming Distance dan algoritma Brute Force

IV. Uji Coba

Pengujian aplikasi dilaksanakan dengan melakukan perbandingan hasil perhitungan terhadap algoritma Levenshtein Distance dalam penelitian yang pernah dilakukan oleh Arifianto pada tahun 2011 [2] dengan objek yang sama. Adapun data uji yang digunakan berjumlah 10 pasang *source code*, dimana lima pasang *source code* dinyatakan sebagai plagiat leksikal (*dataset* 1), sedangkan lima pasang *source code* lainnya dinyatakan sebagai plagiat struktural (*dataset* 2). Berikut tabel hasil perbandingan kedua buah *dataset*.

TABEL I
Hasil Perbandingan Dataset 1 dengan Preprocessing

Dokumen yang dibandingkan	Persentase Kemiripan Menggunakan Algoritma Levenshtein Distance oleh Arifianto	Persentase Kemiripan Menggunakan Algoritma Hamming Distance
1_A1.c – 1_A2.c	97.06%	81.82%
1_B1.c – 1_B2.c	100%	100%
1_C1.c – 1_C2.c	94.69%	77.78%
1_D1.c – 1_D2.c	100%	100%
1_E1.c – 1_E2.c	100%	100%

TABEL II
Hasil Perbandingan Dataset 2 dengan Preprocessing

Dokumen yang dibandingkan	Persentase Kemiripan Menggunakan Levenshtein Distance oleh Arifianto	Persentase Kemiripan Menggunakan Hamming Distance
2_A1.c – 2_A2.c	93.67%	100%
2_B1.c – 2_B2.c	88.52%	95.56%
2_C1.c 2_C2.c	47.84%	92.86%
2_D1.c – 2_D2.c	45.11%	91.67%
2_E1.c – 2_E2.c	100%	100%

V. SIMPULAN

Berdasarkan perhitungan yang telah dilakukan, dapat disimpulkan bahwa algoritma Hamming Distance dan algoritma Brute Force berhasil diimplementasikan dalam mendeteksi kemiripan pada *source code* pemrograman berbahasa C. Algoritma Hamming Distance tidak dapat berjalan sendiri sebab algoritma ini memerlukan dukungan dari algoritma Brute Force dalam menyeleksi objek-objek yang akan diukur nilai Hamming Distance-nya. Dari hasil perbandingan terhadap algoritma Levenshtein Distance, algoritma Hamming Distance terbukti menghasilkan persentase rata-rata 20.99% lebih tinggi dalam deteksi plagiarisme struktural, namun menghasilkan persentase rata-rata 6.43% lebih kecil pada deteksi plagiarisme leksikal dibanding algoritma Levenshtein Distance. Tingginya persentase hasil perhitungan tidak lepas dari bantuan teknik *preprocessing* yang mampu menambah akurasi pendeteksian.

DAFTAR PUSTAKA

[1] Sulianta, F. 2007. Seri Praktis Konten Internet. Jakarta: PT Alex Media Komputindo.

[2] Arifianto, A, "Rancang Bangun Aplikasi Pendeteksi Plagiarisme Kode Program Bahasa C Menggunakan algoritma Levenshtein Distance dan Brute Force," S.Kom thesis, Fakultas ICT, Universitas Multimedia Nusantara, Tangerang, Indonesia, 2011.

[3] Lovrencic, A. 2009. Detecting Computer Code Plagiarism in Higher Education (p. 4). Varaždin, Croatia: http://www.researchgate.net.

[4] Abdeen, R. (2011). An Algorithm for String Based on Brute Force Algorithm. Internasional Journal of Computer Science and Network Security Vol 11 : 7.

[5] Sinapova, L. 2014. Dipetik 01 17, 2016, dari Simpson College Department of Computer Science: http://faculty.simpson.edu/lydia. sinapova/www/cmsc250/LN250_Leviti n/L05-BruteForce.htm.

[6] Primadani, Y. 2014. Simulasi Algoritma Levenshtein Distance Untuk Fitur Autocomplete Pada Aplikasi Katalog Perpustakaan. Dalam http:// repository.usu.ac.id/handle/123456789/41005, [diakses 20 Oktober 2015].

[7] Murti, D.H, dkk. 2005. Clustering Data Non-Numerik Dengan Pendekatan Algoritma K-Means Dan Hamming Distance Studi Kasus Biro Jodoh. Surabaya: Institut Teknologi Sepuluh November.