| *Journal Computer Science and Information Technology(JCoInT)*
Program Studi Teknologi Informasi, Fakultas Sains & Teknologi, Universitas Labuhanbatu
Volume: 3 Issue 2, 05,2022, Page: 54-63
eISSN: 2774-7751
Journal Homepage : http://jurnal.ulb.ac.id/index.php/JCoInT/index

Implementasi Algoritma Levenshtein Distance Dalam Mendeteksi Plagiarisme

Robiatul Adawiyah[1], Nidia Enjelita Saragih[2]

Email: robiatulbintisyarifuddin@gmail.com1[1], nidia.1924@gmail.com[2]

[1,2] Fakultas Teknik dan Informatika, Universitas Potensi Utama
Correspondence: E-mail: aliakbarritonga@gmail.com

A B S T R A C T S

Perkembangan teknologi membuat kehidupan manusia menjadi lebih mudah. Namun, perkembangan teknologi di hampir semua bidang kehidupan manusia, membawa dampak negatif. Salah satu dampak negatifnya adalah plagiarisme. Banyak sekali kasus plagiarisme yang dilakukan oleh kaum intelektual. Sehingga menjadi tragedi dalam lingkungan pendidikan Indonesia.Salah satu kegiatan plagiarisme yang sering terjadi di perguruan tinggi adalah makalah mahasiswa. Ini akan menjadi kebiasaan jika dibiarkan begitu saja, dan terus berlanjut dan pada gilirannya akan merusak dan mencoreng lingkungan akademik. Oleh karena itu, masalah ini harus mendapat penanganan serius dalam mencegah plagiarisme menjadi kebiasaan umum.Algoritma Levenshtein distance merupakan algoritma yang cocok untuk diterapkan dalam mendeteksi plagiarisme karena memberikan nilai kemiripan dari dua string yang sejenis. Penerapan algoritma ini pada suatu sistem, dapat membantu dosen untuk menentukan aktivitas plagiarisme pada makalah mahasiswa. Sistem yang dikembangkan menggunakan Visual Basic .Net.

A R T I C L E I N F O

Article History:
Received
Revised
Accepted
Available online

Keywords:
Plagiarisme,
Algoritma Levenshtein Distance,
Nilai kesamaan

I. PENDAHULUAN

Perkembangan teknologi membuat kehidupan manusia menjadi lebih mudah. Teknologi memudahkan orang dalam jarak jauh untuk saling berkomunikasi. Hal ini memungkinkan kita untuk mengakses informasi dari berbagai sumber, bahkan dari tempat dan waktu yang berbeda. Sehingga membuat pertumbuhan pengetahuan menjadi lebih cepat.

Namun, perkembangan teknologi di hampir semua bidang kehidupan manusia, membawa dampak negatif. Salah satu dampak negatifnya adalah plagiarisme. Banyak sekali kasus plagiarisme yang dilakukan oleh kaum

intelektual. Sehingga menjadi tragedi dalam lingkungan pendidikan Indonesia. Banyaknya informasi yang tersedia di internet membuat orang terbiasa melakukan copy paste tanpa menyebut referensi. Jadi, sebenarnya banyak sekali karya ilmiah yang berasal dari karya tulis lain sebagai produk plagiat.

Plagiarisme menurut kamus besar bahasa Indonesia (KBBI) adalah suatu perbuatan curang yang melanggar hukum hak cipta, yaitu hak penemu yang dilindungi undang-undang. Menurut Febrina et al, 2016, Plagiarisme adalah tindakan mengambil karya orang lain secara keseluruhan atau sebagian yang dilakukan baik secara sengaja maupun tidak sengaja. Plagiarisme adalah mengambil ciptaan orang lain (seperti pendapat atau ide) dan mengakuinya sebagai penemuan atau idenya sendiri. Beberapa contoh kegiatan plagiarisme adalah menerbitkan karya ilmiah orang lain dan kemudian mengakuinya sebagai miliknya.

Salah satu kegiatan plagiarisme yang sering terjadi di perguruan tinggi adalah karya tulis mahasiswa. Umumnya sebagian besar karya tulis mahasiswa merupakan hasil copy paste dari mahasiswa lain. Jika tidak dicegah, perilaku ini akan terus berlanjut dan pada gilirannya akan merusak dan mencoreng lingkungan akademik.

Oleh karena itu, persoalan ini harus mendapat penanganan serius. Yakni untuk mencegah plagiarisme menjadi perilaku yang dianggap biasa. Sebagai langkah awal untuk mengurangi angka plagiarisme adalah dengan mengembangkan sistem atau aplikasi yang dapat mendeteksi plagiarisme. Harapannya, sistem ini dapat diterapkan oleh para guru, dalam hal ini dosen, untuk mendeteksi makalah plagiarisme dari makalah mahasiswa.

Ada beberapa algoritma yang dapat diterapkan dalam mendeteksi kesamaan dokumen. Salah satunya adalah algoritma jarak Levenshtein. Algoritma jarak Levenshtein memberi kita rumus untuk menghitung nilai kemiripan dari dokumen yang dibandingkan. Algoritma ini dapat membantu kita dalam menentukan makalah mana yang merupakan produk plagiarisme dan mana yang bukan.

II. METODE PENELITIAN

2.1 Analisa Algoritma

Algoritma Levenshtein Distance yang diimplementasikan pada sistem ini harus dianalisis terlebih dahulu oleh penulis untuk memahami setiap langkah pada algoritma ini. Langkah ini dijalankan dengan mengimplementasikan algoritma levenshtein Distance pada dua string yang sama. Hasil perhitungan disimpan dan digunakan sebagai acuan saat pengujian sistem dilakukan.

Melalui proses analisis, antarmuka sistem harus dirancang. Sistem antarmuka yang baik adalah antarmuka sistem yang ramah pengguna. Implementasi algoritma ke dalam bahasa pemrograman sedang dijalankan pada langkah ini. Setiap langkah pada algoritma ini diubah menjadi bahasa pemrograman. Sedangkan Data yang dibutuhkan untuk menjalankan sistem ini adalah dua dokumen yang serupa. Dokumen yang digunakan dalam penelitian ini adalah makalah mahasiswa.

Pengujian sistem dilakukan dengan menginputkan dua buah string ke dalam sistem yang telah dibangun. Langkah ini untuk memastikan bahwa hasil yang didapat dari sistem sama dengan hasil pada perhitungan manual.

I S S N : 2774-7751

2.2 Metode Levenshtein Distance

Levenshtein Distance ditemukan oleh Vladimir Levenshtein pada tahun 1965. Perhitungan edit distance didapat dari matriks yang digunakan untuk menghitung selisih antara dua string. Perhitungan dua string ini ditentukan oleh angka minimum sebagai hasil dari operasi yang dijalankan dalam mengubah A ke B . Ada 3 macam operasi utama yang harus dijalankan dalam algoritma yaitu :

a. Operasi Penyisipan Karakter (Insertion)

Operasi penyisipan karakter berarti menyisipkan karakter ke dalam suatu string. Contohnya string 'disrit' menjadi string 'diskrit', dilakukan penyisipan karakter 'k' di tengah string. Penyisipan karakter tidak hanya dilakukan di tengah string, namun bisa disisipkan diawal maupun disisipkan di akhir string. Ilustrasi:

String 1	d	i	s	k	r	i	t
String 2	d	i	s	-	r	i	t
Insertion				k			

b. Operasi Penghapusan Karakter (Deletion)

Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu string. Contohnya string 'komputers' karakter terakhir dihilangkan sehingga menjadi string 'komputer'. Pada operasi ini dilakukan penghapusan karakter 'n'. Ilustrasi:

String 1	k	o	m	p	u	t	e	r	s
String 2	k	o	m	p	u	t	e	r	s
Deletion									

c. Operasi Penukaran Karakter (Subtitution)

Operasi penukaran karakter merupakan operasi menukar sebuah karakter dengan karakter lain. Contohnya penulis menuliskan string 'gimpunan' menjadi 'himpunan'. Dalam kasus ini karakter 'g' yang terdapat pada awal string, diganti dengan huruf 'h'. Ilustrasi:

String 1	g	i	m	p	u	n	a	n
String 2	h	i	m	p	u	n	a	n
Substitution	h							

2.3. Langkah-Langkah Algoritma Levenshtein Distance

Algoritma Levenshtein distance berjalan mulai dari pojok kiri atas sebuah array dua dimensi (matriks) yang telah diisi sejumlah karakter string awal dan string target. Entri-entri pada matriks tersebut merepresentasikan nilai terkecil dari transformasi string awal menjadi string target.

Entri yang terdapat pada ujung kanan bawah matriks adalah nilai distance yang menggambarkan jumlah perbedaan dua string. Berikut ini adalah langkah-langkah algoritma Levenshtein distance dalam mendapatkan nilai distance: Misalkan S = String Awal, dan T = String Target

Langkah 1: Inisialisasi

a) Hitung panjang S dan T, misalkan m dan n

b) Buat matriks berukuran 0...m baris dan 0...n kolom

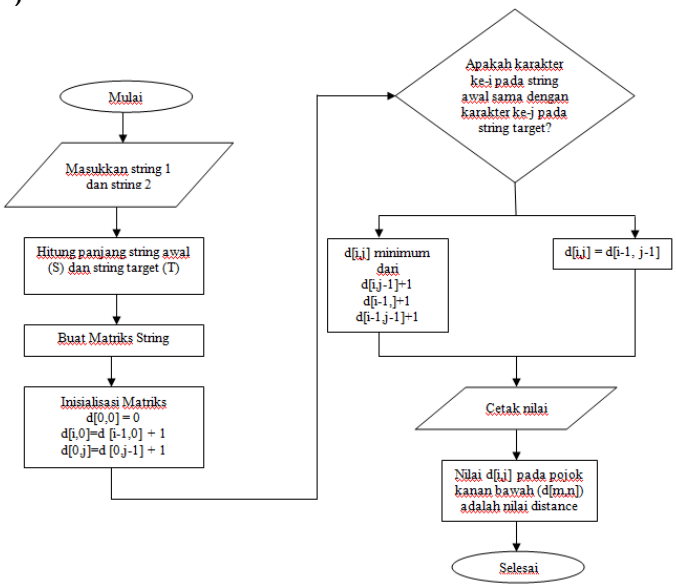
 ISSN : 2774-7751

c) Inisialisasi baris pertama dengan 0...n
d) Inisialisasi kolom pertama dengan 0...m

Langkah 2: Proses
a) Periksa S[i] untuk 1 < i < n
b) Periksa T[j] untuk 1 < j < m
c) Jika S[i] = T[j], maka entrinya adalah nilai yang terletak pada tepat didiagonal atas sebelah kiri, yaitu d[i,j] = d[i-1,j-1]
d) Jika S[i] ≠ T[j], maka entrinya adalah d[i,j] minimum dari:
Nilai yang terletak tepat diatasnya, ditambah satu, yaitu d[i,j-1]+1
Nilai yang terletak tepat dikirinya, ditambah satu, yaitu d[i-1,j]+1
terletak pada tepat didiagonal atas sebelah kirinya, ditambah satu, yaitu d[i-1,j-1]+1

Langkah 3: Hasil entri matriks pada baris ke-i dan kolom ke j, yaitu d[i,j]. Langkah 2 diulang hingga entri d[m,n] ditemukan.

Gambar berikut memperlihatkan setiap tahapan dalam proses algoritma yang ditunjukkan dalam flowchart.



Gambar 2.1 *Flowchart Levenshtein Distance*

Berikut adalah dua string yang digunakan sebagai contoh perhitungan dari algoritma Levenshtein Distance:

String 1(S) RONALDINHO
String 2(T) ROLANDO

Kita lihat, kedua string memiliki 6 perubahan. Artinya, dalam memodifikasi string RONALDINHO menjadi ROLANDO kita membutuhkan 6 operasi, yaitu:
1. Lakukan operasi substitusi (substitution) karakter 'N' dengan 'L'
RONALDINHO ROLALDINHO
2. Lakukan operasi substitusi (substitution) karakter 'L' dengan 'N'
ROLALDINHO ROLANDINHO
3. Lakukan operasi substitusi (substitution) karakter 'I' dengan 'O'

 I S S N : 2774-7751

ROLANDINHO ROLANDONHO

4. Lakukan operasi hapus (deleting) pada karakter 'O'

ROLANDONHO ROLANDONH

5. Lakukan operasi hapus (deleting) pada karakter ' H '

ROLANDONH ROLANDON

6. Lakukan operasi hapus (deleting) pada karakter ' N'

ROLANDON ROLANDO

Selain itu, proses perhitungan nilai distance bisa dilakukan dengan menggunakan matriks sebagaimana yang dijelaskan berikut.

String 1(S) RONALDINHO
String 2(T) ROLANDO
m(panjang karakter S) = 10
n (panjang karakter T) = 7

Kemudian, dibentuk sebuah matriks berukuran 7x10 sebagai berikut:

		R	O	N	A	L	D	I	N	H	O
	0	1	2	3	4	5	6	7	8	9	10
R	1										
O	2										
L	3										
A	4										
N	5										
D	6										
O	7										

Gambar 2.2 Matriks awal perhitungan distance

Pada setiap elemen, dilakukan perbandingan kesamaan. Jika karakternya sama, maka diberikan nilai distance 0, jika tidak maka nilainya adalah nilai distance sebelumnya ditambahkan dengan 1.

s Misalnya pada matriks contoh di atas :

- d(0,1) dibandingkan dengan d(1,0), yaitu karakter 'R' pada string 1(S) dan karakter 'R' pada string 2 (T) . Karena keduanya sama, maka elemen d(1,1) bernilai 0.
- d(0,2) dibandingkan dengan d(2,0), yaitu karakter 'O' pada S dan karakter 'O' pada T. Karena keduanya sama, maka elemen d(2,2) bernilai 0.
- d(0,3) dibandingkan dengan d(3,0), yaitu karakter 'N' pada S dan karakter 'L' pada T. Karena keduanya berbeda, maka elemen d(3,3) bernilai 1.
- d(0,4) dibandingkan dengan d(4,0), yaitu karakter 'A' pada S dan karakter 'A' pada T. Karena keduanya sama, maka elemen d(4,4) bernilai 0.
- d(0,5) dibandingkan dengan d(5,0), yaitu karakter 'L' pada S dan karakter 'N' pada T. Karena keduanya berbeda, maka elemen d(5,5) bernilai 1 + 1 (dari nilai distance sebelumnya) = 2
- d(0,6) dibandingkan dengan d(6,0), yaitu karakter 'D' pada S dan karakter 'D' pada T. Karena keduanya sama, maka elemen d(6,6) bernilai 0.

I S S N : 2774-7751

• d(0,7) dibandingkan dengan d(7,0), yaitu karakter 'I' pada S dan karakter 'O' pada T. Karena keduanya berbeda, maka elemen d(7,7) bernilai 1 + 2 (dari nilai distance sebelumnya) = 3

		R	O	N	A	L	D	I	N	H	O
	0	1	2	3	4	5	6	7	8	9	10
R	1	0									
O	2		0								
L	3			1							
A	4				0						
N	5					2					
D	6						0				
O	7							3			

Gambar 2.3. Matriks perhitungan distance

Kemudian nilai dari setiap elemen lain, diambil dengan menyesuaikan urutan pada elemen yang sudah terisi. Sehingga matriks menjadi :

		R	O	N	A	L	D	I	N	H	O
	0	1	2	3	4	5	6	7	8	9	10
R	1	0	1	2	3	4	5	6	7	8	9
O	2	1	0								
L	3	2		1							
A	4	3			0						
N	5	4				2					
D	6	5					0				
O	7	6						3			

Gambar 2. 4. Matriks pengisian elemen dalam perhitungan distance

Langkah yang sama diulangi hingga seluruh elemen pada matriks terisi dan didapatkan nilai elemen sudut kanan bawah sebagai nilai akhir dari distance sebagaimana ditunjukkan pada matriks berikut :

		R	O	N	A	L	D	I	N	H	O
	0	1	2	3	4	5	6	7	8	9	10
R	1	0	1	2	3	4	5	6	7	8	9
O	2	1	0	1	2	3	4	5	6	7	8
L	3	2	1	1	2	3	4	5	6	7	8
A	4	3	2	2	1	2	3	4	5	6	7
N	5	4	3	3	2	2	3	4	5	6	7
D	6	5	4	4	3	3	2	3	4	5	6
O	7	6	5	5	4	4	3	3	4	5	6

Gambar 2. 5. Matriks akhir perhitungan distance

2.4. Bobot Similarity

Levenshtein distance melakukan perhitungan bobot similarity setelah mendapatkan nilai distance dari dua dokumen yang dibandingkan.

ISSN : 2774-7751

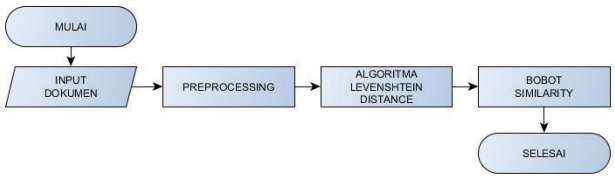
Kemudian menggunakan suatu persamaan dalam menentukan bobot similarity, yaitu

$$Sim = 1 - (\frac{Dis}{MaxLength}) \quad (1)$$

Dengan, Sim = Kesamaan/ nilai kesamaan
Dis = jarak Levenshtein
MaxLength = jumlah string terpanjang

Jika nilai similarity adalah 1 atau dalam persentase sebesar 100%, maka kedua string yang dibandingkan sebenarnya sama, tanpa modifikasi sama sekali. Namun jika nilai similarity adalah 0, maka kedua string yang dibandingkan tidak sama.

Proses perhitungan bobot similarity ini bisa dijelaskan dengan flowchart berikut :



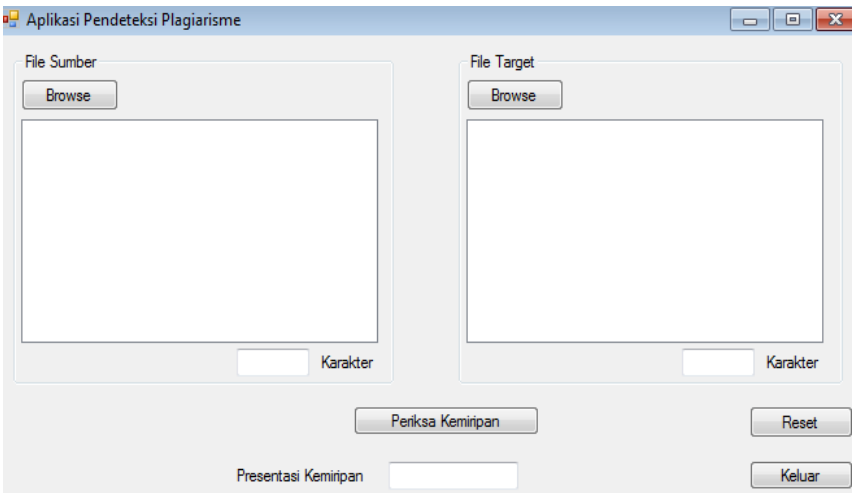
Gambar 2. 6. Flowchart perhitungan similarity

Dalam contoh penerapan kedua string di atas, yakni :
String 1(S) RONALDINHO
String 2(T) ROLANDO
Setelah didapatkan nilai distance, yaitu berada pada elemen kanan bawah sebesar 6. Dan karakter terpanjang dari kedua string yang diperbandingkan adalah 10 karakter (S). Maka jika diimplementasikan ke dalam persamaan di atas :
Dis = 6
MaxLength = 10
Sim = 1 - (6 / 10)
= 0,4
Atau dalam presentase = 40 %
Jadi, nilai similarity dari kedua string adalah sebesar 40 %.

III. HASIL DAN PEMBAHASAN

Implementasi algoritma jarak Levenshtein dalam mendeteksi plagiarisme pada penelitian ini dilakukan dengan menggunakan Visual Basic.Net. Berikut adalah beberapa tampilan sistem:

 I S S N : 2774-7751



Gambar 3.1 Tampilan system

Dalam interface di atas , terdapat dua buah groupbox. Yakni groupbox untuk 'File Sumber' dan groupbox untuk 'File target'. Fungsi groupbox pertama adalah untuk memilih dan menampilkan dokumen sumber. Dokumen sumber berarti yang dokumen pertama dari dua dokumen yang sejenis. Sedangkan groupbox kedua berfungsi untuk memilih dan memunculkan dokumen target. Dokumen target berarti dokumen kedua dari dua dokumen serupa.

Setiap groupbox berisi tombol “Browse”. Tombol ini digunakan untuk memilih file dokumen dari direktori komputer. Setelah dokumen dipilih, itu akan muncul di kotak multiline textbox. Untuk kalimat atau string pendek sederhana, kita dapat mengetikkan string pada textbox, tanpa menggunakan tombol Browse.

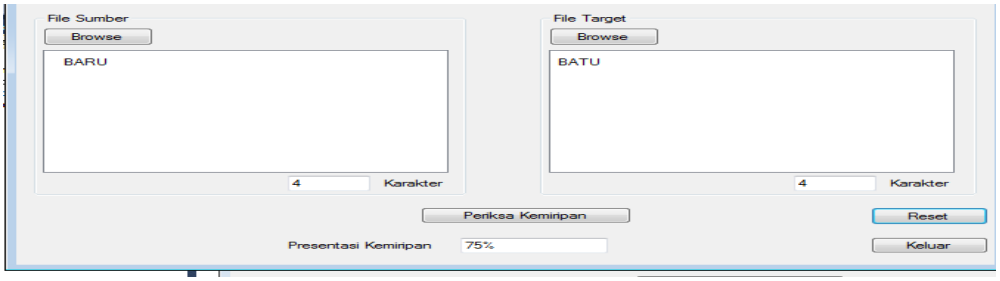
Textbox kedua, yang terletak di bawah textbox multiline pada masing-masing groupbox, digunakan untuk menunjukkan panjang karakter pada dokumen sumber.

Pengguna harus mengetikkan string pada kotak teks pertama atau menelusurinya menggunakan tombol “Browse”. Pengguna melakukan tindakan yang sama pada kotak grup kedua untuk memilih dokumen target.

Selanjutnya pengguna harus mengklik tombol “Periksa Kemiripan” untuk mendapatkan nilai kemiripan berdasarkan algoritma Levenshtein distance yang akan muncul pada textbox di bawah tombol ini.

Tombol “Reset” digunakan untuk mengosongkan semua textbox. Tombol “keluar” digunakan untuk menutup aplikasi.

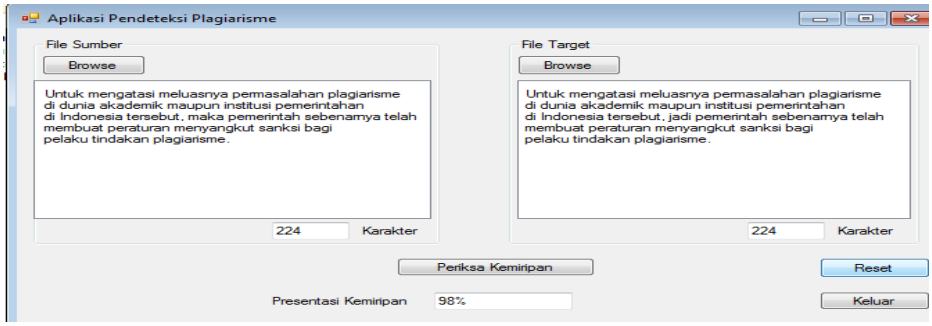
Berikut merupakan tampilan sistem jika diberikan dua buah string sederhana untuk diperiksa nilai kemiripannya menggunakan algoritma Levenshtein Distance.



Gambar 3.2. Antarmuka menggunakan string sederhana

Sedangkan untuk dokumen yang lebih panjang, sistem akan menampilkan tampilan berikut ini:

 I S S N : 2774-7751



Gambar. 3.3. Tampilan apllikasi menggunakan string panjang

IV. Kesimpulan

Pada penelitian ini, penulis dapat merangkum beberapa poin di bawah ini:

1. Algoritma jarak Levenshtein ini cocok untuk diterapkan dalam mendeteksi plagiarisme, dan memberikan hasil yang sempurna pada string sederhana yang pendek.
2. Algoritma jarak Levenshtein memberikan rumus untuk menentukan nilai kemiripan yang membantu pengguna untuk memastikan dokumen mana yang merupakan plagiarisme.
3. Algoritma Levenshtein distance harus diterapkan pada posisi yang tepat dari dua dokumen, karena memeriksa satu per satu karakter antara dua kalimat. Artinya, untuk dokumen yang panjang, dengan posisi yang tidak teratur, itu tidak berfungsi dengan baik.

Ucapan Terimakasih

Penulis mengucapkan terima kasih kepada Universitas Potensi Utama yang telah mendukung terlaksananya penelitian ini, juga kepada pihak LPPM Universitas yang memberi perbaikan penulisan laporan penelitian.

V. Daftar Pustaka

[1] Ni Made Muni Adriyani et all(2012).Implementasi Algoritma Levenshtein Distance dan Metode EmpirisUntuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia.Universitas Udayana

[2] Uli Fitrianti, Mutammimul Ula (2017).”Jurnal Sistem Informasi ISSN : 2598-599X”.Implementasi Algoritma Levenhstein Distance Dan Algoritma Knuth Morris Pratt pada Aplikasi Asmaul Husna Berbasis Android.

[3] Ida Bagus Ketut Surya Arnawa(2017).”jurnal Sistem dan Informatika”.Implementasi Algoritma Levenshtein Pada Sistem Pencarian Judul Skripsi/Tugas Akhir.Bali

[4] Pandawa, J.P. 2015. Perancangan Sistem Deteksi Kemiripan Dokumen Menggunakan Algoritma Winnowing. Skripsi Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Hidayatullah, Jakarta.

[5] Nafik, M.Z., Indriati dan A. Ridok. (2014). Sistem Penilaian Otomatis Jawaban Esai Menggunakan Algoritma Levenshtein Distance. Repositori Jurnal Mahasiswa PTIIK UB Vol 3 No. 4. Universitas Brawijaya, Malang.

 I S S N : 2774-7751

[6] Junedy, Richard. 2014. Perancangan Aplikasi Deteksi Kemiripan Isi Dokumen Teks dengan Menggunakan Metode Leveshtein Distance. Jurnal Pelita Informatika Budi Darma Vol. VII No.2, Jurusan Teknik Informatika, STMIK Budi Darma, Medan.
[7] Ariyani,N.H., Sutardi, and Ramadhan,R., 2016, Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari, Vol.2, No.1, pp. 279-286, ISSN : 2502-8928.
[8] Firdaus,A., Ernawati dan Vatresia,A., 2014, Aplikasi Pendeteksi Kemiripan Pada Dokumen Teks Menggunakan Algoritma Nazief & Adriani Dan Metode Cosine Similarity. Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu, Bengkulu, Vol.10, No.1, pp. 97-110.
[9] Pratama,B.P., Pamungkas,S.A., 2016, Analisis Kinerja Algoritma Levenshtein Distance Dalam Mendeteksi Kemiripan Dokumen Teks. Program Studi Matematika, Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah, Jakarta, Jilid 6, No.2, hal. 131-143.
[10] Putra,D.A., Sijaini,H., dan Pratiwi,H.S., 2015, Implementasi Algoritma Rabin-Karp untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Tanjungpura, Vol.1, No.1.
[11] A. Munif, R. J. Akbar, R. I. Tantra, dan R. Ilavi. “Rancang Bangun Sistem E-Learning Pemrograman pada Modul Deteksi Plagiarisme Kode Program dan Student Feedback System” Jurnal Ilmiah Teknologi Informasi, Vol. 15, No. 1, January, 2017.
[12] Setiadi, I., 2013. Damerau-Levenshtein Algorithm and Bayes Theorem for Spell Checker Optimization Damerau-Levenshtein Algorithm and Bayes Theorem for Spell Checker Optimization. , (November).
[13] Rachmatul Candra Ariani, 2013, Opini Mahasiswa Ilmu Sosial dan Ilmu Politik Terhadap Plagiarisme. Media Komunitas. Vol 2 No.1