ALGORITMA: Jurnal Ilmu Komputer dan Informatika
Volume: 01, Number : 01, November 2017 ISSN 2598-6341 (online)

DETEKSI TINGKAT KEMIRIPAN DATA SISWA PESERTA UJIAN BERBASIS KOMPUTER PADA BIMBINGAN BELAJAR EXPERT DENGAN MENGGUNAKAN METODE LEVENSHTEIN DISTANCE

[1]Guntur Syahputra
[2]Jijon Raphita Sagala

Email: [1]guntur_capt@yahoo.co.id, [2]sisagala@gmail.com

Abstrak

Dalam dunia pendidikan evaluasi merupakan hal yang wajib untuk mengukur tingkat keberhasilan proses pendidikan. Ujian yang dilaksanakan dengan berbasis komputer sering kali membuat siswa mengalami kegagalan karena kesalahan pengisian biodata. Penelitian ini bertujuan untuk mengukur tingkat kemiripan data identitas peserta didik sehingga kesalahan kecil yang dilakukan peserta dapat ditolerir oleh komputer pada saat pengisian data dan pada akhirnya dapat menurunkan tingkat kegagalan peserta ujian berbasis komputer. Penelitian ini sangat penting mengingat perubahan metode mengajar dan belajar peserta didik yang kini menitik beratkan pada penguasaan teknologi. Dengan menggunakan pendekatan metode Levenstain Distaince dapat mendeteksi kemiripan biodata peserta ujian sehingga bila dilihat ada kemiripan maka dapat memperbaiki penulisan data yang salah. Dengan mengimplemetasikan metode ini kedalam aplikasi koreksi soal sesuai dengan kebutuhan sistem di Bimbel Expert.
Kata Kunci : Levenstain Distance, Kemiripan Data, Ujian

Abstract

In the world of education evaluation is mandatory to measure the level of success educational process. Computer-based Test often make students fail because of incorrect filling of biodata. This research to measure the level of similarity student identity so small mistake made by participants can be tolerated by the computer when filling of biodata and ultimately can reduce the failure rate of computer-based Test participants. This research is very important considering the change of Study methods this time to using of technology. By using approach of Levenshtein Distance method can detect the resemblance of biodata of exam participants so that when seen there is similarity it can improve the writing of wrong data. By implementing this method into the problem correction application according to the needs of the system in Bimbel Expert.
Keywords : *Levenstain Distance, Similarity Of Data, Examination.*

1. PENDAHULUAN

Teknologi komputer pada era sekarang ini sudah menjadi kebutuhan pokok dihampir semua aspek kehidupan. Teknologi komputer tidak hanya cepat dalam proses namun diharapkan mampu menyajikan data yang tepat bebas dari kesalahan dan yang paling penting mampu memenuhi kebutuhan sistem.

Dalam bidang pendidikan penerapan teknologi komputer juga sudah dirasakan manfaatnya, salah satu contohnya dalam pengukuran tingkat kemampuan peserta didik melalui ujian berbasis komputer baik tingkat nasional maupun sektoral. Dalam pelaksanaan teknis dilapangan ujian berbasis komputer tidak hanya memberi dampak positif tetapi juga menimbulkan kecemasan bagi peserta karena tingkat kegagalan jauh lebih tinggi bila dibandingkan dengan ujian secara manual, hal ini disebabkan karena komputer membaca data secara baku.

Dari hasil pengamatan peneliti dalam beberapa ujian Try Out yang dilakukan BIMBEL Expert kegagalan peserta sering diakibatkan oleh kesalahan pengisian biodata peserta. Ujian yang dilakukan setiap peserta baik tingkat SD, SMP dan SMA yang biasanya menggunakan lebih dari satu lembar LJK disesuaikan dengan jumlah mata pelajaran. Apabila salah satu LJK salah dalam pengisian Biodata maka nilai secara kesuluruhan akan mengalami kegagalan. Untuk itu perlu adanya sebuah metode yang dapat menanggulangi kesalahan seperti ini.

Penelitian ini bertujuan untuk mengukur tingkat kemiripan data identitas peserta ujian dari beberapa LJK yang saling terkait sehingga kesalahan kecil yang dilakukan peserta dapat ditolerir oleh komputer dan pada akhirnya dapat menurunkan tingkat kegagalan peserta ujian berbasis Komputer.

2. TINJAUAN PUSTAKA

String metric adalah matriks berbasis karakter atau tekstual yang dapat menghasilkan nilai kesamaan atau ketidaksamaan dari dua buah teks *string* untuk proses perbandingan dan penyamaan. *String metric* biasanya digunakan dalam deteksi kecurangan, analisa fingerprint, deteksi plagiarisme, analisis

DNA dan RNA, data mining, dan lain-lain. Beberapa algoritma yang berdasarkan kepada string metric adalah *Smith-Waterman*, *Levenshtein Distance*, TF/IDF dan lain-lain. *String matching* merupakan metode yang digunakan *string metric* dalam menemukan hasil kesamaan atau ketidaksamaan dari teks yang diberikan [2].

Salah satu langkah penting dalam proses deteksi kemiripan adalah *preprocessing*, yaitu pemrosesan isi dokumen sebelum dibandingkan. Tujuannya adalah untuk menyeragamkan kata dan menghilangkan *noise* yang terdiri dari imbuhan, angka, simbol dan karakter yang tidak relevan, serta kata-kata yang tidak penting [1].

Tahap-tahap *preprocessing* adalah *Case Folding* atau *Converting*, *Tokenizing*, *Stopword Removal*, *Stemming*, dan *Sorting*. *Case folding* atau *converting* adalah proses mengubah semua huruf dalam dokumen menjadi huruf kecil, agar sistem mampu menyamakan tiap karakter dari dokumen satu dengan dokumen lainnya.

Tokenizing adalah tahap pengeliminasian symbol dan karakter yang tidak relevan dihilangkan, seperti tanda baca, angka, dan lain-lain. *Stopword Removal* adalah tahap pengeliminasian kata-kata yang tidak penting dari hasil *tokenizing*, seperti 'yang', 'di', 'ke', 'pada', dan lain-lain. Penghapusan kata-kata ini dimaksudkan untuk lebih mempermudah pembandingan dokumen. Stemming adalah proses untuk mencari *root* kata (kata dasar) dari kata berimbuhan hasil proses *stopword removal*. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu *representasi* yang sama. Tahap ini kebanyakan dipakai untuk teks berbahasa inggris dan lebih sulit diterapkan pada teks berbahasa Indonesia. *Sorting* adalah proses mengurutkan data baik secara menaik *(ascending*) atau secara menurun (*descending*). Data yang diurut bisa bertipe numerik maupun karakter [5].

Algoritma *Levenshtein* adalah sebuah matriks string yang digunakan untuk mengukur perbedaan atau jarak (*distance*) antara dua string Nilai *distance* antara dua *string* ini ditentukan oleh jumlah minimum dari operasi-operasi perubahan yang diperlukan untuk melakukan *transformasi* dari suatu *string* menjadi string lainnya. Operasi-operasi tersebut adalah penyisipan (*insertion*), penghapusan (deletion), atau penukaran (*subtitution*). *Levenshtein distance* merupakan salah satu algoritma yang dapat digunakan dalam mendeteksi kemiripan antara dua *string* yang berpotensi melakukan tindak *plagiarisme* [7]. Operasi-Operasi pada *Levenshtein Distance* Pada algoritma *Levenshtein distance*, terdapat tiga macam operasi yang dapat dilakukan yaitu [3]:

1. Operasi Penyisipan Karakter (*Insertion*)

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

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

2. Operasi Penghapusan Karakter (*Deletion*)

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

String 1	**m**	**A**	**t**	**E**	**m**	**a**	**t**	**i**	**k**	**a**
String 2	**m**	**A**	**t**	**E**	**m**	**a**	**t**	**i**	**k**	**a**	**n**
Deletion										**n**

3. 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**	**H**	**i**	**M**	**p**	**u**	**n**	**a**	**n**
String	**2**	**G**	**i**	**M**	**p**	**u**	**n**	**a**	**n**
Substitution		**H**							

Bobot Similarity

Levenshtein distance melakukan perhitungan bobot *similarity* setelah mendapatkan nilai distance dari dua dokumen yang dibandingkan. Kemudian menggunakan suatu persamaan dalam menentukan bobot *similarity*, yaitu [8]:

$$Bobot\ Similarity = \left(1 - \frac{d[m,n]}{Max(S,T)}\right) * 100\% \quad (1)$$

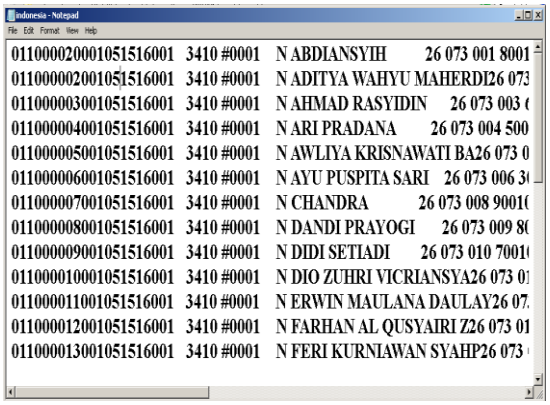
dengan d[m,n] adalah nilai distance, terletak pada baris ke m dan kolom ke n , S adalah panjang string awal, T adalah panjang string target, dan Max(S,T) adalah panjang string terbesar antara string awal dan string target.

Bobot similarity diasumsikan pada rentang 0 (nol) hingga 100 (seratus) persen, yang artinya nilai 100 adalah nilai maksimum yang menunjukan bahwa dua kata adalah sama identik. Pendekatan ini mampu digunakan untuk mengukur bobot similarity antar dua string berdasarkan susunan karakter.

3. METODOLOGI PENELITIAN

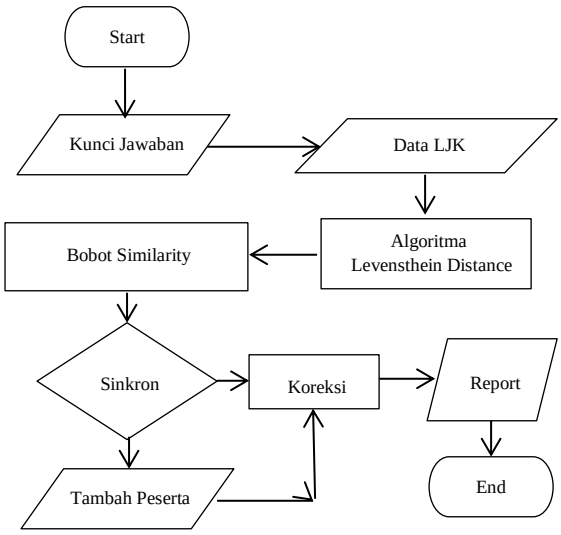
a. Data Penelitian

Data yang digunakan dalam penelitian ini merupakan hasil Scan dari Lembar Jawaban Komputer (LJK) yang dihasilkan oleh mesin OMR dalam bentuk softcopy yang berekstensi *.Dat. dapat dilihat dalam gambar 1.



Gambar 1. Output Mesin OMR
Bidang Studi Bahasa Indonesia

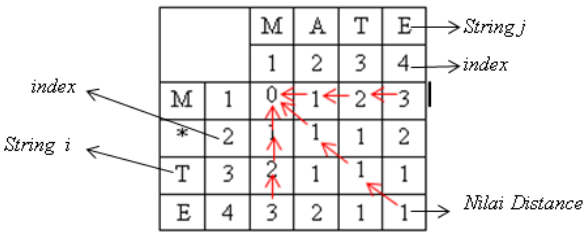
Setiap siswa mengerjakan beberapa lembar LJK sesuai dengan Bidang Studi yang diikuti dalam Try Out. Koreksi data dilakukan tidak hanya menentukan benar atau salah jawaban yang diberikan siswa tetapi dengan mensikronisasi beberapa LJK sesuai identitas yang ada. Kesalahan yang mungkin timbul adalah ketika peserta yang sama tetapi memiliki biodata berbeda pada setiap file mata pelajaran. Didalam penelitian ini LJK yang dijadikan sampel adalah siswa bimbingan Expert Jln. Nyak Makam Medan yang mengikuti ujian pada tahun 2017.



Gambar 2: Flowchart Proses Koreksi Jawaban
Dengan Levensthain Distance

b. Levensthain Distance

Proses Metode Levensthain Distance yang digunakan untuk melihat dapat dijelaskan dalam gambar no 3.



Gambar 3. Metode Levensthein Distance

Gambar diatas menunjukan proses levensthein distance dimana nilai Distance diperoleh dari pengujian String(i,j) bila string berbeda maka nilai ditambah 1 dan bila sama nilai akan mengikuti nilai String(i,j) sebelumnya yang ditunjukan oleh arah panah warna merah dan nilai paling akhir akan menunjukan nilai *Distance*.

Nilai *Distance* yang telah diperoleh akan dijadikan data menentukan tingkat kemiripan kedua String berdasarkan jumlah karakter terpanjang diantara kedua String Sesuai rumus bobot *Similarity*.

4. HASIL DAN PEMBAHASAN

Setelah didapat hasil Scan dari mesin OMR yang terdiri dari beberapa mata pelajaran maka dibandingkan biodata yang terdapat pada masing-masing LJK untuk menggabung hasil koreksi soal

Data yang diinput dibandingkan bobot *similarity* pada setiap field yang menjadi pembanding yaitu nomor peserta, Nama dan Tanggal Lahir dengan data yang sudah diinput sebelumnya.

Tabel 1. Perhitungan *Levensthein Distance*
nomor peserta

		0	1	1	0	0	0	0	1	3
		1	2	3	4	5	6	7	8	9
0	1	0	1	2	2	2	2	2	3	4
1	2	1	0	0	1	2	3	4	4	5
1	3	2	0	0	1	2	3	4	4	5
*	4	3	1	1	1	2	3	4	5	6
0	5	3	2	2	1	1	1	1	2	3
0	6	3	3	3	1	1	1	1	2	3
*	7	4	4	4	2	2	2	2	3	4
1	8	5	4	4	3	3	3	3	2	3
3	9	6	5	5	4	4	4	4	3	2

Tabel 2. Perhitungan *Levensthein Distance*
untuk Nama Siswa

		A	L	E	X	A
		1	2	3	4	5
A	1	0	1	2	3	3
L	2	1	0	1	2	3
*	3	2	1	1	2	3
X	4	3	2	2	1	2
A	5	3	3	3	2	1

Tabel 3. Perhitungan *Levensthein Distance*
untuk Tanggal Lahir

		1	9	*	8	9	6
		1	2	3	4	5	6
1	1	0	1	2	3	4	5
9	2	1	0	1	2	2	3
0	3	2	1	1	2	3	4
*	4	3	2	2	2	3	4
9	5	4	2	3	3	2	3
6	6	5	3	4	4	3	2

Dari tabel diatas didapat nilai *distance* dari setiap field yang menjadi pembanding yaitu: Nomor Peserta sebesar 2, Nama Siswa sebesar 1 dan Tanggal Lahir sebesar 2, sehingga Bobot *Similarity* sebagai berikut:

$$\text{Nomor Peserta} = \left(1 - \frac{d[m,n]}{Max(S,T)}\right) * 100\%$$
$$= \left(1 - \frac{2}{9}\right) * 100\% = 77{,}78\% \quad (2)$$

$$\text{Nama Siswa} = \left(1 - \frac{d[m,n]}{Max(S,T)}\right) * 100\%$$
$$= \left(1 - \frac{1}{5}\right) * 100\% = 80\% \quad (3)$$

$$\text{Tanggal Lahir} = \left(1 - \frac{d[m,n]}{Max(S,T)}\right) * 100\%$$
$$= \left(1 - \frac{2}{6}\right) * 100\% = 66{,}67\% \quad (4)$$

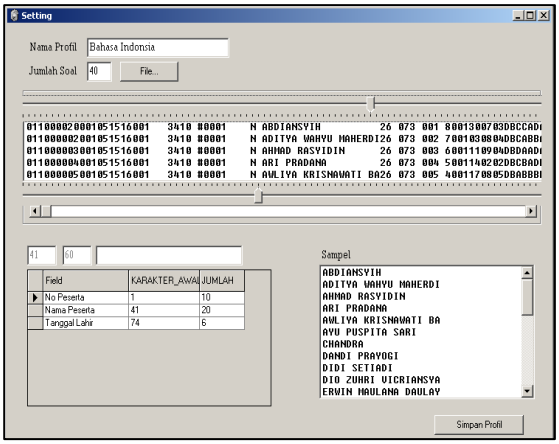
Untuk memberikan tingkat toleransi terhadap kesalahan yang terjadi baik peserta maupun mesin pembaca LJK maka diberikan nilai toleransi, dari hasil yang telah dilakukan beberapa kali pengujian maka diberikan nilai toleransi untuk setiap field pembanding yaitu:

Nilai Toleransi=(no peserta=23%) and
((nama peserta=10%) or
(tanggal lahir=2%)) (5)

Dari nilai toleransi yang telah dirumuskan maka Bobot *Similarity* dapat dibandingkan dan hasilnya belum memenuhi tingkat kemiripan sehingga harus data harus ditempatkan sebagai peserta yang baru.

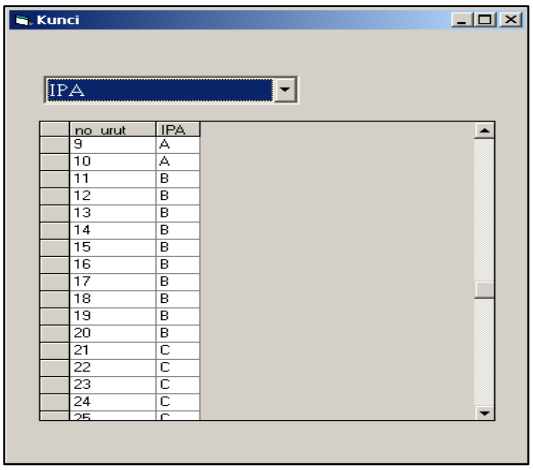
No Peserta= 100-77,78=22,2 memenuhi toleransi
And
Nama Siswa=100-80=20 tidak memenuhi toleransi
Or
Tanggal lahir=100-66,67=22,2 tidak memnuhi toleransi

Hasil pembacaan mesin OMR belum dapat menentukan field yang diperlukan untuk mengkoreksi jawaban peserta, untuk itu diperlukan proses pembentukan field dengan menetukan posisi field tersebut dan jumlah karakter yang dipakai yang disimpan sebagai sebuah profil. Tampilan tersebut dapat dilihat dalam gambar 4.



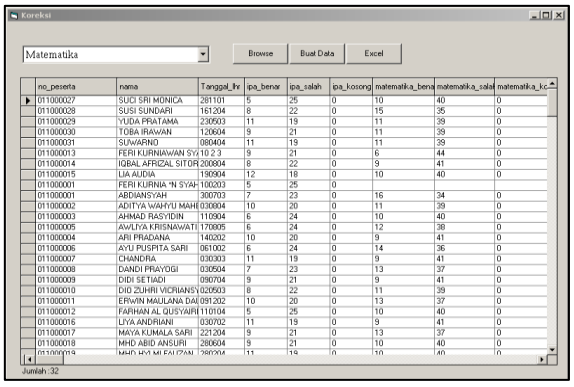
Gambar 4. Form Setting Profil

Setelah profil untuk LJK dibuat maka langkah selanjutnya adalah menginput kunci jawaban terhadap masing-masing profil yang sudah diatur dalam sistem. Tampilan Form input Kunci Jawaban dapat dilihat dalam Gambar 5.



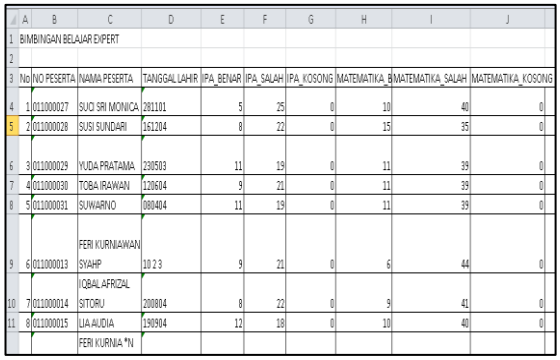
Gambar 5. Form Input Kunci Jawaban

Dan berikut ini Form untuk melakukan proses koreksi dengan memilih nama profil dan mencari file yang akan dikoreksi dengan menekan tombol browse seperti pada gambar 6.



Gambar 6. Form Koreksi

Hasil dari koreksi seluruh mata pelajaran yang sudah digabung dijadikan laporan dalam format berextention *.XLS seperti pada gambar nomor 10.



Gambar 7. Laporan Hasil Koreksi

Pengujian Sistem

Bedasarkan pengujian yang dilakukan terhadap hasil Tryout sebanyak 31 Peserta ada 14 orang yang nomor pesertanya tidak sesuai baik kesalahan dalam pengisian nomor peserta atau pembacaan mesin OMR yang kurang baik. Dalam sistem berhasil menemukan 13 peserta yang dianggap mirip sehingga nilai langsung digabung. Namun ada 1 orang yang tidak dapat deteksi kemiripan datanyanya karena kesalahan terjadi melebihi batas toleransi yang diberikan sistem. Seperti pada tabel dibawah ini.

Tabel 4. Hasil Singkronisasi Data.

NO	NO PESERTA	NAMA PESERTA	TANGGAL LAHIR
1	011000027	SUCI SRI MONICA	281101
2	011000028	SUSI SUNDARI	161204
3	011000029	YUDA PRATAMA	230503
4	011000030	TOBA IRAWAN	120604
5	011000031	SUWARNO	080404
6	011000013	FERI KURNIAWAN SYAHP	10 2 3
7	011000014	IQBAL AFRIZAL SITORU	200804
8	011000015	LIA AUDIA	190904
9	011000001	FERI KURNIA *N SYAHP	100203
10	011000001	ABDIANSYAH	300703
11	011000002	ADITYA WAHYU MAHERDI	030804
12	011000003	AHMAD RASYIDIN	110904
13	011000005	AWLIYA KRISNAWATI BA	170805
14	011000004	ARI PRADANA	140202
15	011000006	AYU PUSPITA SARI	061002
16	011000007	CHANDRA	030303
17	011000008	DANDI PRAYOGI	030504
18	011000009	DIDI SETIADI	090704
19	011000010	DIO ZUHRI VICRIANSYA	020503
20	011000011	ERWIN MAULANA DAULAY	091202
21	011000012	FARHAN AL QUSYAIRI Z	110104
22	011000016	LIYA ANDRIANI	030702
23	011000017	MAYA KUMALA SARI	221204
24	011000018	MHD ABID ANSURI	280604
25	011000019	MHD HYLMI FAUZAN	280204
26	011000020	AYUP	101004
27	011000021	MUHAMMAD DIKY ADITIA	250804
29	011000022	NIRINA PUTRI ANDINI	290704
30	011000023	SATRIA WHIBOWO	170904
31	011000024	SINDI AULIA	211104
32	011000025	SITI RISMAWATI	250904

5. KESIMPULAN

Berdasarkan serangkaian uji coba yang dilakukan dalam penelitian ini, dapat disimpulkanan bahwa:

1 Algoritma Levenshtein distance dapat diimplementasikan untuk mendeteksi kemiripan suatu dokumen teks dengan dokumen teks lain dengan keluaran berupa nilai *Distance*(Jarak Perbedaan) yang merupakan nilai balik dari nilai *Similarity* (Kemiripan)

2 Implementasi algoritma Levenshtein distance dapat diimplementasikan terhadap aplikasi koreksi soal sehingga dapat mensinkronisasi perbedaan biodata peserta ujian.

3 Levenshtein Distance dapat membentuk kombinasi lebih dari satu field pembanding sehingga lebih mendekati kebenaran

REFERENSI

[1] Februariyanti, Herny. 2013. Perancangan Pengindeks Kata pada Dokumen Teks Menggunakan Aplikasi Berbasis Web. Jurnal Teknologi Informasi DINAMIK Volume 18 Nomor 2, Program Studi Sistem Informasi, Universitas Stikubank.

[2] Hultberg, J. dan J.P. Helger. 2007. Seminar Course in Algorithms – Project Report.

[3] 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.

[4] Pratama B. P., S. A. Pamungkas. 2016. Analisis kinerja algoritma levenshtein distance dalam mendeteksi kemiripan dokumen teks. Jurnal "LOG!K@" , Jilid 6, No. 2

[5] Sjukani, Moh. 2013. Algoritma (Algoritma dan Struktur Data 1) dengan C, C++, dan Java. Jakarta: Mitra Wacana Media.

[7] Winarsono, D., D.O. Siahaan dan U. Yuhana. 2009. Sistem Penilaian Otomatis Kemiripan Kalimat Menggunakan Syntactic Semantic Similarity pada Sistem *ELearning*. Jurnal Ilmiah KURSOR Menuju Solusi Teknologi Informasi Volume 5 Nomor 2, Jurusan Teknik Informatika, ITS, Surabaya.

[8] Zhan Su, Byung-Ryul Ahn, Ki-yol Eom, Min-koo Kang, Jin-Pyung Kim, dan MoonKyun Kim. 2008. Plagiarism Detection Using the Levenshtein Distance and SmithWaterman Algorithm. The 3rd Intetnational Conference on Innovative Computing Information and Control, Department of Artificial Intelligence, University of Sungkyunkwan, Cheoncheon dong, Jangan-gu, Suwon, Korea. Diakses tanggal 12 mei 2017, dari http://ieeexplore.ieee.org.