JSRCS Journal of Students' Research in Computer Science

e-ISSN: 2722 – 290X
Vol. 5 No. 2 (November 2024), Page: 147 – 154
https://doi.org/10.31599/mzr20n84
Available Online at http://ejurnal.ubharajaya.ac.id/index.php/JSRCS

Algoritma *Levenshtein Distance* sebagai Solusi Efisiensi Pencarian pada Training Registration System

Iqbal Faris Akbar[1], Mugiarso[1,*], Khairunnisa Fadhilla Ramdhania[1], Rasim[1]

* Korespondensi: e-mail: mugiarso@dsn.ubharajaya.ac.id

[1] Informatika; Universitas Bhayangkara Jakarta Raya; Jl. Perjuangan No. 81, Marga Mulya, Bekasi Utara Bekasi Jawa Barat 17143 Telp/Fax: (021) 88955882; e-mail: iqbalfaris2002@gmail.com, mugiarso@dsn.ubharajaya.ac.id, khairunnisa.fadhilla@dsn.ubharajaya.ac.id, rasim@dsn.ubharajaya.ac.id

Submitted : **18 September 2024**
Revised : **3 Oktober 2024**
Accepted : **14 November 2024**
Published : **30 November 2024**

Abstract

The training registration process at Company XYZ faced several challenges, including the use of physical documents, slow file submission, non-standardized document numbering, and inefficient file storage. This research aimed to develop a web-based application called "Training Registration System" using the Waterfall software development method to address these issues, as well as implementing the Levenshtein Distance algorithm in the search feature to enhance training search efficiency. The results showed that the application successfully improved the efficiency and accuracy of the training registration and search process by eliminating the need for physical file submission, accelerating the workflow, and reducing the potential for errors. The implementation of the Levenshtein Distance algorithm also proved effective in improving the efficiency of training searches based on training names, even in cases of typos, such as the strings "Trainning" and "Training" which had a similarity score of 87.5% based on similarity weight calculations.

Keywords: *Black Box, Laravel, Levenshtein Distance, React*

Abstrak

Proses registrasi pelatihan di Perusahaan XYZ menghadapi kendala seperti penggunaan dokumen fisik, pengiriman berkas yang lambat, penomoran dokumen yang tidak terstandar, dan penyimpanan berkas. Penelitian bertujuan untuk mengembangkan aplikasi berbasis web "Training Registration System" menggunakan metode pengembangan perangkat lunak *Waterfall* untuk mengatasi masalah, serta menerapkann algoritma *Levenshtein Distance* pada fitur pencarian untuk meningkatkan efisiensi pencarian. Hasil penelitian menunjukkan bahwa aplikasi berhasil meningkatkan efisiensi dan akurasi dalam proses registrasi dan pencarian pelatihan, dengan menghilangkan proses pengiriman berkas fisik, mempercepat alur kerja, dan mengurangi potensi kesalahan. Penerapan algoritma *Levenshtein Distance* terbukti efektif dalam meningkatkan efisiensi pencarian pelatihan berdasarkan nama pelatihan, bahkan ketika terjadi kesalahan pengetikan, seperti pada kasus string "Trainning" dan "Training" yang memiliki tingkat kemiripan sebesar 87,5% berdasarkan perhitungan bobot similaritas.

Kata kunci: *Black Box, Laravel, Levenshtein Distance, React*

1. Pendahuluan

Transformasi digital telah menjadi pusat dari revolusi industri yang telah mempengaruhi hampir setiap aspek kegiatan manufaktur (Novita & Zahra, 2024). Perusahaan XYZ, yang beroperasi di sektor farmasi dengan lima lokasi berbeda, menghadapi masalah dalam proses registrasi dan administrasi pelatihan karyawan. Kendala-kendala seperti penggunaan dokumen fisik, ketergantungan pada kurir, perbedaan sistem penomoran, dan penyimpanan berkas fisik menyebabkan inefisiensi, keterlambatan, kesulitan pelacakan, dan risiko kehilangan data. Dalam menghadapi perubahan bisnis yang pesat dan persaingan yang semakin ketat, perusahaan-perusahaan modern di Indonesia dituntut untuk meningkatkan efisiensi operasional serta memberikan layanan yang lebih baik kepada pelanggan. PT. XYZ (A. Irawan et al., 2024), Teknologi informasi (TI) telah menjadi alat penting bagi organisasi dan perusahaan untuk meningkatkan efisiensi dan daya saing (Prihandono & Amir, 2024).

Untuk mengatasi masalah dikembangkan aplikasi berbasis web *Training Registration System* yang bertujuan untuk meningkatkan efisiensi dalam proses pengajuan dan manajemen pelatihan. *Training* atau pelatihan adalah suatu kegiatan dari perusahaan yang bermaksud untuk dapat memperbaiki dan mengembangkan sikap, tingkah laku, keterampilan dan pengetahuan dari para karyawan, sesuai dengan keinginan dari perusahaan yang bersangkutan (Nadeak, 2019)

Algoritma *Levenshtein Distance* digunakan untuk mencari alternatif kemiripan suatu judul berdasarkan input data. Algoritma ini menghitung perbedaan antara dua string dalam bentuk matriks dengan mencatat jumlah perubahan minimum yang diperlukan, seperti penambahan, penghapusan, atau penggantian karakter (B. H. Irawan et al., 2021)

Penelitian lain yang dilakukan oleh (Daniati et al., 2022) Algoritma *Levenshtein Distance* dapat berfungsi sebagai fitur koreksi otomatis untuk pencarian kata dalam informasi data buku, di mana sistem akan menemukan kata dengan jarak terdekat dari kata yang dicari. Informasi buku yang ditampilkan didasarkan pada jarak terpendek; semakin kecil jarak yang dihitung, semakin mirip hasil yang muncul dengan kata yang dimasukkan pengguna. Algoritma *Levenshtein Distance* yang dilakukan oleh (Ginantaka, 2023) Dengan penerapan algoritma *Levenshtein Distance*, diperoleh tingkat akurasi sebesar 80,76% dan presisi rata-rata data mencapai 46,43%. Sebaliknya, penggunaan algoritma Hamming Distance hanya menghasilkan akurasi sebesar 78,34% dengan presisi data rata-rata sebesar 30,50%.

2. Metode Penelitian

2.1. Metode Pengumpulan Data

Dengan menggunakan metode pengumpulan data primer untuk mendapatkan pemahaman yang komprehensif tentang permasalahan yang ada dan kebutuhan pengguna. Observasi dan wawancara dilakukan di departemen *Human Capital & Operation* (HCO) di Perusahaan XYZ yang merupakan sebuah perusahaan farmasi yang berlokasi di Kawasan Industri Delta Silicon Lippo, Cikarang Selatan, Kabupaten Bekasi, Jawa Barat. Berdasarkan

hasil wawancara peneliti mendapatkan informasi sistem yang sedang berjalan dan juga terkait kebutuhan aplikasi yang diharapkan seperti fitur, bisnis proses, batasan pengguna dan desain. Penelitian juga didukung oleh data sekunder yang diperoleh melalui studi pustaka.

2.2. Analisa Algoritma *Levenshtein Distance*

Perhitungan atau pendekatan *Levenshtein Distance* yang efisien, sebuah metrik yang banyak digunakan untuk mengevaluasi kesamaan urutan (Wei et al., 2024). *Levenshtein Distance* adalah metrik jarak sederhana yang diukur berdasarkan jumlah operasi *edit* yang diperlukan untuk mengubah satu *string* menjadi *string* lainnya. *Levenshtein Distance* adalah metode untuk mengukur kesamaan antara dua string dengan menghitung jumlah perubahan minimum yang diperlukan untuk mengubah satu string menjadi *string* lainnya. Dalam bentuk dasarnya, *Levenshtein Distance* antara dua string, misalnya a dan b, adalah jumlah terkecil dari operasi penyisipan, penghapusan, atau penggantian karakter yang diperlukan untuk mengonversi *string* a menjadi b atau sebaliknya. Metode menghitung jumlah minimum operasi edit yang diperlukan untuk menjadikan dua string identik (Yuslena et al., 2021).

$$lev_{a,b}(i,j) = \begin{cases} max(i,j), & \text{jika } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1 \end{cases} & \text{, jika } (a_i \neq b_j) \end{cases} \quad (1)$$

Dimana a = String Sumber, b= String Target, lev= Jarak *Levenshtein Distance*, i = Nilai Karakter String Sumber, j = Nilai Karakter String Target. Ada tiga jenis operasi utama yang dapat dilakukan oleh algoritma ini yaitu penambahan karakter, penghapusan karakter, penggantian karakter (Adawiyah & Saragih, 2022) yaitu:

a. Pada operasi penambahan karakter terjadi ketika sebuah karakter perlu ditambahkan ke *string* agar sama dengan *string* lainnya. Sebagai contoh, jika terdapat *string* "pelatihan" dan "pelatihn", maka operasi penambahan karakter diperlukan untuk menambahkan karakter "a" ke *string* kedua agar sama dengan *string* pertama.
b. Pada operasi penghapusan karakter terjadi ketika sebuah karakter perlu dihapus dari *string* agar sama dengan *string* lainnya. Sebagai contoh, jika terdapat *string* "pelatihan" dan "pelatihn", maka operasi penghapusan karakter diperlukan untuk menghapus karakter "a" dari *string* pertama agar sama dengan *string* kedua.
c. Pada operasi penggantian karakter terjadi ketika sebuah karakter perlu diganti dengan karakter lain agar sama dengan *string* lainnya. Sebagai contoh, jika terdapat *string* " traiming" dan "training", maka operasi penggantian karakter diperlukan untuk mengganti karakter "m" di *string* pertama dengan karakter "n" agar sama dengan *string* kedua.

2.3. Perhitungan Algoritma

Tabel 1 penerapan perhitungan manual Algoritma *Levenshtein Distance* tanpa menggunakan bahasa pemrograman, dengan string sumber "TRAINNING" (9 indeks) dan string target "TRAINING" (8 indeks).

Tabel 1. Operasi Matriks “TRAINNING”

			String Target							
			T	R	A	I	N	I	N	G
String Sumber		0	1	2	3	4	5	6	7	8
	T	1	0	1	2	3	4	5	6	7
	R	2	1	0	1	2	3	4	5	6
	A	3	2	1	0	1	2	3	4	5
	I	4	3	2	1	0	1	2	3	4
	N	5	4	3	2	1	0	1	2	3
	N	6	5	4	3	2	1	1	2	3
	I	7	6	5	4	3	2	1	1	2
	N	8	7	6	5	4	3	2	1	1
	G	9	8	7	6	5	4	3	2	1

Sumber: Hasil Penelitian (2024)

$$d(1,2) = min\begin{cases} d(1-1,2)+1 = d(0,2)+1 & = 3 \\ d(1,2-1)+1 = d(1,1)+1 & = 1 \\ d(1-1,2-1)+1 = d(0,1)+1 & = 2 \end{cases}$$
$$= \min(3,1,2)$$
$$= 1$$

$$d(1,3) = min\begin{cases} d(1-1,3)+1 = d(0,3)+1 & = 4 \\ d(1,3-1)+1 = d(1,2)+1 & = 2 \\ d(1-1,3-1)+1 = d(0,2)+1 & = 3 \end{cases}$$
$$= \min(4,2,3)$$
$$= 2$$

$$d(1,4) = min\begin{cases} d(1-1,4)+1 = d(0,4)+1 & = 5 \\ d(1,4-1)+1 = d(1,1)+1 & = 3 \\ d(1-1,4-1)+1 = d(0,1)+1 & = 4 \end{cases}$$
$$= \min(5,3,4)$$
$$= 3$$

$$d(1,5) = min\begin{cases} d(1-1,5)+1 = d(0,5)+1 & = 6 \\ d(1,5-1)+1 = d(1,1)+1 & = 4 \\ d(1-1,5-1)+1 = d(0,1)+1 & = 5 \end{cases}$$
$$= \min(6,4,5)$$
$$= 4$$

Perhitungan dilanjutkan hingga keseluruhan tabel terisi, berdasarkan data hasil perhitungan menggunakan Persamaan (1) pada Tabel 1, dapat disimpulkan bahwa jarak antara *string* “TRAINNING’ dengan *string* “*TRAINING*” adalah sebanyak 1. Operasi yang dilakukan yaitu penghapusan karakter “N” yang berada ditengah-tengah *string* sumber. Perhitungan nilai bobot *similaritas dari kata “*TRAINNING”:

$$similarity = \left(1 - \frac{1}{8}\right) x\ 100\%$$

$$similarity = (1 - 0.125)\ x\ 100\%$$

$$similarity = 0.875\ x\ 100\%$$

$$similarity = 87{,}5\%$$

3. Hasil dan Pembahasan

3.1 Penerapan Algorima *Levenshtein Distance* pada Sistem

Dalam penerapan aplikasi *Training Registration System* menggunakan Algoritma *Levenshtein Distance* dengan baris kode seperti terlihat pada Gambar 1.

```php
// Ubah string pencarian menjadi huruf kecil
$search = strtolower($request->input('TrainingName'));

// Batas jarak Levenshtein yang diizinkan
$threshold = 1;

// Ambil semua data dari database
$allTrainings = TrainingSubmission::get();

// Lakukan pencarian manual dengan algoritma Levenshtein
$matchingTrainings = [];
foreach ($allTrainings as $training) {
    // Normalisasi nama training ke huruf kecil
    $trainingName = strtolower($training->TrainingName);

    // Hitung jarak Levenshtein antara kata kunci pencarian dan nama training
    $levenshteinDistance = levenshtein($search, $trainingName);

    // Periksa apakah ada kesamaan kata kunci pencarian menggunakan strpos
    $hasSubstring = strpos($trainingName, $search) !== false;

    // Jika nama training mengandung kata kunci atau jarak Levenshtein kurang dari atau
       sama dengan threshold, tambahkan ke hasil pencarian
    if ($hasSubstring || $levenshteinDistance <= $threshold) {
        $matchingTrainings[] = $training;
    }
}

// Tampilkan hasil pencarian
foreach ($matchingTrainings as $match) {
    echo "Training Name: " . $match->TrainingName . "\n";
}
```

Sumber: Hasil Penelitian (2024)

Gambar 1. Program dengan Algoritma pada Aplikasi

Berdasarkan tampilan kode program dalam penggunaan algoritma, penjelasan bagian kode aplikasi *Training Registration System* menggunakan Algoritma *Levenshtein Distance*:

a. Mengubah *String* Pencarian.

   Mengubah *string* pencarian menjadi huruf kecil menggunakan fungsi *strtolower* mengubah input pencarian dari pengguna menjadi huruf kecil untuk memastikan pencarian tidak peka terhadap huruf besar atau kecil.

   $search = strtolower($request->input('TrainingName'));

b. Batas *Levenshtein Distance*.

   Menetapkan batas *Levenshtein Distance* yang diizinkan dengan nilai 1. Batas *Levenshtein Distance* ditetapkan ke 1. *Levenshtein Distance* adalah ukuran perbedaan antara dua string. Semakin kecil jaraknya, semakin mirip kedua string.

   $threshold = 1;

c. Mengambil Data dari *Database.*

Mengambil semua data dari *database* menggunakan *TrainingSubmission::get().* Kode mengambil semua data training dari *database.*

```
$allTrainings = TrainingSubmission::get();
```

d. Algoritma Pencarian.

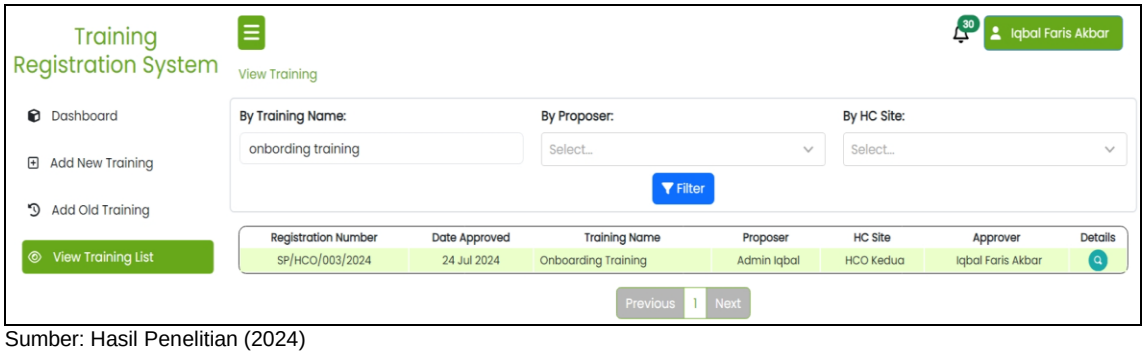
Melakukan pencarian manual dengan algoritma *Levenshtein Distance* seperti kode untuk menemukan kecocokan antara kata kunci pencarian dan nama training.

```
$matchingTrainings = [];
foreach ($allTrainings as $training) {
    $trainingName = strtolower($training->TrainingName);
    $levenshteinDistance = levenshtein($search, $trainingName);
    $hasSubstring = strpos($trainingName, $search) !== false;
    if ($hasSubstring || $levenshteinDistance <= $threshold) {
        $matchingTrainings[] = $training;
    }
}
```

Dengan normalisasi nama training yaitu mengubah nama training menjadi huruf kecil. Kemudian menghitung *Levenshtein Distance* antara kata kunci pencarian dan nama training. Lalu melakukan pemeriksaan *Substring*: Memeriksa apakah nama training mengandung kata kunci pencarian. Setelah itu menambahkan ke hasil pencarian, jika nama training mengandung kata kunci atau *Levenshtein Distance* kurang dari atau sama dengan *threshold*, maka training tersebut ditambahkan ke hasil pencarian.

3.2. Hasil Aplikasi *Training Registration System*

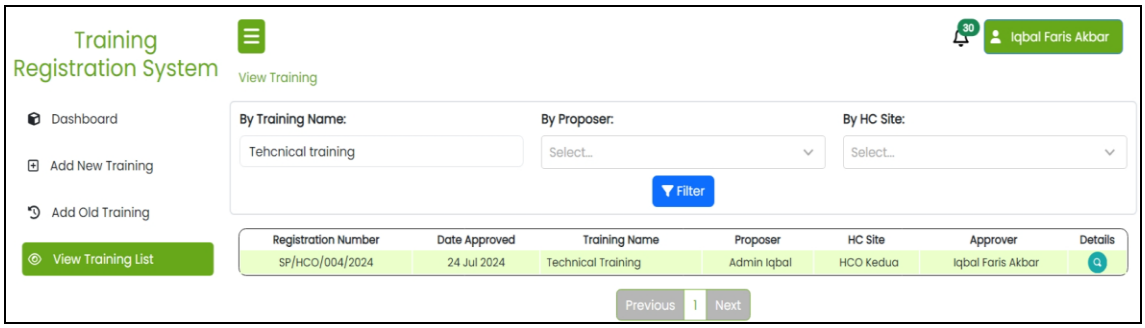
Pada Gambar 2 merupakan tampilan hasil aplikasi *Training Registration System* menggunakan algoritma *Levenshtein Distance.*



Sumber: Hasil Penelitian (2024)

Gambar 2. Tampilan Sistem dengan *input "onbording training*"

Dalam tampilan Gambar 2 dengan data sumber dengan *string "onbording training"* dengan data *string target* yaitu "*Onboarding Training*". Sistem dapat menampilkan data target meskipun dengan data *string* sumber memiliki kesalahan pengetikan.



Sumber: Hasil Penelitian (2024)

Gambar 3. Tampilan Sistem dengan input "*Tehcnial training*"

Dalam tampilan Gambar 3 data sumber dengan *string "Tehcnial training"* dengan data string target yaitu "*Technical Training*". Sistem dapat menampilkan data target meskipun dengan data *string* sumber memiliki kesalahan pengetikan.

4. Kesimpulan

Berdasarkan hasil dari penelitian dan pembahasan yang sudah dilakukan oleh peneliti mengenai penerapan algoritma *Levenshtein Distance* untuk meningkatkan efisiensi pencarian training pada aplikasi *training registration system* berbasis *web* dapat disimpulkan bahwa algoritma *Levenshtein Distance* telah dapat diterapkan pada sistem untuk mencari data pelatihan berdasarkan nama pelatihan yang diketikkan pengguna, meskipun terdapat kesalahan ketik atau perbedaan ejaan. Algoritma menghitung "jarak" antara kata yang dicari dengan nama-nama pelatihan yang tersimpan di dalam sistem, sehingga dapat menampilkan hasil yang paling relevan.

Daftar Pustaka

Adawiyah, R., & Saragih, N. E. (2022). Implementasi Algoritma *Levenshtein Distance* Dalam Mendeteksi Plagiarisme. *Journal Computer Science and Information Technology (JCoInT)*, *5*(1), 54–63. https://jurnal.ulb.ac.id/index.php/JCoInT/article/view/3086/2437

Daniati, Y. N., Zulkarnain, I. A., & Nurfitri, K. (2022). Penerapan Algoritma *Levenshtein Distance* Pada Sistem Pencarian Data Buku Berbasis Web. *Komputek*, *6*(1), 81. https://doi.org/10.24269/jkt.v6i1.1144

Ginantaka, G. S. (2023). *Uji Ketepatan Pengenalan Sidik Jari Dengan Metode Levenshtein Distance dan Hamming Distance* [Universitas Kristen Duta Wacana]. https://katalog.ukdw.ac.id/8056/

Irawan, A., Ningsih, D. P., Setiawan, F. D., Saputra, I. E., Marvin, J., Wisnu, R., & Pamungkas, P. (2024). Perancangan Aplikasi Power Apps (Consumable) PT. XYZ. *Nusantara Journal of Multidisciplinary Science*, *2*(1), 161–172. https://jurnal.intekom.id/index.php/njms/article/view/286

Irawan, B. H., Simarangkir, M. S. H., & Erlinna, E. (2021). Deteksi Kemiripan Judul Skripsi Menggunakan Algoritma *Levenshtein Distance* pada Kampus STMIK MIC Cikarang. *Edutic - Scientific Journal of Informatics Education*, *7*(2), 143–149. https://doi.org/10.21107/edutic.v7i2.10051

Nadeak, B. (2019). Manajemen Pelatihan dan Pengembangan. In *Buku Materi Pembelajaran Manajemen Pelatihan dan Pengembangan*.

Novita, Y., & Zahra, R. (2024). Penerapan Artificial Intelligence (AI) untuk Meningkatkan Efisiensi Operasional di Perusahaan Manufaktur : Studi Kasus PT XYZ. *Jurnal Manajemen Dan Teknologi*, *1*(1), 11–21. https://doi.org/https://doi.org/10.35870/jmt.v1i1.773

Prihandono, G., & Amir, M. T. (2024). Implementasi Teknologi Informasi dalam Meningkatkan Efisiensi Organisasi dan Daya Saing Perusahaan. *Journal of Economics and Business UBS*, *13*(2), 577–587. https://doi.org/10.52644/joeb.v13i2.1556

Wei, X., Guo, A. J. X., Sun, S., Wei, M., & Yu, W. (2024). *Levenshtein Distance* Embedding with Poisson Regression for DNA Storage. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(14), 15796–15804. https://doi.org/10.1609/aaai.v38i14.29509

Yuslena, Khatimi, H., & Fajrin, R. A. (2021). Deteksi Plagiarisme Menggunakan Algoritma *Levenshtein Distance*. *Jurnal Teknologi Informasi Universitas Lambung Mangkurat (JTIULM)*, *6*(1), 31–38. https://doi.org/10.20527/jtiulm.v6i1.66