Jurnal Serina Sains, Teknik dan Kedokteran
Vol. 1, No. 1, Feb 2023: hlm 18-26 ISSN-L 2986-5204 (Versi Elektronik)

# *OPTICAL CHARACTER RECOGNITION* MENGGUNAKAN *UIPATH* DAN PENCOCOKAN DATA SERTIFIKAT DENGAN *ALGORITMA LEVENSHTEIN DISTANCE*

**Cynthia Natalie[1], Viny Christanti Mawardi[2] & Manatap Dolok Lauro Sitorus[3]**

[1]Prograsm Studi Sarjana Teknik Informatika, Universitas Tarumanagara Jakarta
*Email: cynthia.535190034@stu.untar.ac.id*
[2]Fakultas Teknologi Informasi, Universitas Tarumanagara Jakarta
*Email: vinym@fti.untar.ac.id*
[3]Fakultas Teknologi Informasi, Universitas Tarumanagara Jakarta
*Email: manataps@fti.untar.ac.id*

***ABSTRACT***

*The collection of the certificates was one of the requirements for graduating from Tarumanagara University that the certificate became an important point of attention to improving the competency of Tarumanagara University students. Certificates can be collected from seminars, workshops, courses, and so forth. The Certificate Information Extraction design were created using Uipath Studio application that uses vb.Net programming language and Levenshtein Distance Algorithm. The design aims to assist the study program in validation on student certificates file by obtaining better accuracy using the Levenshtein Distance Algorithm.. The design uses input data as student certificates that's next to be processed by text preprocessing consisting of text deductions (parsing), case folding, lexical analysis (tokenizing), and text removal (stopword removal). After processing, the Levenshtein Distance Algorithm will be used to calculate the minimum distance between one text and the other with a two-dimensional matrix operation, thus determining the validity of student certificates. The results of this design represent that using the Levenshtein Distance Algorithm, obtaining the best word accuracy result of 83.52% and RPA running time of 94.7 ms.*

***Keywords:*** *Certificates, Confusion Matrix, Levenshtein Distance, RPA, Student's data*

**ABSTRAK**

Pengumpulan sertifikat merupakan salah satu syarat kelulusan mahasiswa Universitas Tarumanagara sehingga sertifikat menjadi salah satu poin penting untuk diperhatikan dalam meningkatkan kompetensi mahasiswa Universitas Tarumanagara. Sertifikat dapat dikumpulkan dari seminar, *workshop,* kursus, dan sebagainya. Rancangan ekstraksi informasi file sertifikat ini dibuat dengan aplikasi *UiPath Studio* yang menggunakan bahasa pemrograman *VB.Net* dan *Algoritma Levenshtein Distance* (LD). Rancangan ini bertujuan untuk membantu program studi dalam melakukan validasi pada file sertifikat mahasiswa dengan memperoleh akurasi yang lebih baik menggunakan *Algoritma Levenshtein Distance.* Rancangan menggunakan data input berupa sertifikat mahasiswa yang selanjutnya akan diproses dengan *text preprocessing* yang terdiri dari: (a) penguraian teks *(parsing); (b) case folding;* (c) analisis leksikal *(tokenizing)*; dan (d) penghapusan teks *(stopword removal)*. Setelah melalui *text preprocessing, Algoritma Levenshtein Distance* akan digunakan untuk menghitung jarak minimum antara teks yang satu dengan teks yang lain dengan operasi – operasi perhitungan matriks dua dimensi, sehingga dapat menentukan validitas sertifikat mahasiswa. Hasil dari rancangan ini menunjukkan dengan menggunakan *Algoritma Levenshtein Distance* dapat diperoleh hasil akurasi kata terbaik sebesar 83,52% dan waktu *RPA* berjalan sebesar 94,7 ms.

**Kata Kunci:** Data mahasiswa, Levenshtein Distance, RPA, Sertifikat, Confusion Matrix

## 1. PENDAHULUAN

Sertifikat adalah surat keterangan sebagai tanda bukti telah mengikuti suatu kegiatan, organisasi, ataupun telah berhasil menyelesaikan kursus tertentu. Saat ini, sertifikat dikumpulkan oleh mahasiswa dengan berbagai cara seperti: (a) menghadiri kegiatan seminar atau *workshop*; (b) bergabung dalam suatu organisasi; dan (c) mengikuti bakti sosial serta PKM. Validasi sertifikat mahasiswa dapat dilakukan dengan menggunakan *Robotic Process Automation (RPA)* dalam mengekstrak informasi dari data berupa sertifikat mahasiswa. *Robotic Process Automation (RPA)*

 https://doi.org/10.24912/jsstk.v1i1.22747

adalah sebuah teknologi berupa perancangan "robot" untuk membantu pengguna dalam mengerjakan tugas yang berulang dalam waktu yang lebih singkat sehingga pengerjaan selesai dengan lebih cepat dan efisien. Terdapat berbagai macam tools yang dapat digunakan dalam perancangan *RPA* yakni: (a) *Automation Anywhere;* (b) *Blue Prism;* dan (c) *UiPath*. (Florentina, 2020). Dalam merancang *RPA* untuk melakukan ekstraksi informasi file sertifikat, dibutuhkan algoritma yang dapat digunakan untuk melakukan pencocokan *string,* salah satunya adalah *Algoritma Levenshtein Distance.* Proses *RPA* diawali dengan mempersiapkan 3 data persiapan seperti: (a) data mahasiswa; (b) data sertifikat; dan (c) data kegiatan mahasiswa, lalu menjalankan *RPA* dengan menggunakan *UiPath Assistant,* kemudian *RPA* melakukan pembacaan data dengan *Optical Character Recognition (OCR)* untuk mendapatkan informasi dalam sertifikat dan menuliskan hasil *OCR* tersebut ke dalam *excel* untuk dilakukan perbandingan antar karakter dengan mengimplementasikan *Algoritma Levenshtein Distance* sehingga dapat dilakukan validasi antar sertifikat.

Berdasarkan penelitian yang dilakukan dengan judul "*Analyzing and Experimenting Open Source OCR Engines in RPA with Levenshtein Distance Algorithm*", didapatkan akurasi sebesar 76.19% dan 71.05% masing – masing menggunakan *Microsoft OCR* dan *Tesseract OCR* disertai *Algoritma Levenshtein Distance* sebagai metode penelitiannya. (AK & CS 2020). Oleh karena itu, pada penelitian ini *OCR* dilakukan dengan mengimplementasikan *Algoritma Levenshtein Distance* pada sertifikat untuk memperoleh akurasi yang lebih baik. *Levenshtein Distance* adalah salah satu algoritma yang bertujuan untuk mengoptimalkan pencocokan *string*, dimana perbandingan file dilakukan antara data input dan data hasil *OCR* dengan menggunakan operasi – operasi tertentu seperti operasi penyisipan karakter, penghapusan karakter, dan penukaran karakter. (Hossain et al, 2019). *Output* yang dihasilkan dari perancangan *RPA* adalah berupa *excel* yang berisikan data berupa nama mahasiswa, NIM mahasiswa, kegiatan yang diikuti dan tanggal kegiatan tersebut disertai status sertifikat *(valid/not valid).*

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka rumusan rancangan yang akan dijelaskan adalah sebagai berikut: kebutuhan validasi sertifikat mahasiswa dengan cepat dan efisien menggunakan *RPA* sebagai salah satu persyaratan kelulusan dan penerapan *Algoritma Levenshtein Distance* untuk memperoleh akurasi yang lebih baik dalam menggunakan *OCR* untuk pembacaan sertifikat.

## 2. METODE PENELITIAN

Sistem yang dirancang adalah *RPA* dalam mengimplementasikan *Algoritma Levenshtein Distance* untuk ekstraksi informasi file sertifikat. Pengguna meletakkan data input berupa *excel* yang berisikan data mahasiswa disertai NIM dan email sebagai kata kunci pencarian ke dalam folder yang berisikan input berupa sertifikat mahasiswa dan menjalankan *RPA* dengan menggunakan *UiPath Assistant.*

*Natural Language Processing* adalah salah satu cabang ilmu pengetahuan yang berkaitan dengan kecerdasan buatan yang berfokus pada pengolahan bahasa manusia baik yang diucapkan maupun yang ditulis dengan kata – kata. Terdapat beberapa level dalam *Natural Language Processing* yang dijelaskan sebagai berikut (Chowdhary, 2020):

(a) Fonologi, merupakan level *NLP* yang berkaitan dengan suara yang dapat menghasilkan kata – kata yang dapat dikenali dengan menggunakan aplikasi kecerdasan buatan;

(b) Morfologi, merupakan level *NLP* yang berkaitan dengan mendefinisikan asal usul dari suatu kata dan mengenai pembentukan suatu kata menggunakan aplikasi kecerdasan buatan;

(c) Leksikal, merupakan level *NLP* yang berkaitan dengan mendefinisikan suatu teks atau kalimat yang diubah ke dalam pecahan kata untuk diidentifikasi oleh aplikasi kecerdasan buatan;

(d) Sintaktik, merupakan level *NLP* yang berkaitan dengan urutan kata dalam pembentukan suatu teks atau kalimat dalam melakukan identifikasi menggunakan aplikasi kecerdasan buatan. Pada level ini, *Natural Language Processing* digunakan dalam cakupan tanya dan jawab;

(e) Semantik, merupakan level *NLP* yang berkaitan dengan pemahaman suatu arti dalam kata – kata dimana kata – kata tersebut membentuk suatu kalimat yang utuh dengan menggunakan aplikasi kecerdasan buatan;

(f) Discourse, merupakan level *NLP* yang berkaitan dengan pengenalan hubungan antar kalimat yang bertujuan untuk mengidentifikasi penggunaan kata ganti, keterangan tempat, waktu dan lainnya yang digunakan dalam sebuah kalimat menggunakan aplikasi kecerdasan buatan;

(g) Pragmatik, merupakan level *NLP* yang berkaitan dengan konteks dalam sebuah kata atau kalimat yang mencerminkan keadaan atau situasi dimana sebuah kata atau kalimat tersebut digunakan dengan aplikasi kecerdasan buatan.

*Text Preprocessing* merupakan tahap awal dalam *Natural Language Processing* yang bertujuan untuk mempersiapkan sebuah data teks atau kalimat yang tidak terstruktur menjadi data yang baik dan dapat diolah oleh sistem. Dalam melakukan *text preprocessing,* ada beberapa proses yang dilakukan sebelum mengolah suatu data teks atau kalimat. Berikut beberapa proses yang dilakukan dalam *text preprocessing* (Jain et al, 2018):

(a) Penguraian Teks *(Parsing),* merupakan proses untuk memecah suatu teks menjadi kalimat per kalimat dengan tujuan untuk menentukan teks yang diproses;
(b) *Case Folding,* merupakan proses untuk melakukan konversi teks yang menggunakan huruf besar dan huruf kecil menjadi teks yang hanya menggunakan huruf besar atau huruf kecil;
(c) Analisis Leksikal / Tokenisasi, merupakan proses untuk melakukan pemotongan suatu teks atau kalimat menjadi bagian – bagian yang lebih kecil. Pada tahapan ini, dilakukan penghapusan tanda baca dan angka – angka yang dianggap tidak berpengaruh terhadap pemrosesan suatu teks atau kalimat;
(d) *Stopword Removal,* merupakan proses untuk menyaring kata – kata yang dibutuhkan dalam pemrosesan suatu teks atau kalimat yang akan digunakan pada tahapan berikutnya.

Algoritma Pencocokan *String* adalah algoritma yang digunakan dalam melakukan pencarian *string* dengan menggunakan beberapa komponen tertentu seperti pola, teks, dan karakter. Secara garis besar, algoritma pencocokan *string* dibedakan menjadi dua jenis yaitu:

(a) *Exact String Matching,* merupakan algoritma pencocokan *string* yang dilakukan dengan membandingkan suatu karakter dalam *string* dengan jumlah dan urutan yang sama persis. (Hakak, et al, 2019);
(b) *Inexact String Matching,* merupakan algoritma pencocokan *string* yang dilakukan dengan membandingkan suatu karakter dalam *string* dengan melakukan pencarian *string* pendek (pola) atau *string* panjang (teks), baik secara penulisan (approximate string matching) maupun dengan ucapan *(phonetic string matching)* dimana kedua *string* yang dibandingkan memiliki jumlah dan urutan yang berbeda. Algoritma ini biasa dilakukan pencarian mulai dari kiri ke kanan, contohnya dalam penggunaan algoritma Brute Force dan algoritma Morris & Pratt yang selanjutnya dikembangkan oleh Knuth, Morris, dan Pratt dengan pendekatan pola dari sebuah *string.* (Rumapea, 2021)

 https://doi.org/10.24912/jsstk.v1i1.22747

*Algoritma Levenshtein Distance* atau yang biasanya sering disebut sebagai edit *distance* merupakan salah satu algoritma dalam melakukan pencocokan *string* dengan jumlah dan urutan karakter yang berbeda. Algoritma ini bertujuan untuk mengoptimalkan pencocokan *string* dengan menghitung jumlah minimal dalam bertransformasi dari suatu *string* ke dalam bentuk *string* yang lain dengan menggunakan operasi – operasi seperti penyisipan karakter, penghapusan karakter, dan penukaran karakter. (Daniati, Nurfitri, & Zulkarnain, 2022).

Dalam melakukan implementasi *Algoritma Levenshtein Distance*, matriks dua dimensi digunakan untuk menghitung perbedaan antara kedua string yang dibandingkan. Dalam matriks dua dimensi tersebut, terdapat jumlah – jumlah hasil perhitungan berupa penyisipan, penghapusan, dan penukaran yang diperlukan dalam melakukan transformasi dari suatu *string* ke dalam bentuk *string* yang lain. Berikut operasi – operasi perhitungan dalam penggunaan *Algoritma Levenshtein Distance* (Iswari, Rusli, & Setiabudi, 2021):

(a) Penyisipan karakter, merupakan salah satu operasi perhitungan dalam *Algoritma Levenshtein Distance* yang bertujuan untuk menyisipkan karakter dalam suatu teks atau kalimat;
(b) Penghapusan karakter, merupakan salah satu operasi perhitungan dalam *Algoritma Levenshtein Distance* yang bertujuan untuk menghapus kelebihan karakter dalam suatu teks atau kalimat;
(c) Penukaran karakter, merupakan salah satu operasi perhitungan dalam *Algoritma Levenshtein Distance* yang bertujuan untuk melakukan penukaran antar karakter sehingga karakter berada dalam posisi yang benar sesuai dengan urutan dalam suatu teks atau kalimat.

Hasil perhitungan *Algoritma Levenshtein Distance* berupa jarak perbedaan antara suatu *string* dengan *string* yang lain. Rumus 1, 2, 3 digunakan untuk menghitung jarak *Levenshtein Distance* antara kedua *string,* yakni s = $s_1$.....$s_n$ dan t = $t_1$.....$t_m$ yaitu :

$$lev_{a,b}(i, j) = \{0 \qquad i$$

dimana: s = *string* yang direpresentasikan sebagai baris
t = *string* yang direpresentasikan sebagai kolom
i = *index* dari *string* s
j = *index* dari *string* t
n = panjang *string* s
m = panjang *string* t
lev = jarak perbedaan antara *string* s dan *string* t

Rumus – rumus terkait algoritma *Levenshtein Distance* tersebut digunakan dalam alur *algoritma Levenshtein Distance* langkah per langkah yakni sebagai berikut :

**Langkah 1**: Inisialisasi

a. Inisialisasi S sebagai *string* pertama.
b. Inisialisasi T sebagai *strin*g kedua.
c. Hitung panjang *string* S dan T dengan variabel m dan n.
d. Buat matriks dengan ukuran 0....m baris dan 0....n kolom.
e. Inisialisasi baris pertama dengan urutan 0....n.
f. Inisialisasi kolom pertama dengan urutan 0....m.

**Langkah 2**: Proses perhitungan matriks

a. Cek apakah S[i] berada dalam rentang 1<i<n.
b. Cek apakah T[j] berada dalam rentang 1<j<m.
c. Jika S[i] = T[j], maka input nilai yang terletak tepat pada diagonal atas sebelah kiri. yaitu dengan rumus d[i, j] = d[i-1, j-1].
d. Jika S[i] ≠ T[j], maka input nilai minimum dari :
   i. Nilai yang terletak tepat di atasnya dan ditambah satu, yaitu dengan rumus d[i, j] = d[i, j-1]+1.
   ii. Nilai yang terletak tepat di sebelah kirinya dan ditambah satu, yaitu dengan rumus d[i, j] = d[i-1, j]+1.
   iii. Nilai yang terletak tepat pada diagonal atas sebelah kirinya dan ditambah satu, yaitu dengan rumus d[i, j] = d[l-1, j-1]+1.

**Langkah 3:** Input nilai matriks pada baris ke-i dan kolom ke-j, yaitu pada d[i, j].

**Langkah 4:** Ulangi langkah 3 hingga d[m,n] ditemukan.

*Confusion Matrix.* Dalam perancangan *RPA,* dibutuhkan evaluasi untuk menguji akurasi terkait hasil *RPA* dengan *algoritma Levenshtein Distance* menggunakan *Confusion Matrix.* (Dershowitz & Kissos, 2016). Berdasarkan tabel 2 tersebut, terdapat 4 hasil evaluasi yakni: (a) *True Positive* (TP); (b) *False Positive* (FP); (c) *False Negative* (FN); dan (d) *True Negative* (TN). Berikut penjelasan terkait *confusion matrix*:

(a) Akurasi *(accuracy)* adalah perbandingan untuk mengukur seberapa akurat hasil prediksi dengan hasil aktualnya. Rumus untuk menghitung akurasi dapat dilihat pada rumus 4 berikut.

$$accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (4)$$

(b) *Recall* adalah perbandingan untuk mengukur seluruh data prediksi yang benar sesuai dengan data aktual yang benar *(True Positive)* dengan seluruh data aktual yang benar. Rumus untuk menghitung *recall* dapat dilihat pada rumus 5 berikut.

$$recall = \frac{TP}{(TP+FN)} \quad (5)$$

(c) *Precision* adalah perbandingan untuk mengukur seluruh data prediksi yang benar sesuai dengan data aktual yang benar *(True Positive)* dengan seluruh data prediksi yang benar. Rumus untuk menghitung precision dapat dilihat pada rumus 6 berikut.

$$precision = \frac{TP}{(TP+FP)} \quad (6)$$

(d) *F-Measure* adalah nilai rata – rata antara *recall* dan *precision* yang merepresentasikan hasil dengan skor 1 atau 0. Rumus untuk menghitung *F-Measure* dapat dilihat pada rumus 7 berikut.

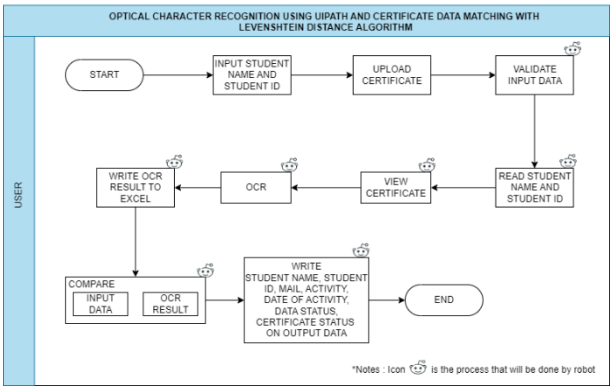
$$f - measure = \frac{2*recall*precision}{(recall+precision)} \quad (7)$$

## 3. HASIL DAN PEMBAHASAN

Data *input* yang digunakan dalam perancangan *RPA* terdapat 3 data persiapan yakni: (a) data mahasiswa; (b) data sertifikat; dan (c) data kegiatan mahasiswa. Proses perancangan *RPA* dapat dilihat pada Gambar 1.

 https://doi.org/10.24912/jsstk.v1i1.22747

**Gambar 1**
*Proses perancangan RPA*




Data sertifikat yang akan digunakan berjumlah 88 data sertifikat yang dibagi menjadi 2 bagian yaitu: (a) 10 sertifikat data latih; dan (b) 78 sertifikat data pengujian. Data – data yang divalidasi pada sertifikat dengan menggunakan *RPA* dan *Algoritma Levenshtein Distance* dapat dilihat pada Gambar 2.

**Gambar 2**
*Sertifikat dengan menggunakan RPA dan Algoritma Levenshtein Distance*


Hasil pengujian validasi sertifikat menggunakan *RPA* dapat dilihat pada Tabel 1 sebagai berikut.

**Tabel 1**
*Hasil pengujian validasi sertifikat menggunakan RPA*

| *Name* | *Certificate File* | *Certificate Status* | *Validation Result (True/False)* |
|---|---|---|---|
| Dian Anggraini Cahyaningtyas |  | Not Valid | False |
| Dian Anggraini Cahyaningtyas |  | Valid | True |
| Raymond Tjahyadi |  | Valid | True |
| Raymond Tjahyadi |  | Valid | True |
| Cynthia Natalie |  | Not Valid | False |
| Cynthia Natalie |  | Valid | True |
| Aldo Valerian |  | Not Valid | True |
| Aldo Valerian |  | Valid | True |
| Andri Firnandius |  | Valid | True |
| Andri Firnandius |  | Valid | True |

Berdasarkan Tabel 1, terdapat 2 hasil validasi, yaitu *valid* atau tidak *valid*. Sertifikat *valid* didasarkan pada *OCR* yang disebabkan oleh sertifikat yang terbaca dengan jelas menggunakan *font* yang dapat dibaca oleh *RPA*. Selain itu, sertifikat tidak *valid* didasarkan pada sertifikat yang tidak terbaca dengan jelas menggunakan *font* yang tidak dapat dibaca oleh *RPA* atau *font* yang terlalu kecil sehingga tidak terbaca dengan menggunakan *OCR*. Hasil pengujian dapat dilihat pada Tabel 2 dan Tabel 3.

**Tabel 2**
*Hasil pengujian RPA tanpa Algoritma Levenshtein Distance*

| *Testing Categories* | *Valid* | *Not Valid* |
|---|---|---|
| Words Accuracy | 86.36% | 79.54% |
| False Positive | 2.2% | 0.0% |
| Run Time Average (ms) | 185.3 | 17.6 |
| Words Accuracy Average | (86.36+79.54)/2 = 82.95% | |

 https://doi.org/10.24912/jsstk.v1i1.22747

**Tabel 3**

*Hasil pengujian RPA dengan Algoritma Levenshtein Distance*

| ***Testing Categories*** | ***Valid*** | ***Not Valid*** |
|---|---|---|
| Words Accuracy | 86.36% | 80.68% |
| False Positive | 1.1% | 0.0% |
| Run Time Average (ms) | 94.7 | 10.4 |
| Words Accuracy Average | (86.36+80.68)/2 = 83.52% | |

Berdasarkan Tabel 2 dan Tabel 3 di atas, hasil pengujian *RP*A menunjukkan akurasi yang lebih baik dengan menggunakan *Algoritma Levenshtein Distance,* dari segi waktu rata – rata *RPA* berjalan dan akurasi rata – rata kata yang dibandingkan dengan hasil pengujian *RPA* tanpa menggunakan *Algoritma Levenshtein Distance.*

## 4. KESIMPULAN DAN SARAN

Berdasarkan implementasi *Algoritma Levenshtein Distance* dan analisis *RPA,* didapatkan kesimpulan sebagai berikut: berdasarkan hasil pengujian validasi sertifikat, *Algoritma Levenshtein Distance* dapat digunakan dengan sangat baik pada level sintaktik, yakni dalam pengenalan kata – kata pada sertifikat dan berdasarkan hasil pengujian *RPA,* hasil akurasi kata terbaik adalah dengan menggunakan *Algoritma Levenshtein Distance* dengan akurasi sebesar 83,52% dengan waktu *RPA* berjalan sebesar 94,7 ms.

Saran yang diajukan sebagai perbaikan dan pengembangan sistem selanjutnya adalah sebagai berikut: menyesuaikan dengan menggunakan *Google Cloud OCR* untuk mendapatkan hasil *OCR* terbaik dan menambahkan data sertifikat lebih banyak untuk data latih dan data pengujian dalam melakukan validasi sertifikat.

**Ucapan Terima Kasih** (*Acknowledgement*)

Terima kasih kepada pihak – pihak utama yang mendukung penelitian ini, terutama kepada Fakultas Teknologi Informasi Universitas Tarumanagara yang membantu serta mendukung dalam proses penelitian "*Optical Character Recognition* menggunakan *UiPath* dan Pencocokan Data Sertifikat dengan *Algoritma Levenshtein Distance"* sehingga penelitian ini dapat berjalan dengan baik.

## REFERENSI

Chowdhary, K. (2020). *Fundamentals of artificial intelligence.* Springer Nature India Private Limited 2020. New Delhi, India.

Daniati, Y. N.; Nurfitri, K and Zulkarnain, I. A. (2022). Penerapan Algoritma Levenshtein Distance pada Sistem Pencarian Data Buku Berbasis Web. *KOMPUTEK, Vol. 6(1)*.

Das, A. K.; Hossain, M. M.; Labib, M. F.; Mukta, M and Rifat, A. S. "Auto-correction of english to bengali transliteration system using levenshtein distance", (pp. 1-5). 2019 7th International Conference on Smart Computing & Communications (ICSCC), Malaysia, 28-30 June 2019,

Dershowitz, N and Kissos, I. (2016). “OCR error correction using character correction and feature-based word classification”. 2016 12th IAPR Workshop on Document Analysis Systems (DAS), Greece, 11-14 April 2016, pp. 198-203,

Florentina, M. (2020). Web Data extraction with robot process automation. study on linkedin web scraping using uipath studio. *Annals of 'Constantin Brancusi' University of Targu-Jiu. Engineering Series(1)*.

Gilkar, G. A.; Hakak, S. I.; Imran, M; Kamsin, A.; Khan, W. Z. and Shivakumara, P. (2019). Exact string matching algorithms: Survey, issues, and future research directions. *IEEE Access, 7*, 69614-69637.

Iswari, Ni Made Satvika; Rusli, Andre and Setiabudi, Reza. (2021). Enhancing text classification performance by preprocessing misspelled words in Indonesian language. *TELKOMNIKA (Telecommunication Computing Electronics and Control). Vol. 19(4)*.

Jain, Aditya; Kulkarni, Gandhar and Shah, Vraj. (2018). Natural Language Processing. *International Journal of Computer Sciences and Engineering, Vol. 16(1)*, pp. 2647-2693.

Rumapea, Humuntal. (2021). Deteksi Kemiripan Artikel Melalui Keywords dengan Metode Fuzzy String Matching dalam Natural Language Processing" *METHOMIKA: Jurnal Manajemen Informatika & Komputerisasi Akuntansi. Vol 5(1)*, pp. 60-66.

Swashthika, A.K. & Diwaan, C. S. (2020). Analyzing and experimenting open source ocr engines in rpa with levenshtein distance algorithm. *International Research Journal on Advanced Science Hub. 2(125)*.