IMPLEMENTASI ALGORITMA LEVENSHTEIN DISTANCE SEBAGAI *AUTOCORRECTION* PADA SISTEM PENCARIAN JUDUL FILM BERBASIS APLIKASI ANDROID

Yohanes Dwiki Kurniawan, Agus Sidiq Purnomo
Teknik Informatika, Fakultas Teknologi Informasi, Universitas Mercu Buana Yogyakarta
Jl. Jembatan Merah No. 84 C Gejayan Yogyakarta
yohanesdwiki08@gmail.com

ABSTRAK

Perkembangan teknologi memudahkan pengguna dalam mengakses informasi, termasuk pencarian judul film. Namun, sistem pencarian yang ada sering kali tidak efektif dalam menangani kesalahan ketik atau *typo*, sehingga menurunkan kenyamanan pengguna. Untuk mengatasi masalah tersebut dibutuhkan sebuah fitur sebagai *autocorrection keyword*. Penelitian ini bertujuan untuk mengimplementasikan algoritma Levenshtein Distance sebagai fitur *autocorrection* pada sistem pencarian judul film berbasis aplikasi Android. Algoritma Levenshtein Distance menghitung jarak antara dua *string* berdasarkan jumlah operasi *add*, *delete*, atau *substitution* yang diperlukan untuk mengubah satu *string* menjadi string lainnya. Dataset yang digunakan terdiri dari 100.000 data film yang diambil dari API TMDB dan diproses melalui tahap *preprocessing* dengan *tokenization* dan *filtering*. Pengujian dilakukan dengan metode Black Box Testing untuk mengevaluasi akurasi fitur *autocorrection*. Hasil pengujian menunjukkan bahwa algoritma ini mampu mencapai akurasi sebesar 86,67% dalam mengenali dan mengoreksi kesalahan ketik. Penelitian ini membuktikan efektivitas algoritma Levenshtein Distance dalam memperbaiki kesalahan ketik pada pencarian judul film.

Kata kunci : *Levenshtein Distance, Autocorrection, Pencarian, Android*

1. PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi pada zaman sekarang memudahkan manusia dalam mengelola berbagai hal. Repositori merupakan contoh penggunaan kemajuan teknologi ini untuk menyimpan koleksi dalam bentuk digital [1]. Sistem pencarian judul film adalah sebuah mekanisme yang memungkinkan pengguna untuk menemukan film tertentu berdasarkan kata kunci atau sebagian kata yang mereka masukkan[2]. Sistem pencarian ini biasanya dirancang untuk mencocokkan masukkan pengguna dengan data judul film yang ada [2]. Saat ini, sistem pencarian judul film yang ada di berbagai platform memiliki kelemahan dalam hal penanganan kesalahan penulisan atau *typo*. Pengguna sering kali harus mengetikkan *keyword* judul film secara akurat untuk mendapatkan hasil yang relevan.

Untuk mengatasi masalah kesalahan *keyword* pencarian, diperlukan implementasi algoritma koreksi otomatis atau *autocorrection* pada sistem yang mampu mengenali dan mengoreksi kesalahan ketik pengguna. Salah satu algoritma yang bisa digunakan adalah Levenshtein Distance, dengan menghitung jarak antara dua kata berdasarkan jumlah operasi yang diperlukan untuk mengubah satu kata menjadi kata lainnya [3], [4]. Algoritma ini dapat digunakan untuk menemukan judul film yang mendekati pengetikan pengguna, meskipun terdapat kesalahan dalam pengejaan.

Tujuan penelitian ini adalah implementasi dan evaluasi algoritma Levenshtein Distance untuk pencarian judul film pada aplikasi android. Membangun fitur *autocorrection* yang terintegrasi dengan sistem pencarian pada aplikasi tersebut. Sehingga sistem pencarian dapat memberikan hasil yang relevan dan memudahkan pengguna dalam menemukan film yang diinginkan tanpa harus menuliskan judul secara akurat.

2. TINJAUAN PUSTAKA

2.1. Algoritma Levenshtein Distance

Algoritma Levenshtein Distance dikembangkan oleh ilmuwan Soviet bernama Vladimir Levenshtein, pada tahun 1965. Algoritma ini dirancang untuk mengukur perbedaan antara dua string atau kata dengan menghitung jumlah minimum operasi (*add*, *delete*, atau *subtitution*) yang diperlukan untuk mengubah satu *string* menjadi *string* lainnya.[5]

Algoritma ini menggunakan *array* dua dimensi dalam menghitung biaya yang dibutuhkan untuk mengubah *string*. Penghitungan dimulai dari pojok kiri atas sampai diketahui jumlah biaya yang berada pada pojok kanan bawah.[6]

2.2. Aplikasi Android

Android adalah sistem operasi yang dirancang untuk perangkat seluler, mencakup sistem operasi, middleware, serta aplikasi yang berbasis pada Linux[7]. Kemudian aplikasi android adalah implementasi berupa kumpulan instruksi *(instruction)* atau pernyataan *(statement)* yang dirancang sedemikian rupa agar perangkat android mampu memproses masukan menjadi keluaran[7].

2.3. Pengujian Black Box

Pengujian dilakukan untuk memastikan fungsi aplikasi sudah sesuai dengan yang diharapkan pengembang, sebelum diluncurkan kepada pengguna

umum[8]. Metode pengujian Black Box, yaitu evaluasi fungsionalitas aplikasi yang berfokus pada *input* dan *output* tanpa memeriksa kode atau implementasi sistem[8], [9].

2.4. Penelitian Dengan Levenshtein Distance

Implementasi algoritma Levenshtein Distance sebagai *autocorrection* pernah dilakukan pada aplikasi Drugs e-dictionary oleh peneliti lain. Dengan fitur ini, pengguna dapat terbantu untuk mencari nama obat yang sulit pengejaannya. Sistem menampilkan keluaran berupa nama obat-obatan yang berdekatan dengan kata masukkan. Berdasarkan pengujian mendapatkan hasil *recall* dan *precision* sebesar 90%.[4]

Penelitian lain dengan algoritma ini juga pernah dilakukan dalam modul pencarian yang diterapkan pada aplikasi Lagu-Lagu Nasional. Pada penelitian ini dilakukan *preprocessing* terhadap dataset judul lagu-lagu nasional melalui beberapa tahapan *tokenizing*, *filtering*, dan *stemming*. Setelah itu, tahap *processing* dari menggunakan algoritma Levenshtein Distance untuk membandingkannya sampai pencarian judul lagu-lagu.[5]

3. METODOLOGI PENELITIAN

3.1 Pengumpulan Data

Dataset film dibutuhkan sebagai pembanding dengan *keyword* dari inputan pengguna aplikasi. Dataset menggunakan 100.000 data informasi film yang diambil dari situs web Kaggle. Kaggle adalah sebuah *platform online* yang dikembangkan oleh Google untuk menyediakan data untuk kompetisi sains, dataset publik, dan komunitas untuk berbagi pengetahuan. Atribut data berisikan *id*, *title*, *release_date*, *popularity*, dan *genres* yang berasal dari TMDB (The Movie Database). TMDB merupakan website database untuk film dan acara TV yang dibuat oleh komunitas sejak tahun 2008.

Data lengkap informasi film mengambil dari API (Application Programming Interface) yang disediakan oleh TMDB. API adalah seperangkat aturan dan protokol yang memungkinkan berbagai perangkat lunak untuk saling berkomunikasi dan bertukar data secara terstandar[10]. API bertindak sebagai jembatan antara sistem, memungkinkan aplikasi mengakses layanan atau data dari aplikasi lain secara efisien tanpa harus memahami detail implementasi internal[10].

3.2 *Preprocesssing* Data

Tahap *preprocessing* atau pra-pemrosesan adalah proses menyiapkan data mentah untuk dianalisis dan digunakan dalam sistem aplikasi [11], [12]. Tahapan *preprocessing* data ini bermaksud untuk memperoleh sebuah daftar kata-kata unik yang merupakan penggalan dari judul film. *Preprocessing* data melalui dua tahap pengolahan yaitu *tokenization* dan *filtering.*

a. Tahap *Tokenization*, proses ini memecah untaian teks menjadi bagian-bagian kecil yang berupa kata [11], [13]. Pada Tabel 1 dibawah ini merupakan contoh proses *tokenization* untuk memecah judul fim menjadi per kata.

Tabel 1. Proses *Tokenization*

Teks	Token
The Avengers	‘the’, ‘avengers’
The Amazing Spider-man	‘the’,’amazing’,’spider-man’

b. Tahap *Filtering*, tahap ini bertujuan untuk mengurangi *noise* dengan membuang token-token yang mempunyai nilai sama [13]. Pada Tabel 2 dibawah ini merupakan contoh proses *filtering* dengan membuang kata-kata duplikat dari penggalan judul film sebelumnya.

Tabel 2. Proses *Filtering*

Sebelum	Sesudah
‘the’	‘the’
‘avengers’ ‘the’ ‘amazing’ ‘spider-man’	‘avengers’ ‘amazing’ ‘spider-man’

3.3 Langkah-langkah Menghitung Nilai Levenshtein Distance

Setelah melalui tahapan *preprocessing*, selanjutnya adalah mencari jarak antara penggalan kata *keyword* dengan penggalan kata judul film dari dataset. Berikut adalah langkah-langkah menghitung menghitung jarak antara dua kata ‘AVENGR’ dengan ‘AVENGERS’ :

1. Membuat matriks inisialisasi. Ukuran matriks adalah (panjang 'AVENGR' + 1) × (panjang 'AVENGERS' + 1). Kemudian isi baris pertama pertama (dari kiri ke kanan) diisi dengan 0 hingga panjang string kedua ('AVENGERS'). Kolom pertama (dari atas ke bawah) diisi dengan 0 hingga panjang string pertama ('AVENGR').

		A	V	E	N	G	E	R	S
	0	1	2	3	4	5	6	7	8
A	1								
V	2								
E	3								
N	4								
G	5								
R	6								

Gambar 1. Matriks inisiasi

2. Mengisi nilai kolom dan baris matriks dengan membandingan persamaan huruf baris dengan huruf kolom, kemudian hitung dengan aturan sebagai berikut:
   a. Jika kedua huruf sama, ambil nilai diagonal kiri-atas.

$$d[i,j] = d[i-1, j-1] \quad (1)$$

b. Jika kedua huruf berbeda, ambil nilai minimal pada sisi kiri, sisi atas, dan sisi diagonal kiri-atas kemudian tambahkan 1.

$$d[i,j] = 1 + \min(d[i-1,j], d[i,j-1], d[i-1,j-1]) \quad (2)$$

		A	V	E	N	G	E	R	S
	0	1	2	3	4	5	6	7	8
A	1	0							
V	2								
E	3								
N	4								
G	5								
R	6								

Gambar 2. Pengisian matriks

3. Setelah semua kolom matriks terisi, jarak kedua string didapatkan dari nilai ujung diagonal kanan bawah pada matrik, yaitu 2. Ini berarti dibutuhkan 2 operasi *substitute, add,* atau *delete.*

		A	V	E	N	G	E	R	S
	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
V	2	1	0	1	2	3	4	5	6
E	3	2	1	0	1	2	2	3	4
N	4	3	2	1	0	1	2	3	4
G	5	4	3	2	1	0	1	2	3
R	6	5	4	3	2	1	1	1	2

Gambar 3. Matriks levenshtein distance

3.4 Alur Kerja Aplikasi

Berdasarkan alur kerja aplikasi diatas, sistem dimulai dari pengguna meninputkan sebuah kata atau beberapa kata pada kolom pencarian di dalam aplikasi. Dari inputan tersebut dipenggal per-kata sehingga menjadi sebuah *input wordlist.* Dilanjutkan dengan melakukan perulangan untuk mem-*filter* dataset *film wordlist.* Proses ini akan memangkas dataset dari awalnya dari 100.000 judul film menjadi beberapa ratus *wordlist* saja. Tahap selanjutnya adalah melakukan perhitungan jarak Levenshtein Distance pada setiap kata yang ada pada *wordlist* dengan kata inputan pengguna. Diurutkan dengan jarak Levenshtein Distance terkecil ada pada susunan paling atas. Hasil akhir perulangan ini adalah sebuah kata atau rangkaian kata dari penggalan judul film yang paling mendekati dengan inputan pengguna. Tahap terakhir adalah pemanggilan API pencarian film dengan *keyword* tersebut, sehingga mendapatkan hasil informasi film yang mungkin diharapkan oleh pengguna aplikasi.

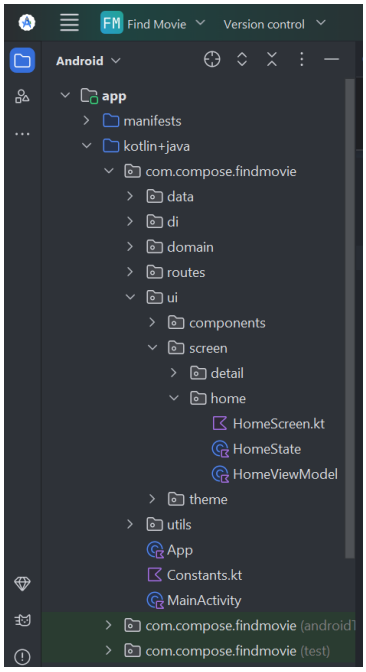
Gambar 4. Alur kerja aplikasi

4. HASIL DAN PEMBAHASAN

4.1. Arsitektur Pengembangan Aplikasi

Pengembangan aplikasi android pada penelitian ini menggunakan arsitektur Model-View-ViewModel (MVVM). MVVM merupakan salah satu arsitektur populer untuk pengembangan aplikasi android yang menggunakan pola pemisahan antara logika bisnis dan antarmuka pengguna[14]. Projek aplikasi ini dibagi menjadi 3 lapisan, yaitu Model, View, dan ViewModel[15].

a. Lapisan Model, lapisan ini berisikan model *entity* adyang digunakan pada logika bisnis.
b. Lapisan View, lapisan ini berisikan UI atau antarmuka pengguna.
c. Lapisan ViewModel, ini merupakan lapisan yang bertugas untuk membuat interaksi antara Lapisan Model dengan Lapisan View.

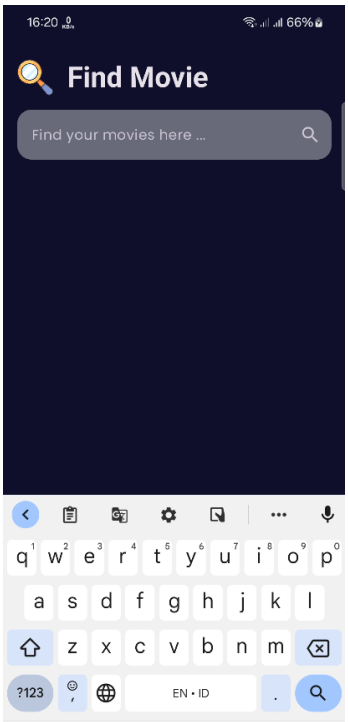


Gambar 5. Susunan *package kode* program aplikasi

Susunan modul dan *package* kode program aplikasi Find Movie dapat dilihat pada Gambar 5.

4.2. Halaman Beranda

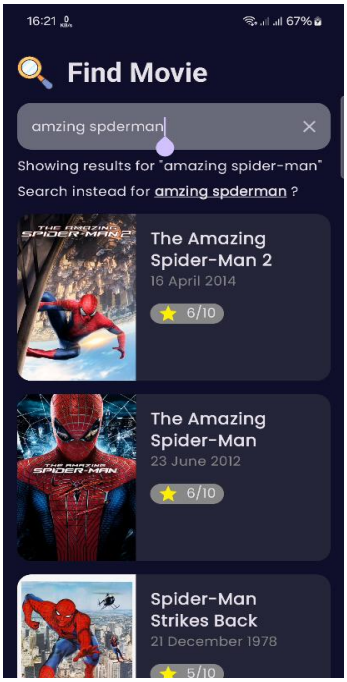
Halaman awal pengguna saat membuka aplikasi Find Movie, dapat dilihat pada Gambar 6.



Gambar 6. Halaman beranda

4.3. Pengguna Melakukan Pencarian Judul Film Dengan *Autocorrect*

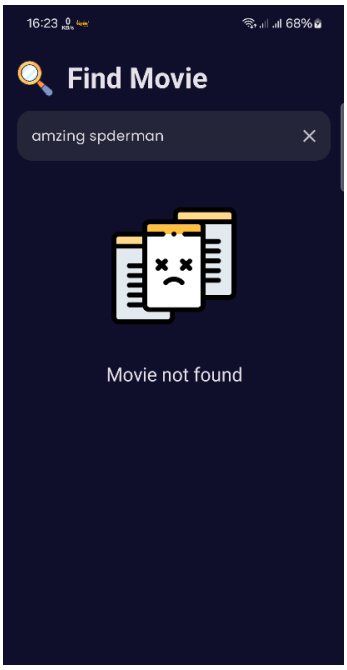
Pengguna dapat melakukan pencarian judul film dengan mengetikkan *keyword* pada kolom seperti pada Gambar 7.



Gambar 7. Pencarian dengan a*utocorrection*

4.4. Pengguna Melakukan Pencarian Judul Film Tanpa *Autocorrection*

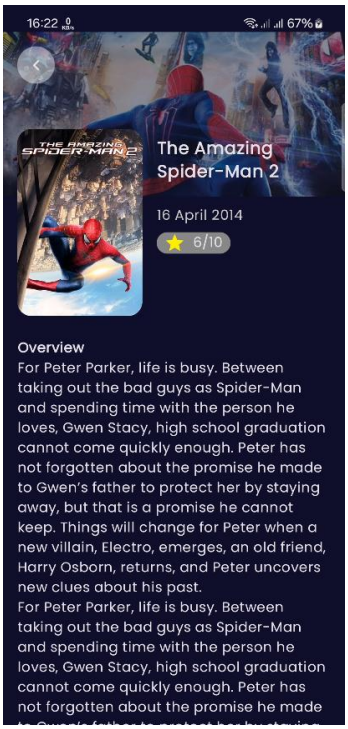
Secara otomatis sistem aplikasi akan melakukan pencarian dengan *autocorrection keyword*. Untuk melakukan pencarian tanpa menggunakan *autocorrection*, pengguna dapat memilih 'Search instead for …' seperti pada Gambar 8.



Gambar 8. Pencarian tanpa a*utocorrection*

4.5. Halaman Detail Informasi Film

Pengguna aplikasi dapat melihat informasi detail film pada hasil pencarian seperti pada Gambar 9.



Gambar 8. Halaman detail informasi film

4.6. Pengujian Akurasi Koreksi

Tujuan pengujian Black Box ini adalah untuk mengukur akurasi fitur *autocorrection* pada pencarian judul film di dalam aplikasi. Diharapkan dapat mengoreksi kesalahan ketik dan menghasilkan rekomendasi judul film yang sesuai dengan ekspetasi pengguna. Hasil pengujian dapat dilihat pada Tabel 3.

Tabel 3. Hasil Pengujian

No	Keyword	Ekspetasi	Hasil	Ket.
1.	dedpol	Deadpool	Deadpool	Sesuai
2.	spdermn	Spiderman	Spiderman	Sesuai
3.	avngr	Avenger	Avenger	Sesuai
4.	sperman	Superman	Superman	Sesuai
5.	betman	Batman	Batman	Sesuai
6.	insepsion	Inception	Inception	Sesuai
7.	infiniti war	Infinity War	Infinity War	Sesuai
8.	interselar	Interstellar	Interstellar	Sesuai
9.	pirats caribian	Pirates Caribbean	Pirates Caribbean	Sesuai
10.	ultrmn	Ultraman	Ultron	Tidak Sesuai
11.	ultrman	Ultraman	Ultraman	Sesuai
12.	dedpol and wofrine	Deadpool and Wolverine	Deadpool and Wolverine	Sesuai
13.	kapten america	Captain America	Kapitan America	Tidak Sesuai
14.	captaen americn	Captain America	Captain America	Sesuai
15.	iside out	Inside Out	Inside Out	Sesuai

Berdasarkan tabel pengujian diatas, dihitung presentase akurasi dengan rumus sebagai berikut :

$$Akurasi(\%) = \left(\frac{Jml.\,hasil\ yang\ sesuai}{Total\ pengujian}\right) x\ 100 \quad (3)$$

Sehingga perhitungan akurasi hasil pengujian adalah sebagai berikut :

$$Akurasi(\%) = \left(\frac{13}{15}\right) x\ 100$$
$$Akurasi(\%) = 86{,}67 \quad (4)$$

Maka didapatkan hasil pengujian aplikasi dengan akurasi *autorection* yang sesuai dengan ekspetasi pengguna adalah sebesar 86,67%.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil mengimplentasikan algoritma Levenshtein Distance sebagai fitur *autocorrection* pada aplikasi Find Movie. Aplikasi sudah secara otomatis memperbaiki *keyword* masukkan pengguna sebelum memanggil API pencarian film. Kemudian berdasarkan hasil pengujian, akurasi *autocorrection* yang didapatkan adalah sebesar 86,67% sesuai dengan ekspetasi pengguna. Hal ini membuktikan bahwa algoritma Levenshtein Distance efektif dalam mengenali dan memperbaiki kesalahan ketik *(typo)* pengguna, sehingga meningkatkan pengalaman pencarian. Sebagai saran, implementasi algoritma ini dapat ditingkatkan lebih lanjut dengan mengintegrasikan teknik pendukung pembobotan. Seperti dalam konteks pencarian film, dapat ditambahkan pembobotan pada popularitas film sehingga dapat meningkatkan akurasi ekspetasi pengguna.

DAFTAR PUSTAKA

[1] I. G. Anugrah, "Penerapan Metode N-Gram dan Cosine Similarity Dalam Pencarian Pada Repositori Artikel Jurnal Publikasi," *Building of Informatics, Technology and Science (BITS)*, vol. 3, no. 3, Art. no. 3, Dec. 2021, doi: 10.47065/bits.v3i3.1058.

[2] A. Setiawan and Saprudin, "Implementasi Sistem Pendeteksian Kemiripan Judul Skripsi Dengan Algoritma Levenshtein Distance Pada Perpustakaan Universitas Pamulang," *LOGIC : Jurnal Ilmu Komputer dan Pendidikan*, vol. 2, no. 1, Art. no. 1, Dec. 2023.

[3] Halimah Tus Sadiah, Lia Dahlia Iryani, Tjut Awaliyah Zuraiyah, Yuli Wahyuni, and Cantika Zaddana, "Implementation of Levenshtein Distance Algorithm for Product Search Query Suggestions on Koro Pedang Edutourism E-Commerce," *ARASET*, vol. 42, no. 2, pp. 188–196, Apr. 2024, doi: 10.37934/araset.42.2.188196.

[4] H. T. Sadiah, M. S. N. Ishlah, and N. N. Rokhmah, "Autocorrect pada Modul Pencarian Drugs e-Dictionary Menggunakan Algoritma Levenshtein Distance," *Vol* ., no. 1, 2020.

[5] H.-T. Duong and T.-A. Nguyen-Thi, "A review: preprocessing techniques and data augmentation for sentiment analysis," *Computational Social Networks*, vol. 8, no. 1, p. 1, Jan. 2021, doi: 10.1186/s40649-020-00080-x.

[6] Teknik Informatika STIKI Malang, N. N. Syarafina, J. F. Palandi, and Teknik Informatika STIKI Malang, "Designing a word recommendation application using the Levenshtein Distance algorithm," *MATRIX*, vol. 11, no. 2, pp. 63–70, Jul. 2021, doi: 10.31940/matrix.v11i2.2419.

[7] E. Suharyanto, "PERANCANGAN APLIKASI PENGENALAN BUDAYA NUSANTARA BERBASIS ANDROID DENGAN METODE RAD," *Jurnal Ilmu Komputer*, vol. 5, no. 1, pp. 30–30, May 2022.

[8] A. Fahrezi, F. N. Salam, G. M. Ibrahim, R. R. Syaiful, and A. Saifudin, "Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT. AINO Indonesia," *LOGIC : Jurnal Ilmu Komputer dan Pendidikan*, vol. 1, no. 1, Art. no. 1, Dec. 2022.

[9] M. Mintarsih, "Pengujian Black Box Dengan Teknik Transition Pada Sistem Informasi Perpustakaan Berbasis Web Dengan Metode Waterfall Pada SMC Foundation," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 5, no. 1, Art. no. 1, Feb. 2023, doi: 10.47233/jteksis.v5i1.727.

[10] K. Gowell and Suprihadi, "Perancangan Web Service REST API Menggunakan PHP dan Framework Laravel di Tenta Tour Salatiga," *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 8, no. 1, Art. no. 1, Jan. 2024, doi: 10.35870/jtik.v8i1.1269.

[11] Syahril Dwi Prasetyo, Shofa Shofiah Hilabi, and Fitri Nurapriani, "Analisis Sentimen Relokasi Ibukota Nusantara Menggunakan Algoritma Naïve Bayes dan KNN," *komtekinfo*, pp. 1–7, Jan. 2023, doi: 10.35134/komtekinfo.v10i1.330.

[12] M. Lihawa, A. D. Hartanto, N. Norhikmah, D. Prabowo, I. N. Fajri, and W. Widayani, "Implementation of the Levenshtein Distance Algorithm and the Regular Search Expression Method for Detecting Typors in Javascript," *Sistemasi: Jurnal Sistem Informasi*, vol. 12, no. 2, Art. no. 2, May 2023.

[13] "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis," ar5iv. Accessed: Nov. 01, 2024. [Online]. Available: https://ar5iv.labs.arxiv.org/html/1707.01780

[14] C. Diantoni, O. Komarudin, and A. Rizal, "ARSITEKTUR MVVM DAN FRAMEWORK JETPACK COMPOSE PADA PENGEMBANGAN APLIKASI ANDROID:," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 3, Art. no. 3, May 2024, doi: 10.36040/jati.v8i3.9638.

[15] I. P. R. P. Putra and H. Tolle, "Pengembangan Aplikasi Pembelajaran Bahasa Bali berbasis Android menggunakan MVVM Architecture dan Jetpack Compose," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 7, no. 5, Art. no. 5, Aug. 2023.