Pengaplikasian *Spellchecker* Pada Aplikasi Kamus Bahasa Bugis Dengan Metode *Levenshtein*

SKRIPSI

AHMAD WILDAN DZAKKI ADAM 105841101820

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS M UHAMMADIYAH MAKASSAR

2024

Pengaplikasian *Spellchecker* Pada Aplikasi Kamus Bahasa Bugis Dengan Metode *Levenshtein*

Diajukan sebagai Salah Satu Syarat untuk Menyusun Skripsi

Program Studi Informatika

Disusun dan Diajulan Oleh :

AHMAD WILDAN DZAKKI ADAM 105841101820

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS M UHAMMADIYAH MAKASSAR

2024

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

PENGESAHAN

Skripsi atas nama AHMAD WILDAN DZAKKI ADAM dengan nomor induk Mahasiswa 105841101820, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 300/05/A.5-II/VIII/46/2024, sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Sabtu tanggal 29 Agustus 2024.

Makassar, 24 Safar 1446 H / 29 Agustus 2024 M

Panitia Ujian :

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Dr. Ir. H. Abd. Rakhim Nanda, ST., MT., IPU. : .....
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, ST., MT. : .....
2. Penguji
   a. Ketua : Dr. Ir. Zahir Zainuddin, M.Sc. .....
   b. Sekertaris : Lukman, S. Kom., M. T .....
3. Anggota : 1. Rizki Yusliana Bakti ST., MT. : .....
   2. Lukman Anas, S. Kom., MT. : .....
   3. Fahrim Irhamna Rahman S. Kom., MT. : .....

Mengetahui :

Pembimbing I	Pembimbing II
Titin Wahyuni, S. Pd., MT.	**Muhyiddin A. M Hayat, .Kom., MT.**

Dekan

Dr. Ir. Hj. Nurnawaty, ST., MT., IPM.
NBM : 795 108

UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK
GEDUNG MENARA IQRA LT. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: http://teknik.unismuh.makassar.ac.id

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **PENGAPLIKASIAN SPELL CHECKER PADA APLIKASI KAMUS BAHASA BUGIS DEGAN METODE LEVENSHTEIN**

Nama : AHMAD WILDAN DZAKKI ADAM
Stambuk : 105841101820

Makassar, 30 Agustus 2024

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing;

Pembimbing I	Pembimbing II
Titin Wahyuni, S. Pd., MT.	**Muhyiddin A M Hayat, S.Kom., MT.**

Mengetahui,
Ketua Program Studi Informatika

Muhyiddin A M Hayat, S.Kom., MT.
NBM 1504 577

ABSTRAK

AHMAD WILDAN DZAKKI ADAM. Pengaplikasian *Spellchecker* Pada Aplikasi Kamus Bahasa Bugis Dengan Metode *Levenshtein* (dibimbing oleh Titin Wahyuni, S. Pd., MT. dan Muhyiddin A M Hayat S.Kom., M.T).

Penelitian ini bertujuan untuk mengimplementasikan metode Levenshtein sebagai spell checker pada aplikasi kamus Bahasa Bugis. Metode ini digunakan untuk mengukur jarak antara dua string, memungkinkan pendeteksian dan koreksi kesalahan ejaan dengan menghitung perbedaan karakter antara kata yang dimasukkan oleh pengguna dan kata yang ada dalam kamus. Aplikasi ini dirancang untuk membantu pengguna menemukan kata yang tepat meskipun terdapat kesalahan pengetikan. Metodologi yang digunakan meliputi penerapan algoritma Levenshtein pada sistem spell checker dalam aplikasi kamus Bahasa Bugis, dengan evaluasi kinerja berdasarkan akurasi deteksi kesalahan ejaan dan efektivitas koreksi yang dihasilkan. Hasil pengujian menunjukkan bahwa metode Levenshtein mencapai tingkat akurasi yang tinggi, dengan rata-rata akurasi berkisar antara 80% hingga 90% dalam mendeteksi dan memperbaiki kesalahan ejaan. Implementasi ini diharapkan dapat mempermudah pelestarian dan pembelajaran Bahasa Bugis di era modern.

Kata Kunci: Bahasa Bugis, Spell Checker, Metode Levenshtein, Aplikasi Kamus, Koreksi Ejaan.

Abstrac

AHMAD WILDAN DZAKKI ADAM. Pengaplikasian *Spellchecker* Pada Aplikasi Kamus Bahasa Bugis Dengan Metode *Levenshtein* (dibimbing oleh Titin Wahyuni, S. Pd., MT. dan Muhyiddin A M Hayat S.Kom., M.T).

This research aims to implement the Levenshtein method as a spell checker in a Bugis language dictionary application. This method is used to measure the distance between two strings, allowing for the detection and correction of spelling errors by calculating the difference in characters between the user-entered word and the words in the dictionary. The application is designed to help users find the correct word even with typing errors. The methodology involves applying the Levenshtein algorithm to the spell checker system within the Bugis language dictionary application, with performance evaluation based on spelling error detection accuracy and the effectiveness of the corrections produced. Testing results indicate that the Levenshtein method achieves a high accuracy rate, with an average accuracy ranging from 80% to 90% in detecting and correcting spelling errors. This implementation is expected to facilitate the preservation and learning of the Bugis language in the modern era.

Keywords: Bugis Language, Spell Checker, Levenshtein Method, Dictionary Application, Spelling Correction.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabaraktuh

Dengan penuh rasa syukur ke hadirat Allah Subhanallahu Wa Ta'ala, atas nikmat iman, Islam, dan kesehatan yang senantiasa terlimpahkan. sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul "**Pengaplikasian *Spellchecker* Pada Aplikasi Kamus Bahasa Bugis Dengan Metode *Levenshtein***". Shalawat serta salam teriring kepada junjungan Nabi Muhammad SAW, pembawa rahmat bagi semesta alam. yang telah membawa kita dari Zaman jahiliah menuju Zaman yang serba modern seperti saat ini.

Dalam penyusunan berbagai hambatan dan keterbatasan dihadapi oleh penulis mulai dari tahap persiapan sampai dengan penyelesaian tulisan, namun berkat bantuan bimbingan dan kerja sama berbagai pihak, hambatan dan kesulitan tersebut dapat teratasi.

Ucapan terima kasih yang tidak terhingga penulis sampaikan kepada semua pihak yang telah membantu dan memberikan dukungan dalam penyusunan Laporan Tugas Akhir ini, khususnya:

1. Allah SWT, yang telah memberikan petunjuk, kekuatan, dan rahmat-Nya selama proses penulisan.
2. Kepada kedua orang tua tercinta saya, Bapak **Syamsul Adam** dan Ibu **Nuraeni**, persembahan terima kasih yang tak terhingga atas segala pengorbanan, kasih sayang, dan bimbingan, yang telah diberikan. Tanpa do'a dan dukungan kalian, saya tidak akan bisa mencapai titik ini. Kehadiran dan cinta kalian adalah sumber inspirasi dan kekuatan bagi saya.
3. Ibu **Dr.Ir.Hj Nurnawati, S.T., M.T., I.P.M**, selaku Dekan Fakultas Teknik.
4. Bapak **Muh. Syafaat S Kuba, S.T., M.T**, selaku Wakil Dekan Fakultas Teknik.
5. Bapak **Muhyiddin A M Hayat S.Kom., M.T**, selaku Ketua Prodi Informatika.
6. Ibu **Titin Wahyuni, S. Pd., MT.** , selaku Dosen Pembimbing 1 proposal.
7. Pak **Muhyiddin A M Hayat S.Kom., M.T** selaku Dosen Pembimbing 2 Proposal.

8. Dosen dan Staf Fakultas Teknik Universitas Muhammadiyah Makassar.
9. Teman-teman Khususnya Angkatan 2020 Fakultas Teknik, Universitas Muhammadiyah Makassar, terima kasih atas dukungan dan doanya.
10. Teman-teman kelas A angkatan 2020 Program Studi Informatika Universitas Muhammadiyah Makassar

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan laporan ini di masa depan. Harapan penulis, semoga Laporan Tugas Akhir ini dapat memberikan manfaat bagi penyandang disabilitas tunanetra dalam meningkatkan kemandirian dan kualitas hidup mereka. Akhir kata, penulis mohon maaf atas segala kekurangan dan kekhilafan yang terdapat dalam Laporan Tugas Akhir ini.

Billahi fisabililhaq, fastabiqul khairat.
Wassalamualaikum Wr.Wb.

Makassar, Mei 2024
Penulis,

AHMAD WILDAN DZAKKI ADAM

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Application	Sebuah subkelas dari perangkat lunak komputer yang memanfaatkan kemampuan komputer untuk membantu pengguna melakukan tugas-tugas tertentu.
Algorithm	Langkah-langkah atau prosedur terdefinisi dengan jelas yang digunakan untuk memecahkan masalah atau melakukan perhitungan.
Spell Checker	Alat atau program yang digunakan untuk mendeteksi dan memperbaiki kesalahan ejaan dalam teks.
Edit Distance	Metode untuk mengukur seberapa mirip atau berbeda dua string dengan menghitung jumlah minimum operasi yang diperlukan untuk mengubah satu string menjadi yang lain.
Data	Kumpulan fakta, statistik, atau informasi yang digunakan untuk analisis atau pemrosesan.
String	Serangkaian karakter yang dianggap sebagai satu unit, seperti kata atau kalimat dalam pemrograman komputer.
Character	Huruf, angka, tanda baca, atau simbol lain yang digunakan dalam teks.
Dictionary	Kumpulan kata-kata dari suatu bahasa yang disusun berdasarkan urutan abjad, biasanya dilengkapi dengan arti,

	pelafalan, atau informasi lainnya.
Input	Data atau informasi yang dimasukkan ke dalam sistem atau program untuk diproses.
Output	Data atau informasi yang dihasilkan oleh sistem atau program setelah diproses.
Distance	Ukuran seberapa berbeda atau mirip dua entitas, seperti string, satu sama lain.
Levenshtein	Sebuah algoritma yang digunakan untuk mengukur jarak antara dua string dengan menghitung jumlah minimum operasi penyisipan, penghapusan, atau substitusi karakter yang diperlukan untuk mengubah satu string menjadi string lainnya.

BAB I
PENDAHULUAN

A. Latar Belakang

Bahasa adalah cara penting untuk berkomunikasi dalam kehidupan sehari-hari (S. Fakhiratunnisa, dkk, 2022). Bahasa Indonesia sangat penting untuk interaksi sosial dan pendidikan di Indonesia. Namun, banyaknya kosakata dan struktur bahasa yang kompleks seringkali menjadi tantangan bagi siswa untuk mempelajarinya, terutama bagi siswa yang baru mengenal lingkungan berbahasa Indonesia. Bahasa Indonesia adalah bahasa negara, menurut UUD 1945 (N.F.N. Asrif, 2019). Pasal 33 Undang-Undang Sistem Pendidikan Nasional Nomor 20 Tahun 2003 juga mengatur penggunaan bahasa Indonesia sebagai bahasa pengantar. Bahasa Indonesia ditetapkan sebagai bahasa resmi dalam pendidikan masyarakat oleh undang-undang. Undang-undang Pemerintahan Daerah Republik Nomor 24 tentang Bendera, Bahasa, dan Lambang Negara Indonesia serta Lagu Kebangsaan Indonesia Tahun 2009 memperkuat gagasan bahwa bahasa daerah dapat digunakan sebagai bahasa pengantar pada tahap awal pendidikan, apabila diperlukan untuk memberikan pengetahuan dan keterampilan tertentu. (I.R Maharani, A.M. kepada Bukhari, dan L. Putriyanti, 2023). Menurut undang-undang, "Bahasa Indonesia wajib digunakan sebagai bahasa pengantar dalam pendidikan nasional" (Felicia, 2019).

Orang-orang di Sulawesi Selatan, Indonesia, berbicara menggunakan bahasa Bugis. Kamus Bugis telah menjadi salah satu alat yang sangat bermanfaat bagi masyarakat untuk memahami dan menggunakan bahasa ini di era modern. Namun, beberapa masalah yang sering dihadapi saat menggunakan aplikasi kamus termasuk kesalahan ejaan saat mencari kata-kata dalam kamus. Kesalahan ejaan ini dapat menyebabkan hasil pencarian yang tidak sesuai dengan harapan, membuat pengguna tidak dapat mendapatkan informasi yang tepat. Penggunaan teknologi telah memungkinkan aplikasi kamus untuk memproses dan memperbaiki kesalahan ejaan. Metode

Levenshtein, yang berbasis pada algoritma yang dapat menghitung jarak antara dua kata, digunakan dalam beberapa aplikasi kamus untuk memperbaiki kesalahan ejaan dan meningkatkan akurasi hasil pencarian.

Algoritma *Levenshtein Distance* adalah algoritma yang dikembangkan oleh Vladimir Levenshtein pada tahun 1965.Algoritma ini menghitung jarak antara kata yang dimasuukkan oleh pengguna atau *user* dengan string pada *database* dengan menghitung perbedaan antara kedua string dalam matriks. Kemudian, operasi perubahan digunakan untuk mengubah string A menjadi string B, yang melibatkan proses penyisi. (Yuyun Nia Daniati, 2022).

Namun, masih ada beberapa masalah yang perlu diselesaikan saat menggunakan metode *Levenshtein* dalam kamus Bahasa Bugis. Salah satu masalah adalah bagaimana menggunakannya dalam kamus Bahasa Bugis sehingga dapat mendeteksi dan memperbaiki kesalahan ejaan. Selain itu, perlu diketahui seberapa efektif metode ini dalam menemukan dan memperbaiki kesalahan ejaan dalam pencarian kamus Bahasa Bugis. Oleh karena itu, tujuan dari penelitian ini adalah untuk menerapkan metode *Levenshtein* dalam *Spell Check* pada aplikasi kamus Bahasa Bugis dan untuk mengetahui seberapa efektif metode ini dalam menemukan dan memperbaiki kesalahan ejaan dalam pencarian kamus Bahasa Bugis. Akibatnya, penelitian ini diharapkan dapat membantu dalam pengembangan aplikasi kamus Bahasa Bugis yang lebih efisien dan akurat untuk memperbaiki dan memproses kesalahan ejaan.

B. Rumusan Masalah

1. Bagaimana mengimplementasikan metode *Levenshtein* dalam *Spell Check* pada aplikasi kamus Bahasa Bugis?
2. Seberapa efektif metode *Levenshtein* dalam mendeteksi dan memperbaiki kesalahan ejaan dalam pencarian pada kamus Bahasa Bugis?

C. Tujuan Penelitian

1. Untuk mengimplementasikan metode *Levenshtein* dalam *Spell Check* pada aplikasi kamus Bahasa Bugis.
2. Untuk mengetahui keefektifan metode *Levenshtein* dalam mendeteksi dan memperbaiki kesalahan ejaan dalam pencarian pada kamus Bahasa Bugis.

D. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

a. Bagi penulis, penelitian ini akan meningkatkan kemampuan penulis dalam bidang pemrograman dan pengolahan bahasa alami. Penulis juga akan mendapatkan pemahaman yang lebih dalam tentang algoritma *Levenshtein* yang digunakan untuk memeriksa ejaan.
b. Bagi Universitas, Hasil penelitian ini dapat meningkatkan reputasi Universitas sebagai Lembaga yang mendukung inovasi dan pelestarian Bahasa Bugis dan juga dapat menjadi sumber referensi untuk penelitian selanjutnya yang berkaitan dengan teknologi informasi dan bahasa.
c. Bagi pembaca, Pengaplikasian ini akan membantu dalam pelestarian Bahasa Bugis dengan menyediakan alat yang memudahkan pembelajaran dan penggunaan bahasa.

E. Ruang Lingkup Penelitian

Penelitian ini dibuatkan ruang lingkup penelitian agar penelitian tetap pada jalur yang telah ditentukan diantaranya yaitu:

1. Data yang diambil berasal dari Balai Bahasa yang terdapat di dalam aplikasi Bahasa Bugis.
2. Memimplementasikan algoritma *Levenshtein* untuk *Spell Checker.*

F. Sistematika Penulisan

Secara garis besar penulisan laporan tugas akhir ini terbagi menjadi beberapa bab yang tersusun yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah, rumusan masalah,Batasan masalah,tujuan, manfaat dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori-teori yang melandasi penulisan dalam melaksanakan skripsi.

BAB III METODE PENELITIAN

Membahas tentang metode penelitian dan alat yang digunakan untuk pembuatan sistem.

BAB IV HASIL PENELITIAN

Pada bab ini menjelaskan hasil dari penelitian yang sudah di lakukan sebelumnya, pada bab inilah di jelaskan hasil penelitian dan pengujian.

BAB V PENUTUP

Pada bab ini dijelaskan kesimpulan yang di hasilkan dari penelitian yang telah di lakukan

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. Aplikasi

Perangkat lunak, juga dikenal sebagai "aplikasi", adalah subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk mengontrol pengguna untuk melakukan tugas yang mereka inginkan. Program komputer yang dimaksudkan untuk membantu pengguna melakukan tugas tertentu disebut aplikasi (Hermawan, 2019). Aplikasi adalah set perintah yang dilakukan komputer. Program adalah set petunjuk yang akan digunakan oleh *software* pemroses. Program ini mengontrol logika sistem komputer. Program ini mengatur semua operasi pemrosesan. Program terdiri dari komponen logika manusia yang telah diterjemahkan ke dalam bahasa mesin yang dapat digunakan. Program ini dibuat untuk melakukan hal yang sama seperti aplikasi lainnya.

2. Bahasa

Bahasa adalah alat pertama yang digunakan manusia untuk berkomunikasi, baik secara individual maupun kelompok sosial. Secara individual, bahasa adalah alat untuk mengungkapkan pikiran, ide, dan keinginan seseorang, dan secara kelompok atau sosial, bahasa adalah alat untuk berinteraksi dengan lingkungannya. Menurut Tarigan (Heryati, 2022), keterampilan berbahasa seseorang bergantung pada jumlah dan kualitas kosa kata yang dimilikinya, jadi semakin banyak kosa kata yang dimiliki seseorang, semakin baik keterampilan berbahasanya. Ekspresi mengandung elemen segmental dan suprasegmental, baik lisan maupun kinesik, sehingga kalimat dapat menyampaikan berbagai pesan dengan cara yang berbeda. Bahasa adalah sistem komunikasi yang terdiri dari simbol-simbol, baik berupa suara, tulisan, gerakan, atau tanda, yang digunakan oleh manusia untuk menyampaikan pikiran, perasaan, dan informasi. Bahasa memiliki beberapa karakteristik utama, yaitu:

1. **Arbitrariness (Kesesuaian Acak)**: Tidak ada hubungan intrinsik antara kata dan objek atau konsep yang diwakilinya. Misalnya, kata "meja" dalam bahasa Indonesia tidak memiliki hubungan alamiah dengan objek meja itu sendiri.
2. **Productivity (Produktivitas)**: Bahasa memungkinkan pembicara untuk menciptakan dan memahami kalimat-kalimat baru yang belum pernah mereka dengar sebelumnya.
3. **Cultural Transmission (Transmisi Budaya)**: Bahasa dipelajari melalui interaksi sosial dan diwariskan dari satu generasi ke generasi berikutnya dalam suatu masyarakat.
4. **Discreteness (Diskrit)**: Bahasa terdiri dari unit-unit diskrit seperti fonem (bunyi bahasa) yang dapat digabungkan untuk membentuk kata dan kalimat.
5. **Displacement (Pemindahan)**: Bahasa memungkinkan pembicara untuk membicarakan hal-hal yang tidak ada di hadapan mereka, seperti peristiwa masa lalu, masa depan, atau sesuatu yang imajiner.

Bahasa juga mencakup aspek-aspek fonetik (bunyi), fonologi (sistem bunyi), morfologi (struktur kata), sintaksis (struktur kalimat), semantik (makna), dan pragmatik (penggunaan dalam konteks). Setiap bahasa memiliki aturan-aturan yang mengatur cara penggunaan simbol-simbol tersebut untuk membentuk komunikasi yang bermakna.

Dalam konteks yang lebih luas, bahasa tidak hanya merujuk pada bahasa lisan dan tulisan, tetapi juga mencakup bahasa isyarat yang digunakan oleh komunitas tuli, serta berbagai bentuk komunikasi nonverbal lainnya yang digunakan manusia untuk berinteraksi satu sama lain.

3. Kamus

Kamus, yang biasanya disusun secara alfabetis, adalah buku referensi yang berisi daftar kata atau gabungan kata dengan keterangan tentang berbagai aspek makna dan penggunaannya dalam suatu bahasa tertentu. Kamus sangat penting bagi banyak orang, termasuk pembelajar bahasa asing. Mereka

membantu mereka belajar bahasa asing (Suryadarma & Fakhiroh, 2020).Menurut Taubah (2019), ada beberapa ahli bahasa yang berpendapat bahwa kemampuan bahasa seseorang hanya ditentukan oleh tingkat penguasaan kosakata, dan kamus ini adalah jawabannya. Menurut Harun (2019), kamus yang disusun dengan memperhatikan elemennya dapat dianggap ideal dan mudah digunakan oleh pengguna bahasa. Kamus adalah sebuah buku referensi atau sumber daya digital yang mengandung daftar kata-kata dari suatu bahasa yang disusun secara alfabetis, beserta penjelasan atau definisi, terjemahan, sinonim, antonim, atau informasi lain yang relevan tentang kata-kata tersebut. Kamus berfungsi sebagai alat untuk memahami makna kata, ejaan yang benar, penggunaan dalam konteks, dan kadang-kadang informasi tambahan seperti asal usul kata (etimologi) atau panduan pelafalan. Fungsi utama kamus adalah untuk membantu pengguna dalam memahami dan menggunakan kata-kata dengan benar serta memperkaya kosakata mereka.

4. *Spell Checker*

Spell checker, atau pemeriksa ejaan, adalah fitur yang biasanya terdapat dalam perangkat lunak pengolah kata, email, atau aplikasi penulisan lainnya. Fungsinya adalah untuk membantu pengguna dalam mengidentifikasi dan memperbaiki kesalahan ejaan dalam teks. Ketika pengguna mengetikkan kata yang salah eja, *spell checker* akan menandai kata tersebut dan menawarkan saran koreksi agar teks menjadi lebih benar secara ejaan.

Dalam pemrosesan teks digital, penggunaan *spell checker* telah meningkatkan akurasi dan efisiensi komunikasi tertulis. Algoritma biasanya digunakan oleh *spell checker* untuk membandingkan setiap kata dalam teks dengan kamus yang komprehensif. Kata ditandai sebagai potensi kesalahan ketika tidak ada entri kamus yang sesuai dengannya. Selain itu, *spell checker* yang canggih memiliki algoritma yang sensitif terhadap konteks, yang meningkatkan efektivitas perangkat lunak dengan menangani homofon dan nuansa gramatikal (Sari, A., & Putra, B. (2021). Fitur utama spellchecker meliputi:

1. **Deteksi Kesalahan Ejaan**: Mengidentifikasi kata-kata yang tidak ditemukan dalam kamus internalnya dan menandai kata-kata tersebut sebagai mungkin salah eja.
2. **Saran Perbaikan**: Menyediakan daftar kata-kata yang kemungkinan dimaksudkan oleh pengguna berdasarkan kesalahan ejaan yang terdeteksi.
3. **Otomatis Mengoreksi Kesalahan**: Dalam beberapa kasus, spellchecker dapat secara otomatis mengoreksi kesalahan ejaan yang umum.
4. **Penambahan Kata Baru**: Pengguna dapat menambahkan kata-kata yang tidak ada dalam kamus internal spellchecker, seperti nama-nama khusus atau istilah teknis, sehingga tidak dianggap sebagai kesalahan di masa depan.
5. **Pengaturan Bahasa**: Spellchecker dapat diatur untuk bekerja dalam berbagai bahasa sesuai kebutuhan pengguna.

Spellchecker sangat berguna untuk meningkatkan akurasi dan profesionalisme dalam penulisan dengan mengurangi kesalahan ejaan dan membantu pengguna menulis dengan lebih percaya diri.

5. ***Levenshtein***

"Iwan Saputera (2013) menyatakan dalam jurnal penelitian yang berjudul "Aplikasi SMS Filtering Pada *Smartphone Android* Dengan Metode *Levenshtein Distance*". Perhitungan jarak edit *Levenshtein* diciptakan oleh Vladimir Levenshtein pada tahun 1965 dan digunakan untuk menghitung jumlah perbedaan string antara dua string. Dalam jarak *Lavenshtein*, string sumber (s) adalah kemiripan dua dokumen dan string tujuan (t). Jumlah spasi yang dibutuhkan untuk penghapusan atau subtitusi untuk mengkonversi string s menjadi t adalah peningkatan dari jarak *Lavenshtein*. String sumber dapat berfungsi sebagai input, dan string target adalah masukan yang diberikan pada teks (Hakak et al., 2019). Metode Levenshtein, atau jarak Levenshtein, adalah sebuah algoritma untuk mengukur perbedaan antara dua urutan karakter. Ini sering digunakan dalam pemrosesan teks dan linguistik komputasional untuk

mengukur seberapa mirip atau berbeda dua string (kata atau kalimat) satu sama lain. Jarak Levenshtein dihitung sebagai jumlah minimum operasi yang diperlukan untuk mengubah satu string menjadi string lainnya. Operasi-operasi yang diperbolehkan adalah:

1. **Substitusi**: Mengganti satu karakter dengan karakter lain.
2. **Penghapusan**: Menghapus satu karakter.
3. **Penyisipan**: Menyisipkan satu karakter.

Jarak antara dua string ditentukan dari jumlah operasi perubahan yang diperlukan untuk mengubah string A menjadi string B. Untuk menghitung jarak antara dua kata, kita perlu menggunakan persamaan matriks berikut:

Max (i,j) if min (i,j) = 0
otherwise
Lev a, b (i,j) =
Lev a, b (i-1, j) + 1 Min Leva, b (i, j-1) + 1
Lev a, b (i-1, j-1) + 1 (a i ≠ b j)
If +0 = a(i) = b(j)

lev a,b adalah matriks lev a,b i adalah baris matriks
j adalah kolom matriks
Berikut ini merupakan tabel matriks untuk mencari nilai *levenshtein distance* antara dua *string* .

Tabel 1. Tabel Matriks

		S	L	A	M	A	T
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
E	2	1	1	2	3	4	5
L	3	2	1	2	3	4	5
A	4	3	2	1	2	3	4
M	5	4	3	2	1	2	3
A	6	5	4	3	2	1	2
T	7	6	5	4	3	2	1

Dalam hal ini, tabel matriks digunakan untuk menghitung jarak Levenshtein antara kata "selamat" dan "slamat". Dari tabel matriks ini, nilai *Levenshtein* jarak adalah satu, yang dihasilkan melalui operasi penambahan satu.

B. Penelitian Terkait

Penelitian yang dilakukan oleh Ira Zulfa, dkk (2024) yang berjudul "Sistem Pendeteksi Plagiarisme Pada Laporan Skripsi Dan Magang Mahasiswa Menggunakan Metode *Levenshtein Distance* (Studi Kasus : Fakultas Teknik Universitas Gajah Putih)" Hasil pengujian data menunjukkan bahwa metode *Levenshtein Distance* berhasil mendeteksi plagiarisme dan dapat mempercepat penerbitan dan pencetakan laporan skripsi dan magang mahasiswa karena banyaknya penjiplakan. Algoritma *Levenshtein Distance* juga dapat mengidentifikasi dan mengukur seberapa mirip dokumen laporan skripsi dan magang mahasiswa yang diunggah ke sistem.

Penelitian yang dilakukan oleh Marcel Rino Batisya (2023) yang berjudul "Penerapan algoritma *Levenshtein Distance* untuk *misspelled word* pada pencarian lagu Melayu berbasis website" menunjukkan bahwa berhasilnya penerapan algoritma *Levenshtein Distance* untuk kata-kata yang dimisspell pada situs web pencarian Lagu Melayu ditunjukkan oleh tanggapan

85 orang yang disurvei, di mana 23.53% sangat setuju dan 60% setuju bahwa algoritma tersebut dapat mengurangi kesalahan ejaan kata pada situs web pencarian Lagu Melayu.

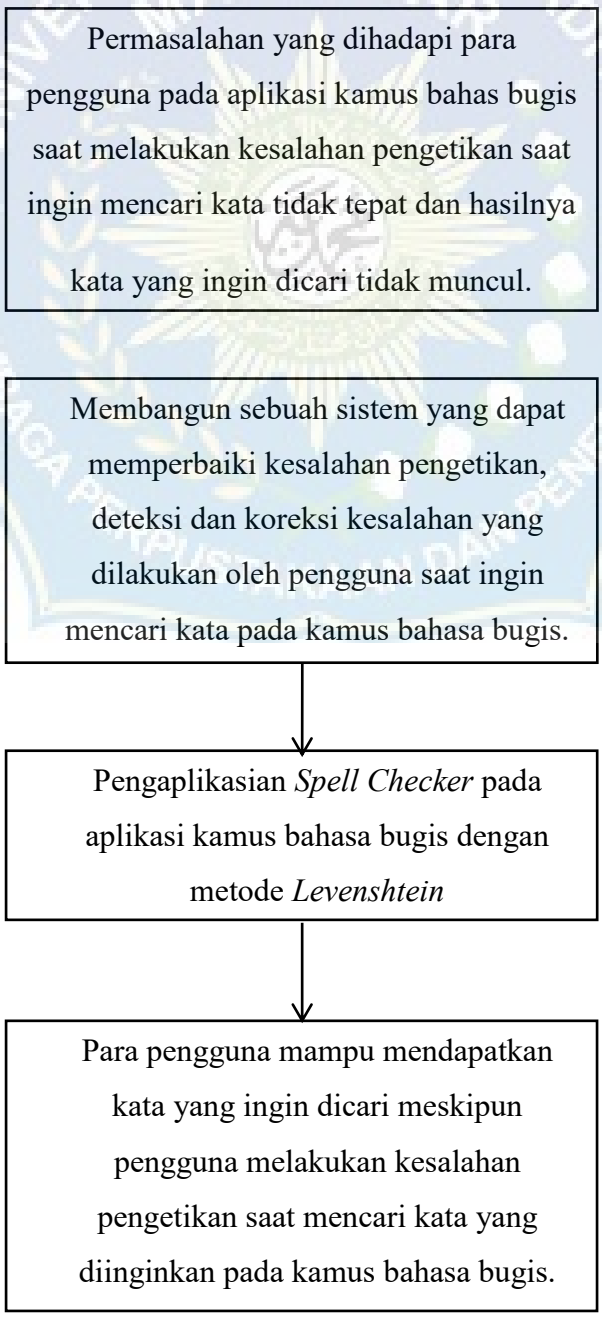
Penelitian yang dilakukan oleh Hendro Nugroho, dkk (2023) dengan judul "Penerapan Metode *Levenshtein Distance* untuk Mengukur Similaritas pada Pola Suara Burung yang Menggunakan *Discrete Cosine Transform*" Hasil pengujian data menunjukkan bahwa Dalam penelitian yang menggunakan metode *Levenshtein Distance* untuk ekstraksi fitur suara burung menggunakan DCT, jenis suara burung dibagi menjadi data T dan data S. Hasil menunjukkan bahwa jenis suara burung Kenari pada data T memiliki kemiripan 37% dan 32%, sedangkan jenis suara burung Red Lories pada data T memiliki kemiripan 32%.

Berdasarkan hasil penelitian yang dilakukan oleh Haidar Ananta Kusuma (2023) dengan judul "Rancang Bangun Sistem Cerdas *Emergency Assistant* Berbasis *Chatbot* Telegram Menggunakan Metode NLP dan *Levenshtein Distance*" menunjukkan bahwa hasil pengujian kinerja sistem cukup baik, seperti yang ditunjukkan oleh nilai ketepatan yang cukup tinggi—89.32%, nilai recall sebesar 86.26%, nilai F1-score sebesar 85.08%, dan nilai akurasi sebesar 85.32%.

Berdasarkan hasil penelitian yang dilakukan oleh Muhammad Thomas Fadhila Yahya dan Sigit Doni Ramdan (2021) yang berjudul "MENORMALISASIKAN TEKS PADA BOT SISTEM INFORMASI AKADEMIK MENGGUNAKAN ALGORITMA *DAMERAU–LEVENSHTEIN DISTANCE* DAN *PREFIX TREE* (STUDI KASUS: UNIVERSITAS TEKNOKRAT INDONESIA)" menunjukkan bawah hasil pengujian menunjukkan bahwa algoritme *Damerau–Levenshtein Distance* dengan fungsi Perhitungan Kedekatan Huruf menghasilkan nilai rata-rata ketepatan rata-rata sebesar 0,86, dengan nilai rata-rata ketepatan rata-rata tertinggi untuk kata dengan 8 karakter atau lebih di atas 0,90; algoritme tanpa fungsi Perhitungan Kedekatan Huruf menghasilkan nilai rata-rata ketepatan rata-rata sebesar 0,77. Data korpus yang berasal dari Kamus Besar Bahasa

Indonesia berjumlah 59.093 kata dan disusun menjadi 161.828 node dalam bentuk trie atau prefix tree. Waktu proses program normalisasi teks untuk memperbaiki kata dengan data korpus berbentuk prefix tree adalah 0.004 detik untuk kata dengan panjang tiga karakter dan 0.002 detik untuk kata dengan panjang sebelas karakter. Untuk kata dengan panjang lebih dari sebelas karakter, waktu proses berub bentuk menjadi lebih lama sebanyak waktu yang diperlukan untuk berub bentuk kata.

C. Kerangka Pikir



Tabel 2. Kerangka pikir

BAB III
METODE PENELITIAN

A. Tempat Dan Waktu Penelitian

1. Tempat penelitian

Penetlitian ini dilakukan dengan mengambik data pada aplikasi kamu Bahasa Bugis.

2. Waktu Penelitian

Adapun waktu pelaksaan penelitian ini dilakukan selama bulan Mei – Juli 2024.

B. Alat dan Bahan

Adapun alat dan bahan yang akan digunakan dalam penelitian ini adalah sebagai berikut:

1. Kebutuhan *Hardware* (Perangkat Keras)
   a. Laptop
   b. Hp (Redmi 9C)
2. Kebutuhan *Softaware* (Perangkat Lunak)
   a. VS Code

C. Perancangan Sistem

Untuk memudahkan proses perancangan, peneliti menggunakan flowchart untuk membuat perancangan sistem, seperti yang ditunjukkan pada gambar berikut:

Gambar 1. Flowchart Sistem

1. Mulai: Ini adalah titik di mana proses aplikasi *Spell Checker* dimulai. Ini menandakan dimulainya alur kerja aplikasi.
2. Input Kata: Pengguna dapat memasukkan kata-kata yang ingin dicek kebenarannya. Pengguna mungkin memiliki keraguan tentang ejaan kata ini atau ingin memastikan apakah ada dalam kamus bahasa Bugis.
3. Pengecekan Kata: Aplikasi akan membandingkan kata pengguna dengan kamus yang telah dimuat sebelumnya. Proses berakhir jika kata tersebut ditemukan dalam kamus; jika tidak, aplikasi akan melanjutkan ke langkah berikutnya.
4. Pengunaan Metode *Levenshtein*: Jika kata yang dimasukkan tidak ditemukan dalam kamus, aplikasi akan menggunakan metode Levenshtein untuk menghitung "jarak" yang ada antara kata yang dimasukkan dan kata-kata yang sudah ada dalam kamus. Jarak Levenshtein adalah jumlah paling sedikit

perubahan karakter yang diperlukan untuk mengubah satu kata menjadi kata lain.

5. Penentuan Kata Terdekat: Aplikasi akan menemukan kata dalam kamus yang memiliki jarak terkecil dengan kata input setelah menghitung jarak *Levenshtein*. Kata ini dianggap paling tepat atau paling dekat dengan apa yang dimaksud pengguna.
6. Saran Kata: Setelah itu, aplikasi akan menampilkan kata-kata yang telah dikoreksi atau kata-kata yang dekat dengannya sebagai rekomendasi. Ini memungkinkan pengguna untuk memilih kata yang tepat atau yang paling sesuai dengan konteks yang diinginkan.
7. Selesai: Proses berakhir setelah pengguna menerima rekomendasi kata atau jika kata yang dimasukkan benar dan ada dalam kamus.

D. Teknik Pengujian Sistem

Algoritma *Levenshtein*, yang juga dikenal sebagai edit *distance*, adalah metode yang umum digunakan untuk mengukur perbedaan antara dua string. Dalam konteks *spellchecker*, algoritma ini berguna untuk mengidentifikasi kesalahan ejaan dan menyarankan koreksi yang paling dekat.

E. Teknik Analisis Data

Analisis data adalah proses pencarian dan pengaturan sistematis hasil dan bahan-bahan yang dikumpulkan untuk meningkatkan pemahaman dan presentasi temuan. Proses analisis data yang digunakan dalam penelitian adalah sebagai berikut:

1. Preprocessing Data

Langkah pertama dalam menganalisis data adalah mempersiapkan data. Ini dapat mencakup hal-hal seperti membersihkan data dari suara atau data yang tidak penting, mengubah data ke format yang sesuai, dan memastikan bahwa data konsisten.

2. **Implementasi *Spell Checker***

   Menggunakan metode *Levenshtein* untuk menggunakan *SpellChecker*. Metode ini memerlukan pemrograman untuk menggunakan algoritma *Levenshtein*, yang menghitung jarak antara dua string yang berbeda. Algoritma ini diterapkan dalam bahasa pemrograman yang dipilih, dan kemudian diintegrasikan ke dalam aplikasi kamus Bahasa Bugis.

3. **Pengujian *Spell Checker***

   Setelah implementasi, *SpellChecker* diuji untuk memastikan bahwa itu efektif dalam menemukan dan memperbaiki kesalahan ejaan dalam pencarian kata. Ini dilakukan dengan menggunakan data uji yang mencakup kata-kata dengan kesalahan ejaan yang telah diketahui sebelumnya.

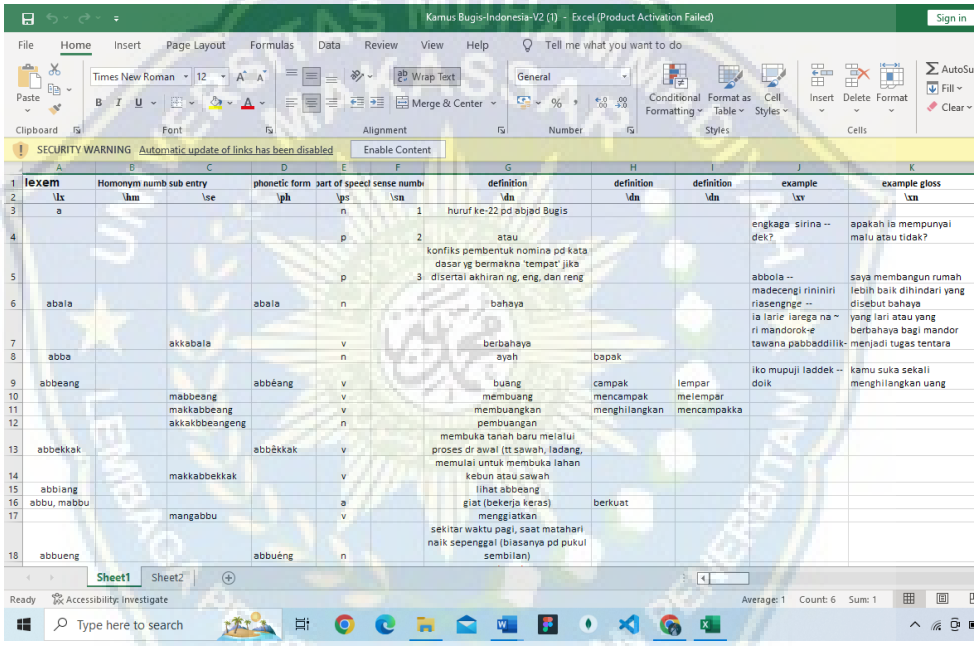
4. **Evaluasi Hasil**

   Berdasarkan hasil yang akan didapatkan, maka pengguna dapat menilai kegunaan dan efektivitas *SpellChecker* dalam kamus Bahasa Bugis menggunakan metode *Levenshtein* berdasarkan hasilnya.

BAB IV
HASIL DAN PEMBAHASAN

A. Pengambilan Data

Pada proses pengambilan data untuk pengaplikasian *spellchecker* pada aplikasi kamus Bahasa bugis menggunakan metode *levenshtein* ini diambil di Balai Bahasa Sulawesi Selatan.



	A	B	C	D	E	F	G	H	I	J	K
1	lexem	Homonym numb	sub entry	phonetic form	part of speec	sense numb	definition	definition	definition	example	example gloss
2	\lx	\hm	\se	\ph	\ps	\sn	\dn	\dn	\dn	\xv	\xn
3	a				n	1	huruf ke-22 pd abjad Bugis				
4					p	2	atau			engkaga sirina -- dek?	apakah ia mempunyai malu atau tidak?
5					p	3	konfiks pembentuk nomina pd kata dasar yg bermakna 'tempat' jika disertai akhiran ng, eng, dan reng			abbola --	saya membangun rumah
6	abala			abala	n		bahaya			madecengi rininiri riasengnge --	lebih baik dihindari yang disebut bahaya
7			akkabala		v		berbahaya			ia larie larega na ~ ri mandorok-e tawana pabbaddilik-	yang lari atau yang berbahaya bagi mandor menjadi tugas tentara
8	abba				n		ayah	bapak			
9	abbeang			abbéang	v		buang	campak	lempar	iko mupuji laddek -- doik	kamu suka sekali menghilangkan uang
10			mabbeang		v		membuang	mencampak	melempar		
11			makkabbeang		v		membuangkan	menghilangkan	mencampakka		
12			akkakbbeangeng		n		pembuangan				
13	abbekkak			abbékkak	v		membuka tanah baru melalui proses dr awal (tt sawah, ladang,				
14			makkabbekkak		v		memulai untuk membuka lahan kebun atau sawah				
15	abbiang						lihat abbeang				
16	abbu, mabbu				a		giat (bekerja keras)	berkuat			
17			mangabbu		v		menggiatkan				
18	abbueng			abbuéng	n		sekitar waktu pagi, saat matahari naik sepenggal (biasanya pd pukul sembilan)				

Gambar 2. Dataset aplikasi kamus Bahasa Bugis

1. **\lx (Lexem)**: Kolom ini berisi lexem atau lema, yaitu entri kata dalam kamus. Contohnya adalah "a", yang merupakan bentuk dasar dari kata yang dicari dalam kamus.
2. **\hm (Homonym Number)**: Kolom ini mencatat nomor homonim. Homonim adalah kata yang memiliki bentuk sama tetapi memiliki arti yang berbeda. Nomor ini membantu membedakan makna yang berbeda dari kata yang sama.

Misalnya, jika ada kata "a" dengan dua makna berbeda, masing-masing akan memiliki nomor homonim yang berbeda.

3. **\se (Sub Entry)**: Kolom ini berisi sub lema atau entri tambahan yang terkait dengan lexem utama. Sub entri ini bisa mencakup variasi atau bentuk turunan dari lexem utama.
4. **\ph (Phonetic Form)**: Kolom ini menunjukkan bentuk fonetik atau pelafalan dari lexem. Ini membantu pengguna mengetahui bagaimana kata tersebut diucapkan.
5. **\ps (Part of Speech)**: Kolom ini menunjukkan kelas kata, seperti noun (kata benda), verb (kata kerja), dan sebagainya. Ini memberikan informasi tentang fungsi grammatical dari lexem dalam kalimat.
6. **\sn (Sense Number)**: Kolom ini mencatat nomor makna atau polisemi dari lexem. Jika suatu lexem memiliki beberapa makna, nomor ini membantu mengidentifikasi dan membedakan masing-masing makna.
7. **Definition**: Kolom ini memberikan definisi atau arti dari lexem tersebut. Ini menjelaskan makna dari kata dalam konteks tertentu.
8. **Example**: Kolom ini memberikan contoh penggunaan kata dalam kalimat. Contoh ini membantu pengguna memahami bagaimana kata digunakan dalam konteks nyata.
9. **Example Gloss**: Kolom ini memberikan penjelasan atau terjemahan dari contoh kalimat yang diberikan dalam kolom Example.

B. Penerapan metode *Levenshtein*

```
function koreksi(kata: string, target: string, isFirst?:
boolean) {
    const isReverse = kata.length > target.length;
//‘isReverse’: Sebuah boolean yang menentukan apakah panjang
‘kata’ lebih besar dari ‘target’. Hal ini digunakan untuk
memastikan iterasi dilakukan pada string yang lebih pendek
terlebih dahulu, yang dapat mengoptimalkan kinerja.

    const matriksA = Array.from({ length: (isReverse ?
target.length : kata.length) + 1 });
    const matriksB = Array.from({ length: (isReverse ?
kata.length : target.length) + 1 });
```

```
//’matriksA’ dan ‘matriksB’ adalah array yang dibuat untuk menyimpan dimensi matriks biaya (cost matrix). Panjangnya bergantung pada panjang string yang lebih pendek dan lebih panjang.

    const cost: number[][] = [];
//‘cost’: Matriks 2D yang akan digunakan untuk menyimpan biaya operasi.

    for (let i = 0; i < matriksA.length; i++) {
        cost[i] = [];
        cost[i][0] = i;
    }

    for (let j = 0; j < matriksB.length; j++) {
        cost[0][j] = j;
    }
//Dua loop pertama ini mengisi nilai awal matriks biaya:
   a. ‘cost[i][0] = i’: Biaya mengubah ‘kata’ menjadi string
      kosong adalah ‘i’ (biaya penghapusan).
   b. ‘cost[0][j] = j’: Biaya mengubah string kosong menjadi
      ‘target’ adalah ‘j’ (biaya penyisipan).


    for (let i = 1; i < matriksA.length; i++) {
        for (let j = 1; j < matriksB.length; j++) {

            const costt = kata[isReverse ? j - 1 : i - 1] ==
target[isReverse ? i - 1 : j - 1] ? 0 : 1;

//costt: Biaya substitusi, yaitu 0 jika karakter saat ini sama, dan 1 jika berbeda.

            const hasil = [
                cost[i][j - 1] + 1, // Insert
                cost[i - 1][j - 1] + costt, // Replace
                cost[i - 1][j] + 1 // Delete
            ];
            const min: number = Math.min(...hasil);
            cost[i][j] = min;
   1. //hasil: Array yang menyimpan biaya untuk tiga operasi
      dasar:
      a. cost[i][j - 1] + 1: Biaya penyisipan.
      b. cost[i - 1][j - 1] + costt: Biaya substitusi.
      c. cost[i - 1][j] + 1: Biaya penghapusan.
```

2. cost[i][j] = min: Menentukan biaya minimum dari tiga operasi dasar dan menyimpannya di cost[i][j].

```
            if (i > 1 && j > 1 && kata[isReverse ? j - 1 : i - 1] === target[isReverse ? i - 2 : j - 2] && kata[isReverse ? j - 2 : i - 2] === target[isReverse ? i - 1 : j - 1]) {
                cost[i][j] = Math.min(cost[i][j], cost[i - 2][j - 2] + costt);
            }
        }
    }
```

1. //Kondisi ini mengecek apakah karakter saat ini dan sebelumnya dalam kedua string adalah sama jika diurutkan secara terbalik.
2. cost[i][j] = Math.min(cost[i][j], cost[i - 2][j - 2] + costt): Mengupdate biaya cost[i][j] jika transposisi (penukaran dua karakter yang bersebelahan) memberikan biaya yang lebih rendah.

```
    return cost[matriksA.length - 1][matriksB.length - 1];
}
//Mengembalikan nilai dari cost pada indeks terakhir yang merupakan jarak edit antara kata dan target.
```

Matriks biaya `cost` diinisialisasi dengan ukuran `(kata.length + 1) x (target.length + 1)`. Baris pertama diisi dengan nilai `0, 1, 2, ..., kata.length` yang melambangkan biaya menghapus semua karakter dari `kata`. Kolom pertama diisi dengan nilai `0, 1, 2, ..., target.length` yang melambangkan biaya menyisipkan semua karakter dari `target`.

Kemudian algoritma Levenshtein diterapkan. Untuk setiap posisi (i, j) dalam matriks:

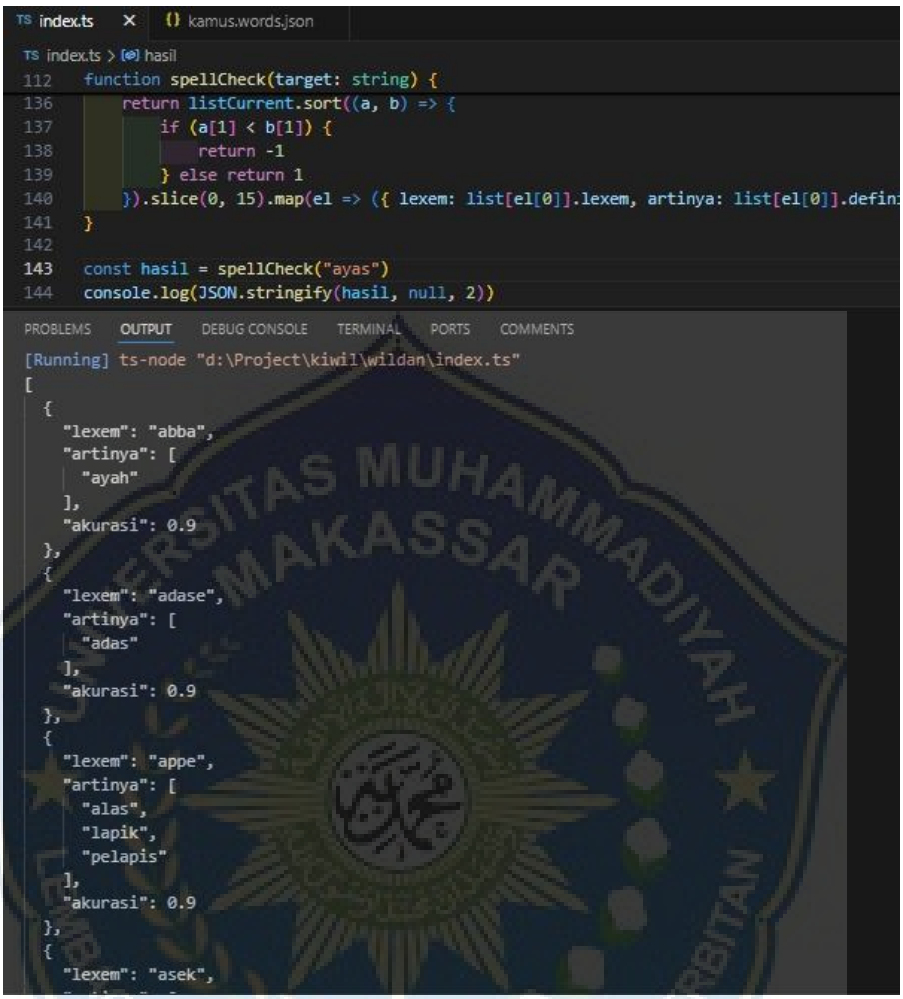
1. **Biaya Substitusi (Substitution Cost):** Jika karakter kata[i-1] sama dengan karakter target[j-1], biayanya 0, jika tidak 1.
2. **Operasi Minimum (Minimum Operation):** Menghitung biaya minimum dari tiga operasi:
    a. **Insert:** Biaya cost[i][j-1] + 1, menyisipkan karakter pada target.

b. **Replace:** Biaya cost[i-1][j-1] + costt, mengganti karakter pada kata dengan karakter pada target.
c. **Delete:** Biaya cost[i-1][j] + 1, menghapus karakter dari kata.

Memilih biaya minimum dari ketiga operasi tersebut dan menyimpannya di cost[i][j].

Dalam kode di atas, algoritma Levenshtein diimplementasikan dalam fungsi koreksi menggunakan matriks dynamic programming untuk menghitung jarak edit antara dua string. Matriks ini diinisialisasi dengan biaya dasar, kemudian diisi dengan biaya transformasi minimum dari satu karakter ke karakter lain. Akhirnya, biaya minimum untuk mengubah seluruh string kata menjadi target dikembalikan.

C. Teknik Pengujian Sistem



Gambar 3. Hasil *Output* pencarian kata

1. **"lexem": "abba"**

   "artinya": ["ayah"]

   "akurasi": 0.9

   Penjelasan: kata "ayah" memiliki kemiripan tinggi dengan kata target "ayas" dengan akurasi 90%. Meskipun bukan sinonim langsung, kata ini hanya memiliki satu perbedaan karakter, yaitu "s" menjadi "h".

2. **"lexem": "adase"**

   "artinya": ["adas"]

   "akurasi": 0.9

   Penjelasan: Kata "adase" memiliki arti "adas". Jarak edit antara "adas" dan "ayas" adalah 1 (satu operasi substitusi dari 'd' menjadi 's'). Akurasinya dihitung sebagai (100 - (1 * 10)) / 100 = 0.9 (atau 90%).

3. **"lexem": "appe"**

   "artinya": ["alas"]

   "akurasi": 0.9

   Penjelasan: Kata "appe" memiliki arti "alas". Jarak edit antara "alas" dan "ayas" adalah 1 (satu operasi substitusi dari 'l' menjadi 'y'). Akurasinya juga 0.9 (atau 90%).

Pengukuran kinerja algoritma *Levenshtein* dapat dilihat pada nilai *akurasi* yang dihasilkan.

$$Akurasi = \frac{100 - (1 \text{ x } 10)}{100}$$

$$= \frac{90}{100} \text{ x } 100\% = 90\%$$

Kata "ayuh" dalam kamus mungkin memiliki jarak edit 1 dari kata input "ayah" (satu operasi diperlukan untuk mengubah dari ‘u’ menjadi ‘a’), sehingga akurasinya adalah 90%.

Algoritma Levenshtein, yang dikenal juga sebagai edit distance, adalah metode yang umum digunakan untuk mengukur perbedaan antara dua string.

Dalam konteks spellchecker, algoritma ini berguna untuk mengidentifikasi kesalahan ejaan dan menyarankan koreksi yang paling dekat. Berikut adalah langkah-langkah untuk mengevaluasi kinerja spellchecker menggunakan beberapa metrik evaluasi.

BAB V
PENUTUP

A. KESIMPULAN

Berdasarkan tujuan penelitian yang telah ditetapkan, yaitu mengimplementasikan metode Levenshtein dalam spell check pada aplikasi kamus Bahasa Bugis serta mengevaluasi keefektifan metode tersebut dalam mendeteksi dan memperbaiki kesalahan ejaan, berikut adalah kesimpulan yang dapat diambil:

1. **Implementasi Metode Levenshtein:**
   a. Metode Levenshtein berhasil diimplementasikan dengan baik dalam spell check pada aplikasi kamus Bahasa Bugis.
   b. Mengimplementasikan metode Levenshtein dalam spell check pada aplikasi kamus Bahasa Bugis melibatkan penggunaan algoritma ini untuk mengukur jarak antara kata yang dimasukkan pengguna dengan kata-kata yang ada dalam kamus. Algoritma ini menghitung jumlah minimum operasi (penyisipan, penghapusan, atau substitusi) yang diperlukan untuk mengubah kata yang salah eja menjadi kata yang benar. Aplikasi kemudian memilih kata-kata dengan jarak terendah sebagai saran koreksi. Hasilnya, pengguna dapat dengan mudah menemukan kata yang tepat meskipun ada kesalahan pengetikan, sehingga meningkatkan efisiensi pencarian dan mendukung pelestarian Bahasa Bugis.
2. **Keefektifan Metode Levenshtein:**
   a. Evaluasi terhadap keefektifan metode Levenshtein menunjukkan bahwa metode ini sangat efektif dalam mendeteksi dan memperbaiki kesalahan ejaan dalam pencarian pada kamus Bahasa Bugis.
   b. Berdasarkan hasil pengujian, metode ini menunjukkan akurasi yang tinggi dalam memperbaiki kesalahan ejaan. Akurasi dapat diukur dengan menggunakan persentase saran yang benar dibandingkan dengan total

saran yang diberikan. Misalnya, jika dalam 100 percobaan, 90 di antaranya benar, maka akurasi mencapai 90%.

B. Saran

Berdasarkan hasil penelitian dan kesimpulan yang telah diambil, berikut adalah beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut:

1. **Pengujian Lebih Lanjut:**
   a. Melakukan pengujian lebih lanjut dengan berbagai jenis teks dan skenario pengguna untuk mengevaluasi kinerja spell checker dalam kondisi yang lebih beragam.
   b. Melibatkan lebih banyak pengguna dalam pengujian aplikasi untuk mendapatkan umpan balik yang lebih luas dan beragam.
2. **Pengembangan Antarmuka Pengguna:**
   a. Meningkatkan antarmuka pengguna aplikasi kamus agar lebih intuitif dan mudah digunakan, sehingga pengguna dapat dengan mudah memanfaatkan fitur spell check yang telah diimplementasikan.
   b. Menyediakan panduan atau tutorial tentang penggunaan fitur spell check untuk membantu pengguna memahami cara kerja dan manfaatnya.

Dengan mengikuti saran-saran di atas, diharapkan aplikasi kamus Bahasa Bugis dengan fitur spell check berbasis metode Levenshtein dapat menjadi lebih bermanfaat dan memberikan pengalaman yang lebih baik bagi pengguna. Penelitian dan pengembangan lebih lanjut akan membantu dalam memperkuat dan menyempurnakan teknologi ini sehingga dapat memberikan kontribusi yang lebih besar dalam pelestarian dan pembelajaran Bahasa Bugis.

DAFTAR PUSTAKA

Amelia Aprilianti, d. (2024). Penggunaan Bahasa Indonesia Baku Di Kalangan Mahasiswa Pada Base Twitter Colle. Bahasa dan Sastra , 1-7.

Anggasta Tirta Adi Kusuma, C. I. (2023). Perbandingan Metode Peter Norvig, Levenshtein distance, dan Damerau-Levenshtein distance : Tinjauan Literatur. (Jurnal) Universitas Islam Indonesia, 1-5.

Anis Nurma Sabila, A. M. (2023). Komponen dan Metode Penyusunan Kamus Hifdz Al-Mufrodat(Menghafal Kosakata). Bahasa dan Sastra, 1-14.

ASRIANI, R. F. (2021). Pengoreksian Ejaan Bahasa Minangkabau Menggunakan *Levenshtein Distance*. (Tugas Akhir) Universitas Islam Negeri Sultan Syarif Kasim Riau, 1-120.

Batisya, M. R. (2023). Implementasi Algoritma *Levenshtein Distance* Untuk *Misspelled Word* Pada Pencarian Lagu Melayu. Jurnal Informatika, 1-7.

Dede Abdurahman, D. S. (2023). Perancangan Aplikasi Sistem Informasi Perpustakaanmenggunakan Algoritma Levenshtein Distancedi Perpustakaan Sma Islam Al-Mizan. Infotech Journal, 1-6.

Hoerudin, C. W. (2023). Penerapan Media Vocabulary Card Dalam Meningkatkan Penguasaan Kosa Kata Bahasa Indonesia Anak Usia 4-5 Tahun. Jurnal Plamboyan Edu, 1-12.

Ira Zulfa, d. (2024). Sistem Pendeteksi Plagiarisme Pada Laporan Skripsi Dan Magang Mahasiswa Menggunakan Metode Levenshtein Distance (Studi Kasus : Fakultas Teknik Universitas Gajah Putih). JURTIE, 1-20.

Kusumanegara, A. (2020). Derivasi Generatif pada Nomina Bahasa Bahasa Bugis: Sebuah Benang Merah pada Bahasa Melayu. Tuah, 1-6.

Muhammad Thomas Fadhila Yahya, ,. S. (2021). Menormalisasikan Teks Pada Bot Sistem Informasi Akademik Menggunakan Algoritma *Damerau–Levenshtein Distance* Dan *Prefix Tree* (Studi Kasus: Universitas Teknokrat Indonesia). Ilmuteknik, 1-9.

Siti Shofiyah, A. K. (2023). Meningkatkan Penguasaan Kosakata Bahasa Indonesia Melalui Media Audio. Journal of Basic Education, 1-6.

Susi Rianti, R. A. (2023). Perbandingan Algoritma Edit Distance, Levenshtein Distance, Hamming Distance, Jaccard Similarity Dalam Mendeteksi String Matching. Jurnal Sistem Informasi, 1-10.

Yunita Purnama Sari, d. (2019). Pengembangan Aplikasi Kamus Bahasa Bima - Bahasa Indonesia Menggunakan Algoritma *Levenshtein Distance* Sebagai

Spell Checker Berbasis Android. Kumpulan Artikel Mahasiswa Pendidikan Teknik Informatika(Karmapati), 1-10.

LAMPIRAN

Lampiran 1. Source code

```
// Import kata-kata dari file kams.words.json
const list: Array<{
    definition: string[],
    lexem: string,
}> = require("./kamus.words.json")


function koreksi(kata: string, target: string, isFirst?: boolean)
{

    const isReverse = kata.length > target.length


    const matriksA = Array.from({
        length: (isReverse ? target.length : kata.length) + 1
    })

    const matriksB = Array.from({
        length: (isReverse ? kata.length : target.length) + 1
    })


    const cost: number[][] = []


    for (let i = 0; i < matriksA.length; i++) {
        cost[i] = []
        cost[i][0] = i
    }


    for (let j = 0; j < matriksB.length; j++) {
        cost[0][j] = j
    }


    if (isFirst) {
        const customCost: any[][] = structuredClone(cost)
        const kats = isReverse ?  kata : target
        console.log(kats)
        customCost.unshift([])
```

```
        for(let z = kats.length - 1 ; z >= 0 ; z--) {
            customCost[0].unshift(kats[z])
        }
        customCost[0].unshift("-")
        customCost[0].unshift("-")
        customCost[1].unshift("-")
        customCost.map((el, ind) => {
            if (ind < 2) return el
            const elAdd = el
            return elAdd.unshift(isReverse ? target[ind - 2] :
kata[ind - 2])
        })
        printMatrix(customCost)
    }


    for (let i = 1; i < matriksA.length; i++) {
        for (let j = 1; j < matriksB.length; j++) {

            const costt = kata[isReverse ? j - 1 : i - 1] ==
target[isReverse ? i - 1 : j - 1] ? 0 : 1

            const hasil = [
                cost[i][j - 1] + 1,
                cost[i - 1][j - 1] + costt,
                cost[i - 1][j] + 1
            ]
            const min: number = Math.min(...hasil)
            cost[i][j] = min

            if (i > 1 && j > 1 && kata[isReverse ? j - 1 : i - 1]
=== target[isReverse ? i - 2 : j - 2] && kata[isReverse ? j - 2 :
i - 2] === target[isReverse ? i - 1 : j - 1]) {
                cost[i][j] = Math.min(cost[i][j], cost[i - 2][j -
2] + costt);
            }
        }



        if (isFirst) {
            const customCost: any[][] = structuredClone(cost)
            const kats = isReverse ?  kata : target
            customCost.unshift([])
            for(let z = kats.length - 1 ; z >= 0 ; z--) {
```

```
                customCost[0].unshift(kats[z])
            }
            customCost[0].unshift("-")
            customCost[0].unshift("-")
            customCost[1].unshift("-")
            customCost.map((el, ind) => {
                if (ind < 2) return el
                const elAdd = el
                return elAdd.unshift(isReverse ? target[ind - 2]
: kata[ind - 2])
            })
            printMatrix(customCost)
        }
    }


    return cost[matriksA.length - 1][matriksB.length - 1]
}

function printMatrix(matr: any[][]) {
    for (let i = 0; i < matr.length; i++) {
        let text = ""
        for (let j = 0; j < matr[i].length; j++) {
            text += matr[i][j] + "\t"
        }
        console.log(text)
        console.log("\n")
    }
}


function spellCheck(target: string) {
    const listCurrent: Array<[number, number]> = []

    for (let i = 0; i < list.length; i++) {

        let future: null | number = null


        list[i].definition.forEach((el, ind) => {
            const currentIndexNow = koreksi(el.toLowerCase(),
target.toLowerCase(),  false)
            if (future == null) {
```

```
                future = currentIndexNow < 2 ? currentIndexNow :
null

            } else if (currentIndexNow < future) {
                future = currentIndexNow
            }
        })


        if (future != null) {
            listCurrent.push([i, future])
        }
    }

    return listCurrent.sort((a, b) => {
        if (a[1] < b[1]) {
            return -1
        } else return 1
    }).slice(0, 15).map(el => ({ lexem: list[el[0]].lexem,
artinya: list[el[0]].definition, akurasi : (100 - (el[1] * 10) )/
100}))
}


const hasil = spellCheck("ayuh")
console.log(JSON.stringify(hasil, null, 2))
```

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl.Sultan Alauddin N0.259 Makassar 90221 Tlp.(0411) 866972,881593, Fax.(0411) 865588

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar,
Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Ahmad Wildan Dzakki Adam

Nim : 105841101820

Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	10 %	10 %
2	Bab 2	19 %	25 %
3	Bab 3	8 %	10 %
4	Bab 4	5 %	10 %
5	Bab 5	5 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan seperlunya.

Makassar, 30 Agustus 2024
Mengetahui,

Kepala UPT- Perpustakaan dan Pernerbitan,

Nursinah, S.Hum.,M.I.P
NBM. 964 591

Jl. Sultan Alauddin no 259 makassar 90222
Telepon (0411)866972,881 593,fax (0411)865 588
Website: www.library.unismuh.ac.id
E-mail : perpustakaan@unismuh.ac.id