Arus Jurnal Sains dan Teknologi (AJST)
Website: http://jurnal.ardenjaya.com/index.php/ajst
Email: jurnal.ajst@ardenjaya.com

Pengaplikasian *Spell Checker* pada Aplikasi Kamus Bahasa Bugis dengan Metode *Levenshtein*

INFO PENULIS

Ahmad Wildan Dzakki Adam
Universitas Muhammadiyah Makassar
Ahmadwildandzakki2@gmail.com

Titin Wahyuni
Universitas Muhammadiyah Makassar

Muhyiddin A. M Hayat
Universitas Muhammadiyah Makassar

INFO ARTIKEL

ISSN: 3026-3603
Vol. 2, No. 2 Oktober 2024
http://jurnal.ardenjaya.com/index.php/ajst

Saran Penulisan Referensi:

Adam, A. W. D., Wahyuni, T., & Hayat, M. A. M. (2024). Pengaplikasian spell checker pada aplikasi kamus bahasa bugis degan metode levenshtein. *Arus Jurnal Sains dan Teknologi, 2* (2), 564-571.

Abstrak

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.

Abstract

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.*

A. Pendahuluan

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. Metodologi

a. *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:

a) Substitusi: Mengganti satu karakter dengan karakter lain
b) Penghapusan: Menghapus satu karakter.

c) 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. *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).

c. 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.

d. 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.

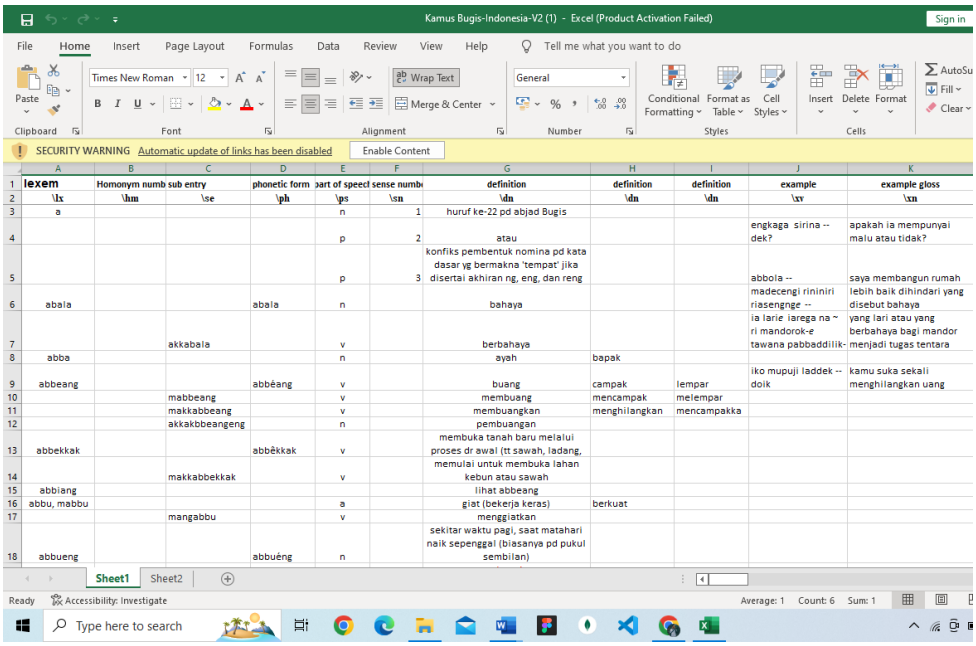
e. Kamus

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.

C. 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 1. Dataset aplikasi kamus Bahasa Bugis

a) **\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.
b) **\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.

c) **\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.
d) **\ph (Phonetic Form)**: Kolom ini menunjukkan bentuk fonetik atau pelafalan dari lexem. Ini membantu pengguna mengetahui bagaimana kata tersebut diucapkan.
e) **\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.
f) **\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.
g) **Definition**: Kolom ini memberikan definisi atau arti dari lexem tersebut. Ini menjelaskan makna dari kata dalam konteks tertentu.
h) **Example**: Kolom ini memberikan contoh penggunaan kata dalam kalimat. Contoh ini membantu pengguna memahami bagaimana kata digunakan dalam konteks nyata.
i) **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.
```

Gambar 2. Penerapan metode *Levenshtein* 1

```
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).
```

Gambar 3. Penerapan metode *Levenshtein* 2

```
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].
```

Gambar 4. Penerapan metode *Levenshtein* 3

```
                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.
```

Gambar 5. Penerapan metode *Levenshtein* 4

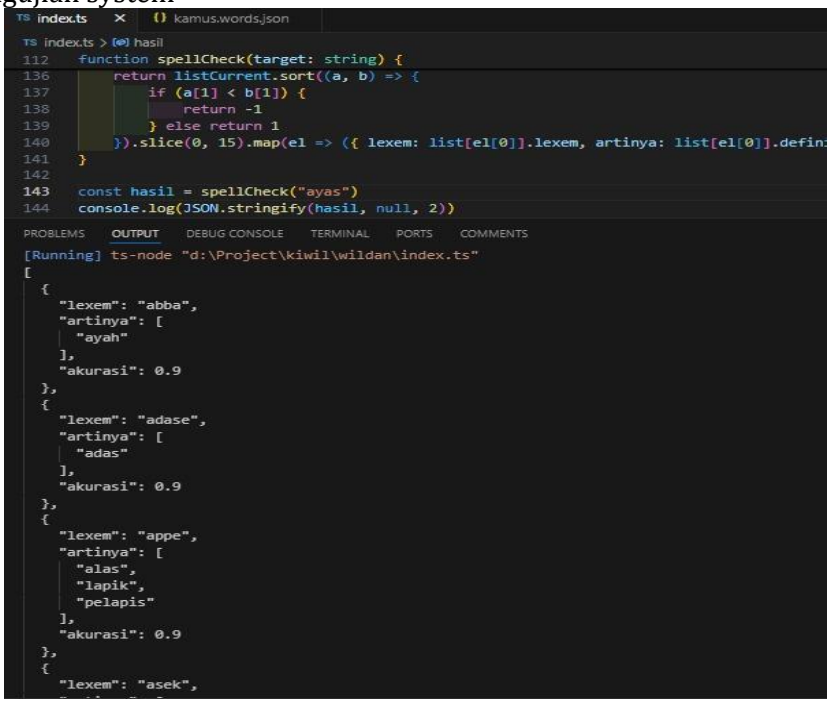
Matriks biaya cost diinisialisasi dengan ukuran (kata.length + 1) x (target.length + 1). Baris pertama diisi dengan nilai 0, 1, 2, ..., kata.length yang melambangkan biaya menghapus semua karakter dari kata. Kolom pertama diisi dengan nilai 0, 1, 2, ..., target.length yang melambangkan biaya menyisipkan semua karakter dari target. Kemudian algoritma Levenshtein diterapkan. Untuk setiap posisi (i, j) dalam matriks:

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



Gambar 6. Hasil *Output* pencarian kata

a) **"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".

b) **"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%).

c) **"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.

D. Kesimpulan

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.

E. Referensi

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.