IMPLEMENTASI *FUZZY SEARCH* UNTUK PENDETEKSI KATA ASING PADA DOKUMEN *MICROSOFT WORD*

Ichsan Taufik[1], Izma Dewi Aishia[2], Jumadi[3]

Jurusan Teknik Informatika, Fakultas Sains dan Teknologi UIN SGD Bandung
Jl. A.H Nasution 105 Bandung 40614
ichsan@uinsgd.ac.id[1], izma.dewi@uinsgd.ac.id[2], jumadi@uinsgd.ac.id[3]

ABSTRAK

Pendeteksian kata-kata bahasa asing ini dimaksudkan untuk membantu dalam mengurangi kesalahan dalam penulisan karya ilmiah dan pernyataan tesis sebagai pengerjaan. Ada aturan dalam penulisan karya ilmiah bahwa dokumen harus memenuhi peraturan penulisan bahasa Indonesia yang baik dan salah satunya adalah penggunaan huruf miring dalam bahasa asing (bahasa Inggris). Oleh karena itu, dibuat sebuah aplikasi untuk membantu dalam pendeteksian kata-kata bahasa asing sehingga huruf-huruf yang tidak ada dalam tata kelola bahasa Indonesia bisa dibuat menjadi betuk *italics*. Dalam pembuatan aplikasi digunakan metode pencarian fuzzy untuk mendeteksi kata asing dengan menggunakan matriks jarak jauh levenshtein untuk menghitung jarak kemiripan sebuah kata. Pencarian fuzzy dilakukan dengan *fuzzy matching* yang mengembalikan daftar hasil berdasarkan variabel yang telah ditentukan. Pencarian fuzzy menggunakan matriks dengan levenshtein mewakili jarak untuk menghitung jarak kemiripan kata, sehingga akan menghasilkan kata yang memiliki huruf miring dan tidak berhuruf miring jika ada dalam tata bahasa Indonesia. Dari hasil pengujian, penggunaan algoritma pencarian fuzzy untuk mendeteksi kata-kata bahasa asing dalam sebuah dokumen kata dengan jumlah kata yang berbeda menghasilkan nilai akurasi rata-rata 89,6%. Hasil dari proses ini membuat aplikasi ini bisa membantu mendeteksi tatabahasa asing dalam penulisan karya ilmiah.

Kata Kunci*: dokumen word, gaya italic, jarak edit levenshtein, pencarian fuzzy*

ABSTRACT

Detection of foreign language words is intended to help in reducing errors in the writing of scientific papers and thesis statements as workmanship. There are rules in writing scientific papers that document must meet the rules of Indonesian language good writing rule and one of them is the use of italics in the foreign language (English) words. Therefore, an application was made to aid in the detection of foreign language words so the letters that do not exist in the Indonesian word grammar can be made into italics form. In the application making, the fuzzy search method used for detecting foreign language words using levenshtein long distance matrix to calculate the distance semblance of a word. Fuzzy search conducted with fuzzy matching which returns a list of results based on the variables that have been determined. A fuzzy search using a matrix with levenshtein represented distance to calculate the distance of the similarity of the word, so will produce a word that has italic style and not in italic if there is word in Indonesian grammar. From the test results, the use of fuzzy search algorithms to detect foreign language words in a word document with a different number of words resulted in an average accuracy value of 89.6%. The result of this process makes this application can help to detect foreign language grammar in writing scientific papers.

Keywords*: word document, style italic, levenshtein edit distance, fuzzy search*

DOI: 10.15408/jti.v10i1.6804

I. PENDAHULUAN

Perkembangan teknologi informasi, banyak digunakan dalam pencocokan *string*, seperti penggunaan *internet* yaitu dengan memasukkan data *input* oleh *user* dengan data yang berdasarkan pada *database* misalnya *database google*. Sehingga algoritma pencarian atau pencocokan *string* merupakan proses pencarian yang penting dan dibutuhkan dalam berbagai aplikasi.Di dalam pencocokan *string* ataukata terdapat beberapa metode yang dapat digunakan seperti *fuzzy search.* F*uzzy search* dapat melakukan proses pencarian efisien dengan data yang besar. Seperti telah diteliti *fuzzy search* dapat menghabiskan waktu sekitar 20 detik pada 2010 dengan data dari *wikipedia* bahasa inggris (23GB XML, yang sebagian besar teks artikel). Jika menggunakan *fuzzy search* maka terdapat pekerjaan yang kecil dalam mengefisienkan proses pencarian. Cara kerja yang dilakukan dari *fuzzy search* ini akan menampilkan *range* yang ada di dalam proses pencarian, seperti derajat keanggotaan yang dimiliki yaitu mirip, tidak mirip, cukup mirip.

Dalam pembahasan mengenai metode pada pencocokan string, hal yang akan diteliti dalam penelitian ini adalah penulisan kata-kata asing dalam karya ilmiah atau *document Microsoft Word* jika menggunakan tata bahasa Indonesia yang baik dan benar maka kata tersebut harus dimiringkan. Salah satu pemakaiannya adalah menuliskan nama ilmiah atau ungkapan asing yang belum disesuaikan ejaannya. Penulisan kata asing dalam *Microsoft Word* dapat sulit terdeteksi jika jumlah data tersebut banyak misalnya dalam penulisan 1 paragraf saja kata asing akan sulit terdeteksi karena sifat manusia yang tidak cermat dalam mengamati kata-kata asing tersebut.

Berdasarkan permasalahan tersebut, maka terdapat permasalahan yang menarik yaitu bagaimana pencarian kata asing dalam mencocokkan kata tersebut sehingga diubah menjadi *style font italic* oleh *system* sehingga dapat menghasilkan karya ilmiah atau *document Microsoft Word* yang sesuai dengan kaidah Indonesia yang baik dan benar. Sehingga disusunlah penelitian mengenai "**Implementasi *Fuzzy Search* Untuk Pendeteksi Kata Asing Pada *Document Microsoft Word***". Adapun artikel ini merupakan hasil orisinil dari penelitian penulis.

II. METODOLOGI

2.1 Pencocokan *String*

String merupakan susunan dari karakter-karakter (angka, alfabet, atau karakter yang lain) dan biasanya direpresentasikan sebagai struktur data *array*. String dapat berupa kata, frase, atau kalimat. Pencocokan string merupakan bagian penting dari sebuah proses pencarian *string* (*string searching*) dalam sebuah dokumen. Hasil dari pencarian sebuah string dalam dokumen tergantung dari teknik atau cara pencocokan *string* yang digunakan. Untuk mengetahui isi dokumen yang benar sesuai dengan kebutuhan informasi, diperlukan metode pencarian *string* (*string searching*) isi dokumen yang bagus. Proses pencocokan *string* (*string searching*) yang merupakan bagian utama dalam proses pencarian string memegang peranan penting untuk mendapatkan dokumen yang sesuai dengan kebutuhan informasi tersebut [28].

2.2 *Fuzzy Search*

Sebuah *fuzzy search* adalah suatu proses yang menempatkan halaman *web* yang mungkin relevan dengan argumen pencarian bahkan ketika argumen tidak sesuai untuk informasi yang diinginkan. Sebuah *fuzzy search* dilakukan dengan pencocokan *fuzzy*, yang mengembalikan daftar hasil berdasarkan kemungkinan relevansi meskipun pencarian kata argumen dan ejaan mungkin tidak tepat. *Fuzzy search* berfungsi untuk menemukan semua kata yang memiliki kemiripan dengan *query* yang diberikan pada kamus tersebut atau *database* yang ada.

Fuzzy search dapat mengkompensasi kesalahan pengetikan *input* yang umum serta kesalahan yang diperkenalkan oleh pengenalan karakter optik *scanning* dokumen yang dicetak. Program pencocokan *fuzzy* biasanya menampilkan istilah yang tidak relevan serta yang relevan. Hasil yang berlebihan mungkin juga untuk memadupadankan dengan beberapa arti, hanya salah satu makna yang dimaksud pengguna. *Fuzzy search* jauh lebih kuat jika dibandingkan dengan *exaxt string search* bila digunakan untuk penelitian dan penyelidikan. *Fuzzy search* sangat berguna ketika meneliti bahasa-bahasa asing atau istilah-istilah yang menggunakan ejaan yang tepat dan tidak diketahui secara luas. *Fuzzy search* juga dapat digunakan untuk mencari individu yang

didasarkan pada informasi lengkap ataupun sebagian guna memperoleh data yang akurat.

2.3 *Levenshtein distance*

Lavenshtein distance dibuat oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* di dapatkan dari *matriks* yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*[1]. Perhitungan jarak antara dua *string* ini ditentukan dari jumlah minimum operasi perubahan untuk membuat *string* A menjadi string B.

Algoritma ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang memberikan jumlah perbedaan dua *string*.

Pencocokan dilakukan per kalimat. Kalimat dari proses *sorting* yang dalam bentuk *array* diubah dulu menjadi bentuk *string*. Kemudian *string* tersebut akan dihitung nilai *similarity* menggunakan rumus:

$$Plagiarized\ Value = \left\{1 - \frac{Diff}{Max(CS, ST)}\right\} * 100 \quad (1)$$

Keterangan rumus:

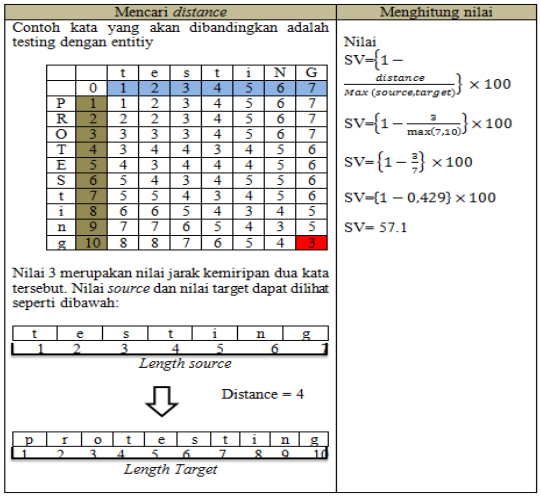
Plagiarize value = kesamaan kemiripan
Diff = nilai dari jarak *string* asli ke *string* pembanding
CS = *String* asli
ST = *String* pembanding
Max(CS, ST) = jarak *string* terbesar dari *string* pembanding dan *string* asli

2.4 Penggunaan *fuzzy search* untuk data

Contoh proses penggunaan *levenshtein edit distance* yaitu sebagai berikut: terdapat teks yang ada pada *user* yang dimasukkan ke dalam aplikasi.

Kemudian kata-kata tersebut akan melakukan proses *levenshtein edit distance* dengan menggunakan fungsi *matrix*. Algoritma ini berjalan mulai dari pojok kiri atau sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string*. Contoh perhitungan yang akan dilakukan seperti terlihat pada Gambar 1.

Jika karakter sama maka nilai diambil yang terkecil jika karakter berbeda nilai terkecil ditambah 1. Untuk perhitungan jarak kemiripan.



Gambar 1. Perhitungan nilai kemiripan

Perhitungan *distance* pada Gambar 1 berdasarkan algoritma pada nilai *levenshtein edit distance* yaitu jika karakter 1 *source* (data yang diinputkan *user*) dengan 1 target (data yang ada pada *database*) sama maka nilai yang diambil adalah minimum. Tetapi jika karakter berbeda maka nilai *cell* didapatkan dari nilai minimum dari karakter tersebut.

Jadi, setiap karakter dibandingkan kemudian nilai minimum ditambah 1 untuk karakter yang berbeda. Jika karakter yang sama tidak perlu ditambah 1.

Setelah didapat nilai *distance* kemudian dilakukan proses *fuzzy searchnya*. Di dalam *fuzzy* terdapat *variabel* yang digunakan. Dilakukan penamaan terhadap grup yang mewakili kondisi tertentu dengan menggunakan bahasa alami. Variabel tersebut diantaranya tidak mirip, kurang mirip, cukup mirip, mirip dan sangat mirip. Untuk nilai yang ada pada setiap variabelnya akan sebutkan pada Tabel 1 yang membahas mengenai setiap nilai dari setiap variabel. Variabel yang digunakan pada berdasarkan Tabel 1 adalah berjumlah 5.

Tabel 1. Nilai untuk setiap variabel

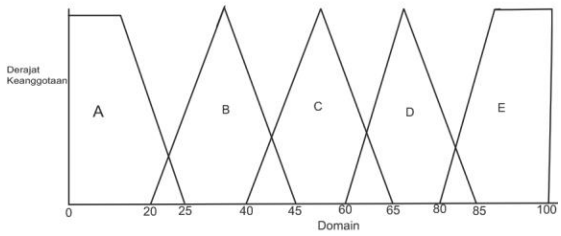
No	Variabel	Nilai
1	Tidak mirip	0-25
2	Kurang mirip	20-45
3	Cukup mirip	40-65
4	Mirip	60-85
5	Sangat mirip	80-100

Sehingga didapatkan *rule* dari variabel tersebut. Nilai *distance* merupakan nilai *crisp* atau disebut juga nilai input. Diinisialisasi menjadi X. jika variabel setiap nilai sudah disebutkan berdasarkan Tabel 1, maka *rule evaluation* berdasarkan variabel tersebut dicantumkan pada Tabel 2 *Rule evaluation.*

Tabel 2. Rule evaluation

No	*Rule*
1	*If* X lebih dari 0 dan kurang dari sama dengan 25 then masuk variabel tidak mirip (A)
2	If X lebih dari 20 dan kurang dari sama dengan 40 then masuk variabel kurang mirip (B)
3	*If X lebih dari 40 dan kurang dari sama dengan 65 then masuk variabel cukup mirip (C)*
4	*If X lebih dari 60 dan kurang dari sama dengan 85 then masuk variabel mirip (D)*
5	*If X lebih dari 80 dan kurang sama dengan 100 then masuk variabel kurang mirip (E)*

Pada Tabel 1 klasifikasi pada variabel nilai kemiripan menggunakan bahasa alami. Sedangkan Tabel 2 merupakan *rule* jika nilai *crisp* berada pada nilai-nilai variabel tersebut. Untuk fungsi keanggotaan (*membership function*) untuk menunjukkan pemetaan titik-titik input ke dalam nilai keanggotaannya (derajat keanggotaan) dapat dilihat pada Gambar 2.



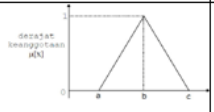
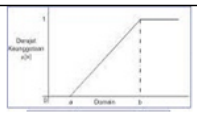
Gambar 2. Representasi Nilai Kemiripan

Keterangan :
A untuk variabel tidak mirip
B untuk variabel kurang mirip
C untuk variabel cukup mirip
D untuk variabel mirip
E untuk variabel sangat mirip.

Pada Gambar 2. memperlihatkan reprensentasi nilai kemiripan menggunakan kurva segitiga dan kuva bentuk bahu. Nilai variabel yaitu dari 0 sampai 100. Nilai *distance* pada Tabel 1 akan dimasukkan ke dalam variabel sesuai nilai yang ada pada gambar 2. pada Tabel 4 Merupakan contoh klasifikasi awal dari nilai *distance* yang telah disebutkan

Rumus untuk setiap variabel dapat ditunjukkan pada Tabel 3 merupakan rumus untuk setiap variabel. Terdapat dua bentuk yaitu bentuk bahu dan bentuk segitiga. rumus untuk setiap bentuk dapat ditunjukkan pada Tabel 3.

Tabel 3. Rumus Variabel

No	Bentuk	Gambar	Rumus
1	Segitiga		x <= a atau x >= c μ[x] =0 a <= x <= b μ[x] =(x - a) /(b - a) b <= x < = c μ[x] = (b - x) / (c - b)
2	Bahu		x <= a μ[x] = 0 a <= x <= b μ[x] =(x - a) / (b - a) x >= b μ[x] = 1
3	Bahu		x <= a μ[x] = 1 a <= x <= b μ[x] = (b - x) / (b - a) x >= b μ[x] = 0

Sedangkan Tabel 4 merupakan rumus untuk setiap variabel yang digunakan pada setiap nilai di dalam variabel.

Tabel 4. Keterangan Rumus

No	Nilai	Variabel	Bentuk rumus	Rumus
1	Nilai < 20	A	Bahu	Bahu variabel A 1. 15 <= nilai dan 15 <=25 Hasil = 25-nilai/25-15 2. Nilai >=15 dan nilai <=25 Hasil =0
2	Nilai <=25 dan nilai >=20	A dan B	Bahu dan segitiga	**Bahu Variabel A:** 1. Jika 15 <= nilai dan 15 <=25 Hasil = 25-nilai/25-15 2. Jika Nilai >=15 dan nilai <=25 Hasil =0 **Segitiga Variabel B:** 1. jika 20 <= nilai dan nilai <=32.5 Hasil = nilai – 20 / 32.5 - 20 2. jika 32.5 <= nilai dan nilai <=45 Hasil = 45 – nilai / 45 – 32.5 3. jika nilai <= 20 atau nilai <= 45 Hasil = 0
3	Nilai >25 dan nilai <40	B	Segitiga	**Segitiga variabel B:** 1. jika 20 <= nilai dan nilai <=32.5 Hasil = nilai – 20 / 32.5 - 20 2. jika 32.5 <= nilai dan nilai <=45 Hasil = 45 – nilai / 45 – 32.5 3. jika nilai <= 20 atau nilai <= 45 Hasil = 0
4	Nilai >=40 dan nilai <=45	B dan C	Segitiga	**Segitiga variabel B:** 1. jika 20 <= nilai dan nilai <=32.5 Hasil = nilai – 20 / 32.5 - 20 2. jika 32.5 <= nilai dan nilai <=45 Hasil = 45 – nilai / 45 – 32.5 3. jika nilai <= 20 atau nilai <= 45 Hasil = 0
				Segitiga Variabel C: 1. jika 40 <= nilai dan nilai <=52.5 Hasil = nilai – 40 / 52.5 - 40 2. jika 52.5 <= nilai dan nilai <= 65 Hasil = 65 – nilai / 65 – 52.5 3. jika nilai <= 40 atau nilai <= 65 Hasil = 0
5	Nilai>45 dan nilai <60	C	Segitiga	**Segitiga Variabel C:** 1. jika 40 <= nilai dan nilai <=52.5 Hasil = nilai – 40 / 52.5 - 40 2. jika 52.5 <= nilai dan nilai <= 65 Hasil = 65 – nilai / 65 – 52.5 3. jika nilai <= 40 atau nilai <= 65 Hasil = 0
6	Nilai >=65 dan nlai <=60	C dan D	Segitiga	**Segitiga Variabel C:** 1. jika 40 <= nilai dan nilai <=52.5 Hasil = nilai – 40 / 52.5 - 40 2. jika 52.5 <= nilai dan nilai <= 65 Hasil = 65 – nilai / 65 – 52.5 3. jika nilai <= 40 atau nilai <= 65 Hasil = 0
				Segitiga Variabel D: 1. jika 60 <= nilai dan nilai <=72.5 Hasil = nilai – 60 / 72.5 - 60 2. jika 72.5 <= nilai dan nilai <= 85 Hasil = 85 – nilai / 85 – 72.5 3. jika nilai <= 60 atau nilai <= 85 Hasil = 0
7	Nilai < 80 dan nilai > 65	D	Segitiga	**Segitiga Variabel D:** 1. jika 60 <= nilai dan nilai <=72.5 Hasil = nilai – 60 / 72.5 - 60 2. jika 72.5 <= nilai dan nilai <= 85 Hasil = 85 – nilai / 85 – 72.5 3. jika nilai <= 60 atau nilai <= 85 Hasil = 0
8	Nilai >=80 dan nilai <=85	D dan E	Segitiga dan bahu	**Segitiga Variabel D:** 1. jika 60 <= nilai dan nilai <=72.5 Hasil = nilai – 60 / 72.5 - 60 2. jika 72.5 <= nilai dan nilai <= 85 Hasil = 85 – nilai / 85 – 72.5 3. jika nilai <= 60 atau nilai <= 85 Hasil = 0 **Bahu Variabel E:** 1. Jika 80 <= nilai dan nilai <=90 Hasil = 80 – nilai / 90 - 80 2. Jika 80 >=nilai Hasil =0 3. Jika nilai >= 90 Hasil = 1
9	Nilai > 85	E	Bahu	**Bahu variabel E:** 1. Jika 80 <= nilai dan nilai <=90 Hasil = 80 – nilai / 90 - 80 2. Jika 80 >=nilai Hasil =0 3. Jika nilai >= 90 Hasil = 1

Nilai pada Tabel 4 merupakan nilai *distance* yang dihasilkan dari menghitung nilai kemiripan satu kata dengan kata yang lainnya.

Tabel 5. Nilai klasifikasi awal

No	*Source*	*Target*	Nilai *distance*	variabel
1	*Testing*	*Protesting*	70	D
		Protestingly	58.34	C
		Test	57.15	C
		Testing	100	E
2	*Form*	*Conform*	57.16	C
		Deformation	36.7	B
		Filiform	50	C
		Form	100	A

Setelah didapatkan nilai distance dan identifikasi variabel yang ada. Maka selanjutnya yaitu perhitungan nilai *fuzzy*. Gambar 3 Merupakan contoh perhitungan yang dilakukan.

No	Perhitungan		
1	Testing → protesting (x= 70) di variabel D A <= X <= B → (X - A)/(B-A) μ mirip [70] = $\frac{70-60}{72.5-60}$ = 0.81	2	Testing → protestingly (x=58.34) di variabel C B <= X <= C → (c-x)/(c-b) μcukup mirip[58.34] = $\frac{65-58.34}{65-52.5}$ = 0.54
3	Testing → test di variabel C (x=57.15) B <= X <= C → (c-x)/(c-b) μcukup mirip[57.15] = $\frac{65-57.15}{65-52.5}$ = 0.63	4	Testing →testing di variabel E X>=b μsangat mirip[100] = 1
5	form → conform (x= 57.15) di variabel C B <= X <= C → (c-x)/(c-b) μcukup mirip[57.15] = $\frac{65-57.15}{65-52.5}$ = 0.63	6	Form→ deformation (x=36.37) di variabel B B <= X <= C → (c-x)/(c-b) μkurangmirip[36.37] = $\frac{45-36.37}{45-32.5}$ =0.7
7	Form →filiform (x=50) di C A <= X <= B → (x-a)/(b-a) μ cukup mirip[50] = $\frac{50-40}{52.5-40}$ = 0.81	8	form→form di variabel E X>=b μsangat mirip[100] = 1

Gambar 3. Contoh perhitungan

Berdasarkan perhitungan pada Gambar 3 maka dapat direpresentasikan nilai pada setiap *rule Evaluation* setelah proses perhitungan *fuzzy* dapat ditunjukkan berdasarkan Tabel 6 Nilai dan saran yang akan dihasilkan

Tabel 6 Nilai dan saran

No	Nilai Variabel					Saran
	Tidak mirip (A)	Kurang mirip (B)	Cukup mirip (C)	Mirip (D)	Sangat mirip (E)	
1	-	-	-	0.81	-	Testing
2	-	-	0.54	-	-	
3	-	-	0.63	-	-	
4	-	-	-	-	1	
5	-	-	0.63	-	-	Form
6	-	0.7	-	-	-	
7		-	0.81	-	-	
8	-	-	-	-	1	

Berdasarkan Tabel 6, kata tersebut ada pada *database* bahasa inggris dengan berada pada variabel sangat mirip. Setelah di cek pada nilai variabel. Sehingga *testing* berada di variabel sangat mirip dengan nilai 1, dan form berada pada variabel sangat mirip dengan nilai 1. Maka proses selanjutnya adalah melakukan kemiringan huruf.

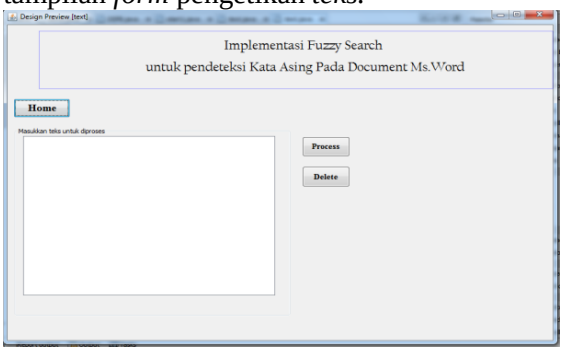
Berdasarkan perhitungan fuzzy maka *output* yang dihasilkan sebagai berikut dengan perhitungan kata asing yang telah dilakukan:
Setelah proses implementasi maka dilakukan ***testing*** untuk pengecekan aplikasi. Aplikasi ini menggunakan satu ***form*** dalam prosesnya.

III. HASIL DAN PEMBAHASAN

3.1 Implementasi Program

Aplikasi pendeteksi kata asing pada *document Ms.word* ini jenis aplikasi yang berbasis *desktop.* Pada bab ini akan diperlihatkan bagaimana tampilan program setelah diimplementasikan ke dalam *coding* dan berdasarkan perancangan pada analisa. Berikut salah satu form yang ada di dalam aplikasi**.**

Form pengetikan teks ini digunakan agar lebih mempermudah *user* dalam proses penggunaan aplikasi. Kemudahan yang diberikan *user* ini yaitu dengan memberikan *textarea* agar *user* dapat mengetikkan teks pada halaman yang disediakan. Berikut Gambar 4 tampilan *form* pengetikan *teks.*



Gambar 4. *Form* pengetikan *teks*

3.2 Pengujian

Berikut adalah pengujian dengan menggunakan waktu pada Tabel 3.1

Tabel 7. Pengujian jumlah kata

No	Jumlah kata	Jumlah nilai asing	Jumlah eksekusi program	Waktu/s
1	30	6	5	2.16
2	Document (182)	10	9	3.27
3	Latar Belakang	30	25	11.38
4	Document (2985 kata)	205	193	1 menit 33.59 detik
5	Bab 2 (8363)	223	217	3 menit 28.69 detik

Untuk perhitungan akurasi setiap digunakan rumus:

$$\text{Akurasi} = \frac{jumlah\ yang\ cocok}{jumlah\ keseluruhan} \times 100$$

$$\text{Akurasi} = \frac{25}{30} \times 100 = 83.33\%$$

Sehingga berdasarkan akurasi untuk setiap data yang digunakan dapat dilihat pada tabel 8 Perhitungan nilai akurasi

Tabel 8 Perhitungan nilai akurasi

No	Jumlah kata	Jumlah nilai asing	Jumlah eksekusi program	Waktu/s	Hasil akurasi
1	30	6	5	2.16	83.33%
2	Document (182)	10	9	3.27	90%
3	Latar Belakang	30	25	11.38	83.33%
4	Document (2985 kata)	205	193	1 menit 33.59 detik	94.15%
5	Bab 2 (8363)	223	217	3 menit 28.69 detik	97.31%

Sehingga nilai akurasi secara keseluruhan atau nilai rata-rata yang diperoleh adalah

$$\text{Rata-rata} = \frac{83.33+90+83.33+94.15+97.31}{5}$$

$$\text{Rata-rata} = 89.624\%$$

Sehingga didapat nilai akurasi berdasarkan pengujian tersebut adalah 89.624%. hasil akurasi tersebut menggunakan *fuzzy search* dengan *matrix* levenshtein *distance*.

IV. PENUTUP

4.1 Simpulan

Berdasarkan hasil implementasi penggunaan *fuzzy search* dalam proses pencocokan *string* dengan menggunakan matrix levenshtein *distance* dapat membantu dalam mengatasi kemiripan dari setiap kata. Dengan memeriksa terlebih dahulu kata yang berbahasa Indonesia, jika kata tersebut tidak ada maka akan diproses apakah kata tersebut kata asing atau tidak. Setelah percobaan pengujian dengan menggunakan teks yang sedikit, sedang dan banyak dari hasil akurasi yang diperoleh dengan nilai masing-masing sehingga menghasilkan nilai rata-rata yaitu 89.6%.

Berdasarkan nilai rata-rata tersebut terdapat nilai *error* yang ada artinya jumlah kata yang diperhitungkan secara manual dengan nilai yang dieksekusi program tidak sesuai karena terdapat beberapa kata yang ada yang dianggap kata asing belum masuk ke dalam *database* bahasa inggris, sehingga *system* mengenal kata tersebut bukan kata asing yang masuk ke dalam *database*. Sehingga jika terdapat kata yang dianggap kata asing maka harus ditambahkan ke dalam *database* terlebih dahulu.

Program yang telah dibuat dapat meminimalisir kesalahan penulisan EYD (Ejaan Yang Disempurnakan) yaitu dengan memiringkan kata asing pada pembuatan *document* seperti karya ilmiah atau skripsi yang mengharuskan penulisan sesuai penulisan bahasa Indonesia yang baik dan benar atau aturan EYD.

4.2 Saran

Saran yang dapat disampaikan bahwa dalam proses implementasi terdapat beberapa kekurangan, seperti dalam waktu yang cukup lama jika data yang dimasukkan banyak. Oleh sebab itu untuk penelitian selanjutnya dapat memperhitungkan waktu dengan baik dengan membandingkan algoritma *fuzzy search* dengan adanya proses *text processing* di dalamnya.

DAFTAR PUSTAKA

[1] Joseph F Hair, G Thomas M Hult, Christian M Ringle, and Marko Sarstedt, *A Primer on Partial Least Square Struktural Equation Modeling*. New York: SAGE, 2013.

[2] J. J Hox, *Multilevel Analysis Techniques and Applications*, 2nd ed. New York, USA: Routledge, 2010.

[3] K Ringdal, "Method for Multilevel Analysis," *Acta Sosiologica*, vol. 35, pp. 235-243, 1992.

[4] B. Tantular, Aunuddin, and H. Wijayanto, "Pemilihan Model Regresi Linier Multilevel Terbaik," in *Forum Statistika dan Komputasi*, vol. 14, 2009, pp. 1-7.