DOI: https://doi.org/10.51903/elkom.v17i1.1445

JURNAL ILMIAH ELEKTRONIKA DAN KOMPUTER, Vol.17, No.1, Juli 2024, pp.65-73
p-ISSN : 1907-0012 (print)
e-ISSN : 2714-5417 (online)
http://journal.stekom.ac.id/index.php/elkom

Aplikasi *Spelling Correcting* Pada Penulisan Bahasa Indonesia Dengan Metode *Jaro Winkler*

Fauzan Risang Agung Samudero[1], Jati Sasongko Wibowo[2]
[1]Teknik Informatika – Unisbank Semarang, fauzanrisang37@gmail.com
[2]Teknik Informatika – Unisbank Semarang, jatisw@edu.unisbank.ac.id
Jalan Tri Lomba Juang Semarang, Telp (024) 8451976

ARTICLE INFO

Article history:

Received September 29, 2023
Received in form 2 Oktober 2023
Accepted 8 November 2023
Available online 1 Juli 2024

ABSTRACT

Typing errors in documents are fairly common. The process of checking typing errors manually will take a lot of time. Therefore, a system called Indonesian spelling correcting with Jaro Winkler is needed. Spelling Correcting is a feature that can check and correct word writing errors automatically. Jaro Winkler is an algorithm used for the word correction process by calculating the value obtained from the results of the operation of modifying one word with another word with the help of a matrix. Based on the results of trials on 13 words, namely "Alaman Amus Seperti Pada Gamba Tersebut Digunakan Untuk Memasukkan Data Kata Pada KBBI ", this spelling correcting application can produce 3 correct word corrections to " halaman kamus seperti pada gambar tersebut digunakan untuk data pada kbbi ", each Word improvement is calculated using the Jaro Winkler formula, getting a score above 0.94 for each word tested.

Keywords: *Spelling Correcting, Jaro Winkler, Indonesian Language*

1. Pendahuluan

Kesalahan pengetikan pada dokumen cukup umum terjadi. Ditambah lagi dengan kesadaran masyarakat untuk menuangkan idenya ke dalam artikel, jurnal ilmiah, tugas kuliah ataupun dokumen lainnya mengalami peningkatan. Kesalahan dalam penulisan kata ataupun kalimat dapat dibagi menjadi dua jenis, yaitu kesalahan yang bukan kata sebenarnya dan kesalahan kata yang sebenarnya. Kesalahan yang bukan kata sebenarnya yaitu penangangan kata yang salah eja disebabkan oleh *typo* atau tipografi. Kesalahan kata yang sebenarnya ditekankan pada penanganan kesalahan penempatan kata dalam sebuah kalimat. Hal ini menyebabkan kesalahan dalam ejaan sehingga menyebabkan arti yang berbeda. Begitu pula dalam Bahasa Indonesia sebagai contoh kata ‘sya’ yang penulisan seharusnya adalah ‘saya’ maka jika terjadi kesalahan dalam ejaan maka akan memiliki arti yang berbeda atau bahkan tidak memiliki arti. Kesalahan tersebut juga dapat membuat sebagian orang menjadi kebingungan untuk menentukan apa kata atau kalimat yang tepat dalam penulisan tersebut.

Proses pengecekan kesalahan pengetikan dengan cara manual akan memakan banyak waktu dan memerlukan suatu sumber pasti sebagai acuan bahwa kata tersebut memang salah dalam proses penulisannya. Efisiensi waktu yang dibutuhkan jika dilakukan dengan manual tentunya tidak akan optimal dan cukup membosankan sehingga kemungkinan adanya *human error* dapat mengakibatkan proses pengecekan kata menjadi tidak optimal.

Untuk melakukan pengecekan secara otomatis terhadap kesalahan penulisan dibutuhkan suatu sistem yang disebut *spelling correcting* Bahasa Indonesia dengan *jaro winkler*. *Spelling Correcting* adalah suatu fitur yang dapat memeriksa dan memperbaiki kesalahan

Received September 29, 2023; Received Oktober 2, 2023; Accepted November 27, 2023

penulisan kata secara otomatis. Menurut (Gueddah, Yousfi, & Belkasmi, 2016) pemeriksaan ejaan terdiri dari perbandingan antara kata yang salah dengan daftar kata pada basis data dan menyarankan kata-kata yang mirip dengan kata yang salah dengan menghitung kemiripan jarak antara kata-kata tersebut.

Penelitian oleh (Adriyani, Santiyasa, & Muliantara, 2012) bertujuan untuk menampilkan saran perbaikan kesalahan pengetikan dokumen berbahasa Indonesia. Penelitian lain oleh (Prasetyo, Baihaqi, & Had, 2018) menggunakan *autocorrect* untuk memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis. Berdasarkan penelitian yang telah dilakukan bahwa banyaknya kesalahan pada penulisan kata atau kalimat. Penyebab kesalahan penulisan tersebut menurut (Adriyani, Santiyasa, & Muliantara, 2012) dapat disebabkan karena letak huruf pada keyboard yang berdekatan, kesalahan karena kegagalan mekanis atau slip dari tangan atau jari, serta kesalahan yang disebabkan karena ketidaksengajaan. Sebagai contoh pada kata 'sya' yang penulisan sebenarnya adalah 'saya' kesalahan dalam penulisan ini biasa disebut dengan *typo*.

Penelitian oleh (Tannga, Rahman, & Hasniati, 2017) melakukan perbandingan algoritma *levenshtein distance* dan *jaro winkler* dimana hasil analisis perbandingan yang didapat untuk rata-rata *similarity* dari algoritma *jaro winkler* yaitu sebesar 80.92 % sedangkan untuk algoritma *levensthein distance* rata-rata nilai similarity-nya yaitu sebesar 49.43%.

Penelitian ini bertujuan untuk mengoreksi kesalahan kata bahasa Indonesia dengan metode *Jaro winkler*. *Jaro winkler* merupakan salah satu algoritma yang digunakan untuk proses koreksi kata dengan menghitung nilai yang didapat dari hasil operasi modifikasi satu kata dengan kata yang lain dengan bantuan matriks. Algoritma *jaro winkler* selain banyak digunakan algoritma ini juga dapat menghasilkan tingkat akurasi yang bagus, sehingga hasil yang didapat juga rata-rata berhasil. Namun algoritma *jaro winkler* ini hanya sebatas pengecekan kesalahan ejaan dengan kata lain *typo* dan bukan pengecekan terhadap pola kalimat.

2. Metode Penelitian

2.1 Algoritma *Jaro Winkler*

Algoritma *Jaro Winkler* adalah sebuah algoritma untuk mengukur kesamaan antar 2 string. Semakin mirip antara 2 string tersebut maka semakin tinggi pula Jaro Distance. Nilai normal pada algoritma *Jaro Winkler* ialah 0 dan 1. Nilai 0 berarti tidak ada kesamaan antara *string* yang dibandingkan, dan jika mendapat nilai 1 artinya ada kesamaan antara *sting* yang dibandingkan.

Algoritma *Jaro Winkler* memiliki tiga bagian dasar yaitu :

1. Menghitung panjang string kata.
2. Menemukan jumlah karakter yang mirip pada dua string.
3. Menemukan jumlah transposisi.

Berikut adalah rumus yang digunakan pada algoritma *Jaro Winkler* untuk menghitung jarak (dj) antara dua string *S1* dan *S2.*

$$d_j = \frac{1}{3} * \left(\frac{m}{s_1} + \frac{m}{s_n} + \frac{m-t}{m}\right)$$

Keterangan:

Dj = Nilai *Jaro Winkle*.
m = Jumlah karakter yang sama pada string yang dibandingkan.
$|s1|$ = Jumlah karakter *String* 1.
$|s2|$ = Jumlah karakter *String* 2.
t = Setengah dari jumlah karakter yang posisinya tertukar (*transposisi*).

Pada algoritma Jaro *Winkler* nilai jarak maksimal adalah 1, yang artinya ada kesamaan antar *string* yang dibandingkan mencapai angka seratus persen. Setelah mendapat nilai dj, lalu nilai tersebut diimplementasikan dalam perhitungan *Jaro winkler distance* menggunakan rumus sebagai berikut

$$d_w = d_j + (l_p(1\text{-}d_j))$$

Keterangan :

dj = *Jaro distance*.
l = Panjang *prefix* maksimal 4 karakter (panjang karakter yang sama sebelum ditemukan karakter yang tidak sama).
p = konstanta *scaling factor* yang sudah ditetapkan oleh *Winkler* yaitu 0.1.

2.2 *Text Preprocessing*

a. Case Folding

Tahap *case folding* dilakukan dengan cara mengubah semua teks dokumen menjadi *lower case.* Adapun contoh dari penerapan case folding dapat dilihat pada table 1.

Tabel 1. Contoh Penerapan *Case Folding*

Teks Data Uji	Teks Hasil *Case Folding*
Halaman Kamus Seperti Pada Gambar Tersebut Digunakan Untuk Memasukkan Data Pada KBBI	halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data pada kbbi

b. Tokenizing

Tokenizing merupakan proses memisahkan teks yang terdapat dalam dokumen menjadi kata per kata. Adapun contoh tahap *tokenizing* dapat dilihat pada tabel 2.

Tabel 2. ContohPenerapan*Tokenizing*

Teks Data Uji	Teks Hasil Penenerapan ***Tokenizing***
halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data pada kbbi	halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data pada kbbi

c. ***Stopword Removal***

Stopword removal merupakan proses untuk menghilangkan kata yang tidak relevan atau tidak bermakna. Cara menentukan kata yang tidak relevan adalah dengan membandingkan pada *stoplist* yang sudah ditentukan. Daftar kata *stoplist* berdasarkan teks data uji bisa dilihat pada tabel 3.

Tabel 3. Daftar Kata *Stopword*

Teks Data Uji	Teks yang mengandung ***stopword***
halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data kata pada kbbi	seperti pada tersebut digunakan untuk pada kbbi

d. *Filtering*

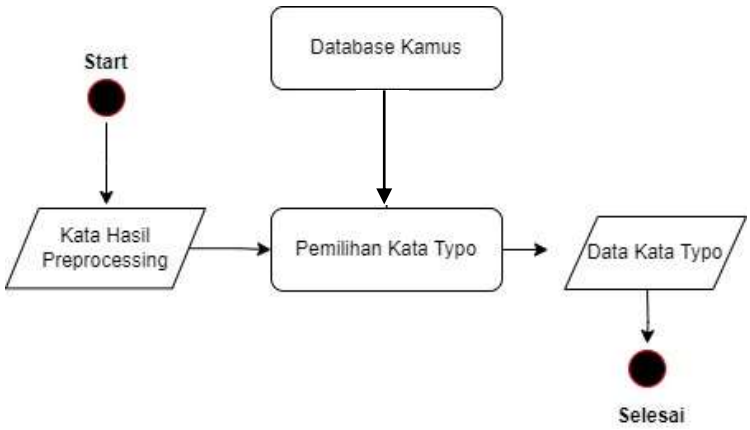
Filtering adalah proses membuang kata duplikat atau kata yang mengandung stopword. Contoh penerapan tahap filtering dapat dilihat pada tabel 4.

Tabel 4. Contoh Penerapan *Filtering*

Teks Data Uji	Teks hasil *filtering*
halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data pada kbbi	halaman kamus gambar memasukkan data

2.3 Pemilihan Kata *Typo*

Pemilihan kata yang salah atau *typo* dilakukan setelah proses *preprocessing*, yaitu setiap kata yang merupakan hasil *preprocessing* akan dicocokan dengan kata yang terdapat pada *database*, jika kata tersebut tidak sesuai dengan *database* maka akan dianggap sebagai kata *typo*.



Gambar 1. *Flowchart* Pemilihan Kata *Typo*

3. Results and Analysis

3.1 Perhitungan *Jaro Winkler*

Proses *spelling correcting* pada penulisan bahasa Indonesia dengan metode *jaro winkler* dengan menuliskan kalimat ”Alaman Amus Seperti Pada Gambar Tersebut Digunakan Untuk Memasukkan Data Pada KBBI”. Proses perhitungan metode *jaro winkler* pada *spelling correcting* pada penulisan bahasa Indonesia yaitu

1. *Text processing* kalimat bahasa Indonesia yang dimasukkan yang meliputi proses *lower case* dan tidak akan memproses kata yang mengandung *stopword* dan kata yang tidak terdapat dalam kamus seperti Tabel 5.

Tabel 5. Text Preprocessing Bahasa Indonesia

Kata	Lowercase	Stopword
Alaman	alaman	-
Amus	amus	-
Seperti	seperti	seperti
Pada	pada	pada
Gamba	gamba	-

Tersebut	tersebut	tersebut
Digunakan	digunakan	digunakan
Untuk	untuk	untuk
Memasukkan	memasukan	-
Data	data	-
Pada	pada	pada
KBBI	kbbi	kbbi

Dari tabel 5 didapatkan kata yang akan diproses dengan metode *jaro winkler* yang diperlihatkan seperti tabel 6.

Tabel 6. Hasil Text Prerocessing

Kode	**Kalimat**
S1	alaman
S2	amus
S3	gamba
S4	memasukkan
S5	data

2. Menghitung prosentase kemiripan berdasarkan jarak antar kata dengan *jaro winkler*. Langkah – langkah perhitungan prosentase dijelaskna sebagai berikut:

a. Hitung Jarak S1
Perhitungan jarak antara kata S1 diperlihatkan seperti Tabel 7.

Tabel 7.. PerhitunganJarak S1

S	**String**	**S**	**M**	**T**
S1	alaman	6	6	0
Sn	halaman	7		
			Jaro	0,952
			Jaro Winkler	0,971

1) Panjang *string* (S) pada S1 (alaman) adalah 6.
2) Panjang string (S) pada Sn (halaman) adalah 7.
3) Nilai M (karakter yang sama) antara S1 dan Sn adalah kata alaman dengan jumlah karakter adalah 6.
4) Nilai Jaro dihitung dengan menggunakan rumus

$$d_j = \frac{1}{3} * \left(\frac{m}{S_1} + \frac{m}{S_n} + \frac{m-t}{m}\right)$$

$$d_j = \frac{1}{3} * \left(\frac{6}{6} + \frac{6}{7} + \frac{6-0}{6}\right)$$

$$d_j = 0{,}952$$

5) Nilai Jaro Winkler dihitung dengan menggunakan rumus

$d_w = d_j + (l_p(1-d_j))$ dimana p adalah 0,1 dan panjang l maksimal adalah 4.

$$d_w = 0{,}952 + (4 * 0{,}1 * (1-0{,}952))$$

$$d_w = 0{,}971$$

b. Hitung Jarak S2
Perhitungan jarak antara kata S2 diperlihatkan seperti Tabel 8.

Tabel 8. Perhitungan Jarak S2

S	String	S	M	T
S2	amus	5	5	0
Sn	kamus	6		
			Jaro	0,944
			Jaro Winkler	0,967

1) Panjang *string* (S) pada S2 (amus) adalah 5.
2) Panjang *string* (S) pada Sn (kamus) adalah 6.
3) Nilai M (karakter yang sama) antara S2 dan Sn adalah kata amus dengan jumlah karakter adalah 5.
4) Nilai *Jaro* dihitung dengan menggunakan rumus

$$d_j = \frac{1}{3} * \left(\frac{m}{S_2} + \frac{m}{S_n} + \frac{m-t}{m}\right)$$

$$d_j = \frac{1}{3} * \left(\frac{5}{5} + \frac{5}{6} + \frac{5-0}{5}\right)$$

$$d_j = 0{,}944$$

5) Nilai *Jaro Winkler* dihitung dengan menggunakan rumus
$d_w = d_j + (l_p(1-d_j))$dimana p adalah 0,1dan panjang l maksimal adalah 4.

$$d_w = 0{,}944 + (4 * 0{,}1 * (1-0{,}944))$$

$$d_w = 0{,}967$$

c. Hitung Jarak S3
Perhitungan jarak antara kata S3 diperlihatkan seperti Tabel 9.

Tabel 9. Perhitungan Jarak S3

S	String	S	M	T
S3	gamba	5	5	0
Sn	gambar	6		
			Jaro	0,944
			Jaro Winkler	0,967

1) Panjang *string* (S) pada S3 (gamba) adalah 5.
2) Panjang *string* (S) pada Sn (gambar) adalah 6.
3) Nilai M (karakter yang sama) antara S3 dan Sn adalah kata gamba dengan jumlah karakter adalah 5.
4) Nilai *Jaro* dihitung dengan menggunakan rumus

$$d_j = \frac{1}{3} * \left(\frac{m}{S_3} + \frac{m}{S_n} + \frac{m-t}{m}\right)$$

$$d_j = \frac{1}{3} * \left(\frac{5}{5} + \frac{5}{6} + \frac{5-0}{5}\right)$$

$$d_j = 0{,}944$$

5) Nilai *Jaro Winkler* dihitung dengan menggunakan rumus
$d_w = d_j + (l_p(1-d_j))$dimana p adalah 0,1dan panjang l maksimal adalah 4.

d_w = 0,944 + $(4 * 0{,}1 * (1\text{-}0{,}944))$

d_w = 0,967

d. Hitung Jarak S4

Perhitungan jarak antara kata S4 diperlihatkan seperti Tabel 10.

Tabel 10. Perhitungan Jarak S4

S	String	S	M	T
S4	memasukkan	10	10	0
Sn	memasukkan	10		
			Jaro	1,000
			Jaro Winkler	1,000

1) Panjang *string* (S) pada S4 (memasukkan) adalah 10.
2) Panjang *string* (S) pada Sn (memasukkan) adalah 10.
3) Nilai M (karakter yang sama) antara S4 dan Sn adalah kata memasukkan dengan jumlah karakter adalah 10.
4) Nilai *Jaro* dihitung dengan menggunakan rumus

$d_j = \frac{1}{3} * \left(\frac{m}{S_4} + \frac{m}{S_n} + \frac{m-t}{m}\right)$

$d_j = \frac{1}{3} * \left(\frac{10}{10} + \frac{10}{10} + \frac{10-0}{10}\right)$

d_j = 1,000

5) Nilai *Jaro Winkler* dihitung dengan menggunakan rumus

$d_w = d_j + (l_p(1\text{-}d_j))$ dimana p adalah 0,1dan panjang l maksimal adalah 4.

d_w = 1,000 + $(4 * 0{,}1 * (1\text{-}1{,}000))$

d_w = 1,000

e. Hitung Jarak S5

Perhitungan jarak antara kata S5 diperlihatkan seperti Tabel 11.

Tabel 11. Perhitungan Jarak S5

S	String	S	M	T
S5	data	4	4	0
Sn	data	4		
			Jaro	1,000
			Jaro Winkler	1,000

1) Panjang *string* (S) pada S5 (data) adalah 4.
2) Panjang *string* (S) pada Sn (data) adalah 4.
3) Nilai M (karakter yang sama) antara S4 dan Sn adalah kata data dengan jumlah karakter adalah 4.
4) Nilai *Jaro* dihitung dengan menggunakan rumus

$d_j = \frac{1}{3} * \left(\frac{m}{S_5} + \frac{m}{S_n} + \frac{m-t}{m}\right)$

$d_j = \frac{1}{3} * \left(\frac{4}{4} + \frac{4}{4} + \frac{4-0}{4}\right)$

d_j = 1,000

5) Nilai *Jaro Winkler* dihitung dengan menggunakan rumus
d_w = d_j + (l_p(1-d_j))dimana p adalah 0,1dan panjang l maksimal adalah 4.

d_w = 1,000 + $(4 * 0{,}1 * (1\text{-}1{,}000))$

d_w = 1,000

Setelah dilakukan prosentase kemiripan berdasarkan jarak antar kata dengan *jaro winkler* kemudian kata disusun menjadi kalimat utuh berdasarakan urutan awal kalimat dan menghasilkan kalimat halaman kamus seperti pada gambar tersebut digunakan untuk memasukkan data kata pada kbbi.

4. Kesimpulan dan Saran

4.1 Kesimpulan

Dalam penelitian ini dapat ditarik suatu simpulan sebagai berikut:

1. Algoritma jaro winkler dapat digunakan untuk melakukan *spelling correcting* bahasa Indonesia sebagai pengukur tingkat similaritas antara kata yang dimasukkan dengan kata yang ada dalam tabel dan diurutkan dari nilai kemiripan yang paling besar sampai dengan nilai kemiripan yang paling kecil.
2. Metode *jaro winkler* dapat digunakan untuk melakukan *spelling correcting* bahasa Indonesia dengan maksimal *correcting* 1000 kata.
3. Berdasarkan hasil uji coba pada 13 kata, aplikasi *spelling correcting* ini dapat memunculkan 3 perbaikan kata dengan tepat, setiap perbaikan kata dihitung mengunakan rumus *jaro winkler* mendapat nilai diatas 0.94 untuk setiap kata yang diuji.

4.2 Saran

Berdasarkan permasalahan, analisa, dan kesimpulan diatas, maka penulis berusaha memberikan saran-saran yaitu untuk kedepannya, melakukan *spelling correcting* bahasa Indonesia dapat ditambahkan *spelling correcting* untuk bahasa Jawa atau bahasa Inggris.

Daftar Pustaka

[1] Frando, J., Ruslianto, I., Hidayati, R., Penerapan Jaro Winkler Distance Dalam Aplikasi Pengoreksi Kesalahan Penulisan Bahasa Indonesia Berbasis Web. Jurnal Komputer dan Aplikasi. Vol. 7. No. 3. 2019.

[2] Gueddah, H., Yousfi, A., & Belkasmi, M. The filtered combination of the weighted edit distance and the Jaro-Winkler distance to improve spellchecking Arabic texts. *AICCSA*,1-6.2016

[3] Prasetyo, A., Baihaqi, W., & Had, I. lgoritma Jaro-Winkler Distance: Fitur Autocorrect Dan Spelling Suggestion Pada Penulisan Naskah Bahasa Indonesia di BMS TV. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK); V*(4), 435-444.2018

[4] Sujani, H., Muhardi, H., Simanjuntak, J. Aplikasi Pengoreksi Ejaan (Spelling Correction) pada Naskah Jurnal Bidang Informatika Dengan N-Gram dan Jaro-Winkler Distance.Jurnal Edukasi dan Penelitian Informatika, Vol.8, 2022.

[5] Pinajeng, I., Sukarsa, I., Putra, I., Perbaikan Kata Pada Sistem Chatbot dengan Metode Jaro Winkler. Jurnal Ilmiah Teknlogi dan Komputer; Vol. 1, No. 2, 2020.

[6] Londo, G., Purnomo, Y., Maslim, M. Pembangunan Aplikasi Identifikasi Kesalahan Ketik Dokumen Berbahasa Indonesia Menggunakan Algoritma Jaro-Winkler Distance. Juita : Jurnal Informatika. Vol. 8. No. 1.2020

[7] Simanjuntak, M., Sujiani, H., Safriadi, N. Spelling Corrector Bahasa Indonesia dengan Kombinasi Metode Peter Norvig dan N-Gram. Jurnal Edukasi dan Penelitian Informatika. Vol. 4. No. 1. 2018

[8] Tannga, M., Rahman, S., & Hasniati. Analisis Perbandingan Algoritma Levenshtein.

[9] Distance Dan Jaro Winkler Untuk Aplikasi Deteksi Plagiarisme Dokumen Tek. *JTRISTE, IV*(1), 44-54. 2017.