PERBANDINGAN ALGORITMA *EDIT DISTANCE, LEVENSHTEIN DISTANCE, HAMMING DISTANCE, JACCARD SIMILARITY* DALAM MENDETEKSI *STRING MATCHING*

Susi Rianti[1], Riza Adrianti Supono[2]
[1,2] Universitas Gunadarma
[1]susirianti06@gmail.com, [2]adrianti@staff.gunadarma.ac.id,

Abstract

Saat sekarang ini dunia membutuhkan pencaraian yang cepat, tepat dan efisien. Pencarian yang dilakukan Penulis tertarik untuk membandingkan algoritma string matching karena belum ada yang membandingkan empat algoritma yaitu algoritma *Edit Distance, Levenshtein, Hamming* dan *Jaccard similarity*. Tujuan penulisan ini membandingkan kecepatan akses 4 (empat) algoritma yaitu, *algoritma Edit Distance, Levenstein Distance, Humming Distance*, dan *Jaccard Similarity* mana yang lebih cepat, tepat dengan tingkat error yang rendah, menentukan sililaritas dokumen, dalam pencarian. Hal ini diakibatkan karena sering terjadi kesalahan pengetikan sehingga terjadi kesalahan ejaan. Kesalahan pengetikan ini biasanya terjadi saat adalah, penghapusan, penambahan, pengantian serta modofikasi data. Penulis mengunakan data dari kamus besar bahasa Indonesia yang meliputi kata dasar, kata ber-awalan, kata ber-akhiran, kata ber-imbuhan, kata sambung, kata depan, dan kata yang termasuk dalam kata bahasa baku. Metode evaluasi yang dipakai dalam penelelitian ini adalah mAP bahasa pemograman yang digunakan adalah C++. Berdasarkan ujicoba yang dilakukan rata-rata waktu pencarian terhadap algorima Edit Distance = 47,55 ms, Levensthtein Distance = 13,125 ms, Hamming Distance = 14,25 ms, dan Jaccard similarity = 37,125 ms Ini berarti Levenshtein Distance memiliki waktu pencarian tercepat.

Keywords: Edit DistanceHumming Distance, Jaccard, Levenhstain Distance, Perbandingan algoritma

PENDAHULUAN

Teknologi informasi sekarang ini teknologi yang dipakai untuk menyusun, menyimpan, memanipulasi data dalam berbagai cara untuk menghasilkan informasi yang berkualitas. Saat ini dalam kehidupan sehari-hari kita membutuhkan akses yang cepat, tepat dan efisien dalam pencarian informasi. Saat pencarian informasi sering terjadi kesalahan pengetikan. Jenis kesalahan pengetikan ini seperti terjadi pengetikan penambahan huruf, terjadi kesalahan pengetikan penenpatan huruf yang salah, terjadi kesalahan pengetikan penghapusan huruf dan terjadi juga saat modifikasi. Masalah lain yang juga terjadi adalah plagiarisme. Plagiarisme banyak sekali terjadi. Salah satu cara untuk mendapatkan akses informasi yang cepat, tepat, mengecek plagiarisme, adalah dengan diimplentasikan algoritma. Pengimplementasian algoritma juga menimbulkan masa lah. Masalah yang timbul adalah algoritma mana yang akan dipilih yang mempunyai waktu akses yang paling cepat, tingkat ketepatan hitung, tingkat error yang rendah dalam mencari informasi. Algoritma yang akan dibahas, dibandingkan dalam penelitian ini adalah algorima *Ediit Distance, Hamiing Similarity, Levenhtein Distance,* dan algorima *Jaccard Similarity*.

METODE PENELITIAN

Algoritma untuk mendeteksi bermacam-macam masalah seperti bioinformatics *(DNA analysis), text prosesing* (seperti pengecekan ejaan, pendeteksian plagiarisme, koreksi *error*), *information retrival*, pemrosesan sinyal, *speech recogniton* dan *web mining* hal ini dilakukan dengan cara membandingkan dua string, membandingkan string atau *String matching*. (Yousaf *et al.*, 2018). *String maching* adalah bagian dari pemrosesan bahasa alami integral dengan *multimedia information retrival* yang fokus pada deteksi dalam mengenali text dan video, image documen dan sosial media. Plagiarisme dapat dilakukan dengan mudah dengan bantuan koneksi internet (Hakak *et al.*, 2019). Plagiarisme adalah menggunakan literatur, pemikiran dan informasi orang lain tanpa menyebutkan sumber aslinya. Penelitian terkait penggunaan rumus jarak *(distance)* dalam plagiarisme menggunakan *Edit Distance, Levenstein Distance, Humming Distance* dan *Jaccard Similarity* dilakukan peneliti sebelumnya.

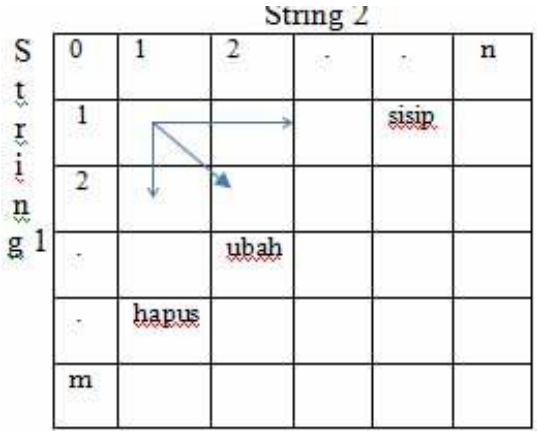
Edit Distance

Edit Distance adalah algoritma yang sering digunakan untuk menyisipkan, menghapus, mengubah string dengan perhitungan matriks (Awaludin, 2015). Penghitungan *Edit Distance* dimulai dari nilai awal yang terkecil, kemudian diproses hingga selesai dilakukan perhitungan. Operasi perhitungan matrik akan memperlihatkan berapa kali dilakukan perubahan terhadap posisi. Algoritma *Edit Distance* melaku-kan operasi perubahan terhadap karakter string. Operasi tersebut adalah operasi penghapusan karakter, operasi penambahan karakter, operasi penghapusan karakter.

1. Pengubahan Karakter.
   Pengubhan karakter terjadi akibat salah pengetikan. Contoh kata pulajg. Kata pulajg tidak memiliki arti dalam bahasa Indonesia. Algoritma *Edit Distance* akan mengubah kata pulajg menjadi pulang. Proses pengubahannya adalah string huruf "j" diubah menjadi string huruf "n". Kata akhir setelah pengubangan ini menjadi "pulang".
2. Penambah an Karakter.
   Operasi ini terjadi bila terjadi kesalahan ketidak sempurnaan pengetikan oleh user. Contoh kata "Bers". Kata "Bers" dalam bahasa Indonesia tiadak memiliki arti. Kat "Bers" agar mempunyai arti ditambahkan huruf "a", sehingga menjadi "Beras".

3. Penghapusan Karakter.
   Penghapusan karakter dilakukan ketika pengguna melakukan pengetikan melebihi panjang kata yang ingin diketik. Contoh "sayangs", penghapusan string "s" akan mengakibatkan string tersebut jadi lebih relevan. Penghapusan string "s", sehingga kata "sayangs" menjadi "sayang".

Tabel 2.4. *Edit Distance*



Keterangan :
S1 = String 1, S2 = String 2

Perhitungan tabel diatas mengunakan rumus sebagai berikut:

(n+1) x (m+1) dimana : n = panjang string 1, m= panjang string 2
S1 adalah variable string yang akan diperbaiki. Variable string S1 akan dibandingkan dengan variable string S2.S2 adalah string yang digunakan sebagai refensi S1, saat proses perbandingan dan pengubahan dilakukan. (Iskandar, 2018)

Levenshtein Distance.

Dalam Lavenshtein *distance "source string* (s) adalah *similarity* dari dua dokumen, *dan destination string* (t). Jumlah spasi dalam penghapusan, atau-pun subtitusi dibutuhkan untuk meng-konversi s menjadi t. *String* yang menjadi berbeda adalah peningkatan dari lavenshtein distance. *Source string* dapat dijadikan input dan target string adalah masukan yang diberikan pada text. (Hakak *et al.*, 2019)."

Berikut adalah contoh algoritma levenshtein dengan pengunaan kata yang benar dan kata yang salah S: ALGORITMH → m = 9, T : ALGORRIHM → 10.

Code1. Algoritma levenshtein distance

```
for i ← 1 to m do {source prefixes initialization}
    d[i] [0] ← i
endfor
for j ← 1 to n do {target prefixes initialization}
    d[0] [j] ← j
endfor

{using Levenshtein Algoritm to check]
for i ← 1 to n do
  for j ← 1 to m do
     if (s[i] == t[j])
        then
        d[i] [j] ← d[i-1] [j-1] {same character}
      else
        d[i] [j] ← minimum
                 (
                  d[i-1] [j] + 1,{ deletion}
                  d[i] [j-1] + 1,{ insertion}
                  d[i-1] [j-1] + 1 , {subtitution}
                 )
      endif
    endfor
  endfor
→ d[m] [n] {return}
```

Hamming Distance

Algoritma "Hamming Distance" merupakan metode untuk menghitung jarak antara dua string yang mempunyai ukuran sama. *Hamming Distance* menghitung jarak dua string dari banyaknya simbol dalam dua cicncin yang berbeda. Cara kerja *Hamming Distance* yaitu dengan mengukur jarak dua buah string yang mempunyai ukuran sama, dengan membandingkan simbol-simbol yang terdapat pada kedua buah string pada posisi yang sama (Awaludin, 2015). Tahapan hitung pada *HammingDistance* yaitu:
Jika diberikan :X = sate ⟶ | | = n = 4
y soto ⟶ | y| = n =4

maka cara hitung adalah sebagai berikut :
n = 4 > 0
i =1
j =1, i =*j*, $^{x}1$ = s =y1, maka $C_{1,1} = C_{1,1} = C_{0,0} + 0 = 0$
j= 2, i ≠ j, maka $C_{1,2} = 0$.
j = 3, i ≠ j, maka $C_{1,3} = 0$.
j = 4 n, i ≠ j, maka $C_{1,4} = 0$.
i = 2.
j = 1, i ≠ j, maka $C_{2,1} = 0$
. j = 2,i = j, X2 = a ≠ o = y2, maka $C_{2,2}$ = C1,1 + 1= 1.
j = 3, i ≠ j, maka $C_{2,3} = 0$.
j = 4 = n, i ≠ j, maka $C'_{2,4} = 0$.
i = 3.
j = 1, i ≠ j, maka $C_{3,1} = 0$.
j = 2, i ≠ j, maka $C_{3,2} = 0$.

j = 3, i = j, X3 = t = y3, maka $C_{3,3} = C_{2,2} + 0 = 1$.
j = 4, i ≠ j, maka $C_{3,4} = 0$.
i = 4 = n.
j = 1, i ≠ j, maka $C_{4,1} = 0$.
j = 2, i ≠ j, maka $C_{4,2} = 0$.
j = 3, i = j, maka C4,3 = 0.
j = 4 = n, i ≠ j, X4 = e ≠ o = y4, maka $C_{4,4} = C_{3,3} + 1 = 2$.

Output : 2. Artinya, jarak antara x dan y adalah 2. Tabel perhitungan jarak Hamming pada contoh diatas dengan menggunakan tabel:

Tabel 2.6. Jarak Hamming

-	-	0	1	2	3	4
-	-	-	s	A	t	e
0	-	0	0	0	0	0
1	s	0	0	0	0	0
2	o	0	0	1	0	0
3	t	0	0	0	1	0
4	o	0	0	0	0	2

Terlihat dari contoh diatas sel Cn n = C4,4 = 2 ini berarti jarak Hamming antara teks x dan y adalah 2. (Gulo, 2022)

Jaccard Similarity.

Jaccard digunakan untuk menghitung .kemiripan kesamaan antar kalimat atau kata. Caranya dengan membagi jumlah kata yang berpotongaan atara dua teks . Dengan persamaan sebagai berikut :

Formula Jaccard :

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

A : Set A pada array data A
B: Set B pada array data B

Contoh Jaccard Similarity
A={1,2,3,4}, B={1.2,4], and C={1,2,4,5}
A ∩ B = {1,2,4}; |A ∩ B| =3
A ∪ B = {1,2,3,4}; |A ∪ B| = 4

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = 3/4$$

$$J(B,C) = \frac{|A \cap B|}{|A \cup B|} = 3/4$$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = 3/5$$

(Indriani *et al.*, 2018)

Perhitungan *Error*

Perbandingan antra kesalahan (*error*) dengan bit yang dikirimkan keseluruhan disebit dengan *Bit Error Rate (BER). Bit Error Rate (BER)* bertugas untuk menguji kessalahan pembacaan disisi penerima setiap detik. Misalkan terjadi kesalahan pembacaan atau penerimaan sebesar 1 *bit.* Nilai BER yang dipakai sebesai 10^{13}. Bentuk persamaannya adalah sebagai berikut:

$$BER = \frac{1}{2} \text{ercf}\left(\frac{Q}{\sqrt{2}}\right) \quad (1)$$

ercf adalah fungsi error seperti persamaan 2:

$$\text{ercf}(x) = \frac{1}{\sqrt{2}} \int_x^{\sim} e^{\frac{-x2}{2}} \_ \, dx$$

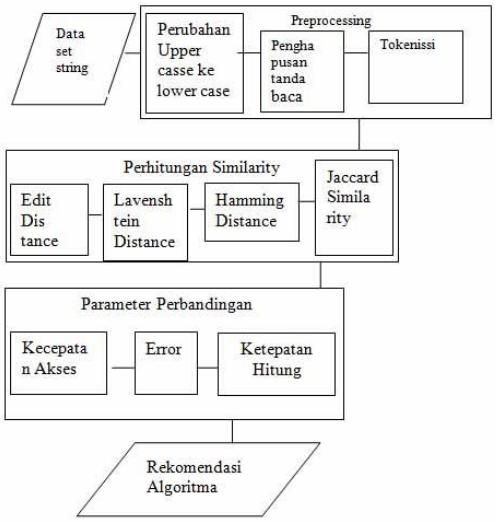
Didapat persamaan 3:

$$BER = Pe\ (Q) = \frac{1}{\sqrt{2}} \quad \frac{e^{\frac{-Q2}{2}}}{Q}$$ Q=Q faktor dan Pe = *Error Probability*

(Prakoso, Wahyudi and Masykuroh, 2021)

PEMBAHASAN

Metode penelitian memberikann rancangan penelitian yang meliputi langkah-langkah yang ditempuh, prosedur, sumber data, selanjutnya data diolah dan dianalisis.



Gambar 3.1. Metode penelitian

Dataset Penelitian

Penelitian ini mengunakanan data dari kamus besar bahasa Indonesia. Kata yang terdapat dalam penelitian ini meliputi kata dasar,kata ber-awalan, kata ber-akhiran kata ber-imbuhan, kata sambung, kata depan dan jenis kata yang termasuk dalam kata baku bahasa Indonesia.

Preeprocessing String

Preeprocessing adalah teknik yang digunakan untuk mengubah data mentah kedalam bentuk yang format yang berguna dan efisien.

Perubahan Uppercase ke Lowercase

Dalam sebuah dokumen mengan-dung bermacam-macam bentuk huruf sampai tanda baca. Semua huruf harus diseragamkan bentuknya. Proses ini merupakan proses ekrstaksi kata. Semua huruf besar (*uppercase*) bentuknya menjadi huruf kecil (lowercase). Hanya huruf 'a' sampai dengan huruf 'z' yang bisa diterima.

Penghapusan tanda baca

Sebelum mengolah data tulisan, menghapus tanda baca seperti (",?,.#!) dan lain-lain adalah proses yang harus dilakukan. Proses menghilangkan tanda baca ini juga digunakan untuk meng-hilangkan *noise* pada saat pengambilan informasi. Dengan proes ini data lebih bersih dan memiliki dimensi yang lebih kecil, lebih terstruktur, sehingga bisa diolah untuk proses selanjutnya. (Jumeilah, 2017).

Tokenisasi

Proses tokenisasi adalah proses pemotongan string input berdasarkan tiap kata kenyusunnya. Tokenisasi adalah proses memisahkan setaip kata yang menyusun suatu dokumen. Proses ini menghilangkan tanda baca dan karakter selain huruf alfabet, karena karakter-karekter tersebut dianggp se-bagai pemisah kata (delimeter) dan tidak memiliki pengaruh terhadap pemrosesan text (Awaludin and Yasin, 2020). Tahap ini juga dilakukan *cleaning* yaitu proes membersihkan dokmen dari komponen-komponen yang tidak me-miliki hubungan dengan informasi yang ada pada dokumen, seperti tag html, link dan *script* dan sebagainya.

Perhitungan *similarity*

Plagiarisme dapat dideteksi dengan konsep *similarity* atau kemiripan doku-men. *Similarity* mendetekksi *copy* dan *paste* dan *disguised plagiarism*. Proses algoritma yang dibahas adalah Levenshtein *Distance*, *Hamming Distance*, Minimum Edit *Distance*, dan Jaccard *Similarity.*

Parameter Perbandingan Algoritma

Faktor yang mempengaruhi kinerja pencocokan string adalah bergantung pada urutan pemindaian pola. Masalah penting dalam pencocokan string adalah evaluasi kinerja. (Ryu and Park, 2018). Evaluasi kinerja dalam penulisan ini menggunakan mAP (*mean Average Precision).* Dalam penelitian ini akan dihitung rata-rata dan keseluruhan *average precision.*

Kecepatan *Acsess*

Performa sebuah algoritma dipengaruhi banyak hal. Antara lain dipengaruhi oleh dataset. Dataset yang besar akan mempengaruhi kecepatan dari sebuah algoritama. Performa juga dipengaruhi oleh praktek pengkodean yang baik. Dan yang tidak kalah penting adalah pemilihan algoritma yang akan dipakai. Pemilihan algoritma yang baik memiliki dampak yang besar dari pada performa. Dalam membandingkan 2 (dua) buah algoritma atau lebih maka ada beerapa hal akan jadi bahan pertimbangan. Hal-hal yang harus dipertimbangkan dalam membandingkan algoritma sebagi berikut:

a. Hardware
Menghitung jumlah sebuah operasi adalah sesuatu yang tidak tergantung pada kecepatan prosesor.

b. Dataset properti
Dataset properti adalah data yang hampir diurutkan, semua data mempunyai satu nilai, data yang diurutkan terbalik. Ambil data set yang terburuk, yang akan memberikaan nilai atas dalam menghitung berapa lama algoritma akan diproses. Serta tidak mudah untuk memutuskan apa dan rata-rata himpunan data.

c. Ukuran dataset
Mengambarkan *running time* dari sebuah algoritma sebagai sebauah fungsi dari dari ukuran data set. (Cse, 2017)

Error

Menghitung *error rate* menggunakan rumus:

$$\textit{Error Rate} \% = \frac{Jumlah\ pengurutan\ data\ salah}{total\ jumlah\ pengurutan\ data} * 100$$

(Poetra and Hayati, 2022)

Ketepatan Hitung

Menghitung akurasi dengan rumus:

Akurasi % =100% - *Error Rate* %

Errir rate adalah perbandingan banya.knya unit data yang salah dengan keseluruhan data.

(Poetra and Hayati, 2022)

Hasil Ekperimen dan Analisa

Data

Data yang digunakan dalam penulisan ini adalah data yang digunakan untuk pengujian. Data-data tersebut adalah data karena terjadi penyisipan huruf, terjadi penghapusan huruf, kesalahan penulisan huruf, kesalahan karena terjadi penukaran huruf. Dari 50 kata salah terbagi menjadi 12 kata pada jenis kesalahan pertama, 12 kata pada jenis kesalahan kedua, 13 kata jenis kesalahan ketiga. Semua jenis kesalahan diproses dengan mencocokkan kata berdaasarkan algoritma Edit Distanc, Hamiing Distance, Levensshtein Distance, dan Jaccard Similarity yang diperlihatkan dalam tabel berikut ini.

Tabel 1.Kesalahan Penulisan Penghapusan Huruf

No	Kata ke	Kata yang salah	Kata yang benar
1.	Kata 1	du	Dua
2.	Kata 2	etuju	Setuju
3	Kata 3.	produsi	Produksi
4.	Kata 4.	sar	Sari
5.	Kata 5	car	Cari
6.	Kata 6	da	Dan
7.	Kata 7	denga	dengan
8.	Kata 8	denga	dengar
9.	Kata 9	cum	cuma
10	Kata 10	car	Cara

Tavel 2 Kesalahan Penulisan karena penambahan huruf

No.	Kata ke	Kata yang salah	Kata yang benar
1.	Kata 1	ssoleha	soleha
2.	Kata 2	pesertaa	peserta
3	Kata 3.	haarapa n	harapan
4.	Kata 4.	selurruh	seluruh
5.	Kata 5	carii	cari
6.	Kata 6	dann	dan
7.	Kata 7	dengaan	dengan
8.	Kata 8	dengarr	dengar
9.	Kata 9	cumii	cumi
10	Kata 10	rambuta	rambut

Tabel 3.Kesalahan Penulisan Karena Pengantian Huruf

No	Kata ke	Kata yang salah	Kata yang benar
1.	Kata 1	kireta	kereta
2.	Kata 2	puntai	pantai
3	Kata 3.	kalapa	kelapa
4.	Kata 4.	monis	manis
5.	Kata 5	kepola	kepala
6.	Kata 6	saun	daun
7.	Kata 7	rindah	rendah
8.	Kata 8	tanggi	tinggi
9.	Kata 9	pircu ma	percuna
10	Kata 10	begus	bagus

Tavel 4 Kesalahan Penulisan karena penukaran huruf

No.	Kata ke	Kata yang salah	Kata yang benar
1.	Kata 1	capugn	capung
2.	Kata 2	suumr	sumur
3	Kata 3.	sllap	sulap
4.	Kata 4.	renad	renda
5.	Kata 5	canitk	cantik
6.	Kata 6	hodna	honda
7.	Kata 7	ubnga	bunga
8.	Kata 8	tarda	datar
9.	Kata 9	ogsok	gosok
10	Kata 10	petkar	karpet

Skenario Eksperimen

Eksperimen ini menggunakan dua skenario yang berbeda. Eksperimen satu mengujicobakan 50 kata yang salah sebagai data masukkan untuk proses identifikasi kesalahan pengetikan dan saran perbaikannya berdasarkan algo-ritma Hamming Distan ce, Lavenshtein Distance, Jaccard Similarity dan Edit Distance yang selanjutnya dihitung nilai MAP (Mean Average Precision).

Eksperimen kedua dilakukan dalam lingkungan yang sama dengan meng-gunakan 50 kata yang salah yang disebabakan oleh kesalahan penggetikan yang terbagi dalam empat katagori kesalahan yaitu kesalahan penghapusan huruf, kesalahn penambahan huruf, kesalahan penggantian huruf dan kesa-lahan penukaran huruf.Total kata adaa 50, yang terdiri dari 12 kata pada Tiap jenis kesalahn diproses dengan men-cocokkan kata dengan dalam kamus berdasarkan algoritma Humiing Distance, Levenshtein Distance, Jaccard Similarity dan algorima Edit Distance.

Hasil dan Analisa

Hasil dari eksperimen telah diteliti dan dianalisa (a) Hasil perhitungan rata-rata. Data yang digunakan adalah 50 data yang salah yang diproses dengan empat algoritma menghasilkan nilai MAP masing-masing. Dengan dua *user relevance judment* terlibat dalam menentukan kata saran yang muncul relevan atau tidak relevan. Hasil eksperimen menghitung rata-rata dapat dilihat dengan nilai tertinggi dihasilkan oleh algoritma *Levenshtein Distance* dengan nilai 0,87, dan yang terendah sedangkan yang terendah adalah algoritma *Edit Distance* dengan nilai 0,25. Berdasarkan eksperiment ini dapat diambil kesimpulan bahwa algoritma terbaik adalah

Tabel.5. Hasil Rata-Rata Total Pengujian

Algoritma	Hasil Proses mAP Eksperimen 1
Hamming Distance	0,64
Levenshtein Distance	0,87
Jaccard Similarity	0,30
Edit Distance	0,25

Tabel 6. Hasil Pengujian Eksperimen 2

Algoritama	Jenis Kesalahan			
	Hapus	Tambah	Ganti	Tukar
Hamming Distance	14,9 ms	14,6 ms	13,9 ms	14,2 ms
Levenshtein Distance	14,5 ms	11,4 ms	12,9 ms	13,7 ms
Jackard Similarity	41,4 ms	34,1 ms	39,6 ms	34 ms
Edit Distance	38,8 ms	44,3 ms	29.1 ms	39 ms

Menghitung *Error*

Pada penelitian ini diasumsikan bahwa, data yang dijadikan contoh adalah data yang salah. Apabila dalam kehidupan sehari-hari ada kalanya data yang dimasukkan adalah data benar, maka *error rate* tergantung dari masukan sistem.

Rumus untuk menghitung *error rate* adalah :

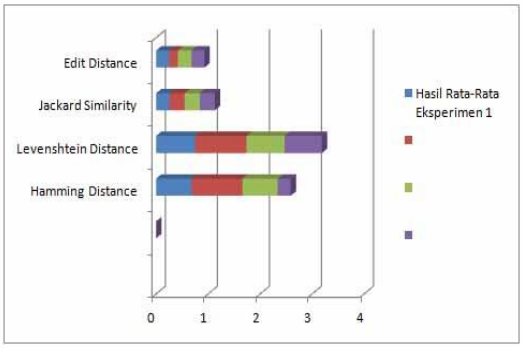
$$\textit{Error Rate}\ \% = \frac{Jumlah\ data\ salah}{total\ jumlah\ data} * 100$$

$$\text{Error rate} \quad 20\ \% \quad = \frac{10}{50} \text{ x } 100$$

Ketepatan hitung

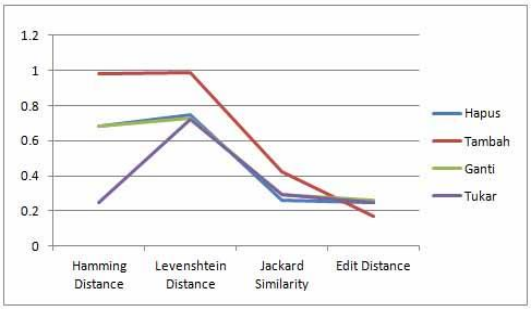
80% = 100% - 20%

Grafik MAP eksperimen 1



Gambar 1
Grafik perhitungan rata-rata eksperimen 1

Grafik MAP eksperimen 2



Gambar 2 Grafik perhitungan rata-rata eksperimen 2

SIMPULAN

Kesimpulan yang dapat diambil dari dari ekperimen ini dengan emapt algorima yaitu *Edit Distance, Hamming Distance, Levenshtein Distance dan Jaccard Similarity* dengan operasi penambahan huruf, penghapusan huruf, penggantian huruf dan penukaran huruf adalah nilai penghapusaan huruf yang memiliki nilai MAP paling baik adalah Levenshtein Distance yaitu dengan nilai 13,125 ms, yang menunjukkan akses paling cepat dalam penghapusan,

pengantian huruf, tambah dan tukar huruf. Dari ekperimen ke dua adalah Levenshtein Distance dengan nilai yang paling rendah yaitu dengan nilai 13,125. Algoritma yang direkomentasikan dari penelitian ini adalah algoritma *Levenshtein Distance.*

REFERENSI

Awaludin, M. (2015) 'Penerapan Metode Distance Transform Pada Linear Discriminant Analysis Untuk Kemunculan Kulit Pada Deteksi Kulit', *Journal of Intelligent Systems*, 1(1), pp. 49–55.

Awaludin, M. and Yasin, V. (2020) 'APPLICATION OF ORIENTED FAST AND ROTATED BRIEF (ORB) AND BRUTEFORCE HAMMING IN LIBRARY OPENCV FOR CLASSIFICATION OF e-ISSN : 2598-8719 (Online)', *Journal of Information System, Applied, Managemgent, Accounting, and Reserarch*, 4(3), pp. 51–59.

Cse, U. W. (2017) 'Algorithmic complexity : Speed of algorithms How fast does your program run ?', pp. 1–18.

Gulo, Y. N. (2022) 'Penerapan Algoritma Hamming Distance Untuk Pencarian Teks Pada Aplikasi Ensiklopedia Indonesia', 1(2), pp. 50–54.

Hakak, S. *et al.* (2019) 'Exact String Matching Algorithms ':, *IEEE Access*. IEEE, PP(c), p. 1. doi: 10.1109/ACCESS.2019.2914071.

Indriani, A. *et al.* (2018) 'IMPLEMENTASI JACCARD INDEX DAN N-GRAM PADA REKAYASA', 2621–069X, pp. 95–101.

Iskandar, I. D. (2018) 'IMPLEMENTASI ALGORITMA EDIT DISTANCE PADA PENGEMBANGAN APLIKASI E-LEARNING BSI'.

Prakoso, R. P., Wahyudi, E. and Masykuroh, K. (2021) 'Optimalisasi Bit Error Rate (BER) Jaringan Optik Hybrid Pada Sistem DWDM Berbasis Soliton', *Journal of Telecommunication, Electronics, and Control Engineering (JTECE)*, 3(2), pp. 62–70. doi: 10.20895/jtece.v3i2.320.

Ryu, C. and Park, K. (2018) 'Improved pattern-scan-order algorithms for string matching ☆', *Journal of Discrete Algorithms*. Elsevier B.V., 49, pp. 27–36. doi: 10.1016/j.jda.2018.05.002.

Sistem, R. (2017) 'JURNAL RESTI', 1(1), pp. 19–25.

Sort, B. and Sort, Q. (2022) 'Performa Algoritma Bubble Sort Dan Quick Sort Pada Framework Flutter Dan Dart SDK (Studi Kasus Aplikasi E-Commerce)', 9(2), pp. 806–816.

Yousaf, M. M. *et al.* (2018) 'Computation To cite this version ':