ID: 43

Perbandingan Kinerja Jaro Winkler dan Levensthein Distance pada Autocorrect Sistem Pencarian Hadis

Performance Comparison of Jaro Winkler dan Levensthein Distance on Autocorrect Hadith Search System

Ichsan Taufik[1*], Manarul Huda[2], Yana Aditia Gerhana[3]
[1,2,3] Jurusan Teknik Informatika Fakultas Sains dan Teknologi
UIN Sunan Gunung Djati Bandung
Jl. AH. Nasution No. 105 Bandung
ichsan@uinsgd.ac.id[1], manarulhuda57@gmail.com[2], yanagerhana@uinsgd.ac.id[3]

Abstrak – *Hadis merupakan hukum Islam kedua setelah Alquran. Banyaknya hadis yang tersebar di berbagai kitab hadis, mendorong umat Islam mencari cara untuk mendapatkan hadis yang tepat dalam waktu yang sesingkat-singkatnya. Salah satunya adalah Kitab Bulughul Maram yang berjumlah lebih kurang 1.371 buah hadis. Maka diperlukan implementasi untuk Sistem Pencarian Hadis. Namun saat dilakukan proses pencarian, kata kunci yang dipilih terkadang terjadi kesalahan penulisan. Oleh karenanya, diperlukan autocorrect untuk mengatasi hal tersebut. Kemudian, diperlukan analisa untuk mengetahui kinerja autocorrect pada Algoritma Jaro Winkler dan Levensthein Distance. Hasil yang didapatkan setelah 20 kali pengujian, akurasi Jaro Winkler sebesar 97,45%, lebih tinggi dibandingkan dengan akurasi Levensthein Distance yang sebesar 90,27%. Sedangkan untuk waktu tempuh, Levensthein Distance lebih singkat sebesar 0,09 detik dibandingkan Jaro Winkler sebesar 0,74 detik. Lalu untuk memori yang digunakan, Levensthein Distance lebih ringan dengan kapasitas 12.396,41kb dibandingkan dengan Jaro Winkler 12.664,84kb.*

Kata Kunci*: Hadis, Autocorrect, Jaro Winkler, Levensthein Distance.*

Abstract *– Hadith is the second Islamic law after the Koran. The number of hadiths that scattered in various books of hadith, encouraging Muslims to find ways to get the right hadith in the shortest possible time. One of them is the Book of Bulughul Maram which amounts to approximately 1,371 hadiths. So implementation is required for Hadith Search System. However, when the process search, the selected keyword sometimes error occurs. By hence, autocorrect is required to overcome tis. Then, required analysis to determine the performance of autocorrect on the Jaro Winkler Algorithm and Levensthein Distance. The results obtained after 20 times of testing, Jaro Winkler's accuracy 97.45%, higher than Levensthein's accuracy The distance is 90.27%. As for travel time, Levensthein Distance shorter by 0.09 seconds than Jaro Winkler by 0.74 seconds. Then for the memory used, Levensthein Distance is lighter with capacity 12,396.41kb compared to Jaro Winkler's 12,664.84kb.*

Keywords*: Hadith, Autocorrect, Jaro Winkler, Levensthein Distance*

1. Pendahuluan

Hadis merupakan hukum islam kedua setelah Alqur'an. Untuk mempelajari hadis, diperlukan sumber dari ulama-ulama yang sanadnya sampai ke Rasulullah SAW. Salah satu Kitab Ulama hadis yang populer saat ini adalah Kitab Bulughul Maram karangan Ibnu Hajar Al-Asqalani. Kitab ini merupakan kitab hadis tematik yang memuat hadis-hadis yang dijadikan sumber pengambilan hukum fikih (istinbath) oleh para ahli fikih yang berjumlah lebih kurang 1.371 buah hadis dan menjadi rujukan utama khususnya bagi fikih dari Mazhab Syafi'i. Kitab ini termasuk kitab fikih yang menerima pengakuan global dan juga banyak diterjemahkan ke berbagai bahasa di seluruh dunia [1].

Beberapa penelitian yang telah dilakukan sebelumnya, diantaranya yang dilakukan oleh Arie Satia Dharma, dkk pada tahun 2018 tentang Analisis Algoritma Approximate String Matching pada fitur Autocorrect dalam pencarian data. Penelitian tersebut berfokus pada perbandingan algoritma Levensthein Distance dan Hamming Distance. Data yang dipakai berupa nama dan tipe mobil sekitar 370.000 data. Pengujian dilakukan pada 10 data uji yang melalui 3 tahap yakni pengujian running time dan memory, akurasi pengurangan karakter dan akurasi pergantian karakter. Dari pengujian dapat dihasilkan bahwa algoritma Hamming Distance lebih baik dalam melakukan pencarian 10,527 ms dan memakan memory 3.479.294,2 byte dibandingkan dengan algoritma Levenshtein 1.201,8 ms dan 6.553.652,2 byte. Algoritma Levenshtein Distance memiliki akurasi lebih tinggi 79,4% dibandingan dengan algoritma Hamming Distance 25,8%. Serta akurasi pergantian kata, algortima Levenshtein Distance 79,1% dan algoritma Hamming Distance 77,5% [2].

Selanjutnya penelitian yang dilakukan oleh Agung Prasetyo, dkk pada tahun 2018 tentang penggunaan algoritma Jaro Winkler untuk fitur Autocorrect dan Spelling Sugestion pada penulisan naskah bahasa Indonesia di BMS TV. Dalam hal ini, Jaro Winkler digunakan untuk menangani kesalahan penulisan ejaan kata pada naskah bahasa Indonesia. Penggunaan algoritma tersebut melalui dua tahap yakni tahap Preprocessing dan tahap pengukuran jarak kedekatan kata. Dari pengujian yang dilakukan pada 60 kata yang yang salah pengejaannya, dihasilkan 49 yang mampu ditangani dengan tepat. Dengan demikian, algoritma tersebut mampu mengurangi kesalahan dalam penulisan naskah bahasa Indonesia [3].

Berdasarkan penelitian sebelumnya, pada umumnya perbandingan algoritma Levensthein Distance dan Hamming Distance, akurasi algoritma Levensthein Distance lebih tinggi dibandingkan dengan Hamming Distance. Sedangkan, waktu tempuh dilakukannya pencarian, algoritma Hamming Distance lebih cepat dibandingkan dengan Levensthein Distance. Lalu, kapasitas memori yang digunakan, algoritma Levensthein Distance lebih banyak memakan memori dibandingakan dengan algoritma Hamming Distance. Begitupun algoritma Jaro Winkler pada penelitian sebelumnya, umumnya dalam akurasi pengoreksian kata lebih tinggi dibandingkan dengan algoritma lainnya. Maka, penelitian ini diusulkan "Perbandingan kinerja Jaro Winkler Distance dan Levensthein Distance pada autocorrect Sistem Pencarian Hadis."

2. Metode Penelitian

Metode pembangunan perangkat lunak yang digunakan yaitu model prototipe. Dimana metode ini menerapkan secara langsung hasil analisa perbagian ke dalam sebuah model tanpa harus menunggu sistem selesai dibuat [10]. Metode pengembangan ini cocok digunakan untuk pengerjaaan proyek jangka pendek. Karena memiliki kelebihan diantaranya, pendefinisian kebutuhan pemakai menjadi lebih baik, peningkatan kepuasan pemakai, mempersingkat waktu pengembangan, pendeteksian kesalahan di setiap versi *Prototipe*, interaksi pengembang dan pelanggan dalam setiap perubahan, dan menekan biaya produksi [5]. Ada beberapa tahapan yang dilakukan pada pembangunan perangkat lunak ini :

1. Pengumpulan Kebutuhan

   Melakukan komunikasi dengan narasumber perihal pencarian hadis, sehingga teridentifikasi dan terkumpul data-data penting yang sesuai dengan kebutuhan sistem yang akan dibangun.
2. *Quick Design*

   Quick Desain bisa disebut juga dengan kegiatan dalam melakukan perancangan sekaligus dengan pemodelan. Pada tahap ini yakni, menerjemahkan dari analisis metode pencarian

ISBN: 978-602-60581-7-1

hadis dan kebutuhan sistem ke bentuk yang mudah dimengerti oleh pengguna seperti halnya diagram dan *interface.*

3. Implementasi

   Setelah dilakukan *Quick Design,* maka tahap selanjutnya ialah mengimplementasikan ke dalam kode program menjadi suatu perangkat lunak yang utuh.

4. *Testing*

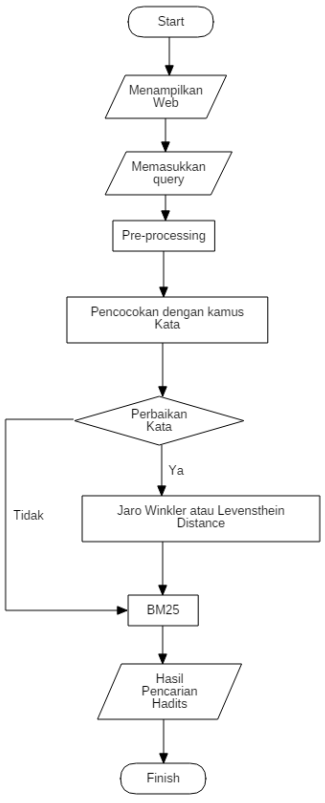
   Setelah implementasi sudah menjadi suatu perangkat lunak yang utuh, maka perlu dilakukan uji coba terlebih dahulu sebelum dipakai oleh pengguna.

5. Evaluasi

   Bila perangkat lunak sudah diuji coba, maka perlu adanya evaluasi dari serangkaian tahap yang sudah dikerjakan untuk dapat diperbaiki jika masih ada kekurangan ataupun kesalahan.

3.1. Alur Penelitian

Alur dari penelitian yang akan dilakukan bisa dilihat pada Gambar 1 berikut.



Gambar 1 *Flowchart* Aplikasi

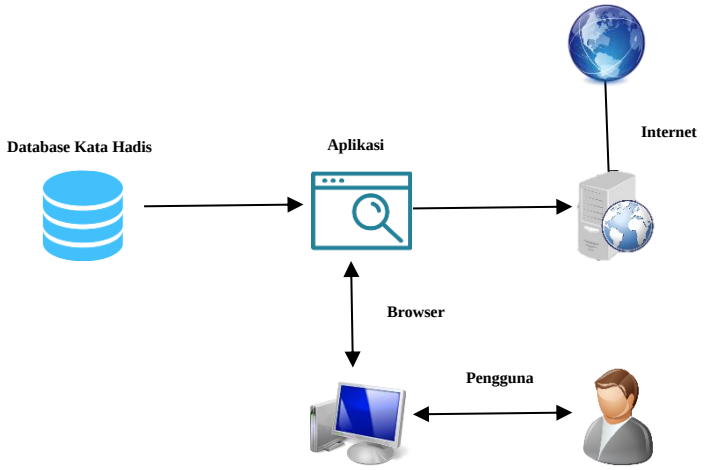
Pada Gambar 1, sebelum melakukan pencarian, *query* masuk tahap *preprocessing*. Sehingga dari banyaknya kata pada *query,* hanya kata kunci yang dapat dijadikan perhitungan untuk pencarian nantinya. Jika, kata kunci tidak ada kekeliruan pada penulisan, maka dilanjutkan pada proses pencarian. Namun, jika kata kunci terdapat kekeliruan pada penulisan, maka dilakukan *autocorrect* menggunakan *Jaro Winkler* atau *Levensthein Distance.* Setelah direkomendasikan

ISBN: 978-602-60581-7-1

kata kunci baru, maka lanjut pada proses pencarian.

3.2. Arsitektur system

Pada Gambar 2 Berikut merupakan arsitektur sistem pada penelitian ini, yang menggambarkan sistem yang dibuat serta spesifikasi dari sistem yang telah dibuat.



Gambar 2 Arsitektur sistem

3. Hasil dan Pembahasan

3.1. Implementasi Algoritma Jaro Winkler dan Levensthein Distance

Pada penelitian ini, data yang akan diolah untuk pencarian merupakan data hadis dari kitab Bulughul Maram yang diiputkan pada koleksi data berupa tabel hadis. Data tersebut seperti pada Tabel 1 berikut:

Tabel 1 Data Hadis

No.	Judul Hadis	Isi Hadis
1.	Kesucian Air	Dari Abu Sa'id al-Khudri rodhiyallohu 'anhu, ia berkata: Rosulullоh Shollallohu 'alaihi wa Sallam bersabda, "Sesungguhnya air itu mensucikan tidak dapat dinajiskan oleh sesuatu pun." Dikeluarkan oleh imam yang tiga (Abu Dawud, at-Tirmidzi, dan an-Nasa-i) dan dishohihkan olehAhmad.
2.	Laki-laki mandi dengan air bekas wanita dan sebaliknya.	Dari seorang laki-laki Sahabat Nabi Shollallohu ‘alaihi wa Sallam, ia berkata, “Rosulullah Shollallohu ‘alaihi wa Sallam melarang wanita untuk mandi dengan air bekas laki-laki dan laki-laki dengan air bekas wanita.” Musaddad menambah, “Hendaklah keduanya meciduk.” Dikeluarkan oleh Abu Dawud dan an-Nasa-i dan sanadnya shohih.

Dari data tersebut akan diolah dan dikenali oleh algoritma BM25. Hasil pencarian akan dilakukan pemeringkatan dari skor yang paling tinggi. Sebagai contoh dapat dilakukan seperti pada Tabel 2.
Query : Hadis tentang waktu Salat.

ISBN: 978-602-60581-7-1

Tabel 2 Dokumen Hadis

Dokumen ke-1	Dari Abdullah bin Amr RA bahwa Rasulullah SAW bersabda : "Waktu zuhur itu jika matahari telah condong dan bayangan seseorang sama dengan tingginya selama waktu asar belum tiba, waktu asar itu ada selama matahari belum menguning, waktu salat magrib itu selama awan merah belum menghilang, waktu salat isya itu hingga tengah malam dan waktu salat subuh itu mulai terbitnya fajar selama matahari belum terbit." HR Muslim.[612]

Setelah *query* dan dokumen yang tersedia, maka dilanjutkan dengan tahap *Preprocessing* yang melewati tahapan *Casefolding, Tokenizing, Filtering* dan *Stemming.*

1. *Casefolding*

Query : hadis tentang waktu shalat.

Tabel 3 *Casefolding* dokumen hadis

Dokumen ke-1	dari abdullah bin amr ra bahwa rasulullah saw bersabda waktu zuhur itu jika matahari telah condong dan bayangan seseorang sama dengan tingginya selama waktu asar belum tiba waktu asar itu ada selama matahari belum menguning waktu salat magrib itu selama awan merah belum menghilang waktu salat isya itu hingga tengah malam dan waktu salat subuh itu mulai terbitnya fajar selama matahari belum terbit hr muslim

2. *Tokenizing*

a. *Query*

Tabel 4 *Tokenizing Query*

Kata	Kata
Hadis	Waktu
Tentang	Salat

b. Dokumen Ke-1

Tabel 5 *Tokenizing* Dokumen ke-1

Kata	Kata	Kata	Kata	Kata
dari	matahari	tiba	Awan	salat
abdullah	Telah	waktu	Merah	subuh
bin	condong	asar	Belum	itu
amr	Dan	itu	menghilang	mulai
ra	bayangan	ada	Waktu	terbitnya
bahwa	seseorang	selama	Salat	fajar
rasullah	Sama	matahari	Isya	selama
saw	Dengan	belum	Itu	matahari
bersabda	tingginya	menguning	hingga	belum

ISBN: 978-602-60581-7-1

Kata	Kata	Kata	Kata	Kata
waktu	Selama	waktu	tengah	terbit
zuhur	Waktu	magrib	Malam	hr
itu	Asar	itu	Dan	muslim
jika	belum	selama	waktu	

3. Filtering

a. *Query*

Tabel 6 *Filtering Query*

Kata
Hadis
Salat

b. Dokumen Ke-1

Tabel 7 *Filtering* Dokumen ke-1

Kata	Kata	Kata	Kata	Kata
abdullah	zuhur	matahari	Isya	terbit
bin	matahari	menguning	malam	hr
amr	condong	magrib	salat	muslim
Ra	bayangan	awan	subuh	
rasullah	tingginya	merah	terbitnya	
saw	asar	menghilang	fajar	
bersabda	asar	salat	matahari	

4. Stemming

a. Query

Tabel 8 *Stemming Query*

Kata
Hadis
Salat

b. Dokumen Ke-1

Tabel 9 *Stemming* Dokumen Ke-1

Kata	Kata	Kata	Kata	Kata
abdullah	zuhur	matahari	Isya	terbit
bin	matahari	kuning	malam	hr
amr	condong	magrib	salat	muslim
Ra	bayang	Awan	subuh	
rasullah	tinggi	Merah	terbit	
saw	asar	Hilang	fajar	

ISBN: 978-602-60581-7-1

Dari hasil *preprocessing* di atas, dilanjutkan dengan perhitungan *autocorrect* menggunakan algoritma *Jaro Winkler* dan *Levensthein Distance.* Untuk *autocorrect* menggunakan *Jaro Winkler*, contoh koreksi kata "hadsi" menuju "hadis" langkahnya sebagai berikut:

$$dj = \frac{1}{3}\left(\frac{m}{S1} + \frac{m}{S2} + \frac{m-t}{m}\right) = \frac{1}{3} \times \left(\frac{5}{5} + \frac{5}{5} + \frac{5-1}{5}\right) = \frac{1}{3} \times \left(1 + 1 + \frac{4}{5}\right) = \frac{1}{3} \times (1 + 1 + 0{,}8) = \frac{1}{3}(2{,}8)$$
$$= 0{,}93$$
$$dw = dj + (l\ x\ p(1 - dj) = 0{,}93 + \big(3 \times 0{,}1(1 - 0{,}93)\big) = 0{,}93 + (3 \times 0{,}1(0{,}07)$$
$$= 0{,}93 + (3 \times 0{,}007) = 0{,}93 + 0{,}007 = 0{,}937$$

Begitupun jika menggunakan *autocorrect Levenshtein Distance*. Terlebih dahulu tentukan banyaknya karakter yang akan dikoreksi dengan matriks string sebagai berikut:

Tabel 10 Matriks string *Levensthein Distance*

		H	**A**	**D**	**I**	**S**
	0	1	2	3	4	5
H	1	0	1	2	3	4
A	2	1	0	1	2	3
D	3	2	1	0	1	2
S	4	3	2	1	1	2
I	5	4	3	2	2	2

Dari matriks string tersebut, angka yang diberi warna merah adalah banyaknya karakter yang akan dirubah. Untuk mengetahui akurasi merubah kata yang asalnya "hadsi" menuju "hadis" sebagai berikut :

$$Bobot\ kemiripan = \left(1 - \frac{d[m,n]}{Max\ (S,T)}\right) * 100\% = \left(1 - \frac{2}{5}\right) * 100\% = (1 - 0{,}4) * 100\%$$
$$= 0{,}6 * 100\% = 60\%$$

Dari kedua algoritma yang telah dihitung, untuk kompleksitas dari masing-masing algoritma sebagai berikut:

1. Kompleksitas Algoritma *Jaro Winkler*

Tabel 11 Kompleksitas Algoritma *Jaro Winkler*

Code	***Cost***	***Time***	***Cost.Time***
`Int (matchRange;cs1;cs2;)`	C1	n	C1n
`matchRange=max(cs1.length(),cs2.length()) / 2 - 1`	C2	n^2	C2n
`return matchRange;`	C3	n^2	$C3n^2$
`transposition(jaroCompare;jaroDistance);`	C4	n^2	$C4n^2$
`jaroCompare(cs1,cs2)=(matches(cs1,cs2)/cs1.length()+ matches(cs1,cs2) / cs2.length()+(matches(cs1,cs2)-transposes(cs1,cs2))/matches(cs1,cs2) ) / 3`	C5	n^2	$C5n^2$
`jaroDistance(cs1,cs2)=1-jaroCompare(cs1,cs2)`	C6	n^2	$C6n^2$

ISBN: 978-602-60581-7-1

Code	*Cost*	*Time*	*Cost.Time*
`return FALSE;`	C7	n	C7n
`Winkler(jaroWinklercompare;boostThreshold;prefixSize;jaroMeasure;)`	C8	n^2	$C8n^2$
`jaroWinklerCompare(cs1,cs2,boostThreshold,prefixSize)=jaroMeasure(cs1,cs2)<=boostThreshold?jaroMeasure(cs`	C9	n^2	$C9n^2$
`1,cs2):jaroMeasure(cs1,cs2)+0.1*prefixMatch(cs1,cs2,prefixSize)*(1.0-jaroDistance(cs1,cs2))`			
`jaroWinklerDistance(cs1,cs2,boostThreshold,prefixSize)=1-jaroWinklerCompare(cs1,cs2,boostThreshold,prefixSize)`	C10	n^2	$C10n^2$
`Return jaroWInklerCompare;`	C11	n	C11n

$$T(n) = \sum_i^n = 1$$
$$= (C1 + C7 + C11)n + (C2 + C3 + C4 + C5 + C6 + C8 + C9 + C10)n^2$$
$$= \theta(n)^2$$

Jumlah $= n = 3n$

$= n^2 = 8n^2$

Dari tabel diatas, hasil dari perhitungan kompleksitas algoritma *Jaro Winkler* menunjukkan bahwa jumlah (n) terdapat 3 sedangkan untuk nilai $(n)^2$ terdapat 8, artinya nilai n lebih kecil daripada nilai n^2 dengan selisih 5 berarti untuk proses pengoreksian kata kunci menggunakan algoritma *Jaro Winkler* adalah lambat.

2. Kompleksitas Algoritma *Levensthein Distance*

Tabel 12 Kompleksitas Algoritma *Levensthein Distance*

Code	*Cost*	*Time*	*Cost.Time*
`kolom declare int d[0..m, 0..n]`	C1	n	C1n
`for i from 0 to m`	C2	n^2	$C2n^2$
`d[i, 0] := i`	C3	n	C3n
`for j from 0 to n`	C4	n^2	$C4n^2$
`d[0, j] := j`	C5	n	C5n
`for i from 1 to m`	C6	n^2	$C6n^2$
`for j from 1 to n {`	C7	n^2	$C7n^2$
`if s[i] = t[j] then cost := 0`	C8	n^2	$C8n^2$
`else cost := 1`	C9	n	C9n
`d[i, j] := minimum( d[i-1, j] + 1)`	C10	n^2	$C10n^2$
`d[i, j-1] + 1`	C11	n	C11n
`d[i-1, j-1] + cost`	C12	n	C12n
`} return d[m, n]`	C13	n	C13n

ISBN: 978-602-60581-7-1

$$T(n) = \sum_i^n = 1$$
$$= (C1 + C3 + C5 + C9 + C11 + C12 + C13)n + (C2 + C4 + C6 + C7 + C8 + C10)n^2$$
$$= \theta(n)^2$$

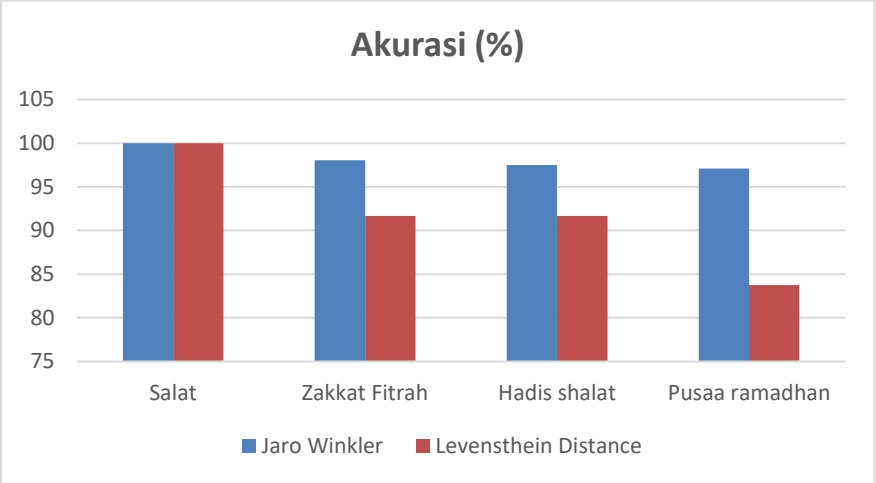
Jumlah $= n = 7n$

$= n^2 = 6n^2$

Dari tabel diatas, hasil dari perhitungan kompleksitas algoritma *Levensthein Distance* menunjukkan bahwa jumlah (n) terdapat 7 sedangkan untuk nilai $(n)^2$ terdapat 6, artinya nilai n lebih besar daripada nilai n^2 dengan selisih 1 berarti untuk proses pengoreksian kata kunci menggunakan algoritma *Levensthein Dostance* adalah cepat.

3.2. Pengujian

Berdasarkan hasil 20 kali pengujian dari algoritma *Jaro Winkler* dan *Leventhein Distance* terhadap data *training* sejumlah 28530 kata. Tiap kata memiliki skornya masing-masing. Untuk pengujian sendiri menggunakan 4 skenario yakni kata benar semua, kata pertama salah dan kata kedua benar, kata pertama benar dan kata kedua salah, dan kata salah semua. Agar lebih terarah pada perbandingannya. Untuk perbandingan algoritma dari sisi akurasi dapat dilihat pada Gambar 3.

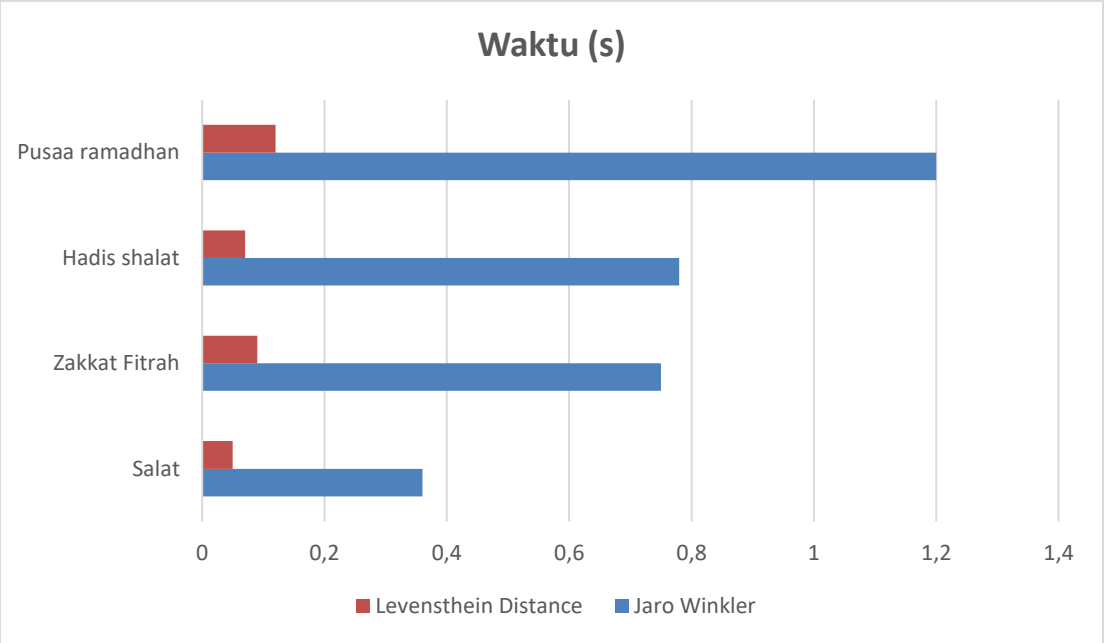


Gambar 3 Perbandingan Akurasi Algoritma

Pada Gambar 3, perwakilan kata untuk tiap skenario yang dihasilkan adalah kata awal sebelum dilakukan *autocorrect*. Setelah dilakukan *autocorrect*, menghasilkan kata baru dan kata baru tersebut adalah skor yang tertinggi. Akurasi yang dihasilkan belum tentu menunjukkan kata yang diharapkan. Setelah dibandingkan kedua algoritma tersebut, akurasi dari *Jaro Winkler* lebih baik dari *Levensthein Distance.*

Untuk waktu tempuh yang digunakan, perbandingan dari Algoritma *Jaro Winkler* dan *Levensthein Distance* dapat dilihat pada Gambar 4.

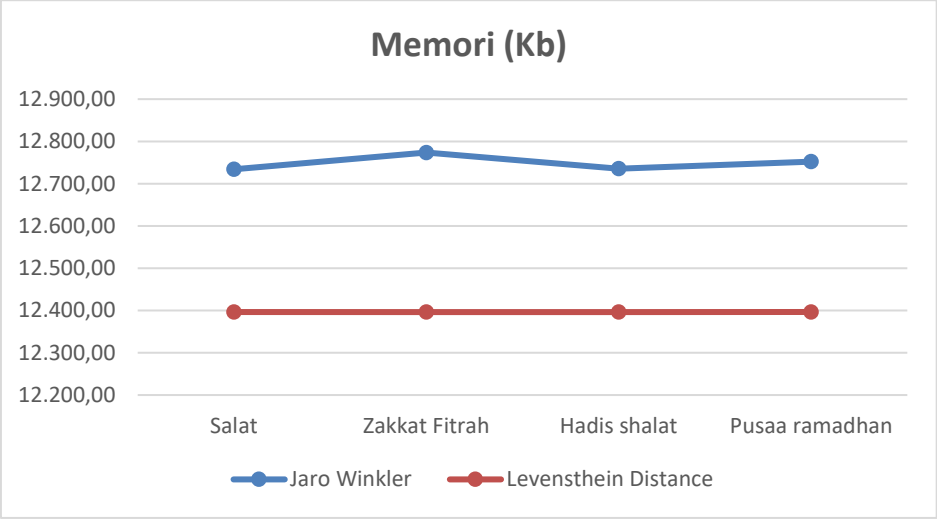
ISBN: 978-602-60581-7-1



Gambar 4 Perbandingan Waktu *Jaro Winkler* dan *Le*venshtein *Distance*

Pada Gambar 4, perbedaan waktu yang muncul lumayan jauh. Perhitungan waktu didapat dari selisih awal dan akhir dari pemrosesan algoritma. Waktu yang digunakan bisa saja sewaktu-waktu berubah menyesuaikan dengan kondisi perangkat keras yang digunakan. Namun saat ini, Algoritma *Levensthein Distance* lebih unggul dibandingkan dengan algoritma *Jaro Winkler*.

Untuk kapasitas yang digunakan, perbandingan algoritma *Jaro Winkler* dan *Levensthein Distance* dapat dilihat pada Gambar 5.



Gambar 5 Perbandingan Memori *Jaro Winkler* dan *Levensthein Distance*

Pada Gambar 5, memori yang digunakan adalah selisih dari akhir dan awal pemrosesan algoritma. Hasil dari perhitungan menunjukkan bahwa Algoritma *Levensthein Distance* lebih ringan dibandingkan dengan Algoritma *Jaro Winkler*.

Dari pengujian akurasi, waktu tempuh dan memori didapatkan hasil akhir pada Tabel 4.6 Sebagai berikut:

ISBN: 978-602-60581-7-1

Tabel 13 Perbandingan Algoritma *Jaro Winkler* dan *Levensthein Distance*

No.	Algoritma	Akurasi(%)	Waktu(s)	Memori(kb)
1.	*Jaro Winkler*	97,45	0,74	12.664,84
2.	*Levensthein Distance*	90,27	0,09	12.396,41

Dari hasil rata-rata nilai uji untuk koreksi kata masing-masing algoritma, selisih akurasi *Jaro Winkler* dengan *Levensthein Distance* sekitar 7,18%. Hal ini dikarenakan, algoritma *Jaro Winkler* dalam perhitungannya selain dari menghitung panjang dari dua *string,* dan menemukan jumlah karakter yang sama di dalam dua string, algoritma ini juga terdapat perhitungan untuk menemukan jumlah transposisi. Sehingga meningkatnya tingkat akurasi dari masing-masing kata yang tersedia di data *training.* Untuk waktu tempuh koreksi kata, memiliki selisih 0,65 detik. Karena, algoritma *Levensthein Distance* tidak terlalu banyak tahapan yang dilaluinya. Sehingga waktu tempuh yang dilalui pun lebih singkat. Lalu untuk kapasitas memori yang digunakan, memiliki selisih kapasitas 268,4kb. Karena, *performansi* dari algoritma *Levensthein Distance* lebih unggul dengan data *training* yang tersedia. *Untuk* penggunaan algoritmanya pada sistem pencarian hadis disesuaikan dengan kondisi dari sarana pendukung yang dimiliki.

4. Kesimpulan

Setelah dilakukannya implementasi dan pengujian algoritma, maka kesimpulannya sebagai berikut:

1. Algoritma *Jaro Winkler* yang telah diterapkan pada koreksi kata kunci telah melalui tahapan perhitungan panjang dari dua *string,* ditemukannya jumlah karakter yang sama di dalam dua *string,* dan ditemukannya jumlah transposisi. Lalu dilakukan perhitungan *Jaro Winkler* dengan dihasilkannya nilai maksimum dari skor masing-masing kata dasar. Sedangkan algoritma *Levensthein Distance,* telah melalui tahapan perhitungan panjang dua string dan matriks string. Lalu dilakukan perhitungan *Levensthein Distance* dengan dihasilkannya nilai minimum dari skor masing-masing kata dasar.
2. Berdasarkan hasil pengujian dengan menggunakan 20 data sampel dengan data *training* 28530 kata dalam membandingkan kinerja Algoritma *Jaro Winkler* dan *Levensthein Distance* dapat disimpulkan bahwa algoritma *Jaro Winkler* mendapatkan nilai akurasi lebih tinggi sebesar 97,45%. Sedangkan untuk waktu tempuh koreksi kata, algoritma *Levenesthein Distance* mendapatkan waktu lebih cepat sebesar 0,09 detik. Lalu untuk memori yang digunakan, algoritma *Levensthein Distance* lebih ringan kapasitas sebesar 12.396,41 kb.

Referensi

[1] "Bulughul Maram," 2018. [Online]. Available: https://id.wikipedia.org/wiki/Bulughul_Maram. [Accessed: 30-Jun-2019].

[2] A. S. Dharma, J. Banjarnahor, O. Nainggolan, and Y. Sihombing, "Analisis Algoritma Approximate String Matching Pada Fitur Autocorrect dalam Pencarian Data," vol. 01, no. 01, pp. 1–6, 2018.

[3] A. Prasetyo, W. M. Baihaqi, and I. S. Had, "ALGORITMA JAROWINKLER DISTANCE: FITUR AUTOCORRECT DAN SPELLING SUGGESTION PADA PENULISAN NASKAH BAHASA INDONESIA DI BMS TV," J. Teknol. Inf. dan Ilmu Komput., vol. 5, no. 4, pp. 435–444, 2018.

[4] J. Sarwono, Metode Penelitian Kuantitatif dan Kualitatif, Cetakan Pe. Yoogyakarta: Graha Ilmu, 2006.

[5] A. Kadir, Pengenalan Sistem Informasi Edisi Revisi. Yogyakarta: CV ANDI, 2014.

ISBN: 978-602-60581-7-1

[6] M. Hakim, "SISTEM PAKAR MENGIDENTIFIKASI JENIS HADITS MENGGUNAKAN METODE FORWARD CHAINING," pp. 28–29, 2016.
[7] P. D. I. W, "IMPLEMENTASI PEMBELAJARAN KITAB KUNING SEBAGAI UP PENINGKATAN RELIGIUSITAS PESERTA DIDIK DI PONDOK PESANTREN TARBIYATUL MUBTADIIN BEKASI TIMUR," Yogyakarta, 2018.
[8] N. K. bin Kurdian, "STUDI KOMPARASI ANTARA BAB AL-MIYAH DI KITAB ALMUHARRAR FI AHADITS AL-AHKAM DENGAN BAB ALMIYAH DI KITAB BULUGHUL MARAM MIN ADILLAH AL-AH KAM," Al-Majaalis, vol. 3, no. 2, pp. 1–33, 2016.
[9] W. Zaelani, "Penerapan Algoritma Jaro Winkler Distance Untuk Penilaian Esai Otomatis Pada Mata Pelajaran Kewarganegaraan," Bandung, 2017.
[10] Tinaliah and T. Elizabeth, "Perbandingan Hasil Deteksi Plagiarisme Dokumen Dengan Metode Jaro Winkler Distance dan Metode Latent Semantic Analysis," J. Teknol. dan Sist. Komput., vol. Vol 6, 2018.
[11] I. Taufik, I. D. Aishia, I. I. Data, and D. A. N. Metode, "IMPLEMENTASI FUZZY SEARCH UNTUK PENDETEKSI KATA ASING PADA," no. April, pp. 1–8, 2017.
[12] A. Nurul and S. Ulfah, "Implementasi Algoritma Levensthein Distance untuk koreksi kesalahan ejaan pada dokumen teks," Diploma Thesis UIN SGD Bandung, 2016.
[13] Hartanto, "TEXT MINING DAN SENTIMEN ANALISIS TWITTER PADA GERAKAN LGBT," vol. 9, no. 1, pp. 18–25, 2017.
[14] R. Siringoringo, "Text Mining dan Klasterisasi Sentimen Pada Ulasan Produk Toko Online," Tek. Inform. Univ. Prima Indones. Medan, vol. 2, pp. 1–6, 2019.
[15] A. Sofyan and S. Santosa, "TEXT MINING UNTUK KLASIFIKASI PENGADUAN PADA SISTEM LAPOR MENGGUNAKAN METODE C4.5 BERBASIS FORWARD SELECTION," Teknol. Inf., vol. 12, no. April, pp. 74–83, 2016.
[16] R. R. Bintana and S. Agustian, "Penerapan Model OKAPI BM25 Pada Sistem Temu Kembali Informasi," pp. 273–279, 2012.
[17] A. Dan et al., "KEBUDAYAAN."
[18] F. Sanjaya, "Pemanfaatan Sistem Temu Kembali Informasi dalam Pencarian Dokumen Menggunakan Metode Vector Space Model," J. Inf. Technol., vol. 5, no. 2, pp. 147–153, 2017.
[19] A. S. Nayak, A. P. Kanive, N. Chandavekar, and R. Balasubramani, "Survey on Pre-Processing Techniques for Text Mining," vol. 5, no. 16875, pp. 16875–16879, 2016.
[20] D. Susandi and U. Sholahudin, "Pemanfaatan Vector Space Model pada Penerapan Algoritma Nazief Adriani , KNN dan Fungsi Similarity Cosine untuk Pembobotan IDF dan WIDF pada Prototipe Sistem Klasifikasi Teks Bahasa Indonesia," vol. 3, no. 1, pp. 22–29, 2016.
[21] M. W. Sardjono, M. Cahyanti, M. Mujahidin, and R. Arianty, "BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA STEMMING NAZIEF-ADRIANI," jurnal.wicida.ac.id, pp. 138–146, 2018.
[22] D. Wahyudi, T. Susyanto, and D. Nugroho, "Implementasi dan analisis algoritma stemming nazief & adriani dan porter pada dokumen berbahasa indonesia," pp. 49–56, 2013.
[23] K. Frinta and P. P. Adikara, "Pencarian Berita Berbahasa Indonesia Menggunakan Metode BM25," vol. 3, no. 3, pp. 2589–2595, 2019.
[24] J. R. Embongbulan, Y. F. A, and A. P. Kurniati, "RETRIEVAL Latar belakang Kebutuhan informasi yang semakin tinggi , memaksa manusia untuk," 2010.
[25] R. A. S. and M. Shalahuddin, Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. 2016.
[26] R. A.S and M. Shalahuddin, Pemrograman Berorientasi Obyek. Bandung: Modula, 2010.
[27] P. Hidayatullah and J. K. Kawstara, Pemrograman Web, Edisi Revi. Bandung: Informatika Bandung, 2017.
[28] B. Raharjo, Belajar Otodidak Codeigniter, Pertama. Bandung: Informatika Bandung, 2015.
[29] R. Abdullah, Web Programming is Easy. Jakarta: PT. Elex Media Komputindo, 2015.
[30] R. S. Pressman, Rekayasa Perangkat Lunak Pendekatan Praktisi. Yogyakarta: Andi, 2002.

ISBN: 978-602-60581-7-1