Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.1148080
Vol. 11, No. 2, Agustus 2024, hlm. 715-722 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

KOMPARASI KINERJA ALGORITMA *BLOCKING* PADA PROSES *INDEXING* UNTUK DETEKSI DUPLIKASI

M. Miftakul Amin[*1], Yevi Dwitayanti[2]

[1,2]Politeknik Negeri Sriwijaya, Palembang
Email: [1]miftakul_a@polsri.ac.id, [2]yevi_dwitayanti@polsri.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 23 November 2023, diterima untuk diterbitkan: 8 Agustus 2024)

Abstrak

Proses integrasi data dari *heterogeneous data sources* memerlukan kualitas data yang baik. Salah satu ciri kualitas data yang baik adalah terhindar dari terjadinya duplikasi data. Untuk melakukan deteksi duplikasi, langkah yang dapat dilakukan adalah membandingkan setiap *record* dalam sebuah dataset sehingga membentuk *candidate record pair*. Teknik *blocking* digunakan untuk proses *indexing* yang dapat mengurangi jumlah pasangan *record* dalam proses deteksi duplikasi. Penelitian ini bertujuan untuk melakukan perbandingan beberapa algoritma *blocking* sehingga diperoleh rekomendasi algoritma mana yang paling optimal digunakan. Penelitian ini melakukan investigasi terhadap 6 buah algoritma *blocking*, yaitu *Soundex*, *NYSIIS*, *Metaphone*, *Double Metaphone*, *Jaro Winkler Similarity*, dan *Cosine Similarity*. Dataset yang digunakan dalam penelitian ini adalah dataset *restaurant* yang berisi 112 *record*, yang di dalamnya terdapat beberapa *record* yang terindikasi duplikat. Hasil penelitian menunjukkan bahwa algoritma *NYSIIS* memberikan hasil *record blocking* paling optimal, yaitu sebesar 97 *record*. Sedangkan algoritma *Soundex* dan *Cosine Similarity* memberikan hasil yang paling optimal, yaitu sebesar 8 buah *candidate record pair*. Sedangkan dari sisi waktu eksekusi algoritma *Soundex* dan *NYSIIS* memberikan proses yang paling cepat dengan durasi 0,04 detik.

Kata kunci: algoritma *blocking, candidate record pair, deteksi duplikasi, dataset restaurant*

PERFORMANCE COMPARISON OF BLOCKING ALGORITHMS IN THE INDEXING PROCESS FOR DUPLICATION DETECTION

Abstract

The process of integrating data from heterogeneous data sources requires good data quality. One of the characteristics of good data quality is avoiding data duplication. To perform duplication detection, a step that can be done is to compare each record in a dataset to form a candidate record pair. The blocking algorithm is used for the indexing process which can reduce the number of record pairs in the duplication detection process. This research aims to compare several blocking algorithms so as to obtain recommendations on which algorithm is most optimally used. This research investigates 6 blocking algorithms, namely Soundex, NYSIIS, Metaphone, Double Metaphone, Jaro Winkler Similarity, and Cosine Similarity. The dataset used in this research is a restaurant dataset containing 112 records, in which there are several records that indicate duplicates. The results showed that the NYSIIS algorithm provided the most optimal record blocking results, which amounted to 97 records. While the Soundex and Cosine Similarity algorithms provide the most optimal results, which are 8 candidate record pairs. In terms of execution time, the Soundex and NYSIIS algorithms provide the fastest process with a duration of 0.04 seconds.

Keywords: *blocking algorithm, candidate record pair, duplicate detection, restaurant dataset*

1. PENDAHULUAN

Besarnya jumlah data yang tersimpan dalam *storage* dan komunikasi data menggunakan jaringan internet membawa konsekuensi data tersimpan dan diperoleh dari beberapa sumber (*heterogeneous data source*). Hal ini membawa peluang terhadap proses integrasi data, dan konsekuensi terhadap tersedianya data yang valid dan berkualitas. Bahkan (Cahyono & Sucahyo, 2020) menyebut bahwa Kualitas data berbanding lurus dengan kualitas informasi. Salah satu tugas penting dalam menjaga kualitas data adalah mencegah terjadinya duplikasi. Proses mengeliminasi data yang duplikat dalam basis data menurut (Priya et al., 2021) disebut dengan istilah *deduplication.*

Bahkan menurut (Christen, 2012a) proses untuk menemukan entitas yang sama dalam sebuah basis data dikenal dengan istilah *deduplication*, sedangkan jika diaplikasikan pada beberapa basis data dikenal dengan istilah *record linkage*.

Salah satu tantangan dalam deteksi duplikasi adalah volume data yang besar. Padahal *record* duplikat dari sisi jumlah tidak terlalu besar dalam sebuah basis data. Sehingga diperlukan sebuah mekanisme untuk mengurangi jumlah *record* yang akan dibandingkan dalam penentuan status duplikasi. Menurut (Amin & Triwahyuni, 2024) proses ini dikenal dengan *indexing*. *Indexing* merupakan proses yang dilakukan untuk mengurangi jumlah *candidate record pair* yang akan digunakan dalam proses *matching* untuk menentukan apakah pasangan *record* terindikasi duplikat atau tidak (Christen, 2012b). Salah satu teknik *indexing* menurut (O'HARE *et al.*, 2022) adalah teknik *blocking*. Dalam deteksi duplikasi pada *record* basis data, proses *blocking* digunakan untuk mengurangi jumlah perbandingan antara kandidat pasangan *record* (*candidate record pairs*).

Beberapa penelitian terkait dengan fungsi *similarity* telah dilakukan, diantaranya oleh (Amalia *et al.*, 2021), yang menggunakan fungsi *Cosine Similarity* untuk mengukur tingkat kemiripan ujian *online* pada soal esai. Hasil pengujian menggunakan nilai *precision*, *recall*, dan *f-measure* diperoleh nilai rata-rata 81,00%. Demikian juga penelitian yang dilakukan oleh (Sanjaya *et al.*, 2023), telah menguji kemiripan makna atara 2 buah kalimat menggunakan algoritma *Cosine Similarity*. Hasil penelitian ini menghasilkan tingkat kemipiran rata-rata sebesar 94,48%.

Penelitian yang dilakukan oleh (Amin *et al.*, 2022), telah mengembangkan sebuah *ground truth dataset* yang berisi informasi pasangan *record* yang memiliki status duplikat dalam proses deteksi duplikasi pada basis data riset. Penelitian ini menggunakan fungsi *Jaro Winkler Similarity* dan model *decision support system (DSS)* dalam sistem yang dikembangkan.

Teknik *string matching* dan *word embedding* juga telah digunakan dalam mengindentifikasi entitas nama (Maity *et al.*, 2021). Berdasarkan hasil pengukuran diperoleh nilai precision 96,57%, recall 96,57%, dan F1-measure sebesar 96,40%. Sedangkan algoritma *Metaphone* telah digunakan oleh (Valencio *et al.*, 2020) dalam penelitiannya tentang pengembangan linkungan data cleaning untuk proses integrase data.

Penelitian tentang algoritma *blocking* pada pembentukan *candidate record pair* dalam proses deteksi duplikasi belum banyak yang melakukan. Oleh karena itu penelitian ini penting untuk dilakukan, dalam rangka melakukan investigasi terhadap beberapa algoritma *blocking*, dan memberikan rekomendasi algoritma yang dapat dipilih dalam pembentukan dataset.

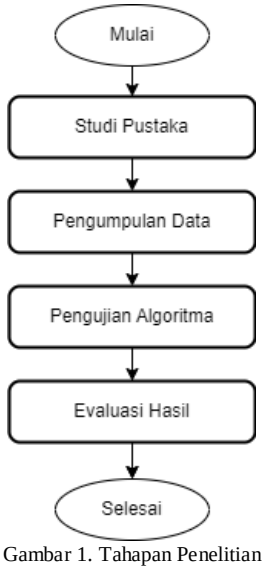
Lebih lanjut Penelitian yang dilakukan oleh (Amin *et al.*, 2024), telah mengembangkan model deteksi duplikasi menggunakan pendekatan *threshold-based* dan *rule-based* dalam mekanisme deteksi duplikasi. Pada penelitian ini menggunakan *dataset* berupa basis data riset yang berasal dari pangkalan data SINTA. Penelitian ini menghasilkan deteksi duplikasi yang cukup baik dengan menggunakan *confusion matrix* sebagai parameter evaluasi kinerja model.

Penelitian ini bertujuan melakukan perbandingan kinerja dari beberapa algoritma *string similarity*, sehingga akan diketahui algoritma yang memberikan kinerja yang baik dengan *dataset* tunggal. Penggunaan *dataset* tunggal ini akan memberikan fokus informasi kinerja dari masing-masing algoritma, karena menggunakan data yang sama dalam pemrosesannya.

2. METODE PENELITIAN

2.1. Tahapan Penelitian

Gambar 1 memperlihatkan tahapan yang dilakukan dalam penelitian ini, dimulai dengan studi pustaka dengan menelaah tinjauan pustaka dan penelitian terdahulu yang relevan. Kemudian dilanjutkan dengan pengumpulan data, yang dalam hal ini mencari dataset yang akan digunakan sebagai studi kasus dalam penelitian. kemudian dilanjutkan dengan pengujian algoritma untuk proses *blocking*. Setelah pengujian beberapa algoritma selesai, kemudian dilanjutkan dengan evaluasi hasil dari *blocking*.



Gambar 1. Tahapan Penelitian

2.2. Dataset

Pada penelitian ini *dataset* yang digunakan adalah dataset *restaurant*, yang berisi 112 *record*. Di dalam dataset tersebut terdapat beberapa *record* yang terindikasi duplikat. Dataset dapat diunduh secara daring pada laman https://dedupe.io/data/restaurant-1.csv. Dataset ini terdiri dari 5 buah field sebagai fitur dengan struktur data yang dapat dijelaskan seperti pada Tabel 1.

Tabel 1. Struktur Dataset Restaurant

Field	Keterangan
name	Nama restoran
address	Alamat
city	Kota
cuisine	Jenis masakan
unique_id	Record ID

2.3. Fungsi *Soundex*

Algoritma *Soundex* merupakan algoritma yang masuk dalam kategori *phonetic similarity metric* (James, 2020). Tahapan dalam algoritma *Soundex* dapat dilakukan sebagai berikut:

1. Pertahankan huruf pertama dari string sebagai huruf awalan dan abaikan semua kemunculan W dan H di posisi lain;
2. Tetapkan kode berikut ini untuk huruf-huruf yang tersisa:
   - B, F, P, V → 1
   - C, G, J, K, Q, S, X, Z → 2
   - D, T → 3
   - L → 4
   - M, N → 5
   - R → 6
3. A, E, I, O, U dan Y tidak diberi kode tetapi berfungsi sebagai pemisah;
4. Gabungkan urutan kode yang identik dengan hanya menyimpan kemunculan pertama dari kode tersebut;
5. Hilangkan pemisah;
6. Pertahankan awalan huruf dan tiga kode pertama, isi dengan angka nol jika kurang dari tiga kode.

2.4. Fungsi NYSIIS

New York State Identification and Intelligence System (NYSIIS) juga merupakan algoritma yang masuk dalam *phonetic similarity metric* (Yerai *et al*., 2024). Tahapan algoritma NYSIIS adalah sebagai berikut:

1. Ubah huruf awal string seperti yang ditunjukkan dalam Tabel 2.

Tabel 2. Pemetaan Perubahan Huruf Langkah 1

Huruf Asli	Perubahan Huruf
MAC	MCC
KN	NN
K	C
PH	FF
PF	FF
SCH	SSS

2. Ubah huruf terakhir dari string seperti yang ditunjukkan dalam Tabel 3.

Tabel 3. Pemetaan Perubahan Huruf Langkah 2

Huruf Asli	Perubahan Huruf
EE	Y
IE	Y
DT	D
RT	D
RD	D
NT	D
ND	D

3. Karakter pertama dari string berkode *NYSIIS* adalah huruf pertama dari string (yang mungkin telah diubah).
4. Posisikan "*pointer*" pada huruf kedua dari string (yang mungkin sudah diubah).
5. (Mengubah huruf saat ini dari string - yaitu huruf yang berada pada posisi "pointer" saat ini). Jalankan salah satu dari operasi berikut ini, mulai dari atas ke bawah:
   a) Jika kosong, lanjutkan ke Langkah 7.
   b) Jika huruf saat ini adalah "E" dan huruf berikutnya adalah "V," ubah "EV" ke "AF."
   c) Mengubah vokal ("AEIOU") menjadi "A."
   d) Ubah "Q" menjadi "G".
   e) Change “Z” to “S.”
   f) Ubah "M" menjadi "N".
   g) Jika huruf saat ini adalah huruf "K," maka ubah "K" menjadi "C" kecuali jika huruf berikutnya adalah "N." Jika "K" diikuti dengan "N", maka ganti "KN" dengan "N."
   h) Ubah "SCH" menjadi "SSS."
   i) Ubah "PH" menjadi "FF."
   j) Jika "H" didahului atau diikuti oleh huruf yang bukan vokal (AEIOU), maka ganti huruf saat ini dalam string dengan huruf sebelumnya.
   k) Jika "W" didahului oleh huruf hidup, ganti huruf saat ini dalam string dengan huruf sebelumnya.
6. Karakter berikutnya dari kode *NYSIIS* adalah huruf posisi saat ini dalam string setelah menyelesaikan Langkah 5 (tetapi menghilangkan huruf yang sama dengan karakter terakhir yang sudah ditempatkan dalam kode).
   Setelah memasukkan karakter ke dalam kode, pindahkan penunjuk ke huruf berikutnya dari string. Kemudian kembali ke Langkah 5.
7. (Ubah karakter terakhir dari string berkode NYSIIS). Jika dua karakter terakhir dari string berkode NYSIIS adalah "AY," maka ganti "AY" dengan "Y." Jika karakter terakhir dari string berkode NYSIIS adalah "S" atau "A", maka hapuslah.

2.5. Fungsi *Metaphone* dan *Double Metaphone*

Metaphone adalah algoritma fonetik, yang diciptakan oleh Lawrence Philips pada tahun 1990, untuk mengindeks kata-kata berdasarkan pengucapan bahasa Inggris (Raykar *et al.*, 2023). Pada dasarnya, algoritma ini memperbaiki algoritma *Soundex* dengan menggunakan informasi tentang variasi dan ketidakkonsistenan dalam pengejaan dan pengucapan bahasa Inggris untuk menghasilkan pengkodean yang lebih akurat, dengan melakukan pekerjaan yang lebih baik dalam mencocokkan kata dan nama yang terdengar mirip. Seperti halnya *Soundex*, kata-kata yang terdengar mirip harus memiliki kunci yang sama.

Kode *Metaphone* menggunakan 16 simbol konsonan 0BFHJKLMNPRSTWXY. '0' mewakili

"th" (sebagai perkiraan ASCII untuk Θ), 'X' mewakili "sh" atau "ch", dan yang lainnya mewakili pengucapan bahasa Inggris yang biasa. Vokal AEIOU juga digunakan, tetapi hanya pada awal kode. Tahapan yang dilakukan dalam algoritma ini adalah:

1. Hapus huruf kedua dari huruf yang digandakan, kecuali C.
2. Jika kata dimulai dengan 'KN', 'GN', 'PN', 'AE', 'WR', hilangkan huruf pertama.
3. Hilangkan huruf 'B' jika setelah huruf 'M' di akhir kata.
4. 'C' berubah menjadi 'X' jika diikuti oleh 'IA' atau 'H' (kecuali dalam kasus terakhir, ini adalah bagian dari '-SCH-', yang dalam hal ini berubah menjadi 'K'). 'C' berubah menjadi 'S' jika diikuti oleh 'I', 'E', atau 'Y'. Jika tidak, 'C' berubah menjadi 'K'.
5. 'D' berubah menjadi 'J' jika diikuti oleh 'GE', 'GY', atau 'GI'. Jika tidak, 'D' berubah menjadi 'T'.
6. Hapus 'G' jika diikuti oleh 'H' dan 'H' tidak berada di akhir atau sebelum huruf hidup. Lepaskan 'G' jika diikuti oleh 'N' atau 'NED' dan berada di akhir.
7. 'G' berubah menjadi 'J' jika sebelum 'I', 'E', atau 'Y', dan tidak ada dalam 'GG'. Jika tidak, 'G' berubah menjadi 'K'.
8. Hilangkan huruf 'H' jika setelah huruf hidup dan bukan sebelum huruf hidup.
9. 'CK' berubah menjadi 'K'.
10. 'PH' berubah menjadi 'F'.
11. 'Q' berubah menjadi 'K'.
12. 'S' berubah menjadi 'X' jika diikuti dengan 'H', 'IO', atau 'IA'.
13. 'T' berubah menjadi 'X' jika diikuti oleh 'IA' atau 'IO'. 'TH' berubah menjadi '0'. Hilangkan 'T' jika diikuti oleh 'CH'.
14. 'V' berubah menjadi 'F'.
15. 'WH' berubah menjadi 'W' jika di awal. Hilangkan 'W' jika tidak diikuti oleh vokal.
16. 'X' berubah menjadi 'S' jika di awal. Jika tidak, 'X' berubah menjadi 'KS'.
17. Hilangkan huruf 'Y' jika tidak diikuti dengan huruf vokal.
18. 'Z' berubah menjadi 'S'.
19. Hilangkan semua huruf vokal kecuali di awal kata.

Sedangkan algoritma *Double Metaphone* adalah generasi kedua dari algoritma *Metaphone*. Disebut "Double" karena dapat mengembalikan kode primer dan sekunder untuk sebuah string.

2.6. Fungsi *Jaro Winkler Similarity*

Salah satu fungsi *similarity* yang masuk dalam kategori *character-based similarity* adalah fungsi *jaro-winkler* yang merupakan varian dari *jaro distance metric*, yang berfungsi untuk mengukur tingkat similarity antara dua buah string (Amin *et al.*, 2022). Formula (1) dan (2) memperlihatkan fungsi *similarity jaro-winkler*.

$$sim_{jaro}(s1,s2) = \frac{1}{3}\left(\frac{c}{s1} + \frac{c}{s2} + \frac{c-t}{c}\right) \quad (1)$$

$\text{Sim}_{winkler}$ (s1,s2) =

$$\text{sim}_{jaro}\text{ (s1,s2)} + (1.0 - \text{sim}_{jaro}\text{ (s1,s2)})\ l_p \quad (2)$$

Variabel c merupakan jumlah karakter yang sama persis, s1 merupakan panjang string 1, s2 merupakan panjang string 2, t merupakan jumlah transposisi atau karakter yang sama pada string). Variabel l merupakan panjang prefiks umum di awal string dengan nilai maksimalnya 4 karakter, sedangkan p merupakan konstanta scalling factor dengan nilai default-nya adalah 0,1.

2.7. Fungsi *Cosine Similarity*

Menurut (Amalia *et al.*, 2021) menyebutkan bahwa nilai *cosinus* pada sebuah ruang dimensi antara dua buah *vector* A dan B dapat digunakan untuk mengukur tingkat kemiripannya seperti dapat dilihat pada formula (3).

$$Sim = \cos(\theta) = \frac{A \cdot B}{\|A\|\ \|B\|} \quad (3)$$

3. HASIL DAN PEMBAHASAN

Pada pengukuran kinerja algoritma *blocking* ini akan dilihat dari 3 buah parameter yang digunakan dalam pengujian masing-masing algoritma, yaitu dari sisi jumlah *block* yang terbentuk (*record blocking*), jumlah pasangan *record* yang terindikasi duplikasi (*candidate record pair*), dan yang ketiga adalah waktu eksekusi dalam satuan detik. Proses pengujian yang dilakukan ini menggunakan Bahasa pemrograman *Python* dan menggunakan Jupyter Notebook dalam implementasi kode programnya.

3.1. Hasil Pengujian

3.1.1. Pengujian *Record Blocking*

Tabel 4 memperlihatkan hasil *record blocking* yang menggambarkan berapa banyak *block* yang terbentuk untuk mengelompokkan *record* berdasarkan algoritma *blocking* yang digunakan.

Tabel 4. Hasil *Record Blocking*

No.	Fungsi Blocking	Record Blocking
1.	Soundex	107
2.	NYSIIS	94
3.	Metaphone	112
4.	Double Metaphone	112
5.	Jaro Winkler Similarity (Θ >= 0.80)	103
6.	Cosine Similarity (Θ >= 0.80)	108

3.1.2. Pengujian *Candidate Record Pair*

Pada Tabel 5 disajikan jumlah *candidate record pair* yang diperoleh dari setiap *record blocking*. Jika *record blocking* berisi jumlah *block* yang dihasilkan, maka pada *candidate record pair* adalah pasangan

record yang dijadikan dasar klasifikasi dalam penentuan duplikasi dengan memilih *record blocking* dengan jumlah *record* lebih dari 1 dalam setiap *block*.

Tabel 5. Hasil *Candidate Record Pair*

No.	Fungsi Blocking	Candidate Record Pair
1.	Soundex	8
2.	NYSIIS	25
3.	Metaphone	112
4.	Double Metaphone	112
5.	Jaro Winkler Similarity (Θ >= 0.80)	15

Beberapa record *dataset* yang terindikasi sebagai pasangan *record* yang duplikasi dapat dilihat pada Tabel 6. Data-data ini merupakan hasil dari proses *blocking*, dimana *record* akan dikelompokkan ke dalam *block* yang sama jika memiliki nilai kemiripan yang berdekatan.

Tabel 6. Hasil *Candidate Record Pair*

id	name	address	city	cuisine
89	ritzcarlton cafe buckhead	3434 peachtree rd ne	atlanta	american new
90	ritzcarlton dining room buckhead	3434 peachtree rd ne	atlanta	american new
41	le bernardin	155 w. 51st st.	new york city	seafood
42	les celebrites	155 w. 58th st.	new york city	french (classic)
58	second avenue deli	156 second ave	new york city	delis
71	second street grill	200 e fremont st	las vegas	pacific rim

3.1.3. Pengujian Waktu Eksekusi

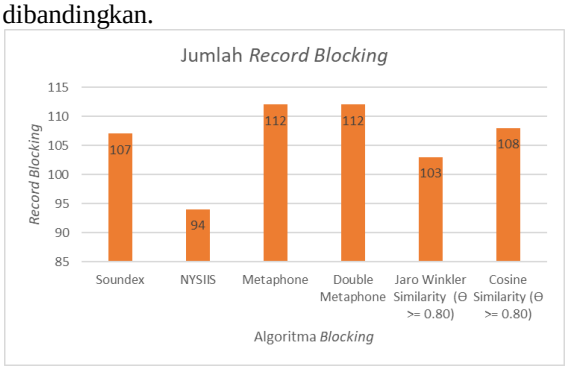
Melihat dari sisi waktu, maka Tabel 7 menyajikan waktu yang digunakan dalam proses *blocking* setiap algoritma. Satuan yang digunakan dalam waktu eksekusi adalah detik.

Tabel 7. Waktu Eksekusi Algoritma Blocking

No.	Fungsi Blocking	Waktu Eksekusi (detik)
1.	Soundex	0.0424
2.	NYSIIS	0.0401
3.	Metaphone	0.3291
4.	Double Metaphone	0.4495
5.	Jaro Winkler Similarity (Θ >= 0.80)	0.0777
6.	Cosine Similarity (Θ >= 0.80)	0.1519

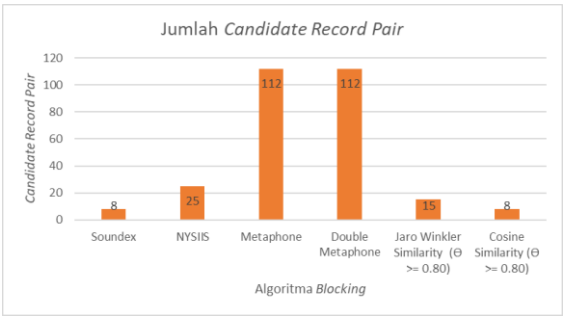
3.2. Pembahasan

Berdasarkan hasil *blocking* yang telah dilakukan, maka diperoleh *record blocking* seperti ditampilkan pada Gambar 2. Tampak bahwa algoritma *NYSIIS* memiliki hasil paling optimal sebesar 92 *block*. Selanjutnya algoritma *Jaro Winkler Similarity* sebesar 103 *block*, algoritma *Soundex* sebesar 107 *block*, dan algoritma *Cosine Similarity* sebesar 108 *block*. Sedangkan algoritma *Metaphone* dan *Double Metaphone* memperoleh hasil *record blocking* sejumlah 112, yang berarti tidak diperoleh hasil yang signifikan. Pemilihan nilai threshold sebesar 0.80 pada algoritma *Jaro Winkler* dan *Cosine Similarity* didasarkan pada nilai rata-rata dari proses pengukuran kemiripan dari setiap string yang dibandingkan.



Gambar 2. Hasil Jumlah *Record Blocking*

Hasil *blocking* dari aspek jumlah *candidate record pair* diperlihatkan pada Gambar 3, seperti ditampilkan bahwa algoritma *Soundex* dan *Cosine Similarity* menghasilkan jumlah yang paling optimal, yaitu sebesar 8 buah *candidate record pair*. Sedangkan algoritma *Metaphone* dan *Double Metaphone* memiliki jumlah *record* paling banyak, yaitu sebesar 112 *record*. Artinya kedua metode ini tidak menghasilkan *candidate record pair* sebagai tujuan utama dari proses *blocking*, yaitu mengurangi jumlah pasangan *record* untuk proses deteksi duplikasi.

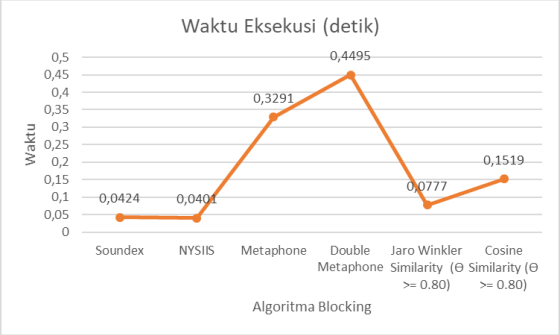


Gambar 3. Hasil Jumlah *Candidate Record Pair*

Gambar 4 merupakan tampilan distribusi waktu eksekusi dari beberapa algoritma *blocking* yang diimplementasikan. Dilihat dari waktu eksekusi, maka algoritma *Soundex* dan *NYSIIS* memiliki waktu eksekusi paling cepat dibandingkan dengan algoritma yang lain, yaitu sebesar 0,40 detik. Sedangkan algoritma *Metaphone* dan *Double Metaphone* memiliki waktu eksekusi paling lama, yaitu sebesar 0,3291 dan 0,4495.

Pada pengujian waktu eksekusi peneliti menggunakan spesifikasi *processor Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz, Installed RAM* sebesar 8.00 GB (7.88 GB *usable*), yang bekerja pada sistem operasi *64-bit operating system, x64-based processor*. Jika diimplementasikan pada spesifikasi perangkat yang berbeda, baik lebih tinggi maupun lebih rendah, maka waktu eksekusi dapat saja berbeda hasilnya.

Pengujian dari sisi waktu eksekusi didasarkan pada proses eksekusi untuk masing-masing algoritma *string similarity*. Sebuah variabel dengan nama *start_time* diberi nilai dengan waktu mulai eksekusi menggunakan Bahasa pemrograman Python dengan segmen kode program *start_time = time.time()*. Kemudian setelah eksekusi algoritma *string similarity* selesai dilakukan, sebuah variabel dengan nama *end_time* diberi nilai waktu akhir eksekusi seperti segmen kode program *end_time = time.time()*. Kemudian selisih antara *end_time* dan *start_time* merupakan waktu eksekusi dari masing-masing algoritma. Dengan demikian, maka waktu eksekusi akan menyesuaikan dan bersifat independen dengan lingkungan kerja dari sumber daya komputasi yang berbeda-beda baik dari sisi kecepatan processor, kapasitas RAM, dan parameter sumber daya komputasi lainnya.



Gambar 4 Hasil Waktu Eksekusi

4. KESIMPULAN

Berdasarkan hasil dan pembahasan yang telah dilakukan dalam penelitian ini dapat disimpulkan bahwa dari 112 *record* dataset *restaurant* sebagai *dataset* yang diuji, maka dari sisi jumlah *record blocking* algoritma *NYSIIS* merupakan algoritma yang paling optimal dengan menghasilkan nilai 94. Sedangkan dari aspek jumlah *candidate record pair* yang dihasilkan, maka algoritma *Soundex* dan *Cosine Similarity* cukup optimal dengan menghasilkan pasangan *record* sebanyak 8 buah. Dari sisi kecepatan eksekusi, maka algoritma *Soundex* dan *NYSIIS* merupakan algoritma tercepat dengan waktu eksekusi sebesar 0,40.

Sebagai penelitian lanjutan, dapat dilakukan pengujian pada beberapa *dataset* dengan beragam variasi tipe data, dan variasi nilai data, serta volume *dataset*. Tidak menutup kemungkinan melihat kinerja algoritma dari aspek yang lainnya, dimana penelitan ini masih menggunakan 3 buah parameter saja, yaitu jumlah *record blocking*, jumlah *candidate record pair*, dan waktu eksekusi.

DAFTAR PUSTAKA

AMALIA, E.L., JUMADI, A.J., MASHUDI, I.A., & WIBOWO, D.W. 2021. Analisis Metode Cosine Similarity Pada Aplikasi Ujian Online Otomatis (Studi Kasus JTI POLINEMA), *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(2), pp. 343–348. Tersedia di: https://doi.org/10.25126/jtiik.2021824356.

AMIN, M.M., STIAWAN, D., ERMATITA, E., SURBOTO, I.M.I, & LUKMAN, L. 2022. Decision Support System using Weighting Similarity Model for Constructing Ground-Truth Dataset, *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2022-Octob(October), pp. 56–60. Tersedia di: https://doi.org/10.23919/EECSI56542.2022.9946630.

AMIN, M.M. & TRIWAHYUNI, A. 2024. Deterministic Record Linkage untuk Pembentukan Master Data Dosen, Jurnal TEKNIKA, 18(1), pp. 323-330.

AMIN, M.M., STIAWAN, D., ERMATITA, BUDIARTO, R., 2024. Proposed Threshold-Based and Rule-Based Approaches to Detecting Duplicates in Bibliographic Database. Bulletin of Electrical Engineering and Informatics (BEEI), 13(3), p.2036-2047.

CAHYONO, S.H. & SUCAHYO, Y.G. 2020. Pengukuran Kualitas Data Menggunakan Framework Total Data Quality Management (TDQM): Studi Kasus Sistem Informasi Beasiswa Universitas Indonesia, Jurnal IPTEK-KOM (Jurnal Ilmu Pengetahuan dan Teknologi Komunikasi), 22(2), pp. 193–206.

CHRISTEN, P. 2012a. A survey of indexing techniques for scalable record linkage and deduplication, *IEEE Transactions on Knowledge and Data Engineering*, 24(9), pp. 1537–1555. Tersedia di: https://doi.org/10.1109/TKDE.2011.127.

CHRISTEN, P. 2012b. *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. [ebook]. Berlin: Springer. Tersedia di: https://doi.org/10.1007/978-3-642-31164-2.

JAMES P. H. 2020. Phonetic Spelling Algorithm Implementations for R, Journal of Statistical Software, 98(2), pp.1-21. Tersedia di: https://doi.org/10.18637/jss.v095.i08.

MAITY, S., DAS, N., MAJUMDER, M. & DASADHIKARY, D.R. 2021. Word Embedding and String-Matching Techniques for Automobile Entity Name Identification from Web Reviews, *EAI Endorsed Transactions on Scalable Information Systems*, 8(33), pp. 1–11. Tersedia di: https://doi.org/10.4108/EAI.14-5-2021.169918.

O'HARE, K., JUREK-LOUGHREY, A., & DE CAMPOS, C. 2022. High-Value Token-Blocking: Efficient Blocking Method for Record Linkage, ACM Transactions on Knowledge Discovery from Data, 16(2), pp. 1-17. Tersedia di: https://doi.org/10.1145/3450527

PRIYA, J., VINOTHINI, C., DINESH P.S., RESHMI, T.S. 2021. Data Deduplication Techniques: A Comparative Analysis, International Journal of Aquatic Science, 12(3), pp.1057-1065.

RAYKAR, N., KUMBHARKAR, P., JAYATIAL, D.H. 2023. De-duplication avoidance in regional names using an approach based on pronunciation, International Journal of Advances in Electrical Engineering, 4(10), pp.10-17. Tersedia di: https://doi.org/10.22271/27084574.2023.v4.i1a.32.

SANJAYA, A., SETIAWAN, A.B., MAHDIYAH, U., FARIDA, I.N., & PRASETYO, A.R. 2023. Pengukuran Kemiripan Makna Menggunakan Cosine Similarity dan Basis Data Sinonim Kata, *Jurnal Teknologi Informasi dan Ilmu Komputer*, 10(4), 747–752. Tersedia di: https://doi.org/10.25126/jtiik.20241046864.

VALENCIO, C.R., JARDINI, T., MARTINS, V.H.P., COLOMBINI, A.C., FORTES, M.Z. 2020. A System Proposal for Automated Data Cleaning Environment, Journal of Engineering and Technology for Industrial Applications, 6(25), pp. 4-15. Tersedia di: https://doi.org/10.5935/jetia.v6i25.685.

YERAI, D., VILARES, M., VILARES, J. 2024. On the performance of phonetic algorithms in microtext normalization, arXiv, pp.1-26. Tersedia di:https://arxiv.org/pdf/2402.02591.

Halaman ini sengaja dikosongkan