Analisis Komparatif Pengukuran Kemiripan Artikel Ilmiah menggunakan *Jaccard* dan *Levenshtein* serta *Blocking*

http://dx.doi.org/10.28932/jutisi.v9i2.6414

Riwayat Artikel
Received: 10 April 2023 | *Final Revision*: 07 Juli 2023 | *Accepted*: 11 Juli 2023

Muhammad Rizqi Nur✉ #1, Gandhi Surya Buana#2, Nur Aini Rakhmawati#3

Sistem Informasi, Institut Teknologi Sepuluh Nopember
Jalan Raya ITS, Kota Surabaya, 60111, Indonesia
[1]6026221012@mhs.its.ac.id
[2]6026221023@mhs.its.ac.id
[3]nur.aini@is.its.ac.id

✉Corresponding author: rizqinur2010@gmail.com

Abstrak **— Mesin pencarian artikel telah memudahkan studi literatur bagi akademisi, tapi mudah belum tentu akurat, terutama untuk topik *niche* tertentu. *Snowballing* bisa membantu tapi terbatas pada artikel awal yang dimiliki. *Database* artikel menyediakan rekomendasi artikel yang relevan, tapi terbatas pada artikel yang mereka miliki. Sebuah alat untuk mencari artikel mirip tanpa tergantung *database* tertentu bisa membantu, tapi sebelum itu, metode pengukuran kemiripan artikel yang tepat perlu ditentukan. Penelitian ini bertujuan mengevaluasi Weighted Jaccard *Measure* dan Levenshtein *distance* untuk *article matching* berdasarkan judul, penulis, dan *keywords*. Penelitian ini juga membandingkan penggunaan *overlap blocking* dan penghapusan *stop words*. Hasil Jaccard cukup buruk dengan nilai F1 0,366 dan AUC 0,255, Levenshtein + Jaccard lebih buruk lagi dengan nilai F1 0,065 dan AUC 0,514, sedangkan Levenshtein Set + Jaccard cukup baik dengan nilai F1 0,490 dan AUC 0,722. Ini berarti terlalu memperhitungkan urutan dalam teks justru berdampak buruk. Ini juga berarti metode pengukuran harus bisa memperhitungkan kata yang mirip tapi tidak persis sama. Selain itu, ditemukan bahwa menitikberatkan pembobotan pada judul menghasilkan hasil terbaik. *Overlap blocking* dan penghilangan *stop words* justru meningkatkan waktu pemrosesan rata-rata sebanyak 139,01% dan 37,82%. *Overlap blocking* dapat mengurangi jumlah pengukuran hingga hampir setengahnya dengan jumlah *overlap*=1, tapi jumlah *overlap* di atas 1 akan membuang banyak pasangan yang seharusnya mirip. *Overlap blocking* dengan jumlah *overlap*=1 tidak mempengaruhi kinerja, tapi penghilangan *stop words* justru menurunkan kinerja secara umum. Selain itu, ditemukan bahwa parameter *threshold* adalah parameter sensitif.**

Kata kunci— ***blocking; semantic matching; stop words*; studi literatur**

Comparative Analysis on Academic Paper Similarity using Jaccard and Levenshtein and Blocking

Abstract — ***Paper search engines have simplified literature reviews for academics, but ease doesn't guarantee accuracy, especially for certain niche topics. Snowballing may help but it is limited by the initially owned articles. Article repositories offer article recommendations, but it is limited to articles they own. Therefore, a general tool to search for similar articles can be helpful, but first, it is crucial to determine the appropriate method to measure article similarity. This study aims to evaluate Weighted Jaccard Measure and Levenshtein distance for article matching based on title, authors, and keywords. This study also compares usage of overlap blocking and stop word removal. Jaccard results were poor with F1 score of 0.366 and AUC score of 0.255, Levenshtein + Jaccard performed even worse with F1 score of 0.065 and AUC value of 0.514, but Levenshtein Set + Jaccard yielded good results with F1 score of 0.490 and AUC value of 0.722. This suggests that overemphasizing order in text is detrimental. This also suggests that the measurement method should account for similar but not identical words. It was also found that emphasizing weight on title yielded best results. Overlap blocking and stop words removal were found to increase processing time significantly instead by 139.01% and 37.82% respectively. Overlap blocking halved measurements with an overlap size of 1, but discarded many similar pairs with overlap size above 1. Overlap blocking at overlap 1 had no impact on prediction performance, whereas stop words removal decreased performance instead. Furthermore, the threshold parameter proved sensitive.***
Keywords— ***blocking; semantic matching; literature review; stop words***

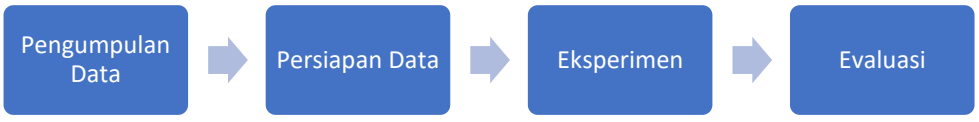
I. PENDAHULUAN

Kini pencarian artikel untuk studi literatur sudah mudah dilakukan. Tiap penerbit jurnal membuat mesin pencarian mereka sendiri dan ada juga Google Scholar yang dapat mengambil dari berbagai sumber [1]. Bahkan, mesin pencarian menjadi sebuah kebutuhan karena banyaknya artikel ilmiah yang tersebar luas di internet [2]. Namun, mudah bukan berarti akurat. Sebuah istilah dapat berarti hal yang berbeda dalam bidang yang berbeda, sehingga pencarian dengan kata kunci umum ini dapat menghasilkan beberapa artikel temuan yang sebenarnya sama sekali tidak relevan [3]. Sebagian mesin pencari menyediakan fitur *query* yang lebih kompleks; fitur ini dapat digunakan untuk mengatasi masalah ini hingga tingkat tertentu [4]. Akan tetapi, untuk *niche* tertentu terkadang hasil pencarian masih belum baik atau *query* yang kompleks justru secara tidak sengaja menghilangkan artikel yang relevan.

Seorang akademisi dapat menilai sendiri relevansi sebuah artikel, tapi waktu dan tingkat kesalahan meningkat dengan jumlah artikel. Sebagai alternatif, *snowballing* dapat dilakukan dengan membaca artikel-artikel yang dirujuk atau merujuk artikel yang sudah dinilai relevan [5]. Namun, artikel dari *snowballing* terbatas pada artikel awal yang dimiliki; pada akses dan pengetahuan penulis saat penulisan artikel. Sebagian *database* artikel memiliki fitur rekomendasi artikel relevan [6], tapi ini terbatas pada artikel yang mereka miliki saja. Dengan demikian, adanya alat umum untuk menemukan artikel relevan dengan sebuah atau kelompok contoh artikel akan sangat membantu. Penelitian sebelumnya telah dilakukan untuk pencarian artikel dari hubungan sitasi [2], [7], tapi belum ada penelitian untuk mencari berdasarkan kemiripan artikel tanpa tergantung pada *database* tertentu. Akan tetapi, sebelum membuat mesin pencari tersebut, perlu dicari metode yang tepat untuk mengukur kemiripan antar artikel.

Penelitian ini bertujuan melakukan pengukuran kemiripan artikel berdasarkan judul, penulis, dan *keyword* menggunakan *Weighted Jaccard Measure* dan mengevaluasinya. *Jaccard* dipilih karena cepat dan tidak tergantung bahasa [8]. Karena kata harus persis sama dalam Jaccard dan urutan mungkin berpengaruh, *Levenshtein distance* pada judul juga digunakan untuk dibandingkan. Penelitian ini menggunakan data artikel dari Google Scholar dengan *keywords* yang diperoleh dari Mendeley. Pengukuran untuk setiap kemungkinan pasangan dan dengan judul utuh akan memerlukan sumber daya komputasi yang sangat besar, sehingga perlu dilakukan *blocking*. *Blocking* mengurangi jumlah pasangan yang perlu dibandingkan [9]. Ini berarti ada pasangan artikel yang dibuang tanpa dibandingkan, dan artikel relevan mungkin terbuang karenanya. Jadi, penelitian ini juga melakukan analisis komparatif terhadap penggunaan *blocking*. Hasil penelitian ini diharapkan dapat membantu akademisi dalam melakukan studi literatur.

II. METODE



Gambar 1. Alur penelitian

Penelitian dilakukan dengan alur pada Gambar 1. Tahap pertama adalah pengumpulan data. Data, yaitu artikel, dikumpulkan dari Google Scholar menggunakan aplikasi Publish or Perish. Dikumpulkan sebanyak 100 data dengan *query* "*Topic modelling*" dari publikasi 5 tahun terakhir. Kemudian, data diekspor sebagai BiBTex untuk diimport ke Mendeley Desktop. Data diperbarui secara otomatis di Mendeley untuk mengambil *keyword*. *Keyword* yang masih kosong dicari secara

manual. Terakhir, diambil 50 data teratas dengan urutan awal yang memiliki *keyword*, yaitu urutan relevansi menurut Google Scholar.

Tahap kedua adalah persiapan data. Dari semua atribut, diambil hanya atribut judul, nama penulis, dan *keyword*. Data dipasangkan secara kombinatorik dengan dirinya sendiri sehingga diperoleh 1225 pasangan. Kemudian, tiap pasangan diberi label secara manual; 1 jika pasangan artikel mirip. Setiap teks diubah menjadi huruf kecil. Tanda baca dihilangkan untuk nama penulis. Titik koma dan "*and*" diubah menjadi koma untuk penulis dan *keywords*. Tokenisasi dilakukan dengan cara memisahkan string berdasarkan spasi untuk judul, sedangkan penulis dan *keywords* berdasarkan koma.

Tahap ketiga adalah eksperimen. Artikel diukur kemiripannya menggunakan *Jaccard Similarity Coefficient* (Rumus 1, [10]) untuk tiap atribut. Nilai *similarity* dari ketiga atribut digabungkan menggunakan *weighted sum* [11]. Pembobotan dilakukan dengan empat konfigurasi yang didasarkan pada nilai korelasi kemiripan terhadap label dihitung dengan *Kendall tau*. *Kendall tau* digunakan karena label biner. Data dianggap mirip jika nilai akhir di atas *threshold* tertentu. Karena kata harus persis sama dalam Jaccard dan urutan mungkin berpengaruh, *Levenshtein distance* (Rumus 2, [12]) pada judul juga digunakan untuk dibandingkan. Nilai *Levenshtein distance* ditransformasikan menjadi nilai *similarity* dalam jangkauan 0 hingga 1 dengan Rumus 3. Selain dengan cara biasa, *Levenshtein* juga digunakan dengan melakukan pra proses terhadap input dengan Algoritma 1 [13]. Ini dilakukan untuk mengantisipasi jika terlalu memperhitungkan urutan justru buruk tapi sifat *Levenshtein distance* yang bisa bekerja dengan kata yang tidak sama persis tetap diinginkan.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0, \\ lev_{a,b}(i-1,j-1) & if\ a[i] == a[j], \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1 \end{cases} & otherwise. \end{cases} \qquad (2)$$

$$levsim(a,b) = 1 - \frac{lev_{a,b}}{\max(|a|,|b|)} \qquad (3)$$

ALGORITMA 1
PRA PROSES INPUT LEVENSHTEIN SET [13]

1:	Require: string A, string B
2:	Let A_Set be the unique set of words in A
3:	Let B_Set be the unique set of words in B
4:	I_Set ← A_Set ∩ B_Set
5:	I ← join_by_space(sort(I_Set))
6:	A ← concat(I, join(sort(A_Set-I_Set)))
7:	B ← concat(I, join(sort(B_Set-I_Set)))
8:	return A, B

Dalam percobaan ini, hanya ada 1225 pasangan, sehingga masih memungkinkan untuk dihitung semuanya. Namun, untuk penggunaan sesungguhnya, perlu dilakukan *blocking* untuk meringankan beban komputasi. *Overlap blocking* dilakukan untuk membuang data yang dianggap tidak relevan, yaitu jika jumlah token yang sama di bawah jumlah batas tertentu [14]. *Overlap blocking* dilakukan menggunakan *library* py_entitymatching. Parameter *blocking* dioptimasi untuk meminimalkan jumlah pasangan relevan yang terbuang. Selanjutnya, stop words dihilangkan dari atribut yang telah di tokenisasi. *Stop words* adalah kata/*token* yang sering muncul dalam teks tapi biasanya tidak atau hanya sedikit berarti dalam pemrosesan teks [15]. Penghilangan *stopwords* dilakukan menggunakan *stopwords* bahasa inggris dari *library* NLTK.

Tahap terakhir adalah evaluasi. Evaluasi dilakukan dengan nilai akurasi (Rumus 4, [16]), F1 (Rumus 5, [16]), dan AUC (Rumus 8) serta *confusion matrix*. F1 dan AUC digunakan karena data mungkin tidak seimbang sehingga akurasi bukan metrik yang baik [16].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100\% \qquad (4)$$

F1 score (Rumus 5) adalah ukuran kinerja suatu model yang mempertimbangkan *precision* dan *recall*. *Precision* (Rumus 6) adalah proporsi prediksi positif yang benar, sedangkan *recall* (Rumus 7) adalah proporsi positif sebenarnya yang diidentifikasi dengan benar oleh model. Nilai F1 adalah rata-rata harmonis antara *precision* dan *recall* dalam jangkauan antara 0 dan 1 dengan nilai yang lebih tinggi menunjukkan kinerja yang lebih baik. [16]

$$F1 = \frac{2*Precision*Recall}{Precision+Recall} \quad (5)$$

$$Precision = \frac{TP}{TP+FP} * 100\% \quad (6)$$

$$Recall = \frac{TP}{TP+FN} * 100\% \quad (7)$$

Area di bawah kurva (*area under curve*, AUC) *receiving operator characteristic* (ROC) adalah salah satu ukuran yang umum digunakan untuk mengukur akurasi model. Kurva ROC adalah plot dari tingkat positif benar (*true positive rate*, TPR) terhadap tingkat positif palsu (*false positive rate*, FPR) pada berbagai *threshold*. AUC (Rumus 8) adalah area di bawah kurva ini dengan nilai berkisar antara 0 dan 1 dengan nilai yang lebih tinggi menunjukkan kinerja yang lebih baik. AUC sering digunakan dalam masalah klasifikasi biner untuk mengevaluasi kemampuan model dalam membedakan contoh positif dan negatif.

$$AUC = \int_{-\infty}^{\infty} TPR\left(FPR^{-1}(t)\right)dt \quad (8)$$

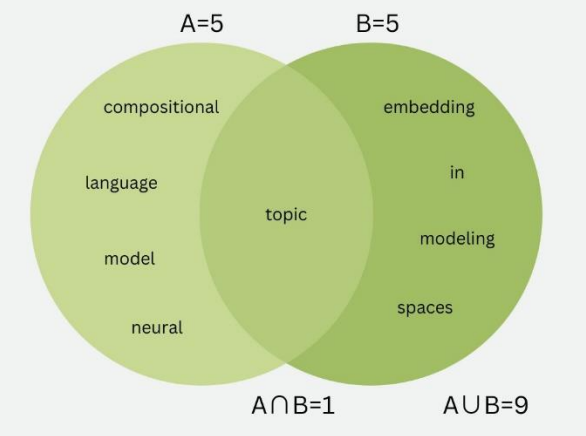
III. Hasil dan Pembahasan

Dari 1225 pasangan artikel, 74 pasangan (6.04%) dianggap mirip. Data sangat tidak seimbang sehingga akurasi adalah metrik yang buruk. Nilai *Jaccard similarity* untuk masing-masing atribut sangat kecil, tapi nilai *Levenshtein distance* pada judul cukup besar (Tabel 1). Sebagai contoh, nilai *Jaccard similarity* antara judul "topic compositional neural language model" dan "topic modeling in embedding spaces" hanya 0.111 (Persamaan $J = \frac{1}{9} = 0.111$ (9) karena kata yang beririsan hanya "topic" (Gambar 2). Contoh matriks perhitungan Levenshtein distance dengan input yang sama dilampirkan pada Gambar 3. Nilai paling atas dan kiri diisi dengan indeks berurutan. Perhitungan dimulai dari pojok kiri atas dengan melihat huruf sesuai titik yang dikerjakan. Jika huruf sama, diambil nilai di kiri atas, dan jika tidak, diambil nilai paling kecil di antara nilai di atas, kiri, atau kiri atas sel yang sedang dikerjakan. Ini dilakukan hingga matriks penuh. Nilai akhir Levenshtein distance ada pada pojok kanan bawah matriks, yaitu 27. Nilai ini ditransformasikan menjadi *similarity* 0.341 (Persamaan $levsim = 1 - \frac{27}{41} = 0.341$ (10). Levenshtein set mengubah input menjadi "topic compositional language model neural" dan "topic embedding in modeling spaces". Input yang telah diubah tersebut menghasilkan nilai similarity 0.390 (Persamaan $levsim_{set} = 1 - \frac{25}{41} = 0.390$ (11).

$$J = \frac{1}{9} = 0.111 \quad (9)$$

$$levsim = 1 - \frac{27}{41} = 0.341 \quad (10)$$

$$levsim_{set} = 1 - \frac{25}{41} = 0.390 \quad (11)$$



Gambar 2. Irisan contoh *Jaccard similarity*

TABEL 1
STATISTIK DESKRIPTIF NILAI *SIMILARITY SERTA KORELASI TERHADAP LABEL*

Similarity	Atribut	*Min*	*Q1*	*Median*	*Q3*	*Mean*	*Max*	Korelasi terhadap Label
Jaccard	Title1_Title2_Score	0.000	0.000	0.040	0.062	**0.043**	0.333	**0.251**
	Authors1_Authors2_Score	0.000	0.000	0.000	0.000	0.001	0.333	-0.016
	Keywords1_Keywords2_Score	0.000	0.000	0.000	0.000	0.022	1.000	0.142
Levenshtein	Title1_Title2_Score	0.111	0.207	0.230	0.254	0.232	0.395	0.106
Levenshtein Set	Title1_Title2_Score	0.135	0.229	0.255	0.291	**0.263**	0.555	**0.282**

		t	**o**	**p**	**i**	**c**		**m**	**o**	**d**	**e**	**l**	**i**	**n**	**g**		**i**	**n**		**e**	**m**	**b**	**e**	**d**	**d**	**i**	**n**	**g**		**s**	**p**	**a**	**c**	**e**	**s**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
t	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
o	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
p	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
i	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
c	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
c	7	6	5	4	3	2	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	25	26	27
o	8	7	6	5	4	3	2	2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
m	9	8	7	6	5	4	3	2	2	2	3	4	5	6	7	8	9	10	11	12	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
p	10	9	8	7	6	5	4	3	3	3	3	4	5	6	7	8	9	10	11	12	13	13	14	15	16	17	18	19	20	21	21	22	23	24	25
o	11	10	9	8	7	6	5	4	3	4	4	4	5	6	7	8	9	10	11	12	13	14	14	15	16	17	18	19	20	21	22	22	23	24	25
s	12	11	10	9	8	7	6	5	4	4	5	5	5	6	7	8	9	10	11	12	13	14	15	15	16	17	18	19	20	20	21	22	23	24	24
i	13	12	11	10	9	8	7	6	5	5	5	6	5	6	7	8	8	9	10	11	12	13	14	15	16	16	17	18	19	20	21	22	23	24	25
t	14	13	12	11	10	9	8	7	6	6	6	6	6	6	7	8	9	9	10	11	12	13	14	15	16	17	17	18	19	20	21	22	23	24	25
i	15	14	13	12	11	10	9	8	7	7	7	7	6	7	7	8	8	9	10	11	12	13	14	15	16	16	17	18	19	20	21	22	23	24	25
o	16	15	14	13	12	11	10	9	8	8	8	8	7	7	8	8	9	9	10	11	12	13	14	15	16	17	17	18	19	20	21	22	23	24	25
n	17	16	15	14	13	12	11	10	9	9	9	9	8	7	8	9	9	9	10	11	12	13	14	15	16	17	17	18	19	20	21	22	23	24	25
a	18	17	16	15	14	13	12	11	10	10	10	10	9	8	8	9	10	10	10	11	12	13	14	15	16	17	18	18	19	20	21	21	22	23	24
l	19	18	17	16	15	14	13	12	11	11	11	10	10	9	9	9	10	11	11	11	12	13	14	15	16	17	18	19	19	20	21	22	22	23	24
	20	19	18	17	16	15	14	13	12	12	12	11	11	10	10	9	10	11	11	12	12	13	14	15	16	17	18	19	19	20	21	22	23	23	24
n	21	20	19	18	17	16	15	14	13	13	13	12	12	11	11	10	10	10	11	12	13	13	14	15	16	17	17	18	19	20	21	22	23	24	24
e	22	21	20	19	18	17	16	15	14	14	13	13	13	12	12	11	11	11	11	11	12	13	13	14	15	16	17	18	19	20	21	22	23	23	24
u	23	22	21	20	19	18	17	16	15	15	14	14	14	13	13	12	12	12	12	12	12	13	14	14	15	16	17	18	19	20	21	22	23	24	24
r	24	23	22	21	20	19	18	17	16	16	15	15	15	14	14	13	13	13	13	13	13	13	14	15	15	16	17	18	19	20	21	22	23	24	25
a	25	24	23	22	21	20	19	18	17	17	16	16	16	15	15	14	14	14	14	14	14	14	14	15	16	16	17	18	19	20	21	21	22	23	24
l	26	25	24	23	22	21	20	19	18	18	17	16	17	16	16	15	15	15	15	15	15	15	15	15	16	17	17	18	19	20	21	22	22	23	24
	27	26	25	24	23	22	21	20	19	19	18	17	17	17	17	16	16	16	15	16	16	16	16	16	16	17	18	18	18	19	20	21	22	23	24
l	28	27	26	25	24	23	22	21	20	20	19	18	18	18	18	17	17	17	16	16	17	17	17	17	17	17	18	19	19	19	20	21	22	23	24
a	29	28	27	26	25	24	23	22	21	21	20	19	19	19	19	18	18	18	17	17	17	18	18	18	18	18	18	19	20	20	20	20	21	22	23
n	30	29	28	27	26	25	24	23	22	22	21	20	20	19	20	19	19	18	18	18	18	18	19	19	19	19	18	19	20	21	21	21	21	22	23
g	31	30	29	28	27	26	25	24	23	23	22	21	21	20	19	20	20	19	19	19	19	19	19	20	20	20	19	18	19	20	21	22	22	22	23
u	32	31	30	29	28	27	26	25	24	24	23	22	22	21	20	20	21	20	20	20	20	20	20	20	21	21	20	19	19	20	21	22	23	23	23
a	33	32	31	30	29	28	27	26	25	25	24	23	23	22	21	21	21	21	21	21	21	21	21	21	21	22	21	20	20	20	21	21	22	23	24
g	34	33	32	31	30	29	28	27	26	26	25	24	24	23	22	22	22	22	22	22	22	22	22	22	22	22	22	21	21	21	21	22	22	23	24
e	35	34	33	32	31	30	29	28	27	27	26	25	25	24	23	23	23	23	23	22	23	23	22	23	23	23	23	22	22	22	22	22	23	22	23
	36	35	34	33	32	31	30	29	28	28	27	26	26	25	24	23	24	24	23	23	23	24	23	23	24	24	24	23	22	23	23	23	23	23	23
m	37	36	35	34	33	32	31	30	29	29	28	27	27	26	25	24	24	25	24	24	23	24	24	24	24	25	25	24	23	23	24	24	24	24	24
o	38	37	36	35	34	33	32	31	30	30	29	28	28	27	26	25	25	25	25	25	24	24	25	25	25	25	26	25	24	24	24	25	25	25	25
d	39	38	37	36	35	34	33	32	31	30	30	29	29	28	27	26	26	26	26	26	25	25	25	25	25	26	26	26	25	25	25	25	26	26	26
e	40	39	38	37	36	35	34	33	32	31	30	30	30	29	28	27	27	27	27	26	26	26	25	26	26	26	27	27	26	26	26	26	26	26	27
l	41	40	39	38	37	36	35	34	33	32	31	30	31	30	29	28	28	28	28	27	27	27	26	26	27	27	27	28	27	27	27	27	27	27	**27**

Gambar 3. Matriks perhitungan Levenshtein distance; nilai akhir di pojok kanan bawah

Nilai korelasi Kendall tau antara nilai *similarity* dan label dilampirkan pada Tabel 1. Nilai Jaccard memiliki korelasi yang rendah dengan label kecuali untuk judul. Akan tetapi, nilai Levenshtein untuk judul memiliki korelasi yang jauh lebih rendah, sedangkan nilai Levenshtein set justru paling tinggi.

BERDASARKAN NILAI KORELASI, DIHITUNG PEMBOBOTAN MENGGUNAKAN RUMUS 12 SEBAGAI DASAR DAN DILAMPIRKAN TABEL 2. DENGAN KONFIGURASI PEMBOBOTAN TERSEBUT, DIHITUNG NILAI KEMIRIPAN AKHIR. NILAI AKHIR YANG HANYA BERDASARKAN JACCARD SANGAT KECIL, TAPI NILAI AKHIR YANG JUGA MENGGUNAKAN LEVENSHTEIN CUKUP BESAR (

TABEL 3). KORELASI NILAI AKHIR TERHADAP LABEL DILAMPIRKAN PADA

Tabel 3. Nilai akhir Levenshtein Set + Jaccard memiliki korelasi paling tinggi dengan label. Meningkatkan bobot nama penulis dan *keyword* menurunkan nilai korelasi, sedangkan meningkatkan bobot judul juga meningkatkan nilai korelasi.

$$BaseWeights = \frac{\max(0,Correlation)}{\sum \max(0,Correlation)} \qquad (12)$$

TABEL 2
KONFIGURASI PEMBOBOTAN

Similarity	Pembobotan	Judul	Nama Penulis	*Keyword*	Keterangan
Jaccard	0	0.64	0.00	0.36	Basis dari nilai korelasi
	1	0.59	0.10	0.31	Peningkatan bobot penulis
	2	0.54	0.00	0.46	Peningkatan bobot keyword
	3	0.74	0.00	0.26	Peningkatan bobot judul
Levenshtein + Jaccard	0	0.43	0.00	0.57	Basis dari nilai korelasi
	1	0.38	0.10	0.52	Peningkatan bobot penulis
	2	0.33	0.00	0.67	Peningkatan bobot keyword
	3	0.53	0.00	0.47	Peningkatan bobot judul
Levenshtein Set + Jaccard	0	0.67	0.00	0.33	Basis dari nilai korelasi
	1	0.62	0.10	0.28	Peningkatan bobot penulis
	2	0.57	0.00	0.43	Peningkatan bobot keyword
	3	0.77	0.00	0.23	Peningkatan bobot judul

TABEL 3
STATISTIK DESKRIPTIF NILAI AKHIR *SERTA KORELASI TERHADAP LABEL*

Similarity	Pembobotan	*Min*	*Q1*	*Median*	*Q3*	*Mean*	*Max*	Korelasi terhadap Label
Jaccard	0	0.000	0.000	0.029	0.053	0.035	0.404	0.249
	1	0.000	0.000	0.026	0.049	0.032	0.351	0.250
	2	0.000	0.000	0.025	0.048	0.033	0.497	0.241
	3	0.000	0.000	0.032	0.057	**0.037**	0.311	**0.255**
Levenshtein + Jaccard	0	0.051	0.091	0.102	0.119	0.113	0.675	0.151
	1	0.045	0.080	0.090	0.106	0.100	0.613	0.150
	2	0.039	0.070	0.078	0.092	0.092	0.751	0.150
	3	0.062	0.112	0.126	0.146	**0.134**	0.600	**0.152**
Levenshtein Set + Jaccard	0	0.091	0.156	0.176	0.203	0.184	0.531	0.282
	1	0.084	0.144	0.163	0.188	0.169	0.466	0.283
	2	0.077	0.132	0.150	0.177	0.160	0.601	0.273
	3	0.104	0.179	0.201	0.230	**0.208**	0.461	**0.287**

Suatu nilai *threshold* diperlukan untuk mentransformasikan nilai *similarity* menjadi label biner. Nilai *similarity* dianggap 1 (mirip) jika di atas *threshold* tersebut dan 0 jika tidak. Nilai *threshold* ditentukan untuk tiap pembobotan menggunakan Rumus 13. Nilai *threshold* ditentukan sebagai nilai yang membagi distribusi nilai *similarity* dengan label 0 dan distribusi nilai *similarity* dengan label 1. W_0 adalah nilai akhir dengan label 0 (tidak mirip) dan W_1 adalah nilai akhir dengan label 1 (mirip), sedangkan Q1 dan Q3 adalah kuartil 1 dan 3. Awalnya nilai *threshold* ditentukan dengan rata-rata nilai *similarity* tertinggi label 0 dan nilai *similarity* terendah label 1, tapi menambahkan nilai kuartil dalam rumus menghasilkan kinerja yang lebih baik. Nilai *threshold* dilampirkan pada Tabel 4.

TABEL 4
NILAI *THRESHOLD* UNTUK TIAP KONFIGURASI *SIMILARITY* DAN PEMBOBOTAN

Similarity	Pembobotan			
	0	1	2	3
Jaccard	0.128	0.112	0.149	0.107
Levenshtein + Jaccard	0.240	0.216	0.242	0.237

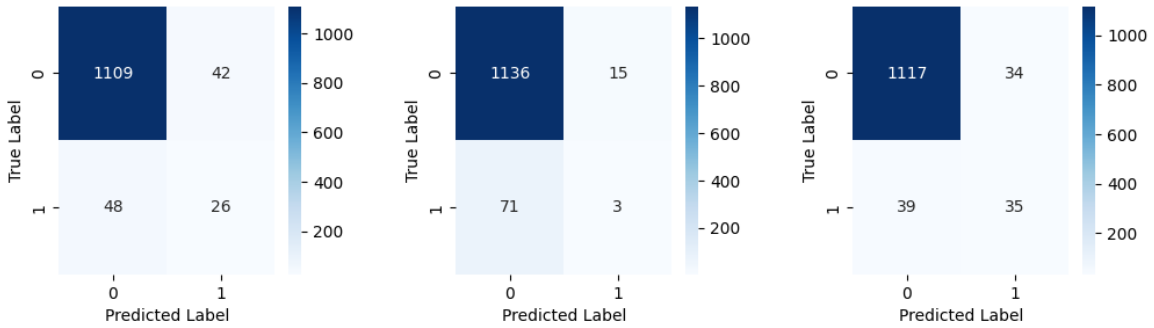
Levenshtein Set + Jaccard	0.280	0.252	0.276	0.282

$$Threshold(W) = \frac{1}{4} * (Q3(W_0) + Q1(W_1) + max(W_0) + \min(W_1)) \quad (13)$$

Evaluasi pengukuran kemiripan ini dilampirkan pada Tabel 5. Nilai akurasi cukup tinggi, tapi sebagaimana telah disebutkan sebelumnya, akurasi bukan metrik yang baik untuk data yang tidak seimbang. Sedangkan, nilai F1 dan AUC cukup buruk, khususnya untuk metode *Levenshtein + Jaccard*. Ini berarti terlalu memperhitungkan urutan dalam teks justru berdampak buruk. Ini juga berarti metode pengukuran harus bisa memperhitungkan kata yang mirip tapi tidak persis sama. Ini dapat dilihat pada *confusion matrix* (Gambar 4) bahwa hanya 26 dari 74 pasangan mirip yang diprediksi dengan benar untuk metode *Jaccard*. Metode Levenshtein + Jaccard justru lebih buruk dengan hanya 3 prediksi tepat, sedangkan metode *Levenshtein Set + Jaccard* memprediksi 35 pasangan dengan tepat. Semua metode mendapat hasil terbaik dengan konfigurasi 3. Ini berarti judul adalah *feature* paling berpengaruh dan seharusnya pembobotan dititikberatkan pada judul.

TABEL 5
HASIL EVALUASI TANPA *BLOCKING*

Similarity	Pembobotan	*Accuracy*	*Precision*	*Recall*	F1	AUC
Jaccard	0	0.936	0.419	0.176	0.248	0.580
	1	0.933	0.395	0.203	0.268	0.591
	2	0.934	0.348	0.108	0.165	0.548
	3	0.927	0.382	0.351	**0.366**	**0.657**
Levenshtein + Jaccard	0	0.931	0.077	0.014	0.023	0.502
	1	0.930	0.071	0.014	0.023	0.501
	2	0.931	0.143	0.027	0.045	0.508
	3	0.930	0.167	0.041	**0.065**	**0.514**
Levenshtein Set + Jaccard	0	0.944	0.571	0.324	0.414	0.654
	1	0.944	0.553	0.351	0.430	0.667
	2	0.944	0.636	0.189	0.292	0.591
	3	0.940	0.507	0.473	**0.490**	**0.722**



(a) (b) (c)

Gambar 4. Confusion matrix untuk (a) Jaccard, (b) Levenshtein + Jaccard dan (c) Levenshtein Set dengan pembobotan terbaik

Blocking dilakukan untuk meringankan beban komputasi. *Overlap blocking* dilakukan untuk membuang pasangan yang dianggap tidak relevan. *Overlap blocking* dilakukan menggunakan *library* py_entitymatching. Jumlah *overlap* yang optimal adalah 1, karena di *overlap blocking* dengan jumlah *overlap* lebih dari 1 menghilangkan pasangan pasangan mirip dengan jumlah yang signifikan (Tabel 6). Waktu proses dihitung dari rata-rata 100 percobaan. Selain *overlap blocking*, dilakukan juga penghilangan *stopwords* dan tanda baca bahasa inggris menggunakan library NLTK ditambah *stopwords* berikut: *'the', 'and', 'of', 'in', 'a', 'an', 'to'*. Penambahan *overlap blocking* dan *stop words* ternyata justru meningkatkan waktu pemrosesan secara signifikan. *Overlap blocking*, khususnya, meningkatkan waktu pemrosesan hingga hampir 2 kali lipat. Ini dapat disebabkan oleh implementasi yang tidak efisien atau *Jaccard* dan *Levenshtein* memang sudah cukup cepat dalam kasus ini sehingga *overhead* dari *blocking* lebih besar dari manfaatnya.

Evaluasi untuk *blocking* dilakukan dengan konfigurasi 3 untuk tiap metode. *Threshold* ditentukan ulang untuk *stopwords*, yaitu 0.083 untuk *Jaccard*, 0.233 untuk *Levenshtein + Jaccard*, dan 0.360 untuk *Levenshtein Set + Jaccard*. Hasil evaluasi dengan *blocking* dilampirkan pada Tabel 7. Nilai evaluasi tidak terpengaruh oleh *overlap blocking* dengan jumlah *overlap* 1,

tapi terpengaruh oleh penghilangan *stop words*. Nilai justru menurun signifikan jika penghilangan *stop words* dilakukan tanpa penyesuaian *threshold*. Ini berarti *threshold* adalah parameter yang sangat sensitif dan perlu ditentukan dengan baik. Akan tetapi, meskipun dengan penyesuaian *threshold*, nilai dengan penghilangan *stopwords* tetap lebih buruk secara umum. Penghilangan *stopwords* hanya berdampak positif untuk metode *Levenshtein + Jaccard*.

TABEL 6
HASIL PERCOBAAN *OVERLAP BLOCKING* DAN *STOP WORDS*

Jumlah Overlap	Jumlah Data yang Tersisa	Pasangan Mirip yang Hilang (dari 74)		Waktu Proses (s)							
		Jumlah	Persentase	Jaccard		Levenshtein + Jaccard		Levenshtein Set + Jaccard		Rata -Rata	
0	1225	0	0.00%	**0.168**		**0.137**		**0.163**		0.156	
1	732	1	1.35%	0.399	+137.50%	0.351	+156.20%	0.364	+123.31%	0.371	+139.01%
2	299	21	28.38%	0.363	+116.07%	0.315	+129.93%	0.331	+103.07%	0.336	+116.36%
3	111	38	51.35%	0.334	+98.81%	0.307	+124.09%	0.326	+100.00%	0.322	+107.63%
4	43	51	68.92%	0.338	+101.19%	0.302	+120.44%	0.335	+105.52%	0.325	+109.05%
5	18	64	86.49%	0.328	+95.24%	0.302	+120.44%	0.315	+93.25%	0.315	+102.98%
Penghilangan stop words				0.252	+50.00%	0.192	+40.15%	0.201	+23.31%	0.215	+37.82%

TABEL 7
HASIL EVALUASI DENGAN *BLOCKING*

Metode	Blocking	Accuracy	Precision	Recall	F1	AUC
Jaccard	*None*	0.927	0.382	0.351	**0.366**	**0.657**
	Overlap 1	0.927	0.382	0.351	**0.366**	**0.657**
	Stopwords	0.926	0.333	0.230	0.272	0.600
	Stopwords ~ threshold	0.904	0.261	0.324	0.289	0.633
Levenshtein + Jaccard	*None*	0.930	0.167	0.041	0.065	0.514
	Overlap 1	0.930	0.167	0.041	0.065	0.514
	Stopwords	0.931	0.222	0.054	0.087	0.521
	Stopwords ~ threshold	0.931	0.238	0.068	**0.105**	**0.527**
Levenshtein Set + Jaccard	*None*	0.940	0.507	0.473	**0.490**	0.722
	Overlap 1	0.940	0.507	0.473	**0.490**	0.722
	Stopwords	0.941	0.516	0.446	0.478	0.710
	Stopwords ~ threshold	0.927	0.424	0.568	0.486	**0.759**

IV. SIMPULAN

Penelitian ini telah melakukan pengukuran kemiripan artikel berdasarkan judul, nama penulis, dan *keyword* menggunakan *weighted Jaccard measure*, *Levenshtein + Jaccard*, dan *Levenshtein Set + Jaccard*. Dengan menggunakan bobot yang ditentukan dengan Kendall tau, hasil pengukuran dengan *Jaccard* cukup buruk, hasil pengukuran dengan *Levenshtein + Jaccard* lebih buruk lagi, tapi hasil pengukuran dengan *Levenshtein Set + Jaccard* cukup baik. Ini berarti terlalu memperhitungkan urutan dalam teks justru berdampak buruk. Ini juga berarti metode pengukuran harus bisa memperhitungkan kata yang mirip tapi tidak persis sama. Selain itu, ditemukan bahwa menitikberatkan pembobotan pada judul menghasilkan hasil terbaik. *Overlap blocking* dan penghilangan *stopwords* justru meningkatkan waktu pemrosesan secara signifikan. *Overlap blocking* dapat mengurangi jumlah pengukuran hingga hampir setengahnya dengan jumlah *overlap*=1, tapi jumlah *overlap* di atas 1 akan membuang banyak pasangan yang seharusnya mirip. Overlap blocking dengan jumlah overlap=1 tidak mempengaruhi kinerja, tapi penghilangan *stopwords* justru menurunkan kinerja secara umum. Selain itu, ditemukan bahwa parameter *threshold* sensitif dan perlu ditentukan dengan baik. Penelitian selanjutnya disarankan mencari cara menentukan *threshold* yang optimal atau menguji metode lain yang tidak terlalu sensitif terhadap urutan tapi bisa memperhitungkan kata yang mirip.

DAFTAR PUSTAKA

[1] A. Martín-Martín, M. Thelwall, E. Orduna-Malea, and E. Delgado López-Cózar, "Google Scholar, Microsoft Academic, Scopus, Dimensions, Web of Science, and OpenCitations' COCI: a multidisciplinary comparison of coverage via citations," *Scientometrics*, vol. 126, 2021.

[2] M. Färber, A. Sampath, and A. Jatowt, "PaperHunter: A System for Exploring Papers and Citation Contexts," in *." In: Azzopardi, L., Stein, B., Fuhr, N., Mayr, P., Hauff, C., Hiemstra, D. (eds) Advances in Information Retrieval. ECIR 2019. Lecture Notes in Computer Science()*, 2019, vol. 11438.

[3] M. K. Linnenluecke, M. Marrone, and A. K. Singh, "Conducting systematic literature reviews and bibliometric analyses," *Aust. J. Manag.*, vol. 45, no. 2, pp. 175–194, 2020.

[4] H. A. Mohamed Shaffril, S. F. Samsuddin, and A. Abu Samah, "The ABC of systematic literature review: the basic methodological guidance for beginners," *Qual. Quant.*, vol. 55, no. 4, pp. 1319–1346, 2021.
[5] E. Mourão, J. F. Pimentel, L. Murta, M. Kalinowski, E. Mendes, and C. Wohlin, "On the performance of hybrid search strategies for systematic literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 123, no. March, 2020.
[6] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific paper recommendation: A survey," *IEEE Access*, vol. 7, pp. 9324–9339, 2019.
[7] G. Papadakis and G. Paliouras, "MyCites: An intelligent information system for maintaining citations," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5138 LNAI, pp. 371–376, 2008.
[8] D. W. Prakoso, A. Abdi, and C. Amrit, "Short text similarity measurement methods: a review," *Soft Comput.*, vol. 25, no. 6, pp. 4699–4723, 2021.
[9] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "Blocking and Filtering Techniques for Entity Resolution: A Survey," *ACM Comput. Surv.*, vol. 53, no. 2, 2020.
[10] N. E. Diana and I. Hanana Ulfa, "Measuring performance of n-gram and jaccard-similarity metrics in document plagiarism application," *J. Phys. Conf. Ser.*, vol. 1196, no. 1, 2019.
[11] J. Xu, J. Mu, and G. Chen, "A multi-view similarity measure framework for trouble ticket mining," *Data Knowl. Eng.*, vol. 127, no. September 2018, p. 101800, 2020.
[12] D. K. Po, "Similarity Based Information Retrieval Using Levenshtein Distance Algorithm," *Int. J. Adv. Sci. Res. Eng.*, vol. 06, no. 04, pp. 06–10, 2020.
[13] SeatGeek, "thefuzz/thefuzz/fuzz.py at master · seatgeek/thefuzz," *Github*, 2023.
[14] P. Konda, S. S. Seshadri, E. Segarra, B. Hueth, and A. H. Doan, "Executing entity matching end to end: A case study," 2019.
[15] F. Zhang, H. Fleyeh, X. Wang, and M. Lu, "Construction site accident analysis using text mining and natural language processing techniques," *Autom. Constr.*, vol. 99, pp. 238–248, 2019.
[16] J. Pereira and F. Saraiva, "A Comparative Analysis of Unbalanced Data Handling Techniques for Machine Learning Algorithms to Electricity Theft Detection," *2020 IEEE Congr. Evol. Comput. CEC 2020 - Conf. Proc.*, 2020.