Analisis perbandingan algoritma Nazief Adriani dan Levenshtein Distance untuk mengukur tingkat similaritas berita berbahasa Jawa menggunakan Rabin Krap

Danang Kastowo*[1], Andy Saputra[2], Wachid Daga Suryono[3] dan Erna Setyowati[4]

1 **PT. Dua Empat Tujuh**
Yogyakarta, Indonesia
danang.kastowo@labs247.id

2,4 **Universitas AMIKOM Yogyakarta.**
Jl. Padjajaran, Ring Road Utara, Kel. Condongcatur, Kec. Depok, Kab. Sleman, Prop. Daerah Istimewa Yogyakarta.
andy@students.amikom.ac.id; erna@students.amikom.ac.id

3 **Universitas Muhammadiyah Surakarta**
Jl. A. Yani, Mendungan, Pabelan, Kec. Kartasura, Kabupaten Sukoharjo.
wachiddaga@ums.ac.id

Abstrak

Bagi masyarakat di Indonesia bahasa daerah merupakan bahasa sehari-hari yang biasa digunakan untuk berkomunikasi. Salah satunya adalah bahasa Jawa. Pada penelitian berbasis bahasa alami, bahasa daerah tergolong bahasa yang sulit untuk dikembangkan, mengingat ketersediaan jumlah dataset yang terbatas. Penelitian ini melakukan analisis terhadap 2 metode *stemming* kata, yaitu metode Nazief- Adriani dan Levenshtein Distance untuk menyelesaikan proses *stemming* kata berbahasa Jawa. Penelitian ini ingin mengetahui metode yang sesuai dengan akurasi terbaik untuk *stemming* kata berbahasa Jawa. Selain itu penelitian ini juga mempertimbangkan pembobotan kata untuk menghasilkan akurasi similaritas artikel yang lebih baik. Metode Nazief Adriani menghasilkan nilai rata-rata similaritas sebesar 6,8% dengan waktu rata-rata eksekusi 0,0443 detik.

Kata Kunci nazief adriani, levenshtein distance, bahasa jawa, tf-idf, rabbin-karp

Digital Object Identifier 10.36802/jnanaloka.v3-no1-1-10

1 Pendahuluan

Bahasa merupakan sarana untuk berkomunikasi. Di dunia ini banyak sekali bahasa, antara lain bahasa Indonesia dan Jawa. Pengetahuan tentang bahasa sangatlah penting karena dalam sebuah percakapan atau pembicaraan memerlukan sebuah bahasa. Bahasa jawa adalah bahasa yang digunakan penduduk suku bangsa Jawa di Jawa Tengah, Yogyakarta, dan Jawa Timur. Selain itu, Bahasa Jawa juga digunakan oleh penduduk yang tinggal beberapa daerah lain seperti di Banten terutama kota Serang, kabupaten Serang, kota Cilegon dan kabupaten Tangerang, Jawa Barat khususnya kawasan Pantai utara terbentang dari pesisir utara Karawang, Subang, Indramayu, kota Cirebon dan kabupaten Cirebon [1].

* Corresponding author.

Jnanaloka
Jurnal Open Access
Yayasan Lentera Dua Indonesia

Bahasa Jawa adalah bahasa daerah yang menjadi ciri khas orang jawa, yang digunakan untuk komunikasi sehari-hari oleh masyarakat Jawa. Di era modern ini penggunaan bahasa Jawa dalam komunikasi sehari-hari mulai tergeser karena mengalami banyak tantangan. Salah satu cara mempertahankan bahasa Jawa adalah dengan cara mempelajari dan mengamalkannya dalam kehidupan sehari-hari. Istilah dalam bahasa Jawa banyak yang mengalami perubahan kata dan makna, karena terdapat banyak imbuhan dalam penyusunan kata Jawa [2]. Banyaknya imbuhan (ater-ater), akhiran (panambang), sisipan (seselan) dan juga awalan-akhiran pada kata dalam bahasa Jawa menjadikan proses pengembalian kata ke bentuk dasarnya memerlukan algoritma *stemming* [3]. Algortima yang digunakan untuk melakukan stemming bahasa Jawa pada penelitian ini adalah Nazief-Adriani dan Levenshtein Distance. Hasil dari *stemming* dari kedua algoritma tersebut akan dilakukan proses pembobotan kata menggunakan *Term Frequency–Inverse Document Frequency* (TF-IDF) yang kemudian akan dilakukan analisis lebih lanjut menggunakan algoritma Rabbin-Karp untuk mengukur similaritas dokumen.

Penelitian sebelumnya yang telah membahas mengenai *stemming* bahasa Jawa dengan algoritma Nazief-Adriani adalah penelitian yang dilakukan oleh [4] yang menghasilkan tingkat akurasi sebesar 95% untuk 366 kata [4]. Penelitian lain yang dengan menggunakan *Damerau Levenshtein Distance* (DLD) dilakukan oleh [3] dengan nilai akurasi sebesar 49,6%. Penelitian pertama dengan dataset sebanyak 366 kata perlu pengujian lebih loanjut pada tingkat akurasi yang didapatkan dengan memberikan tambahan jumlah kata yang digunakan. Pada penelitian kedua akurasi yang dihasilkan masih terbilang rendah, sehingga masih memungkinkan untuk dilakukan penelitian lebih lanjut terhadap algoritma Levenshtein yang digunakan. Perbandingan hasil *stemming* bahasa Jawa dari algoritma Nazief-Adriani dan algoritma Levenshtein dilakukan dalam penelitian didasarkan dari kedua penelitian tersebut. Disamping itu, penelitian ini juga menambahkan pembobotan kata menggunakan TF-IDF atas hasil *stemming* serta mengukur similaritas dua dokumen atau lebih berbahasa Jawa dengan menggunakan algoritma Rabbin-Karp.

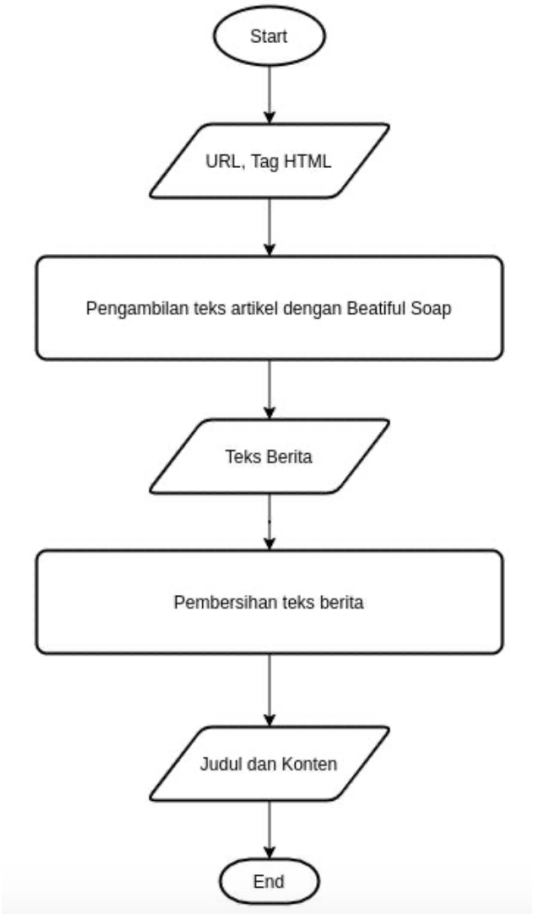
Tujuan dari penelitian ini adalah mengetahui metode yang sesuai untuk proses *stemming* pada bahasa Jawa dengan nilai akurasi terbaik, dan mengetahui pengaruh pembobotan kata terhadap hasil akurasi yang dihasilkan. Fokus penelitian yang dilakukan adalah mencari akurasi terbaik dari dua algoritma *stemming* yang digunakan, yaitu Nazief-Adriani dan Levenshtein Distance dengan pembobotan kata dengan TF-IDF.

2 Metodologi

Pada penelitian ini data yang digunakan berasal dari sebuah situs berbahasa jawa, yaitu Pawarta Bahasa Jawa. Pada situs tersebut, terdapat beberapa kategori berita diantara tentang ekonomi, bencana, kriminal, dan teknologi [5]. Proses pengambilan text pada artikel berita tersebut menggunakan pemrograman python dengan menggunakan library BeautifulSoup. Skenario pengambilan data artikel dapat dilihat pada Gambar 1.

Pengambilan data ini dilakukan melalui URL dan memproses *tag* HTML dimana data artikel tersebut dituliskan. Contoh *tag* html yang digunakan `<div id="body-post-it"> artikel </div>`. Setelah mendapatkan teks artikel, kemudian data tersebut dibersihkan dari *tag* HTML untuk mendapatkan data judul dan konten berita. Konten berita yang telah diperoleh kemudian dilakukan teks *preprocessing* seperti berikut:

1. penghapusan pungtuasi atau tanda baca,
2. mengubah semua huruf dalam dokumen berita menjadi huruf kecil (*case folding*) agar semua karakter di dalam dokumen dapat disamakan oleh sistem dengan dokumen lainnya,



Gambar 1 Proses pengambilan data artikel dari situs berita

3. tokenisasi yaitu proses pemecahan kalimat menjadi per kata berdasarkan spasi,
4. penghapusan karakter karakter penghubung seperti imbuan (ater-ater), akhiran (panambang), sisipan (seselan) serta imbuhan dan akhiran (*stopwords*),
5. pencarian kata dasar bahasa jawa dari kata yang berimbuhan berdasarkan kamus bahasa jawa yang telah ditetapkan (*stemming*).

Di dalam proses tokenisasi, sistem juga melakukan penghapusan simbol atau karakter yang tidak relevan seperti tanda baca, angka dan sebagainya dalam satu waktu.

2.1 Metode Nazief-Adriani

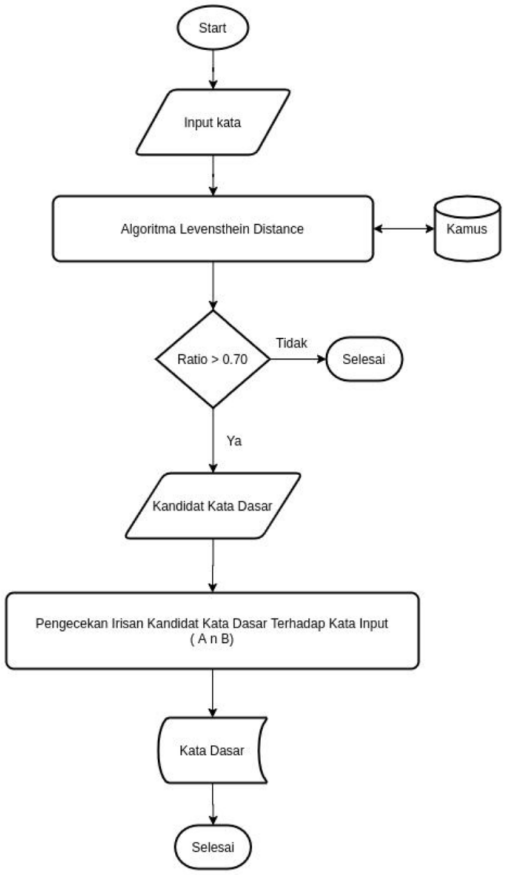
Pada algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani memiliki beberapa tahapan proses [4]. Dari tahapan pengumpulan dokumen, kemudian dilakukan data *cleaning* dengan menghapus *punctuation*, angka dan mengubah teks menjadi huruf kecil. Hasil ini selanjutnya digunakan untuk proses penghapus kata yang dinilai tidak memiliki penambahan *prefix*, *suffix*, dan kombinasi keduanya. Pada tahapan ini juga kata yang memiliki jumlah karakter tiga atau kurang tiga akan dihapus. Setelah tahapan tersebut selesai, maka proses selanjutnya adalah tokenisasi atau membuat data korpus yang berasal dari dokumen teks tersebut. Hasil dari proses tokenisasi dikumpulkan di dalam dataset.

Dataset tersebut digunakan untuk pencarian akar kata dengan teknik *confix stripping* skema Naizief Adriani. Akar kata tersebut dievaluasi guna menentukan tingkat akurasi dari teknik tersebut dan dibandingkan dengan kamus data bahasa jawa. Jika ada kata yang sama maka akan digunakan di tahap selanjutnya dan jika kata tidak ada dalam kamus maka dihilangkan.

2.2 Metode Levenshtein Distance

Levenshtein Distance (LD) lebih dikenal dengan *edit distance* adalah mencari jumlah minimum mutasi titik (*point mutation*) yang diperlukan untuk merubah suatu *string* ke *string* yang lain. Mutasi titik tersebut adalah *insertion*, *substitution* dan *deletion* [6; 7]. Algoritma ini digunakan untuk mencari kata dasar baru terhadap kata yang memiliki imbuhan (ater-ater), akhiran (panambang), sisipan (seselan) serta imbuhan dan akhiran dengan membandingkan dengan kamus (kumpulan kata dasar bahasa jawa) untuk menentukan kanditat kata dasar.

Proses penentuan kandidat kata dasar pada penelitian ini diambil dengan cara menentukan nilai jarak terkecil kemudian mengukur tingkat rasio kandidat kata dasar tersebut terhadap kata input. Nilai rasio di atas 0.70 akan digunakan sebagai kandidat kata dasar yang kemudian dicari irisan kata yang terkandung dengan kata input. Gambar 2 menunjukan skema proses *stemming* yang dilakukan pada penelitian ini dengan metode Levenshtein Distance.



Gambar 2 Proses *stemming* dengan algoritma Levenstein Distance

2.3 Pembobotan kata dengan (TF-IDF)

TF-IDF adalah sebuah metode pembobotan kata yang memanfaatkan index dari teks, metode ini menghitung kata yang muncul dalam sebuah dokumen [8; 9]. Metode ini menghitung nilai *term frequency* (TF) dan *inverse document frequency* (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Proses penghitungan pembobotan kata dengan metode TF-IDF dimulai dengan mendapatkan token kata dasar dari kedua algoritma yang digunakan. Kemudian token kata dasar tersebut digunakan sebagai input untuk menghitung nilai TF dan nilai DF. Selanjutnya menghitung nilai IDF dengan parameter input berupa nilai DF dan jumlah dokumen testing yang digunakan. Setelah mendapatkan nilai IDF dari token-token tersebut, tahap terakhir adalah menghitung nilai TF-IDF dengan cara mengalikan nilai TF dengan nilai IDF. Hasil dari pembobotan tersebut akan diurutkan dari token dengan bobot terbesar ke terkecil. Selanjutnya term-term tersebut digunakan untuk menguji similaritas antar dokumen testing sesuai skenario yang telah ditentukan.

2.4 Algoritma Rabbin-Karp

Rabin-Karp merupakan salah satu algoritma pencocokan string (*string matching*) dengan input sebagai pembanding antar string yang dicari (m) dan substring pada teks (n). Apabila nilai *hash* keduanya sama jadi akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Dan jika hasil ke dua-duanya tidak sama maka substring bergeser ke kanan. Pergeseran dilakukan sebanyak $(n - m)$ kali. Efisiensi nilai hash yang dihitung akan mempengaruhi performa Rabin-Karp [10]. Pada metode ini memilik 2 tahapan utama yaitu $K-$Gram dan *rolling hash*. $K-$Gram merupakan deret suku dengan panjang k. $K-$Gram digunakan untuk mengekstrak huruf karakter dari sejumlah k dari sebuah kata yang terus dibaca dari teks sumber sampai akhir dokumen [11]. Berikut ini adalah contoh penerapan $K-$Gram dengan nilai $k = 3$ dalam kalimat bahasa Jawa:

1. bahasa Jawa : “jeneng yahoo diusulake supaya digenti dadi altaba inc”,
2. hapus semua spasi : “jenengyahoodiusulakesupayadigentidadialtabainc”,
3. dihasilkan rangkaian K-Gram 3 yang di turunkan dari teks sebagai berikut : “jene ngy aho odi usu lak esu pay adi gen tid adi alt aba inc”.

Rolling hash merupakan proses konversi setiap kata yang telah dibagi berdasarkan nilai $K-$Gram menjadi *hash* yang unik [12]. Berikut adalah persamaan untuk menghitung nilai *hash* untuk algoritma Rabin-Karp [13]. Nilai similaritas antara artikel berita satu terhadap artikel lainya dapat dihitung setelah di dapatkan jumlah substring pada artikel berita satu yang match atau cocok dengan substring pada artikel berita lainya. Mengelompokkan hasil *terms* dari $K-$Grams yang sama dengan tujuan untuk mencari nilai similaritas dari kedua artikel berita yang di bandingkan. Langkah berikutnya adalah melakukan perhitungan dengan *Dice's similarity coefficient* untuk untuk menghitung similaritas dari kumpulan kata tersebut.

3 Hasil dan pembahasan

Sumber data yang akan digunakan dalam penelitian ini adalah artikel berita Pawarta Bahasa Jawa yang terdiri dari 2 kategori. Kategori tersebut adalah kriminal dan teknologi. Tiap kategori akan diambil sebanyak 6 artikel artikel sehingga data artikel keseluruhan berjumlah 12 dengan variabel bebas sebanyak 2.952 kata. Dari artikel berita tersebut, dilakukan beberapa *preprocessing* yakni menghapus pungtuasi, tanda baca dan proses *case folding*.

Hasil dari proses *case folding* dapat dilihat pada Tabel 1.

Tabel 1 Hasil *case folding*

Artikel asli	hasil *case folding*
Sakwise Yahoo diakuisisi saka verizon, jeneng Yahoo diusulake supaya digenti dadi Altaba inc Usulan kesebut ana jero laporan sing diserahke Yahoo menyang komisi sekuritas lan saham AS, dina senin kepungkur....	sakwise yahoo diakuisisi saka verizon, jeneng yahoo diusulake supaya digenti dadi altaba inc usulan kesebut ana jero laporan sing diserahke yahoo menyang komisi sekuritas lan saham as, dina senin kepungkur....

Setelah dilakukan *case folding*, proses selanjutnya adalah tokenisasi, yaitu mengubah artikel teks menjadi bentuk token, seperti yang terlihat pada Tabel 2.

Tabel 2 Hasil tokenisasi

Artikel asli	hasil tokenisasi
Sakwise Yahoo diakuisisi saka verizon, jeneng Yahoo diusulake supaya digenti dadi Altaba inc Usulan kesebut ana jero laporan sing diserahke Yahoo menyang komisi sekuritas lan saham AS, dina senin kepungkur....	[‘sakwise’, ‘yahoo’, ‘diakuisisi’, ‘saka’, ‘verizon’, ‘jeneng’, ‘yahoo’, ‘diusulake’, ‘supaya’, ‘digenti’, ‘dadi’, ‘altaba’, ‘inc’, usulan’, ‘kesebut’, ‘ana’, ‘jero’, ‘laporan’, ‘sing’, ‘diserahke’, ‘yahoo’, ‘menyang’, ‘komisi’, ‘sekuritas’, ‘lan’, ‘saham’, ‘as’, ‘dina’, ‘senin’, ‘kepungkur’,....]

Proses selanjutnya adalah, menghapus beberapa kata stopword untuk bahasa jawa. Dari hasil tokenization, dihapuslah kata-kata imbuan (ater-ater), akhiran (panambang), sisipan (seselan) serta imbuhan dan akhiran. Beberapa contoh stopword bahasa jawa antara lain seperti : “apa”, “ana”, amarga”, “dadi”, “dudu”, “gawe”, “iki”, “iku”, “nanging”, “ora”, “lan”, “sing”, dan sebagainya. Dari hasil *stemming* menggunakan metode Metode Nazief-Adriani menunjukkan bahwa terdapat kata yang tidak ter-*steam*. Penyebab tidak terstemmingnya kata ini dikarenakan di dalam *database* tidak ditemukan kata dasarnya, sehingga kata tersebut dikembalikan ke bentuk semula. Metode untuk menambahkan kata dasar jika tidak ada kata yang belum tersteam dengan baik diperlukan sehingga *database* kamus dapat semakin lengkap.

Pada proses *stemming* menggunakan metode Levenshtein Distance input kata berbahasa Jawa akan dibandingkan dengan semua kata pada kamus kata. Tujuan dari proses ini untuk mendapatkan nilai *distance* atau jarak terpendek antar 2 kata yang dibandingkan. Selain mencari *distance* antar kata, juga dihitung nilai rasio perbandingan 2 kata tersebut. Jumlah kamus kata yang disediakan kurang lebih ada 3.202 kata. Dari hasil pengamatan pada saat dilakukan *stemming*, kandidat kata yang diberikan tidak bisa secara langsung digunakan akan tetapi perlu dilakukan pengecekan irisan kata dasar terhadap kata input yang diberikan. Hal ini disebabkan nilai *distance* yang dihasilkan tidak sama dengan 0 atau tingkat rasio tidak sama dengan 1.

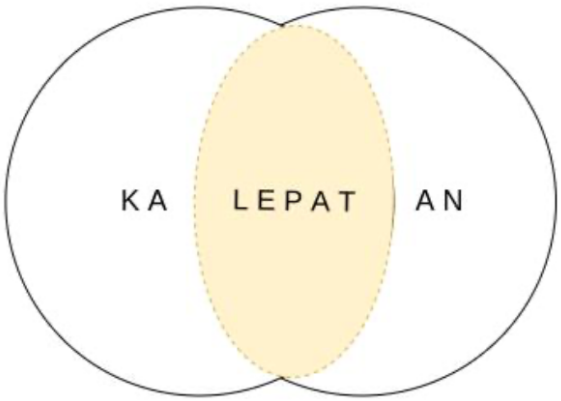
Nilai *distance* 0 atau rasio sama dengan 1, akan dihasilkan apabila input kata yang digunakan adalah kata dasar itu sendiri, serta nilai *distance* yang dihasilkan memiliki kemiripan pada kandidat kata dasar yang diberikan. Dengan demikian, penentuan kandidat kata dasarnya dilakukan dari tingkat rasio yang dihasilkan. Kandidat kata dasar yang diambil yang memiliki tingkat rasio $>= 0.7$.

Sebagai contoh inputan kata “kalepatan” apabila dilakukan pencarian ke dalam kamus kata, maka akan muncul beberapa kata yang memiliki tingkat rasio besar atau nilai distance kecil, hasil yang diberikan kandidat katanya seperti “kalepyan”, “karepan”, “kalan”, “lepat, “kapantha”. Tabel 3 menunjukan tingkat rasio dari kandidat kata yang dihasilkan.

Tabel 3 Tingkat rasio kandidat kata dasar

Input kata	Kandidat kata dasar	Tingkat rasio
kalepatan	kalepyan	0.8235
	karepan	0.75
	kalan	0.715
	lepat	0.714
	kapantha	0.705

Dari hasil perhitungan didapatkan bahwa kata yang memiliki tingkat rasio tertinggi belum tentu menjadikan kata tersebut sebagai kata dasar dari input kata. Merujuk pada nilai *distance* yang mendekati 0 adalah kata “kalepyan" yaitu 2, sedangkan kata “lepat" menghasilkan nilai *distance* 4. Penelitian ini menambahkan skenario pengecekan irisan kata dasar terhadap kata input yang diberikan untuk mengonfirmasi hasil yang diberikan. Sebagai contoh kata lepat dengan kalepatan. Irisan yang dihasilkan adalah kata lepat. Hasil irisan kata dapat dilihat pada Gambar 3.



Gambar 3 Hasil irisan kandidat kata dasar dengan input

Pada penelitian ini pembobotan kata dilakukan pada hasil kedua metode *stemming* yang digunakan, pembobotan kata pada metode Nazief-Adriani menghasilkan jumlah kata dasar yang telah memiliki bobot dengan jumlah 401 kata. Sedangakan pembobotan kata pada metode Levenshtein menghasilkan jumlah kata dasar yang telah memiliki bobot dengan jumlah 422 kata.

Tahap pengujian dengan tahapan *stemming* Nazief Adriani, levenshtein dan tanpa *stemming* dari uji coba 3 dokumen dari dataset, ditunjukan pada Tabel 4 berikut:

Dari hasil pengujian waktu dengan memberikan nilai $K-$Gram $4, 5, 6$ dan nilai bilangan basis prima $5, 11, 23$ menunjukkan bahwa pengujian dengan nilai $K-$Gram $4, 5, 6$ dengan bilangan basis prima$5, 11, 23$ hanya memiliki sedikit selisih perbedaan waktu. Faktor spesifikasi komputer dan penggunaan processor serta RAM mempengaruhi hasil pengujian. Pengujian dilakukan saat keadaan penggunaan CPU sebesar 11 % dari 2.6 GHz dan memori sebesar

Tabel 4 Skenario pengujian

Metode	Waktu (detik)		
	N-Gram 5 prima 11	N-Gram 4 Prima 5	N-Gram 6 Prima 23
Nazief non TFIDF	0.034	0.035	0.037
Nazief TFIDF	0.044	0.041	0.048
Levenshtein non TFIDF	242.021	243.081	245.072
Levenshtein TFIDF	245.021	246.011	243.033
Tanpa *stemming*	0.0075	0.0065	0.0085

43 % dari 8GB RAM. Perbedaan antara menggunakan TF-IDF dan tidak menggunakan TF-IDF memiliki selisih waktu yang tidak signifikan. Namun, pada metode *stemming* Nazief Adriani dan Levenshtein memiliki perbedaan waktu yang signifikan. Metode Nazief Adriani memiliki waktu eksekusi yang lebih cepat daripada Levenshtein.

Prosentase kemiripan dengan memberikan nilai $K-$Gram $4, 5, 6$ dan nilai bilangan prima $5, 11, 23$ memiliki selisih yang signifikan. Nilai $N-$Gram 4 dan Prima 5 memiliki nilai prosentase paling tinggi, baik pada metode Nazief Adriani, Levenshtein dan non *stemming*. Untuk nilai $N-$Gram 6 dan Prima 23 memiliki prosentase paling rendah. Dari ketiga uji coba dapat diambil kesimpulan $N-$Gram 5 dan Prima 11 adalah yang optimal [11] karena nilainya tidak terlalu rendah dan terlalu tinggi untuk hasil prosentasenya.

Waktu proses yang digunakan dengan memberikan nilai $K-$Gram 4,5,6 dan nilai bilangan prima 5,11,23 pada *stemming* Nazief Adriani dengan uji coba 3 kali, membutuhkan waktu rata-rata 0.101 detik. Sementara dengan menggunakan Levenshtein untuk uji coba 3 kali, dengan waktu rata-rata 244.039 detik, dan untuk proses tanpa melalui *stemming* 0.0075 detik. Dari uji coba menggunakan nilai $N-$Gram 5 dan prima 11 menggunakan *stemming* Nazief Adriani Non TF-IDF untuk artikel 1 & 2 menghasilkan similaritas sebesar 7.6%. Uji coba pada artikel 1 & 3 menghasilkan similaritas 7.7% dan uji coba artikel 2 & 3 menghasilkan similaritas sebesar 21.3% . Sedangkan untuk metode stemming Levenstein non TF-IDF artikel 1 & 2 menghasilkan similaritas sebesar 15.7% , artikel 1 & 3 menghasilkan similaritas sebesar 13.0% dan ujicoba artikel 2 & 3 menghasilkan similaritas sebesar 26.5%.

Uji coba menggunakan nilai $N-$Gram 5 dan prima 11 menggunakan *stemming* Nazief Adriani dengan TF-IDF untuk artikel 1 & 2 menghasilkan similaritas 0.6 %, artikel 1 & 3 menghasilkan similaritas 1.1 %, dan artikel 2 & 3 menghasilkan similaritas sebesar 4.2 %. sedangkan untuk metode *stemming* Levenstein dengan TF-IDF artikel 1 & 2 menghasilkan similaritas sebesar 1.5 %, artikel 1 & 3 menghasilkan similaritas 0.5 %, dan artikel 2 & 3 menghasilkan similaritas sebesar 4.5 %. Uji coba artikel non *stemming* artikel 1 & 2 menghasilkan 9.5%, artikel 1 & 3 menghasilkan similaritas sebesar 6.5% dan artikel 2 & 3 similaritas sebesar 6.9 %.

4 Kesimpulan dan saran

Kesimpulan dari penelitian ini adalah bahwa proses *stemming* mempengaruhi pada hasil nilai similaritas, jika tanpa *stemming* maka nilai similaritas lebih tinggi dibanding dengan proses dengan *stemming*. Hal tersebut disebabkan karena jika tanpa *stemming* dan proses text *preprocessing*, maka dokumen tersebut masih bentuk dokumen asli, yang di mana masih ada kata hubung, tanda baca, angka, sehingga yang dapat mempengaruhi tingkat keakuratan. Proses *stemming* menggunakan metode Levenshtein lebih akurat daripada metode Nazief Adriani dilihat dari 3 kali uji coba, menghasilkan hasil similaritas rata-rata

16,95% untuk metode Levenshtein dan 11,9% untuk metode Nazief Adriani. Namun, waktu proses pengecekan dokumen menggunakan metode *stemming* Nazief Adriani lebih cepat daripada menggunakan metode Levenshtein, 0.101 detik dibandingkan dengan 244.039 detik.

Jika diberikan nilai $K-$Gram semakin kecil maka tingkat kesamaan dokumen semakin besar dan sebaliknya, jika nilai $K-$Gram besar maka tingkat kesamaan dokumen semakin kecil. Hasil yang paling baik yang didapatkan dalam penelitian ini adalah dengan menentukan nilai $K-$Gram dan basis bilangan prima yang tidak terlalu kecil dan tidak terlalu besar. Pada pengujian ini paling baik adalah dengan nilai $K-$Gram 5 dan bilangan basis prima 11, serta melewati tahapan *stemming*. Penelitian ini hanya menghitung prosentase similaritas antara artikel berita, apabila artikel tersebut memiliki kesamaan, maka dapat dikelompokkan dalam kategori yang sama. Metode TF-IDF lebih efisien daripada tanpa TF-IDF karena jumlah kata / *term* yang sering muncul pada dokumen akan dihitung dan di ranking sesuai tingkat bobot katanya, dan menghilangkan duplikasi token yang dihasilkan. Metode Levenshtein dengan TF-IDF dengan 3 kali dalam pengujian menghasilkan nilai rata- rata similaritas didapatkan sebesar 7.7% dan membutuhkan waktu rata- rata 244.688 detik. Sedangkan metode Nazief Adriani dengan TF-IDF dengan 3 kali pengujian nilai rata-rata similaritas sebesar 6,8% dan membutuhkan waktu rata-rata 0.0443 detik.

Dari hasil rata-rata eksekusi antara kedua metode, metode Levenshtein lebih akurat sebesar 0.9% dari Nazief Adriani. Waktu eksekusi metode Nazief Adriani lebih cepat daripada metode levenshtein dengan selisih 244.644 detik. Tampak bahwa dalam penelitian ini metode algoritma Nazief Adriani lebih baik daripada Levenshtein. Metode Nazief Adriani dapat digunakan untuk proses *stemming* bahasa jawa karena tingkat akurasi yang baik dan waktu eksekusi yang cepat. Penelitian selanjutnya dapat dilakukan dengan menambah jumlah kata dasar dalam kamus, supaya memperbanyak *stemming* kata yang dihasilkan dan dapat melakukan uji coba dengan berbagai dataset serta dapat dicoba membandingkan dengan metode *stemming* lain.

Pustaka

1 A. D. Hartanto, A. Syaputra, dan Y. Pristyanto, "Best parameter selection of rabin-karp algorithm in detecting document similarity," in *2019 International Conference on Information and Communications Technology (ICOIACT)*. IEEE, 2019, pp. 457–461.

2 M. Fauziyah, "Stemming bahasa jawa menggunakan algoritma levenshtein dan analisa morfologi," 2019. [Online]. Available: http://etheses.uin-malang.ac.id/16387/1/12650132.pdf

3 A. P. Wibawa dan M. N. Hakim, "Stemming bahas jawa menggunakan damerau levenshtein distance (dld)," *Jurnal Teknik Informatika*, vol. 14, no. 1, pp. 22–27, 2021.

4 A. P. Wibawa, F. A. Dwiyanto, I. Zaeni, R. Nurrohman, dan A. Afandi, "Stemming javanese affix words using nazief and adriani modifications," *J. Inform*, vol. 14, no. 1, p. 36, 2020.

5 "Pawarta bahasa jawa," 2021, diakses 29 Oktober 2021. [Online]. Available: https://pawartabahasajawa.blogspot.com

6 H. Maarif, R. Akmeliawati, Z. Htike, dan T. S. Gunawan, "Complexity algorithm analysis for edit distance," in *2014 International Conference on Computer and Communication Engineering*. IEEE, 2014, pp. 135–137.

7 S. Jakšić, E. Bartocci, R. Grosu, T. Nguyen, dan D. Ničković, "Quantitative monitoring of stl with edit distance," *Formal methods in system design*, vol. 53, no. 1, pp. 83–112, 2018.

8 M. E. Sulistyo, R. Saptono, dan A. Asshidiq, "Penilaian ujian bertipe essay menggunakan metode text similarity," *Telematika: Jurnal Informatika dan Teknologi Informasi*, vol. 12, no. 2, pp. 146–158, 2015.

9 S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, 2004.

10 A. H. Purba dan Z. Situmorang, "Analisis perbandingan algoritma rabin-karp dan levenshtein distance dalam menghitung kemiripan teks," *Jurnal Teknik Informatika UNIKA Santo Thomas*, vol. 2, no. 2, pp. 24–32, 2017.

11 A. P. U. Siahaan, M. Mesran, R. Rahim, dan D. Siregar, "K-gram as a determinant of plagiarism level in rabin-karp algorithm," *International Journal of Scientific & Technology Research*, vol. 6, no. 7, 2017.

12 A. H. Lubis, A. Ikhwan, dan P. L. E. Kan, "Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level," *International Journal of Engineering & Technology*, vol. 7, no. 2.27, pp. 17–21, 2018.

13 U. M. Perlis, "Rabin-karp elaboration in comparing pattern based on hash data," *International Journal of Security and Its Applications*, vol. 12, no. 2, pp. 59–66, 2018.