Sebatik Vol. 25 No. 1 Juni 2021
Salim, R.A., Septian, M.R.D, Suhartini, Anggraini, D. and Qomariyah. (2021) "Aplikasi Pendeteksi Kesamaan Dokumen dengan Menggunakan Algoritma Jarak Jaro Winkler dan Levenshtein", Sebatik, 25(1).
ISSN: 1410-3737(p) 2621-069X(e)

Submitted: 2021/04/24 Accepted: 2021/05/26 Published: 2021/06/01
DOI:10.46984/sebatik.v25i1.1309

APLIKASI PENDETEKSI KESAMAAN DOKUMEN DENGAN MENGGUNAKAN ALGORITMA JARAK *JARO WINKLER* DAN *LEVENSHTEIN*

Ravi Ahmad Salim[1)], M Ridwan Dwi Septian[2)], Suhartini[3)], Dyah Anggraini[4)], dan Qomariyah[5)]

[1,2]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Gunadarma
[3]Sistem Komputer, Fakultas Ilmu Komputer, Universitas Gunadarma
[4,5]Sistem Informasi, Fakultas Ilmu Komputer, Universitas Gunadarma
[1,2,3,4,5] Jl. Margonda Raya No.100, Pondok Cina, Depok
E-mail : ravi@staff.gunadarma.ac.id[1)], ridwandwiseptian@staff.gunadarma.ac.id[2)], shartini@staff.gunadarma.ac.id[3)], d_anggraini@staff.gunadarma.ac.id[4)], qomariyah@staff.gunadarma.ac.id[5)]

ABSTRAK

Peranan teknologi di era revolusi 4.0 membuat penggunaan teknologi informasi dan komunikasi semakin luas. Peranan ini juga mencakup segala bidang, salah satunya untuk mempermudah pengguna dalam mendapatkan informasi yang tersedia secara bebas dan tanpa biaya. Akan tetapi hal ini memungkinkan terjadinya pengambilan informasi tertulis (karya tulis, penulisan ilmiah, dokumen dan lain sebagainya) tanpa mencantumkan referensi (menjiplak) yang biasa dikenal sebagai tindakan *plagiarism*. Teknologi informasi pengukur tingkat plagiat suatu dokumen teks berhubungan dengan pencarian informasi dari data yang ada. Dokumen teks merupakan sesuatu yang tertulis atau tercetak yang dapat digunakan sebagai keterangan. Untuk membuat suatu informasi tertentu dibutuhkan waktu yang lama untuk memproses hasil kemiripan dari seluruh isi dokumen teks. Pemrosesan *text mining* menggunakan beberapa algoritma. Salah satunya adalah algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein* . Jarak *Jaro Winkler* adalah sebuah algoritma menghitung panjang kata dalam dokumen, kata yang sama, dan jumlah transposisi. Sedangkan algoritma Jarak *Levenshtein* digunakan untuk menghitung jarak yang dibutuhkan untuk mengubah satu kata menjadi kata lain. Untuk itu dibuat aplikasi pendeteksi untuk melihat kemiripan dokumen teks dengan menerapkan Algoritma jarak *Jaro Winkler* dan jarak *Levenshtein*. Kedua algoritma ini diimplementasikan dan menampilkan hasil dari perhitungan antara *Jaro Winkler* dan jarak *Levenshtein* dalam sebuah aplikasi. Dengan melihat perbandingan kedua algoritma ini pengguna dapat mengetahui algoritma mana yang akan menghasilkan keluaran yang lebih baik jika dibandingkan dengan pencarian kemiripan secara manual.

Kata Kunci: *Jarak, Dokumen, Kesamaan, Jaro Winkler, Levenshtein*

1. PENDAHULUAN

Dokumen merupakan sesuatu yang tertulis atau tercetak yang dapat digunakan sebagai keterangan juga untuk memuat suatu informasi. Salah satu contoh dari dokumen adalah karya tulis yang saat ini dapat ditemukan dalam bentuk digital. Setiap karya tulis yang dituliskan oleh seorang penulis atau peneliti memiliki hak cipta atas setiap tulisannya. Banyak orang yang mencari informasi untuk kebutuhan tugas, penelitian, maupun karya ilmiah dari karya tulis yang telah dipublikasikan tersebut. Karya tulis yang telah dipublikasi dapat diakses dengan bebas dan tanpa biaya (gratis). Hal ini menyebabkan beberapa pengguna atau peneliti yang tidak bertanggung jawab mengambil isi dari karya tulis tersebut secara bebas tanpa memikirkan hak cipta dari penulis yang bersangkutan (Kambey, 2020).

Tindakan mencuri karya tulis atau pikiran maupun gagasan penulis lain disebut dengan tindakan *plagiarism*. Sesuatu disebut plagiat ketika ide atau karya tersebut dibuat ulang dan diakui sebagai ide atau karya orang lain. Selain merugikan pihak penulis yang memiliki pemikiran dalam karya tulis ini, tindakan *plagiarism* juga sangat merugikan banyak pihak (Silvana, 2018).

Kegiatan menjiplak atau mengambil hasil karangan, pendapat, ide dan karya orang lain dan menjadikannya seolah hal tersebut merupakan karangan, pendapat, ide dan karya sendiri merupakan suatu bentuk *plagiarism* atau plagiat (Awasthi, 2019). *Plagiarism* atau plagiat merupakan tindakan pidana karena hal ini merupakan sebuah pencurian, yaitu mencuri hak cipta orang lain. Di dunia Pendidikan, pelaku *plagiarism* akan mendapatkan hukuman berat seperti dikeluarkan dari sekolah atau universitas. Pelaku plagiat disebut sebagai plagiator. Menurut Hermawan plagiat merupakan suatu pencurian karangan milik orang lain (Hermawan, 2019).

Selain itu, plagiat juga dapat diartikan sebagai pengambilan karangan (pendapat dan sebagainya) milik orang lain yang kemudian dijadikan seolah olah menjadi milik pribadi. Setiap karangan yang asli dianggap sebagai hak milik seorang pengarang dan orang lain tidak diperkenankan untuk mencetak ulang tanpa seizin yang mempunyai hak atas karangan tersebut (Farhat, 2019).

Sebelum proses pengecekan *plagiarism*, *text* pada sebuah kalimat akan diolah dengan menggunakan *text mining*. *Text Mining* merupakan suatu proses menemukan hal baru, yang sebelumnya tidak diketahui mengenai informasi yang berpotensi untuk diambil manfaatnya dari sumber data yang tidak terstruktur mencakup dokumen bisnis, komentar *customer*, halaman *web* dan *file* XML (Purnomo, 2017). Tujuan dan proses *text mining* hampir sama dengan *data mining*. Perbedaan dari *text* dan *data mining* terletak pada *file input*-nya. *Input file* pada *text mining* berupa *file* data yang tidak terstruktur seperti dokumen dalam bentuk word, PDF, XML, dan sebagainya (Saputra, 2019).

Pemrosesan *text mining* menggunakan beberapa algoritma. Salah satunya adalah Algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein*. Jarak *Jaro Winkler* merupakan sebuah algoritma yang digunakan untuk menghitung panjang kata dalam suatu dokumen. Selain itu algoritma ini juga digunakan untuk menghitung banyaknya kata yang sama, dan jumlah transposisi. Sedangkan Algoritma Jarak *Levenshtein* digunakan untuk menghitung jarak yang dibutuhkan untuk mengubah satu kata menjadi kata lain (Novantara, 2018).

Penelitian ini bertujuan untuk mendeteksi kesamaan dokumen dalam bentuk publikasi jurnal dan membandingkan hasil kedua metode yang dipakai yaitu, Jarak *Jaro Winkler* dan Jarak *Levenshtein*. Kontribusi dalam penelitian ini agar dapat melakukan pembanding dengan Algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein*.

2. RUANG LINGKUP

Penelitian ini mempunyai ruang lingkup untuk membandingkan hasil kedua algoritma yang digunakan yaitu, Algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein*. Adapun tahapan sebelum menentukan perbandingan dari kedua algoritma tersebut yaitu :

1. Tahapan Pengumpulan Data
   Pada tahapan ini peneliti mengumpulkan 500 data jurnal ter publikasi yang di ambil dari *repository* Universitas Gunadarma dengan alamat *url* https://ejournal.gunadarma.ac.id. Kriteria data yang diambil terdiri dari *field* Penulis, Judul, Abstrak, Kata Kunci, Tahun, Kategori, dan Terbitan. Data tersebut akan disimpan ke dalam basis data.
2. Perhitungan Kesamaan *Text*
   Pada proses perhitungan kesamaan *text*, data *input*-an untuk Judul dan Abstrak dicocokkan ke dalam basis data yang telah dibuat dengan menggunakan Algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein*.

3. BAHAN DAN METODE

Bagian ini menjelaskan bahan dan metode apa yang akan digunakan dalam penelitian ini, di antaranya :

3.1 Dokumen

Menurut Kamus Besar Bahasa Indonesia (KBBI) menyebutkan dokumen adalah sesuatu yang tertulis atau tercetak yang dapat dipergunakan sebagai bukti atau keterangan. Bagi suatu instansi, organisasi, atau negara, dokumen merupakan salah satu hal yang sangat penting. Hal ini karena dokumen merupakan sumber informasi yang diperlukan. Tanpa dokumen, pengguna akan kehilangan data yang diperlukan untuk kegiatan kantor atau organisasi (Ade, 2019).

3.2 Bahasa Baku

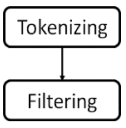
Bahasa terbagi menjadi bahasa baku dan tidak baku. Penggunaan bahasa ini berkaitan dengan situasi dan kondisi pemakaiannya. Ragam bahasa baku biasanya digunakan dalam situasi resmi. Adapun ragam bahasa tidak baku pada umumnya digunakan dalam komunikasi sehari-hari yang memiliki sifat tidak resmi (Frando, 2019).

Di dalam penggunaan Bahasa Indonesia yang baik dan benar terdapat suatu aturan atau kaidah standar (bahasa baku). Bahasa baku merupakan ragam bahasa dimana mengucapkan dan menuliskan kata harus sesuai dengan kaidah-kaidah standar yang berlaku. Kaidah standar dapat berupa pedoman Ejaan Yang Disempurnakan (EYD), tata bahasa baku, dan Kamus Besar Bahasa Indonesia (KBBI). Bahasa baku biasa dipergunakan pada penulisan penelitian, kegiatan seminar, pidato, temu karya ilmiah, dan lain-lain.

Sebaliknya, bahasa tidak baku adalah ragam bahasa yang cara pengucapan dan penulisannya tidak memenuhi kaidah standar tersebut.

3.3 *Pra pemrosesan*

Text pra-pemrosesan merupakan tahap awal dari pengolahan teks. Tahap ini mencakup semua rutinitas, dan proses untuk mempersiapkan data yang akan digunakan pada operasi *knowledge discovery* sistem pengolahan teks (Rustiana, 2017). *Text* pra-pemrosesan pada penelitian ini memiliki tujuan untuk memecah teks menjadi *array* kata. Setiap kata pada *array* tersebut akan diperiksa berdasarkan Kamus Besar Bahasa Indonesia (KBBI). Tahapan dari *text* pra-pemrosesan yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Tahapan *Text Pra pemrosesan*

1. *Tokenizing*

Mendeteksi kata kunci atau *token* dan batas kalimat adalah langkah pra-pemrosesan. Langkah ini sangat penting dalam aplikasi untuk memproses bahasa alami. Sebagian besar *kata kunci* dan batas kalimat beroperasi

baik pada tingkat kata (misalnya, silabus, analisis morfologi) atau kalimat (misalnya pemberian penanda bagian-pidato, penguraian kalimat, kalimat hasil terjemahan mesin). Beberapa studi tentang pemrosesan bahasa mencakup proses *tokenizing* pada tahap pra-pemrosesan. Penelitian yang di lakukan Omar (Omar, 2018) melakukan *tokenizing* pada tahap pra-pemrosesan untuk memperoleh kata kunci bahasa Arab. Noaman pada tahun 2018 menyajikan model bahasa jaringan saraf berdasarkan tokenisasi kata-kata. Tokenisasi kata-kata tersebut menjadi tiga bagian: bagian awalan, bagian batang, dan bagian akhiran. Model pada penelitian ini akan diuji dengan menggunakan *dataset* pengenal ucapan AMI bahasa Inggris serta melebihi model *n-gram* awal (Noaman, 2018)

2. *Case Folding*

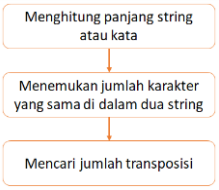
Pada tahap ini hasil dari tokenisasi diubah menjadi huruf kecil atau tidak huruf besar semua. Hal ini karena proses Jarak *Jaro Winkler* bersifat *case sensitive* sehingga besar kecilnya huruf dapat mempengaruhi hasil perhitungan jarak kedekatan. Contoh proses pengubahan semua huruf dari kata kunci yang sudah didapatkan dari proses *tokenizing*, yaitu kata "Tidur" berubah menjadi kata "tidur", sedangkan kata "larut" dan "malam" tidak ada perubahan. Tidak adanya perubahan ini dikarenakan semua kata sudah dalam bentuk huruf kecil (Spelling, 2018).

3. *Stemming*

Rahardian (2017) pada penelitiannya mengatakan bahwa proses *indexing* dilakukan menggunakan data berupa teks dengan tujuan untuk membuat pengelompokan pada judul majalah elektronik. Masing-masing kata kunci yang terbentuk diproses kembali untuk membentuk kata dasar. Pada tahap *indexing* semua jenis imbuhan kata, tanda baca "/", "//", dan penanda *html* dihapus selama proses penghitungan Jarak *Jaro Winkler* berlangsung. Tujuan mendapatkan *root word* adalah untuk mencegah kesalahan perhitungan ketika melakukan ekstraksi fitur sintaksis (Prayogo, 2018).

3.4 Jarak *Jaro Winkler*

Tahapan selanjutnya setelah data siap diproses adalah mengecek pengejaan setiap kata kunci berdasarkan daftar kata pada basis data. Jika ditemukan kecocokan maka pengejaan kata tersebut dianggap benar dan tidak perlu melalui tahap penghitungan jarak. Namun jika tidak ditemukan kecocokan pada basis data maka sistem akan menghitung jarak kedekatan dengan daftar kata yang berindikasi mempunyai kemiripan berdasarkan jumlah karakter atau huruf kata kunci tersebut. Jarak *Jaro Winkler* mempunyai 3 komponen dasar seperti gambar 2 (Spellingm, 2018).



Gambar 2. Komponen algoritma Jarak *Jaro Winkler*

Pada Algoritma Jarak *Jaro Winkler* (Gambar 2 di atas), jarak antara dua kata k1 dan k2 dihitung menggunakan rumus (1) (Spellingm, 2018).

$$Jaro(k1,k2) = \frac{1}{3}\left(\frac{m}{|k1|} + \frac{m}{|k2|} + \frac{m-t}{m}\right) \qquad (1)$$

Untuk k1 dan k2 adalah dua teks atau kata yang dihitung jarak atau kedekatan. Sedangkan m adalah jumlah karakter yang cocok antara k1 dengan k2, sementara t adalah jumlah transposisi. Jumlah dari karakter pencocokan yang memiliki urutan berbeda akan dibagi dua menjadi nilai transposisi. Karakter k1 dan karakter k2 akan memiliki kecocokan jika nilai yang dihasilkan tidak lebih jauh dari -1 pada rumus (2).

$$\left(\frac{\max(|k1|,|k2|)}{2}\right) < -1 \qquad (2)$$

Kedua kata dikatakan mempunyai kemiripan apabila Nilai Jaro yang dihasilkan bernilai tinggi. Nilai Jaro 0 menunjukkan bahwa kata tidak sama. Sedangkan nilai 1 pada Jaro menunjukkan bahwa keduanya sama (Febrianti, 2018).

Untuk membandingkan karakter k1 dan karakter k2 digunakan *prefix scale* (p)yang dapat memberikan tingkat penilaian yang lebih sedangkan untuk menyatakan panjang awalan yakni panjang karakter yang sama dari karakter yang dibandingkan sampai ditemukan ketidaksamaan digunakan *prefix length*. Jarak *Jaro Winkler* (dw) yang dihasilkan dari perbandingan karakter k1 dan karakter k2 pada rumus (3).

$$dw(k1,k2) = jaro(k1,k2) + (L * p(jaro(k1,k2))) \qquad (3)$$

Di mana Jaro merupakan Jarak *Jaro Winkler* untuk karakter k1 dan k2, t merupakan panjang prefiks umum di awal karakter yang nilai maksimalnya adalah 4 karakter, sedangkan p merupakan konstanta *scaling* faktor dengan nilai standar menurut Winkler adalah 0,1 (Spellingm, 2018).

Sebagai contoh seorang pengguna bermaksud menulis "tidur" tetapi yang tertulis "tydur". Setelah dilakukan pencarian tidak ditemukan kata "tidyr" pada daftar kata di basis data. kata "tidyr" akan dihitung jarak kedekatannya dengan semua kata yang jumlah karakternya antara empat sampai dengan enam karakter yang terdaftar di basis data menggunakan algoritma Jarak *Jaro Winkler*. Contoh: "tiban", "tidak", "tidur", "tifa", dan "tifus" (Spellingm, 2018)

3.5 *Jarak Levenshtein*

Pada tahun 1965 Jarak *Levenshtein* dibuat oleh Vladimir Levenshtein. Dengan menggunakan matriks untuk menghitung jumlah perbedaan karakter antara dua karakter akan didapat hasil perhitungan jarak *edit*. Untuk membuat karakter A menjadi karakter B didapat dari jumlah minimum operasi perubahan pada perhitungan jarak antara dua karakter. Ada 3 macam operasi utama pada Algoritma Jarak *Lavenshtein*. Operasi ini yaitu (Purba, 2017):

1. Operasi Pengubahan Karakter
   Operasi yang dilakukan dengan cara menukar sebuah karakter dengan karakter baru yang bernilai benar contoh penulisan karakter "yamg" akan diubah menjadi "yang" merupakan sebuah operasi pengubahan karakter. Dalam kasus ini karakter "m" diganti dengan huruf "n".
2. Operasi Penambahan Karakter
   Sebuah karakter yang ditambahkan kedalam suatu karakter , contoh karakter "kepad" menjadi karakter "kepada" dinamakan dengan operasi penambahan karakter. Terlihat bahwa dilakukan penambahan karakter "a" di akhir karakter. Penambahan karakter bisa dilakukan pada tiga bagian tempat yaitu akhir, awal maupun disisipkan di tengah karakter.
3. Operasi Penghapusan Karakter
   Apabila pada sebuah karakter ditemukan karakter berlebih maka akan dilakukan operasi penghapusan karakter. Contohnya karakter "barur" karakter terakhirnya dihilangkan sehingga menjadi karakter "baru".

Perhitungan matriks pada Algoritma Jarak *Levenshtein* dimulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter; karakter awal dan karakter target dan diberikan nilai *cost*. Pada ujung kanan bawah terdapat nilai *cost* yang menjadi nilai jarak *edit*. Gambar 3 menggambarkan jumlah perbedaan antara 2 karakter.

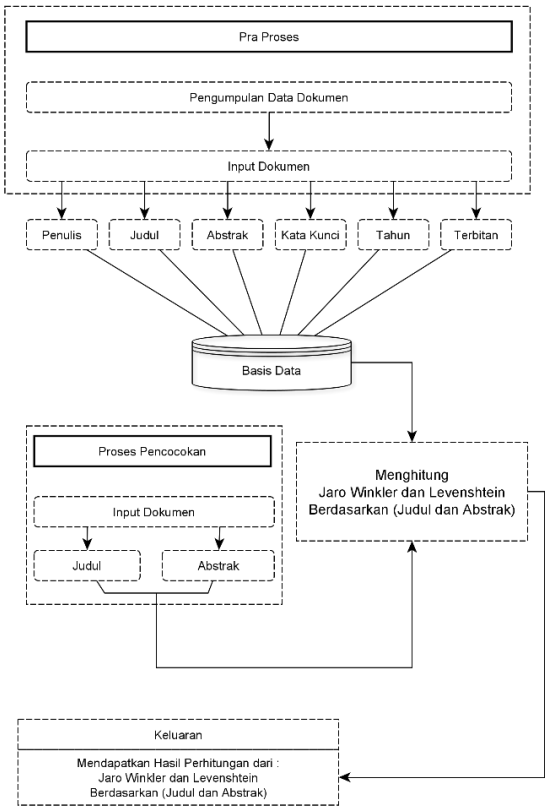
		s	a	y	a
	0	1	2	3	4
s	1	0	1	2	3
y	2	1	2	1	2
a	3	2	1	2	1

Gambar 3. Matriks Perhitungan

Contoh dari perhitungan Jarak *Levenshtein* menggunakan 2 karakter yang berbeda kemudian dihitung jarak *edit*-nya. Hasil dari perhitungan jarak *edit* antara 2 karakter "sya" dan "saya" adalah 1. Pengecekan dimulai dari iterasi awal pada kedua karakter kemudian dilakukan operasi untuk penambahan, operasi penyisipan dan operasi penghapusan. Nilai pada jarak *edit* terdapat pada ujung kanan bawah matriks tersebut. Terdapat satu proses penyisipan karakter "a" pada karakter "sya" sehingga menjadi "saya". Pada kasus pengecekan ejaan proses perhitungan ini dilakukan terhadap sejumlah kata yang ada pada basis data.

4. PEMBAHASAN

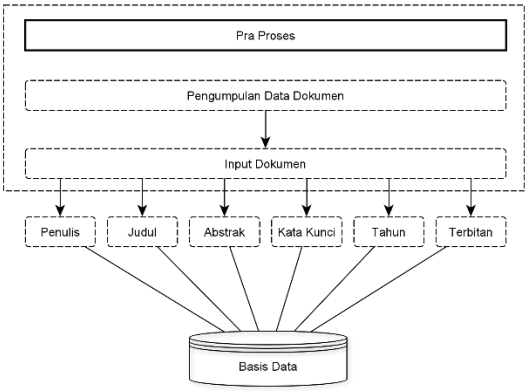
Pada bagian gambar 4 ini membahas langkah-langkah dalam implementasi untuk membandingkan Algoritma Jarak *Jaro Winkler* dan Jarak *Levenshtein*. Di bawah ini adalah rancangan atau bagan yang dibangun yaitu pra-proses dan proses pencocokan.



Gambar 4. Alur Proses Implementasi

4.1 Pra Proses

Bagian pra-proses merupakan tahapan penyimpanan data ke dalam basis data dengan *field* Penulis, Judul, Abstrak, Kata Kunci, Tahun, Kategori, dan Terbitan. Gambar 5 adalah proses penyimpanan data ke dalam basis data.



Gambar 5. Alur Pra Proses

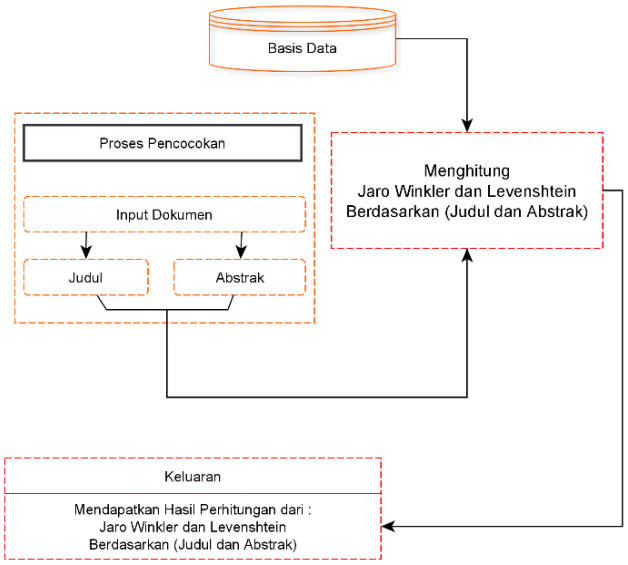
Berikut ini adalah struktur basis data yang digunakan, seperti yang terlihat pada Tabel 1 :

Tabel 1. Struktur Basis Data

Field Name	Data Type
ID	Auto number
PENULIS	Long Text
JUDUL	Long Text
ABSTRAK	Long Text
KATA_KUNCI	Long Text
TAHUN	Number
KATEGORI	Long Text
TERBITAN	Long Text

4.2 Proses Pencocokan dan Keluaran

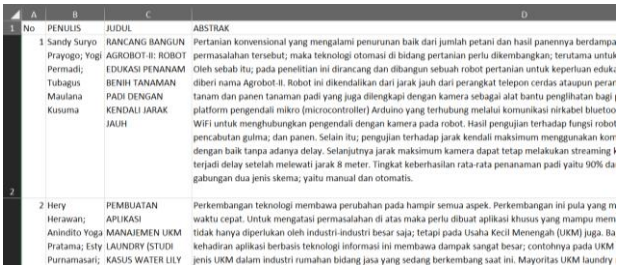
Pada proses pencocokan tahapan yang dilakukan adalah mencari kemiripan antara Algoritma jarak *Jaro Winkler* dan jarak *Levenshtein*. Proses ini diawali dengan memasukkan data dokumen yang akan dicocokkan. Proses pencarian kemiripan berdasarkan pada judul dan abstrak akan dilakukan di dalam basis data dengan beberapa parameter. Parameter tersebut berupa judul dan abstrak. Gambar 6 adalah proses pencocokan yang dilakukan. Setelah menghitung jarak *Jaro Winkler* dan jarak *Levenshtein* antara data *input* terdapat basis data maka menghasilkan sebuah keluaran dari Jarak *Jaro Winkler* dan Jarak *Levenshtein.*



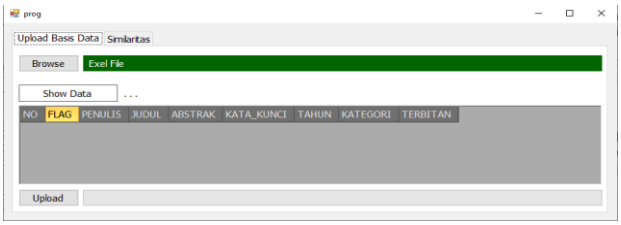
Gambar 6. Alur Pencocokan

4.3 Implementasi Aplikasi

Berikut ini adalah hasil implementasi yang diterapkan dalam bentuk GUI (*Graphic User Interface*). Hasil implementasi ini terdiri dari empat proses. Proses pertama adalah proses mengunggah data yang telah dikumpulkan sebelumnya dalam bentuk *microsoft excel* pada gambar 7, yang kemudian data tersebut diunggah ke dalam bentuk basis data, seperti yang terlihat pada gambar 8 yang berfungsi untuk mengunggah basis data yang telah di kumpulkan.



Gambar 7. Data Excel

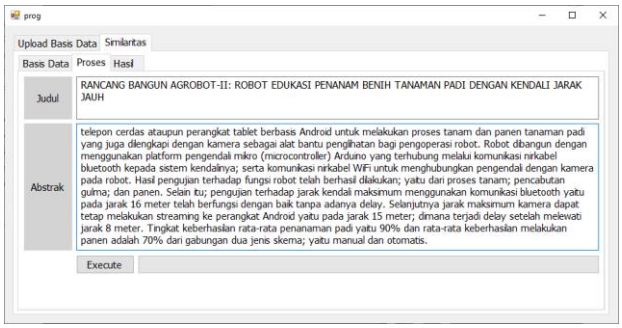


Gambar 8. Proses Upload Data Excel ke Basis Data

Setelah data yang dikumpulkan diunggah dari *microsoft excel*, proses kedua adalah menampilkan data dalam bentuk sebuah *grid* atau penggambaran tabel. Gambar 9 adalah bentuk implementasi dari aplikasi tersebut yang dijelaskan pada gambar 5 sebelumnya.

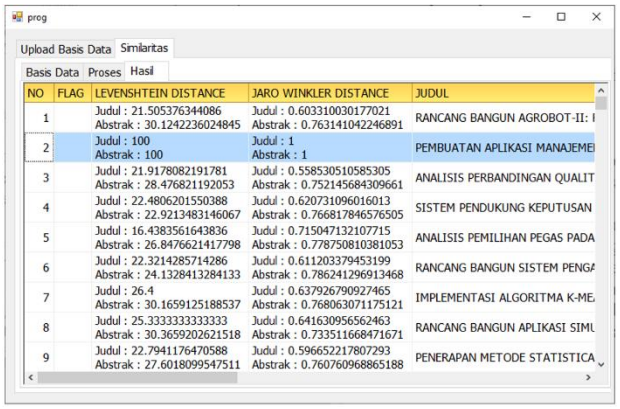
Gambar 9. Basis Data

Dilanjutkan dengan proses pencocokan yaitu proses pencarian kemiripan dengan memasukan data berupa judul dan abstrak pada aplikasi, seperti yang ditunjukkan pada gambar 10.



Gambar 10. Proses Input Data

Proses terakhir adalah menunjukkan gambaran hasil kemiripan dari proses *input* data yang di visualisasi pada gambar 11 dengan menggunakan alur pencocokan seperti gambar 6 sebelumnya.



Gambar 11. Hasil Kemiripan

Pada gambar 11 terlihat dari hasil perbandingan antara jarak *Jaro Winkler* dan jarak *Levenshtein* yang telah dicocokkan dengan basis data, terdapat data nomor 2 memiliki tingkat kesamaan persis dengan penjabaran *Jaro Winkler* bahwa semakin mendekati angka 100 maka semakin mirip pula data dokumen tersebut, untuk *Levenshtein* kemiripan nya semakin mendekati angka 1 maka semakin mirip pula data dokumen tersebut.

5. KESIMPULAN

Jarak *Jaro Winkler* sebuah algoritma menghitung panjang kata dalam dokumen, kata yang sama, dan jumlah transposisi. Sedangkan Algoritma Jarak *Levenshtein* menghitung jarak yang dibutuhkan untuk mengubah satu kata menjadi kata lain.

Berdasarkan pada analisis hasil pengujian aplikasi terhadap program Implementasi untuk mencari kesamaan dokumen dengan membandingkan algoritma Jaro Winkler dan Levenshtein, dapat diambil kesimpulan bahwa dengan algoritma sesuai dengan harapan penulis dengan mendapatkan hasil kesamaan pada data *input* terhadap data yang terdapat di dalam basis data.

Diharapkan pengguna dapat melihat perbandingan antara Algoritma jarak *Jaro Winkler* dan jarak *Levenshtein* tanpa harus menghitung kemiripan secara manual untuk melihat perbandingan tersebut.

Uji coba aplikasi ini menggunakan 500 basis data dokumen. Dokumen ini diambil dengan beberapa atribut diantaranya Penulis, Judul, Abstrak, Kata Kunci, Tahun, dan Terbitan. Aplikasi ini dibuat dengan bahasa pemrograman C# dengan basis *desktop* dan berjalan di sistem operasi *windows*.

6. SARAN

Penelitian ini masih dapat dikembangkan dengan menambahkan jumlah basis data yang digunakan dan memperluas ruang lingkup penelitian selain penggunaan *repository* pada Universitas Gunadarma. Sehingga pengguna dapat melihat dokumen yang akan dicocokkan tersebut memiliki kesamaan dokumen dengan universitas lainnya. Selain itu untuk ke depannya masih dimungkinkan penambahan algoritma yang lain seperti *Long Command Substring* yang dapat mendeteksi kata per kata.

7. DAFTAR PUSTAKA

Ade, R. (2019). Penerapan Metode Word2vec Untuk Mendeteksi Kemiripan Dokumen (Doctoral dissertation, Institut Telkom Purwokerto).

Awasthi, S. 2019. Plagiarism and Academic Misconduct: A Systematic Review. DESIDOC Journal of Library & Information Technology, 39(2).

Farhat, L. 2019. Upaya Pencegahan Tindakan Plagiarisme untuk Meningkatkan Kualitas Penulisan Karya Tulis Ilmiah di Dalam Pembimbingan Tugas Akhir (Skripsi) Bagi Mahasiswa STIE Jambi. J-MAS (Jurnal Manajemen dan Sains), 4(2), 326-333.

Febiawan, M. H., Setiawan, A., & Primadewi, A. 2019. Sistem Pendeteksi Dini Plagiarisme Menggunakan Algoritma *Jarak Levenshtein*. Jurnal Komtika (Komputasi dan Informatika), 3(1), 18-27.

Febrianti, Y. M., & Indriati, A. W. W. 2018. Analisis Sentimen Pada Ulasan "Lazada" Berbahasa Indonesia Menggunakan K-Nearest Neighbor (K-

NN) Dengan Perbaikan Kata Menggunakan *Jaro Winkler Distance*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN, 2548, 964X.

Frando, J., Ruslianto, I., & Hidayati, R. 2019. Penerapan *Jarak Jaro Winkler*dalam Aplikasi Pengoreksi Kesalahan Penulisan Bahasa Indonesia Berbasis Web. Coding Jurnal Komputer dan Aplikasi, 7(03).

Hermawan, A. 2019. Kebijakan Dosen Mengurangi Plagiarisme pada Karya Ilmiah Mahasiswa. IJIP: Indonesian Journal of Islamic Psychology, 1(2), 264-284.

Kambey, G. E. I., Sengkey, R., & Jacobus, A. 2020. Penerapan Clustering pada Aplikasi Pendeteksi Kemiripan Dokumen Teks Bahasa Indonesia. Jurnal Teknik Informatika, 15(2), 75-82.

Noaman, H. M., Sarhan, S. S., & Rashwan, M. A. 2018. Enhancing recurrent neural network-based language models by word *token*ization. Human-centric Computing and Information Sciences, 8(1), 1-13.

Novantara, P. 2018. Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi. Buffer Informatika, 3(2).

Omar, N., & Al-Tashi, Q. 2018. Arabic nested noun compound extraction based on linguistic features and statistical measures. GEMA Online® Journal of Language Studies, 18(2).

Prayogo, A. H., & Mubarok, M. S. 2018. On the structure of Bayesian network for Indonesian *text* document paraphrase identification. In Journal of Physics: Conference Series (Vol. 971, No. 1, p. 012051). IOP Publishing.

Purba, A. H., & Situmorang, Z. 2017. Analisis Perbandingan Algoritma Rabin-Karp Dan *Jarak Levenshtein* Dalam Menghitung Kemiripan Teks. Jurnal Teknik Informatika UNIKA Santo Thomas, 2(2), 24-32.

Purnomo, W. G., & Purnomo, P. P. 2017. Akurasi *Text* Mining Menggunakan Algoritma K-Nearest Neighbour pada Data Content Berita SMS. Format, 6(1), 1-13.

Rahadian, B. A., Kurnianingtyas, D., Mahardika, D. P., Maghfira, T. N., & Cholissodin, I. 2017. Analisis Judul Majalah Kawanku Menggunakan Clustering K-Means Dengan Konsep Simulasi Big Data Pada Hadoop Multi Node Cluster. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN, 2355, 7699.

Rustiana, D., & Rahayu, N. 2017. Analisis Sentimen Pasar Otomotif Mobil: Tweet Twitter Menggunakan Naïve Bayes. Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer, 8(1), 113-120.

Saputra, P. Y., Subhi, D. H., & Winatama, F. Z. A. 2019. Implementasi Sentimen Analisis Komentar Channel Video Pelayanan Pemerintah Di Youtube Menggunakan Algoritma Naïve Bayes. Jurnal Informatika Polinema, 5(4), 209-213.

Silvana, H., Rullyana, G., & Hadiapurwa, A. 2018. Persepsi Mahasiswa Terhadap Tindakan Plagiarisme Dalam Penyusunan Tugas Akhir. EDUTECH, 16(3), 338-347.

Spelling, A. F. D. 2018. Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 5(4).