Sistem Pendeteksi Dini Plagiarisme Menggunakan Algoritma *Levenshtein Distance*

Muhamad Hendra Febiawan[1*], Agus Setiawan[2], Ardhin Primadewi[3]
Teknik Informatika,Universitas Muhammadiyah Magelang
*email: febiawan28@gmail.com

DOI: https://doi.org/10.31603/komtika.v3i1.3464

ABSTRACT

The academic world in Indonesia has growing rapidly which is marked by the development of science and technology. With all the facilities offered, technology has a positive and negative impact on life. One of negative impacts is plagiarism. The plagiarism often occurs among students, therefore detection of plagiarism needs to be done to avoid plagiarism. This reseacrh to detect the similarity of the text content of the document using the Levenshtein Distance algorithm. The type of document used is .pdf . The documents used are thesis proposals and publication papers. In the calculation speed test, the source document which count words 4405 with 3 comparative documents that have words 13465 count produces a calculation duration 3.57 seconds.

***Keywords* :** *Plagiarism, Levenshtein Distance Algorithm, Documents*

ABSTRAK

Dunia akademis di Indonesia sudah semakin pesat berkembang yang ditandai dengan perkembangan ilmu pengetahuan dan teknologi. Dengan segala kemudahan yang ditawarkan teknologi membawa dampak positif dan negatif bagi kehidupan. Salah satu dampak negatif yang ditimbulkan adalah plagiarisme. Fenomena plagiarisme sering terjadi pada kalangan mahasiswa, oleh karena itu pendeteksian plagiarisme perlu dilakukan untuk menghindari penjiplakan. Penelitian ini bertujuan untuk mendeteksi kemiripan isi teks dokumen menggunakan algoritma Levenshtein Distance. Tipe dokumen yang digunakan adalah .pdf . Dokumen yang digunakan adalah proposal skripsi dan Naskah Publikasi. Pada uji kecepatan perhitungan, dokumen sumber yang memiliki count words 4405 dengan 3 dokumen pembanding yang memiliki count words 13465 menghasilkan lama perhitungan selama 3,57 detik.

Kata-kata kunci : Plagiarisme, Algoritma Levenshtein Distance, Dokumen

PENDAHULUAN

Di Indonesia terdapat berbagai macam perguruan tinggi baik negeri atau swasta. Menurut data dari Pangkalan Data Pendidikan Tinggi (PDDIKTI) Kementerian Riset, Teknologi dan Pendidikan Tinggi [1], terdapat total 4.696 perguruan tinggi di Indonesia dengan rincian 1.062 Akademi, 279 Politeknik, 2.535 Sekolah Tinggi, 218 Institut, 584 Universitas dan 20 Akademi Komunitas. Data tersebut adalah total keseluruhan perguruan tinggi baik negeri dan swasta, sedangkan rincian total perguruan tinggi negeri di Indonesia adalah 436 perguruan tinggi dan total perguruan tinggi swasta di Indonesia adalah 4.260 perguruan tinggi. Di Provinsi Jawa Tengah sendiri menurut [1], terdapat 380 perguruan tinggi dengan rincian 38 perguruan tinggi negeri dan 342 perguruan tinggi swasta. Salah satu perguruan tinggi swasta di Provinsi Jawa Tengah adalah Universitas Muhammadiyah Magelang yang memiliki jumlah total dosen tetap berjumlah 183 Dosen dan 6.592 Mahasiswa dengan rasio dosen tetap/jumlah mahasiswa adalah 1 : 36 berdasarkan Data Pelaporan Tahun 2017/2018 menurut PDDIKTI [2]. Berdasarkan data tersebut, dunia akademis di Indonesia sudah semakin pesat berkembang yang ditandai dengan semakin banyaknya perguruan tinggi di Indonesia juga dengan perkembangan ilmu pengetahuan dan teknologi. Kemampuan mahasiswa dibidang ilmu pengetahuan dan teknologi terutama dibidang komputer menjadi

salah satu faktor yang mendukung dengan berkembangnya dunia akademis di Indonesia. Dengan segala kemudahan yang ditawarkan teknologi modern akan sangat membantu proses kegiatan belajar mengajar di lingkungan kampus, salah satunya dalam penulisan karya tulis ilmiah di Perguruan Tinggi yang merupakan bagian dari tuntutan formal akademik.

Dilihat dari tujuan penulisannya, karya tulis ilmiah dibedakan menjadi dua, pertama untuk memenuhi tugas-tugas perkuliahan seperti makalah, laporan dan lain-lain. Kedua, karya tulis ilmiah yang digunakan sebagai salah satu syarat mahasiswa untuk menyelesaikan program studi, yaitu skripsi untuk S1 dan tugas akhir untuk D3 [3]. Seperti di perguruan tinggi pada umumnya, skripsi di Universitas Muhammadiyah Magelang menjadi sangat penting dan merupakan bagian dari karya tulis ilmiah untuk menyelesaikan Program Sarjana S1. Skripsi merupakan kemampuan akademik mahasiswa dalam meneliti suatu kasus yang sesuai bidang keilmuannya. Melalui skripsi mahasiswa dituntut untuk berpikir sistematis. Hasil akhir dari karya tulis skripsi sendiri berbentuk Laporan Skripsi dan Naskah Publikasi. Naskah Publikasi merupakan naskah skripsi mahasiswa yang ditulis kembali dalam bentuk jurnal. Naskah publikasi ini merupakan salah satu bentuk karya tulis ilmiah mahasiswa pada kegiatan penelitian yang terbimbing oleh dosen. Penulisan naskah publikasi dapat diketahui terdapat konten penjiplakan atau *plagiarisme*. Tahap penyimpanan data – data skripsi di Teknik Informatika di Universitas Muhammadiyah masih manual sehingga seorang mahasiswa memiliki peluang untuk melakukan *copy – paste* atau *plagiarisme* proposal atau laporan akhir Skripsi dari awal hingga akhir tanpa diketahui oleh jurusan atau dosen. Dengan demikian *Plagiarisme* adalah tindakan mengambil ide orang lain, mengambil tulisan orang lain, dan mengambil teks secara keseluruhan dan mengakuinya sebagai milik pribadi [4].

Fenomena *plagiarisme* yang lebih spesifik sering terjadi pada dunia akademis, khususnya dilakukan mahasiswa. Hal ini dikarenakan kegiatan tulis menulis sering dilakukan mahasiswa untuk menyelesaikan tugas kuliah maupun tugas akhir. Mahasiswa sering berinteraksi dengan komputer, sehingga mempermudah praktik plagiat [5]. Ada dua cara untuk mengurangi tingkat *plagiarisme,* yaitu dengan mencegah dan mendeteksi. Mencegah berarti menjaga atau menghalangi agar *plagiarisme* tidak dilakukan. Usaha ini harus dilakukan sedini mungkin terutama pada sistem pendidik dan moral masyarakat [5]. Cara mendeteksi dokumen yang memiliki indikasi plagiarisme dapat dilakukan secara manual dengan cara membandingkan dua dokumen secara manual namun hal tersebut tidak efektif yang memakan banyak waktu dan tenaga. Oleh karena itu diperlukan sebuah sistem pendeteksi *plagiarisme* pada dokumen teks yang dilakukan secara kompatibel. Salah satu metode yang dapat digunakan dalam melakukan deteksi kemiripan dokumen teks adalah dengan melakukan perhitungan dengan metode *Levenshtein Distance* yang merupakan salah satu algoritma *text similarity,* yaitu algoritma untuk menghitung kemiripan dua *string input* yang dibandingkan [6].

Banyak metode yang digunakan untuk mendeteksi plagiarism diantaranya adalah *Naive Bayes* yang digunakan untuk mendeteksi kesamaan antar dokumen tugas mahasiswa. Dengan hal tersebut maka dapat memudahkan untuk memeriksa kesamaan dokumen unggahan mahasiswa dalam bentuk presentase *plagiarisme* [7]. Penelitian yang dilakukan [4] membuat sistem *plagiarisme* menggunakan algortima N-Grams dan Winnowing yang membandingkan file dokumen yang berbeda dari tugas akhir mahasiswa. Hasil yang diperoleh berupa persentase *similarity,* namun semakin banyak isi sebuah file yang dideteksi, waktu prosesnya

akan semakin lama. Sistem pendeteksi *plagiarisme* yang dapat digunakan untuk mendeteksi kesamaan dokumen skripsi degan cara melakukan perbandingan antara dokumen asli dan dokumen uji yang diinputkan untuk mengetahui tingkat kemiripan (*similarity*) dari dokumen skripsi yang diuji dengan menggunakan Algoritma Jaro-Winkler Distance [3]. Sistem pendeteksi kemiripan dokumen teks dengan tipe dokumen yang diuji .pdf .docx dan .txt dengan menggunakan algoritma *Levenshtein Distance*. Dokumen yang digunakan untuk perbandingan teks ini adalah dokumen berbahasa Indonesia. Pada pengujian data real yaitu data dokumen berplagiat dengan menghasilkan nilai *similarity* yang tinggi yaitu diatas 77% sampai 100% dengan ketentuan proporsi nilai plagiat dibawah 40%. Sistem pendeteksi plagiarisme dengan metode *K-Means* yang dapat mempartisi data ke dalam *cluster* sehingga data yang memiliki karakteristik sama dikelompokkan ke dalam satu *cluster* yang sama dan data yang mempunyai karakteristik berbeda di kelompokkan ke dalam *cluster* lain yang kemudian dicari nilai terdekat dari kemiripan antar dokumen [9].

Dalam penelitian ini dirancang sistem pendeteksi dini *plagiarisme* pada pengajuan skripsi mahasiswa jurusan teknik informatika menggunakan algoritma *Levenshtein Distance.* dengan *output* dari aplikasi berupa persentase kemiripan dan perbedaan isi teks dokumen agar dapat membantu dosen mendeteksi dini tindak *plagiarisme* sehingga dapat mengurangi tingkat *plagiarisme* pada kalangan mahasiswa Teknik Informatika Universitas Muhammadiyah Magelang.

METODE

Levenshtein Distance dibuat oleh Vladimir Levenshtein tahun 1965. Perhitungan edit *distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*. Perhitungan jarak antara dua string ini ditentukan dari jumlah minimum operasi perubahan untuk membuat *string* A menjadi *string* B. Ada 3 macam operasi utama yang dapat dilakukan oleh algoritma ini yaitu :

a. Operasi Pengubahan Karakter
   Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* ‘yamg’ menjadi ‘yang’. Dalam kasus ini karakter ‘m’ diganti dengan huruf ‘n’.
b. Operasi Penambahan Karakter
   Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string.* Contohnya *string* ‘kepad’ menjadi ‘kepada’, dilakukan penambahan karakter ‘a’ diakhir *string.* Penambahan karakter tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string.*
c. Operasi Penghapusan Karakter
   Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string* . Contohnya *string* ‘barur’ diubah menjadi ‘baru’. Pada operasi ini dilakukan penghapusan huruf ‘r’.

Algortima ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost.* Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string.*

Tabel 1. Tabel Matriks Perhitungan *Edit Distance*

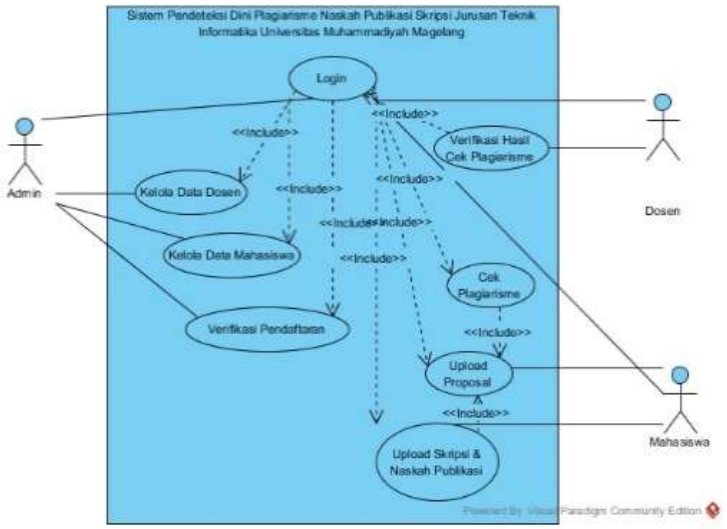
		S	A	Y	A
	0	1	2	3	4
S	1	0	1	2	3
Y	2	1	2	1	2
A	3	2	1	2	1

Contoh dari perhitungan *Levenshtein* menggunakan 2 *string* yang berbeda kemudian dihitung *edit distance* pada tabel 1 dapat dilihat hasil perhitungan *edit distance* antara 2 *string* ‘sya’ dan ‘saya’ adalah 1 perbedaan. Pengecekan dimulai dari iterasi awal dari kedua *string* kemudian dilakukan operasi penambahan, penyisipan dan penghapusan. Nilai *edit distance* yaitu pada ujung kanan bawah matriks. Hanya ada satu proses penyisipan yang dilakukan yaitu penyisipan karakter ‘a’ pada *string* ‘sya’ sehingga menjadi ‘saya’.

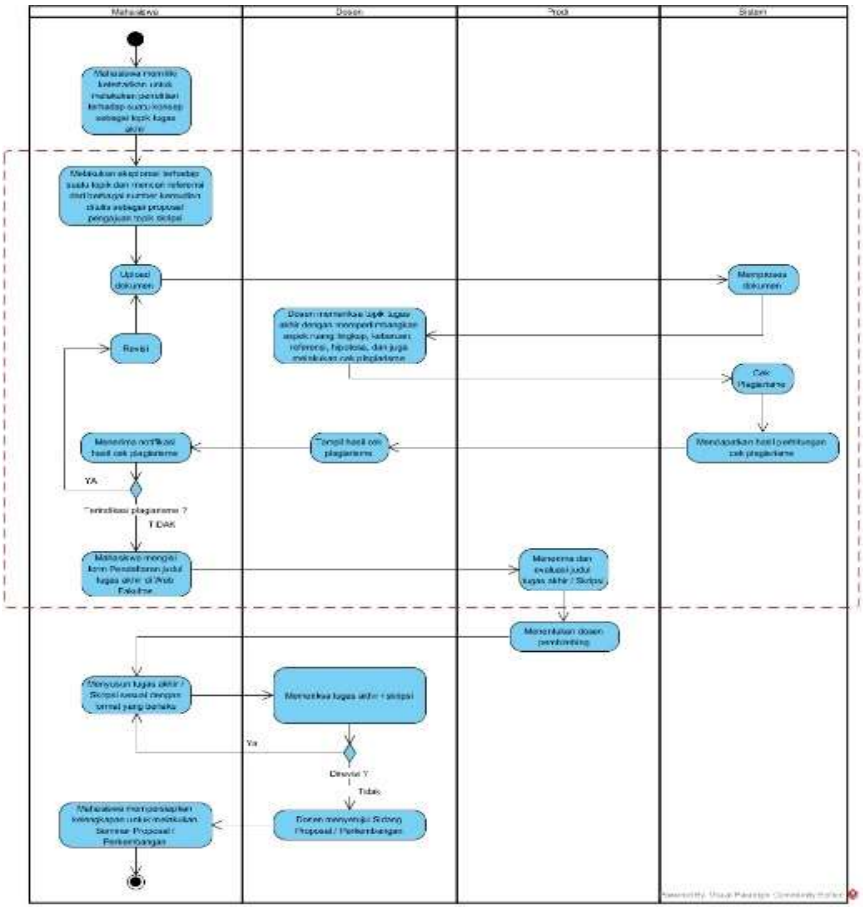
Levenshtein Distance sendiri merupakan salah satu algoritma *text similarity,* yaitu algoritma untuk menghitung kemiripan dua *string input* yang dibandingkan. Dalam kasus ini pengecekan *plagiarisme* memanfaatkan dokumen sumber dan dokumen target melalui tahap *Case Folding*, Tokenizing, Operasi Pengubahan Karakter, Operasi Penambahan Karakter dan Operasi Penghapusan Karakter. Proses *Case Folding* adalah tahap mengubah semua huruf dalam dokumen menjadi huruf kecil hanya huruf a sampai z yang diterima. Proses *Tokenizing* yaitu proses memisahkan setiap kata yang menyusun suatu dokumen. Umumnya setiap kata teridentifikasi atau terpisahkan dengan kata lain oleh karakter spasi. Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* ‘yamg’ menjadi ‘yang’. Dalam kasus ini karakter ‘m’ diganti dengan huruf ‘n’. Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string.* Contohnya *string* ‘kepad’ menjadi ‘kepada’, dilakukan penambahan karakter ‘a’ diakhir *string.* Penambahan karakter tidak hanya dilakukan di akhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string*. Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string* . Contohnya *string* ‘barur’ diubah menjadi ‘baru’. Pada operasi ini dilakukan penghapusan huruf ‘r’.

HASIL DAN PEMBAHASAN

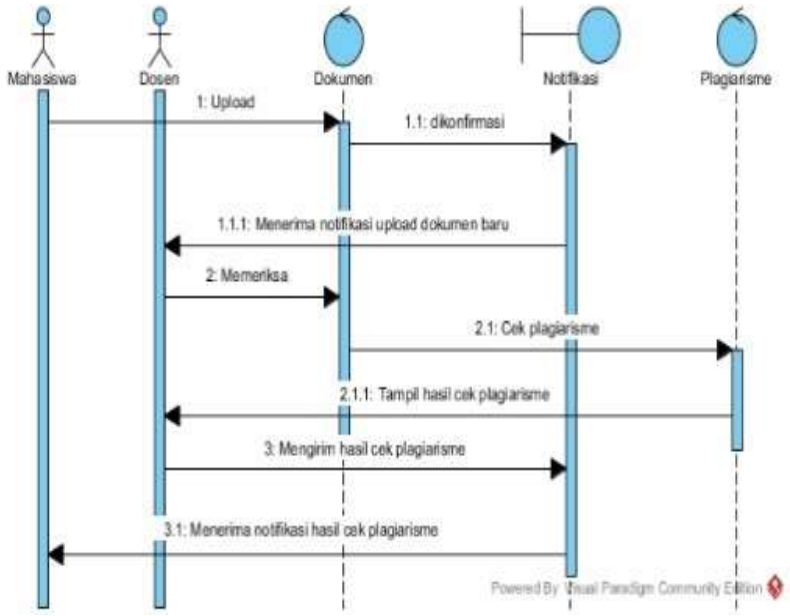
Perancangan sistem deteksi plagiarisme yang akan dirancang menggunakan *Unified Modelling Language* (UML) yang terdiri dari *use case diagram, activity diagram, sequence diagram* dan *class diagram* seperti pada Gambar 1,2,3 dan 4



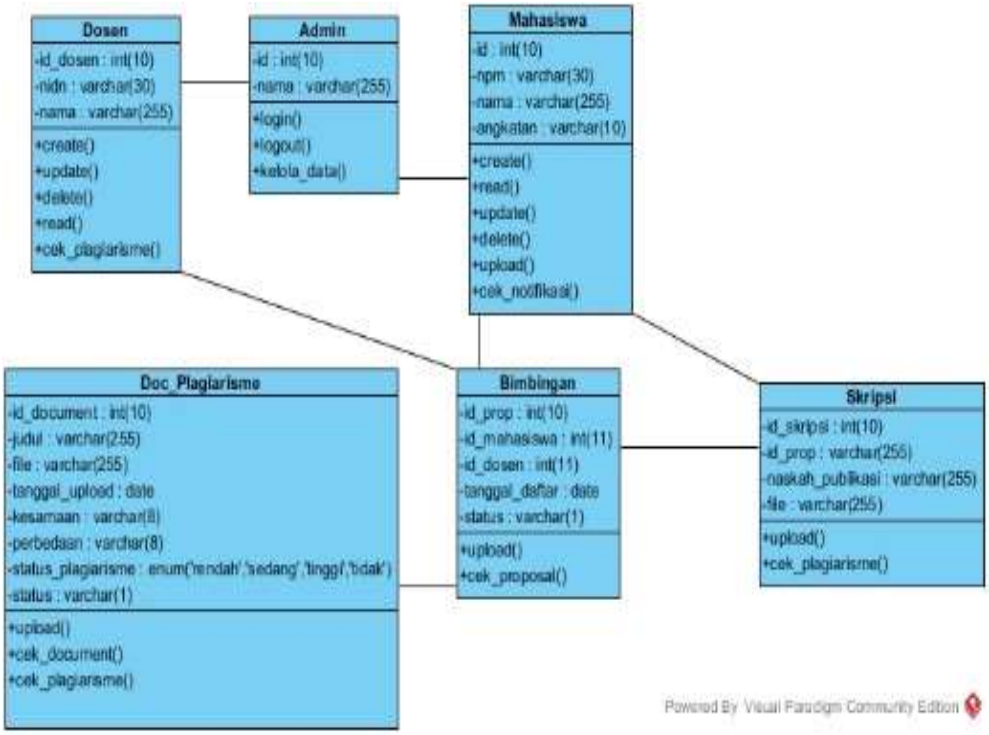
Gambar 1. *Use Case Diagram System*



Gambar 2. *Activity Diagram System*

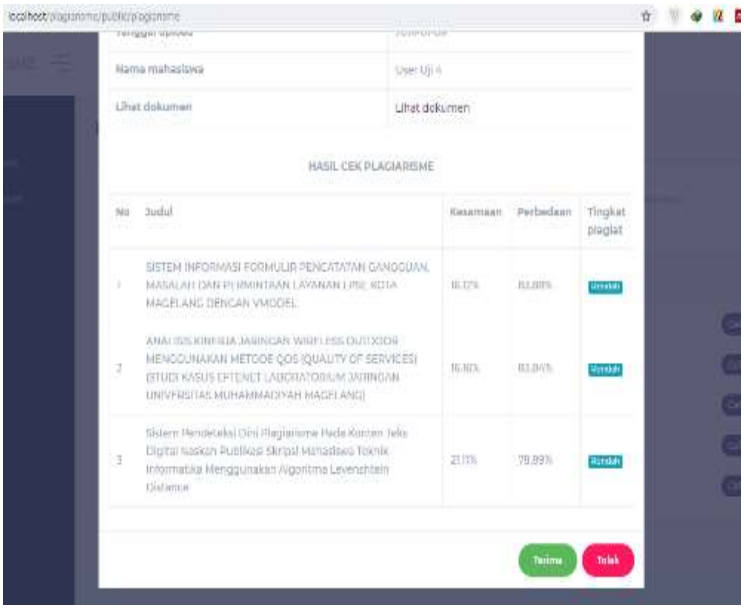


Gambar 3. *Sequence Diagram System*



Gambar 4. *Class Diagram System*

Pengujian algoritma *Levenshtein Distance* dilakukan dengan cara manual dengan cara melakukan *copy* terhadap *string* sumber dan *string* pembanding kemudian melakukan *paste* pada salah satu *variable script* program. Pengujian ini berfungsi untuk mengetahui *script* yang telah dibuat sudah dapat berjalan dengan baik atau belum, sebelum nanti diimplementasikan sebagai *library* pada sistem yang akan dibuat untuk mendeteksi *plagiarisme* pada dokumen.



Gambar 5. Hasil Pengujian Algoritma *Levenshtein Distance*

Tabel 2. Hasil Pengujian Algoritma

No.	Proses	Hasil yang Diharapkan	Kesimpulan
1	*Variable* $string1 dapat digunakan sebagai *string* sumber.	Dapat digunakan untuk meletakkan *string* sumber.	Diterima
2	*Variable* $string2 dapat digunakan sebagai *string* pembanding.	Dapat digunakan untuk meletakkan *string* pembanding.	Diterima
3	*Class Plagiarisme*.php menjadi *requirement* untuk mendapatkan *string* sumber dan pembanding dari *class index*.php	Dapat membaca *string* sumber dan pembanding pada *class index*.php	Diterima
4	*Function* getSimilarityPercentage() dapat menghitung nilai kesamaan kedua *string.*	Dapat menghitung nilai kesamaan.	Diterima
5	*Function* getDifferencePercentage() dapat menghitung nilai perbedaan kedua *string*	Dapat mengitung nilai perbedaan.	Diterima

Pada proses cek *plagiarisme*, tahapan paling utama adalah melakukan perhitungan mencari persentase nilai *string* yang sama. Secara umum mencari nilai persentase kesamaan *string* dengan memanfaatkan algoritma *Levenshtein Distance* dapat ditulis dengan rumus :

$$Plagiarisme\ Value = \left(\frac{diff}{\max(CS, ST)}\right) \times 100\%$$

Dengan Diff adalah Jarak *Levenshtein*, CS adalah *Source String*, ST adalah *Target String*
Contoh kasus, kita ambil kata "twitter v1" dan dengan kata pembanding "twitter v2", kemudian kita lakukan perhitungan dengan memanfaatkan metode *trigrams* :
Trigrams "twitter v1" : {_tw twi wit itt tte ter er_ _v1 v1_}
Trigrams "twitter v2" : {_tw twi wit itt tte ter er_ _v2 v1_}

Sehingga kita mendapatkan 7 angka diff dari kedua *string* diatas melalui cara *trigrams* yaitu 7 diff {_tw twi wit itt tte ter er_} dengan 11 nilai *max(CS,ST)* yaitu {_tw twi wit itt tte ter er_ _v1 _v2 v1_ v2_} maka dapat dihitung,

$$Plagiarisme\ Value = \left(\frac{7}{11}\right) \times 100\% = 63{,}63\ \%\ Similarity$$

Selain itu algoritma yang digunakan juga harus diuji lagi agar dapat mengetahui seberapa efektif algoritma yang digunakan untuk diterapkan pada sistem. Mengingat banyaknya *resources* yang dihitung termasuk banyak *words* yang terkandung dalam sebuah dokumen, maka telah dilakukan uji efektifitas kecepatan algoritma dalam membaca dan menghitung suatu dokumen.

Pada pengujian sistem yang dilakukan, sistem memeriksa dokumen sumber dengan dokumen pembanding. Daftar dokumen sumber yang diuji dapat ditunjukkan oleh Tabel 3. Hasil uji performa disajikan dalam Tabel 4 dan secara grafik disajikan dalam Gambar 6.

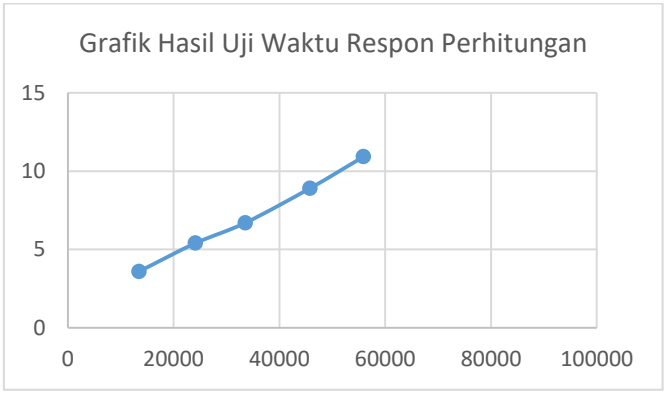
.

Tabel 3. Dokumen Sumber

No.	Nama File	Tipe File	Ukuran File	*Count Words*
1.	Tes Plagiat 1	.pdf	597 KB	4405
2.	Tes Plagiat 2	.pdf	462 KB	3288
3.	Tes Plagiat 3	.pdf	295 KB	1980
4.	Tes Plagiat 4	.pdf	659 KB	2890
5.	Tes Plagiat 5	.pdf	852 KB	2626

Tabel 4. Hasil Uji Performa

No.	Doc. Sumber	Jumlah Doc. Pembanding	*Count Words*	Waktu (H:M:S)
1.	Tes Plagiat 1	3	13465	00:00:03.57
2.	Tes Plagiat 2	5	24108	00:00:05.36
3.	Tes Plagiat 3	7	33554	00:00:06.67
4.	Tes Plagiat 4	9	45766	00:00:08.98
5.	Tes Plagiat 5	11	55871	00:00:10.91



Gambar 6. Grafik Hasil Uji Performa Algoritma

Dari hasil pengujian dokumen tersebut menunjukkan bahwa yang mempengaruhi lama tidaknya perhitungan nilai *similarity* adalah jumlah *string* dan *words* yang terkandung dalam dokumen. Dapat dilihat pada tabel 4, pada Tes Plagiat 1 ke Tes Plagiat 2 memiliki interval kenaikan waktu ± 2 detik dengan kenaikan jumlah *words* sebesar 10643, kemudian pada Tes Plagiat 2 ke 3 memiliki interval waktu ± 2 detik hampir sama dengan interval sebelumnya dengan selisih jumlah *words* 9446. Pada Tes Plagiat 3 ke 4 memiliki interval waktu ± 2 detik hampir 3 detik dengan selisih jumlah total *words* 12212, dan pada Tes Plagiat 4 ke 5 memiliki selang waktu selama ± 2 detik dengan selisih total *words* 10105. Selain itu, proses *parsing* dokumen dari format **.pdf** ke **.txt**, hal ini dapat dilihat melalui perbandingan sebelum dan sesudah *script* diterapkan pada sistem, yang pada awalnya belum dalam bentuk dokumen **.pdf** komputer dapat menghitung dengan sangat cepat, kemudian diubah dengan melalui proses *parsing* dokumen, sistem membutuhkan waktu yang sedikit lama.

KESIMPULAN

Kecepatan perhitungan algoritma *Levenshtein Distance* tergantung pada jumlah total *words* yang terkandung dalam dokumen sumber dan dokumen pembanding. Semakin banyak *words* yang terkandung maka akan membutuhkan waktu yang sedikit lama. Pada pengujian dengan menggunakan data real dokumen yang memiliki perbedaan dari dokumen 1 dengan *count words* berjumlah 629 dan dokumen 2 dengan *counts words* 128 diketahui nilai kesamaan 13,50% dan nilai perbedaan 86,50%. Pada pengujian efektifitas kecepatan perhitungan,dokumen sumber dengan *count words* 4405 diuji dengan 3 dokumen pembanding dengan total *count words* 13465 memiliki kecepatan menghitung selama 3,57 detik dan jika jumlah dokumen ditambah maka lama perhitungan juga akan naik. Proses *parsing* dokumen berbentuk .pdf ke dalam bentuk *string* menggunakan *Levenshtein Distance* mempengaruhi kecepatan waktu perhitungan cek *similarity.*

DAFTAR PUSTAKA

[1] P. KEMENRISTEKDIKTI, "Grafik Jumlah Perguruan Tinggi," 01 11 2018. [Online]. Available: https://forlap.ristekdikti.go.id/perguruantinggi/homegraphpt.

[2] F. PDDIKTI, "Pencarian Perguruan Tinggi," 01 11 2018. [Online]. Available: https://forlap.ristekdikti.go.id/perguruantinggi/search.

[3] O. P. Panji Novantara, "Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokume Skripsi," *Jurnal Buffer Informatika,* p. 8, 2018.

[4] F. L. L. J. K. Rocky Yefrenes Dillak, "Sistem Deteksi Dini Plagiarisme Tugas Akhir Mahasiswa Menggunakan Algortima N-Grams dan Winnowing," *Jurnal Ilmiah Flash, [S.1], v. 2, n. 1, p. 12-18,* p. 12, 2016.

[5] E. O. T. Y. Tudesman, "Sistem Deteksi Plagiarisme Dokumen Bahasa Indonesia Menggunakan Metode Vector Space Model," *Jurnal STMIK GI MDP,* p. Page 2, 2014.

[6] A. S. B. Hamidillah Ajie, "Aplikasi Pendeteksi Dugaan Awal Plagiarisme Pada Tugas SIswa dan Mahasiswa Berdasarkan Kemiripan Isi Teks Menggunakan Algoritma Levenshtein Distance," *Jurnal Pinter Vol. 1 No. 1 Juni 2017,* p. 24, 2017.

[7] A. B. P. D. Dwi Susanto, "Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata Kuliah Desain Web Menggunakan Metode Naive Bayes," *Nusantara Journal of Computers and its Applications,* p. 1, 2016.

[8] S. R. R. Na'firul Hasna Ariyani, "Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance," *semanTIK Vol. 2, No. 1,* p. 279, 2016.

[9] S. R. M. H. K. Mufti Ari Bianto, "Perancangan Sistem Pendeteksi Plagiarisme Terhadap Topik Penelitian Menggunakan Metode K-Means Clustering dan Model Bayesian," *Seminar Nasional Teknologi Informasi dan Multimedia 2018,* p. 19, 2018.

[10] S. Sastroasmoro, "Beberapa Catatan Tentang Plagiarisme," *Majalah Kedokteran Indonesia,* p. Volume 57, 2007.