Analisis Perbandingan Algoritma Rabin-Karp Dan Levenshtein Distance Dalam Menghitung Kemiripan Teks

[1]Andry Hery Purba, [2]Zakarias Situmorang
[1]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
[2]Teknik Informatika Unika St. Thomas S.U; Jln. Setia Budi No.479-F Medan, 061-8210161
e-mail : andryhery604@yahoo.com; [2]zakarias65@yahoo.com

Abstrak

Dalam dunia plagiarisme (penjiplakan) yang semakin marak terjadi di era modern saat ini, dimana dunia plagiarisme mudah dilakukan karena semakin majunya teknologi informasi. Terjadinya plagiarisme disebabkan karena kebiasaan peneliti yang malas dan ingin serba cepat dalam menyelesaikan tugas-tugasnya, sehingga kemungkinan peneliti akan mencari referensi yang terkait baik di internet maupun perpustakaan. Oleh karena itu karena itu diperlukan suatu aplikasi yang baik untuk menyelesaikan persoalan-persoalan di atas. Tugas akhir ini bertujuan untuk mendeteksi tingkat kemiripan teks berupa algoritma pencocokan string atau katadengan menggunakan algoritma Rabin-Karp dan Levenshtein Distance. Proses yang harus dilakukan dalam sistem adalah proses preprocessing pada masing-masing algoritma dan kemudian menghitung tingkat kemiripan teks menggunakan algoritma Rabin-Karp dan Levenshtein Distance. Sehingga mendapatkan hasil dari kedua algoritma yang ada pada sistem untuk dianalisis perbandingan hasilnya berupa grafik. Pada pengujian peneliti menggunakan dokumen teks dengan tipe .pdf dan berbahasa Indonesia.

Kata Kunci: *Teks, Plagiarisme, Rabin-Karp, Levenshtein Distance, Preprocessing*

Abstract

In the world of plagiarism (plagiarism) is increasingly prevalent in the modern era today, where the world of plagiarism is easy to do because of the advancement of information technology. The occurrence of plagiarism caused by the habits of researchers who are lazy and want a fast paced in completing tasks, so the possibility of researchers will look for references related both on the internet and the library. Therefore, it needs a good application to solve the above problems. This final project aims to detect the level of text similarity in the form of string or word matching algorithm using Rabin-Karp and Levenshtein Distance algorithm. Thus, the authors are interested to analyze the comparison between Rabin-Karp algorithm and Levenshtein Distance algorithm to calculate the level of similarity of text content. The process to be done in the system is the preprocessing process on each algorithm and then calculate the level of text similarity using Rabin-Karp and Levenshtein Distance algorithms. So get results from both algorithms that exist in the system to analyze the comparison of the results in the form of graphs. In testing the researcher using text documents with type .pdf and Indonesian language.

Keywords: *Text, Plagiarism, Rabin-Karp, Levenshtein Distance, Preprocessing*

1. PENDAHULUAN

Dalam dunia plagiarisme (penjiplakan) yang semakin marak terjadi di era modern saat ini, dimana dunia plagiarisme mudah dilakukan karena semakin majunya teknologi digital seperti dalam pembuatan laporan tulisan ilmiah di bidang akademik baik tingkat sekolah maupun perguruan tinggi. Terjadinya plagiarisme disebabkan karena kebiasaan peneliti yang malas dan ingin serba cepat dalam menyelesaikan tugas-tugasnya, sehingga kemungkinan peneliti akan mencari referensi yang terkait baik di internet maupun perpustakaan.

Teks ialah ungkapan bahasa yang menurut isi, sintaksis, dan pragmatik merupakan satu kesatuan . Sedangkan Baried [5] mengemukakan bahwa teks adalah kandungan atau muatan naskah, sesuatu yang abstrak hanya dapat dibayangkan saja.

Sebagai contoh, peneliti ingin mencari tulisan yang akan dijadikan sebagai bahan referensi penelitiannya. Ketika peneliti telah menemukan tulisan yang akan dijadikan sebagai referensi yang terkait, seringkali peneliti hanya melakukan teknik *copy*dan*paste* agar dapat menyelesaikan tugas-tugasnya dengan cepat tanpa perlu mempelajari dan memahami materi terlebih dahulu, maka inilah yang disebut plagiarisme karya orang lain.

Klasifikasi berdasarkan proporsi atau persentase kata, kalimat, paragraf yang dibajak yaitu : (1) Plagiarisme ringan, plagiarisme yang jumlah proporsi atau presentase kata, kalimat, maupun kalimat yang dibajak tidak melebihi 30 persen (< 30%), (2) plagiarisme sedang, plagiarisme yang jumlah proporsi atau presentase kata, kalimat, maupun paragraf yang dibajak antara 30-70 persen, (3) plagiarisme berat, plagiarisme yang jumlah proporsi atau presentase kata, kalimat, paragraf yang dibajak lebih dari 70 persen (>70%) [1].

Pelarangan terhadap plagiarisme sebenarnya sudah tercantum di UU Nomor 19 tahun 2002 pada pasal 2 ayat 1, Pasal 3 ayat 1 dan ayat 2, pasal 12, pasal 15, pasal 26 ayat 1, tetapi masih ada saja beberapa pelaku yang tidak terlalu mengerti atau tidak takut hukum. Oleh karena itu, untuk mencegah plagiarisme tulisan orang lain harus dilakukan sedini mungkin terutama dibidang akademis. Cara untuk mendeteksi plagiarisme terhadap tulisan dapat dilakukan dengan cara manual yaitu membandingkan kedua isi tulisan tersebut secara manual namun hal ini kurang efektif. Oleh karena itu diperlukan suatu aplikasi untuk menyelesaikan persoalan-persoalan di atas dengan menggunakan algoritma pencocokan *string* atau katapada teks tulisan tersebut. Algoritma dalam pencocokan *string*ada beberapa macam, antara lain :Rabin-Karp, Levenshtein Distance, Boyer-Moore, Brute Force, Knuth-Morris-Pratt, dan Smith-Waterman.

Pada uraian diatas, penulis tertarik untuk menganalisis perbandingan antara algoritma Rabin-KarpdanalgoritmaLevenshtein Distanceuntuk menghitung tingkat kemiripan isi teks padatulisan. Penulis memilih algoritma Rabin-Karp karena algoritma ini sangat efektif untuk pencarian lebih dari satu kata (multiple pattern) [2] dalam Noprisson,H., Susilo,B., dan Ernawati, 2014). Sedangkan algoritma Levenshtein Distance memperhatikan tigaoperasi dalam menentukan jarak diff, yaitu (1)operasi penyisipan (*insertion*), (2) operasipenghapusan (*deletion*), (3) operasi penggatian(*subtitution*), dalam sebuah huruf yang berdekatan [1].

.

2. METODE PENELITIAN

2.1 Pengertian Teks

Teks ialah ungkapan bahasa yang menurut isi, sintaksis, dan pragmatik merupakan satu kesatuan. Istilah teks sebenarnya berasal dari kata *text* yang berarti ‘tenunan’. Teks dalam filologi diartikan sebagai ‘tenunan kata-kata’, yakni serangkaian kata-kata yang berinteraksi membentuk satu kesatuan makna yang utuh.

Teks dapat terdiri dari beberapa kata, namun dapat pula terdiri dari milyaran kata yang tertulis dalam sebuah naskah berisi cerita yang panjang (Sudardi, 2001:4-5). Baried (1985:56) mengungkapkan bahwa teks adalah kandungan atau muatan naskah, sesuatu yang abstrak hanya dapat dibayangkan saja. Teks terdiri atas *isi*, yaitu ide-ide atau amanat yang hendak disampaikan pengarang kepada pembaca. Dan *bentuk*, yaitu cerita dalam teks yang dapat dibaca dan dipelajari menurut berbagai pendekatan melalui alur, perwatakan, gaya bahasa, dan sebagainya.

Dari pendapat diatas dapat disimpulkan penulis bahwa teks adalah satu kesatuan bahasa yang terdiri dari isi dan sintaksis serta memiliki makna yang akan disampaikan kepada pembaca.

2.2 Similarity Dokumen

Konsep *similarity* sering terjadi terutama pada bidang ilmu pengetahuan. Zaka (2009) dalam Salmuasih, dan Sunyoto (2013) mengemukakan tiga macam teknik yang dibangun untuk menentukan nilai *similarity* (kemiripan) dokumen.

1. *Distance-based similarity measure*
   Distance-based similarity measure mengukur tingkat kesamaan dua buah objek dari segi jarak geometris dari variabel-variabel yang tercakup di dalam kedua objek tersebut. Metode *Distance-based similarity* ini meliputi *Minkowski Distance, Manhattan/City block Distance, Euclidean Distance, Jaccard Distance, Dice's Coefficient, Cosine similarity, Levenshtein Distance, Hamming Distance,* dan *Soundex Distance.*
2. *Feature-based similarity measure*
   Featured-based similarity measure melakukan penghitungan tingkat kemiripan dengan merepresentasikan objek ke dalam bentuk *feature-feature* yang ingin diperbandingkan. *Feature-based similarity measure* banyak digunakan dalam melakukan pengklasifikasian atau *pattern matching* untuk gambar dan teks.
3. *Probabilistic-based similarity measure*
   Probabilistic-based similarity measure menghitung tingkat kemiripan dua objek dengan merepresentasikan dua set objek yang dibandingkan dalam bentuk *probability*. *Kullback Leibler Distance* dan *Posterior Probability* termasuk dalam metode ini.

2.3 Plagiarisme

Plagiarisme merupakan tindakan kriminal yang sering terjadi dalam dunia akademis. Plagiarisme berasal dari kata latin *"Plagiarus"* yang berarti penculik dan "*Plagiare*" yang berarti mencuri. Jadi, secara sederhana plagiarisme berarti mengambil ide, kata-kata, dan kalimat seseorang dan memposisikannya sebagai hasil karyanya sendiri atau menggunakan ide, kata-kata, dan kalimat tanpa mencantumkan sumber dimana seorang penulis mengutipnya [6].

Pengertian lain menurut Sugono. (1997) dalam Salmuasih, dan Sunyoto (2013) mengemukakan plagiarisme atau plagiat adalah penjiplakan atau pengambilan karangan, pendapat orang lain dan menjadikannya seolah-olah karangan sendiri.

Dari pendapat di atas dapat disimpulkan penulis bahwa plagiarisme merupakan "Tindakan kriminal dalam penyalahgunaan karya orang lain berupa hasil pemikiran baik ide atau ciptaan yang dijadikan sebagai hasil pemikirannya**".**

2.4 Text Mining

Tan (1999) dalam Rinusantoro (2014) mengemukakan bahwa *text mining* adalah proses ekstraksi pengetahuan dari dokumen teks yang tidak terstruktur. *Teks mining* merupakan perluasan dari data *mining* atau *knowledge discovery in database* yang menemukan pola-pola menarik dari basis data berskala besar. Sedangkan menurut Firdaus, Ernawati, dan Vatresia (2014) *text mining* adalah proses menganalisis teks untuk mengekstrak informasi yang berguna untuk tujuan tertentu.

2.5 Ruang Lingkup Text Mining

Menurut Affandi (2002) dalam Rinusantoro (2014) Ada beberapa proses yang dilakukan dalam teks *mining* untuk mendapatkan informasi yang tersembunyi, diantaranya adalah pemrosesan awal terhadap teks (*text preprocessing*), transformasi teks (*text transformation*), pemilihan fitur (*feature selection*) dan penemuan pola (*pattern discovery*).

1. *Text Preprocessing*
   Proses mempersiapkan teks menjadi data yang akan mengalami pengolahan lebih lanjut. Dalam *preprocessing* melakukan pembersihan teks untuk tiap-tiap kalimat.
2. *Text Transformation*
   Proses transformasi bertujuan untuk mendapatkan representasi dokumen yang diharapkan. Transformasi teks mengubah kata-kata ke dalam bentuk dasarnya dan pengurangan dimensi kata di dalam dokumen tersebut.
3. *Feature Selection*
   Pemilihan fitur kata merupakan tahap dari proses pengurangan dimensi pada proses transformasi. Selain melakukan penghapusan kata-kata yang dianggap tidak deskriptif (*stopwords*), tidak semua kata dalam dokumen memiliki arti yang penting. Oleh karena itu, untuk mengurangi dimensi dari dokumen dilakukan pemilihan terhadap kata-kata yang dianggap penting dan memiliki hubungan yang erat dengan isi dokumen. Ide dasar dari pemilihan fitur adalah menghapus kata-kata yang kemunculannya di suatu dokumen terlalu sedikit atau terlalu banyak.
4. *Pattern Discovery*
   Pattern discovery merupakan tahap penting untuk menemukan pola atau pengetahuan dari keseluruhan teks. Masukan awal proses *textmining* adalah data teks dan menghasilkan keluaran pola sebagai hasil dari interpretasi. Apabila hasil keluaran dari penemuan pola belum sesuai untuk aplikasi, dilanjutkan evaluasi dengan melakukan iterasi ke satu atau beberapa tahap sebelumnya.

2.6 Ekstraksi Dokumen

Teks yang akan dilakukan proses *text mining*, pada umumnya memiliki beberapa karakteristik diantaranya adalah memiliki dimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Diperlukan tahap *preprocessing*yang bertujuan untuk menyeragamkan kata dan menghilangkan *noise* yang terdiri dari imbuhan, angka, simbol dan karakter yang tidak relevan, serta kata-kata yang tidak penting.

Sjukani (2013) dalam Pratama danPamungkas (2016) mengemukakan tahap-tahap *preprocessing* adalah *Case folding* atau *Converting*, *Tokenizing*, *Stopword removal*, *Stemming*, dan *Sorting*.

1. *Case folding (Converting)*
   *Case folding*adalah proses mengubah semua huruf dalam dokumen menjadi huruf kecil, agar sistem mampu menyamakan tiap karakter dari dokumen satu dengan dokumen lainnya.
2. *Tokenizing*
   *Tokenizing*adalah tahap pengeliminasian simbol dan karakter yang tidak relevan dihilangkan, seperti tanda baca, angka, dan lain-lain.
3. *Stopword removal*
   *Stopword removal*adalah tahap pengeliminasian kata-kata yang tidak penting dari hasil *tokenizing*, seperti ‘yang’, ‘di’, ‘ke’, ‘pada’, dan lain-lain. Penghapusan kata-kata ini dimaksudkan untuk lebih mempermudah pembandingan dokumen.
4. *Stemming*
   *Stemming*adalah proses untuk mencari *root* kata (kata dasar) dari kata berimbuhan hasil proses *stopword removal*. Pada tahap ini dilakukan proses pengembalian berbagai bentukan

kata ke dalam suatu representasi yang sama. Tahap ini kebanyakan dipakai untuk teks berbahasa Inggris dan lebih sulit diterapkan pada teks berbahasa Indonesia.

5. *Sorting*
   *Sorting*adalah proses mengurutkan data baik secara menaik (*ascending*) atau secara menurun (*descending*). Data yang diurut bisa bertipe numerik maupun karakter.

2.6 Algoritma Rabin-Karp

Algoritma Rabin-Karp adalah algoritma pencocokan *string* yang menggunakan fungsi *hash* sebagai pembanding antara *string* yang dicari (m) dengan *substring* pada teks (n). Apabila *hash value* keduanya sama maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila hasil keduanya tidak sama, maka *substring* akan bergeser ke kanan. Pergeseran dilakukan sebanyak (n-m) kali. Perhitungan nilai *hash* yang efisien pada saat pergeseran akan mempengaruhi performa dari algoritma ini.

2.7.1 Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah *string* menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda *string* tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi *hash*, sedangkan nilai yang dihasilkan disebut nilai *hash.* (Firdaus, 2008 dalamPutra, Sujaini, dan Pratiwi, 2015).
Contoh sederhana *hashing* adalah :

Andry; Katolik
Natal; Agustin
Menjadi :
6992 = Andry; 8092 = Katolik
1880 = Natal; 2828 = Agustin

Nilai *hash* pada umumnya digambarkan sebagai *fingerprint* yaitu suatu *string* pendek yang terdiri atas huruf dan angka yang terlihat acak (data biner yang ditulis dalam heksadesimal). (Firdaus, 2008 dalam Noprisson, dkk, 2014).

2.7.2 K-gram

K-gram adalah rangkaian *terms* dengan panjang K. Kebanyakan yang digunakan sebagai *terms* adalah kata. *K-gram* merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau karakter. Metode *K-gram* digunakan untuk mengambil potongan-potongan karakter huruf sejumlah k dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen. (Yoga, 2012 dalamPutra, dkk, 2015).

Contoh penggunaan dari *K-grams*, trigram dari Menganalisis dan Menganalisa, yaitu "Men, eng, nga, gan, ana, nal, ali, lis, isi, sis", dan "Men, eng, nga, gan, ana, nal, ali, lis, isa". Sehingga dapat disimpulkan bahwa tigram dari kedua kata tersebut yang memiliki bentuk yang sama yaitu "Men, eng, nga, gan, ana, nal, ali, lis".

2.7.3Dice's Similarity Coefficient

Perhitungan *similarity* dari kumpulan katamenggunakan *Dice's Similarity Coefficient* untukpasangan kata yang digunakan. Dalam algoritmaini, pertama kata dipotong menjadi *K-gram*, kumpulan karakter dengan panjang K. Kedua, gabungkan kata-kata yang serupa yaitu yangmemiliki*K-gram* identik. Untukmemperkirakan kesamaan kata-kata danmembuatkeputusan mana yang dapat menjadikandidat yang lebih baik untuk penggabungan*Dice's Similarity* (Kosinov 2002 dalamPutri, Mardiyono, Handoko, 2013).
Nilai *Similarity* tersebut dapat dihitung dengan menggunakan :

$$\mathbf{S} = \frac{\mathbf{2 \mid A \cap B \mid}}{\mathbf{|A| + |B|}} \text{x } 100\%$$

Penjelasan :
S = Nilai *similarity*
| A ∩ B |= Jumlah *k-gram* unik dan memiliki struktur yang sama dari masing-masing teks yang dibandingkan
|A| + |B| = Jumlah *k-gram* yang unik dari teks pertama dan teks kedua.

2.8 Algoritma Levenshtein Distance

Levenshtein Distance dibuat oleh Vladimir Levenshtein Distance pada tahun 1965. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*. (Ariyani, dkk, 2016).

Levenshtein Distance adalah sebuah matriks *string* yang digunakan untuk mengukur perbedaan atau jarak (*distance*) antara dua *string*. Nilai *distance* antara dua *string* ini ditentukan oleh jumlah minimum dari operasi-operasi perubahan yang diperlukan untuk melakukan transformasi dari suatu *string* menjadi *string* lainnya. Operasi-operasi tersebut adalah penyisipan (*insertion*), penghapusan (*deletion*), atau penukaran (*subtition*). Levenshtein Distancemerupakan salah satu algoritma yang dapat digunakan dalam mendeteksi kemiripan antara dua *string* yang berpotensi melakukan tindak plagiarisme. (Zhan Su, dkk, 2015 dalamPratama and Pamungkas, 2016).

2.8.1 Langkah-Langkah Algoritma Levenshtein Distance

Algoritma Levenshtein Distance berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi (matriks) yang telah diisi sejumlah karakter *string* awal dan *string* target. Entri-entri pada matriks tersebut merepresentasikan nilai terkecil dari transformasi *string* awal menjadi*string* target. Entri yang terdapat pada ujung kanan bawah matriks adalah nilai *distance* yang menggambarkan jumlah perbedaan dua *string.*(Junedy, 2014 dalamPratama *and* Pamungkas, 2016).

Berikut ini adalah langkah-langkah algoritma Levenshtein Distance dalam mendapatkan nilai *distance* (Haldar dan Mukhopadhyay, 2011 dalam Pratama and Pamungkas, 2016) :
Langkah 1: Inisialisasi
a) Hitung panjang S dan T, misalkan m dan n
b) Buat matriks berukuran 0…m baris dan 0…n kolom
c) Inisialisasi baris pertama dengan 0…n
d) Inisialisasi kolom pertama dengan 0…m
Langkah 2: Proses
a) Periksa S[i] untuk 1 < i < n
b) Periksa T[j] untuk 1 < j < m
c) Jika S[i] = T[j], maka entrinya adalah nilai yang terletak pada tepat didiagonal atas sebelah kiri, yaitu d[i,j] = d[i-1, j-1].
d) Jika S[i] ≠ T[j], maka entrinya adalah d[i,j] minimum dari:
   - Nilai yang terletak tepat diatasnya, ditambah satu, yaitu d[i,j-1]+1
   - Nilai yang terletak tepat dikirinya, ditambah satu, yaitu d[i-1,j]+1
   - Terletak pada tepat didiagonal atas sebelah kirinya, ditambah satu, yaitu d[i-1,j-1]+1

Langkah3: Hasil entri matriks pada baris ke-i dan kolom ke j, yaitu d[i,j]
Langkah 4.Diulang hingga entri d[m,n] ditemukan.

2.8.2 Bobot Similarity

Levenshtein Distancemelakukan perhitungan bobot *similarity* setelah mendapatkan nilai*distance* dari dua dokumen yang dibandingkan. Kemudian menggunakan suatu persamaan dalam menentukan bobot *similarity*, yaitu (Winarsono,dkk, 2009 dalamPratama *and* Pamungkas, 2016) :

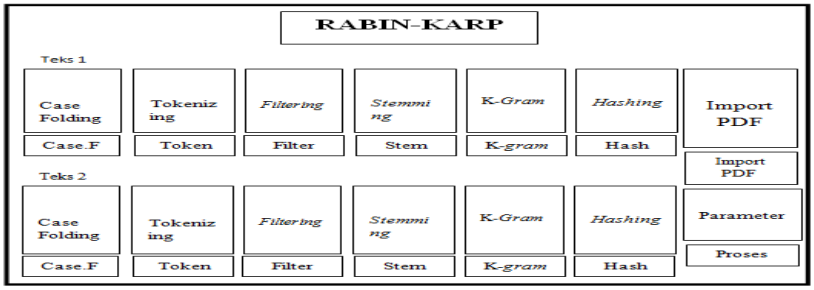
$$B = \left(1 - \frac{d[m,n]}{Max\ (S,T)}\right) * 100\%$$

Dimana d[m,n] adalah nilai *distance*, terletak pada baris ke m dan kolom ke n, S adalah panjang *string* awal, T adalah panjang *string* target, dan Max(S,T) adalah panjang *string* terbesar antara *string* awal dan *string* target.

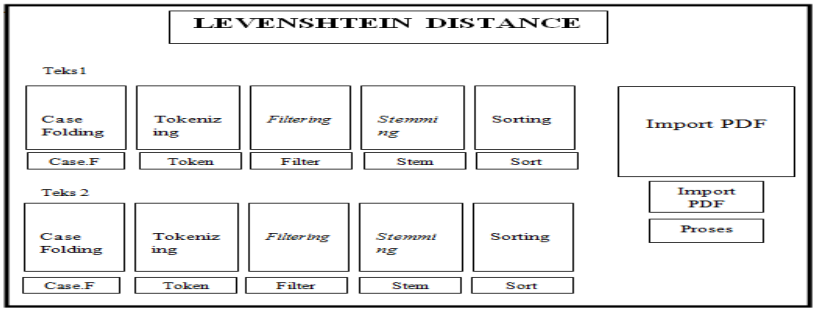
Bobot *similarity* diasumsikan pada rentang 0 (nol) hingga 100 (seratus) persen, yang artinya nilai 100 adalah nilai maksimum yang menunjukan bahwa dua kata adalah sama identik. Pendekatan ini mampu digunakan untuk mengukur bobot *similarity* antar dua *string* berdasarkan susunan karakter. (Erick, 2011).

3. HASIL DAN PEMBAHASAN

3.1. Hasil

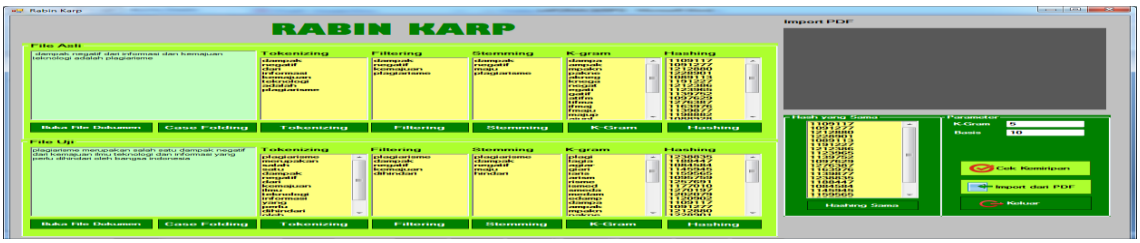


Gambar 3.1 Rancangan Form Algoritma Rabin Karp

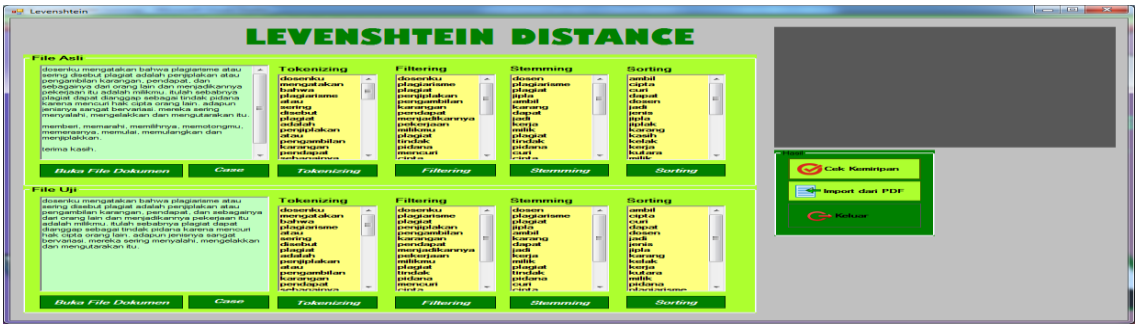


Gambar 3.2 Rancangan Form Algoritma Levenshtein Distance

Setelah tahap implementasiselesai, maka akan dilanjutkan kepadatahap pengujiansistem.Dalamhalinisistem akandiujicobadalammelakukanproses kemiripan teks.Sehinggadapatdiketahuikeberhasilan darisistemyang dibangun.



Gambar 3.3 Form Rabin-Karp



Gambar 3.4 Pendeteksian Kemiripan Teks Dengan Algoritma Levenshtein Distance

Apabila proses pendeteksian berhasil, maka akan ditampilkan form analisis hasil dari kemiripan teks dengan Algoritma Rabin Karp. Tampilan form dapat dilihat pada Gambar 3.5 berikut :

Gambar 3.5 Hasil dari Algoritma Levenshtein Distance

4. KESIMPULAN

Berdasarkan hasil analisis, perancangan, dan pengujian yang telah penulis lakukan maka penulis memperoleh beberapa kesimpulan, diantaranya adalah :

1. Aplikasi yang dibangun berguna untuk mendeteksi tingkat kemiripan antar kedua teks yang bertujuan untuk mennghindari adanya tindakan plagiarisme dengan menggunakan algoritma Rabin-Karp dan Levenshtein Distance.
2. Semakin banyak jumlah karakter atau kalimat dalam teks maka semakin besar juga waktu yang dibutuhkan dalam mendeteksi tingkat kemiripan

5. SARAN

Dalam pengembangan lebih lanjut, maka penulis memberikan saran-saran sebagai berikut:

1. Algoritma Rabin-Karp sangat tepat untuk pencocokkan multiple pattern apabila dibandingkan dengan single pattern.
2. Penelitian dapat dikembangkan dengan melakukan pendeteksian antara teks 1 dengan beberapa teks yang lainnya.
3. Penelitian dapat dikembangkan dengan menambahkan beberapa jenis file yang berbeda, seperti file dokumen (doc, docx).

DAFTAR PUSTAKA

[1] Afdhal, Chalis,T., and Gani,T.A., 2014, Analisa Perbandingan Aplikasi Pendeteksi Plagiat Terhadap Karya Ilmiah, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala, Aceh, pp. 193-199.

[2] Ariyani,N.H., Sutardi, and Ramadhan,R., 2016, *Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance*, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari, Vol.2, No.1, pp. 279-286, ISSN : 2502-8928.

[3] Firdaus,A., Ernawati dan Vatresia,A., 2014, Aplikasi Pendeteksi Kemiripan Pada Dokumen Teks Menggunakan Algoritma Nazief & Adriani Dan Metode Cosine Similarity. Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu, Bengkulu, Vol.10, No.1, pp. 97-110.

[4] Firdaus,H.B., 2008, Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika , Institut Teknologi Bandung, Bandung, Vol.3, No.2.

[5] Kurniawan, Erick. 2011. Cepat Mahir Visual Basic 2010. Edisi satu. C.V ANDI OFFSET. Yogyakarta.

[6] Kurniawati,A., Wicaksana,I.W.S., 2008, Perbandingan Pendekatan Deteksi Plagiarisme Dokumen Dalam Bahasa Inggris, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma, Depok, pp. 284-291.

[7] Noprisson,H., Susilo,B., and Ernawati, 2014, *Implementasi Algoritma Rabin-Karp Untuk Menentukan Keterkaitan Antarpublikasi Penelitian Dosen Tahun 2013*, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Bengkulu, Bengkulu, Vol.10, No.1, pp. 110-127.

[8] Pratama,B.P., Pamungkas,S.A., 2016, Analisis Kinerja Algoritma Levenshtein Distance Dalam Mendeteksi Kemiripan Dokumen Teks. Program Studi Matematika, Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah, Jakarta, Jilid 6, No.2, hal. 131-143.

[9] Putra,D.A., Sijaini,H., dan Pratiwi,H.S., 2015, Implementasi Algoritma Rabin-Karp untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah, Program Studi Teknik Informatika, Fakultas Teknik, Universitas Tanjungpura, Vol.1, No.1.

[10] Putri,D.R., Mardiyono dan Handoko,S., 2013, Portal Open Source Pendeteksi Plagiarisme di Kalangan Negeri Semarang, Jurusan Teknik Elektro, Politeknik Negeri Semarang, Semarang, Vol.2, No.2.

[11] Rinusantoro,S. (2014). *Aplikasi Deteksi Kemiripan Dokumen Teks*. Tesis. Yogyakarta: Universitas Gadjah Mada.

[12] Salmuasih, Sunyoto,A., 2013, Implementasi Algoritma Rabin Karp untuk Pendeteksian Plagiat Dokumen Teks Menggunakan Konsep Similarity, Program Studi teknik Informatika, STMIK AMIKOM Yogyakarta, Yogyakarta, pp. 23-28.