**JARO-WINKLER DISTANCE DAN LONGEST COMMON SUBSEQUENCE PADA SEARCH SUGGESTION**

**Proposal Skripsi**



diajukan oleh :

**AHMAD HAFIZH MUTTAQI**

**15-2016-137**

**PROGAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

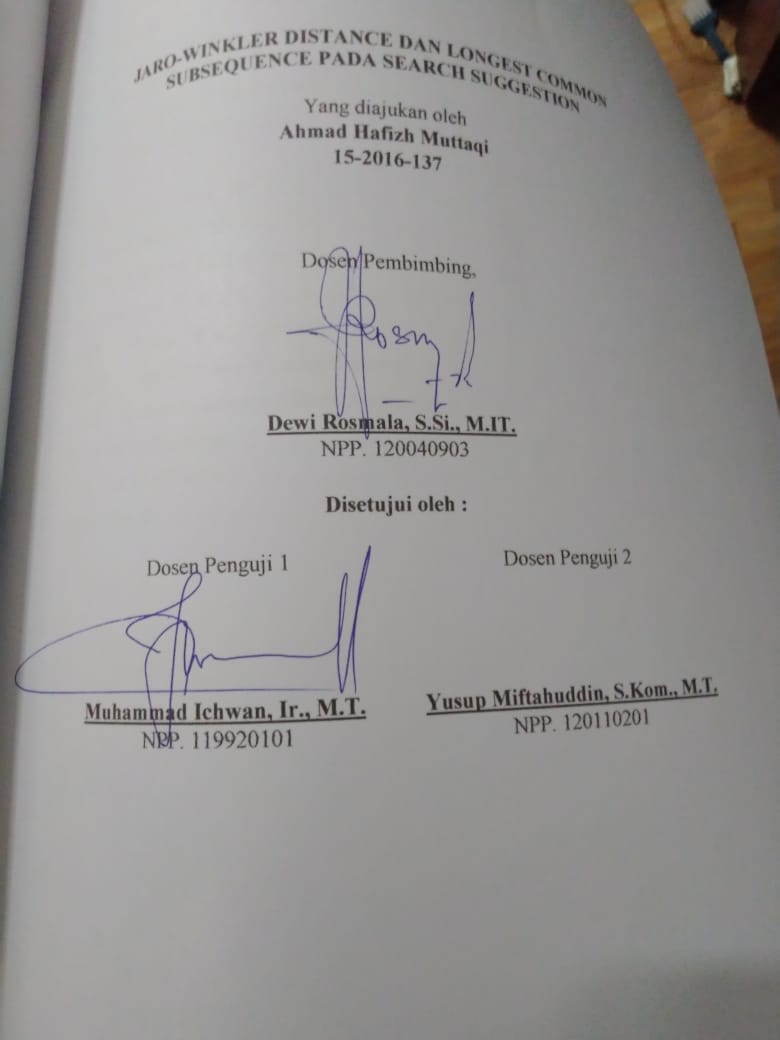
**FAKULTAS TEKNOLOGI INDUSTRI**

**INSTITUT TEKNOLOGI NASIONAL**

**BANDUNG**

**2020**

**Lampiran Lembar Pengesahan**

****



DAFTAR ISI

[DAFTAR ISI i](#_Toc34121735)

[DAFTAR GAMBAR ii](#_Toc34121736)

[1. LATAR BELAKANG 1](#_Toc34121737)

[2. IDENTIFIKASI MASALAH 2](#_Toc34121738)

[3. RUMUSAN MASALAH 2](#_Toc34121739)

[4. TUJUAN 3](#_Toc34121740)

[5. RUANG LINGKUP 3](#_Toc34121741)

[6. KONTRIBUSI PENELITIAN 3](#_Toc34121742)

[7. TINJAUAN PUSTAKA 3](#_Toc34121743)

[8. Landasan Teori 6](#_Toc34121744)

[8.1. Metode String Matching 6](#_Toc34121745)

[8.2. Algoritma Jaro Winkler Distance 6](#_Toc34121746)

[8.3. Longest Common Subsequence 9](#_Toc34121747)

[9. Metode Penelitian 12](#_Toc34121748)

[9.1 Rancangan Pengujian 12](#_Toc34121749)

[DAFTAR PUSTAKA 16](#_Toc34121750)

[LAMPIRAN 18](#_Toc34121751)

# DAFTAR GAMBAR

[Gambar 1. Algoritma Jaro Winkler Distance 8](#_Toc34121791)

[Gambar 2 Contoh Matriks LCS 10](#_Toc34121792)

[Gambar 3 Contoh Matriks LCS 2 10](#_Toc34121793)

[Gambar 4 Contoh Matriks LCS 3 11](#_Toc34121794)

[Gambar 5 Block Diagram Penelitian 12](#_Toc34121795)

[Gambar 6 Ilustrasi Input Kata Kunci 12](#_Toc34121796)

[Gambar 7 Judul Film yang telah menjadi array 13](#_Toc34121797)

[Gambar 8 Ilustrasi Perhitungan LCS 14](#_Toc34121798)

# LATAR BELAKANG

Pencarian merupakan kegiatan yang biasanya dilakukan sehari-hari, proses pencarian pada komputer biasanya dilakukan dengan memasukan kata kunci terlebih dahulu, tujuannya untuk mengetahui data dengan kata kunci ada pada dataset tertentu atau tidak (Rahim, Nurarif, Ramadhan, Aisyah, & Purba, 2017). Tetapi kesalahan pengetikan sering terjadi saat memasukan kata kunci pencarian sehingga menyebabkan hasil pencarian yang diinginkan tidak ditemukan pada dataset, karena ejaan kata kunci yang tidak sesuai. Beberapa faktor yang menyebabkan terjadinya kesalahan pengetikan adalah letak huruf pada keyboard yang berdekatan, kesalahan karena slip pada tangan atau jari, atau kesalahan yang disebabkan oleh ketidaksengajaan (Rochmawati & Kusumaningrum, 2016). Dengan membuat sugesti pencarian berdasarkan kata kunci, hasil pencarian yang diinginkan bisa tetap muncul walaupun dengan beberapa kesalahan penulisan atau dari ejaan kata yang ingin dicari dan memudahkan pengguna dengan sugesti pencariannya. Salah satu pendekatan yang dapat dilakukan untuk mengidentifikasi kesalahan pengetikan adalah metode pencocokan string (Rochmawati & Kusumaningrum, 2016).

Pada penelitian sebelumnya yang dilakukan oleh Rochmawati ada beberapa algoritma yang dapat digunakan untuk pencocokan string yang dibandingkan, diantaranya adalah Hamming Distance, Levenshtein Distance, Damerau Levenshtein Distance dan Jaro Winkler Distance. Hasil penelitiannya menyatakan bahwa Hamming Distance, Levenshtein Distance dan Damerau Levenshtein Distance memiliki nilai Mean Average Precision yang masih berada dibawah Jaro Winkler Distance (Rochmawati & Kusumaningrum, 2016). Kemudian pada penelitian lain yang telah dilakukan oleh Zhang, pada penelitiannya disebutkan bahwa Algoritma dari Levenshtein Distance masih kurang cukup pada aspek akurasi untuk pencarian string (Zhang, Hu, & Bian, 2017) membuat algoritma ini masih kurang cocok untuk memperbaiki kata kunci untuk sugesti pencarian.

Jaro Winkler Distance menunjukan hasil yang memuaskan dengan Mean Average Precision untuk 50 kata ada pada angka 0,87 yang merupakan hasil tertinggi dibandingkan dengan 3 metode lain yang dibandingkan oleh Rochmawati (Rochmawati & Kusumaningrum, 2016). Penelitian lain yang dilakukan oleh Tannga menghasilkan nilai rata-rata similarity dari algoritma Jaro Winkler yaitu sebesar 80.92 %, sedangkan untuk algoritma Levensthein Distance rata-rata nilai similarity-nya yaitu sebesar 49.43% (Tannga, Rahman, Informatika, & Distance, 2017). Karena hasil penelitian-penelitan sebelumya menunjukan keunggulan Jaro Winkler Distance, diharapkan dengan menggunakan metode ini dapat membuat sugesti pencarian yang lebih baik.

Longest Common Subsequence (LCS) adalah metode untuk mencari karakter sama yang dari 2 buah kata, metode ini juga bisa digunakan digunakan untuk mencari kemiripan karakter pada string (Zhang et al., 2017). LCS menghasilkan karakter yang sama dari dua buah string yang nantinya panjang dari karakter yang sama dapat dibuat persentase kemiripan karakter dari dua string yang dibandingkan. Persentase kemiripan dari LCS akan dijadikan parameter tambahan untuk melakukan pencarian data, sehingga dengan menggabungkan LCS dan Jaro Winkler dapat menghasilkan sugesti pencarian yang lebih baik.

# IDENTIFIKASI MASALAH

Hasil penggunaan Algoritma *Jaro Winkler Distance* untuk pencarian stringmenghasilkan nilai jarak yang dirasa kurang cukup untuk dijadikan patokan pada pembuatan sugesti pencarian sehingga akan ditambahkan metode LCS sebagai parameter kedua untuk memperbaiki sugesti pencarian.

# RUMUSAN MASALAH

Berdasakan identifikasi yang telah ditetapkan maka muncul masalah yang akan ditemui, sebagai berikut :

1. Bagaimana algoritma *Jaro Winkler Distance* dan LCSdapat membuat sugesti pencarian dengan memperkirakan kesalahan pada kata kunci
2. Bagaimana membuat persentase kemiripan karakter dari LCS sebagai parameter tambahan untuk mendeteksi kesamaan urutan karakter dalam membuat sugesti pencarian

# TUJUAN

Penelitian ini bertujuan mengimplementasikan algoritma *Jaro Winkler Distance* yang ditambahkan dengan persentase kemiripan karakter dari LCS untuk membuat sugesti pencarian*.*

# RUANG LINGKUP

Dalam penelitian yang dilakukan, dibatasi ruang lingkup yang akan dibahas, sebagai berikut :

1. Sistem yang akan dibangun menggunakan platform web.
2. Sistem yang dibuat adalah menampilkan sugesti pencarian yang memiliki kemungkinan ada kesalahan pengetikan pada kata kuncinya
3. Data set yang digunakan merupakan data film.

# KONTRIBUSI PENELITIAN

Konstribusi pada penelitian ini adalah pada sektor information retrieval menggunakan Algoritma Jaro Winkler Distance dan LCS untuk membuat sugesti pencarian.

# TINJAUAN PUSTAKA

(Zhang et al., 2017) melakukan penelitian yang berjudul Research on String Similarity Algorithm based on Levenshtein Distance. Pada penelitian ini disebutkan bahwa Algoritma dari Levenshtein Distance masih kurang cukup pada aspek akurasi sehingga digabungkan dengan LCS dan LCCS.

(Yulianingsih 2017) melakukan penelitian yang berjudul Implementasi Algoritma Jaro-Winkler Dan Levenstein Distance Dalam Pencarian Data Pada Database dengan kesimpulan Algoritma Jaro-Winkler memiliki kompleksitas waktu quadratic runtime complexity yang sangat efektif pada string pendek dan dapat memperoleh hasil lebih cepat dari algoritma Levenshtein Distance.

(Rochmawati & Kusumaningrum, 2016) melakukan penelitian berjudul Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks. Hasil dari penelitian ini menunjukan bahwa pada eksperimennya *Jaro Winkler Distance* mendapat *mean average precision* yang memuaskan sebesar 0.87.

(Novantara, 2018) melakukan penelitian berjudul Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi. Hasil dari penelitian ini menunjukan bahwa pada eksperimennya *Jaro Winkler Distance* dapat mendeteksi kesamaan string pada dokumen.

(Wang et al. 2017) melakukan penelitian berjudul Efficient Approximate Entity Matching Using Jaro-Winkler Distance dikatakan bahwa Jaro Winkler Distance memiliki kinerja yang baik pada pencocokan string.

(Friendly 2017) pada penelitiannya yang berjudul Perbaikan Metode Jaro–Winkler Distance Untuk Approximate String Search Menggunakan Data Terindeks Aplikasi Multi User dikatakan bahwa Jaro Winkler Distance memiliki tingkat kesuksesan yang tinggi dalam mendeteksi kesalahan kata serta dalam mengetahui kedekatan data.

(Pikies & Ali, 2019) melakukan penelitian berjudul String Similarity Algorithms For A Ticket Classification System dikatakan bahwa kelemahan utama dari kesamaan LCS adalah ketidakpekaannya terhadap sebuah konteks yang bisa menjadi keuntungan jika terjadi kesalahan ejaan manusia pada umumnya akan tetap mendapatkan nilai yang sama untuk menghitung kemiripan string.

(Tinaliah & Elizabeth, 2018) melakukan penelitian berjudul Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro- Winkler Distance dan Metode Latent Semantic Analysis dikatakan bahwa hasil pendeteksian plagiarisme dokumen menggunakan metode Jaro-Winkler Distance memberikan hasil yang lebih baik daripada metode Latent Semantic Analysis, yaitu jika data yang dibandingkan sama persis maka akan menghasilkan nilai plagiat sebesar 100%.

(Tannga et al., 2017) melakukan penelitian berjudul Analisis Perbandingan Algoritma Levenshtein Distance Dan Jaro Winkler Untuk Aplikasi Deteksi Plagiarisme Dokumen Teks dikatakan bahwa hasil analisis perbandingan yang didapat untuk rata-rata similarity dari algoritma Jaro Winkler yaitu sebesar 80.92 %, sedangkan untuk algoritma Levensthein Distance rata-rata nilai similarity-nya yaitu sebesar 49.43%.

(Reza Fauzan, Joni Riadi, 2018) melakukan penelitian berjudul Perbandingan Metode Perhitungan Kemiripan Kata dikatakan bahwa pendeteksian kemiripan teks menjadi tugas yang sulit dilakukan oleh manusia karena banyak dan besarnya teks untuk dibandingkan serta strukturnya yang tidak konsisten dan kompleks.

(Syaiful 2017) melakukan penelitian berjudul Perbandingan Algoritma Jaro-Winkler Distance Dengan Algoritma Rabin-Karp Dalam Mendeteksi Kasus Plagiarism Pada Dokumen dikatakan bahwa segi waktu pemrosesan Algoritma Jaro-Winkler Distance lebih cepat daripada Algoritma Rabin-Karp dan proses stemming pada Algoritma Jaro-Winkler distance, sistem belum mampu bekerja dengan baik karena kurangnya kata-kata berimbuhan yang dimasukkan kedalam sistem.

(Leonardo & Hansun, 2017) melakukan penelitian berjudul Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winkler Distance Algorithms dikatakan bahwa Jaro Winkler memiliki akurasi yang bagus pada perbandingan string pendek dan memiliki kompleksitas *runtime* quadrat yang baik pada string pendek dan mampu bekerja cepat.

(Ulya, Adji, Adhipta, & Mada, 2015) melakukan penelitian berjudul Analisis Pengukuran Perubahan Website Menggunakan Algoritme Jaro-Winkler Distance Sebagai Acuan Pendeteksi Website Defacement dikatakan bahwa penelitiannya menggunakan Jaro Winkler menghasilkan nilai threshold perubahan halaman website sebagai dasar untuk menentukan kriteria web defacement dengan tingkat akurasi hingga 93,3% dari seluruh data pengujian.

# Landasan Teori

Adapun sistem ini menggunakan berbagai teori-teori sebagai pustaka untuk memperkaya proses penelitian. Teori yang digunakan tersebut diantaranya:

* 1. Metode String Matching

Metode string matching adalah sebuah algoritma yang digunakan dalam pencocokkan suatu pola kata tertentu terhadap suatu kata atau kalimat yang lain. Metode string matching sendiri dapat dilakukan dengan beberapa cara tertentu (Rossaria & Susilo, 2015), antara lain Jaro Winkler Distance.

Pengertian string menurut Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST) adalah susunan dari karakter- karakter (angka, alfabet atau karakter yang lain) dan biasanya direpresentasikan sebagai struktur data array. String dapat berupa kata, frase, atau kalimat (Ernawati, Johar, & Setiawan, 2019).

Pencocokan string merupakan bagian pentingdari sebuah proses pencarian string (string searching). Hasil dari pencarian sebuah string pada data tergantung dari teknik atau cara pencocokan string yang digunakan. Pencocokan string (string matching) menurut Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST), diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter string di dalam string lain atau bagian dari isi teks (Ernawati et al., 2019).

* 1. Algoritma Jaro Winkler Distance

Jaro-Winkler distance adalah merupakan varian dari Jaro distance metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua string. Metode ini dibuat oleh Matthew A. Jaro yang dinamakan Jaro Distance yang kemudian dikembangkan dan dimodifikasi oleh William E. Winkler dan Thibaudeau (Leonardo & Hansun, 2017). Semakin tinggi Jaro-Winkler distance untuk dua string, semakin mirip dengan string tersebut. Jaro-Winkler distance terbaik dan cocok untuk digunakan dalam perbandingan string singkat. Skor normalnya 0 menandakan tidak ada kesamaan, dan 1 adalah sama persis (Kurniawati, 2010).

Dasar dari algoritma ini memiliki tiga bagian:

(1) Menghitung panjang string.

(2) Menemukan jumlah karakter yang sama di dalam dua string.

(3) Menemukan jumlah transposisi.

Pada algoritma Jaro digunakan Persamaan 1 untuk menghitung jarak (dj) antara dua string yaitu *S1* dan *S2*, dimana :

*c*: jumlah karakter yang sama persis;

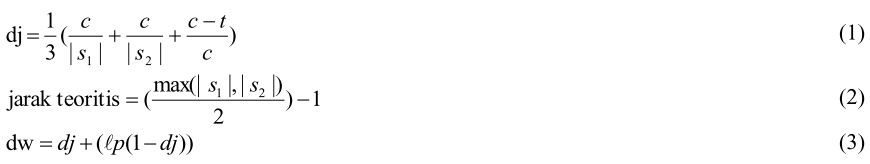
| *S1* | : panjang string 1;

| *S2* | : panjang string 2;

t : jumlah transposisi.

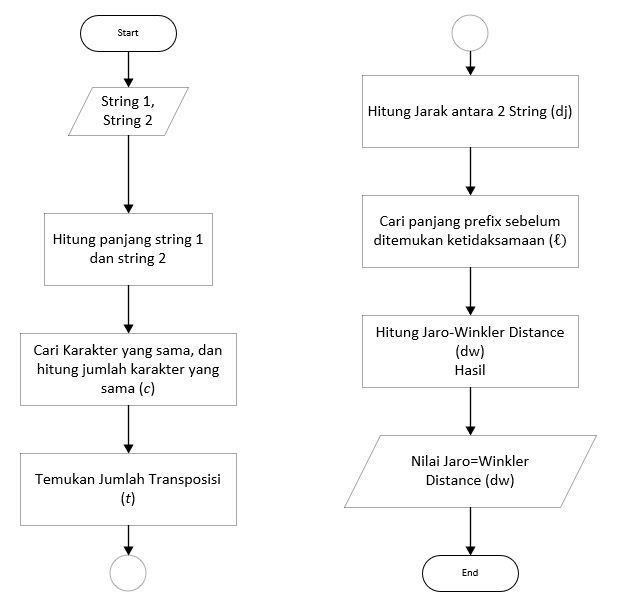
Jarak teoritis dua buah karakter yang dianggap sama dapat dikatakan benar jika tidak melebihi batas seperti yang tercantum pada Persamaan 2.

Jaro Winkler Distance menggunakan prefix scale (p) yang memberikan tingkat penilaian yang lebih dan prefix length (ℓ) yang menyatakan panjang awalan yaitu panjang karakter yang sama dengan string yang dibandingkan sampai ditemukan ketidaksamaan, jumlah karakter yang sama sebelum ditemukan ketidaksamaan adalah maksimal 4. Bila string *S1* dan *S2* yang dibandingkan, maka Jaro Winkler Distance-nya ditentukan dengan Persamaan 3, dimana

dj: Jaro Distance untuk string dan ;

ℓ: panjang prefiks umum di awal string (Panjang karakter yang sama sebelum ditemukan ketidaksamaan max 4);

p: konstanta scaling factor (p=0,1) (Rochmawati & Kusumaningrum, 2016).



Gambar 1. Algoritma Jaro Winkler Distance

Jika dimisalkan “Piratse” adalah string 1 dan “Pirates” adalah string 2, maka panjang S1 dan S2 adalah 7 karena kedua string memiliki panjang yang sama, kemudian dicari jumlah karakter yang sama pada string 1 dan string 2. Karena susunan karakter pada “Piratse” dan “Pirates” terdiri dari karakter yang sama yaitu P-I-R-A-T-E-S maka jumlah karakter yang sama (*c*) adalah 7 karena tidak ada perbedaan karakter susunan dari kedua string yang dimasukan. Kemudian  
karakter yang tertukar hanya ada pada “s” dan “e”, maka transposisi pada string 1 dan string 2 hanya 1 (t = 1).

Maka nilai Jaro Distance (dj) adalah:

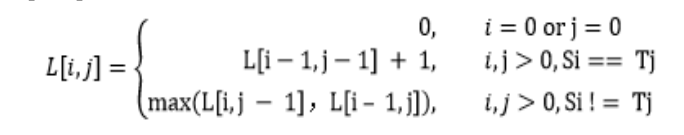
dj =

Karena perbedaan string ditemukan pada string ke 5 maka panjang prefix length (ℓ) jatuh pada angka maksimalnya yaitu 4 sehinggka nilai jaro winklernya adalah :

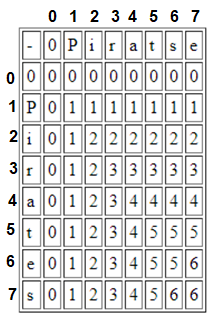
dw = ) = 0.9714

Nilai dw  untuk string “Piratse” dan “Pirates” adalah 0.9714 yang berarti string “Piratse” dan “Pirates” memiliki kemiripan yang cukup tinggi karena hampir mecapai nilai kemiripan maksimum untuk Jaro Winkler Distance yaitu 1.

* 1. Longest Common Subsequence

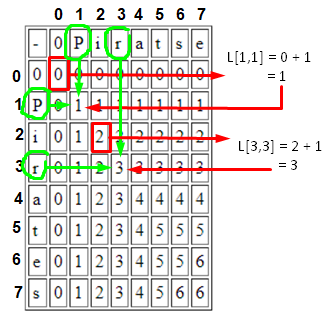
 Longest Common Subsequence (LCS) adalah metode untuk mencari karakter sama yang dari 2 buah kata, metode ini juga bisa digunakan digunakan untuk mencari kemiripan karakter pada string (Zhang et al., 2017). Cara kerja LCS adalah dengan cara membuat matriks L[i + 1][j + 1], untuk inisialisasi nilai pada indeks i = 0 dan j = 0 adalah karakter kosong kemudian indeks selanjutnya untuk i adalah karakter yang ada pada string 1 dan pada j adalah karakter-karakter yang ada pada string 2 kemudian dalam fungsi rekufsif nilai akhir dari LCS bisa didapatkan dengan notasi :

Dimana L adalah nilai LCS dan i,j adalah indeks dari string 1 dan string 2. Jika string 1 (S) adalah “Pirates” dan string 2 (T) adalah “Piratse” dapat digambarkan dengan matriks Gambar 2. Karena pada standarnya indeks ke 0 dari i dan j adalah karakter kosong seperti pada Gambar 2, ketika fungsi rekursif menunjuk pada indeks i = 0 atau j = 0 maka nilai L untuk L[0,j] atau L[i,0] adalah 0.



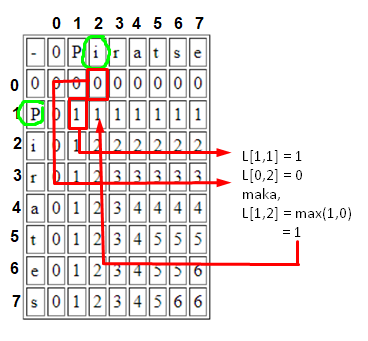
Gambar 2 Contoh Matriks LCS

Pada notasi baris ke 2 nilai untuk L[i ,j], jika i dan j bukan 0 dan karakter Si sama dengan karakter Tj maka L[i , j] akan memiliki nilai dari L[i – 1,j - 1] + 1 contohnya bisa dilihat pada Gambar 3. Dimana pada L[1,1] bertemu S1 dan T1 yang keduanya adalah sama-sama huruf “P” atau S1 == T1 maka nilai L[1,1] adalah L[0,0] yaitu 0 ditambah dengan 1, jadi L[1,1] = 1. Bisa dilihat juga contoh lain pada L[3,3], bertemu S3 dan T3 yang keduanya adalah huruf “r” maka nilai L[3,3] adalah L[2,2] yaitu 2 ditambah dengan 1, jadi L[3,3] = 3.



Gambar 3 Contoh Matriks LCS 2

Pada notasi baris ke 3 nilai untuk L[i ,j], jika i dan j bukan 0 dan karakter Si  tidak sama dengan karakter Tj maka L[i , j] akan memiliki nilai yang paling besar dari L[i ,j - 1] dan L[i - 1 ,j] contohnya bisa dilihat pada Gambar 4. Dimana pada L[1,2] bertemu S1 huruf “P” dan T2 huruf “i” yang tidak sama hurufnya atau S1 tidak sama dengan T2 (S1 != T2) maka nilai L[1,2] adalah nilai terbesar dari L[1,1] yaitu 1 dengan L[0,2] yaitu 0 memiliki nilai terbesar 1, jadi L[1,2] = 1.



Gambar 4 Contoh Matriks LCS 3

Dari hasil perhitungan LCS pada kata “Piratse” dan “Pirates” mendapatkan nilai L = 6. Kemudian L akan dijadikan nilai persentase kemiripian (P) dengan panjang L dibagi dengan string terpangjang dari S dan T dan dinyatakan rumus sebagai berikut:

*P = Lp* / *max* ( S , T ) \* 100

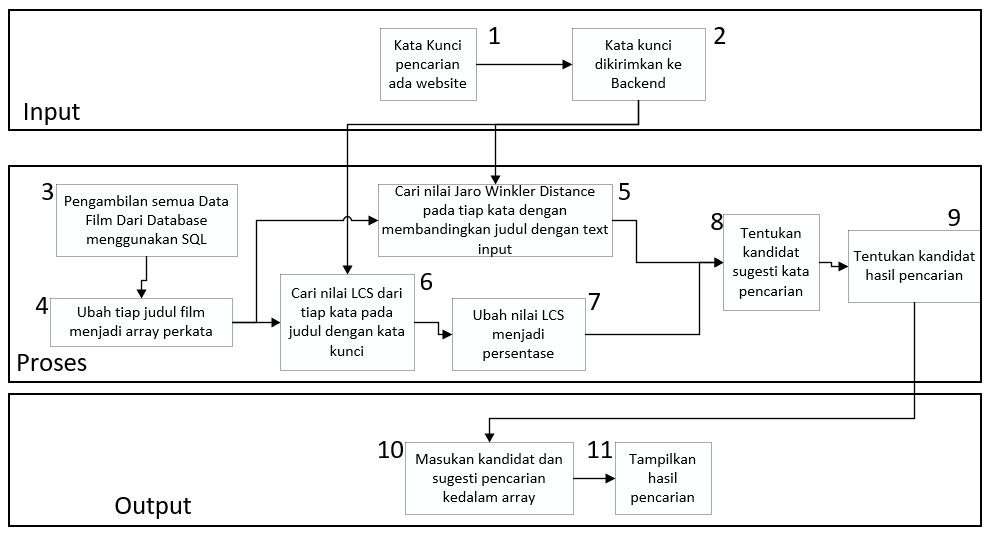
*P =* Persentase kemiripian karakter

*Lp* = Panjang dari kesamaan karakter yang ditemukan LCS

Max(S,T) = Nilai terpanjang dari dua string yang dibandingkan

Karena nilai L = 6 dan Panjang kedua string adalah 7 maka nilai maksimum dari 7 dan 7 adalah 7, maka : *P* = 6 / 7 \* 100 = 85.71%

# Metode Penelitian

Rancangan pengujian dari penelitian yang akan dibuat akan digambarkan dengan block diagram yang ada dibawah kemudian tiap poinnya akan dijelaskan pada bagian bawah gambar dan akan disertakan juga beberapa kasus pengujian.

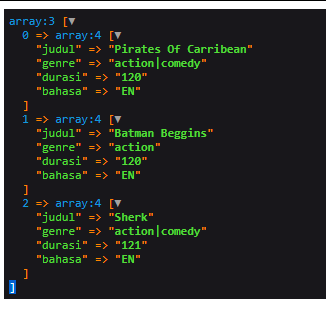
Gambar 5 Block Diagram Penelitian

1. Pada bagian ini akan dimasukan kata kunci pencarian film yang diinginkan pada text box. Misalkan contoh kata kuncinya adalah “Piratse”.



Gambar 6 Ilustrasi Input Kata Kunci

1. Kata kunci dikirim ke program Back End dari website agar kata kunci nantinya bisa dipakai untuk pembanding string saat melakukan pencarian.
2. Pada proses ini program akan mengambil data film yang ada pada database menggunakan SQL “SELECT” kemudian data akan ditampung menggukan array agar bisa dilakukan perulangan untuk mencari data. Isi data juga akan dijadikan array yang berisi data-data film contohnya : Judul film, durasi dalam menit, bahasa yang digunakan, dan genre.



Gambar 7 Ilustrasi Array hasil pengambilan data

1. Kemudian index judul film pada data yang sudah diambil akan dijadikan array untuk setiap katanya misalkan pada index data ke 0 dengan judul “Pirates of the Caribbean” akan dipecah menjadi array menggunakan fungsi explode() pada PHP sehingga judul film menjadi seperti gambar berikut :



Gambar 8 Judul Film yang telah menjadi array

1. Kemudian setiap kata akan dicari nilai Jaro Winkler Distance-nya dimisalkan dengan array judul pada index ke 0 yaitu kata “Pirates”. Kata “Pirates” akan dibandingkan dengan kata kunci yang dicari yaitu “Piratse”.  
   Jika S1 adalah “Pirates” dan S2 adalah “Piratse” sesuai dengan rumus yang sudah dijelaskan sebelumnya, maka :

*c* = 7 (karena tidak ada huruf yang berbeda)  
S1 dan S2 = 7 (Panjang string)  
Karakter yang tertukar hanya ada pada “s” dan “e”, maka t = 1.

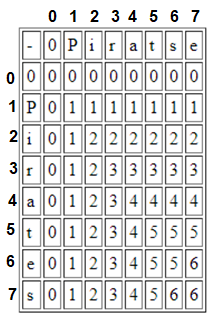
Maka nilai Jaro Distance adalah:

dj =

Karena perbedaan string ditemukan pada string ke 5 maka panjang prefix length (ℓ) jatuh pada angka maksimalnya yaitu 4 sehinggka nilai jaro winklernya adalah :

dw = ) = 0.9714

1. Pada langkah ini kata kunci akan dibandingkan dengan array judul menggunakan LCS dengan perhitungan yang sudah diajabarkan sebelumnya namun akan buat perhitungan nilai LCS LCS pada kata “Piratse” dan “Pirates” menggunakan ilustrasi dari matriks dibawah :



Gambar 9 Ilustrasi Perhitungan LCS

Setelah perhitungan dilakukan dengan menggunakan LCS pada kata “Piratse” dan “Pirates” mendapatkan nilai L terakhir adalah 6.

1. Kemudian nilai L yang didapatkan pada langkah ke 5 akan dijadikan presentase kemiripa karakter. Karen nilai L = 6 dan Panjang kedua string adalah 7, maka :

*P* = 6 / 7 \* 100 = 85.71%

1. Pada langkah ini akan dilihat apakah kata “Pirates” pada judul dapat dijadikan sugesti pencarian atau tidak. Karena threshold yang diinginkan pada penelitian ini adalah 0.75 untuk Jaro Winkler dan 50% untuk LCS maka dengan nilai dw = 0.9714 dan P = 85.71% yang didapat kata “Pirates” cocok untuk dijadikan sugesti pencarian akibat kesalahan ketik
2. Kemudian pada langkah ini akan dilihat apakah ada kata pada judul yang memenuhi threshold yang diinginkan pada penelitian ini yaitu 0.75 untuk Jaro Winkler dan 50% untuk LCS. Maka dengan nilai dw = 0.9714 dan P = 85.71% yang didapat kata “Pirates” pada judul, index data film ini memenuhi syarat untuk ditampilkan.
3. Kandidat pencarian dan sugesti pencarian yang memenuhi threshold yang dinginkan akan dimasukan kedalam array hasil pencarian yang nantinya akan dijadikan nilai kembali dari program back end
4. Pada langkah ini hasil pencarian dengan kata kunci “Piratse” sudah didapatkan dan akan ditampilkan pada interface website berupa table list dengan isi data film yang mirip dengan kata kunci yang dicari.

# DAFTAR PUSTAKA

Ernawati, E., Johar, A., & Setiawan, S. (2019). Implementasi Metode String Matching Untuk Pencarian Berita Utama Pada Portal Berita Berbasis Android (Studi Kasus: Harian Rakyat Bengkulu). *Pseudocode*, *6*(1), 77–82. https://doi.org/10.33369/pseudocode.6.1.77-82

Fitrianti, U., Ula, M., Informatika, T., Informasi, S., Malikussaleh, U., Utara, A., … Husna, A. A. (n.d.). Implementasi Algoritma Levenshtein Distance Dan Algoritma Knuth Morris Pratt Pada Aplikasi Asmaul Husna Berbasis Android. Lhoukseumawe: *Jurnal Sistem Informasi*, 33.

Kurniawati, A. (2010). Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia. *Proceeding, Seminar Ilmiah Nasional Komputer Dan Sistem Intelijen KOMMIT 2008, Depok, Indonesia*.

Leonardo, B., & Hansun, S. (2017). Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 5, pp. 462–471. https://doi.org/10.11591/ijeecs.v5.i2.pp462-471

Novantara, P. (2018). Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi. *Buffer Informatika*, Vol. 3. https://doi.org/10.25134/buffer.v3i2.960

Pikies, M., & Ali, J. (2019). String similarity algorithms for a ticket classification system. *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 36–41. https://doi.org/10.1109/codit.2019.8820497

Rahim, R., Nurarif, S., Ramadhan, M., Aisyah, S., & Purba, W. (2017). Comparison Searching Process of Linear, Binary and Interpolation Algorithm. *Journal of Physics: Conference Series*, *930*(1). https://doi.org/10.1088/1742-6596/930/1/012007

Reza Fauzan, Joni Riadi, F. S. (2018). *Perbandingan Metode Perhitungan Kemiripan*. *5662*(November), 1–6.

Rochmawati, Y., & Kusumaningrum, R. (2016). Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks. *Jurnal Buana Informatika*, *7*(2), 125–134. https://doi.org/10.24002/jbi.v7i2.491

Rossaria, M., & Susilo, B. (2015). Implementasi Algoritma Pencocokan String Knuth-Morris-Pratt Dalam Aplikasi Pencarian Dokumen Digital Berbasis Android. *Jurnal Rekursif*, *3*(2), 183–195.

Tannga, M. J., Rahman, S., Informatika, T., & Distance, A. L. (2017). *Distance Dan Jaro Winkler Untuk Aplikasi Deteksi Plagiarisme Dokumen Teks*. *4*(1).

Tinaliah, T., & Elizabeth, T. (2018). Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode Jaro-Winkler Distance dan Metode Latent Semantic Analysis. *Jurnal Teknologi Dan Sistem Komputer*, *6*(1), 7. https://doi.org/10.14710/jtsiskom.6.1.2018.7-12

Ulya, N. D., Adji, T. B., Adhipta, D., & Mada, U. G. (2015). *Analisis Pengukuran Perubahan Website Menggunakan Algoritme Jaro-Winkler Distance Sebagai Acuan Pendeteksi Website Defacement*. (September), 54–59.

Utis Sutisna, J. A. (2013). Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritme Damerau Levenshtein. *Jurnal Ilmiah Ilmu Komputer*, Vol. 8.

Zhang, S., Hu, Y., & Bian, G. (2017). Research on string similarity algorithm based on Levenshtein Distance. *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2017*, (1), 2247–2251. https://doi.org/10.1109/IAEAC.2017.8054419

# LAMPIRAN

Lampiran 1 Timeline Penelitian

