SISTEM DETEKSI PLAGIASI MENGGUNAKAN ALGORITME FREQUENCY BASED HASHING-S PADA FILE PDF

SKRIPSI

Oleh
Thamrin Auliya
180210117

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM
TAHUN 2021/ 2022

SISTEM DETEKSI PLAGIASI MENGGUNAKAN ALGORITME FREQUENCY BASED HASHING-S PADA FILE PDF

SKRIPSI

Untuk memenuhi salah satu syarat
memperoleh gelar sarjana

Oleh
Thamrin Auliya
180210117

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM
TAHUN 2021/ 2022

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini penulis:

Nama : Thamrin Auliya

NPM : 180210117

Fakultas : Teknik dan Komputer

Program studi : Teknik Informatika

Menyatakan bahwa **"Skripsi"** yang dibuat dengan judul:

"**SISTEM DETEKSI PLAGIASI MENGGUNAKAN ALGORITME FREQUENCY BASED HASHING-S PADA FILE PDF"**

Ini adalah karya sendiri dan bukan "duplikasi" dari karya orang lain. Sejauh yang penulis tahu dalam teks skripsi ini tidak ada karya ilmiah atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang disebutkan dalam teks ini dan disebutkan dalam sumber referensi kutipan.Apabila terdapat didalam naskah Skripsi ini dapat dibuktikan terdapat unsur unsur PLAGIASI, saya bersedia naskah Sikripsi ini digugurkan dan gelar akademik yang saya peroleh dibatalkan, serta diproses sesuai dengan peraturan perundangundang yang berlaku.
Demikian penyataan ini saya buat dengan sebenarnya tanpa ada paksaan dari siapapun.

Batam, 8 Agustus 2022

Thamrin Auliya
180210117

SISTEM DETEKSI PLAGIASI MENGGUNAKAN ALGORITME FREQUENCY BASED HASHING-S PADAFILE PDF

SKRIPSI

Untuk memenuhi salah satu syarat memperoleh gelar sarjana

Oleh
Thamrin Auliya
180210117

Telah disetujui pembimbing pada tanggalseperti yang tertera dibawah ini

Batam, 8 agustus 2022

Cosmas Eko Suharyanto, S.Kom., M.MSI

Pembimbing

ABSTRAK

Di era komputasi digital sekarang ini, telah terjadi pertumbuhan produksi data digital yang sangat tinggi. Dikutip dari laman forbes.com, ada 2,5 quintillion bytes data yang diproduksi setiap hari. Salah satu data digital tersebut adalah tugas mahasiswa yang dikumpulkan melalui media Google Classroom. Dari data mahasiswa yang dikumpulkan tersebut, banyak tugas yang terindikasi plagiasi dengan tugas mahasiswa yang lainnya. Jika dilakukan anilisis secara manual, maka akan memakan waktu yang lama dan sangat melelahkan. Untuk efisiensi pemanfaatan waktu dan sumber daya, dibutuhkan proses penyaringan yang memiliki kemampuan untuk menghitung tingkat plagiasi dari setiap tugas mahasiswa. Metode Approximate Matching merupakan metode yang paling sering digunakan untuk menemukan kesamaan diantara data yang dibandingkan dengan menetapkan skor kesamaan. Pada penelitian ini algoritme Approximate Matching yang digunakan adalah algoritme Frequency Based Hashing-S. Kelebihan algoritme ini adalah aman terhadap serangan aktif dan memiliki tingkat akurasi 98% untuk format data terkompresi. Aplikasi ini bermanfaat bagi dosen yang akan memeriksa tugas mahasiswa yang banyak. Dengan adanya aplikasi ini, dosen hanya perlu memeriksa tugas mahasiswa yang tidak plagiasi saja sehingga mengurangi jumlah tugas mahasiswa yang akan diperiksa secara manual.

Kata Kunci: Frequency Based Hashing-S, Sistem Plagiasi, Sistm Penyaringan Data

ABSTRACT

In today's digital computing era, there has been a very high growth of digital data production. Quoted from the forbes.com page, there are 2.5 quintillion bytes of data created every day. One of the digital data is student assignments collected through Google Classroom media. From the student data collected, many assignments indicated plagiarism with other student assignments. If the analysis is done manually, it will take a long time and be very tiring. For efficient use of time and resources, a screening process is needed that has the ability to calculate the plagiarism level of each student's assignment. The Approximate Matching method is the method most often used to find similarities between the data being compared by setting a similarity score. In this study, the Approximate Matching algorithm used is the Frequency Based Hashing-S algorithm. The advantages of this algorithm are that it is safe against active attacks and has a 98% accuracy rate for compressed data formats. This application is useful for lecturers who will check a lot of student assignments. With this application, lecturers only need to check student assignments that are not plagiarized, thereby reducing the number of student assignments that will be checked manually.

Keywords: Frequency Based Hahsing-S, Plagiarism System, Data Filtering System

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan Kehadirat Tuhan Yang Maha Esa atas kuasa dan limpahan rahmat-Nya sehingga penulis dapat menyelesaikan proposal yang merupakan salah satu syarat untuk menyelesaikan program studi strata satu (S1) pada program studi Teknik Informatika Universitas Putera Batam.

Dengan segala keterbatasan, penulis menyadari pula bahwa proposal ini tidak akan terwujud tanpa bantuan, bimbingan, dan dorongan dari berbagai pihak. Untuk itu dengan segala kerendahan hati, penulis menyampaikan ucapan terimakasih kepada:

1. Ibu Dr. Nur Elfi Husda, S.Kom., M.SI. selaku Rektor Universitas Putera Batam.
2. Bapak Welly Sugianto, S.T., M.M. selaku Dekan Fakultas Teknik dan Komputer.
3. Ketua program studi teknik informatika Bapak Andi Maslan, S.T.,M.SI
4. Bapak Cosmas Suharyanto, S.Kom., M.Kom, selaku pembimbing skripsi pada program Stusi Teknik Informatika Universitas Putera Batam.
5. Kepada orang tua tercinta atas curahan kasih sayang, doa, nasihat, serta pesan yang disampaikan kepada penulis sehingga penulis tetap memiliki semangat juang dalam penyelesaian proposal ini.
6. Serta semua pihak yang tidak dapat saya dapat penulis sebutkan satu persatu yang telah membantu dalam penyusunan skripsi ini. yang telah membantu penyelesaian skripsi ini yang tidak dapat penulis sebutkan satu persatu

Batam, 8 Agustus 2022

Thamrin Auliya

DAFTAR ISI

DAPTAR TABEL

DAPTAR GAMBAR

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Di era digital sekarang ini, telah terjadi lonjakan produksi data digital yang sangat tinggi. Berdasarkan data dari laman forbes.com dalam survei pada tahun 2018, terdapat 2.5 quintillion bytes data yang diproduksi setiap hari dan produksi data digital tersebut akan terus bertambah setiap tahun (Marr, 2018). Dari quintillion data tersebut, sebagian besar merupakan data berformat PDF. Berdasarkan data dari PDF Association pada April 2016, telah diproduksi: (Association, 2018)

- 2,2 miliar file PDF di web
- 20 miliar file PDF di Dropbox
- Airbus, Boeing, dan Departemen Kehakiman AS masing masing memiliki lebih dari 1 miliar PDF
- 2 miliar PDF dibuka setiap tahun di outlook.com
- 73 juta file PDF baru disimpan setiap hari di Google Drive dan Email
- 60% lampiran non-gambar adalah PDF di Outlook Exchange Enterprise.

Hal ini menunjukkan bahwa format *file* PDF sangat penting dan sangat dibutuhkan. Alasan yang paling penting mengapa banyak orang

menggunakan PDF dibandingkan dengan format lain adalah memiliki fleksibilitas yang tinggi. Salah satu pemanfaatan PDF adalah digunakan sebagai format *file* pada tugas mahasiswa. Dengan menggunakan PDF, *file* tersebut tidak akan berubah formatnya walapun di buka dari perangkat yang berbeda.

Dizaman pandemi COVID-19 sekarang ini, banyak kampus yang menyelenggarakan proses belajar mengajar online atau daring menggunkan platform Google Classroom. Tugas mahasiswa berformat PDF akan dikumpulkan pada platform Google Classroom. Namun setelah tugas mahasiswa dikumpulkan dan diperiksa oleh dosen, ternyata banyak tugas yang plagiasi. Dimana, jika tugas mahasiswa tersebut diperiksa secara manual akan membutuhkan waktu yang lama dan melelahkan. Maka dibutuhkan sebuah sistem yang dapat menyaring tugas mahasiswa yang plagiasi dan tidak plagiasi. Sehingga dosen hanya perlu memeriksa tugas siswa yang plagiasi saja.

Teknik yang biasanya digunakan untuk proses penyaringan adalah teknik Approximate Matching. Approximate Matching merupakan sebuah teknik yang digunakan untuk menemukan kesamaan di antara beberapa file berdasarkan skor kesamaan. Beberapa algoritme yang biasa diguanakan adalah Sd-Hash dan Ssdeep. Namun algoitme ini telah terbukti rentan terhadap serangan aktif (Chang, et al., 2015) (Roussev,

2011). Sehingga dibutuhkan algoritme baru yang aman terhadap serangan aktif dan memiliki akurasi yang tinggi untuk menghitung nilai kesamaan diantara beberapa *file*.

Algoritme Frequency Based Hashing-S atau FBHash merupakan algortime baru yang dipublikasikan pada tahun 2019. Pada algoritme FBHash, setiap *byte* dari *chunk* berkontribusi pada skor akhir dan pengaruhnya terhadap skor akhir tergantung pada pentingnya suatu dokumen (Chang, et al., 2019). Untuk membuat skor kesamaan benar benar rendah atau mendekati nol hampir setiap *chunk* harus di modifikasi. Secara keamanan, algoritme FbHash lebih baik dari algoritme SdHash dan Ssdeep, oleh sebab itu penulis memilih untuk menggunakan algoritme FbHash ini.

Algoritme FbHash dibagi menjadi dua versi, yaitu versi FbHash-B untuk mengukur kesamaan ditingkat *byte* seperti format text dan FbHash-S untuk mengukur *syntactic matching* dan menggunakan informasi internal dokumen untuk mengukur kesamaan. Algoritme ini digunakan untuk format *file* terkompresi seperti PDF. Karena object penelitian dari penulis adalah *file* yang terkompresi dalam format PDF, maka penulis menggunakan algoritme FbHash-S.

Berdasarkan penelitian sebelumnya dari Chang, et al., 2019 yang berjudul “FbHash: A New Similarity Hashing Scheme for Digital

Forensics" menyatakan bahwa skema FbHash aman terhadap serangan aktif dan mendeteksi kesamaan dengan akurasi 98% dan memiliki tingkat akurasi 50% lebihi tinggi dari skema lain untuk format data yang terkompresi. Berdasarkan masalah-masalah yang telah diuraikan diatas maka peneliti mencoba melakukan penelitian yang diberi judul : **SISTEM DETEKSI PLAGIASI MENGGUNAKAN ALGORITME FREQUENCY BASED HASHING-S PADA FILE PDF.**

1.2. Identifikasi Masalah

Adapun identifikasi masalah sebagai berikut :

1. Memeriksa tugas mahasiswa secara manual memerlukan waktu yang lama dan melelahkan
2. Algoritme Approximate Matching yang ada sekarang ini rentan terhadap serangan aktif
3. Dibutuhkan algoritme baru yang aman terhadap serangan aktif dan memiliki tingkat akurasi yang tinggi

1.3. Batasan Masalah

Untuk membatasi ruang lingkup penelitian, maka dirumuskan Batasan masalah sebagai berikut:

1. Data yang digunakan adalah file PDF yang didalamnya hanya terdapat teks dan angka, tidak mengandung tabel, diagram

dan gambar ataupun rumus matematika. Hal ini dilakukan untuk memudahkan dalam mengidentifikasi teks pada *file* tersebut.

2. Penelitian ini hanya akan menampilkan nama data beserta tingkat plagiasinya, bukan melakukan penyaringan secara langsung dengan membedakan *file* yang plagiasi dan tidak plagiasi.

1.4. Rumusan Masalah

Berdasarkan latar belakang di atas, maka dirumuskan masalah sebagai berikut :

1. Bagaimana perancangan system deteksi plagiagiasi menggunakan algoritma frequency based hashing-S pada file PDF?
2. Bagaimanakah implementasi algoritme Frequency Based Hashing-S pada sistem deteksi plagiasi untuk menghitung nilai skor kesamaan tugas mahasiswa berformat PDF?
3. Bagaimanakah akurasi perhitungan nilai skor kesamaan dengan mengimplementasikan algoritme Frequency Based Hashing-S pada file PDF?
4. Bagaimanakah integritas data *file* PDF dengan mengimplementasikan algoritme Frequency Based Hashing-S

pada sistem deteksi plagiasi?

1.5. Tujuan Penelitian

Manfaat dari penelitian ini diharapkan dapat menjadi dasar untuk pemeriksaan dokumen dengan format PDF yang membutuhkan metode penyortiran untuk memilah dokumen yang akan diperiksa secara manual sesuai dengan nilai skor kesamaan atau skor plagiasinya. Sehingga dokumen dengan tingkat plagiasi tinggi atau nilai skor kesamaan tinggi tidak perlu diperiksa lagi untuk mengurangi waktu pemeriksaan dokumen secara manual. Dokumen dalam jumlah banyak dapat disaring menjadi dokumen dalam jumlah kecil yang paling bermanfaat tanpa mengurangi tingkat akurasi kesamaan dan dapat menjamin keamanan data.

1.6. Manfaat Penelitian

Manfaat dari penelitian ini diharapkan dapat menjadi dasar untuk pemeriksaan dokumen dengan format PDF yang membutuhkan metode penyortiran untuk memilah dokumen yang akan diperiksa secara manual sesuai dengan nilai skor kesamaan atau skor plagiasinya. Sehingga dokumen dengan tingkat plagiasi tinggi atau nilai skor kesamaan tinggi tidak perlu diperiksa lagi untuk mengurangi waktu pemeriksaan dokumen secara manual. Dokumen dalam jumlah banyak dapat disaring menjadi dokumen dalam jumlah kecil yang

paling bermanfaat tanpa mengurangi tingkat akurasi kesamaan dan dapat menjamin keamanan data.

BAB II
KAJIAN PUSTAKA

Kajian pustaka berisi tentang usaha penulis untuk menggali berbagai informasi berkaitan dengan penelitian yang bersumber dari jurnal ilmiah, laporan penelitian, artikel ilmiah, jurnal survei dan sumber lainnya. Kajian pustaka ini juga untuk menjelaskan tentang variabel-variabel yang menjadi topik dalam penelitian.

2.1 Dasar Teori

2.2.1. Kriptografi

Kriptografi adalah teknik untuk mencapai kerahasiaan pesan (Muhammed & Varol, 2019). Kriptografi secara bahasa memiliki arti tulisan rahasia, diambil dari bahasa yunani yaitu *cryptós* (rahasia) dan *gráphein* (tulisan). Proses melakukan penulisan pesan rahasia dan memecahkan pesan rahasia dinamakan kriptologi yang diperlakukan sebagai bidang studi yang terpisah. Kriptografi memiliki banyak kegunaan seperti enkripsi, dekripsi, otentikasi, tanda tangan digital, hashing, dan sebagainya (Rao, et al., 2019).

2.2.2. *Hashing*

Hashing adalah teknik yang menerima pesan panjang variabel sebagai Input dan menghasilkan panjang tetap dan *string* yang unik (Rao, et al., 2019). Pada mekanisme *hashing*, fungsi *hash* digunakan untuk

menghasilkan nilai *hash*. Nilai *hash* akhir pada skema *hashing* disebut *digest.* Pada algoritma Frequency Based Hashing-S, nilai *digest* akhir digunakan untuk menghitung nilai Cosine Similarity yang merupakan nilai tingkat kesamaan/plagiasi.

2.2.3. Approximate Matching

Approximate Matching merupakan salah satu algoritma *integrity* yang digunakan untuk mengurangi jumlah data yang harus diperiksa oleh investigator secara manual dengan menemukan kesamaan (Lillis, et al., 2018). Teknik Approximate Matching dapat dikategorikan ke dalam tiga kategori berikut: Bytewise Matching, Syntactic Matching dan Semantic Matching.

Bytewise matching dilakukan dengan mengukur kemiripan objek digital pada level *byte* tanpa mempertimbangkan struktur internal objek data. Sintactic matching dilakukan dengan mengukur kemiripan berdasarkan pada struktur internal objek data. Sementara semantic matching mengukur kesamaan berdasarkan atribut kontekstual dari objek digital.

2.2.4. Optical Character Recognition

Optical Character Recognition (OCR) adalah proses yang memungkinkan sistem tanpa campur tangan manusia mengidentifikasi skrip atau abjad yang ditulis ke dalam komunikasi verbal pengguna (Sahu

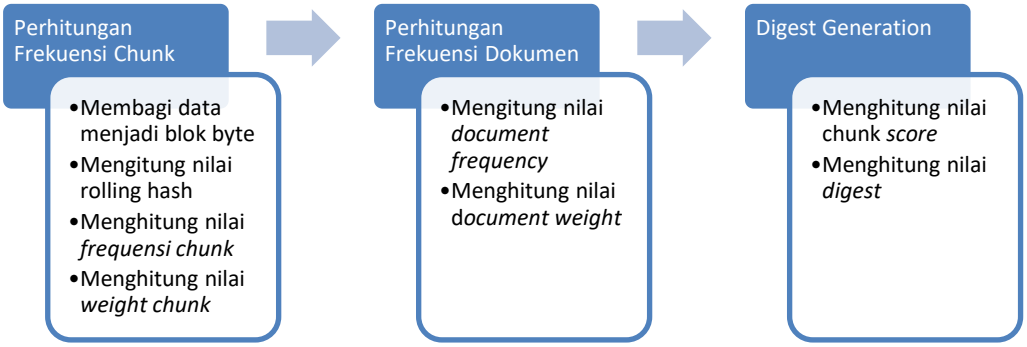
& Sonkusare, 2017). Optical Character Recognition atau OCR digunakan untuk mengekstract teks yang berada dalam *file* PDF. Karena Bahasa yang digunakan adalah Python, maka untuk proses OCR dilakukan menggunakan *library* PyPDF2. Kekurangan utama menggunakan *library* ini adalah skema pengkodean. Dokumen PDF menggunakan penyandian termasuk UTF-8, ASCII, Unicode, dll. Jadi, mengonversi PDF ke teks dapat mengakibatkan hilangnya data karena skema penyandian tersebut.

2.2.5. Cosine Similarity

Cosine similarity merupakan salah satu metode yang dapat digunakan untuk menghitung tingkat kesamaan antara 2 buah objek yang mempunyai kemiripan kata, dihitung menggunakan membandingkan nilai vektornya (Resta, et al., 2021). Dalam penelitian ini, cosine similarity digunakan untuk menghitung nilai similarity dengan membandingkan digest vector D1 (digest 1) dan digest vector D2 (digest 2).

2.2.6. Frequency Based Hashing

Algoritme Frequency Based Hashing atau FbHash mengadopsi Term Frequency Inverse Document Frequency (TF-IDF) untuk menemukan dokumen yang memiliki kesamaan. Cara kerja algoritme ini dibagi menjadi 3 langkah yang ditunjukkan pada Gambar 2.1 (Chang, et al., 2019)



Gambar 2. 1 Tahapan Algoritme Frequency Based Hashing-S

2.2.5.1. Perhitungan Frekuensi *Chunk*

Pada langkah ini, pertama membagi dokumen menjadi blok *byte* tertentu. Setiap blok disebut sebagai *chunk*. Tujuannya adalah untuk menghitung berapa kali setiap *chunk* muncul dalam dokumen yang diberikan, yaitu menghitung frekuensi *chunk*.

1. Misalkan D = $B_0^D, B_1^D, B_2^D, \ldots, B_{l-1}^D$ menjadi dokumen dengan panjang *l byte*, di mana B_i^D menunjukkan *byte* i^{th} dari dokumen *D*. Sebuah *chunk* adalah urutan *k byte* berurutan dari D, di mana:

   $ch_0^D = B_0^D, B_1^D, B_2^D, \ldots, B_{k-2}^D, B_{k-1}^D$

   (2.1)

   $ch_1^D = B_1^D, B_2^D, B_3^D, \ldots, B_{k-1}^D, B_k^D$

   (2.2)

   $ch_2^D = B_2^D, B_3^D, B_4^D, \ldots, B_k^D, B_{k+1}^D$

(2.3)

$ch_i^D = B_i^D, B_{i+1}^D, B_{i+2}^D, \ldots, B_{i+k-2}^D, B_{i+k-1}^D$

(2.4)

$ch_{i-k}^D = B_{l-k}^D, B_{l-k+1}^D, B_{l-k+2}^D, \ldots, B_{l-2}^D, B_{l-1}^D$

(2.5)

2. Untuk menghitung frekuensi dari setiap *chunk* yang teridentifikasi dalam dokumen, teknik *Rolling hash* digunakan. *Rolling hash* adalah fungsi *hash* non-kriptografik yang memungkinkan komputasi *hash* yang cepat untuk setiap *chunk* yang berurutan.

   Rolling hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ modulus n

   (2.6)

3. Setelah nilai *rolling hash* dari sebuah *chunk* dihitung, frekuensi setiap *chunk* akan dihitung dengan berapa kali nilai *rolling hash* muncul. Nilai *rolling hash* di simpan di dalam tabel hash. Untuk menjamin bahwa setiap *chunk* unik mendapatkan nilai *rolling hash* unik, yaitu tidak ada nilai yang sama, nilai *n* diambil sebagai bilangan prima yang lebih besar dari

   2^{56} (karena, $256 * 256^{k-1} = 2^{56}$). (2.7)

4. Berdasarkan frekuensi *chunk*, bobot *chunk* akan dihitung untuk setiap *chunk,* menggunakan rumus berikut:

$$\text{Ch\_wght}_{\text{ch}} = 1 + \log_{10}(\text{chf}_{\text{ch}}^{\text{d}}) \quad (2.8)$$

Jadi, semakin tinggi frekuensi *chunk*, semakin tinggi bobot *chunk* dan sebaliknya.

2.2.5.2. Perhitungan Frekuensi Dokumen

Frekuensi dokumen dari sebuah *chunk* adalah jumlah dokumen yang berisi *chunk* tersebut. Untuk menghitung frekuensi dokumen, nilai N ditetapkan 1000. Frekuensi dokumen dari *chunk* ch disebut sebagai df_{ch}. Frekuensi dokumen dihitung dengan langkah sebagai berikut:

1. Identifikasi *chunk* setiap dokumen dalam dataset. Kemudian menghitung nilai *rolling hash* dari setiap *chunk*. Setelah itu membuat tabel *hash* di mana indeks tabel *hash* menunjukkan nilai *Rolling hash* dari *chunk* dan nilai tabel *hash* menunjukkan frekuensi dokumen dari *chunk* yang sesuai. Setiap *chunk* unik dari setiap dokumen akan meningkatkan nilai tabel *hash* yang di indeks oleh *Rolling hash* sebesar 1.

2. Berdasarkan frekuensi dokumen, bobot dokumen akan dihitung untuk setiap *chunk* yang menunjukkan keinformatifan atau keunikan *chunk*. Di lambangkan sebagai $\text{doc\_wght}_{\text{ch}}$ dan $\text{doc\_wght}_{\text{ch}}$ dihitung sebagai berikut:

Jika $\mathrm{df}_{\mathrm{ch}} > 0$: maka $\mathrm{d}oc\_\mathrm{wght}_{\mathrm{ch}} = \log_{10}(1000/\mathrm{df}_{\mathrm{ch}})$

(2.9)

Sebaliknya: $\mathrm{doc\_wght}_{\mathrm{ch}} = 1$

(2.10)

2.2.5.3. Perhitungan Nilai Digest

Untuk menghitung nilai *digest,* dilakukan beberapa langkah seperti berikut:

1. Setelah memiliki nilai bobot *chunk* dan bobot dokumen dari setiap *chunk* dalam dokumen *D*, skor *chunk* (dilambangkan sebagai $w_{ch_i}^{d}$) kemudian dihitung sebagai berikut:

   $\mathrm{w}_{\mathrm{ch_i}}^{\mathrm{d}} = \mathrm{ch}_{\mathrm{wght}_{\mathrm{ch_i}}^{\mathrm{d}}} * \mathrm{doc\_wght}_{\mathrm{ch_i}}$

   (2.11)

   Skor *chunk* ini akan digunakan untuk menghitung kesamaan antara dua dokumen.

2. Digest FbHash akhir dari dokumen D dapat direpresentasikan sebagai n elemen panjang vektor di mana indeks vektor mewakili nilai $\mathrm{w}_{\mathrm{ch_i}}^{\mathrm{d}}$.

2.2 Variabel (Indikator Masalah)

2.2.1 Pemeriksaan Tugas Siswa Secara Manual Kurang Efektif

Kepercayaan diri adalah salah satu sifat manusia yang unik dan berharga. Setiap orang memiliki tingkat kepercayaan diri yang berbeda-beda. Efikasi Diri (Self Efficacy) adalah salah satu aspek pengetahuan tentang diri yang mempengaruhi kehidupan manusia sehari-hari (Meydiansyah, 2021). Efikasi diri memiliki keterkaitan dengan perilaku menyontek. Perilaku mencontek merupakan salah satu dampak kurangnya percaya diri. Menyontek merupakan sebuah tindakan tidak jujur atau tidak fair dalam rangka memenangkan atau meraih keuntungan.

Pada penelitian yang dilakukan oleh Wati dkk (Wati, et al., 2012) pada siswa kelas X SMA Negeri 52 Jakarta Utara tahun ajaran 2010/2011, dari 153 siswa didapatkan hasil 5,2% selalu menyontek, 37,9% sering menyontek, 52,6% jarang menyontek dan 0,7% tidak pernah menyontek. Melakukan perilaku mencontek berarti bahwa tugas yang dikumpul akan sama atau persis sama dengan tugas siswa yang lainnya. Jika dilakukan pemeriksaan semua tugas siswa secara manual akan melelahkan dan kurang efisien karena dari semua tugas siswa tersebut mungkin terdapat tugas siswa yang plagiasi. Dengan melakukan penyaringan tugas yang tidak plagiasi diantara semua tugas siswa, maka akan mengurangi tugas

yang harus diperiksa secara manual. Sehingga akan mengurangi waktu yang dibutuhkan untuk mengoreksi atau memeriksa tugas siswa.

2.2.2 Metode Pencocokan Fungsi Hash Tradisional Memiliki Keterbatasan

Proses pemfilteran yang digunakan dalam mengekstraksi data biasanya menggunakan Fast Hashing Based Algorithms. Dimana, *file* besar dilewatkan melalui fungsi *hash* untuk menghasilkan keluaran *hash* yang disebut sidik jari digital atau *fingerprint*. Sidik jari dari *file* datatest kemudian dicocokkan dengan kumpulan data referensi dari dataset untuk mengekstrak *file* yang tidak berhubungan. Proses pemfilteran dapat dilakukan dengan *Black Listing* atau *White Listing*. *Black Listing* merupakan proses penyaringan data dengan mencocokkan *file* datatest dengan kumpulan dataset. *File* yang dihasilkan setelah proses ini adalah *file* yang perlu diperiksa secara manual atau tidak plagiasi. Di sisi lain, *White Listing* merupakan proses pemfilteran dengan mencocokkan *file* datatest dengan kumpulan *file* dataset. Berkas yang lolos proses ini tidak perlu diperiksa secara manual atau plagiasi.

Pencocokan berbasis fungsi *hash* tradisional (kriptografik) mengalami keterbatasan bahwa jenis pemfilteran ini hanya menunjukkan salinan persis dari *file* lain. Hal ini disebabkan oleh fakta bahwa bahkan sedikit perubahan dalam konten *file* menghasilkan keluaran *hash* yang

sama sekali tidak terkait dan tampak acak sedangkan persyaratan dalam praktiknya skenario sering menemukan *file* serupa.

Approximate Matching adalah teknik umum untuk menemukan kesamaan di antara *file* yang diberikan, biasanya dengan menetapkan 'skor kesamaan'. Teknik ini digunakan saat ini oleh penyelidik forensik digital.

2.2.3 Algoritme Approximate Matching Rentan Terhadap Serangan Aktif

2.2.3.1 Dcfldd

Pendekatan Approximate Matching pertama untuk forensik digital diusulkan oleh Nicolas Harbour pada tahun 2002 yang disebut dcfldd. dcfldd adalah skema *hashing* berbasis blok (Harbour, 2002). Dalam skema ini, setiap *file* dibagi menjadi blok ukuran tetap dan keluaran *hash* dihasilkan untuk setiap blok. Intisari terakhir adalah rangkaian dari semua *hash* blok.

2.2.3.2 Ssdeep

Perbaikan atas dcfldd diusulkan oleh Kornblum dan diberi nama "Context Triggered Pieceewise Hashing" (CTPH). Skema CTPH didasarkan pada algoritma deteksi email yang disebut spamsum, yang diusulkan oleh Andrew Tridgell. Alih-alih mencirikan blok ukuran tetap, CTPH membagi data dalam blok ukuran variabel dan kemudian setiap blok di-*hash* menggunakan fungsi *hash* (nonkriptografik) yang disebut

hash FNV (Kornblum, 2006). Skema ini terbukti dapat mendeteksi *file* serupa dengan lebih akurat dibandingkan dengan skema pemblokiran *hashing*. Alat yang mengimplementasikan CTPH dikenal sebagai ssdeep, yang juga merupakan nama yang biasa dirujuk untuk skema hashing itu sendiri. Namun, Breitinger menyajikan analisis menyeluruh tentang ssdeep dan menunjukkan bahwa ssdeep tidak menahan musuh aktif untuk menghindari black listing.

2.2.3.3 SdHash

Roussev dkk. (Roussev, 2010) mengusulkan skema baru yang disebut sdhash pada tahun 2010. Ide utama dalam skema sdhash adalah untuk menghasilkan hash akhir hanya dengan menggunakan fitur dokumen yang tidak mungkin secara statistik. Keamanan terperinci dan analisis implementasi sdhash disajikan oleh Breitinger. Pekerjaan ini mengungkap masalah implementasi dan keamanan dan menunjukkan bahwa dimungkinkan untuk mengalahkan skor kesamaan dengan merusak file tertentu tanpa mengubah perilaku persepsi file ini (misalnya, file gambar terlihat hampir sama meskipun ada gangguan). Klaim sekali lagi diverifikasi oleh Chang yang menghadirkan serangan di mana musuh dapat menyesatkan penyidik dengan beberapa file serupa.

2.2.3.4 BbHash dan Mrsh-v2

Dua skema baru yang dikenal sebagai bbHash dan mrsh-telah diusulkan oleh Breitinger pada tahun 2012. Namun, karena kompleksitas waktu proses yang tinggi, bbHash tidak dapat digunakan secara praktis.

2.2.3.5 MvHash-B

Breitinger mengusulkan skema lain pada tahun 2013 yang disebut mvHash-B. Skema ini bekerja dalam tiga fase: pertama mengompresi data input menggunakan voting mayoritas, kemudian melakukan pengkodean run-length dan terakhir menyimpan sidik jari ke dalam filter Bloom. 'B' di mvHash-B menunjukkan representasi filter bloom dari intisari kesamaan. Dalam hal kinerja, mvHash-B adalah salah satu skema yang paling efisien di antara semua skema yang ada dengan kompleksitas waktu proses terendah dan ukuran ringkasan yang kecil. Analisis menyeluruh mvHash-B disajikan oleh Chang yang mengungkap kelemahan dari skema mvHash-B dan menunjukkan bahwa mvHash-B tidak menahan musuh aktif terhadap blscklist dan juga mengusulkan perbaikan pada desain mvHash-B untuk mengurangi kelemahan tersebut.

2.3 Perangkat Lunak Pendukung

Perangkat lunak pendukung merupakan software yang digunakan dalam membangun sistem. Adapaun beberapa perangkat lunak yang digunakan pada penelitian ini yakni:

2.3.1 XAMPP

XAMPP dikembangkan oleh Apache Friends berupa sebuah paket solusi cross- platform yang dapat diakses secara gratis oleh siapa saja. Paker sosuli cross-platform ini terdiri atas Apache HTTP Server, database MariaDB, dan interpreters sebagai script yang ditulis dalam bahasa pemrograman PHP dan Perl (Bulla, et al., 2017). XAMPP adalah singkatan dari Cross Platform (X), Apache (A), MariaDB (M), PHP (P), dan Perl (P). Ini adalah distribusi Apache yang sederhana dan ringan yang membuatnya sangat mudah bagi pengembang untuk membuat server web lokal untuk tujuan pengujian dan penerapan.

2.3.1.1 PHP: Hypertext Preprocessor(PHP)

PHP (Hypertext PreProcessor) adalah bahasa pemrograman yang bisa digunakan untuk mengolah data dari server agar dapat ditampilkan pada website. PHP (Hypertext Preprocessor) yang merupakan bahasa pemrogramma berbasis web yang memiliki kemampuan untuk memproses data dinamis (Purwasih, et al., 2017). Aplikasi aplikasi yang dibangun oleh PHP pada umumnya akan memberikan hasil pada web browser, tetapi prosesnya secara keseluruhan dijalankan di server. PHP dapat digunakan dengan gratis (free) dan bersifat Open Source. Dari beberapa kelebihan pada bahasa pemograman PHP, hal ini mendukung pengguna dalam memasukkan fitur pengolahan data secara cepat.

2.3.1.2 HTML

HTML adalah singkatan dari Hyper Text Markup Language dan digunakan untuk menggambarkan struktur halaman web. HTML adalah Bahasa standar untuk Markup Language (Sharma, 2018). HTML ini biasanya digunakan untuk mengembangkan halaman Web. Dalam HTML digunakan berbagai tag seperti "heading", "paragraph", "table", dan sebagainya.

2.3.1.3 CSS (Cascading Style Sheet)

CSS merupakan singkatan dari Cascading Style Sheets. CSS adalah bahasa style sheet berbasis web yang digunakan untuk presentasi dokumen web (Ndia, et al., 2019). CSS telah berkembang dari CSS1 ke CSS3. CSS ini merupakan bagian integral dari aplikasi berbasis web yang tujuannya adalah untuk memisahkan konten dari presentasi. Pada dasarnya, bahasa CSS memungkinkan untuk menata halaman web pada tema seperti penggunaan warna, font, dan tata letak.

2.3.1.4 Javascript

JavaScript adalah bahasa untuk mengembangkan situs web dinamis. JavaScript masih merupakan pilihan paling populer untuk pemrograman web saat ini (Loki & Gal, 2018). Karena JavaScript adalah inti dari web interaktif kontemporer dimana setiap orang menggunakan jejaring sosial, layanan pemesanan makanan online, atau internet banking,

menggunakan setidaknya satu mesin yang terpasang di browser mereka. Namun web bukan satu-satunya kasus penggunaan JavaScript, ada skenario komunikasi *machine-to-machine* dengan perangkat tertanam yang menjalankan mesin mandiri.

2.3.1.5 Python

Python adalah bahasa pemrograman berorientasi objek tingkat tinggi yang kuat yang dibuat oleh Guido van Rossum. Python menggunakan bahasa pemrograman yang memiliki sintaks tertata rapi dan alat yang kuat untuk menyelesaikan tugas apa pun. Selain itu python memiliki sintaks yang sangat dekat dengan pemikiran matematika sederhana. Python dipilih sebagai bahasa pemrograman utama untuk mahasiswa baru di sebagian besar universitas terkemuka. Python adalah bahasa yang cocok untuk pembelajaran dan pemrograman dunia nyata (Nitnaware, 2019).

2.3.1.6 MySQL dan phpMyAdmin

MySQL adalah RDBMS yang didistribusikan secara gratis dibawah lisensi GNU General Public License, dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial. Sebagai database server, MySQL memiliki kelebihan dalam query datanya dibandingkan dengan database server lain. Hal ini dibuktikan pada query yang dilakukan oleh single user,

dimana kecepatan query MySQL bisa sepuluh kali lebih cepat dari PostgreSQL dan lima kali lebih cepat dibandingkan Interbase. PhpMyAdmin merupakan sebuah aplikasi / perangkat lunak bebas (opensource) dalam bahasa pemrograman PHP dan digunakan untuk menangani administrasi database MySQL, baik melalui jaringan lokal maupun internet. phpMyAdmin mendukung berbagai operasi MySQL, diantaranya (mengelola basis data, tabel-tabel, bidang (fields), relasi (relations), indeks, pengguna (users), perijinan (permissions), dan lain-lain). Perbedaan phpMyAdmin dengan MySQL terletak pada fungsi. PhpMyAdmin merupakan alat untuk memudahkan dalam mengoperasikan database MySQL, sedangkan MySQL adalah database tempat penyimpanan data, phpmyadmin sendiri digunakan sebagai alat untuk mengolah / mengatur data pada MySQL (Standsyah & Restu, 2017).

2.3.2 Sublime

Sublime text adalah text editor berbasis Python, sebuah text editor yang elegan, kaya fitur, cross platform, mudah dan simple yang cukup terkenal dikalangan developer (pengembang) dan desainer. Sublime Text 3 juga digunakan sebagai editor dari Bahasa pemrograman PHP dalam melakukan pengelolaan konten di dalam aplikasi server. Sublime text adalah apliksi editor yang sangat powerful yang dapat meningkatkan

proktivitas dan mengembangkan kualitas kode yang tinggi dan dapat berjalan di berbagai platfrom sistem operasi dengan menggunakan teknologi Phyton API (Pratiwi, 2020).

2.3.3 Penelitian Terdahulu

Berikut ini beberapa penelitian terdahulu mengenai sistem pencocokan yang menjadi referensi dalam penulisan laporan penelitian ini:

1. (Chang, D. et al. 2015) A Collision Attack on Sdhash, In: Proceedings of 10th Intl. Conference on Systematic Approaches to Digital Forensic Engineering, pp. 36e46. Skor kesamaan hashing pada SdHash memiliki kemungkinan untuk diserang karena seluruh konten data tidak berkontribusi pada perhitungan digest akhir, hanya beberapa potongan data yang berkontribusi pada perhitungan digest akhir. Oleh sebab itu dibutuhkan algoritme baru yang aman terhadap serangan aktif.

2. (Roussev 2011) An Evaluation of Forensic Similarity Hashes, Digital Investigation, Volume 8, pp. S34-S41. Ssdeep sangat bergantung pada keberadaan sejumlah besar data yang berkesinambungan, begitu berada di luar zona nyaman, skor

kesamaan dan kemampuan korelasi turun dengan cepat, ditambah lagi tidak adanya hubungan yang mudah dikenali antara tingkat level kesamaan dan skor kesamaan yang membuat penyaringan dan penentuan prioritas bermasalah. Oleh sebab itu dibutuhkan algoritme baru yang aman terhadap serangan aktif

3. (Chang ,D. et al. 2019) FbHash: A New Similarity Hashing Scheme for Digital Forensics, ELSEVIER, Volume 29, pp. S113-S123. Jurnal tersebut memperkenalkan algoritme hashing baru untuk melakukan penyortiran data berdasarkan tingkat kesamaannya. Hasil dari penelitian ini menunjukkan bahwa skema FbHash ini memiliki tingkat akurasi 28% lebih tinggi daripada skema lain untuk format data tidak terkompresi (misalnya, data teks) dan tingkat akurasi 50% lebih tinggi untuk format data terkompresi (mis., Docx dll.). Skema ini juga mampu menghubungkan sebuah fragmen sekecil 1% ke data sumber dengan tingkat deteksi 100% dan mampu mendeteksi kesamaan sekecil 1% antara dua dokumen dengan skor kesamaan yang sesuai. Namun penelitian ini hanya mengimplementasikan skema FbHash pada data berformat text dan docx.

Sementara untuk PDF belum ada penelitian lebih lanjut.

4. Menurut (Rao et al., 2019), dengan judul “Improve the integrity of data using hashing algorithms” ISSN : 2278-3075. penemuan teknologi baru komunikasi informasi sangat cepat dan pengiriman cepatinformasi bahkan untuk jarak jauh menjadi mudah di dunia. Namun, komunikasi online ini telah menimbulkan banyak kerentanan, seperti keandalan dan integritas data. Ada banyak sandi kriptografi, tetapi mereka rentan terhadap berbagai jenis serangan. Buku putih ini berfokus pada berbagai jenis serangan dan bagaimana mereka membahayakan kepercayaan dan integritas data Anda. Algoritma yang berbeda digunakan untuk memastikan keandalan dan integritas data dipertimbangkan, dan studi perbandingan algoritma yang berbeda dan kelemahannya juga dilakukan. Fokusnya adalah pada teknologi hashing.

5. Menurut (Sahu & Sonkusare, 2017), dengan judul “A Study on Optical Character Recognition Techniques” Optical Character Recognition (OCR) merupakan proses yang bisa memungkinkan sistem tanpa campur tangan

manusia. Karakter Optik identifikasi bisa berkembang menjadi individu dari aplikasi pengetahuan yang telah berkembang pesat pada bidang deteksi pola dan kecerdasan buatan. Dalam survei kami, kami mempelajari berbagai teknik OCR. Identifikasi atau klasifikasi karakter optik (OCR) dan Pengenalan Karakter Magnetik (MCR) biasanya digunakan untuk pengenalan pola atau huruf. Secara umum abjadnya adalah dalam berbagai macam gambar piksel serta dapat berupa tulisan tangan atau cap, dari seri, bentuk apa pun atau arah dll. Atau di MCR, huruf dicap dengan tinta magnetik dan mesin belajarmeng kategorikan alfabet berdasarkan medan magnet eksklusif yang dibentuk oleh setiap alfabet.

BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Dalam bab ini akan membahas mengenai tahapan yang digunakan dalam penelitian sistem deteksi plagiasi menggunakan algoritme Frequency Based Hashing-S pada file PDF. Tahapan-tahapannya dapat dilihat pada Gambar 3.1.

Gambar 3. 1 Diagram Alir Penelitian

1. Identifikasi Masalah

Adalah suatu cara untuk sebagai upaya untuk menjelaskan masalah dan membuat masalah dapat di ukur. Penelitian yang dilakukan oleh Chang et al. berjudul “FbHash: A New Similarity Hashing Scheme for

Digital Forensics" menyebutkan bahwa meskipun algoritme yang menggunakan metode *Approximate Matching* yang umum digunakan memiliki tingkat akurasi yang tinggi dalam penyortiran data, namun algoritme tersebut sudah tidak aman lagi terhadap serangan yang aktif. Ini didukung oleh penelitian yang telah dilakukan oleh Chang et al. berjudul "Security analysis of mvhash-b similarity hashing" menunjukkan bahwa penyerang dapat menipu algoritme dengan menyebabkan skor kesamaan mendekati nol bahkan ketika benda sangat mirip. Hal ini memungkinkan karena mvhash mengompresi dokumen input menggunakan run-length encoding (RLE). Ini memberi penyerang kebebasan untuk menurunkan skor kesamaan dengan sedikit modifikasi. Pada penelitian lain yang dilakukan oleh Chang et al. berjudul "A *collision attack* on sdhash similarity *hashing* menunjukkan mekanisme anti-forensik yang memungkinkan seseorang untuk menghasilkan beberapa data yang berbeda sesuai dengan data tertentu dengan 100 % kesamaan, yang dapat membingungkan proses pemfilteran. Kedua serangan tersebut dimungkinkan karena seluruh konten data tidak berkontribusi pada pembuatan *hash* akhir. Hanya beberapa potongan terpilih yang berpartisipasi dalam pembuatan *hash* akhir. Dalam algoritme FbHash ini, setiap *byte* dari dokumen berkontribusi pada skor akhir dan pengaruhnya terhadap skor akhir bergantung pada kepentingannya bagi dokumen.

Karenanya, modifikasi apapun akan memengaruhi skor akhir. Untuk membawa skor kemiripan sangat rendah atau mendekati nol, hampir setiap potongan harus dimodifikasi, yang akan mengubah konten dokumen secara signifikan dan menjadikannya data yang berbeda.

Berdasarkan studi literatur diatas maka permasalahan yang didapat yaitu permasalahan mengenai proses penyortiran data PDF berdasarkan tingkat kesamaannya dengan mengimplementasikan algoritme *Frequency Based Hashing-S*.

2. Perumusan dan Analisis Masalah

Merupukan suatu langkah atau metode yang dilakukan peneliti untuk merancang sebuah system deteksi plagiasi untuk membantu pengecekan tingkat plagiasi.

3. Teknik Pengumpulan Data

Setelah melakukan Perumusan dan analisis masalah langkah berikutnya adalah melakukan pengumpulan data dari hasil observasi guna untuk mendapatkan hasil yang maksimal.

4. Perancangan Aplikasi

Perancangan Aplikasi adalah suatu metode yang dilakukan

peneliti untuk meranang sebuah system deteksi plagiasi yang dapat digunakan untuk pengecekan tingkat plagiasi.

5. Pengujian Aplikasi

Setelah aplikasi dirancang maka langkah selanjutnya adalah melakukan pengujian untuk mendapatkan informasi tentang kekurangan-kekurangan pada system yang dirancang.

6. Implementasi

Implementasi adalah langkah terakhir dari perancangan system dimana peneliti mempublish system tersebut untuk dapat digunakan.

3.2 Perancangan Sistem

Perancangan sistem merupakan sebuah cara yang dilakukan untuk menggambarkan langkah langkah strategis yang diambil dengan memberikan gambaran umum sistem yang dibangun demi tercapainya tujuan penelitian.. Perancangan sistem dibagi menjadi tiga tahap yaitu perancangan data dan UML, perancangan database dan perancangan antar muka.

3.4.1 Perancangan Data dan UML

3.4.1.1 Perancangan Data

Perancangan data merupakan bagian yang akan menjelaskan mengenai data yang digunakan dalam penelitian ini.

1. Data Dataset

Data dataset merupakan semua data yang terdapat di database yang dijadikan data pembanding. Dalam hal ini, data dataset merupakan data semua tugas siswa yang telah dikumpulkan sebelumnya. Tabel 3.1 menampilkan dataset yang digunakan dalam penelitian ini.

Tabel 3. 1 Tabel Dataset

No	Judul File	Teks
1	p2k.unkris.ac.id.pdf	KOTA BATAM ADALAH KOTA TERBESAR DI KEPULAUAN RIAU DAN MERUPAKAN KOTA DENGAN POPULASI TERBESAR KE TIGA DI WILAYAH SUMATRA SETELAH MEDAN DAN PALEMBANG, MENURUT DINAS KEPENDUDUKAN DAN CATATAN SIPIL KOTA BATAM PER APRIL 2012 BANYAK PENDUDUK BATAM MENCAPAI 1.153.860 JIWA. METROPOLITAN BATAM TERDIRI DARI TIGA PULAU, YAITU BATAM, REMPANG DAN GALANG YANG DIHUBUNGKAN OLEH JEMBATAN BARELANG. BATAM

		MERUPAKAN SUATU KOTA DENGAN LETAK SANGAT STRATEGIS. SELAIN BERADA DI JALUR PELAYARAN INTERNASIONAL, KOTA INI MEMILIKI JARAK YANG CUKUP DEKAT DENGAN SINGAPURA DAN MALAYSIA. BATAM MERUPAKAN SALAH SATU KOTA DENGAN PERTUMBUHAN TERPESAT DI INDONESIA. KETIKA DIBANGUN PADA TAHUN 1970-AN AWAL KOTA INI HANYA DIHUNI SEKITAR 6.000 PENDUDUK DAN DALAM TEMPO 40 TAHUN PENDUDUK BATAM BERTUMBUH SAMPAI 158 KALI LIPAT
2	kompas.com.pdf	BATAM ADALAH KOTA TERBESAR DI PROVINSI KEPULAUAN RIAU DAN MERUPAKAN SALAH SATU KOTA INDUSTRI DI INDONESIA YANG PALING MENONJOL. KOTA BATAM SENDIRI BARU DIKEMBANGKAN AWAL TAHUN 1970-AN, NAMUN KINI TELAH MENJADI SALAH SATU KOTA METROPOLIS DI INDONESIA. WILAYAH KOTA BATAM TERDIRI DARI PULAU BATAM, PULAU REMPANG DAN PULAU GALANG DAN PULAU-PULAU KECIL LAINNYA DI KAWASAN SELAT SINGAPURA DAN SELAT MALAKA. PULAU BATAM, REMPANG, DAN GALANG TERKONEKSI OLEH JEMBATAN BARELANG. ASAL-USUL NAMA KOTA BATAM MEMILIKI BERAGAM VERSI. HINGGA KINI, BELUM ADA DATA

		YANG VALID YANG MENJADI ARGUMEN PEMBERIAN NAMA BATAM.
3	kepri.bpk.go.id.pdf	BATAM MERUPAKAN SALAH SATU PULAU YANG BERADA DI ANTARA PERAIRAN SELAT MALAKA DAN SELAT SINGAPURA. TIDAK ADA LITERATUR YANG DAPAT MENJADI RUJUKAN DAN MANA NAMA BATAM ITU DIAMBIL, YANG JELAS PULAU BATAM MERUPAKAN SEBUAH PULAU BESAR DAN 329 PULAU YANG ADA DI WILAYAH KOTA BATAM. SATU-SATUNYA SUMBER YANG DENGAN JELAS MENYEBUTKAN NAMA BATAM DAN MASIH DAPAT DIJUMPAI SAMPAI SAAT MI ADALAH TRAKTAT LONDON (1824). PENDUDUK ASLI KOTA BATAM DIPERKIRAKAN ADALAH ORANG-ORANG MELAYU YANG DIKENAL DENGAN SEBUTAN ORANG SELAT ATAU ORANG LAUT. PENDUDUK INI PALING TIDAK TELAH MENEMPATI WILAYAH ITU SEJAK ZAMAN KERAJAAN TUMASIK (SEKARANG SINGAPURA) DIPENGHUJUNG TAHUN 1300 ATAU AWAL ABAD KE-14. MALAHAN DAN CATATAN LAINNYA, KEMUNGKINAN PULAU BATAM TELAH DIDIAMI OLEH ORANG LAUT SEJAK TAHUN 231 M YANG DI ZAMAN SINGAPURA DISEBUT PULAU UJUNG. PADA MASA JAYANYA KERAJAAN MALAKA, PULAU BATAM BERADA DI BAWAH

		KEKUASAAN LAKSAMANA HANG TUAH. SETELAH MALAKA JATUH, KEKUASAAN ATAS KAWASAN PULAU BATAM DIPEGANG OLEH LAKSAMANA HANG NADIM YANG BERKEDUDUKAN DI BENTAN (SEKARANG P. BINTAN). KETIKA HANG NADIM MENEMUI AJALNYA, PULAU INI BERADA DI BAWAH KEKUASAAN SULTAN JOHOR SAMPAI PADA PERTENGAHAN ABAD KE.18. DENGAN HADIRNYA KERAJAAN DI RIAU LINGGA DAN TERBENTUKNYA JABATAN YANG DIPERTUAN MUDA RIAU, MAKA PULAU BATAM BESERTA PULAU-PULAU LAINNYA BERADA DI BAWAH KEKUASAAN YANG DIPERTUAN MUDA RIAU, SAMPAI BERAKHIRNYA KERAJ AAN MELAYU RIAU PADA TAHUN 1911
4	jdih.batam.go.id.pdf	KOTA BATAM ADALAH SEBUAH KOTA TERBESAR DI PROVINSI KEPULAUAN RIAU, INDONESIA. WILAYAH KOTA BATAM TERDIRI DARI PULAU BATAM, PULAU REMPANG DAN PULAU GALANG DAN PULAU-PULAU KECIL LAINNYA DI KAWASAN SELAT SINGAPURA DAN SELAT MALAKA. PULAU BATAM, REMPANG, DAN GALANG TERKONEKSI OLEH JEMBATAN BARELANG. MENURUT DINAS KEPENDUDUKAN DAN CATATAN SIPIL KOTA BATAM PER 2015, JUMLAH PENDUDUK BATAM MENCAPAI 1.037.187

		JIWA. BATAM MERUPAKAN BAGIAN DARI KAWASAN KHUSUS PERDAGANGAN BEBAS BATAM–BINTAN– KARIMUN (BBK). BATAM MERUPAKAN SALAH SATU KOTA DENGAN LETAK YANG SANGAT STRATEGIS. SELAIN BERADA DI JALUR PELAYARAN INTERNASIONAL, KOTA INI MEMILIKI JARAK YANG SANGAT DEKAT DAN BERBATASAN LANGSUNG DENGAN SINGAPURA DAN MALAYSIA. SEBAGAI KOTA TERENCANA, BATAM MERUPAKAN SALAH SATU KOTA DENGAN PERTUMBUHAN TERPESAT DI INDONESIA. KETIKA DIBANGUN PADA TAHUN 1970-AN OLEH OTORITA BATAM (SAAT INI BERNAMA BP BATAM), KOTA INI HANYA DIHUNI SEKITAR 6.000 PENDUDUK DAN DALAM TEMPO 40 TAHUN PENDUDUK BATAM BERTUMBUH HINGGA 158 KALI LIPAT.
5	bpbatam.go.id.pdf	BATAM ADALAH SALAH SATU PULAU DALAM GUGUSAN KEPULAUAN RIAU DAN MERUPAKAN SEBUAH PULAU DI ANTARA 329 PULAU YANG TERLETAK ANTARA SELAT MALAKA DAN SINGAPURA YANG SECARA KESELURUHAN MEMBENTUK WILAYAH BATAM. LANGKANYA CATATAN TERTULIS TENTANG PULAU INI, DI MANA HANYA ADA SATU LITERATUR YANG MENYEBUT NAMA BATAM, YAITU TRAKTAT LONDON YANG MENGATUR

		PEMBAGIAN WILAYAH KEKUASAAN ANTARA BELANDA DAN INGGRIS. NAMUN, MENURUT PARA PESIAR DARI CHINA, PULAU INI SUDAH DIHUNI SEJAK 231 M KETIKA SINGAPURA MASIH DISEBUT PULAU UJUNG. SEBELUM MENDAPAT PERHATIAN KHUSUS DARI PEMERINTAH PUSAT, BATAM MERUPAKAN SEBUAH PULAU KOSONG BERUPA HUTAN BELANTARA YANG NYARIS TANPA DENYUT KEHIDUPAN. NAMUN, TERDAPAT BEBERAPA KELOMPOK PENDUDUK YANG LEBIH DAHULU MENDIAMI PULAU INI. MEREKA BERPROFESI SEBAGAI PENANGKAP IKAN DAN BERCOCOK TANAM. MEREKA SAMA SEKALI TIDAK BANYAK TERLIBAT DALAM MENGUBAH BENTUK FISIK PULAU INI YANG MERUPAKAN HAMPARAN HUTAN BELANTARA.
6	batam.terkini.id.pdf	BATAM MERUPAKANSALAH SATU KOTA DARI PROVINSI KEPULAUAN RIAU (KEPRI) YANG TERKENAL DI INDONESIA DENGAN KAWASAN INDUSTRINYA. KOTA INI AWALNYA DIKEMBANGKAN SEJAK TAHUN 1970-AN SEBAGAI PUSAT INDUSTRI DAN JALUR TRANSIT LOGISTIK PERDAGANGAN. SEIRING BERJALANNYA WAKTU, KINI BATAM MENJADI SALAH SATU KOTA METROPOLIS MODERN DI INDONESIA DENGAN CIRI MULAI BANYAK GEDUNG-

		GEDUNG TINGGI DAN MOBILISASI MASYARAKAT URBAN. LETAK GEOGRAFIS YANG BERDEKATAN DENGAN SINGAPURA MENJADI SALAH SATU PEMICU PESATNYA KEMAJUAN DI KOTA BATAM. DI SISI LAIN KOTA BATAM JUGA MERUPAKAN SEBUAH PULAU TERBESAR DI DAERAH KEPRI, TETAPI BELUM JELAS DIKETAHUI DARI MANA ASAL USUL LITERATUR SEJARAH MASA LAMPAU DI WAKTU JOHOR DAN RIAU MASIH MERUPAKAN KERAJAAN MELAYU.
7	batam.suara.com.pdf	BATAM MERUPAKAN SALAH SATU KOTA DARI PROVINSI KEPULAUAN RIAU (KEPRI) YANGTERKENAL DI INDONESIA DENGAN KAWASAN INDUSTRINYA. KOTA INI AWALNYA DIKEMBANGKAN SEJAK TAHUN 1970-AN SEBAGAI PUSAT INDUSTRI DAN JALUR TRANSIT LOGISTIK PERDAGANGAN. SEIRING BERJALANNYA WAKTU, KINI BATAM MENJADI SALAH SATU KOTA METROPOLIS MODERN DI INDONESIA DENGAN CIRI MULAI BANYAK GEDUNG- GEDUNG TINGGI DAN MOBILISASI MASYARAKAT URBAN. LETAK GEOGRAFIS YANG BERDEKATAN DENGAN SINGAPURA MENJADI SALAH SATU PEMICU PESATNYA KEMAJUAN DI KOTA BATAM. DI SISI LAIN KOTA BATAM JUGA MERUPAKAN SEBUAH PULAU TERBESAR DI DAERAH KEPRI,

	TETAPI BELUM JELAS DIKETAHUI DARI MANA ASAL USUL LITERATUR SEJARAH MASA LAMPAU DI WAKTU JOHOR DAN RIAU MASIH MERUPAKAN KERAJAAN MELAYU.

2. Data Datatest

Data datatest merupakan data yang digunakan sebagai data yang ingin diuji nilai skor kesamaannya. Dengan mengunggah datatest, akan menjalankan program perhitungan nilai skor kesamaan. Tabel 3.2 memperlihatkan datatest yang digunakan di penelitian ini

Tabel 3. 2 Tabel Datatest

No	**Judul File**	**Teks**
1	batam.terkini.id.pdf	BATAM MERUPAKAN SALAH SATU KOTA DARI PROVINSI KEPULAUAN RIAU (KEPRI) YANG TERKENAL DI INDONESIA DENGAN KAWASAN INDUSTRINYA. KOTA INI AWALNYA DIKEMBANGKAN SEJAK TAHUN 1970- AN SEBAGAI PUSAT INDUSTRI DAN JALUR TRANSIT LOGISTIK PERDAGANGAN. SEIRING BERJALANNYAWAKTU, KINI BATAM MENJADI SALAH SATU KOTA METROPOLIS MODERN DI INDONESIA DENGAN CIRI MULAI BANYAK GEDUNG-GEDUNG TINGGI DAN MOBILISASI MASYARAKAT URBAN. LETAK GEOGRAFIS YANG BERDEKATAN DENGAN SINGAPURA MENJADI SALAH

	SATU PEMICU PESATNYA KEMAJUAN DI KOTA BATAM. DI SISI LAIN KOTA BATAM JUGA MERUPAKAN SEBUAH PULAU TERBESAR DI DAERAH KEPRI, TETAPI BELUM JELAS DIKETAHUI DARI MANA ASAL USUL LITERATUR SEJARAH MASA LAMPAU DI WAKTU JOHOR DAN RIAU MASIH MERUPAKAN KERAJAAN MELAYU.

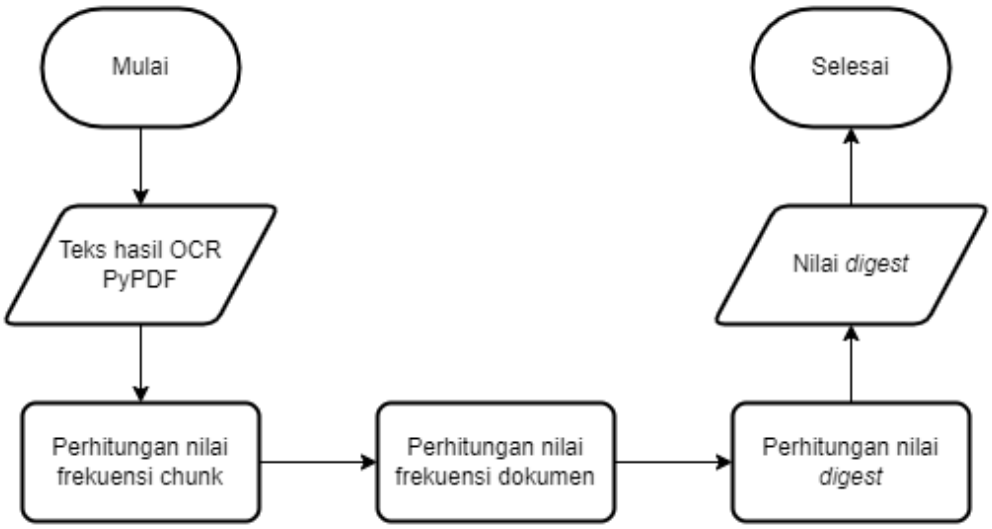
3. Data Pengujian Collision Attack

Data pengujian collision attack merupakan data yang digunakan untuk pengujian collision attack. Tujuan pengujian ini adalah untuk mengetahui apakah dua buah chunk dapat memiliki nilai Rolling hash yang sama. Pengujian ini menggunakan dataset dgn nilai teks “I AM THAMRIN” dan teks dua merupakan nilai random yang terdiri dari huruf besar, huruf kecil, angka dan karakter unik. Menentukan nilai chunk dan rolling Hash dari teks random tersebut

3.4.1.2 Perancangan UML

1. Diagram Alir OCR PyPDF2

OCR PyPDF2 merupakan OCR yang paling umum dan paling sering digunakan sekarang ini untuk mengekstract teks dari *file* pdf. Berikut diagram alir OCR PyPDF2

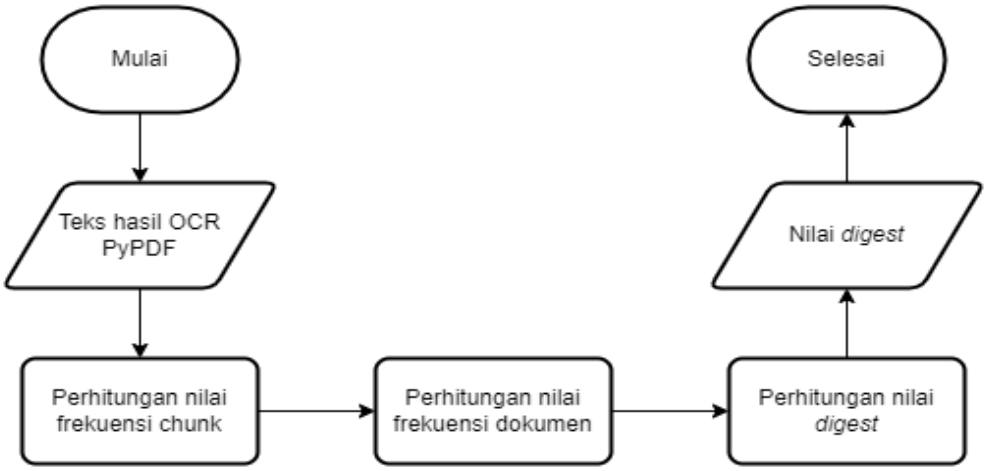


Gambar 3. 2 Diagram alir OCR PyPDF2

Tenaga pendidik mengupload *file* pdf dan judul *file*nya ke sistem. Sistem kemudian akan mengekstrak teks yang berada dalam *file* tersebut menggunakan module PyPDF2 pada Python. Teks tersebut nantinya akan diimplementasikan algoritme Frequency Based Hashing-S untuk menghitung nilai digestnya.

2. Diagram Alir FbHash-S

Proses perancangan algoritme FbHash-S adalah gambaran proses dimana menghitung nilai digest dari masing masing *file*. Berikut diagram alir proses perhitungan nilai digest menggunakan algoritme Frequency Based Hashing-S.

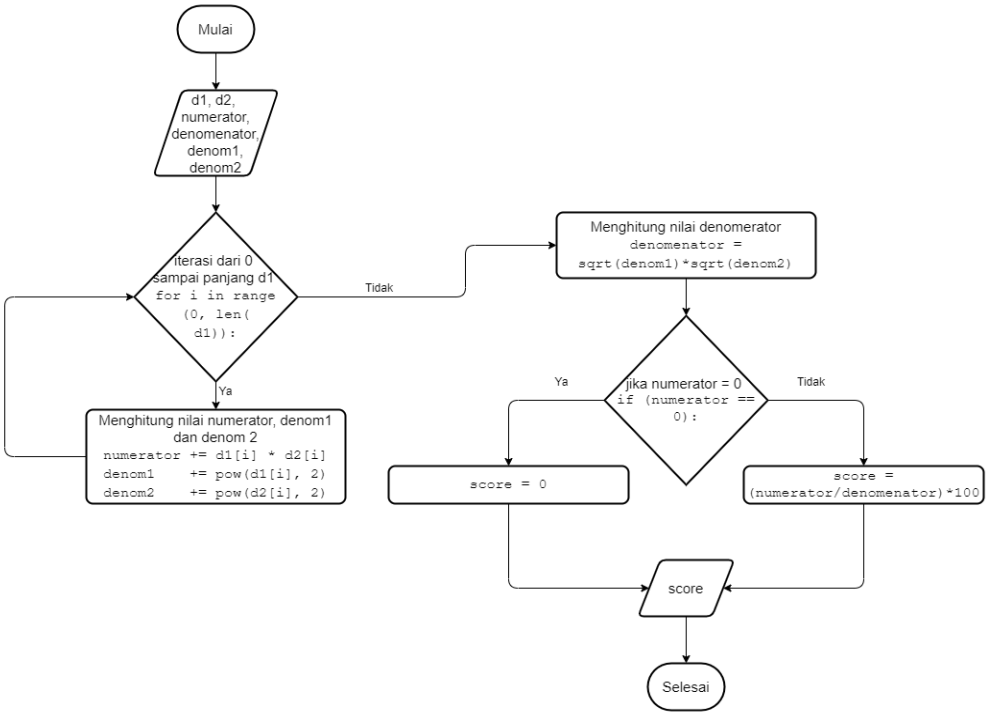


Gambar 3. 3 Diagram alir FbHash-S

Ada tiga tahapan utama pada proses perhitungan nilai digest menggunakan algoritme Frequency Based Hashing-S, yaitu proses perhitungan nilai frekuensi chunk, proses perhitungan nilai frekuensi dokumen dan proses perhitungan nilai digest.

3. Diagram Alir Cosin Similarity

Skor kesamaan digest vector D1 dan digest vector d2 dihitung menggunakan cosin similarity (Chang, et al., 2019). Berikut diagram alir proses perhitungan cosin similarity.



Gambar 3. 4 Diagram alir Cosine Similarity

Input untuk proses perhitungan cosin similarity adalah list d1[], list d2[],numerator, denom1 dan denom2. list d1[] dan list d2 merupakan nilai digest vector digest 1 dan nilai digest vector digest 2, numerator merupakan nilai pembilang pada perhitungan cosin similarity, denomenator merupakan nilai penyebut pada perhitungan cosin similarity, sementara denom1 dan denom2 digunakan untuk menyimpan masing masing nilai pangkat 2 dari digest 1 dan digest 2 yang membantu dalam perhitungan denomenator.

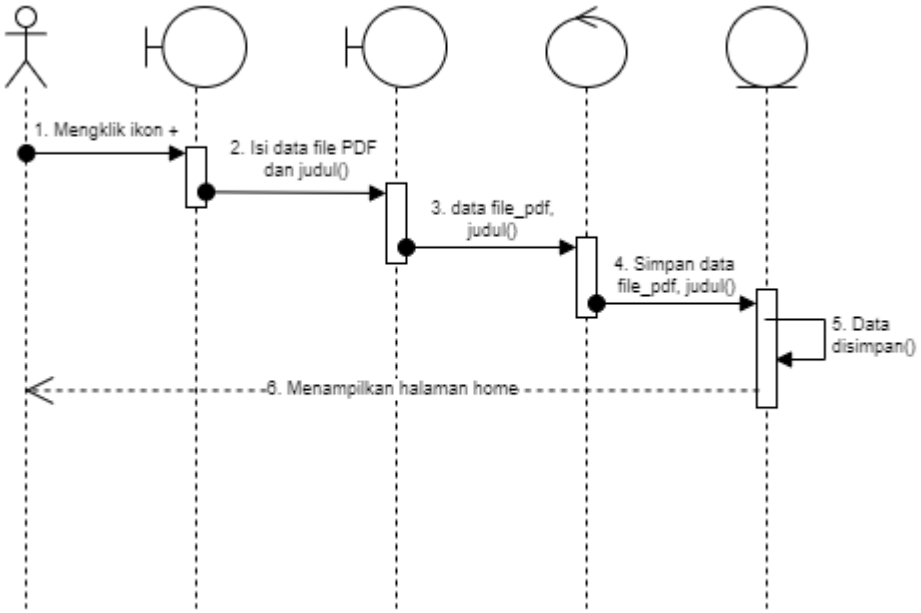
Dilakukan iterasi i sejumlah 0 sampai panjang d1. Jika bernilai benar, menghitung nilai denominator dengan rumus d1[i] * d2[i],

menghitung nilai denom 1 dengan rumus pow(d1[i], 2) serta menghitung nilai denom 2 dengan rumus pow(d2[i], 2). Dilakukan increment nilai numerator, denom1 dan denom2 untuk setiap iterasi. Jika bernilai salah, iterasi akan berhenti dan selanjutnya menghitung nilai pembilang denominator dengan rumus sqrt(denom1) * sqrt(denom2).

Setelah itu terdapat seleksi kondisi dengan syarat numerator bernilai 0. Jika seleksi kondisi benar, maka score ditetapkan sebagai 0. Sementara jika bernilai salah score dihitung dengan rumus (numerator/denomenator)*100. Nilai keluarannya adalah score yang menunjukkan tingkat kesamaannya. Skor kesamaan berkisar antara 0 sampai 100. Dimana 100 menunjukkan file persis sama sedangkan skor 0 menunjukkan tidak ada kesamaan.

4. Sequence Diagram Unggah Dataset

Perancangan proses menambah dataset bertujuan untuk memastikan dokumen dataset yang diunggah dapat disimpan ke dalam basis data.

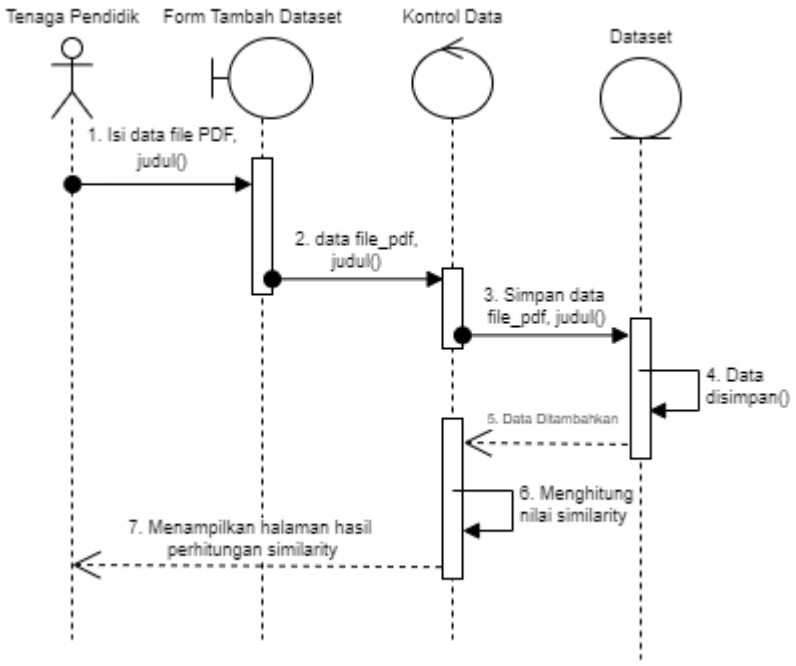


Gambar 3. 5 Sequence diagram unggah dataset

Proses penambahan file pdf dan judul dilakukan oleh tenaga pendidik dengan mengklik ikon + pada halaman home. Tenaga pendidik kemudian mengunggah file pdf yang akan di tambahkan ke database beserta judulnya. Data tersebut ditambahkan ke tabel dataset dan menampilkan halaman home dengan dataset yang sudah bertambah.

5. Sequence Diagram Perhitungan Nilai Similarity

Perancangan proses perhitungan nilai tingkat kesamaan bertujuan untuk memastikan dokumen datatest yang diunggah dapat disimpan kedalam basis data dan dapat menghitung nilai tingkat kesamaan antara dokumen datatest dengan dokumen dataset serta menampilkannya dilayar.



Gambar 3. 6 Sequence diagram perhitungan nilai similarity

Proses penambahan file *pdf* dan judul dilakukan oleh tenaga pendidik dengan mengklik *fil*e pdf yang akan di tambahkan ke database beserta judulnya. Data tersebut ditambahkan ke tabel datatest. Sistem melakukan perhitungan nilai Similarity dan menampilkan halaman hasil perhitungan Similarity.

3.4.2 Perancangan Database

Struktur data dari database dalam penelitian ini dibagi menjadi tiga, yaitu kosin_proses_dataset, kosin_proses_datatest dan juga kosin_proses_tf. Struktur data kosin_proses_dataset dapat dilihat pada tabel 3.3, Struktur data kosin_proses_datatest dapat dilihat pada tabel 3.4

dan struktur data kosin_proses_tf dapat dilihat pada tabel 3.4.

Tabel 3. 3 Tabel Struktur kosin_proses_dataset

No	Tipe Data	Nama Variabel
1	int(11)	id
2	longtext	dataset_judul
3	Varchar(100)	dataset_flie

Tabel 3. 4 Tabel Struktur kosin_proses_datatest

No	Tipe Data	Nama Variabel
1	int(11)	id
2	longtext	datates_judul
3	Varchar(100)	datates_flie

Tabel 3. 5 Tabel Struktur kosin_proses_tf

No	Tipe Data	Nama Variabel
1	Int(11)	Id
2	Varchar(225)	kata
3	Int(11)	frequency
4	Int(11)	dataset_id
5	longtext	dataset_judul

3.4.3 Perancangan Antarmuka

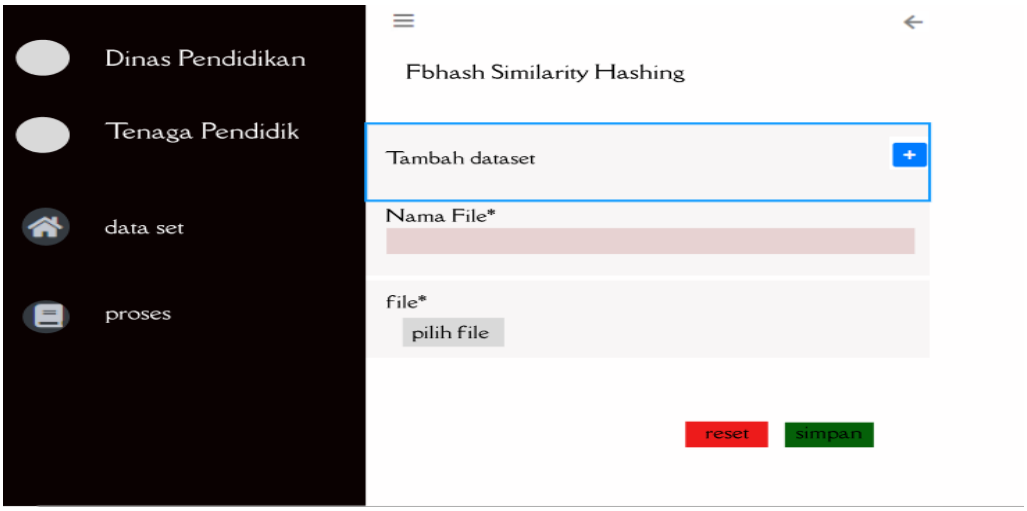
3.4.3.1 Tampilan Home

Tampilan home dapat dilihat pada pada gambar 3.7.

Gambar 3. 7 Tampilan home

3.4.3.2 Tampilan Tambah Dataset

Tampilan tambh dataset dapat dilihat pada pada gambar 3.8



Gambar 3. 8 Tampilan tambah datateset

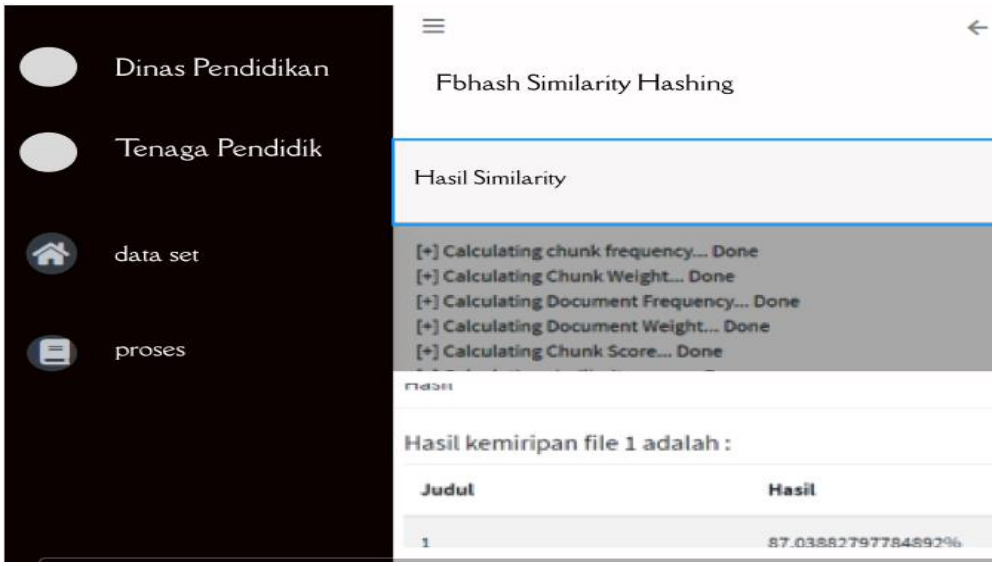
3.4.3.3 Tampilan Tambah Datatest

Tampilan tambah datatest dapat dilihat pada pada gambar 3.9.

Gambar 3. 9 Tampilan tambah datatest

3.4.3.4 Tampilan Hasil Perhitungan Nilai Similarity

Tampilan tambah datatest dapat dilihat pada pada gambar 3.10.



Gambar 3. 10 Tampilan hasil perhitungan nilai similarity

3.3 Perhitungan Manual

Misalkan terdapat dua buah file pdf yang akan dihitung tingkat kesamaannya. File pdf data test merupakan file pdf yang diupload oleh tenaga pendidik untuk dihitung nilai kesamaannya. Sementara data set merupakan file pdf yang berada didalam database.

File pdf data test: 5.pdf

File pdf data set: 2.pdf

3.3.1 Digest 1

Misalkan terdapat dua buah file pdf yang akan dihitung tingkat kesamaannya. File pdf data test merupakan file pdf yang diupload oleh tenaga pendidik untuk dihitung nilai kesamaannya. Sementara data set merupakan file pdf yang berada didalam database.

3.3.1.1 Chunk Frekuensi

3.3.1.1.1 Membagi file manjadi chunk

Perhitungan manual dilakukan terhadap data test 5.pdf dengan teks hasil OCR Pytessetact sebagai berikut:

Datatest = MAMALIA ADA YANG BERENANG

Dari seluruh data teks tersebut kemudian akan dibagi beberapa fragmen file atau chunk dengan panjang 7. Tabel berikut menunjukkan chunk dari data test

ch_i	Chunk	ch_i	Chunk
ch_0	'MAMALIA'	ch_{11}	' Y ANG '
ch_1	'AMALIA '	ch_{12}	'Y ANG '
ch_2	'MALIA A'	ch_{13}	' ANG B'
ch_3	'ALIA AD'	ch_{14}	'ANG BE'
ch_4	'LIA ADA'	ch_{15}	'NG BER'
ch_5	'IA ADA '	ch_{16}	'G BERE'
ch_6	'A ADA Y'	ch_{17}	' BEREN'
ch_7	' ADA Y '	ch_{18}	' BERENA'
ch_8	'ADA Y A'	ch_{19}	'BERENAN'
ch_9	'DA Y AN'	ch_{20}	'ERENANG'
ch_{10}	'A Y ANG'		

3.3.1.1.2 Perhitungan rolling hash

Sebelum dilakukan perhitungan, terlebih dahulu setiap karakter di konversi ke ascii. Berikut nilai ascii dari setiap karakter.

Karakter	spasi	M	A	L	I	D	Y	N	G	B	E	R
Nilai Ascii	32	77	65	76	73	68	89	78	71	66	69	82

$$H = c_1 a^{k-1} + c_2 a^{k-2} + c_3 a^{k-3} + \ldots + c_k a^0 \; modulus \; n$$

$$H_{new} = a * H - c_0 a^k + incoming \; byte \; modulus \; n$$

Dimana: k = 7; a = 255 (konstanta); n = 801385653117583 579 (bilangan prima yang lebih besar dari 2^{56} atau 72 057 594 037 927)

Berikut nilai rolling Hash dari setiap chunk:

ch_i	Chunk	Rolling Hash
ch_0	'MAMALIA'	21240943672735580
ch_1	'AMALIA '	39195471272886637
ch_2	'MALIA A'	60436410849296862
ch_3	'ALIA AD'	78389893886482430
ch_4	'LIA ADA'	99364469906692960
ch_5	'IA ADA '	119505455320065642
ch_6	'A ADA Y'	137411463611115266
ch_7	' ADA Y '	146279979366660043

ch_8	'ADA Y A'	164224802403078243
ch_9	'DA Y AN'	182991078227609946
ch_{10}	'A Y ANG'	200897187399756782
ch_{11}	' Y ANG '	209791427835090619
ch_{12}	'Y ANG '	234296044217719086
ch_{13}	' ANG B'	243164602355243862
ch_{14}	'ANG BE'	261120232796561811
ch_{15}	'NG BER'	282642396870529513
ch_{16}	'G BERE'	302197917577762342
ch_{17}	' BEREN'	311030844217368190
ch_{18}	' BERENA'	319900442689359495
ch_{19}	'BERENAN'	338121358419642348
ch_{20}	'ERENANG'	357181061972551184

Chunk ke-0

Chunk = ‘MAMALIA'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(M)∗ 255^6 + ord(A)∗ 255^5 + ord(M)∗ 255^4 + ord(A)∗ 255^3 +ord(L)∗ 255^2+ord(I)∗ 255^1+ord(A)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)*$255^{k-(i+1)}$
0	77	274941996890625	21170533760578125
1	65	1078203909375	21240617014687500
2	77	4228250625	21240942589985625
3	65	16581375	21240943667775000
4	76	65025	21240943672716900
5	73	255	21240943672735515
6	65	1	21240943672735580
Total			21240943672735580
% n			21240943672735580

Chunk ke-1

Chunk = ‘AMALIA '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)∗ 255^6 + ord(M)∗ 255^5 + ord(A)∗ 255^4 + ord(L)∗ 255^3 +ord(I)∗ 255^2+ord(A)∗ 255^1+ord( )∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)*$255^{k-(i+1)}$
0	65	274941996890625	39112173470626205
1	77	1078203909375	39195195171648080
2	65	4228250625	39195470007938705
3	76	16581375	39195471268123205
4	73	65025	39195471272870030
5	65	255	39195471272886605
6	32	1	39195471272886637
Total			39195471272886637
% n			39195471272886637

Chunk ke-2

Chunk = ‘MALIA_A'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(M)∗ 255^6 + ord(A)∗ 255^5 + ord(L)∗ 255^4 + ord(I)∗ 255^3 +ord(A)∗ 255^2+ord( )∗ 255^1+ord(A)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	77	274941996890625	60366005033464762
1	65	1078203909375	60436088287574137
2	76	4228250625	60436409634621637
3	73	16581375	60436410845062012
4	65	65025	60436410849288637
5	32	255	60436410849296797
6	65	1	60436410849296862
Total			60436410849296862
% n			60436410849296862

Chunk ke-3

Chunk = ‘ALIA AD'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)∗ 255^6 + ord(L)∗ 255^5 + ord(I)∗ 255^4 + ord(A)∗ 255^3 +ord( )∗ 255^2+ord(A)∗ 255^1+ord(D)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	65 76 73 65 32 65 68	274941996890625 1078203909375 4228250625 16581375 65025 255 1	78307640647187487 78389584144299987 78389892806595612 78389893884384987 78389893886465787 78389893886482362 78389893886482430
Total			78389893886482430
% n			78389893886482430

Chunk ke-4

Chunk = 'LIA_ADA'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash (ch_0) = ord(L)$* 255^6$ + ord(I)$* 255^5$ + ord(A)$* 255^4$ + ord( )$* 255^3$ +ord(A)$* 255^2$+ord(D)$* 255^1$+ord(A)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	76 73 65 32 65 68 65	274941996890625 1078203909375 4228250625 16581375 65025 255 1	99285485650169930 99364194535554305 99364469371844930 99364469902448930 99364469906675555 99364469906692895 99364469906692960
Total			99364469906692960
% n			99364469906692960

Chunk ke-5

Chunk = 'IA ADA '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash (ch_0) = ord(I)$* 255^6$ + ord(A)$* 255^5$ + ord( )$* 255^4$ + ord(A)$* 255^3$ +ord(D)$* 255^2$+ord(A)$* 255^1$+ord( )$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	73 65 32 65 68 65 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	119435235679708585 119505318933817960 119505454237837960 119505455315627335 119505455320049035 119505455320065610 119505455320065642
Total			119505455320065642
% n			119505455320065642

Chunk ke-6

Chunk = ‘A ADA Y'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord( )$* 255^5$ + ord(A)$* 255^4$ + ord(D)$* 255^3$ +ord(A)$* 255^2$+ord( )$* 255^1$+ord(Y)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	65 32 65 68 65 32 89	274941996890625 1078203909375 4228250625 16581375 65025 255 1	137376685117956267 137411187643056267 137411462479346892 137411463606880392 137411463611107017 137411463611115177 137411463611115266
Total			137411463611115266
% n			137411463611115266

Chunk ke-7

Chunk = ‘ ADA Y '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )$* 255^6$ + ord(A)$* 255^5$ + ord(D)$* 255^4$ + ord(A)$* 255^3$ +ord( )$* 255^2$+ord(Y)$* 255^1$+ord( )$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 65 68 65 32 89 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	146209607511615266 146279690765724641 146279978286767141 146279979364556516 146279979366637316 146279979366660011 146279979366660043
Total			146279979366660043
% n			146279979366660043

Chunk ke-8

Chunk = ‘ADA_Y_A'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord(D)$* 255^5$ + ord(A)$* 255^4$ + ord( )$* 255^3$ +ord(Y)$* 255^2$+ord( )$* 255^1$+ord(A)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	65 68 65 32 89 32 65	274941996890625 1078203909375 4228250625 16581375 65025 255 1	164151209164550668 164224527030388168 164224801866678793 164224802397282793 164224802403070018 164224802403078178 164224802403078243
Total			164224802403078243
% n			164224802403078243

Chunk ke-9

Chunk = ‘DA Y AN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(D)$* 255^6$ + ord(A)$* 255^5$ + ord( )$* 255^4$ + ord(Y)$* 255^3$ +ord( )$* 255^2$+ord(A)$* 255^1$+ord(N)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0	68	274941996890625	182920858191640743
1	65	1078203909375	182990941445750118
2	32	4228250625	182991076749770118
3	89	16581375	182991078225512493
4	32	65025	182991078227593293
5	65	255	182991078227609868
6	78	1	182991078227609946
Total			182991078227609946
% n			182991078227609946

Chunk ke-10

Chunk = ‘A Y ANG'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord( )$* 255^5$ + ord(Y)$* 255^4$ + ord( )$* 255^3$ +ord(A)$* 255^2$+ord(N)$* 255^1$+ord(G)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	65	274941996890625	200862308025500571
1	32	1078203909375	200896810550600571
2	89	4228250625	200897186864906196
3	32	16581375	200897187395510196
4	65	65025	200897187399736821
5	78	255	200897187399756711
6	71	1	200897187399756782
Total			200897187399756782
% n			200897187399756782

Chunk ke-11

Chunk = ‘ Y ANG '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )$* 255^6$ + ord(Y)$* 255^5$ + ord( )$* 255^4$ + ord(A)$* 255^3$ +ord(N)$* 255^2$+ord(G)$* 255^1$+ord( )$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 89 32 65 78 71 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	209695331300256782 209791291448191157 209791426752211157 209791427830000532 209791427835072482 209791427835072482 209791427835090619
Total			209791427835090619
% n			209791427835090619

Chunk ke-12

Chunk = ‘Y ANG '

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord(Y)} * 255^6 + \text{ord( )} * 255^5 + \text{ord(A)} * 255^4 + \text{ord(N)} * 255^3 + \text{ord(G)} * 255^2 + \text{ord( )} * 255^1 + \text{ord( )} * 255^0$ *modulus n*

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$
0 1 2 3 4 5 6	89 32 65 78 71 32 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	234261265558356244 234295768083456244 234296042919746869 234296044213094119 234296044217710894 234296044217719054 234296044217719086
Total			234296044217719086
% n			234296044217719086

Chunk ke-13

Chunk = ‘ ANG B'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord( )} * 255^6 + \text{ord(A)} * 255^5 + \text{ord(N)} * 255^4 + \text{ord(G)} * 255^3 + \text{ord( )} * 255^2 + \text{ord( )} * 255^1 + \text{ord(B)} * 255^0$ *modulus n*

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 65 78 71 32 32 66	274941996890625 1078203909375 4228250625 16581375 65025 255 1	243094188118219086 243164271372328461 243164601175877211 243164602353154836 243164602355235636 243164602355243796 243164602355243862
Total			243164602355243862
% n			243164602355243862

Chunk ke-14

Chunk = ‘ANG BE'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord(N)$* 255^5$ + ord(G)$* 255^4$ + ord( )$* 255^3$ +ord( )$* 255^2$+ord(B)$* 255^1$+ord(E)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	65 78 71 32 32 66 69	274941996890625 1078203909375 4228250625 16581375 65025 255 1	261035832153134487 261119932058065737 261120232263860112 261120232794464112 261120232796544912 261120232796561742 261120232796561811
Total			261120232796561811
% n			261120232796561811

Chunk ke-15

Chunk = ‘NG__BER'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(N)$* 255^6$ + ord(G)$* 255^5$ + ord( )$* 255^4$ + ord( )$* 255^3$ +ord(B)$* 255^2$+ord(E)$* 255^1$+ord(R)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	78 71 32 32 66 69 82	274941996890625 1078203909375 4228250625 16581375 65025 255 1	282565708554030561 282642261031596186 282642396335616186 282642396866220186 282642396870511836 282642396870529431 282642396870529513
Total			282642396870529513
% n			282642396870529513

Chunk ke-16

Chunk = ‘G BERE'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(G)∗ 255^6 + ord( )∗ 255^5 + ord( )∗ 255^4 + ord(B)∗ 255^3 +ord(E)∗ 255^2+ord(R)∗ 255^1+ord(E)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$
0 1 2 3 4 5 6	71 32 32 66 69 82 69	274941996890625 1078203909375 4228250625 16581375 65025 255 1	302163278649763888 302197781174863888 302197916478883888 302197917573254638 302197917577741363 302197917577762273 302197917577762342
Total			302197917577762342
% n			302197917577762342

Chunk ke-17

Chunk = ‘ BEREN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )∗ 255^6 + ord( )∗ 255^5 + ord(B)∗ 255^4 + ord(E)∗ 255^3 +ord(R)∗ 255^2+ord(E)∗ 255^1+ord(N)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 32 66 69 82 69 78	274941996890625 1078203909375 4228250625 16581375 65025 255 1	310996061478262342 311030564003362342 311030843067903592 311030844212018467 311030844217350517 311030844217368112 311030844217368190
Total			311030844217368190
% n			311030844217368190

Chunk ke-18

Chunk = ‘ BERENA'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )$* 255^6$ + ord(B)$* 255^5$ + ord(E)$* 255^4$ + ord(R)$* 255^3$ +ord(E)$* 255^2$+ord(N)$* 255^1$+ord(A)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	32 66 69 82 69 78 65	274941996890625 1078203909375 4228250625 16581375 65025 255 1	319828988117868190 319900149575886940 319900441325180065 319900442684852815 319900442689339540 319900442689359430 319900442689359495
Total			319900442689359495
% n			319900442689359495

Chunk ke-19

Chunk = ‘BERENAN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(B)$* 255^6$ + ord(E)$* 255^5$ + ord(R)$* 255^4$ + ord(E)$* 255^3$ +ord(N)$* 255^2$+ord(A)$* 255^1$+ord(N)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	66 69 82 69 78 65 78	274941996890625 1078203909375 4228250625 16581375 65025 255 1	338046614484140745 338121010553887620 338121357270438870 338121358414553745 338121358419625695 338121358419642270 338121358419642348
Total			338121358419642348
% n			338121358419642348

Chunk ke-20

Chunk = 'ERENANG'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

$Rolling\ Hash\ (ch_0) = \text{ord(E)} * 255^6 + \text{ord(R)} * 255^5 + \text{ord(E)} * 255^4 + \text{ord(N)} * 255^3 + \text{ord(A)} * 255^2 + \text{ord(N)} * 255^1 + \text{ord(G)} * 255^0$ *modulus n*

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$
0 1 2 3 4 5 6	69 82 69 78 65 78 71	274941996890625 1078203909375 4228250625 16581375 65025 255 1	357092356205095473 357180768925664223 357181060674957348 357181061968304598 357181061972531223 357181061972551113 357181061972551184
Total			357181061972551184
% n			357181061972551184

3.3.1.1.3 Perhitungan chunk frekuensi

Setelah nilai rolling hash dihitung, nilai tersebut harus unik. Dalam hal ini, menggunakan fungsi uniq() pada python. Fungsi tersebut bertujuan agar setiap elemen yang ada didalam fungsi tersebut unik dan tidak ada nilai yang sama. Setelah di implementasikan, hasilnya bisa dilihat pada tabel dibawah.

Karena setiap nilai rolling hash hanya muncul sekali dan tidak terdapat nilai rolling hash yang sama, maka nilai frekuensi untuk setiap ch_i adalah 1.

ch_i	Nilai Rolling Hash	Tabel Chunk Frekuensi	
		Nilai Rolling Hash	chf_{ch}^{d}
ch_0	'MAMALIA'	21240943672735580	1
ch_1	'AMALIA '	39195471272886637	1
ch_2	'MALIA A'	60436410849296862	1
ch_3	'ALIA AD'	78389893886482430	1
ch_4	'LIA ADA'	99364469906692960	1
ch_5	'IA ADA '	119505455320065642	1
ch_6	'A ADA Y'	137411463611115266	1
ch_7	' ADA Y '	146279979366660043	1
ch_8	'ADA Y A'	164224802403078243	1
ch_9	'DA Y AN'	182991078227609946	1
ch_{10}	'A Y ANG'	200897187399756782	1
ch_{11}	' Y ANG '	209791427835090619	1
ch_{12}	'Y ANG '	234296044217719086	1
ch_{13}	' ANG B'	243164602355243862	1
ch_{14}	'ANG BE'	261120232796561811	1
ch_{15}	'NG BER'	282642396870529513	1
ch_{16}	'G BERE'	302197917577762342	1
ch_{17}	' BEREN'	311030844217368190	1
ch_{18}	' BERENA'	319900442689359495	1
ch_{19}	'BERENAN'	338121358419642348	1
ch_{20}	'ERENANG'	357181061972551184	1

3.3.1.1.4 Perhitungan chunk weight

Berdasarkan frekuensi *chunk*, bobot *chunk* akan ditetapkan untuk setiap *chunk*, menggunakan rumus berikut:

chunk-$wght_{ch}$ = 1 + log_{10} (chf_{ch}^{d})

dimana:

chunk-$wght_{ch}$= nilai bobot dari setiap chunk

chf_{ch}^{d} = Chunk frekuensi
Sehingga,

ch_i	Nilai Rolling Hash	chf_{ch}^{d}	$1 + log_{10}(chf_{ch}^{d})$	ch - $wght_{ch_i}^{d}$
ch_0	21240943672735580	1	$1 + log_{10}(1) = 1$	1
ch_1	39195471272886637	1	$1 + log_{10}(1) = 1$	1
ch_2	60436410849296862	1	$1 + log_{10}(1) = 1$	1
ch_3	78389893886482430	1	$1 + log_{10}(1) = 1$	1
ch_4	99364469906692960	1	$1 + log_{10}(1) = 1$	1
ch_5	119505455320065642	1	$1 + log_{10}(1) = 1$	1
ch_6	137411463611115266	1	$1 + log_{10}(1) = 1$	1
ch_7	146279979366660043	1	$1 + log_{10}(1) = 1$	1
ch_8	164224802403078243	1	$1 + log_{10}(1) = 1$	1
ch_9	182991078227609946	1	$1 + log_{10}(1) = 1$	1
ch_{10}	200897187399756782	1	$1 + log_{10}(1) = 1$	1
ch_{11}	209791427835090619	1	$1 + log_{10}(1) = 1$	1
ch_{12}	234296044217719086	1	$1 + log_{10}(1) = 1$	1
ch_{13}	243164602355243862	1	$1 + log_{10}(1) = 1$	1
ch_{14}	261120232796561811	1	$1 + log_{10}(1) = 1$	1
ch_{15}	282642396870529513	1	$1 + log_{10}(1) = 1$	1
ch_{16}	302197917577762342	1	$1 + log_{10}(1) = 1$	1
ch_{17}	311030844217368190	1	$1 + log_{10}(1) = 1$	1
ch_{18}	319900442689359495	1	$1 + log_{10}(1) = 1$	1
ch_{19}	338121358419642348	1	$1 + log_{10}(1) = 1$	1
ch_{20}	357181061972551184	1	$1 + log_{10}(1) = 1$	1

3.3.1.2 Dokumen Frekuensi

3.3.1.2.1 Dokumen Frekuensi

Data test = ‘MAMALIA ADA YANG BERENANG’

Karena data testnya sama, maka perhitungan chunk dan rolling hash sama dengan perhitungan sebelumnya.

ch_i	Chunk	Rolling Hash
ch_0	'Y ANG '	209791427835090619
ch_1	'A Y ANG'	200897187399756782
ch_2	'NG BER'	282642396870529513
ch_3	‘BERENA'	319900442689359495
ch_4	'ADA Y A'	164224802403078243
ch_5	'ERENANG'	357181061972551184
ch_6	'AMALIA '	39195471272886637
ch_7	'DA Y AN'	182991078227609946

ch_8	'G BERE'	302197917577762342
ch_9	'LIA ADA'	99364469906692960
ch_{10}	'A ADA Y'	146279979366660043
ch_{11}	' Y ANG '	234296044217719086
ch_{12}	'BERENAN'	338121358419642348
ch_{13}	'MALIA A'	60436410849296862
ch_{14}	'ALIA AD'	78389893886482430
ch_{15}	' BEREN'	311030844217368190
ch_{16}	'IA ADA '	119505455320065642
ch_{17}	'ANG BE'	261120232796561811
ch_{18}	'MAMALIA'	21240943672735580
ch_{19}	' ANG B'	243164602355243862
ch_{20}	' ADA Y '	137411463611115266

Kemudian, nilai rolling hash dari data test yang dihitung pertama tersebut dibandingkan dengan nilai rolling hash data test baru untuk menghitung nilai dokumen frekensinya.

ch_i	Rolling Hash Pertama	Rolling Hash Baru	df_{ch}
ch_0	21240943672735580	209791427835090619	1
ch_1	39195471272886637	200897187399756782	1
ch_2	60436410849296862	282642396870529513	1
ch_3	78389893886482430	319900442689359495	1
ch_4	99364469906692960	164224802403078243	1
ch_5	119505455320065642	357181061972551184	1
ch_6	137411463611115266	39195471272886637	1
ch_7	146279979366660043	182991078227609946	1
ch_8	164224802403078243	302197917577762342	1
ch_9	182991078227609946	99364469906692960	1
ch_{10}	200897187399756782	146279979366660043	1
ch_{11}	209791427835090619	234296044217719086	1
ch_{12}	234296044217719086	338121358419642348	1
ch_{13}	243164602355243862	60436410849296862	1
ch_{14}	261120232796561811	78389893886482430	1
ch_{15}	282642396870529513	311030844217368190	1
ch_{16}	302197917577762342	119505455320065642	1
ch_{17}	311030844217368190	261120232796561811	1
ch_{18}	319900442689359495	21240943672735580	1
ch_{19}	338121358419642348	243164602355243862	1
ch_{20}	357181061972551184	137411463611115266	1

Karena nilai rolling hash nya sama, maka nilai frekensinya adalah 1. Karena setiap nilai dari rolling hash, memiliki nilai yang sama pada satu dokumen yang lainnya.

3.3.1.2.2 Dokumen Weight

If $df_{ch} > 0$: *doc* - $wght_{ch} = log_{10}\ (1000/df_{ch})$

Else doc - $wght_{ch} = 1$

ch_i	df_{ch})	$log_{10}\ (1000/df_{ch})$	*doc* - $wght_{ch}$
ch_0	1	$log_{10}\ (1000/1) = 3$	3
ch_1	1	$log_{10}\ (1000/1) = 3$	3
ch_2	1	$log_{10}\ (1000/1) = 3$	3
ch_3	1	$log_{10}\ (1000/1) = 3$	3
ch_4	1	$log_{10}\ (1000/1) = 3$	3
ch_5	1	$log_{10}\ (1000/1) = 3$	3
ch_6	1	$log_{10}\ (1000/1) = 3$	3
ch_7	1	$log_{10}\ (1000/1) = 3$	3
ch_8	1	$log_{10}\ (1000/1) = 3$	3
ch_9	1	$log_{10}\ (1000/1) = 3$	3
ch_{10}	1	$log_{10}\ (1000/1) = 3$	3
ch_{11}	1	$log_{10}\ (1000/1) = 3$	3
ch_{12}	1	$log_{10}\ (1000/1) = 3$	3
ch_{13}	1	$log_{10}\ (1000/1) = 3$	3
ch_{14}	1	$log_{10}\ (1000/1) = 3$	3
ch_{15}	1	$log_{10}\ (1000/1) = 3$	3
ch_{16}	1	$log_{10}\ (1000/1) = 3$	3
ch_{17}	1	$log_{10}\ (1000/1) = 3$	3
ch_{18}	1	$log_{10}\ (1000/1) = 3$	3
ch_{19}	1	$log_{10}\ (1000/1) = 3$	3
ch_{20}	1	$log_{10}\ (1000/1) = 3$	3

3.3.1.3 Digest

3.3.1.3.1 Chunk Skor

$w^d_{ch_i}$ = ch - $wght^d_{ch_i}$ * doc - $wght_{ch_i}$

ch_i	ch - $wght^d_{ch_i}$	doc - $wght_{ch_i}$	$w^d_{ch_i}$
ch_0	1	3	3
ch_1	1	3	3
ch_2	1	3	3
ch_3	1	3	3
ch_4	1	3	3
ch_5	1	3	3
ch_6	1	3	3
ch_7	1	3	3
ch_8	1	3	3

ch_9	1	3	3
ch_{10}	1	3	3
ch_{11}	1	3	3
ch_{12}	1	3	3
ch_{13}	1	3	3
ch_{14}	1	3	3
ch_{15}	1	3	3
ch_{16}	1	3	3
ch_{17}	1	3	3
ch_{18}	1	3	3
ch_{19}	1	3	3
ch_{20}	1	3	3

3.3.1.3.2 Digest Vektor

$$Digest\ (D_1) = w_{ch_0}^{D_1}, w_{ch_1}^{D_1}, w_{ch_2}^{D_1}, \ldots, w_{ch_{n-1}}^{D_1}$$

ch_i	$w_{ch_i}^{d}$
ch_0	3
ch_1	3
ch_2	3
ch_3	3
ch_4	3
ch_5	3
ch_6	3
ch_7	3
ch_8	3
ch_9	3
ch_{10}	3
ch_{11}	3
ch_{12}	3
ch_{13}	3
ch_{14}	3
ch_{15}	3
ch_{16}	3
ch_{17}	3
ch_{18}	3
ch_{19}	3
ch_{20}	3

3.3.2 Digest 2

Nilai digest 2 merupakan hasil perbandingan antara data test inputan tenaga pendidik dengan semua data set yang ada di database.

3.3.2.1 Chunk Frekuensi

3.3.2.1.1 Membagi file menjadi chunk

Datatest = MAMALIA ADA YANG BERENANG

Dari seluruh data teks tersebut kemudian akan dibagi beberapa fragmen file atau chunk dengan panjang 7. Tabel berikut menunjukkan chunk dari data test

ch_i	Chunk	ch_i	Chunk
ch_0	'MAMALIA'	ch_{11}	' Y ANG '
ch_1	'AMALIA '	ch_{12}	'Y ANG '
ch_2	'MALIA A'	ch_{13}	' ANG B'
ch_3	'ALIA AD'	ch_{14}	'ANG BE'
ch_4	'LIA ADA'	ch_{15}	'NG BER'
ch_5	'IA ADA '	ch_{16}	'G BERE'
ch_6	'A ADA Y'	ch_{17}	' BEREN'
ch_7	' ADA Y '	ch_{18}	' BERENA'
ch_8	'ADA Y A'	ch_{19}	'BERENAN'
ch_9	'DA Y AN'	ch_{20}	'ERENANG'
ch_{10}	'A Y ANG'		

3.3.2.1.2 Perhitungan rolling hash

Sebelum dilakukan perhitungan, terlebih dahulu setiap karakter di konversi ke ascii. Berikut nilai ascii dari setiap karakter.

Karakter	spasi	M	A	L	I	D	Y	N	G	B	E	R
Nilai Ascii	32	77	65	76	73	68	89	78	71	66	69	82

$$H = c_1 a^{k-1} + c_2 a^{k-2} + c_3 a^{k-3} + \ldots + c_k a^0 \; modulus \; n$$

$$H_{new} = a * H - c_0 a^k + incoming \; byte \; modulus \; n$$

Dimana: k = 7; a = 255 (konstanta); n = 801385653117583 579 (bilangan prima yang lebih besar dari 2^{56} atau 72 057 594 037 927)

Berikut nilai rolling Hash dari setiap chunk

ch_i	Chunk	Rolling Hash
ch_0	'MAMALIA'	21240943672735580
ch_1	'AMALIA '	39195471272886637
ch_2	'MALIA A'	60436410849296862
ch_3	'ALIA AD'	78389893886482430
ch_4	'LIA ADA'	99364469906692960
ch_5	'IA ADA '	119505455320065642
ch_6	'A ADA Y'	137411463611115266
ch_7	' ADA Y '	146279979366660043
ch_8	'ADA Y A'	164224802403078243
ch_9	'DA Y AN'	182991078227609946
ch_{10}	'A Y ANG'	200897187399756782
ch_{11}	' Y ANG '	209791427835090619
ch_{12}	'Y ANG '	234296044217719086
ch_{13}	' ANG B'	243164602355243862
ch_{14}	'ANG BE'	261120232796561811
ch_{15}	'NG BER'	282642396870529513
ch_{16}	'G BERE'	302197917577762342
ch_{17}	' BEREN'	311030844217368190
ch_{18}	' BERENA'	319900442689359495
ch_{19}	'BERENAN'	338121358419642348
ch_{20}	'ERENANG'	357181061972551184

Chunk ke-0

Chunk = ‘MAMALIA'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = \text{ord(M)} * 255^6 + \text{ord(A)} * 255^5 + \text{ord(M)} * 255^4 + \text{ord(A)} * 255^3 + \text{ord(L)} * 255^2 + \text{ord(I)} * 255^1 + \text{ord(A)} * 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	77	274941996890625	21170533760578125
1	65	1078203909375	21240617014687500
2	77	4228250625	21240942589985625
3	65	16581375	21240943667775000
4	76	65025	21240943672716900
5	73	255	21240943672735515
6	65	1	21240943672735580

Total		21240943672735580
% n		21240943672735580

Chunk ke-1

Chunk = ‘AMALIA '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = \text{ord(A)} * 255^6 + \text{ord(M)} * 255^5 + \text{ord(A)} * 255^4 + \text{ord(L)} * 255^3 + \text{ord(I)} * 255^2 + \text{ord(A)} * 255^1 + \text{ord( )} * 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	65	274941996890625	39112173470626205
1	77	1078203909375	39195195171648080
2	65	4228250625	39195470007938705
3	76	16581375	39195471268123205
4	73	65025	39195471272870030
5	65	255	39195471272886605
6	32	1	39195471272886637
Total			39195471272886637
% n			39195471272886637

Chunk ke-2

Chunk = ‘MALIA_A'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = \text{ord(M)} * 255^6 + \text{ord(A)} * 255^5 + \text{ord(L)} * 255^4 + \text{ord(I)} * 255^3 + \text{ord(A)} * 255^2 + \text{ord( )} * 255^1 + \text{ord(A)} * 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	77	274941996890625	60366005033464762
1	65	1078203909375	60436088287574137
2	76	4228250625	60436409634621637
3	73	16581375	60436410845062012
4	65	65025	60436410849288637
5	32	255	60436410849296797
6	65	1	60436410849296862
Total			60436410849296862
% n			60436410849296862

Chunk ke-3

Chunk = ‘ALIA AD'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord(A)} * 255^6 + \text{ord(L)} * 255^5 + \text{ord(I)} * 255^4 + \text{ord(A)} * 255^3 + \text{ord( )} * 255^2 + \text{ord(A)} * 255^1 + \text{ord(D)} * 255^0$ *modulus n*

i	$Ord(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$Ord(B_i^D)$* $255^{k-(i+1)}$
0	65	274941996890625	78307640647187487
1	76	1078203909375	78389584144299987
2	73	4228250625	78389892806595612
3	65	16581375	78389893884384987
4	32	65025	78389893886465787
5	65	255	78389893886482362
6	68	1	78389893886482430
Total			78389893886482430
% n			78389893886482430

Chunk ke-4

Chunk = ‘LIA_ADA'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord(L)} * 255^6 + \text{ord(I)} * 255^5 + \text{ord(A)} * 255^4 + \text{ord( )} * 255^3 + \text{ord(A)} * 255^2 + \text{ord(D)} * 255^1 + \text{ord(A)} * 255^0$ *modulus n*

i	$Ord(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$Ord(B_i^D)$* $255^{k-(i+1)}$
0	76	274941996890625	99285485650169930
1	73	1078203909375	99364194535554305
2	65	4228250625	99364469371844930
3	32	16581375	99364469902448930
4	65	65025	99364469906675555
5	68	255	99364469906692895
6	65	1	99364469906692960
Total			99364469906692960
% n			99364469906692960

Chunk ke-5

Chunk = ‘IA ADA '

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord(I)} * 255^6 + \text{ord(A)} * 255^5 + \text{ord( )} * 255^4 + \text{ord(A)} * 255^3 + \text{ord(D)} *$

255^2+ord(A)∗ 255^1+ord( )∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$
0	73	274941996890625	119435235679708585
1	65	1078203909375	119505318933817960
2	32	4228250625	119505454237837960
3	65	16581375	119505455315627335
4	68	65025	119505455320049035
5	65	255	119505455320065610
6	32	1	119505455320065642
Total			119505455320065642
% n			119505455320065642

Chunk ke-6

Chunk = ‘A ADA Y'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)∗ 255^6 + ord( )∗ 255^5 + ord(A)∗ 255^4 + ord(D)∗ 255^3 +ord(A)∗ 255^2+ord( )∗ 255^1+ord(Y)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$
0	65	274941996890625	137376685117956267
1	32	1078203909375	137411187643056267
2	65	4228250625	137411462479346892
3	68	16581375	137411463606880392
4	65	65025	137411463611107017
5	32	255	137411463611115177
6	89	1	137411463611115266
Total			137411463611115266
% n			137411463611115266

Chunk ke-7

Chunk = ‘ ADA Y '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )∗ 255^6 + ord(A)∗ 255^5 + ord(D)∗ 255^4 + ord(A)∗ 255^3 +ord( )∗ 255^2+ord(Y)∗ 255^1+ord( )∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$

0	32	274941996890625	146209607511615266
1	65	1078203909375	146279690765724641
2	68	4228250625	146279978286767141
3	65	16581375	146279979364556516
4	32	65025	146279979366637316
5	89	255	146279979366660011
6	32	1	146279979366660043
Total			146279979366660043
% n			146279979366660043

Chunk ke-8

Chunk = ‘ADA_Y_A'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord(D)$* 255^5$ + ord(A)$* 255^4$ + ord( )$* 255^3$ +ord(Y)$* 255^2$+ord( )$* 255^1$+ord(A)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0	65	274941996890625	164151209164550668
1	68	1078203909375	164224527030388168
2	65	4228250625	164224801866678793
3	32	16581375	164224802397282793
4	89	65025	164224802403070018
5	32	255	164224802403078178
6	65	1	164224802403078243
Total			164224802403078243
% n			164224802403078243

Chunk ke-9

Chunk = ‘DA Y AN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*

Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*

Rolling Hash (ch_0) = ord(D)$* 255^6$ + ord(A)$* 255^5$ + ord( )$* 255^4$ + ord(Y)$* 255^3$ +ord( )$* 255^2$+ord(A)$* 255^1$+ord(N)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	68 65 32 89 32 65 78	274941996890625 1078203909375 4228250625 16581375 65025 255 1	182920858191640743 182990941445750118 182991076749770118 182991078225512493 182991078227593293 182991078227609868 182991078227609946
Total			182991078227609946
% n			182991078227609946

Chunk ke-10

Chunk = ‘A Y ANG'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)$* 255^6$ + ord( )$* 255^5$ + ord(Y)$* 255^4$ + ord( )$* 255^3$ +ord(A)$* 255^2$+ord(N)$* 255^1$+ord(G)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	65 32 89 32 65 78 71	274941996890625 1078203909375 4228250625 16581375 65025 255 1	200862308025500571 200896810550600571 200897186864906196 200897187395510196 200897187399736821 200897187399756711 200897187399756782
Total			200897187399756782
% n			200897187399756782

Chunk ke-11

Chunk = ‘ Y ANG '

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )$* 255^6$ + ord(Y)$* 255^5$ + ord( )$* 255^4$ + ord(A)$* 255^3$ +ord(N)$* 255^2$+ord(G)$* 255^1$+ord( )$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 89 32 65 78 71 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	209695331300256782 209791291448191157 209791426752211157 209791427830000532 209791427835072482 209791427835072482 209791427835090619
Total			209791427835090619
% n			209791427835090619

Chunk ke-12

Chunk = ‘Y ANG '

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord(Y)} * 255^6 + \text{ord( )} * 255^5 + \text{ord(A)} * 255^4 + \text{ord(N)} * 255^3 + \text{ord(G)} * 255^2 + \text{ord( )} * 255^1 + \text{ord( )} * 255^0$ *modulus n*

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$
0 1 2 3 4 5 6	89 32 65 78 71 32 32	274941996890625 1078203909375 4228250625 16581375 65025 255 1	234261265558356244 234295768083456244 234296042919746869 234296044213094119 234296044217710894 234296044217719054 234296044217719086
Total			234296044217719086
% n			234296044217719086

Chunk ke-13

Chunk = ‘ ANG B'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
$Rolling\ Hash\ (ch_0) = \text{ord( )} * 255^6 + \text{ord(A)} * 255^5 + \text{ord(N)} * 255^4 + \text{ord(G)} * 255^3 + \text{ord( )} * 255^2 + \text{ord( )} * 255^1 + \text{ord(B)} * 255^0$ *modulus n*

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 65 78 71 32 32 66	274941996890625 1078203909375 4228250625 16581375 65025 255 1	243094188118219086 243164271372328461 243164601175877211 243164602353154836 243164602355235636 243164602355243796 243164602355243862
Total			243164602355243862
% n			243164602355243862

Chunk ke-14

Chunk = ‘ANG BE'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(A)∗ 255^6 + ord(N)∗ 255^5 + ord(G)∗ 255^4 + ord( )∗ 255^3 +ord( )∗ 255^2+ord(B)∗ 255^1+ord(E)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$
0 1 2 3 4 5 6	65 78 71 32 32 66 69	274941996890625 1078203909375 4228250625 16581375 65025 255 1	261035832153134487 261119932058065737 261120232263860112 261120232794464112 261120232796544912 261120232796561742 261120232796561811
Total			261120232796561811
% n			261120232796561811

Chunk ke-15

Chunk = ‘NG__BER'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(N)∗ 255^6 + ord(G)∗ 255^5 + ord( )∗ 255^4 + ord( )∗ 255^3 +ord(B)∗ 255^2+ord(E)∗ 255^1+ord(R)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$

0 1 2 3 4 5 6	78 71 32 32 66 69 82	274941996890625 1078203909375 4228250625 16581375 65025 255 1	282565708554030561 282642261031596186 282642396335616186 282642396866220186 282642396870511836 282642396870529431 282642396870529513
Total			282642396870529513
% n			282642396870529513

Chunk ke-16

Chunk = ‘G BERE'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(G)∗ 255^6 + ord( )∗ 255^5 + ord( )∗ 255^4 + ord(B)∗ 255^3 +ord(E)∗ 255^2+ord(R)∗ 255^1+ord(E)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$
0 1 2 3 4 5 6	71 32 32 66 69 82 69	274941996890625 1078203909375 4228250625 16581375 65025 255 1	302163278649763888 302197781174863888 302197916478883888 302197917573254638 302197917577741363 302197917577762273 302197917577762342
Total			302197917577762342
% n			302197917577762342

Chunk ke-17

Chunk = ‘ BEREN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )∗ 255^6 + ord( )∗ 255^5 + ord(B)∗ 255^4 + ord(E)∗ 255^3 +ord(R)∗ 255^2+ord(E)∗ 255^1+ord(N)∗ 255^0 *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)∗ $255^{k-(i+1)}$

0 1 2 3 4 5 6	32 32 66 69 82 69 78	274941996890625 1078203909375 4228250625 16581375 65025 255 1	310996061478262342 311030564003362342 311030843067903592 311030844212018467 311030844217350517 311030844217368112 311030844217368190
Total			311030844217368190
% n			311030844217368190

Chunk ke-18

Chunk = ‘ BERENA'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord( )$* 255^6$ + ord(B)$* 255^5$ + ord(E)$* 255^4$ + ord(R)$* 255^3$ +ord(E)$* 255^2$+ord(N)$* 255^1$+ord(A)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$
0 1 2 3 4 5 6	32 66 69 82 69 78 65	274941996890625 1078203909375 4228250625 16581375 65025 255 1	319828988117868190 319900149575886940 319900441325180065 319900442684852815 319900442689339540 319900442689359430 319900442689359495
Total			319900442689359495
% n			319900442689359495

Chunk ke-19

Chunk = ‘BERENAN'

Rolling Hash $(ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0$ *modulus n*
Rolling Hash $(ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0$ *modulus n*
Rolling Hash (ch_0) = ord(B)$* 255^6$ + ord(E)$* 255^5$ + ord(R)$* 255^4$ + ord(E)$* 255^3$ +ord(N)$* 255^2$+ord(A)$* 255^1$+ord(N)$* 255^0$ *modulus n*

i	Ord(B_i^D)	$255^{k-(i+1)}$	Sebelumnya+Ord(B_i^D)* $255^{k-(i+1)}$

0 1 2 3 4 5 6	66 69 82 69 78 65 78	274941996890625 1078203909375 4228250625 16581375 65025 255 1	338046614484140745 338121010553887620 338121357270438870 338121358414553745 338121358419625695 338121358419642270 338121358419642348
Total			338121358419642348
% n			338121358419642348

Chunk ke-20

Chunk = 'ERENANG'

$Rolling\ Hash\ (ch_i) = B_i^D a^{k-1} + B_{i+1}^D a^{k-2} + B_{i+2}^D a^{k-3} + \ldots + B_{i+k-1}^D a^0\ modulus\ n$

$Rolling\ Hash\ (ch_0) = B_0^D 255^{7-1} + B_{0+1}^D 255^{7-2} + B_{0+2}^D 255^{7-3} + \ldots + B_{0+7-1}^D 255^0\ modulus\ n$

$Rolling\ Hash\ (ch_0) = B_0^D 255^6 + B_1^D 255^5 + B_2^D 255^4 + B_3^D 255^3 + B_4^D 255^2 + B_5^D 255^1 + B_6^D 255^0\ modulus\ n$

$Rolling\ Hash\ (ch_0) = \text{ord(E)} * 255^6 + \text{ord(R)} * 255^5 + \text{ord(E)} * 255^4 + \text{ord(N)} * 255^3 + \text{ord(A)} * 255^2 + \text{ord(N)} * 255^1 + \text{ord(G)} * 255^0\ modulus\ n$

i	$\text{Ord}(B_i^D)$	$255^{k-(i+1)}$	Sebelumnya+$\text{Ord}(B_i^D)$* $255^{k-(i+1)}$
0 1 2 3 4 5 6	69 82 69 78 65 78 71	274941996890625 1078203909375 4228250625 16581375 65025 255 1	357092356205095473 357180768925664223 357181060674957348 357181061968304598 357181061972531223 357181061972551113 357181061972551184
Total			357181061972551184
% n			357181061972551184

3.3.2.1.3 Perhitungan chunk frequensi

Setelah nilai rolling hash dihitung, nilai tersebut harus unik. Dalam hal ini, menggunakan fungsi uniq() pada python. Fungsi tersebut bertujuan agar setiap elemen didalam fungsi tersebut tidak ada nilai yang sama. Setelah di implementasikan, hasilnya dilihat pada tabel dibawah. setiap nilai rolling hash hanya muncul sekali dan tidak terdapat nilai rolling hash yang sama, maka nilai frekuensi untuk setiap ch_i adalah 1.

ch_i	Nilai Rolling Hash	Tabel Chunk Frekuensi	
		Nilai Rolling Hash	chf_{ch}^{d}
ch_0	'MAMALIA'	21240943672735580	1
ch_1	'AMALIA '	39195471272886637	1
ch_2	'MALIA A'	60436410849296862	1
ch_3	'ALIA AD'	78389893886482430	1
ch_4	'LIA ADA'	99364469906692960	1
ch_5	'IA ADA '	119505455320065642	1
ch_6	'A ADA Y'	137411463611115266	1
ch_7	' ADA Y '	146279979366660043	1
ch_8	'ADA Y A'	164224802403078243	1
ch_9	'DA Y AN'	182991078227609946	1
ch_{10}	'A Y ANG'	200897187399756782	1
ch_{11}	' Y ANG '	209791427835090619	1
ch_{12}	'Y ANG '	234296044217719086	1
ch_{13}	' ANG B'	243164602355243862	1
ch_{14}	'ANG BE'	261120232796561811	1
ch_{15}	'NG BER'	282642396870529513	1
ch_{16}	'G BERE'	302197917577762342	1
ch_{17}	' BEREN'	311030844217368190	1
ch_{18}	' BERENA'	319900442689359495	1
ch_{19}	'BERENAN'	338121358419642348	1
ch_{20}	'ERENANG'	357181061972551184	1

3.3.2.1.4 Perhitungan chunk weight

Berdasarkan frekuensi *chunk*, bobot *chunk* akan ditetapkan untuk setiap *chunk*, menggunakan rumus berikut:

chunk-$wght_{ch}$ = 1 + log_{10} (chf_{ch}^{d})

dimana:

chunk-$wght_{ch}$= nilai bobot dari setiap chunk

chf_{ch}^{d} = Chunk frekuensi

Sehingga,

ch_i	Nilai Rolling Hash	chf_{ch}^{d}	$1 + log_{10}(chf_{ch}^{d})$	ch - $wght_{ch_i}^{d}$
ch_0	21240943672735580	1	$1 + log_{10}(1) = 1$	1
ch_1	39195471272886637	1	$1 + log_{10}(1) = 1$	1
ch_2	60436410849296862	1	$1 + log_{10}(1) = 1$	1
ch_3	78389893886482430	1	$1 + log_{10}(1) = 1$	1
ch_4	99364469906692960	1	$1 + log_{10}(1) = 1$	1
ch_5	119505455320065642	1	$1 + log_{10}(1) = 1$	1
ch_6	137411463611115266	1	$1 + log_{10}(1) = 1$	1

ch_7	146279979366660043	1	$1 + log_{10}(1) = 1$	1
ch_8	164224802403078243	1	$1 + log_{10}(1) = 1$	1
ch_9	182991078227609946	1	$1 + log_{10}(1) = 1$	1
ch_{10}	200897187399756782	1	$1 + log_{10}(1) = 1$	1
ch_{11}	209791427835090619	1	$1 + log_{10}(1) = 1$	1
ch_{12}	234296044217719086	1	$1 + log_{10}(1) = 1$	1
ch_{13}	243164602355243862	1	$1 + log_{10}(1) = 1$	1
ch_{14}	261120232796561811	1	$1 + log_{10}(1) = 1$	1
ch_{15}	282642396870529513	1	$1 + log_{10}(1) = 1$	1
ch_{16}	302197917577762342	1	$1 + log_{10}(1) = 1$	1
ch_{17}	311030844217368190	1	$1 + log_{10}(1) = 1$	1
ch_{18}	319900442689359495	1	$1 + log_{10}(1) = 1$	1
ch_{19}	338121358419642348	1	$1 + log_{10}(1) = 1$	1
ch_{20}	357181061972551184	1	$1 + log_{10}(1) = 1$	1

3.3.2.2 Dokumen Frekuensi

Tujuan perhitungan dokumen frekuensi adalah untuk menghitung berapa banyak dokumen yang muncul hash tersebut. Dokumen data test inputan tenaga pendidik akan dibandingkan dengan dokumen lain yang ada di database

3.3.2.2.1 Rolling Hash

Datatest = MAMALIA TIDAK BERTELUR

ch_i	Rolling Hash
ch_0	21240943672735580,
ch_1	39195471272886637,
ch_2	60436410849296862,
ch_3	78389893886482430,
ch_4	99364469906692960,
ch_5	119505455320065642,
ch_6	137411463611115266,
ch_7	146279979366660043,
ch_8	164224802403078243,
ch_9	182991078227609946,
ch_{10}	200897187399756782,
ch_{11}	209791427835090619,
ch_{12}	234296044217719086,
ch_{13}	243164602355243862,

ch_{14}	261120232796561811,
ch_{15}	282642396870529513,
ch_{16}	302197917577762342,
ch_{17}	311030844217368190,
ch_{18}	319900442689359495,
ch_{19}	338121358419642348,
ch_{20}	357181061972551184

3.3.2.2.2 Chunk Frekuensi

Berikut nilai Chunk Frekuensi dari setiap chunk

ch_i	Chunk Frekuensi	chf_{ch}^{d}
ch_0	137411463611115266: 1	1
ch_1	319900442689359495: 1	1
ch_2	357181061972551184: 1	1
ch_3	261120232796561811: 1	1
ch_4	302197917577762342: 1	1
ch_5	234296044217719086: 1	1
ch_6	78389893886482430: 1	1
ch_7	209791427835090619: 1	1
ch_8	146279979366660043: 1	1
ch_9	243164602355243862: 1	1
ch_{10}	182991078227609946: 1	1
ch_{11}	21240943672735580: 1	1
ch_{12}	60436410849296862: 1	1
ch_{13}	99364469906692960: 1	1
ch_{14}	164224802403078243: 1	1
ch_{15}	282642396870529513: 1	1
ch_{16}	119505455320065642: 1	1
ch_{17}	338121358419642348: 1	1
ch_{18}	39195471272886637: 1	1
ch_{19}	200897187399756782: 1	1
ch_{20}	311030844217368190: 1	1

3.3.2.2.3 Chunk Weight

Berdasarkan frekuensi *chunk*, bobot *chunk* akan ditetapkan untuk setiap *chunk*, menggunakan rumus berikut:

chunk-$wght_{ch}$ = 1 + log_{10} (chf_{ch}^{d})

dimana:

chunk-$wght_{ch}$= nilai bobot dari setiap chunk

chf_{ch}^{d} = Chunk frekuensi

Sehingga,

ch_i	Chunk Frequency	chf_{ch}^{d}	$1 + log_{10}\,(chf_{ch}^{d})$	ch - $wght_{ch_i}^{d}$
ch_0	137411463611115266: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_1	319900442689359495: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_2	357181061972551184: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_3	261120232796561811: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_4	302197917577762342: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_5	234296044217719086: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_6	78389893886482430: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_7	209791427835090619: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_8	146279979366660043: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_9	243164602355243862: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{10}	182991078227609946: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{11}	21240943672735580: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{12}	60436410849296862: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{13}	99364469906692960: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{14}	164224802403078243: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{15}	282642396870529513: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{16}	119505455320065642: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{17}	338121358419642348: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{18}	39195471272886637: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{19}	200897187399756782: 1	1	$1 + log_{10}\,(1) = 1$	1
ch_{20}	311030844217368190: 1	1	$1 + log_{10}\,(1) = 1$	1

3.3.2.2.4 Dokumen Frekuensi

Data set = ‘MAMALIA TIDAK BERTELUR’

Kemudian, nilai rolling hash dari data test yang dihitung pertama tersebut dibandingkan dengan nilai rolling hash data test baru untuk menghitung nilai dokumen frekensinya.

ch_i	Chunk Freuency	df_{ch}
ch_0	'Y ANG ': 0	0
ch_1	'A Y ANG': 0	0
ch_2	'NG BER': 0	0
ch_3	' BERENA': 0	0
ch_4	'ADA Y A': 0	0
ch_5	'ERENANG': 0	0
ch_6	'AMALIA ': 1	1
ch_7	'DA Y AN': 0	0
ch_8	'G BERE': 0	0

ch_9	'LIA ADA': 0	0
ch_{10}	'A ADA Y': 0	0
ch_{11}	' Y ANG ': 0	0
ch_{12}	'BERENAN': 0	0
ch_{13}	'MALIA A': 0	0
ch_{14}	'ALIA AD': 0	0
ch_{15}	' BEREN': 0	0
ch_{16}	'IA ADA ': 0	0
ch_{17}	'ANG BE': 0	0
ch_{18}	'MAMALIA': 1	1
ch_{19}	' ANG B': 0	0
ch_{20}	' ADA Y ': 0	0

3.3.2.2.5 Dokumen Weight

If $df_{ch} > 0$: $doc\text{-}wght_{ch} = log_{10}(1000/df_{ch})$

Else $doc\text{-}wght_{ch} = 1$

ch_i	df_{ch})	$log_{10}(1000/df_{ch})$	$doc\text{-}wght_{ch}$
ch_0	0	$doc - wght_{ch} = 1$	1.0
ch_1	0	$doc - wght_{ch} = 1$	1.0
ch_2	0	$doc - wght_{ch} = 1$	1.0
ch_3	0	$doc - wght_{ch} = 1$	1.0
ch_4	0	$doc - wght_{ch} = 1$	1.0
ch_5	0	$doc - wght_{ch} = 1$	1.0
ch_6	1	$log_{10}(1000/1) = 3$	3.0
ch_7	0	$doc - wght_{ch} = 1$	1.0
ch_8	0	$doc - wght_{ch} = 1$	1.0
ch_9	0	$doc - wght_{ch} = 1$	1.0
ch_{10}	0	$doc - wght_{ch} = 1$	1.0
ch_{11}	0	$doc - wght_{ch} = 1$	1.0
ch_{12}	0	$doc - wght_{ch} = 1$	1.0
ch_{13}	0	$doc - wght_{ch} = 1$	1.0
ch_{14}	0	$doc - wght_{ch} = 1$	1.0
ch_{15}	0	$doc - wght_{ch} = 1$	1.0
ch_{16}	0	$doc - wght_{ch} = 1$	1.0
ch_{17}	0	$doc - wght_{ch} = 1$	1.0
ch_{18}	1	$log_{10}(1000/1) = 3$	3.0
ch_{19}	0	$doc - wght_{ch} = 1$	1.0
ch_{20}	0	$doc - wght_{ch} = 1$	1.0

3.3.2.3 Digest

3.3.2.3.1 Chunk Skor

$$w^{d}_{ch_i} = \text{ch - } wght^{d}_{ch_i} * \text{doc - } wght_{ch_i}$$

ch_i	ch - $wght^{d}_{ch_i}$	doc - $wght_{ch_i}$	$w^{d}_{ch_i}$
ch_0	1	1.0	1.0
ch_1	1	1.0	1.0
ch_2	1	1.0	1.0
ch_3	1	1.0	1.0
ch_4	1	1.0	1.0
ch_5	1	1.0	1.0
ch_6	1	3.0	3.0
ch_7	1	1.0	1.0
ch_8	1	1.0	1.0
ch_9	1	1.0	1.0
ch_{10}	1	1.0	1.0
ch_{11}	1	1.0	1.0
ch_{12}	1	1.0	1.0
ch_{13}	1	1.0	1.0
ch_{14}	1	1.0	1.0
ch_{15}	1	1.0	1.0
ch_{16}	1	1.0	1.0
ch_{17}	1	1.0	1.0
ch_{18}	1	3.0	3.0
ch_{19}	1	1.0	1.0
ch_{20}	1	1.0	1.0

3.3.2.3.2 Digest Vektor

$$Digest\ (D_1) = w^{D_1}_{ch_0}, w^{D_1}_{ch_1}, w^{D_1}_{ch_2}, \ldots, w^{D_1}_{ch_{n-1}}$$

ch_i	$w^{d}_{ch_i}$
ch_0	1.0
ch_1	1.0
ch_2	1.0
ch_3	1.0
ch_4	1.0
ch_5	1.0
ch_6	3.0
ch_7	1.0
ch_8	1.0
ch_9	1.0
ch_{10}	1.0
ch_{11}	1.0

ch_{12}	1.0
ch_{13}	1.0
ch_{14}	1.0
ch_{15}	1.0
ch_{16}	1.0
ch_{17}	1.0
ch_{18}	3.0
ch_{19}	1.0
ch_{20}	1.0

3.3.3 Cosin Similarity

$$Similarity(D_1, D_2) = \frac{\sum_{i=0}^{n-1} W_{ch_i}^{D1} * w_{ch_i}^{D_2}}{\sqrt{\sum_{i=0}^{n-1} W_{ch_i}^{D1^2}} * \sqrt{\sum_{i=0}^{n-1} W_{ch_i}^{D_2^2}}} * 100$$

D1=D1:D1
D2=D1:D2

$$Similarity(D_1, D_2) = \frac{75}{83.62415918859813} * 100$$

$$Similarity(D_1, D_2) = 89.6870004167719$$

3.4 Lokasi dan Jadwal Penelitian

Dalam bab ini akan membahas mengenai lokasi yang digunakan dalam penelitian sistem deteksi plagiasi menggunakan algoritme Frequency Based Hashing-S pada file PDF.

3.4.1 Lokasi Penelitian

Tidak terdapat lokasi pada khusus pada penelitian. Karena data yang digunakan adalah dummy

3.4.2 Jadwal Penelitian

Setiap rancangan penelitian perlu dilengkapi dengan jadwal

kegiatan yang akan dilaksanakan dimana jadwal tersebut berisi jadwal kegiatan apa saja yang akan dilakukan selama penelitian. Berikut jadwal pada penelitian ini.

Tabel 3. 6 Jadwal Penelitian

No	Kegiatan	Waktu Kegiatan 2022															
		Maret 2022				April 2022				Juni 2022				Juli 2022			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pemilihan Judul																
2	Pengajuan Judul Proposal																
3	Penyusunan Bab I																
4	Penyusunan Bab II																
5	Penyusunan Bab III																
6	engumpulan																