**DETEKSI TINGKAT KEMIRIPAN ABSTRAK TUGAS AKHIR MAHASISWA MENGGUNAKAN ALGORITME *WINNOWING* BERBASIS *N-GRAM* DAN *JACCARD SIMILARITY* PADA UNIVERSITAS BUDI LUHUR**

**TUGAS AKHIR**

****

**Oleh :**

**WAHYU DESENA**

**NIM : 1711502821**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS BUDI LUHUR**

**JAKARTA**

**2021**

**DETEKSI TINGKAT KEMIRIPAN ABSTRAK TUGAS AKHIR MAHASISWA MENGGUNAKAN ALGORITME *WINNOWING* BERBASIS *N-GRAM* DAN *JACCARD SIMILARITY* PADA UNIVERSITAS BUDI LUHUR**

**Diajukan untuk memenuhi salah satu persyaratan**

**memperoleh gelar Sarjana Komputer (S.Kom)**

**TUGAS AKHIR**

****

**Oleh :**

**WAHYU DESENA**

**NIM : 1711502821**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS BUDI LUHUR**

**JAKARTA**

**2021**

**TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS BUDI LUHUR**

**PERSETUJUAN TUGAS AKHIR**

Nama : WAHYU DESENA

Nomor Induk Mahasiswa : 1711502821

Program Studi : Teknik Informatika

Bidang Peminatan : *Programming Expert*

Jenjang Studi : Strata 1

Judul : Deteksi Tingkat Kemiripan Abstrak Tugas Akhir Mahasiswa Menggunakan Algoritme *Winnowing* berbasis *N-Gram* dan *Jaccard Similarity* pada Universitas Budi Luhur

Disetujui untuk dipertahankan dalam sidang Tugas Akhir periode semester Gasal tahun ajaran 2020/2021

|  |
| --- |
| Jakarta, 08 Februari 2021 |
| Dosen Pembimbing |
|  |
| ( Dr. Achmad Solichin, S.Kom, M.T.I.) |

# Abstrak

**DETEKSI TINGKAT KEMIRIPAN ABSTRAK TUGAS AKHIR MAHASISWA MENGGUNAKAN ALGORITME *WINNOWING* BERBASIS *N-GRAM* DAN *JACCARD SIMILARITY* PADA UNIVERSITAS BUDI LUHUR**

**Oleh : Wahyu Desena (1711502821)**

Dokumen Skripsi merupakan dokumen yang mempresentasikan penelitian dan riset yang dilakukan oleh mahasiswa jenjang strata satu. Untuk menghasilkan skripsi yang berkualitas dibutuhkan penelitian yang kompeten, salah satu faktornya adalah originalitas, dalam arti tidak plagiarisme. Untuk mengantisipasi plagiarisme dibutuhkan suatu cara yang dapat mendeteksi tingkat *similarity* kata, diantaranya menggunakan algoritme *winnowing.* Algoritme *winnowing* digunakan untuk melakukan proses pengecekan kesamaan kata (*document fingerprinting*) untuk mengidentifikasi tingkat *similarity.* Algoritme ini dapat diproses dengan cara mencari nilai *hash* dari setiap *string* yang sudah dipotong pada tahapan *n-gram* berdasarkan nilai *k-gram* yang digunakan*.* Kemudian dilakukan proses pembentukan *window* berdasarkan nilai *w-gram* yang digunakan. Setelah pembentukan *window,* kemudian dicari nilai *fingerprint* dari *window* tersebut (nilai terkecil dari *window* yang dibentuk). Setelah nilai *fingerprint* didapatkan, maka proses selanjutnya adalah mencari nilai *similarity* menggunakan metode *jaccard similarity,* metode ini digunakan untuk membandingkan dua sampel yaitu dokumen yang satu dengan yang lainnya. Hasil dari perhitungan *jaccard similarity* ditampilkan dalam bentuk persentase. Berdasarkan hasil pengujian, nilai *k-gram* dan *w-gram* mempengaruhi persentase *similarity*, penggunaan *k-gram* dan *w-gram* yang tepat sangat diperlukan. Nilai *k-gram* dan *w-gram* yang terlalu kecil mengakibatkan nilai persentase *similarity* yang besar, begitupun sebaliknya. Pada tahap pengujian dengan nilai *k-gram*=3 dan *w-gram*=4 menghasilkan nilai *similarity* terbesar yaitu dengan persentase 26.50%. Serta pengujian pada dokumen yang sama menghasilkan nilai *similarity* dengan persentase 100%, dengan demikian disimpulkan bahwa metode *jaccard similarity* dapat mendeteksi tingkat *similarity* dengan hasil yang cukup baik dan tepat. Diharapkan dengan adanya sistem pendeteksi *similarity* ini, dapat mengatasi tindakan plagiarisme pada dokumen abstrak skripsi mahasiswa Budi Luhur.

**Kata Kunci :** *Similarity, N-gram, Winnowing, Jaccard Similarity,* Abstrak

Xii + 85 halaman; 40 gambar; 22 tabel

# 

# kata pengantar

Puji serta syukur Alhamdulillah, penulis panjatkan kehadirat Allah Azza Wa Jalla yang telah melimpahkan rahmat dan karunia-Nya, sehingga pada akhirnya penulis dapat menyelesaikan Tugas Akhir ini dengan baik. Adapun Tugas Akhir ini disusun untuk memenuhi persyaratan dalam menyelesaikan tingkat Pendidikan Strata 1 (S1) pada Program Studi Teknik Informatika, Fakultas Teknologi Informasi Universitas Budi Luhur dengan judul tugas akhir yang penulis ambil yaitu **“Deteksi Tingkat Kemiripan Abstrak Tugas Akhir Mahasiswa Menggunakan Algoritme Winnowing Berbasis N-Gram Dan Jaccard Similarity Pada Universitas Budi Luhur”.**

Penulis berharap laporan ini dapat memberikan manfaat kepada para pembaca umumnya dan kepada mahasiswa khususnya. Selain itu, diharapkan laporan ini juga dapat menjadi bahan perbandingan dalam melakukan karya penelitian selanjutnya. Terselesaikannya penelitian ini tidak lepas dari bantuan berbagai pihak, rasa terimakasih yang mendalam ingin penulis sampaikan kepada mereka yang telah berjasa dalam membantu penyusunan tugas akhir ini, yaitu kepada:

1. Allah Azza Wa Jalla, atas segala petunjuk, kemudahan, serta nikmat-Nya yang diberikan sehingga penulis dapat menyelesaikan penyusunan tugas akhir ini dengan baik.
2. Segenap keluarga penulis, khususnya orang tua tercinta Ayah & Ibu, serta saudari khususnya Kakak, yang telah membantu memberikan dukungan baik moral maupun material, dan selalu memberikan doa restu.
3. Bapak Dr. Ir. Wendi Usino, M.Sc., M.M., selaku Rektor Universitas Budi Luhur.
4. Bapak Dr. Deni Mahdiana, M.M., M.Kom, selaku Dekan Fakultas Teknologi Informasi Universitas Budi Luhur.
5. Bapak Gunawan Pria Utama, M.Kom, selaku Dosen Penasehat Akademik.
6. Bapak Dr. Indra, S.Kom., M.T.I., selaku Kaprodi Teknik Informatika Fakultas Teknologi Informasi Universitas Budi Luhur.
7. Bapak Dr. Achmad Solichin, S.Kom, M.T.I., selaku Dosen Pembimbing Tugas Akhir yang telah membantu penulis dalam memberikan saran-saran dalam penyelesaian laporan tugas akhir ini.
8. Ibu Painem, M.Kom., selaku Kepala Laboratorium ICT Universitas Budi Luhur yang selalu memberikan arahan dan ilmu selama penulis mengabdi di LAB ICT.
9. Teman-teman LDK Al-Azzam Universitas Budi Luhur khususnya periode 2020 yang telah memberikan dukungan dalam penyelesaian laporan tugas akhir ini.
10. Rekan-rekan Asisten Laboratorium ICT Terpadu Universitas Budi Luhur khususnya angkatan 2017, sebagai rekan kerja selama 3 tahun di LAB ICT.
11. Fatahillah Al Mahfudz, Dony Ramadhan Saputra, Mus Priandi, dan Gabriel Yoda Gustiegan sebagai sahabat maupun teman yang telah memberikan bantuan, arahan, do’a, serta saran-saran dalam penyelesaian laporan tugas akhir ini.

Jakarta Selatan, 08 Februari 2021

Penulis

# daftar Tabel

[Tabel 2.4 Studi Literatur 10](#_Toc62400050)

[Tabel 3.1 Sampel Data Abstrak 18](#_Toc62400051)

[Tabel 4.1 *Tahapan Preprocessing* 28](#_Toc62400052)

[Tabel 4.2 Pembentukan *N-Gram* 28](#_Toc62400053)

[Tabel 4.3 Kode ASCII 29](#_Toc62400054)

[Tabel 4.4 Perhitungan *Rolling Hash* 30](#_Toc62400055)

[Tabel 4.5 Perhitungan *Hash Dataset* 30](#_Toc62400056)

[Tabel 4.6 Perhitungan *Hash* Data Tes 32](#_Toc62400057)

[Tabel 4.7 Hasil *Fingerprint* 34](#_Toc62400058)

[Tabel 4.8 Uji Coba Nilai *k-gram* dan w-*gram* Sama 53](#_Toc62400059)

[Tabel 4.9 Uji Coba Nilai *k-gram* dan *w-gram* Berbeda 53](#_Toc62400060)

[Tabel 4.10 Uji Coba *Dataset* dijadikan Data Tes 1 54](#_Toc62400061)

[Tabel 4.11 Uji Coba Dataset dijadikan Data Tes 2 55](#_Toc62400062)

[Tabel 4.12 Uji Coba Dataset dijadikan Data Tes 3 55](#_Toc62400063)

# daftar gambar

[Gambar 2.1 Konsep Algoritme *Winnowing*(Sunardi et al., 2018) 9](#_Toc62400078)

[Gambar 3.1 Tahapan Metode 19](#_Toc62400079)

[Gambar 3.2 *Case folding* 21](#_Toc62400080)

[Gambar 3.3 Menghapus karakter selain a-z 22](#_Toc62400081)

[Gambar 3.5 Mengganti *Slang word* 23](#_Toc62400082)

[Gambar 3.6 Menghapus *Stop word* 24](#_Toc62400083)

[Gambar 4.7 Menghapus Spasi 24](#_Toc62400084)

[Gambar 4.1 *Flowchart* Keseluruhan Sistem 36](#_Toc62400085)

[Gambar 4.2 *Flowchart* Data Dokumen 37](#_Toc62400086)

[Gambar 4.3 *Flowchart Preprocessing* 38](#_Toc62400087)

[Gambar 4.4 *Flowchart Case folding* 39](#_Toc62400088)

[Gambar 4.5 *Flowchart Cleaning* 40](#_Toc62400089)

[Gambar 4.6 *Flowchart Slang word* 41](#_Toc62400090)

[Gambar 4.7 *Flowchart Stop word* 42](#_Toc62400091)

[Gambar 4.8 *Flowchart* Hapus Spasi 43](#_Toc62400092)

[Gambar 4.9 *Flowchart* Algoritme *Winnowing* 44](#_Toc62400093)

[Gambar 4.10 *Flowchart Rolling Hash* 45](#_Toc62400094)

[Gambar 4.11 *Flowchart Fingerpring* 46](#_Toc62400095)

[Gambar 4.12 *Flowchart Jaccard Similarity* 47](#_Toc62400096)

[Gambar 4.13 Tampilan Layar *Home* 56](#_Toc62400097)

[Gambar 4.13 Tampilan Layar *Home* (Lanjutan) 57](#_Toc62400098)

[Gambar 4.15 Tampilan Layar *Document Data* 57](#_Toc62400099)

[Gambar 4.16 Tampilan Layar *Document Data* (Lanjutan) 58](#_Toc62400100)

[Gambar 4.17 Tampilan Layar *Preprocessing* 58](#_Toc62400101)

[Gambar 4.18 Tampilan Layar Detail *Preprocessing* 59](#_Toc62400102)

[Gambar 4.19 Tampilan Layar Detail *Preprocessing* (Lanjutan) 59](#_Toc62400103)

[Gambar 4.20 Tampilan Layar *Menu Slang word* 60](#_Toc62400104)

[Gambar 4.21 Tampilan Layar *Menu Add Slang word* 60](#_Toc62400105)

[Gambar 4.22 Tampilan Layar *Menu Edit Slang word* 60](#_Toc62400106)

[Gambar 4.23 Tampilan Layar Menu Stop word 61](#_Toc62400107)

[Gambar 4.24 Tampilan Layar *Menu Add Stop word* 61](#_Toc62400108)

[Gambar 4.25 Tampilan Layar *Menu* *Edit Stop word* 62](#_Toc62400109)

[Gambar 4.26 Tampilan Layar *Menu Scan Document* 62](#_Toc62400110)

[Gambar 4.27 Tampilan Layar *Document* Data Tes 63](#_Toc62400111)

[Gambar 4.28 Tampilan Layar *Document* Data Tes (Lanjutan) 63](#_Toc62400112)

[Gambar 4.29 Tampilan Layar Nilai *Similarity* 64](#_Toc62400113)

[Gambar 4.30 Tampilan Layar Nilai *Similarity* (Lanjutan) 64](#_Toc62400114)

# daftar simbol

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
|  | ***Terminator* (Terminal)**  Menggambarkan awal atau akhir sebuah aliran data. |
|  | ***Connector* (Penghubung)**  Menggambarkan arah proses.. |
|  | ***Data (Input/ Output)***  Menggambarkan masukan atau keluaran yang dihasilkan. |
|  | ***Decision* (Kondisi)**  Menggambarkan suatu kondisi yang harus dipilih oleh program. |
|  | ***Predefined Process***  Menggambarkan proses – proses yang masih dapat dijabarkan dalam algoritme. |
|  | ***Process***  Menggambarkan sebuah proses. |
|  | ***Off Page Reference***  Menggambarkan penghubung dengan *flowchart* yang ada di halaman lain. |
|  | ***On Page Reference***  Menggambarkan penghubung dalam satu halaman yang sama. |

# daftar isi

[Abstrak iv](#_Toc62834591)

[v](#_Toc62834592)

[kata pengantar vi](#_Toc62834593)

[daftar Tabel viii](#_Toc62834594)

[daftar gambar ix](#_Toc62834595)

[daftar simbol x](#_Toc62834596)

[daftar isi xi](#_Toc62834597)

[bab i pendahuluan 1](#_Toc62834598)

[1.1. Latar Belakang 1](#_Toc62834599)

[1.2. Perumusan Masalah 2](#_Toc62834600)

[1.3. Batasan Masalah 2](#_Toc62834601)

[1.4. Tujuan 3](#_Toc62834602)

[1.5. Manfaat 3](#_Toc62834603)

[1.6. Sistematika Penulisan 3](#_Toc62834604)

[bab ii landasan teori 5](#_Toc62834605)

[2.1. Data *Mining* 5](#_Toc62834606)

[2.2. *Text Mining* 5](#_Toc62834607)

[2.3. Plagiarisme 5](#_Toc62834608)

[2.4. *Preprocessing* 6](#_Toc62834609)

[2.4.1. *Case folding* 6](#_Toc62834610)

[2.4.2. Menghapus karakter kecuali a sampai z 6](#_Toc62834611)

[2.4.3. Mengganti *slang word* 7](#_Toc62834612)

[2.4.4. Menghapus *stop word* 7](#_Toc62834613)

[2.5. *N-Gram* 7](#_Toc62834614)

[2.6. *Jaccard Similarity* 7](#_Toc62834615)

[2.7. Algoritme *Winnowing* 8](#_Toc62834616)

[2.8. Pengujian 10](#_Toc62834617)

[2.9. Studi Literatur 10](#_Toc62834618)

[bab iii metodologi penelitian 18](#_Toc62834619)

[3.1. Data Penelitian 18](#_Toc62834620)

[3.2. Penerapan Metode 19](#_Toc62834621)

[3.2.1. Data Dokumen 20](#_Toc62834622)

[3.2.2. *Preprocessing* 20](#_Toc62834623)

[3.2.3. *N-Gram* 25](#_Toc62834624)

[3.2.4. Algoritme *Winnowing* 25](#_Toc62834625)

[b. Pembentukan *Window* 25](#_Toc62834626)

[3.3. Rancangan Pengujian 26](#_Toc62834627)

[bab iV hasil dan pembahasan 27](#_Toc62834628)

[4.1. Lingkungan Percobaan 27](#_Toc62834629)

[4.1.1. Spesifikasi Perangkat Keras 27](#_Toc62834630)

[4.1.2. Spesifikasi Perangkat Lunak 27](#_Toc62834631)

[4.2. Implementasi Metode dan Langkah Pengujian 27](#_Toc62834632)

[4.2.1. Data Dokumen 27](#_Toc62834633)

[4.2.2. Tahapan *Preprocessing* 27](#_Toc62834634)

[4.2.3. Pembentukan *N-Gram* 28](#_Toc62834635)

[4.2.4. Algoritme *Winnowing* 29](#_Toc62834636)

[a. Perhitungan *Rolling Hash* 29](#_Toc62834637)

[b. Pembentukan *Window* 33](#_Toc62834638)

[c. Pencarian *Fingerprint* 34](#_Toc62834639)

[4.2.5. *Jaccard Similarity* 34](#_Toc62834640)

[4.3. *Flowchart* Tahapan Metode 35](#_Toc62834641)

[4.3.1. *Flowchart* Keseluruhan Sistem 35](#_Toc62834642)

[4.3.2. *Flowchart* Data Dokumen 37](#_Toc62834643)

[4.3.3. *Flowchart Preprocessing* 37](#_Toc62834644)

[4.3.4. *Flowchart Case folding* 38](#_Toc62834645)

[4.3.5. *Flowchart Cleaning* 39](#_Toc62834646)

[4.3.6. *Flowchart Slang word* 40](#_Toc62834647)

[4.3.7. *Flowchart Stop word* 41](#_Toc62834648)

[4.3.8. *Flowchart* Hapus Spasi 42](#_Toc62834649)

[4.3.9. *Flowchart* Algoritme *Winnowing* 43](#_Toc62834650)

[4.3.10. *Flowchart Rolling Hash* 44](#_Toc62834651)

[4.3.11. *Flowchart Fingerprint* 45](#_Toc62834652)

[4.3.12. *Flowchart Jaccard Similarity* 46](#_Toc62834653)

[4.4. Algoritme Tahapan Metode 47](#_Toc62834654)

[4.4.2. Algoritme Data Dokumen 48](#_Toc62834655)

[4.4.3. Algoritme *Preprocessing* 48](#_Toc62834656)

[4.4.4. Algoritme *Case folding* 48](#_Toc62834657)

[4.4.5. Algoritme *Cleaning* 49](#_Toc62834658)

[4.4.6. Algoritme *Slang word* 49](#_Toc62834659)

[4.4.7. Algoritme *Stop word* 49](#_Toc62834660)

[4.4.8. Algoritme Hapus Spasi 50](#_Toc62834661)

[4.4.9. Algoritme *Winnowing* 50](#_Toc62834662)

[4.4.10. Algoritme *Rolling Hash* 50](#_Toc62834663)

[4.4.11. Algoritme *Fingerprint* 51](#_Toc62834664)

[4.4.12. Algoritme *Jaccard Similarity* 52](#_Toc62834665)

[4.5. Pengujian 52](#_Toc62834666)

[4.5.1. Pengujian dengan nilai *k-gram* dan *w-gram* yang sama 52](#_Toc62834667)

[4.5.2. Pengujian dengan nilai *k-gram* dan *w-gram* yang berbeda 53](#_Toc62834668)

[4.5.3. Pengujian dengan satu *dataset* sebagai data tes. 54](#_Toc62834669)

[4.6. Tampilan Layar Aplikasi 56](#_Toc62834670)

[4.6.1. Tampilan Layar *Home* 56](#_Toc62834671)

[4.6.2. Tampilan Layar *Document Data* 56](#_Toc62834672)

[4.6.3. Tampilan Layar *Document Dataset* 57](#_Toc62834673)

[4.6.4. Tampilan Layar *Slang word* 59](#_Toc62834674)

[4.6.5. Tampilan Layar *Stop word* 60](#_Toc62834675)

[4.6.6. Tampilan Layar *Similarity Check* 61](#_Toc62834676)

[bab v penutup 64](#_Toc62834677)

[5.1. Kesimpulan 64](#_Toc62834678)

[5.2. Saran 64](#_Toc62834679)

[daftar pustaka 65](#_Toc62834680)

# bab i

# pendahuluan

## Latar Belakang

Plagiarisme merupakan tindakan kejahatan yang sudah tidak asing lagi dalam dunia akademik, khususnya di Indonesia. Plagiarisme merupakan perbuatan yang tidak baik disebabkan mengambil karya seperti ide, gagasan, dan pikiran orang lain tanpa izin, tanpa menyatakan karya yang diambilnya merupakan karya orang lain atau tidak mencantumkan sumbernya, dan mengakui bahwa karya itu miliknya. Tindakan tersebut sama seperti tindakan mencuri (Sunardi, Yudhana, & Mukaromah, 2018). Tindakan plagiarisme ini semakin mudah dengan semakin pesatnya perkembangan dunia teknologi, serta semakin mudahnya mendapatkan suatu informasi. Hal ini menyebabkan banyak orang-orang yang mengambil karya orang lain dari internet, tanpa mencantumkan sumbernya.

Dalam dunia pendidikan, khususnya tingkat perguruan tinggi, kesamaan dokumen atau plagiarisme bukanlah suatu hal yang baru. Namun sudah terjadi semenjak teknologi belum secanggih ini. Pelajar maupun mahasiswa ketika mengerjakan tugas-tugas sekolah maupun kuliah sangat jarang mencantumkan sumbernya atau referensi yang digunakan dalam pengerjaan tugas, yang menyebabkan banyaknya kesamaan tugas siswa maupun mahasiswa satu sama lain. Pelajar maupun mahasiswa banyak melakukan manipulasi pembuatan tugas seperti karya ilmiah, makalah, kuliah kerja praktek, maupun skripsi yang menyebabkan kreativitas seseorang tidak tersalurkan dengan baik.

Semakin maraknya tindakan plagiarisme, salah satunya pembuatan skripsi di kampus Budi Luhur, sehingga untuk mengatasi tindakan plagiarisme ini perlu dilakukan pencegahan awal untuk mengurangi praktik plagiarisme. Pendeteksian dapat dilakukan secara manual, namun tidak efektif sebab dibutuhkan waktu yang cukup lama, karena banyaknya dokumen skripsi yang perlu diperiksa nilai *similarity*nya dengan dokumen skripsi yang sudah dipublikasi ke perpustakaan Budi Luhur. Cara menangani hal tersebut yaitu dengan membuat sebuah sistem yang dapat mendeteksi nilai *similarity* dari dokumen skripsi mahasiswa Budi Luhur. Sistem ini di fokuskan untuk pendeteksian awal, sehingga dokumen yang diperiksa dari sistem ini adalah dokumen abstrak, di mana data dokumen abstrak dari skripsi mahasiswa Budi Luhur yang sudah dipublikasi ke perpustakaan dijadikan sebagai *dataset* dari penelitian ini.

Beberapa penelitian sebelumnya telah membuat pendeteksian *similarity* menggunakan beberapa algoritme yang mempunyai fungsi sama sebagai dokumen *fingerprint*, tetapi permasalahan yang sering terjadi pada hasil atau tingkat keakuratan (Sunardi et al., 2018). Penggunaan model *Bayesian* juga telah digunakan, namun tidak sepenuhnya dapat mendeteksi *similarity*. Pada penelitian yang dilakukan oleh (Rahmatulloh, Kurniati, Darmawan, Asyikin, & Witarsyah, 2019) dalam pendeteksian *similarity* dokumen berbahasa indonesia, penelitian ini menggunakan kinerja algoritme *fingerprint* dan *winnowing*, hasil tertinggi pada analisis tes akurasi algoritme *fingerprint* sebesar 92,8% dengan nilai *threshold* 0,1 dan nilai *n-gram* berada di 3, algoritme *winnowing* sebesar 91,8%. hasil yang berbeda ditunjukkan pada tingkat relevansi akurasi dengan topik, hasil akurasi dari algoritme *winnowing* mendapatkan korelasi yang lebih kuat 37,1% dibandingkan *fingerprint* 33,6%.

Pada penelitian ini sistem yang akan dibuat menggunakan metode *N-gram*, *Jaccard Similarity,* dan algoritme *Winnowing*. *N-gram* merupakan algoritme yang akan digunakan untuk mengambil potongan huruf sejumlah n dan mempunyai pengaruh yang tinggi terhadap hasil *similarity*. Algoritme yang akan digunakan untuk mengetahui tingkat kemiripan teks atau dokumen yaitu *Jaccard Similarity* atau sering disebut *Jaccard Coefficient*. *Jaccard Similarity* merupakan algoritme yang berfungsi untuk membandingkan antar dua dokumen dengan menghitung *similarity* atau perbedaan dari dokumen tersebut (Sunardi et al., 2018). Untuk menjalankan *n-gram* dan *jaccard similarity* diperlukan algoritme yang berfungsi sebagai dokumen *fingerprint* dan algoritme yang akan digunakan untuk mendukung hal tersebut yaitu algoritme *winnowing*. Algoritme *winnowing* merupakan algoritme yang dapat digunakan untuk membantu mendeteksi kemiripan kata atau dokumen yang akan digunakan sebagai suatu cara untuk mendeteksi adanya suatu tindak plagiarisme. Diharapkan dengan adanya sistem pendeteksi *similarity* menggunakan metode *n-gram* dan *Jaccard similarity* terhadap algoritme *winnowing* dapat memberikan keakuratan yang tepat dalam mengatasi tindakan suatu plagiarisme yang terjadi di dunia akademik, tepatnya di kampus Budi Luhur.

## Perumusan Masalah

Berdasarkan latar belakang yang telah disebutkan di atas maka didapatkan rumusan masalah yaitu :

1. Bagaimana menerapkan metode *N-gram*, *Jaccard Similarity,* dan algoritme *Winnowing* untuk pemeriksaan kesamaan abstrak pada dokumen skripsi mahasiswa Budi Luhur?
2. Bagaimana mengukur tingkat persentase kesamaan dalam penerapan metode *N-gram*, *Jaccard Similarity,* dan algoritme *Winnowing*?

## Batasan Masalah

Adapun batasan atau ruang lingkup masalah antara lain sebagai berikut :

1. Aplikasi menggunakan bahasa pemrograman PHP.
2. *Platform* yang digunakan hanya berbasis *web.*
3. Deteksi *similarity* hanya dilakukan pada dokumen abstrak.
4. Fitur *import* hanya dapat mengenali file masukan berupa *.pdf.*
5. Hanya dapat mengenali tulisan di dalam file masukan.
6. Waktu pemrosesan meningkat seiring dengan jumlah data yang diproses.
7. Pemeriksaan dokumen hanya dilakukan untuk mendapatkan persentase *similarity* dari dokumen yang diuji, tidak untuk mendapatkan *output* berupa keterangan plagiat atau tidak plagiat.

## Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut :

1. Menerapkan metode *n-gram* untuk mengambil potongan karakter huruf pada dokumen yang akan dicocokan.
2. Menerapkan metode *jaccard similarity* untuk menghitung berapa persen kesamaan pada sebuah dokumen yang diproses menggunakan algoritme *winnowing.*
3. Mengimplementasikan rancangan model yang dibuat ke dalam aplikasi berbasis web.

## Manfaat

Adapun manfaat dari penelitian ini adalah sebagai acuan untuk mendeteksi awal plagiarisme pada dokumen skripsi mahasiswa Budi Luhur yang bersumber dari dokumen abstrak. Sehingga diharapkan tidak terjadi lagi plagiarisme dalam penulisan dokumen skripsi di Universitas Budi Luhur.

## Sistematika Penulisan

Sistematika penulisan penelitian ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Sistematika penulisan tugas akhir ini adalah sebagai berikut:

**BAB I : PENDAHULUAN**

Bagian ini berisi tentang latar belakang, rumusan masalah, batasan masalah, manfaat dan tujuan penelitian, dan juga membahas mengenai sistematika penulisan.

**BAB II : LANDASAN TEORI**

Bagian ini berisi tentang algoritme dan metode yang akan dibahas, serta teori-teori yang berkaitan dengan penelitian ini, yaitu pengertian dan pemahaman plagiarisme, *text mining*, data *mining, preprocessing, n-gram,* *Jaccard Similarity,* Algoritme *Winnowing*, serta studi literatur.

**BAB III : METODOLOGI PENELITIAN**

Bagian ini berisi tentang sumber data penelitian, penerapan atau tahapan metode yang digunakan. Bab ini juga berisi tentang rancangan pengujian dari ekstraksi informasi yang didapat.

**BAB IV : HASIL DAN PEMBAHASAN**

Bagian ini berisi mengenai lingkungan percobaan sistem yang dibuat, implementasi metode, *flowchart* tahapan metode, dan uraian algoritme pada proses, serta analisa pengujian sistem yang telah dibangun apakah data hasil pengelompokan yang didapat sudah sesuai dan relevan.

**BAB V : PENUTUP**

Bagian ini berisi tentang kesimpulan yang dapat ditarik dari penelitian dan saran untuk pengembangan lebih lanjut mengenai topik terkait dalam penelitian berikutnya.

# bab ii

# landasan teori

## 2.1. Data *Mining*

Data *mining* merupakan serangkaian proses untuk menggali informasi dengan melakukan analisa data untuk menemukan suatu pola dari kumpulan data tersebut. Data mining mampu menganalisa data yang besar menjadi ekstraksi berupa pola yang mempunyai arti bagi pendukung keputusan (Gunadi & Sensuse, 2012). Data mining juga bisa disebut knowledge discovery adalah proses pengambilan pola pada data yang akan diproses lalu output tersebut berupa informasi yang sangat penting. Proses yang dilakukan untuk mengekstrak pengetahuan dalam data mining adalah pengenalan pola, clustering, asosiasi, prediksi dan klasifikasi (Fitri, Nurjanah, & Astuti, 2018). Data *mining* memiliki variasi untuk menemukan pola dari ekstraksi sebuah kumpulan sekumpulan data tekstual yang disebut dengan *text mining*. *Text mining* memiliki fokus pada pengolahan data berupa kata atau teks.

## 2.2. *Text Mining*

Dalam jurnal (Februariyanti, 2012), menurut Hearst *text mining* diartikan sebagai penemuan informasi yang baru dan tidak diketahui sebelumnya oleh komputer, dengan secara otomatis mengekstrak informasi dari sumber-sumber yang berbeda. Kunci dari proses ini adalah menggabungkan informasi yang berhasil diekstraksi dari berbagai sumber . Sedangkan menurut Harlian *text mining* memiliki definisi menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen.

*Text mining* merupakan bagian dari *data mining*, yang mana digunakan untuk mendapatkan informasi dari sebuah data atau dokumen berupa sekumpulan teks yang memiliki format yang terstruktur maupun tidak terstruktur dengan jumlah yang besar. Dalam *text mining* memiliki tugas khusus yaitu klasifikasi dan klasterisasi. Sedangkan dalam penerapannya, *text mining* berfungsi untuk mencari pola dalam teks, menganalisis teks agar bisa menghasilkan keluaran berupa informasi yang bermanfaat pada tujuan tertentu. Dikarenakan data yang diproses pada *text mining* merupakan sebuah teks yang tidak terstruktur, maka diperlukan pemilihan teks sebelum dilakukan proses selanjutnya, pada tahap ini dikenal dengan prapemrosesan (*preprocessing*).

## 2.3. Plagiarisme

Dalam jurnal (Sunyoto, 2013). Plagiarisme atau plagiat adalah penjiplakan atau pengambilan karangan, pendapat orang lain dan menjadikannya seolah-olah karangan sendiri. Pendekatan deteksi plagiat terbagi menjadi *intrinsic* dan *external*. Pendekatan *external* terbagi lagi menjadi tiga, yaitu perbandingan teks lengkap, kesamaan kata kunci dan *fingerprinting*. Perbandingan teks lengkap diterapkan untuk membandingkan semua isi dokumen, kesamaan kata kunci bekerja dengan cara mengekstrak dan membandingkan kata kunci antara dokumen, dan *fingerprinting* untuk mendeteksi kemiripan antar dokumen dengan prinsip *hashing.*

Berdasarkan penelitian ini mengkategorikan praktek plagiat berdasarkan cara yang digunakan, diantaranya :

* *Copy Paste plagiarism*, menyalin setiap kata tanpa perubahan.
* *Disguised plagiarism*, tergolong ke dalam praktek menutupi bagian yang disalin, teridentifikasi ke dalam empat teknik, yaitu *shake paste, expansive plagiarism, contractive plagiarism,* dan *mosaic plagiarism.*
* *Technical disguise*, teknik meringkas untuk menyembunyikan konten plagiat dari deteksi otomatis dengan memanfaatkan kelemahan dari metode analisis teks dasar, misal dengan mengganti huruf dengan simbol huruf asing.
* *Undue paraphrasing*, sengaja menuliskan ulang pemikiran asing dengan pemilihan kata dan gaya plagiator dengan menyembunyikan sumber asli.
* *Translated plagiarism*, mengkonversi konten dari satu bahasa ke bahasa lain.
* *Idea plagiarism*, menggunakan ide asing tanpa menyatakan sumber.

*Self plagiarism*, penggunaan sebagian atau keseluruhan tulisan pribadi yang tidak dibenarkan secara ilmiah.

## 2.4. *Preprocessing*

Tahapan *preprocessing* atau praproses merupakan bagian yang sangat penting dalam menyiapkan data, hal ini dikarenakan struktur data yang dihasilkan pada tahap pengumpulan tidak beraturan, sehingga menyebabkan proses menjadi tidak berjalan dengan baik.

Merujuk pada penelitian sebelumnya yang dilakukan oleh Himalatha dalam jurnal (Filcha & Hayaty, 2019) maka pada penelitian ini akan dibahas beberapa tahapan *preprocessing* teks antara lain, *case folding,* menghapus karakter selain a-z, menghapus teks dengan 1 karakter, mengganti *slang word,* menghapus *stop word.*

### 2.4.1. *Case folding*

Pada proses ini bertujuan untuk mengubah semua karakter huruf menjadi huruf kecil (*lowercase*), hal ini dilakukan untuk menyamakan arti dari suatu kata yang sama, apabila penulisan besar kecilnya huruf tidak sama.

### 2.4.2. Menghapus karakter kecuali a sampai z

Pada proses ini dilakukan penghapusan untuk seluruh karakter berupa simbol dan angka, atau menyisakan hanya karakter angka, termasuk menghapus hashtag (#) dan mention (@), hal ini dilakukan karena simbol dan angka dianggap tidak terlalu penting, tetapi jika ini diperlukan, maka proses ini dihilangkan.

### 2.4.3. Mengganti *slang word*

Teks yang tidak terstruktur membuat sebuah teks terkadang tidak sesuai dengan ejaan bahasa Indonesia yang baku (EYD) pada konteks ini, kata yang tidak baku disebut dengan *slang word,* untuk mendapatkan informasi dari teks agar maksimal, kata-kata tidak baku, baik kata gaul, singkatan atau yang lain sebanyak mungkin ditampung ke dalam kamus *slang word,* untuk kemudian dilakukan *replace* supaya menjadi kata dengan bahasa Indonesia yang baku sesuai EYD.

### 2.4.4. Menghapus *stop word*

*Stop word* merupakan salah satu kata yang diabaikan dalam pemrosesan, Ringkasnya *stop word* adalah kata hubung atau kata sambung dalam sebuah kalimat, seperti “di”, “pada”, “karena”, “sebuah”, “oleh”, dll. Sebelum melakukan proses penghapusan *stop word*, kumpulkan daftar atau kamus *stop word* yang diberi nama *stoplist*. Kemudian lakukan perbandinngan antara sebuah teks dengan *stoplist*. Jika terdapat kata-kata yang terdapat dalam *stoplist*, maka kata tersebut dihilangkan. Untuk *stoplist* dalam bahasa Indonesia, datanya bersumber dari (Tala, 2003).

## 2.5. *N-Gram*

Dalam jurnal (Sunardi et al., 2018)menjelaskan bahwa, *n-gram* merupakan metode yang yang dilakukan dengan mengambil rangkaian *substring* dari *string* sejumlah n (rangkaian token sepanjang n). Metode *n-gram* sering digunakan pada teknik analisis statistik dan juga bahasa. *N-gram* paling banyak digunakan dalam teks mining (pengolahan kata) dan pengolahan bahasa. Dalam mendeteksi plagiarisme, *n-gram* digunakan untuk mengambil potongan-potongan karakter huruf atau pemisahan *string* sepanjang n dari sebuah kata atau dokumen secara berkelanjutan (kontinuitas) hingga bergeser sesuai dengan *offsite* yang diberikan atau akhir dari suatu kata atau dokumen.

*N-gram* dalam deteksi plagiarisme sangat mempengaruhi tingkat akurasi atau tingkat *similarity* dari sebuah dokumen yang dibandingkan, maka teknik *n-gram* dipadukan dengan pendekatan statistika untuk memperoleh *similarity* dari antar dokumen seperti *Simple Matching, Cosine Similarity, Jaccard Similarity* dan *Dice Coefficient*. Ada 2 (dua) teknik *n-gram* yaitu membagi *string* menjadi suatu set *substring* dengan panjang n (*overlapping n-gram*) dan mengecek untuk membentuk substring yang memiliki struktur yang sama.

## 2.6. *Jaccard Similarity*

Dalam jurnal (Sunardi et al., 2018)menjelaskan bahwa, *Jaccard Similarit*y dan *Jaccard Coefficient* merupakan algoritme yang fungsinya untuk membandingkan dua sampel yaitu dokumen yang satu dengan yang lainnya berdasarkan kata yang dimilikinya. *Jaccard similarity* biasanya digunakan untuk membandingkan dokumen dan menghitung nilai kemiripan (*similarity*) dari dua buah objek atau dokumen. *Jaccard similarity* dapat dirumuskan sebagai berikut:

*Similarity* (X,Y)=…(2.1)

Dimana:

X = *Dataset*

Y = Data tes

Rumus 1 merupakan dari rumus *Jaccard Similarity* atau *Jaccard Coefficient* yang digunakan untuk mencari persamaan dan perbedaan pada dua sampel. sebagai contoh diketahui X “Magister Teknik Informatika Yogyakarta”, dan Y “Magister Teknik Sipil Yogyakarta”. Maka akan menghasilkan nilainya:

*Similarity* (X,Y)= …..(2.2)

*Similarity* (X,Y)== 0,6 x 100% = 60%

Dari dua contoh sampel di atas setelah dihitung kesamaannya menggunakan *Jaccard similarity*, bahwa kedua sampel tersebut memiliki kesamaan atau kemiripan sebesar 60%.

Ukuran tingkat kesamaan:

* a. 0% : dua dokumen tidak memiliki kesamaan
* b. <15% : memiliki sedikit kesamaan
* c. 15-50% : termasuk dalam kategori plagiarisme sedang
* d. 50% : mendeteksi adanya plagiarisme
* e. 100% : dokumen tersebut plagiarism

## 2.7. Algoritme *Winnowing*

Dalam jurnal (Sunardi et al., 2018)menjelaskan bahwa, Algoritme *winnowing* merupakan salah satu algoritme yang berfungsi sebagai dokumen *fingerprint* atau algoritme yang digunakan untuk mendeteksi tindakan plagiarisme dengan menggunakan teknik *hashing*. Input dari algoritme *winnowing* berupa dokumen teks, dan akan menghasilkan keluaran berupa kumpulan nilai hash yang terbentuk dari perhitungan ASCII pada setiap karakter. Dan nilai-nilai *hash* yang akan digunakan sebagai *fingerprint* untuk mendeteksi adanya suatu tindakan plagiarisme. Gambar 2.1 merupakan konsep algoritme *winnowing*.

*Preprocessing*

Membentuk *n-gram*

Penghitung *substring* menjadi nilai *hash*

*Window*

*Fingerprint*

Gambar 2.1 Konsep Algoritme *Winnowing*(Sunardi et al., 2018)

Konsep dari algoritme *winnowing* yang ditujukan pada gambar 2.1 yaitu menghapus karakter yang tidak relevan, membentuk *n-gram* dengan panjang n, menghitung nilai *hash*, membentuk nilai *window*, dan memilih nilai *hash* sebagai dokumen *fingerprint*. Nilai *hash* didapatkan dari rumus *Rolling hash*. *Rolling hash* adalah metode *hashing* yang digunakan untuk mencari nilai *hash* dari rangkaian *grams* yang telah terbentuk dan memberikan kemampuan untuk menghitung nilai tanpa mengulangi seluruh *string*. Nilai *hash* merupakan nilai numerik yang dibentuk dari kode ASCII[15]. Berikut rumus *rolling hash*



Dimana:

H (C1..Cl) = nilai hash

Cl = nilai ASCII karakter ke -1 pada string

l = panjang string

b = nilai basis hash

Perhitungan awal pada rangkaian *n-gram* paling awal dihitung menggunakan rumus nomor 2, dan rangkaian *n-gram* berikutnya sampai rangkaian *gram* terakhir dihitung menggunakan rumus nomor 3, sehingga proses akan jauh lebih cepat karena tidak menghitung lagi dari awal. Algoritme *winnowing* merupakan ekstensi dari algoritme *Rabin-karp*, proses yang dilakukan hampir sama hanya pada algoritme *winnowing* ditambahkan dengan konsep *window*.

## 2.8. Pengujian

Dalam mengevaluasi performa dari metode yang diusulkan, perlu adanya sebuah pengujian dengan membandingkan hasil dengan data asli yang sudah valid. Dalam penelitian ini, dilakukan pengujian secara *recall* yaitu menentukan tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi sebenarnya.

## 2.9. Studi Literatur

Berdasarkan landasan teori yang telah dijelaskan , terdapat penelitian yang sudah ada sebelumnya, yang dirangkum dalam Tabel 2.4 berikut :

Tabel 2.4 Studi Literatur

| **No** | **Penulis** | **Judul** | **Jurnal** | **Deskripsi** |
| --- | --- | --- | --- | --- |
| 1 | Abdullah, Ibnu Aribowo, Eko | Rancang Bangun Aplikasi Pengecekan Kemiripan Judul Skripsi Dengan Metode *Cosine Similarity* (Studi Kasus : Program Studi Teknik Informatika Uad) | JSTIE (Jurnal Sarjana Teknik Informatika) (E-Journal), Tahun 2018, *e-ISSN 2338-5197* | Melakukan pengecekan kemiripan judul skripsi di teknik informatika UAD. Dengan menggunakan metodologi pengembangan sistem yaitu *waterfall model*. Metode *waterfall* dimulai dengan tahap menentukan *system requirement specification,* tahap perancangan sistem, tahap implementasi, dan tahap pengujian sistem. Sistem diuji dengan 2 metode yaitu *Black Box Test* dan *Usability Test.* Pengujian dengan metode *Black Box Test* dihasilkan bahwa 100% sistem sudah berjalan dengan baik, layak digunakan dan sesuai kebutuhan. Sedangkan pengujian dengan metode *Usability Test* 0 % responden mengatakan aplikasi tidak diterima, 20 % responden memberikan penilaian marginal dan 80 % responden menyatakan sistem aplikasi bisa diterima.(Abdullah & Aribowo, 2018) |
| 2 | Khidfi, Muhammad Nur Sari, Jayanti Yusma | Rancang bangun aplikasi pendeteksian kesamaan pada dokumen teks menggunakan algoritme *Enhaced confix stripping* dan algoritme *winnowing* | Jurusan Teknik Informatika, Fakultas Teknik Universitas Halu Oleo, Kendari, Tahun 2018, *ISSN: 2502-8928* | Melakukan pengecekan plagiarisme pada dokumen tugas mahasiswa. Pada penelitian ini menggunakan algoritme *Enhanced Confix Stripping (ECS) Stemmer* untuk proses *stemming* teks dan algoritme *Winnowing* untuk menghitung tingkat kesamaan *(similarity)* antar dokumen. Dengan ditetapkanya nilai *gram* dan *window* pada perhitungan algoritme *Winnowing,* memudahkan user dalam menentukan nilai *similarity* yang akurat. Dari hasil pengujian 5 pasang bab 1 tugas akhir mahasiswa yang berkategori sama menghasilkan nilai *similarity* sekitar 45-20% .(Khidfi & Sari, 2018) |
| 3 | Ariantini, Dewa Ayu Rai Lumenta, Arie S. M. Jacobus, Agustinus | Pengukuran Kemiripan Dokumen Teks Bahasa Indonesia Menggunakan Metode *Cosine Similarity* | Jurnal Teknik Informatika, Tahun 2016, *ISSN : 2301-8364* | Melakukan pengecekan plagiat suatu karya tulis. Tahap pengujian sistem dilakukan untuk evaluasi perangkat lunak yang dibagun dengan hasil yang diharapkan, dapat dilakukan dengan cara membandingkan nilai hasil prediksi dan *actual*, nilai prediksi diperoleh dari sistem, kemudian nilai *actual* diperoleh dari nilai yang diperiksa secara manual, kemudian nilai yang sudah diketahui prediksi dan actual dengan menggunakan data *dummy* pada sistem deteksi kemiripan dokumen sudah berhasil diterapkan dan sudah bisa digunakan untuk mengukur tingkat kemiripan dokumen. (Ariantini, Lumenta, & Jacobus, 2016) |
| 4 | Fataruba, Firmansyah | Penerapan Metode *Cosine Similarity* Untuk Pengecekan Kemiripan Jawaban Ujian Siswa | Jurnal Mahasiswa Teknik Informatika, Tahun 2018 | Melakukan pengecekan kemiripan jawaban *essay* siswa. Tujuan dari penelitian ini adalah untuk membantu pengajar memberikan nilai yang objektif dengan menggunakan metode *cosine similarity* pada sistem agar dapat melakukan penilaian jawaban *essay* dengan membandingkan kunci jawaban pengajar dengan jawaban peserta didik. Dan juga dapat membandingkan tingkat kemiripan antara siswa satu dengan siswa yang lain. *Cosine similarity* adalah metode untuk menghitung kesamaan dan kemiripan dari dua dokumen, dokumen yang di hitung perbandingan dalam penelitian ini adalah kunci jawaban pengajar dan jawaban siswa. Berdasarkan hasil penelitian bahwa sistem pengecekan kemiripan jawaban siswa menggunakan metode *cosine similarity* telah berjalan dengan baik untuk ujian *essay* Biologi Hasil uji coba menunjukkan kesesuaian nilai sistem dengan nilai yang diberikan oleh pengajar tingkat akurasinya 80% menggunakan *Confusion Matrix.*(Fataruba, 2018) |
| 5 | Ulfa, Nur Fadillah Mustikasari, Metty Bastian, Irwan | Pendeteksian tingkat similaritas dokumen berbasis web menggunakan algoritme *winnowing* | Konferensi Nasional Teknologi Informasi dan Komunikasi (KNASTIK), Tahun 2016, *ISSN: 2338-7718* | Melakukan pendeteksian *plagiarisme* menggunakan teknik *winnowing.* Penelitian ini bertujuan untuk membangun sebuah aplikasi berbasis website menggunakan algoritme *Winnowing* untuk mencari kesamaan pada dua dokumen teks yang diuji. Algoritme yang digunakan untuk mencari nilai *hash* dalam *Winnowing* adalah *rolling hash*. Nilai *hash* merupakan nilai numerik yang terbentuk dari perhitungan ASCII (*American Standard Code for Information Interchange*) tiap karakter. Pada hasil eksperimen pengukuran kemiripan ini disimpulkan bahwa semakin kecil tingkat *persentase* kesamaan dokumen teks yang diuji, maka tingkat kemiripan dokumen kecil dan tidak termasuk plagiat, tetapi jika hasil dari pengujian pada dua dokumen semakin besar, maka dokumen tersebut mempunyai tingkat kemiripan yang tinggi dan tindakan tersebut dianggap plagiat. Penelitian ini juga menambahkan perbandingan nilai *k-gram*, basis (bilangan prima), nilai *window*, keterangan *persentase*, dan kategori *plagiarisme.*(Ulfa, Mustikasari, & Bastian, 2016) |
| 6 | Priambodo, Joko | Pendeteksian Plagiarisme Menggunakan Algoritme *Rabin-Karp* dengan Metode *Rolling Hash* | Jurnal Informatika Universitas Pamulang, Tahun 2018, *ISSN 2541-1004* | Melakukan pendeteksian kemiripan suatu dokumen. Pendeteksian plagiarisme menggunakan algoritme *Rabin-Karp* dengan metode *rolling hash* dari hasil pengujian 30 dokumen teks yang sudah dijelaskan pada bab sebelumnya menghasilkan tingkat akurasi yang terbesar yaitu 47.58%. Hasil persentase tersebut termasuk dalam kategori tingkat plagiat 15-50%, berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang. Sedangkan tingkat akurasi yang terkecil yaitu 19.28%, berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang. Berdasarkan analisis proses pendeteksian tingkat plagiarisme menggunakan algoritme raibin-karp dengan metode rolling hash bisa membaca karakter berupa huruf, simbol seperti titik (.), koma (,), dan lain-lain. (Priambodo, 2018) |
| 7 | Nurdin, Nurdin Rizal, Rizal Rizwan, Rizwan | Pendeteksian Dokumen Plagiarisme dengan Menggunakan Metode *Weight Tree* | Telematika, Tahun 2019, *ISSN : 1979 – 925X e-ISSN : 2442 - 4528* | Melakukan pendeteksian plagiat suatu dokumen dengan menggunakan algoritme *Weight Tree,* algoritme *Weight Tree* yaitu sebuah metode untuk melakukan klasifikasi kemiripan dokumen berdasarkan bobot dari dokumen. Tujuan penelitian ini adalah untuk membangun sebuah sistem pendeteksian kemiripan dari dua dokumen teks yang berbeda untuk jenis dokumen teks berbahasa indonesia dengan format file dokumen yaitu: doc, docx, pdf, rtf. Tahapan yang dilakukan pada penelitian ini terdiri dari pengumpulan data, perancangan sistem, pembuatan aplikasi dan pengujian terhadap aplikasi. Hasil pengujian sistem dapat dikategorikan sebagai sistem pendeteksian atau pengetesan kemiripan dokumen. Nilai rata-rata persentase kemiripan dalam pengujian sistem ini adalah 71,60%. Sistem yang di bangun ini berhasil dengan tingkat keakuratan mencapai 90%. Algoritme *Weight Tree* yang diterapkan pada sistem ini terbukti mampu mengidentifikasi dengan baik kemiripan dokumen plagiarisme. (Nurdin, Rizal, & Rizwan, 2019) |
| 8 | Rinartha, Komang | Pemodelan Penilaian *Essay* Otomatis Secara *Realtime* Menggunakan Kombinasi *Text Stemming* Dan *Cosine Similarity* | Konferensi Nasional Sistem & Informatika, Tahun 2017 | Melakukan pengecekan ujian siswa yang berbentuk *essay,* soal ujian yang akan dikembangkan adalah soal ujian berbasis web yang bisa memberikan nilai secara otomatis. Pemberian nilai otomatis dilakukan dengan menggunakan *stemming Porter* yang dikombinasikan dengan *Cosine similarity*. Algoritme *Porter* digunakan untuk mencari kata dasar dari kata-kata yang digunakan dalam kunci jawaban. Hasil dari *ekstraksi* kata dasar pada kunci jawaban dan jawaban siswa akan dibandingkan dengan menggunakan *Cosine similarity*. Proses pemodelan dilakukan dengan mengembangkan dari model yang telah ada dengan menyesuaikan konsep *real* *time* yang akan direncanakan. Model yang didapatkan adalah proses *text stemming* dan proses *ekstrak* kata dari jawaban dilakukan pada komputer *client* (pengajar dan siswa) dan tidak membebankan ke server. (Rinartha, 2017) |
| 9 | Sunardi, Sunardi Yudhana, Anton Mukaromah, Iif Alfiatul | Implementasi Deteksi Plagiarisme Menggunakan Metode *N-Gram* Dan *Jaccard Similarity* Terhadap Algoritme *Winnowing* | *Transmisi*, Tahun 2018, *ISSN 2407–6422* | Melakukan pengecekan kemiripan suatu dokumen yang terjadi akibat banyaknya orang mengerjakan tugas dengan meng*copy paste*  dari internet, tanpa menyebutkan sumbernya. Hal tersebut bisa diatasi secara dini menggunakan metode *n-gram* untuk mengambil potongan karakter huruf pada dokumen yang akan dicocokan dan *Jaccard similarity* untuk menghitung berapa persen kesamaan pada sebuah dokumen yang diproses menggunakan algoritme *winnowing.* Algoritme *winnowing* merupakan algoritme yang digunakan untuk mencocokan karakter huruf atau angka pada dua dokumen dengan teknik *hashing.*(Sunardi et al., 2018) |
| 10 | Filcha, Asvarizal Hayaty, Mardhiya | Implementasi Algoritme *Rabin-Karp* untuk Mendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa | JUITA : Jurnal Informatika, Tahun 2019, *ISSN: 2579-9801* | Melakukan pengecekan plagiarisme pada tugas mahasiswa, ini terjadi karena tugas yang dikumpulkan dalam bentuk digital. Metode yang dapat digunakan adalah menggunakan algoritme *Rabin-Karp.* Algoritme *Rabin- Karp* memiliki keunggulan pencarian *string* dengan pola yang panjang. Algoritme *Rabin-karp* dalam sistem ini memiliki langkah - langkah *text preprocessing* yang terdiri *case folding, tokenizing, punctuation removal, stop word removal* *and* *stemming*. Hasil dari *text preprocessing* inilah yang akan diproses menggunakan algoritme *Rabin-karp*. Hasil dari metode ini dihitung menggunakan *dice coefficient*. Perhitungan akurasi dengan melakukan 20 perbandingan antara sistem pendeteksi plagiarisme dan *software* Plagiarisme Checker X menggunakan *confusion matrix* menghasilkan tingkat keakuratan sebesar 90%. (Filcha & Hayaty, 2019) |
| 11 | Anggara, Yudha | Deteksi plagiarisme dokumen bahasa indonesia dengan algoritme *jaro-winkler Distance* | Program studi teknik informatika universitas muhammadiyah jember, Tahun 2016 | Melakukan pengecekan tingkat kesamaan antar dokumen. Pada penelitian ini akan dibuat sebuah aplikasi untuk menghitung tingkat kesamaan dokumen dengan algoritme *Jaro-Winkler distance*. Tujuan dari penerapan algoritme ini adalah membandingkan kesamaan antar dokumen teks berbahasa Indonesia, sehingga dapat ditentukan sebuah dokumen tersebut plagiat atau tidak. Pengujian terhadap aplikasi menggunakan data abstrak jurnal skripsi. Dari hasil analisis dokumen uji 1 memiliki kesamaan tertinggi dengan dokumen nomor 55 dengan nilai 86,267%, dokumen 2 memiliki kesamaan tertinggi dengan dokumen 66 dengan nilai 95,922% dan dokumen 3 memiliki kesamaan tertinggi dengan dokumen 23 dengan nilai 98.361 %. (Anggara, 2016) |
| 12 | Alamanda, Rio Suhery, Cucu Brianorman, Yulrio | Aplikasi pendeteksi plagiat terhadap karya tulis berbasis web menggunakan natural language processing dan algoritme knuth-morris-pratt | Jurnal *Coding*, Sistem Komputer Untan, Tahun 2016, *ISSN : 2338-493x* | Melakukan pendeteksian kemiripan jurnal dimulai dari algoritme rangkuman, yaitu *TF/IDF*. Kemudian proses dilanjutkan pada proses *Tokenizing, Filtering* dan *Stemming*. Selanjutnya dilanjutkan dengan Algoritme *Knuth-Morris-Pratt* dan proses menghitung persentase kemiripan dengan *Cosine Similarity*. Dari hasil pengujian yang telah dilakukan, rata-rata persentase kemiripan yang dihasilkan dari pendeteksian tanpa menggunakan proses *TF-IDF* sebesar 45,98%, nilai tersebut lebih rendah dibandingkan dengan pendeteksian menggunakan proses *TF-IDF*, *Tokenizing, Filtering* dan *Stemming* yang menghasilkan persentase kemiripan 41,09%. Sedangkan tanpa menggunakan proses *Stemming* yaitu 40,58% serta tanpa menggunakan proses *TF-IDF* dan *Stemming* sebesar 40,54% (Alamanda, Suhery, & Brianorman, 2016) |
| 13 | Shu, Kai Wang, Suhang Lee, Dongwon Liu, Huan | *Mining disinformation and fake news: Concepts, methods, and recent advancements* | *International Journal of Computer Applications,* Tahun 2020 | Melakukan pendeteksian membutuhkan metode untuk menyelesaikan masalahnya. Di penelitian ini membahas metode pengukuran kemiripan dokumen menggunakan algoritme berbasis string yang berbeda (*String Based). String Based* ini dibagi menjadi 2, yaitu *Character Based* dan  *Term Based.*  Dari 2 pembagian ini ada beberapa metode yang bisa digunakan, sesuai dengan kebutuhan. Untuk *Character Based*  pembagianya ada beberapa metode diantaranya, *Longest Common SubString (LCS), Longest Common SubString (LCS), Jaro, Jaro–Winkler, Needleman-Wunsch, Smith-Waterman, N-gram.* Untuk *Term Based* pembagianya ada, *Block Distance, Cosine similarity, Dice’s coefficient, Euclidean distance, Jaccard similarity, Matching Coefficient, Overlap coefficient.* |
| 14 | Wang, Jiapeng Dong, Yihong | *Measurement of Text Similarity: A Survey* | *Information*, Tahun 2020 | Melakukan pendeteksian kemiripan suatu dokumen pasti menggunakan suatu metode untuk penyelesaian suatu kasus. Penelitian ini menganalisis keuntungan dan kerugian dari suatu metode yang digunakan, mengembangkan sistem deskripsi klasifikasi yang lebih komprehensif dari algoritme pengukuran kesamaan teks, dan meringkas arah pengembangan di masa depan. Dengan tujuan memberikan referensi bagi penelitian dan aplikasi terkait, metode pengukuran kemiripan teks dijelaskan melalui dua aspek yaitu jarak teks dan representasi teks. Jarak teks dapat dibagi menjadi jarak panjang, jarak distribusi, dan jarak semantik; Representasi teks dibagi menjadi teks berbasis string, berbasis korpus, teks semantik tunggal, teks multi-semantik, dan representasi berbasis struktur grafik. Terakhir, perkembangan kemiripan teks juga dirangkum pada bagian pembahasan. (Shu, Wang, Lee, & Liu, 2020) |
| 15 | Mujilahwati | *PreProcessing*  *Text Mining*  Pada *Data*  *Twitter* | Seminar  Nasional  Teknologi  Informasi  Dan Komunikasi,  Tahun 2016,  *ISSN 2089-*  *9815* | Membahas teknik penanganan data pram pemrosesan data komentar dari Twitter. Hasil ekstraksi ini kemudian diujikan pada pengklasifikasian layanan sebuah perusahaan telekomunikasi serta didapatkan hasil akurasi mencapai 93,11% dengan 450 data uji. |

# bab iii

# metodologi penelitian

## 3.1. Data Penelitian

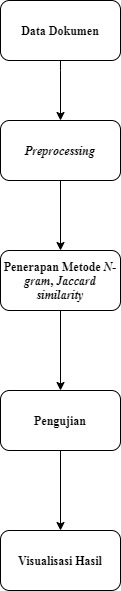
*Dataset* atau sumber data yang digunakan dalam penelitian ini bersumber dari data dokumen abstrak skripsi mahasiswa Budi Luhur. Data tersebut kemudian diolah melalui tahap *preprocessing* sehingga data tersebut bisa digunakan sebagai *dataset*. Berikut adalah contoh *record* dari data dokumen abstrak skripsi mahasiswa Budi Luhur hasil *preprocessing* dapat dilihat pada Tabel 3.1 berikut :

Tabel 3.1 Sampel Data Abstrak

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **nim** | **uploaddate** | **file\_size** | **content** |
| 1 | 1511500025 | 29-12-20 | 367036 | implementasi algoritme multinomial naive bayes classifier tf ltf stemming nazief adriani klasifikasi sentimen alam interview twitter rausal valino adjir humaika resources consulting konsultan human resources department hrd management depan percaya naung….. |
| 2 | 1511500082 | 29-12-20 | 23007 | implementasi algoritme depth first search maze based level generator prototype game d top down shooter preposterous defiance bas mobile fadel achmad assegaf video game hibur suka dewasa video game hilang jenuh jalan aktivitas hari latih konsentrasi fokus…. |
| 3 | 1511500132 | 29-12-20 | 41297 | implementasi aman file algoritme advanced encryption standard aes bit bas web smk pgri ciledug imbar yuli hartoko dikarenakan kembang komputer aman komputer simpan tukar sifat rahasia smk pgri ciledug instansi gerak tangerang file soal uji file sifat rahasia……. |
| 4 | 1511500157 | 29-12-20 | 177537 | implementasi augmented reality metode marker based tracking briefing terima calon karyawan magang suwiwi deddy rivaldy teliti tuju kembang aplikasi augmented reality briefing media kertas sewiwi briefing isi tulis jarang calon karyawan magang bingung baya…. |
| 5 | 1511500199 | 29-12-20 | 99221 | aplikasi aman file algoritme advanced encryption standard aes bas web toko khayangan outdoor yayat khayangan outdoor gerak jual alat daki buku ketik ulang komputer file rentan curi rusak kam guna mesti simpan sadap susup tanggung curi pindah simpan komput…. |

## 3.2. Penerapan Metode

Untuk membangun sistem pendeteksi plagiarisme menggunakan metode *n-gram* dan j*accard similarity* terhadap algoritme *winnowing,* terdapat beberapa tahapan yang menjadi rancangan utama, rancangan ini sebagai gambaran proses tahapan awal hingga akhir sistem berjalan, yang terdapat pada Gambar 3.1 berikut.



Gambar 3.1 Tahapan Metode

Pada Gambar 3.1, proses *upload* dokumen dilakukan untuk mendapatkan *dataset* dan proses pengujian *similarity,* dalam penelitian ini *dataset* diperoleh dari data dokumen abstrak skripsi mahasiswa Budi Luhur. Kemudian dokumen abstrak dalam bentuk *pdf*  ini di­*upload* ke dalam sistem agar bisa di *convert* menjadi teks, lalu di simpan ke *database* untuk dilakukan proses *preprocessing.* Agar bisa dilakukan proses *case folding,* menghilangkan karakter, *slang word* dan *stop word* sehingga dokumen ini bisa digunakan sebagai *dataset.* Kemudian untuk proses pengecekan *similarity,* pertama dilakukan proses *scan* dokumen abstrak berpentuk *pdf* yang di *scan* melalui sistem, lalu sistem meng*convert* dokumen *pdf* tadi menjadi teks, kemudian masuk ke tahap implementasi metode untuk mengecek tingkat *similarity* dari dokumen yang di*scan* tadi.

### 3.2.1. Data Dokumen

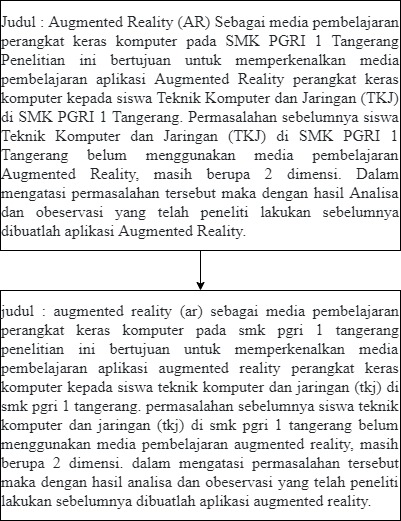
Pada tahap ini dilakukan proses pengumpulan data dokumen abstrak dari skripsi mahasiswa Budi Luhur. Dimana data dokumen ini ada yang dijadikan sebagai *dataset* dan ada juga yang menjadi data tes. *Dataset* ini berasal dari dokumen abstrak skripsi mahasiswa yang sudah di publish di perpustakaan Budi Luhur. Sedangkan data tes berasal dari dokumen abstrak mahasiswa yang akan menjalankan skripsi, sehingga mahasiswa bisa melakukan pengecekan *similarity* di dokumen abstraknya.

### 3.2.2. *Preprocessing*

Pada tahapan *preprocessing* ini, dilakukan beberapa proses untuk mendapatkan *dataset* yang bersih, sehingga proses pengujian tingkat *similarity* lebih tepat dan akurat. Proses *preprocessing* yang digunakan pada penelitian ini diantaranya yaitu :

1. *Case Folding*

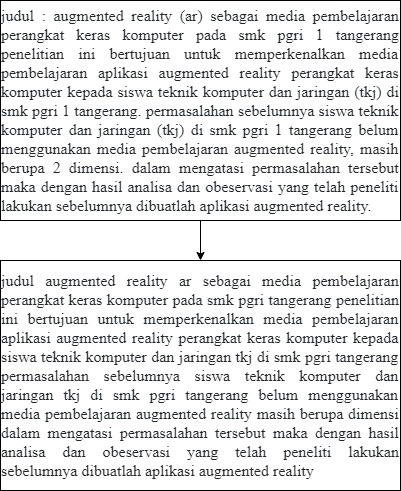
Pada Gambar 3.2 dilakukan penyetaraan kata yang mengandung huruf besar untuk diubah menjadi huruf kecil, misalnya Tujuan menjadi tujuan, Adapun menjadi adapun, dan seterusnya.



Gambar 3.2 *Case folding*

1. Menghapus karakter selain a-z

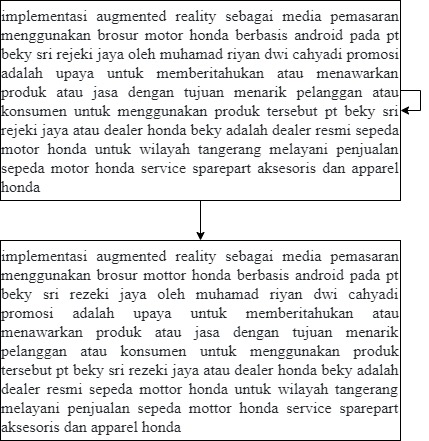
Pada Gambar 3.3 menjelaskan bahwa karakter selain a sampai z atau karakter selain huruf, karakter tersebut dihilangkan, seperti contoh 8266, / (garis miring), dan seterusnya sehingga hanya menyisakan karakter huruf saja.



Gambar 3.3 Menghapus karakter selain a-z

1. Mengganti *Slang word*

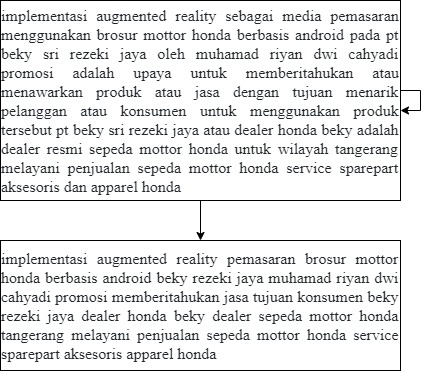
Pada Gambar 3.4 mengganti kata tidak baku seperti “analisa” dan “rejeki”, kata yang tidak baku tersebut biasanya berupa singkatan atau bahasa yang kekinian, untuk itu supaya kata-kata dalam teks setara dengan EYD, maka kata tersebut digantikan dengan kata baku yang seharusnya yaitu menjadi “analisis” dan “rezeki”, pergantian kata-kata ini berdasarkan kamus yang terdapat dalam library *slang word*.



Gambar 3.5 Mengganti *Slang word*

1. Menghapus *Stop word*

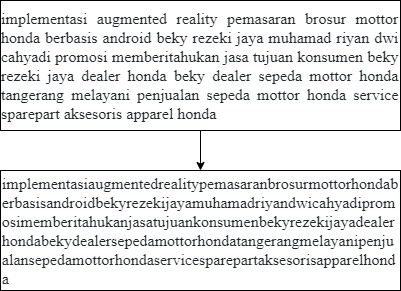
Dalam penelitian Tala, (2003), terdapat kumpulan kata-kata yang termasuk ke dalam *stoplist* yaitu kata umum yang dianggap tidak terlalu memiliki makna yang penting dan kemunculan kata ini sangat tinggi frekuensinya, seperti pada contoh teks di bawah ini yaitu pada kata “untuk”, “dan”, “bagi”, kata tersebut dihilangkan karena masuk ke dalam daftar *stop word*.



Gambar 3.6 Menghapus *Stop word*

1. Menghapus Spasi

Pada Gambar 3.7 menghapus spasi dari setiap kata, untuk melakukan proses selanjutnya, yaitu pada proses pencarian tingkat *similarity* sebuah dokumen.



Gambar 4.7 Menghapus Spasi

### 3.2.3. *N-Gram*

*N-gram* adalah *substring* penggabungan karakter sejumlah *k* pada teks dokumen. Dalam menentukan hasil deteksi plagiarisme dengan menggunakan metode *n-gram*,dimana dokumen atau sekumpulan kata akan diproses dan akan dibentuk sebanyak *n-gram* atau memisahkan string sepanjang n yang akan dihitung pergeserannya secara terus menerus ke depan sejumlah nilai n sampai akhir dokumen. Sebagai contoh *n-gram* dari kalimat **N = Sortir berbasis Arduino**, dengan nilai N=3 maka (sor ort rti tir irb rbe ber erb rba bas asi sis isa sar ard rdu dui uin ino).

### 3.2.4. Algoritme *Winnowing*

*Winnowing* adalah algoritme yangdigunakan untuk melakukan proses pengecekan kesamaan kata (*document fingerprinting*) untuk mengidentifikasi tingkat *similarity.*

1. *Rolling Hash*

Pada tahapan ini dilakukan perhitungan *rolling hash* untuk mencari nilai *hash* dari setiap *string* yang sudah dipotong pada tahapan *n-gram*. Setiap *string* diubah menjadi ASCII lalu dihitung dengan rumus *hash* dibawah:

…(3.1)

c1 = nilai ascii dari huruf pertama dari satu buah *k*-*gram*

b = bilangan prima

k = nilai *k-gram*

Setelah hasil dari *hash* pertama didapatkan selanjutnya digunakan rumus *rolling hash* untuk menghitung nilai *hash* ke dua dan seterus nya tanpa menghitung ASCII yang sudah di hitung sebelumnya.

..(3.2)

Hasil dari perhitungan sebelumnya dikurangi hasil *hash* dari huruf pertama *gram* sebelumnya lalu dikali dengan hasil *hash* dari perhitungan ASCII terakhir dari *gram* sekarang.

### Pembentukan *Window*

Setelah nilai *hash* ditemukan semua maka langkah selanjutnya adalah pembentukan *window*. Nilai dari *rolling hash* akan dikelompokan berdasarkan nilai *wgram*. Pada kasus ini nilai *window* yang dipergunakan adalah w=3.

1. Nilai *Fingerprint*

Proses selanjutnya setelah pembentukan *window* adalah mencari nilai *fingerprint.* Nilai *fingerprint* ditentukan berdasarkan nilai terkecil dari setiap *window* yang ada.

1. *Jaccard Similarity*

Pada tahapan ini untuk mencari nilai *similarity* pada sebuah dokumen dengan rumus *Jaccard similarity* dengan melihat *irisan* dan *union* dari *fingerprint* antar dua dokumen. Berikut adalah rumus *jaccard similarity* :

*Similarity* (X,Y) =x 100% ……………………(3.3)

X = *Dataset*

Y = Data Tes

Dimana *irisan* dari dokumen X dan Y dibagi dengan *union* dari dokumen X dan Y kemudian di kali 100% maka menghasilkan persentase *similarity* dari dokumen X dan Y.

## 3.3. Rancangan Pengujian

Pengujian dilakukan untuk mengetahui tingkat *similarity* dari sebuah dokumen. Pada penelitian ini digunakan metode pengujian *Black box Testing* dalam melakukan proses pengujian program. *Black Box Testing* merupakan pengujian yang dapat dilakukan dengan melakukan pengamatan, pada hasil eksekusi melalui beberapa data tes dan memeriksa fungsional yang terdapat pada perangkat lunak. Pada langkah pengujian ini akan dijelaskan mengenai proses yang berjalan pada sistem yang dibangun, mulai dari *upload* dokumen, menjadikan dokumen sebagai *dataset* dan data tes sampai menampilkan hasil perhitungan tingkat *similarity.*

# bab iV

# hasil dan pembahasan

## 4.1. Lingkungan Percobaan

Agar sistem yang telah dikembangkan dapat berjalan dengan semestinya, dibutuhkanlah perangkat dengan spesifikasi tertentu, adapun dalam penelitian ini menggunakan spesifikasi perangkat diantaranya:

### 4.1.1. Spesifikasi Perangkat Keras

Perangkat keras yang mendukung aplikasi ini berjalan dengan baik sebagai berikut:

1. *Processor* : Intel(R) Pentium(R) CPU 2020M @ 2.40GHz
2. *RAM* : 6 GB DDR3
3. *Harddisk* : 500 GB
4. *VGA* : AMD HD8570 2GB dedicated

### 4.1.2. Spesifikasi Perangkat Lunak

Perangkat lunak yang mendukung aplikasi ini berjalan dengan baik sebagai berikut:

1. Sistem Operasi : Windows 7 Ultimate 64-bit
2. IDE : Visual Studio Code v.1.38.1
3. DBMS : Mysql Database
4. Web Server : Apache (XAMPP v3.2.2, PHP versi 7.3.0)
5. Browser : Google Chrome

## 4.2. Implementasi Metode dan Langkah Pengujian

### 4.2.1. Data Dokumen

Data dokumen merupakan tahap untuk pengumpulan *dataset* dan proses pengujian data tes. Data yang digunakan berupa dokumen abstrak skripsi mahasiswa Budi Luhur dalam bentuk *pdf* yang kemudian di *convert* ke dalam bentuk teks. Data dokumen didapatkan dari perpustakaan Universitas Budi Luhur. Data dokumen yang sudah berbentuk teks akan dilakukan tahap *preprocessing* pembentukan *n-gram, rolling hash,* pembentukan *window* dan pencarian *fingerprint.* Sehingga diketahui tingkat *similarity* dari sebuah dokumen.

### 4.2.2. Tahapan *Preprocessing*

Setelah data dokumen didapat, maka dokumen akan memasuki tahap *preprocessing* pada dokumen yang sudah menjadi teks, proses yang dilakukan yaitu, *case folding,* menghapus karakter selain a-z, mengganti *slang word,* menghapus *stop word.* Berikut bisa dilihat di Tabel 4.1 berikut.

Tabel 4.1 *Tahapan Preprocessing*

|  |  |  |
| --- | --- | --- |
| **Keterangan** | **Data Awal** | **Hasil *Preprocessing*** |
| *Dataset* | IMPLEMENTASI AUGMENTED REALITY MENGGUNAKAN METODE MARKER BASED TRACKING BERBASIS ANDROID SEBAGAI MEDIA PROMOSI SEKOLAH PADA SMK YADIKA 4 KARANG TENGAH | implementasiaugmentedrealitymetodemarkerbasedtrakingberbasisandroidpromosismkyadikakarangtengah |
| Data Tes | IMPLEMENTASI TEKNOLOGI AUGMENTED REALITY SEBAGAI MEDIA PEMBELAJARAN DENGAN METODE MARKER BASED TRACKING BERBASIS MOBILE ANDROID PADA PAUD ANANDA KREO SELATAN | implementasiteknologiaugmentedrealitypembelajaranmetodemarkerbasedtrackingberbasismobileandroidpaudanandakreoselatan |

### 4.2.3. Pembentukan *N-Gram*

Setelah dokumen melalui tahap *preprocessing,* kemudian masuk ke tahap pembentukan *n-gram* yaitu, proses memecahkan *string* teks yang dikelompokan berdasarkan nilai *n-gram* yang akan dihitung pergeserannya secara terus menerus ke depan sejumlah nilai n sampai akhir dokumen. Contoh tahapan pembentukan *k-gram* dengan nilai N=3. Berikut bisa dilihat di Tabel 4.2 berikut.

Tabel 4.2 Pembentukan *N-Gram*

| **Keterangan** | **Hasil *Preprocessing*** | ***N-Gram*** |
| --- | --- | --- |
| *Dataset* | implementasiaugmentedrealitymetodemarkerbasedtrakingberbasisandroidpromosismkyadikakarangtengah | imp mpl ple lem eme men ent nta tas asi sia iau aug ugm gme men ent nte ted edr dre rea eai ali lit ity tym yme met eto tod ode dem ema mar ark rke ker erb rba bas ase sed edt dtr tra rak aki kin ing ngb gbe ber erb rba bas asi sia isa san and ndr dro roi oid idp dpr pro rom omo mos osi sis ism smk mky kya yad adi dik ika kak aka kar ara ran ang ngt gte ten eng nga gah |
| Data Tes | implementasiteknologiaugmentedrealitypembelajaranmetodemarkerbasedtrackingberbasismobileandroidpaudanandakreoselatan | imp mpl ple lem eme men ent nta tas asi sit ite tek ekn kno noi oio iog ogi gia iau aug ugm gme men ent nte ted edr dre rea eal ali lit ity typ ype pem emb mbe bel ela laj aja jar ara ran anm nme met eto tod ode dem ema mar ark rke ker erb rba bas ase sed edt dtr tra rac ack cki kin ing ngb gbe ber erb rba bas asi sis ism smo mob obi bil ile lea ean and ndr dro roi oid idp dpa pau aud uda dan ana nan and nda dak akr kre reo eos ose sel ela lat ata tan |

### 4.2.4. Algoritme *Winnowing*

Algoritme *winnowing* merupakan salah satu algoritme yang berfungsi sebagai dokumen *fingerprint* atau algoritme yang digunakan untuk mendeteksi tindakan plagiarisme dengan menggunakan teknik *hashing*. *Input* dari algoritme *winnowing* berupa dokumen teks, dan akan menghasilkan keluaran berupa kumpulan nilai *hash* yang terbentuk dari perhitungan ASCII pada setiap karakter. Berikut tahapannya.

### Perhitungan *Rolling Hash*

Setelah pembentukan *n-gram* maka proses selanjutnya yaitu, perhitungan *rolling hash* untuk mencari nilai *hash* dari setiap *string* yang sudah dipotong pada tahap *n-gram*.Setiap *string* yang sudah dipotong diubah menjadi ASCII lalu dihitung dengan rumus *hash.* Misal kita mengambil dari potongan kalimat *dataset* dan data tes yang ada. Rumus *hash:*

…(4.1)

c1 = nilai ascii dari huruf pertama dari satu buah *k-gram*

b = bilangan prima

k = nilai *k-gram*

Tabel 4.3 Kode ASCII

|  |  |  |  |
| --- | --- | --- | --- |
| **Karakter** | **Nilai Unicode** | **Nilai ANSI ASCII** | **Keterangan** |
| (heksadesimal) | (desimal) |
| a | 61 | 97 | Huruf latin b kecil |
| b | 62 | 98 | Huruf latin b kecil |
| c | 63 | 99 | Huruf latin c kecil |
| d | 64 | 100 | Huruf latin d kecil |
| e | 65 | 101 | Huruf latin e kecil |
| f | 66 | 102 | Huruf latin f kecil |
| g | 67 | 103 | Huruf latin g kecil |
| h | 68 | 104 | Huruf latin h kecil |
| i | 69 | 105 | Huruf latin i kecil |
| j | 006A | 106 | Huruf latin j kecil |
| k | 006B | 107 | Huruf latin k kecil |
| l | 006C | 108 | Huruf latin l kecil |
| m | 006D | 109 | Huruf latin m kecil |
| n | 006E | 110 | Huruf latin n kecil |
| o | 006F | 111 | Huruf latin o kecil |
| p | 70 | 112 | Huruf latin p kecil |
| q | 71 | 113 | Huruf latin q kecil |
| r | 72 | 114 | Huruf latin r kecil |
| s | 73 | 115 | Huruf latin s kecil |
| t | 74 | 116 | Huruf latin t kecil |
| u | 75 | 117 | Huruf latin u kecil |
| v | 76 | 118 | Huruf latin v kecil |
| w | 77 | 119 | Huruf latin w kecil |
| x | 78 | 120 | Huruf latin x kecil |
| y | 79 | 121 | Huruf latin y kecil |
| z | 007A | 122 | Huruf latin z kecil |

Berikut contoh perhitungan *rolling hash* dengan nilai k=3, bisa dilihat di Tabel 4.4 berikut.

Tabel 4.4 Perhitungan *Rolling Hash*

|  |  |  |
| --- | --- | --- |
| **Keterangan** | **Data Contoh** | ***N-Gram*** |
| *Dataset* | implementasipengamananfile | imp mpl ple lem eme men ent nta tas asi sip ipe pen eng nga gam ama man ana nan anf nfi fil ile |
| Data Tes | aplikasipengamananfile | apl pli lik ika kas asi sip ipe pen eng nga gam ama man ana nan anf nfi fil ile |

Pada Tabel 4.5 dan Tabel 4.6 akan menjelaskan contoh perhitungan dari potongan kata, untuk mencari nilai *hash* pada *dataset* dan data tes*.* Berikut penjabaran perhitungan nilai *hash* pada *dataset,* bisa dilihat di Tabel 4.6 berikut.

Tabel 4.5 Perhitungan *Hash Dataset*

| **Kalimat** | **Desimal ASCII** | **Penjabaran** | **Hasil** |
| --- | --- | --- | --- |
|
| imp | i=105 m=109 p=112 | 105\*7^(3-1)+109\*7^(3-2)+112\*7^(3-3) | 6020 |
| mpl | m=109 p=112 l=108 | 109\*7^(3-1)+112\*7^(3-2)+108\*7^(3-3) | 6233 |
| ple | p=112 l=108 e=101 | 112\*7^(3-1)+108\*7^(3-2)+101\*7^(3-3) | 6345 |
| lem | l=108 e=101 m=109 | 108\*7^(3-1)+101\*7^(3-2)+109\*7^(3-3) | 6108 |
| eme | e=101 m=109 e=101 | 101\*7^(3-1)+109\*7^(3-2)+101\*7^(3-3) | 5813 |
| men | m=109 e=101 n=110 | 109\*7^(3-1)+101\*7^(3-2)+110\*7^(3-3) | 6158 |
| ent | e=101 n=110 t=116 | 101\*7^(3-1)+110\*7^(3-2)+116\*7^(3-3) | 5835 |
| nta | n=110 t=116 a=97 | 110\*7^(3-1)+116\*7^(3-2)+97\*7^(3-3) | 6299 |
| tas | t=116 a=97 s=115 | 116\*7^(3-1)+97\*7^(3-2)+115\*7^(3-3) | 6478 |
| asi | a=97 s=115 i=105 | 97\*7^(3-1)+115\*7^(3-2)+105\*7^(3-3) | 5663 |
| sip | s=115 i=105 p=112 | 115\*7^(3-1)+105\*7^(3-2)+112\*7^(3-3) | 6482 |
| ipe | i=105 p=112 e=101 | 105\*7^(3-1)+112\*7^(3-2)+101\*7^(3-3) | 6030 |
| pen | p=112 e=101 n=110 | 112\*7^(3-1)+101\*7^(3-2)+110\*7^(3-3) | 6305 |
| eng | e=101 n=110 g=103 | 101\*7^(3-1)+110\*7^(3-2)+103\*7^(3-3) | 5822 |
| nga | n=110 g=103 a=97 | 110\*7^(3-1)+103\*7^(3-2)+97\*7^(3-3) | 6208 |
| gam | g=103 a=97 m=109 | 103\*7^(3-1)+97\*7^(3-2)+109\*7^(3-3) | 5835 |
| ama | a=97 m=109 a=97 | 97\*7^(3-1)+109\*7^(3-2)+97\*7^(3-3) | 5613 |
| man | m=109 a=97 n=110 | 109\*7^(3-1)+97\*7^(3-2)+110\*7^(3-3) | 6130 |
| ana | a=97 n=110 a=97 | 97\*7^(3-1)+110\*7^(3-2)+97\*7^(3-3) | 5620 |
| nan | n=110 a=97 n=110 | 110\*7^(3-1)+97\*7^(3-2)+110\*7^(3-3) | 6179 |
| anf | a=97 n=110 f=102 | 97\*7^(3-1)+110\*7^(3-2)+102\*7^(3-3) | 5625 |
| nfi | n=110 f=102 i=105 | 110\*7^(3-1)+102\*7^(3-2)+105\*7^(3-3) | 6209 |
| fil | f=102 i=105 l=108 | 102\*7^(3-1)+105\*7^(3-2)+108\*7^(3-3) | 5841 |
| ile | i=105 l=108 e=101 | 105\*7^(3-1)+108\*7^(3-2)+101\*7^(3-3) | 6002 |

Setelah dilakukan perhitungan diperoleh hasil dari *rolling hash* “6020 6233 6345 6108 5813 6158 5835 6299 6478 5663 6482 6030 6305 5822 6208 5835 5613 6130 5620 6179 5625 6209 5841 6002”.

Berikut penjabaran perhitungan nilai *hash* pada data tes, bisa dilihat di Tabel 4.6 berikut.

Tabel 4.6 Perhitungan *Hash* Data Tes

| **Kalimat** | **Desimal ASCII** | **Penjabaran** | **Hasil** |
| --- | --- | --- | --- |
|
| apl | a=97 p=112 l=108 | 97\*7^(3-1)+112\*7^(3-2)+108\*7^(3-3) | 5645 |
| pli | p=112 l=108 i=105 | 112\*7^(3-1)+108\*7^(3-2)+105\*7^(3-3) | 6349 |
| lik | l=108 i=105 k=107 | 108\*7^(3-1)+105\*7^(3-2)+107\*7^(3-3) | 6134 |
| ika | i=105 k=107 a=97 | 105\*7^(3-1)+107\*7^(3-2)+97\*7^(3-3) | 5991 |
| kas | k=107 a=97 s=115 | 107\*7^(3-1)+97\*7^(3-2)+115\*7^(3-3) | 6037 |
| asi | a=97 s=115 i=105 | 97\*7^(3-1)+115\*7^(3-2)+105\*7^(3-3) | 5663 |
| sip | s=115 i=105 p=112 | 115\*7^(3-1)+105\*7^(3-2)+112\*7^(3-3) | 6482 |
| ipe | i=105 p=112 e=101 | 105\*7^(3-1)+112\*7^(3-2)+101\*7^(3-3) | 6030 |
| pen | p=112 e=101 n=110 | 112\*7^(3-1)+101\*7^(3-2)+110\*7^(3-3) | 6305 |
| eng | e=101 n=110 g=103 | 101\*7^(3-1)+110\*7^(3-2)+103\*7^(3-3) | 5822 |
| nga | n=110 g=103 a=97 | 110\*7^(3-1)+103\*7^(3-2)+97\*7^(3-3) | 6208 |
| gam | g=103 a=97 m=109 | 103\*7^(3-1)+97\*7^(3-2)+109\*7^(3-3) | 5835 |
| ama | a=97 m=109 a=97 | 97\*7^(3-1)+109\*7^(3-2)+97\*7^(3-3) | 5613 |
| man | m=109 a=97 n=110 | 109\*7^(3-1)+97\*7^(3-2)+110\*7^(3-3) | 6130 |
| ana | a=97 n=110 a=97 | 97\*7^(3-1)+110\*7^(3-2)+97\*7^(3-3) | 5620 |
| nan | n=110 a=97 n=110 | 110\*7^(3-1)+97\*7^(3-2)+110\*7^(3-3) | 6179 |
| anf | a=97 n=110 f=102 | 97\*7^(3-1)+110\*7^(3-2)+102\*7^(3-3) | 5625 |
| nfi | n=110 f=102 i=105 | 110\*7^(3-1)+102\*7^(3-2)+105\*7^(3-3) | 6209 |
| fil | f=102 i=105  l= 108 | 102\*7^(3-1)+105\*7^(3-2)+108\*7^(3-3) | 5841 |
| ile | i=105 l=108 e=101 | 105\*7^(3-1)+108\*7^(3-2)+101\*7^(3-3) | 6002 |

Setelah dilakukan perhitungan diperoleh hasil dari *rolling hash* “5645 6349 6134 5991 6037 5663 6482 6030 6305 5822 6208 5835 5613 6130 5620 6179 5625 6209 5841 6002”.

### Pembentukan *Window*

Setelah didapatkan hasil perhitungan *rolling hash,* maka tahap selanjutnya adalah pembentukan *window* dari nilai *hash* yang didapat. Kemudian nilai *hash* dikelompokan sebanyak nilai *w-gram* atau w=4. Berikut contoh pembentukan *window.*

1. *Dataset*

Berikut pembentukan *window* pada *dataset* dengan w=4.

Nilai *hash :*

6020 | 6233 | 6345 | 6108 | 5813 | 6158 | 5835 | 6299 | 6478 | 5663 | 6482 | 6030 | 6305 | 5822 | 6208 | 5835 | 5613 | 6130 | 5620 | 6179 | 5625 | 6209 | 5841 | 6002

*Window* (w=4) :

**6020** | 6233 | 6345| 6108

**5813** | 6158 | 5835| 6299

6478 | **5663** | 6482| 6030

6305 | **5822** | 6208| 5835

**5613** | 6130 | 5620| 6179

**5625** | 6209 | 5841| 6002

1. Data Tes

Berikut pembentukan *window* pada data tesdengan w=4.

Nilai *hash :*

5645 | 6349 | 6134 | 5991 | 6037 | 5663 | 6482 | 6030 | 6305 | 5822 | 6208 | 5835 | 5613 | 6130 | 5620 | 6179 | 5625 | 6209 | 5841 | 6002

*Window* (w=4) :

**5645** | 6349 | 6134| 5991

6037 | **5663** | 6482| 6030

6305 | **5822** | 6208| 5835

**5613** | 6130 | 5620| 6179

**5625** | 6209 | 5841| 6002

### Pencarian *Fingerprint*

Setelah pembentukan *window* selesai, maka tahap selanjutnya adalah pencarian *fingerprint* dari setiap *window* yang ada. Nilai *fingerprint* ditentukan berdasarkan nilai terkecil dari setiap *window* yang ada. Berikut hasil dari *fingerprint* dari contoh sebelumnya.

1. *Dataset*

Berikut hasil pencarian *fingerprint* dari *dataset:*

6020 | 5813 | 5663 | 5822 | 5613 | 5625

1. Data Tes

Berikut hasil pencarian *fingerprint* dari data tes:

5645 | 5663 | 5822 | 5613 | 5625

### 4.2.5. *Jaccard Similarity*

Setelah *fingerprint* dari dokumen ditemukan langkah selanjutnya adalah perhitungan *similarity* dari *fingerprint* yang ada menggunakan *jaccard Similarity*. Berikut hasil setiap *fingerprint* ada pada Tabel 4.7 berikut.

Tabel 4.7 Hasil *Fingerprint*

|  |  |  |
| --- | --- | --- |
| **Keterangan** | **Contoh Kalimat** | ***Fingerprint*** |
| *Dataset* | implementasipengamananfile | 6020 | 5813 | 5663 | 5822 | 5613 | 5625 |
| Data Tes | aplikasipengamananfile | 5645 | 5663 | 5822 | 5613 | 5625 |

Berikut adalah rumus *jaccard similarity* :

*Similarity* (X,Y) =x 100%

X = {6020,5813,**5663**,**5822**,**5613**,**5625**}

Y = {5645,**5663**,**5822**,**5613**,**5625**}

X∩Y = {5663,5822,5613,5625}

X∪Y = {6020,5813,5663,5822,5613,5625,5645}

(X,Y) =x 100%

(X,Y) =x 100% = 0,571 x 100%= **57,1%**

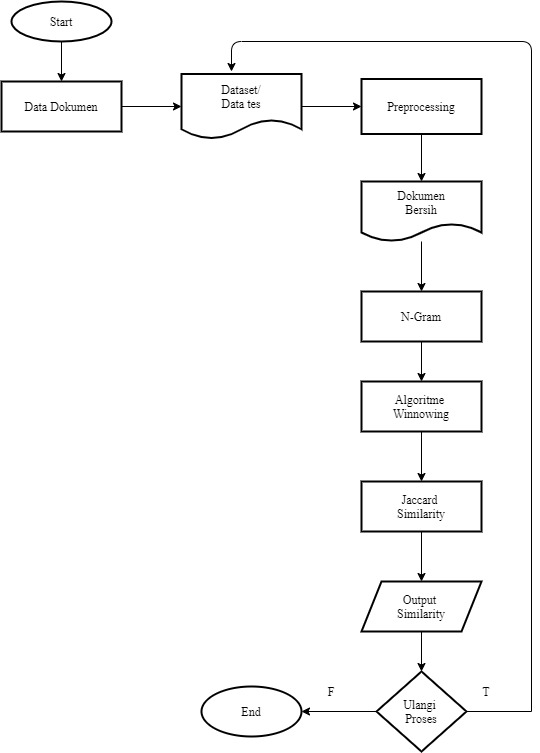
Sehingga tingkat *similarity* dari dokumenadalah 57,1%

## 4.3. *Flowchart* Tahapan Metode

*Flowchart* adalah suatu bagan atau simbol-simbol yang menggambarkan alur kerja atau urutan proses pada suatu program. Berikut adalah penjabaran *flowchart* pada tahapan metode yang digunakan:

### 4.3.1. *Flowchart* Keseluruhan Sistem

Pada *flowchart* ini menjelaskan tahapan mengenai berjalannya sistem, mulai dari pengumpulan data dokumen, hingga mendapatkan hasil tingkat *similarity* dari sebuah dokumen.

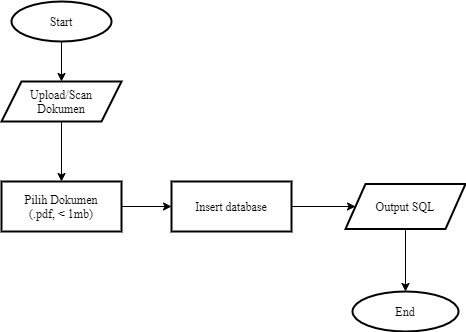


Gambar 4.1 *Flowchart* Keseluruhan Sistem

Pada gambar 4.1 menjelaskan proses keseluruhan sistem yang dibuat dengan tahap awal yaitu proses pengumpulan data dokumen, kemudian sebelum masuk ke tahapan n*-gram*, dokumen yang merupakan *dataset* atau data tes ini dilakukan tahapan pembersihan data atau *preprocessing*, setelah menjadi dokumen bersih, selanjutnya dilakukan proses tahapan utama *n-gram*, pembentukan *hash* melalui algoritme *winnowing,* perhitungan *similarity* dengan *jaccard similarity* hingga menghasilkan output, berupa tingkat *similarity*. Proses yang sama akan berulang jika dimasukkan dokumen untuk dijadikan *dataset* atau data tes.

### 4.3.2. *Flowchart* Data Dokumen

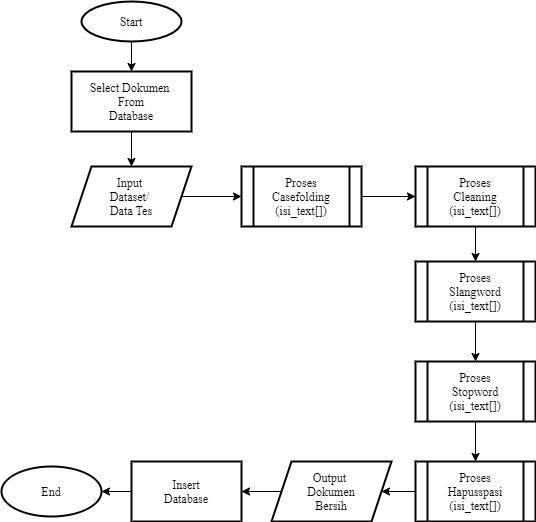
Pada *flowchart* ini, menjelaskan tahapan proses pengumpulan *dataset* serta proses *scan* untuk pengecekan tingkat *similarity* sebuah dokumen. Dokumen yang bisa diproses berupa file *.pdf* serta berukuran <1 mb, setelah itu dokumen yang sudah diproses akan disimpan ke *database.* Proses yang sama akan berulang jika ingin memproses dokumen kembali.



Gambar 4.2 *Flowchart* Data Dokumen

### 4.3.3. *Flowchart Preprocessing*

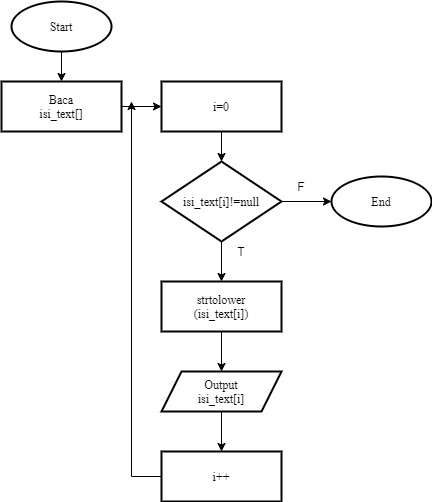
Pada *flowchart* ini, menjelaskan tahapan *preprocessing* data dokumen. Data dokumen hasil *upload/scan* yang sudah di*impor* ke dalam *database* dipilih untuk kemudian dilakukan sub proses *case folding, cleaning*, *slang word, stop word,* dan penghapusan spasi antar kata. Setelah itu dokumen bersih disimpan ke dalam *array* isi\_text[], dan dimasukkan kembali ke dalam *database*.



Gambar 4.3 *Flowchart Preprocessing*

### 4.3.4. *Flowchart Case folding*

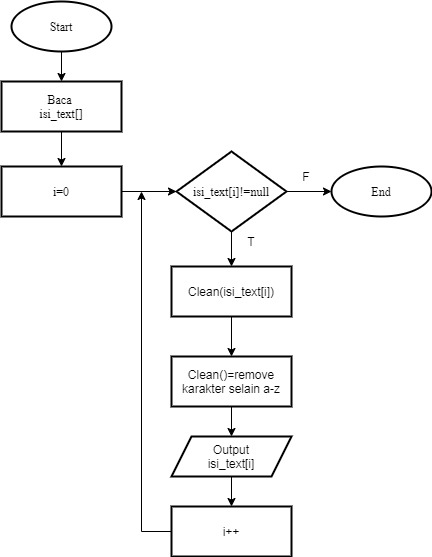
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *case folding*. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan proses *case folding* yaitu perubahan isi dokumen menjadi huruf kecil. Dengan fungsi strtolower(). Proses ini dijalankan sampai dokumen habis atau dokumen paling terakhir.



Gambar 4.4 *Flowchart Case folding*

### 4.3.5. *Flowchart Cleaning*

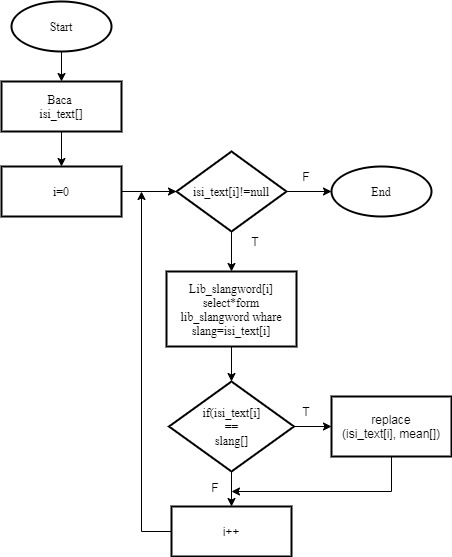
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *cleaning*. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan penghilangan pada karakter selain a-z dengan fungsi clean(). Proses ini dijalankan sampai dokumen habis atau dokumen paling terakhir.



Gambar 4.5 *Flowchart Cleaning*

### 4.3.6. *Flowchart Slang word*

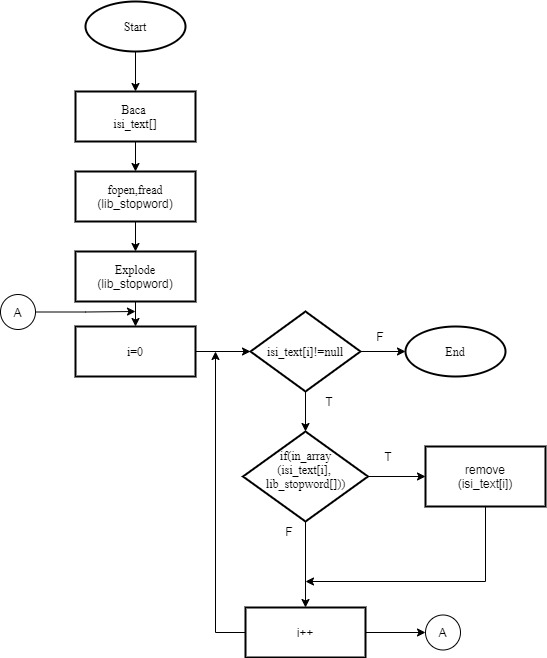
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *slang word.* Proses *slang word* diawali dengan membaca dokumen pada *array* isi\_text[] dan memanggil *library slang word* yang sama pada *array* isi\_text[i], jika ada yang sama maka dilakukan perintah replace(isi\_text[i], mean) yang artinya mengganti kata tersebut dengan arti pada kamus *library slang word*. Proses ini berulang sampai dokumen habis atau dokumen paling terakhir.



Gambar 4.6 *Flowchart Slang word*

### 4.3.7. *Flowchart Stop word*

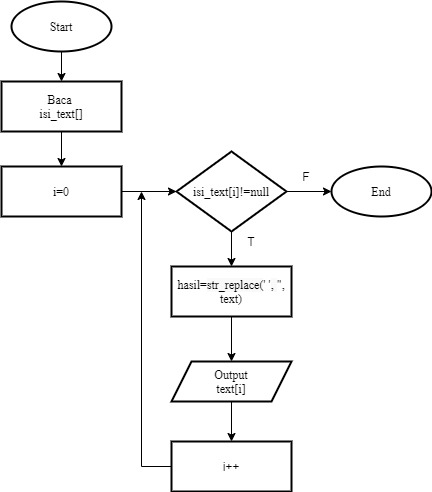
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *stop word,* proses ini diawali dengan membaca *library stop word* yang berisi kamus *stop word.* Kemudian dilakukan pencarian dengan fungsi in\_array (isi\_text[i], lib-stop word[]), artinya jika sebuah kata dalam array isi\_text[i] terdapat dalam kamus *stop word,* maka kata tersebut dihapus menggunakan fungsi remove(isi\_text[i]). Proses ini berulang sampai dokumen habis atau dokumen paling terakhir.



Gambar 4.7 *Flowchart Stop word*

### 4.3.8. *Flowchart* Hapus Spasi

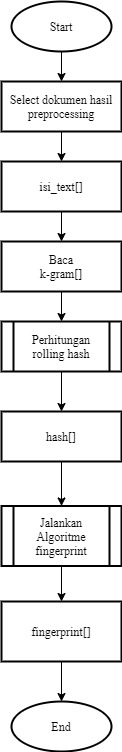
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu hapus spasi. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan penghapusan spasi antara satu kata dengan kata lainya. Menggunakan fungsi str\_replace(). Proses ini berulang sampai dokumen habis atau dokumen paling terakhir.



Gambar 4.8 *Flowchart* Hapus Spasi

### 4.3.9. *Flowchart* Algoritme *Winnowing*

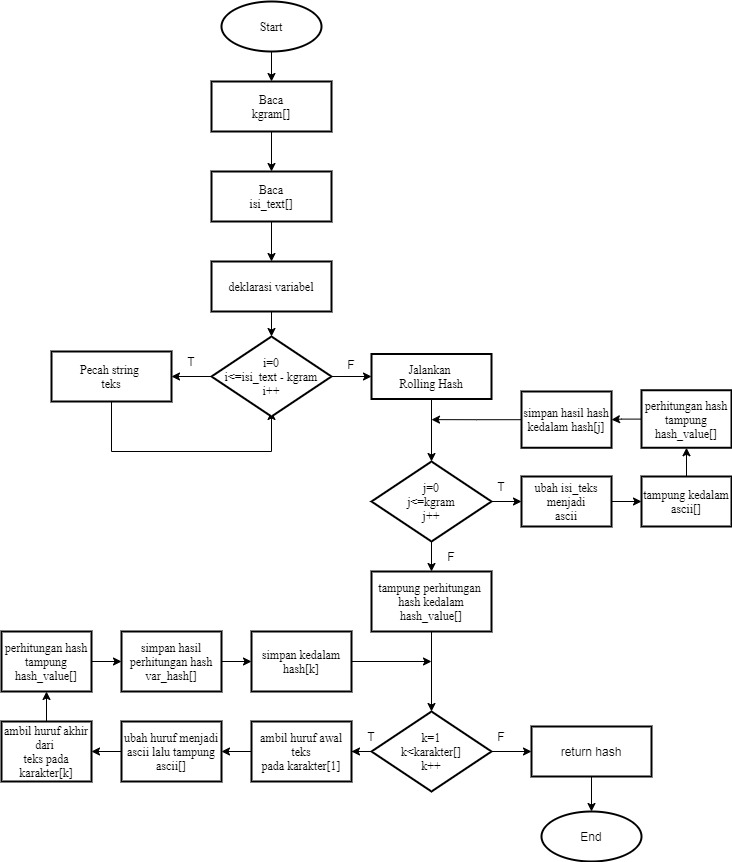
Pada *flowchart* ini, menjelaskan tahapan dari algoritme *winnowing* untuk mencari nilai *hash* dari sebuah dokumen. Setelah dokumen yang diproses pada tahap *preprocessing* dan telah disimpan ke dalam *database,* kemudian carilah nilai *hash* dari dokumen tersebut, melalui beberapa sub proses *rolling hash,* serta pencarian *fingerprint.* Setelah mendapatkan nilai *hash* dari dokumen yang diproses, maka setelah itu dapat dilakukan proses pencarian tingkat *similarity* dari sebuah dokumen.



Gambar 4.9 *Flowchart* Algoritme *Winnowing*

### 4.3.10. *Flowchart Rolling Hash*

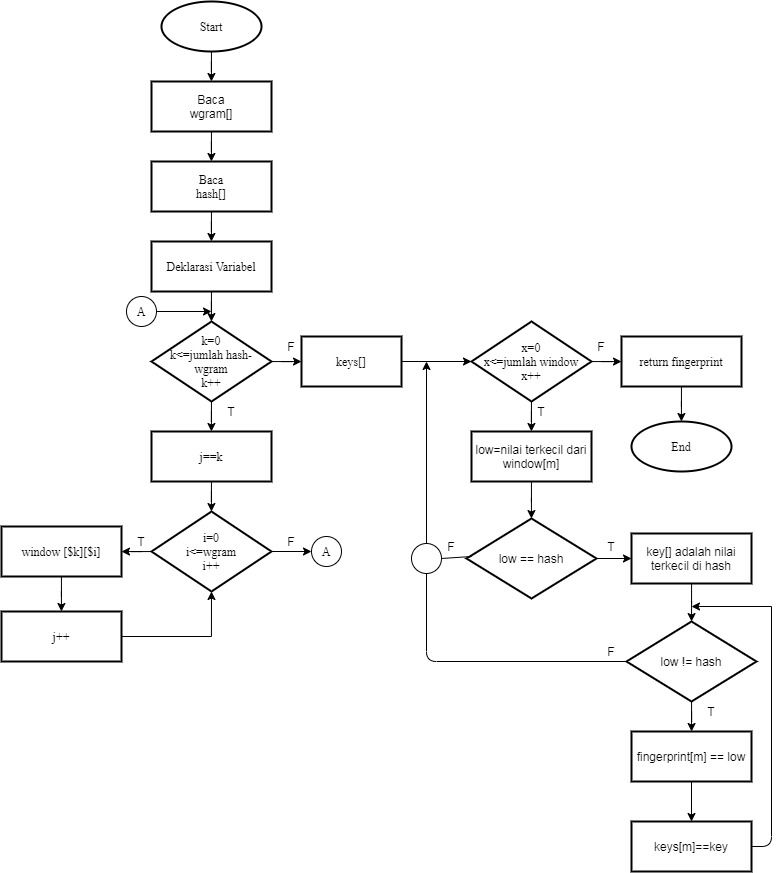
Pada *flowchart* ini, menjelaskan tahapan dari *rolling hash* yang merupakan sub proses dari algoritme *winnowing.* Proses pertama yang dilakukan adalah menentukan nilai *k-gram* untuk menentukan jumlah pecahan teks yang akan diproses, setelah teks dipecah, maka proses selanjutnya mencari nilai ASCII dari setiap huruf yang dipecah, kemudian ditampung di *variabel* hash\_value(), setelah mendapatkan nilai ASCII dari setiap huruf yang akan diproses, maka proses selanjutnya mencari nilai *hash* dari teks tersebut.



Gambar 4.10 *Flowchart Rolling Hash*

### 4.3.11. *Flowchart Fingerprint*

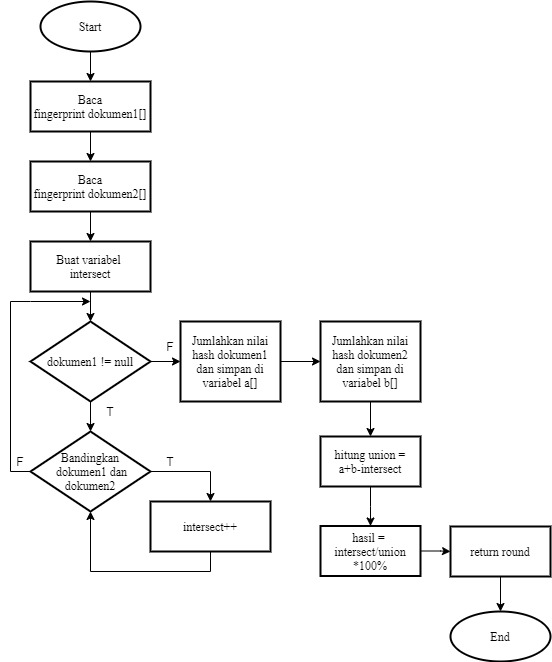
Pada *flowchart* ini, menjelaskan tahapan pencarian nilai *fingerprint* dari proses sebelumnya. Proses pertama yang dilakukan adalah pembentukan nilai *window,* pembentukan ini bertujuan untuk mengelompokan nilai *hash* yang telah didapatkan. Setelah dilakukan pembentukan *window,* maka tahap selanjutnya mencari nilai terkecil dari setiap *window* yang dibentuk, sehingga didapatkanlah nilai *fingerprint* dari setiap *window* yang telah dibentuk.



Gambar 4.11 *Flowchart Fingerprint*

### 4.3.12. *Flowchart Jaccard Similarity*

Pada *flowchart* ini, menjelaskan tahapan dari *jaccard similarity.* Tahapan ini merupakan tahapan untuk mencari persentase *similarity* dari sebuah dokumen. Setelah didapatkan nilai *fingerprint* pada tahap sebelumnya, maka pada proses selanjutnya adalah membandingkan nilai *fingerprint* dari dokumen *dataset* dengan dokumen data tes, sehingga dapat diperoleh persentase *similarity* dari dokumen yang telah diproses.



Gambar 4.12 *Flowchart Jaccard Similarity*

## 4.4. Algoritme Tahapan Metode

Algoritme adalah urutan atau alur tahapan proses yang dijabarkan dalam bentuk tulisan, algoritme ini merupakan representasi dari flowchart yang telah dijelaskan sebelumnya.

4.4.1. Algoritme Keseluruhan Sistem

Pada algoritme ini dijelaskan tentang proses sistem secara keseluruhan pada metode yang digunakan.

**Algoritme 4.1 Keseluruhan Sistem**

|  |
| --- |
| 1. *Start*  2. Proses dokumen  3. Baca dokumen *dataset* || data tes  4. Proses *preprocessing*  5. Hasil dokumen bersih  6. Proses *n-gram*  7. proses algoritme *winnowing*  8. proses *jaccard similarity*  9. *Output* tingkat *similarity*  10. if (ulangi proses)  11. kembali ke nomor 2  12. *End if*  13. *end* |

### 4.4.2. Algoritme Data Dokumen

Pada algoritme ini dijelaskan tentang proses tahapan pengolahan dokumen yang akan diproses.

**Algoritme 4.2 Data Dokumen**

|  |
| --- |
| 1. *Start*  2. *Input* : dokumen  3. Proses dokumen : .pdf,<1 mb  4. Proses *insert* dokumen ke *database*  5. *Output : SQL*  6. *End* |

### 4.4.3. Algoritme *Preprocessing*

Pada algoritme ini dijelaskan tentang proses tahapan pada *preprocessing* secara keseluruhan.

**Algoritme 4.3 *Preprocessing***

|  |
| --- |
| 1. *Start*  2. Proses mengambil dokumen dari *database*  3. *Input* : dokumen *dataset* || data tes  4. Lakukan proses *case folding()*  5. Lakukan proses *cleaning text()*  6. Lakukan proses *slang word()*  7. Lakukan proses *stop word()*  8. Lakukan proses penghapusan spasi  9.  *Output :* dokumen bersih  10. Proses simpan ke *database*  11. *End* |

### 4.4.4. Algoritme *Case folding*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada *preprocessing* yaitu *case folding*.

**Algoritme 4.4 *Case folding***

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. Merubah teks menjadi huruf kecil  6. *Output* isi\_text[i]  7. i++  8. Kembali ke nomor 3  9.  *End if*  10. *End* |

### 4.4.5. Algoritme *Cleaning*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada *preprocessing* yaitu *cleaning* dokumen.

**Algoritme 4.5 *Cleaning***

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. Proses Hapus isi\_text[i]  6. Proses Hapus karakter selain a-z  7. *Output* isi\_text[i]  8. i++  9. Kembali ke nomor 4  10. *End if*  11. *End* |

### 4.4.6. Algoritme *Slang word*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada *preprocessing* yaitu *slang word*.

**Algoritme 4.6 *Slang word***

|  |
| --- |
| 1. *Start*  2. Baca isi *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. *select lib\_slang word where slang= new\_text[i]*  6. *if*(isi\_text[i] == slang[]  7. *replace*(new\_text[i],mean)  8. i++  9. Kembali ke nomor 4  10.  *Endif*  11. Kembali ke nomor 8  12. *Endif*  13. *End* |

### 4.4.7. Algoritme *Stop word*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada *preprocessing* yaitu *slang word*.

**Algoritme 4.7 *Stop word***

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. Buka & baca *array library stop word*  4. *Explode library stop word*  5. i=0  6. *if*(isi\_text[i] ada)  7. *if*(isi\_text[i] terdapat dalam library stop word[])  8. *remove* (isi\_text[i])  9. i++  10. Kembali ke nomor 5  11. *Endif*  12. Kembali ke nomor 9  13. *Endif*  14. *End* |

### 4.4.8. Algoritme Hapus Spasi

Pada algoritme ini dijelaskan tentang proses sub tahapan pada *preprocessing* yaitu menghapus spasi pada sebuah teks dokumen.

**Algoritme 4.8 Hapus Spasi**

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. Proses Hapus spasi antar kata  6. *Output* isi\_text[i]  7. i++  8. Kembali ke no 4  9. *End if*  10. *End* |

### 4.4.9. Algoritme *Winnowing*

Pada algoritme ini dijelaskan tentang proses tahapan utama pada algoritme *winnowing* yaitu *rolling hash* dan  *fingerprint.*

**Algoritme 4.9 *Winnowing***

|  |
| --- |
| 1. *Start*  2. Ambil dokumen hasil *preprocessing*  3. Simpan dalam *array* isi\_text[]  4. Baca *array k-gram*  5. Lakukan proses perhitungan *rolling hash*  6. Simpan dalam *array hash*  7. Lakukan proses *fingerprint*  8. *End* |

### 4.4.10. Algoritme *Rolling Hash*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada algoritme *winnowing* yaitu *rolling hash*.

**Algoritme 4.10 *Rolling Hash***

|  |
| --- |
| 1. *Start*  2. Baca *array* kgram  3. Baca *array* isi\_text[]  4. *deklarasi* *variabel*  5. *if*(i=0, i<=isi\_text,kgram, i++)  6. Pecah *string* teks  7. *Endif*  8. Jalankan *rolling hash*  9. *if*(j=0,j<=kgram,j++)  10. Ubah isi\_text() menjadi ASCII  11. Tampung dalam *array* ascii[]  12. Proses perhitungan hash dan simpan ke dalam *array* hash\_value[]  13. Simpan ke dalam *array* hash[j]  14. *Endif*  15. Tampung perhitungan *hash* ke dalam *array* hash\_value[]  16. *if*(k=1,k<karakter(),k++)  17. ambil huruf awal teks pada karakter[i]  18. ubah huruf menjadi ascii dan tampung ke dalam array ascii[]  19. ambil huruf akhir dari teks pada karakter[k]  20. proses perhitungan *hash*, lalu tampung ke dalam *array* hash\_value[]  21. simpan hasil perhitungan *hash* ke dalam *array* var\_hash[]  22. simpan ke dalam hash[k]  23. *Endif*  24. *return hash*  25. *End* |

### 4.4.11. Algoritme *Fingerprint*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada algoritme *winnowing* yaitu *rolling hash*.

**Algoritme 4.11 *Fingerprint***

|  |
| --- |
| 1. *Start*  2. Baca *array* wgram[]  3. Baca *array* hash[]  4. *Deklarasi variabel*  5. if(k=0, k<=jumlah hash-wgram, k++)  6. j==k  7. if(i=0, i<=wgram, i++)  8. window[$k][$i]  9. j++  10.  *Endif*  11. kembali ke nomor 5  12. *Endif*  13. Buat *array* key[]  14. if(x=0, x<=jumlah window, x++)  15. low=nilai terkecil dari window[m]  16. if(low == hash)  17. isi *array* key[] adalah nilai terkecil di *hash*  18. if(low != hash)  19. array fingerprint[m] == low  20. keys[m]==key  21. Kembali ke nomor 18  22. *Endif*  23. Kembali ke nomor 13  24.  *Endif*  25. Kembali ke nomor 13  26.  *Endif*  27. *return fingerprint*  28. *End* |

### 4.4.12. Algoritme *Jaccard Similarity*

Pada algoritme ini dijelaskan tentang proses tahapan metode *jaccard similarity.*

**Algoritme 4.12 *Jaccard Similarity***

|  |
| --- |
| 1. *Start*  2. Baca *array* fingerprint dokumen1[]  3. Baca *array* fingerprint dokumen2[]  4. Buat *variabel intersect*  5. if(dokumen1 != null)  6. if(Bandingkan dokumen1 dan dokumen2)  7. intersect++  8. Endif  9. Kembali ke nomor 5  10. Endif  11. Jumlahkan nilai hash dokumen1 dan simpan di variabel a[]  12. Jumlahkan nilai hash dokumen2 dan simpan di variabel b[]  13. hitung union dokumen1 dan dokumen2 dikurang intersect  14. Proses hasil : intersect/union \*100%  15. *return round*  16. *End* |

## 4.5. Pengujian

Pengujian merupakan salah satu hal yang perlu dilakukan dalam setiap pengembangan sistem untuk mengevaluasi, menganalisa dan mengetahui tingkat akurasi atau kesamaan hasil yang telah dicapai oleh sistem yang telah dirancang. Pengujian dilakukan dengan beberapa cara yaitu, pertama pengujian dengan nilai *k-gram* dan *w-gram* yang sama, kedua pengujian dengan nilai *k-gram* dan *w-gram* yang berbeda, dan terakhir pengujian dengan satu *dataset* sebagai data tes.

### 4.5.1. Pengujian dengan nilai *k-gram* dan *w-gram* yang sama

Pengujian menggunakan 13 dokumen abstrak yang semua data dokumen nya berbeda sebagai *dataset*, dan data tes yang digunakan berbeda dengan *dataset*. Pada pengujian ini digunakan nilai *k-gram* dan *w-gram* dari 2 sampai 5 dimana nilainya sama pada Abstrak\_1513500346. Hasil pengujian dapat dilihat pada Tabel 4.8.

Tabel 4.8 Uji Coba Nilai *k-gram* dan w-*gram* Sama

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Nim** | **Similarity** | | | |
| **k=2 w=2** | **k=3 w=3** | **k=4 w=4** | **k=5 w=5** |
| 1 | 1511500025 | 46.96% | 21.78% | 5.84% | 2.08% |
| 2 | 1511500082 | 49.55% | 24.41% | 8.20% | 3.31% |
| 3 | 1511500132 | 44.95% | 20.62% | 8.66% | 3.32% |
| 4 | 1511500157 | 46.53% | 25.15% | 6.95% | 4.88% |
| 5 | 1511500199 | 50.51% | 21.39% | 7.11% | 3.19% |
| 6 | 1511500207 | 45.54% | 23.93% | 8.06% | 5.26% |
| 7 | 1511500215 | 44.66% | 19.23% | 7.65% | 3.64% |
| 8 | 1511500231 | 50.96% | 21.43% | 8.02% | 1.96% |
| 9 | 1511500249 | 51.40% | 20.51% | 8.33% | 3.74% |
| 10 | 1511500264 | 40.21% | 23.57% | 7.95% | 3.14% |
| 11 | 1511500272 | 47.66% | 20.53% | 7.48% | 3.59% |
| 12 | 1511500298 | 49.49% | 24.24% | 8.84% | 3.64% |
| 13 | 1511500314 | 43.33% | 22.47% | 6.55% | 2.36% |
| **Persentase Similarity Terbesar** | | **51.40%** | **25.15%** | **8.84%** | **5.26%** |
|

### 4.5.2. Pengujian dengan nilai *k-gram* dan *w-gram* yang berbeda

Pengujian menggunakan 13 dokumen abstrak yang semua data dokumen nya berbeda sebagai *dataset*, dan data tes yang digunakan berbeda dengan *dataset*. Pada pengujian ini digunakan nilai *k-gram* = 3 dan *w-gram* = 4, *k-gram* = 4 dan *w-gram* = 3, *k-gram* = 2 dan *w-gram* = 5, *k-gram* = 5 dan *w-gram* = 2, dimana nilainya berbeda pada Abstrak\_1513500346. Hasil pengujian dapat dilihat pada Tabel 4.9.

Tabel 4.9 Uji Coba Nilai *k-gram* dan *w-gram* Berbeda

| **No** | **Nim** | **Similarity** | | | |
| --- | --- | --- | --- | --- | --- |
| **k=3 w=4** | **k=4 w=3** | **k=2 w=5** | **k=5 w=2** |
| 1 | 1511500025 | 20.59% | 7.58% | 35.71% | 3.17% |
| 2 | 1511500082 | 22.15% | 8.59% | 36.54% | 3.87% |
| 3 | 1511500132 | 19.08% | 7.14% | 36.54% | 3.93% |
| 4 | 1511500157 | 21.37% | 8.13% | 44.44% | 4.42% |
| 5 | 1511500199 | 17.39% | 7.39% | 38% | 3.96% |
| 6 | 1511500207 | 23.62% | 9.17% | 40.43% | 5.34% |
| 7 | 1511500215 | 20.44% | 6.58% | 36.54% | 3.97% |
| 8 | 1511500231 | 21.64% | 8.33% | 35.29% | 3.34% |
| 9 | 1511500249 | 21.09% | 8.39% | 45.65% | 4.20% |
| 10 | 1511500264 | 26.50% | 6.96% | 41.86% | 4.46% |
| 11 | 1511500272 | 17.33% | 6.88% | 38.30% | 3.66% |
| 12 | 1511500298 | 23.26% | 8.44% | 44.44% | 4.68% |
| 13 | 1511500314 | 20.88% | 6.92% | 36.07% | 4% |
| **Persentase Similarity Terbesar** | | **26.50%** | **9.17%** | **45.65%** | **5.34%** |
|

### 4.5.3. Pengujian dengan satu *dataset* sebagai data tes.

Pengujian menggunakan 13 dokumen abstrak yang semua data dokumen nya berbeda sebagai *dataset*, dan 1 dokumen data tes yang digunakan sama dengan *dataset*. Pada pengujian ini digunakan nilai *k-gram* dan *w-gram* dari 2 sampai 5 dimana nilainya berbeda pada Abstrak\_1511500025. Hasil pengujian dapat dilihat pada Tabel 4.10.

Tabel 4.10 Uji Coba *Dataset* dijadikan Data Tes 1

| **No** | **Nim** | **Similarity** | | | |
| --- | --- | --- | --- | --- | --- |
| **k=2 w=2** | **k=3 w=3** | **k=4 w=4** | **k=5 w=5** |
| 1 | 1511500025 | 100% | 100% | 100% | 100% |
| 2 | 1511500082 | 69.17% | 26.26% | 10.30% | 4.6% |
| 3 | 1511500132 | 63.87% | 27.49% | 12.50% | 5.46% |
| 4 | 1511500157 | 62.28% | 26.07% | 8.70% | 3.54% |
| 5 | 1511500199 | 58.97% | 24.89% | 9.12% | 5.56% |
| 6 | 1511500207 | 64.29% | 25.76% | 7.89% | 3.83% |
| 7 | 1511500215 | 58.97% | 23.17% | 6.83% | 2.34% |
| 8 | 1511500231 | 63.03% | 24.80% | 7.52% | 2.75% |
| 9 | 1511500249 | 64.46% | 27.89% | 13.03% | 3.3% |
| 10 | 1511500264 | 53.10% | 22.27% | 10.27% | 4.08% |
| 11 | 1511500272 | 62.50% | 31.25% | 11.86% | 4.64% |
| 12 | 1511500298 | 62.28% | 26.52% | 9.59% | 3.15% |
| 13 | 1511500314 | 67.20% | 30% | 10.08% | 7.01% |
| **Persentase Similarity Terbesar** | | **100%** | **100%** | **100%** | **100%** |
|

Pada pengujian ini digunakan nilai *k-gram* dan *w-gram* dari 2 sampai 5 dimana nilainya berbeda pada abstrak 1511500157\_Deddy Rivaldy. Hasil pengujian dapat dilihat pada Tabel 4.11.

Tabel 4.11 Uji Coba Dataset dijadikan Data Tes 2

| **No** | **Nim** | **Similarity** | | | |
| --- | --- | --- | --- | --- | --- |
| **k=2 w=2** | **k=3 w=3** | **k=4 w=4** | **k=5 w=5** |
| 1 | 1511500025 | 62.28% | 26.07% | 8.70% | 3.54% |
| 2 | 1511500082 | 63.96% | 29.41% | 15.42% | 5.9% |
| 3 | 1511500132 | 58.18% | 29.44% | 13.88% | 7.76% |
| 4 | 1511500157 | 100% | 100% | 100% | 100% |
| 5 | 1511500199 | 60.19% | 23.53% | 11.16% | 3.92% |
| 6 | 1511500207 | 75.27% | 44.81% | 44.94% | 27.74% |
| 7 | 1511500215 | 55.66% | 23.32% | 7.66% | 5% |
| 8 | 1511500231 | 64.76% | 37.50% | 21.23% | 12.44% |
| 9 | 1511500249 | 61.82% | 33.33% | 18.03% | 11.63% |
| 10 | 1511500264 | 60% | 27.22% | 17.84% | 15.92% |
| 11 | 1511500272 | 67.31% | 28.21% | 12.17% | 7.32% |
| 12 | 1511500298 | 67.35% | 36.97% | 22.95% | 13.17% |
| 13 | 1511500314 | 62.07% | 32.89% | 16.30% | 8.17% |
| **Persentase Similarity Terbesar** | | **100%** | **100%** | **100%** | **100%** |
|

Pada pengujian ini digunakan nilai *k-gram* dan *w-gram* dari 2 sampai 5 dimana nilainya berbeda pada Abstrak\_1511500298. Hasil pengujian dapat dilihat pada Tabel 4.12.

Tabel 4.12 Uji Coba Dataset dijadikan Data Tes 3

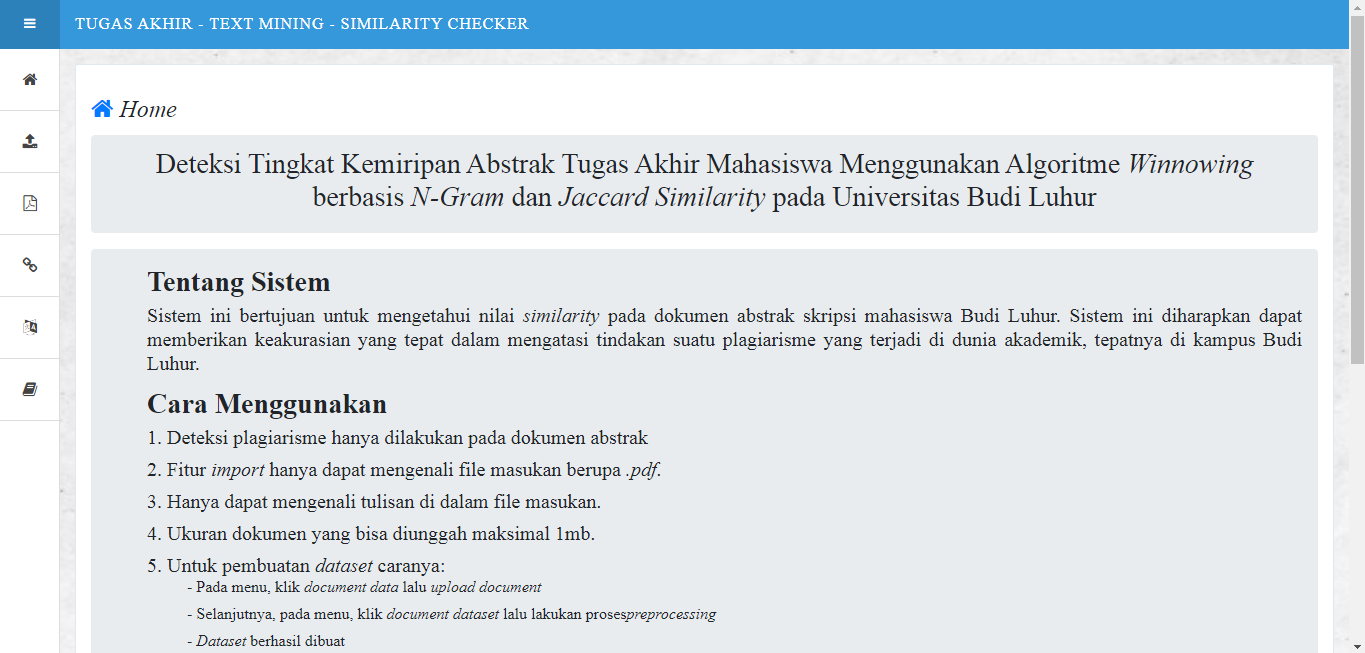
| **No** | **Nim** | **Similarity** | | | |
| --- | --- | --- | --- | --- | --- |
| **k=2 w=2** | **k=3 w=3** | **k=4 w=4** | **k=5 w=5** |
| 1 | 1511500025 | 62.28% | 26.52% | 9.59% | 3.15% |
| 2 | 1511500082 | 66.97% | 27.60% | 13.78% | 7.66% |
| 3 | 1511500132 | 56.76% | 29.38% | 12.65% | 5.38% |
| 4 | 1511500157 | 67.35% | 36.97% | 22.95% | 13.17% |
| 5 | 1511500199 | 55.66% | 24.73% | 8.26% | 6.03% |
| 6 | 1511500207 | 68.04% | 36.88% | 21.51% | 13.87% |
| 7 | 1511500215 | 58.65% | 21.88% | 7.25% | 2.73% |
| 8 | 1511500231 | 61.68% | 34.46% | 17.59% | 12.50% |
| 9 | 1511500249 | 61.82% | 34.04% | 19.82% | 12.21% |
| 10 | 1511500264 | 56.70% | 33.54% | 18.78% | 13.84% |
| 11 | 1511500272 | 52.63% | 28.13% | 10.39% | 6.31% |
| 12 | 1511500298 | 100% | 100% | 100% | 100% |
| 13 | 1511500314 | 62.07% | 31.70% | 15.22% | 9.06% |
| **Persentase Similarity Terbesar** | | **100%** | **100%** | **100%** | **100%** |
|

## 4.6. Tampilan Layar Aplikasi

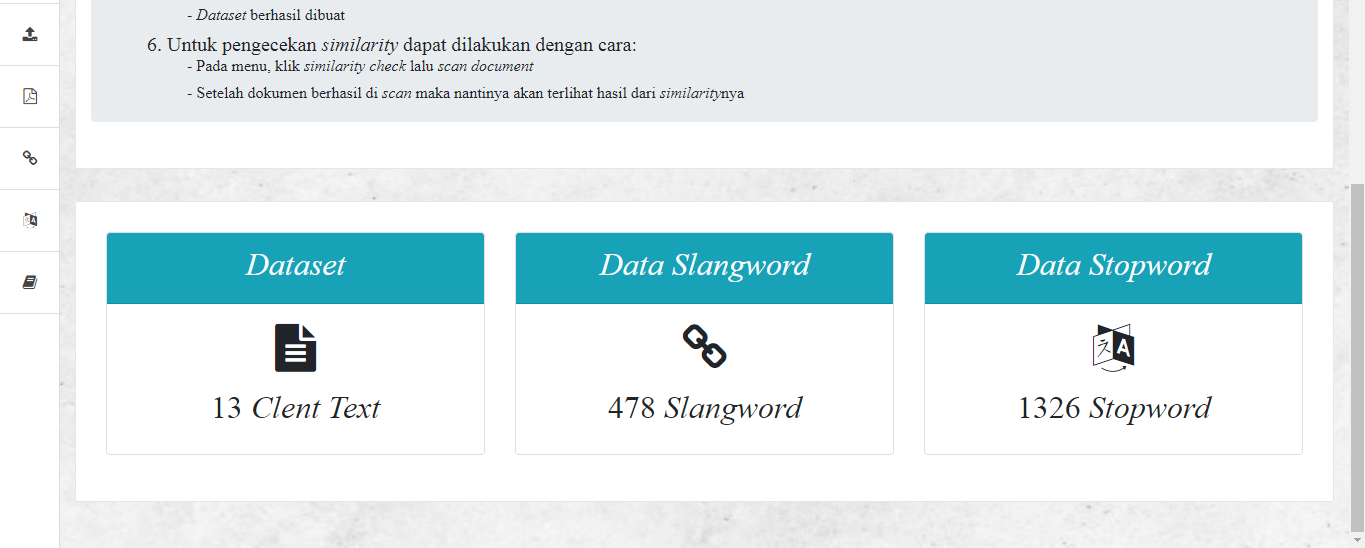
Setelah melewati beberapa tahap seperti implementasi metode, serta tahap pengujian sistem, pada tahap ini akan dijelaskan tampilan layar dari sistem yang telah dibuat.

### 4.6.1. Tampilan Layar *Home*

Tampilan layar *home* adalah tampilan saat pertama kali masuk ke dalam sistem. Di *menu home* ini menjelaskan tentang sistem yang dibuat, cara menggunakanya, serta jumlah *dataset, data slang word,* dan *data stop word.* Tampilan layar *home* bisa dilihat pada gambar 4.13 dan 4.14.



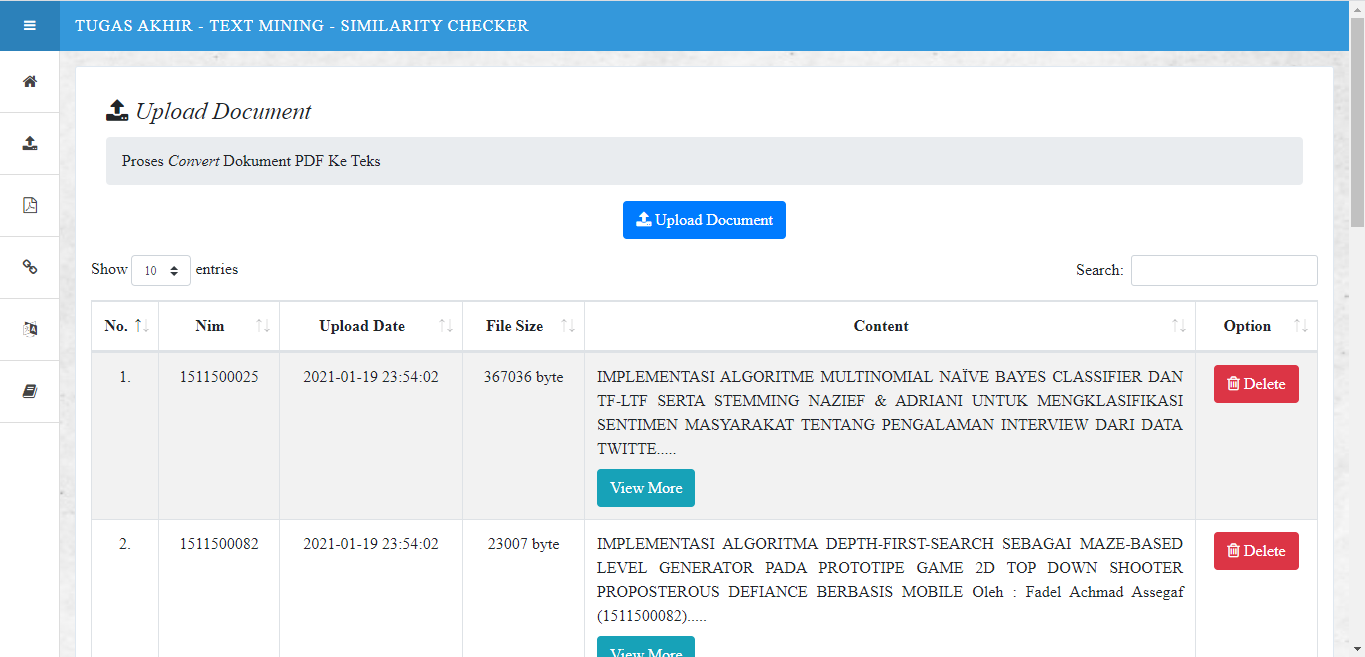
Gambar 4.13 Tampilan Layar *Home*



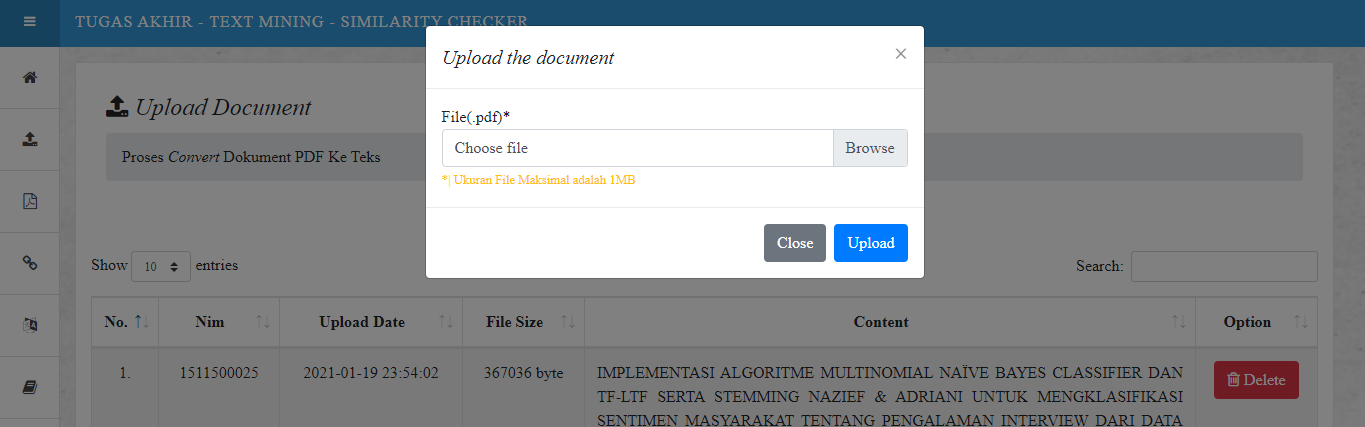
Gambar 4.14 Tampilan Layar *Home* (Lanjutan)

### 4.6.2. Tampilan Layar *Document Data*

Tampilan layar *document data* terdapat proses untuk *upload* dokumen. Di*menu* ini dokumen yang di*upload* berupa dokumen abstrak yang berbentuk *.pdf,* serta berukuran <10mb. Disini bisa dilihat juga dokumen yang sudah di*upload.* Tampilan layar *document data* bisa dilihat pada Gambar 4.15 dan Gambar 4.16.



Gambar 4.15 Tampilan Layar *Document Data*



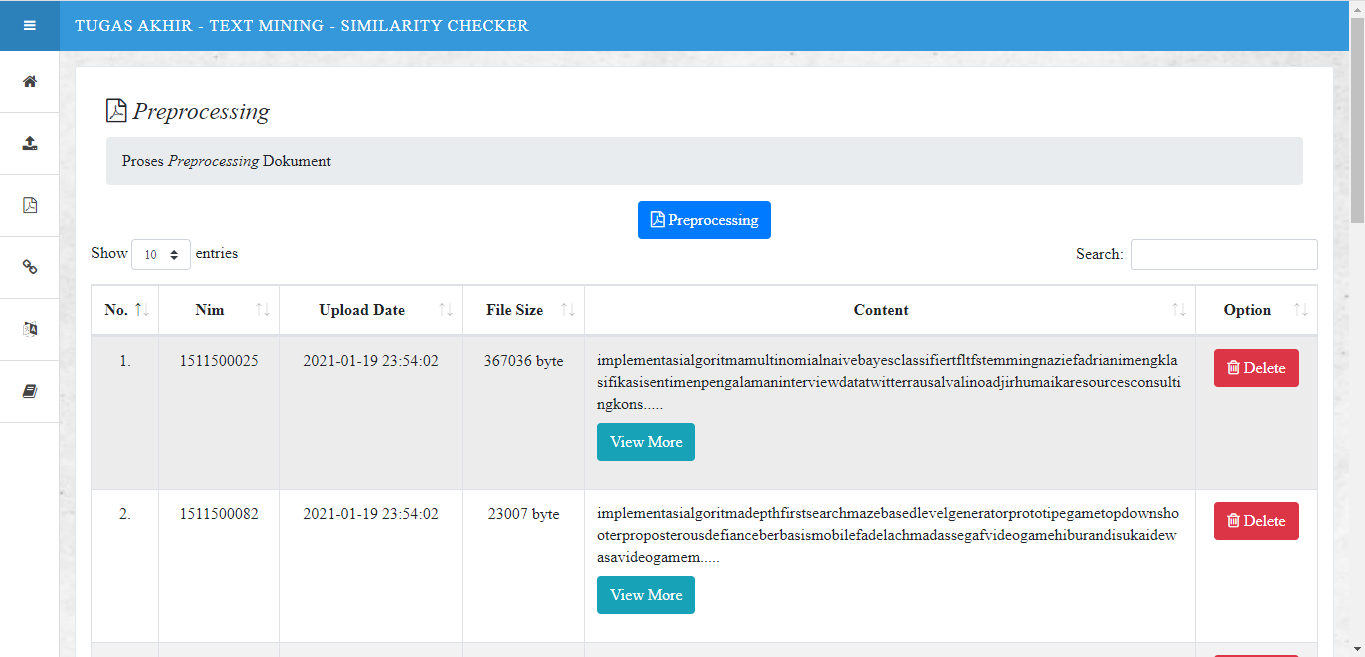
Gambar 4.16 Tampilan Layar *Document Data* (Lanjutan)

### 4.6.3. Tampilan Layar *Document Dataset*

Tampilan layar *document dataset* terdapat proses *preprocessing* untuk menghasilkan kumpulan dokumen *dataset.* Berikut gambar tampilan layarnya.

1. Tampilan Layar *Preprocessing*

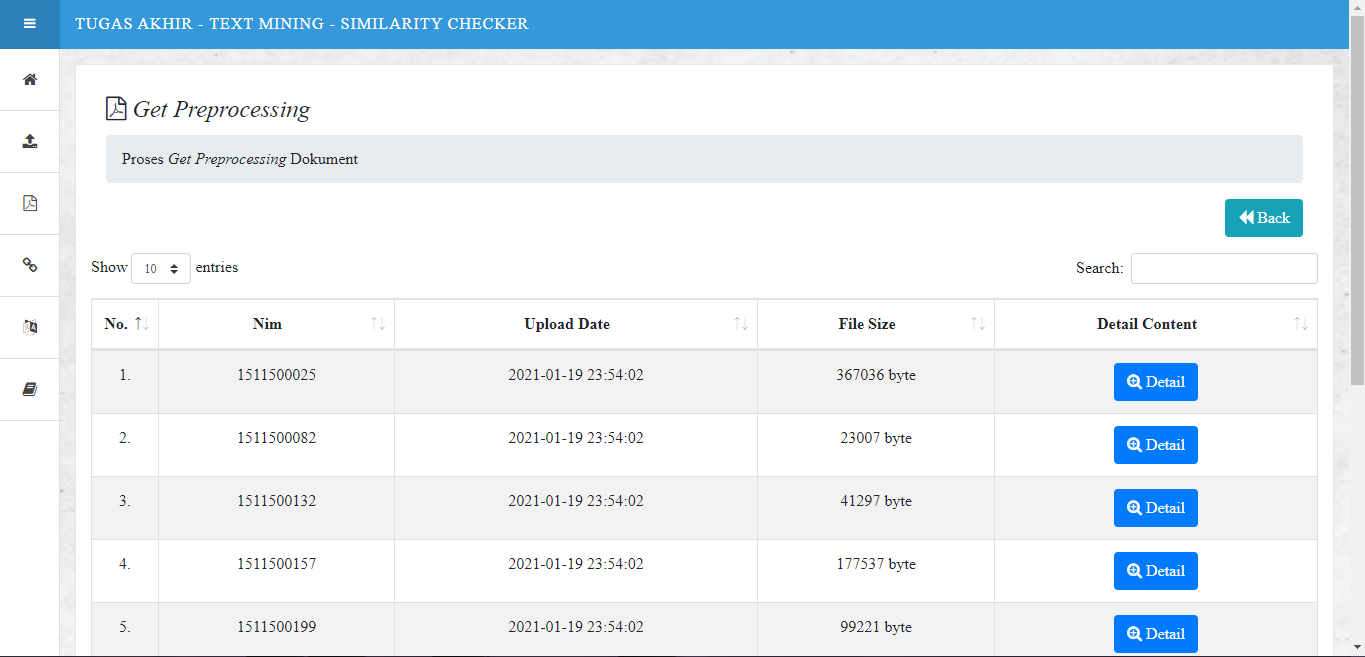
Pada Gambar 4.17 menjelaskan proses *preprocessing,* dari proses *case folding,* menghilangkan karakter selain a-z, proses *slang word, stop word,* dan penghapusan spasi antar karakter.



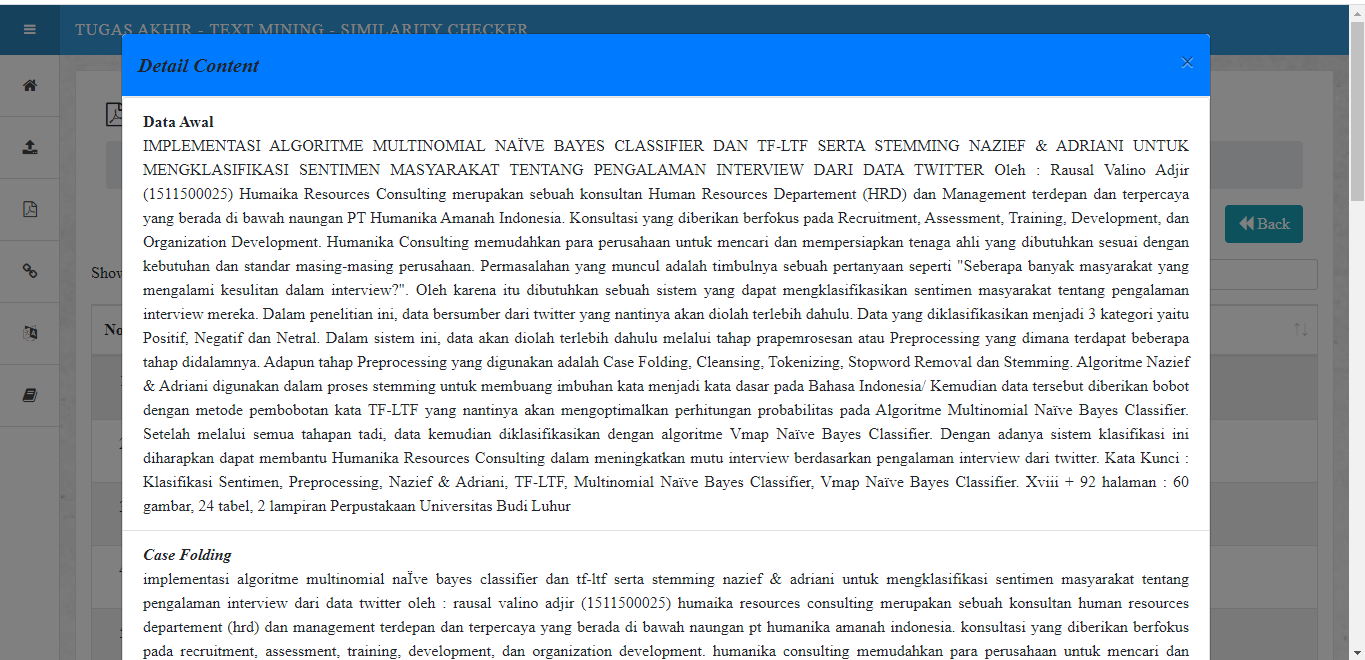
Gambar 4.17 Tampilan Layar *Preprocessing*

1. Tampilan Layar Detail *Preprocessing*

Pada Gambar 4.18 merupakan sub *menu* dari *document dataset.* Tampilan layar ini menjelaskan *detail* proses *preprocessing,* dan bisa dilihat lebih detail lagi pada Gambar4.19.



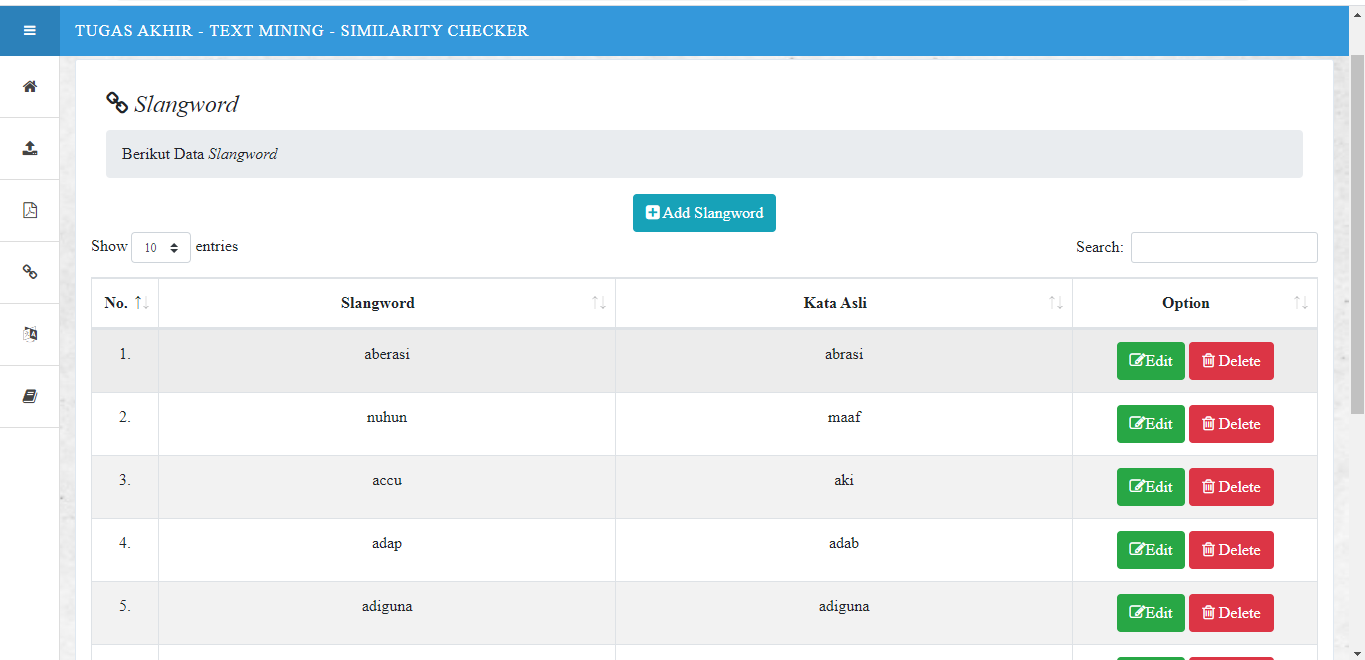
Gambar 4.18 Tampilan Layar Detail *Preprocessing*



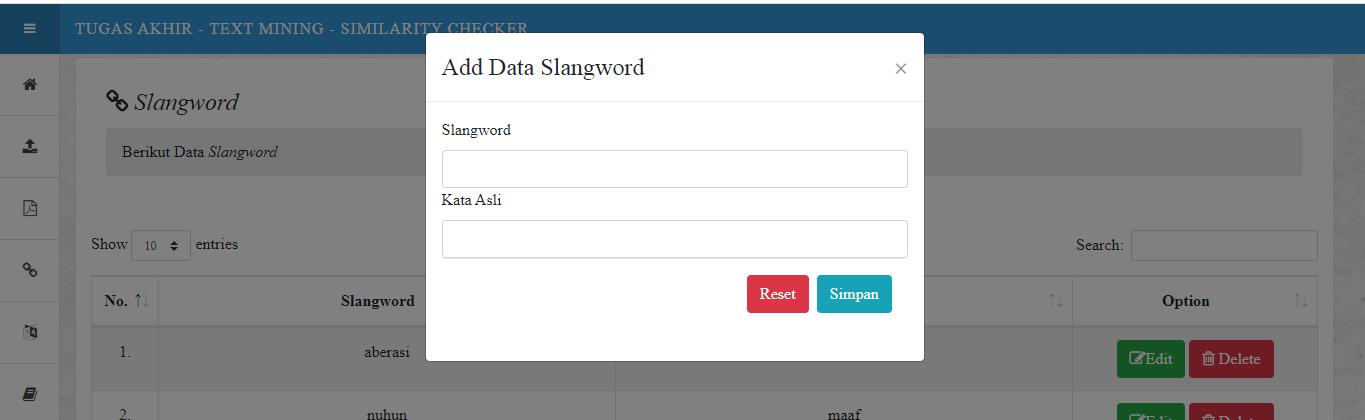
Gambar 4.19 Tampilan Layar Detail *Preprocessing* (Lanjutan)

### 4.6.4. Tampilan Layar *Slang word*

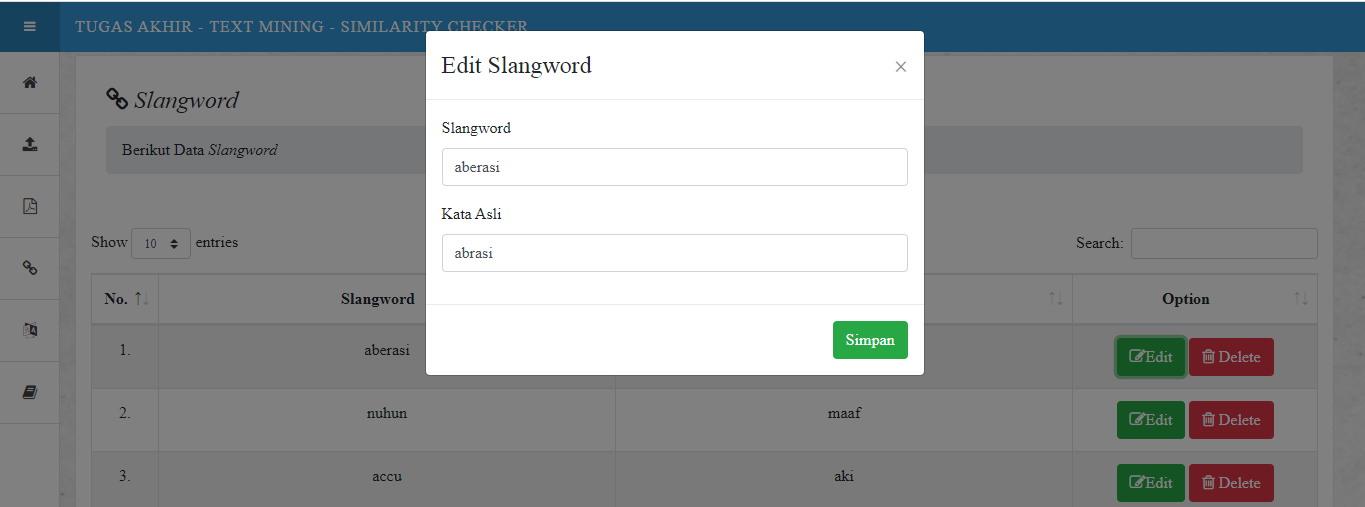
Tampilan layar pada Gambar 4.20, Gambar 4.21, dan Gambar 4.22 ini merupakan *menu slang word,* halaman ini berisi kumpulan data *slang word* yang digunakan untuk proses *preprocessing* nantinya, pada *menu* ini bisa dilakukan proses menambah, menghapus dan edit data *slang word.*



Gambar 4.20 Tampilan Layar *Menu Slang word*



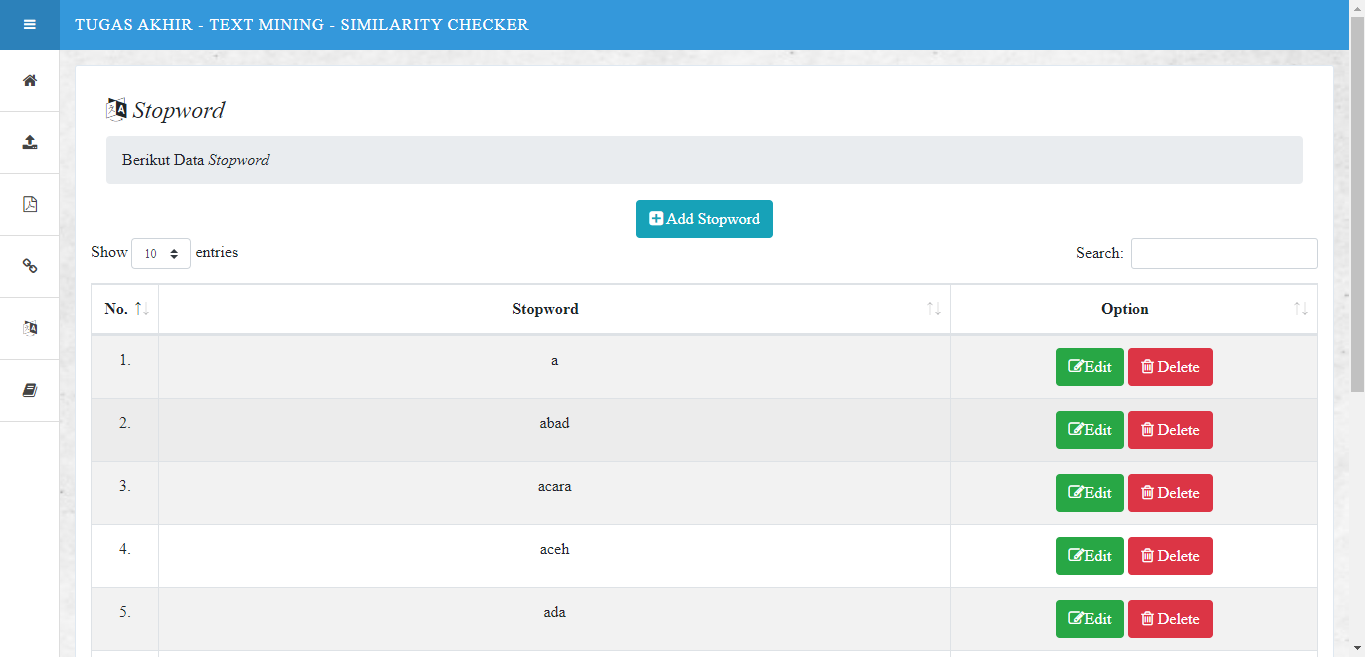
Gambar 4.21 Tampilan Layar *Menu Add Slang word*



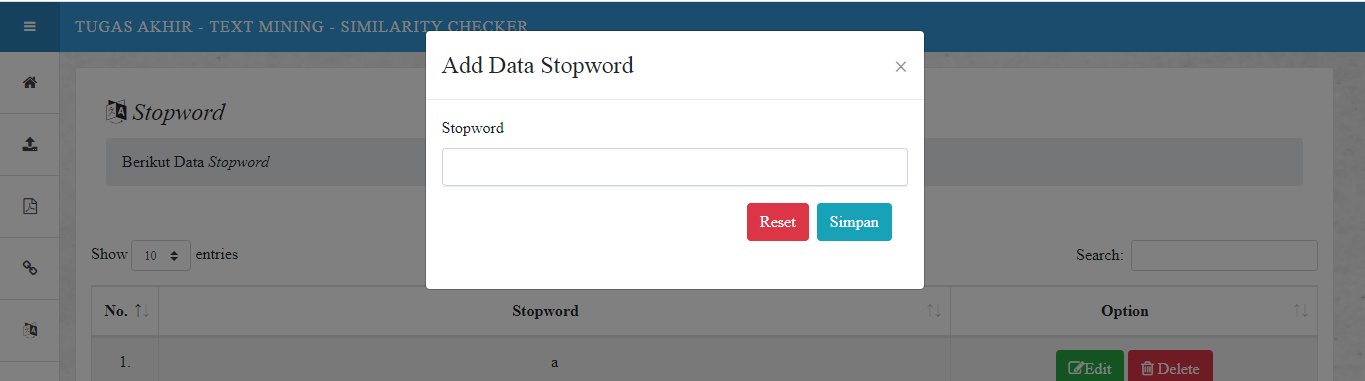
Gambar 4.22 Tampilan Layar *Menu Edit Slang word*

### 4.6.5. Tampilan Layar *Stop word*

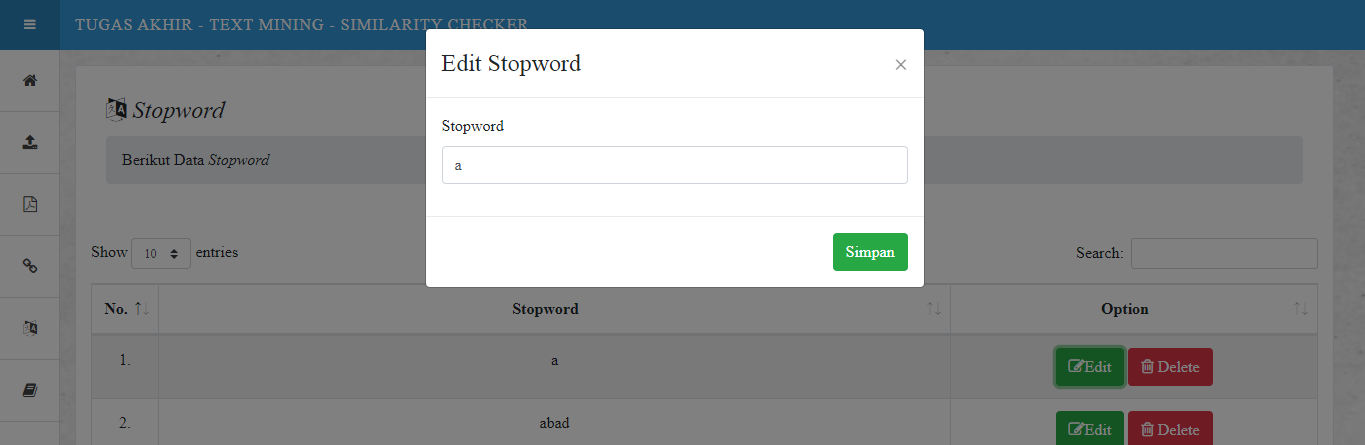
Tampilan layar pada Gambar 4.23, Gambar 4.24, dan Gambar 4.25 ini merupakan *menu stop word,* halaman ini berisi kumpulan data *stop word* yang digunakan untuk proses *preprocessing* nantinya, pada *menu* ini bisa dilakukan proses menambah, menghapus dan edit data *stop word.*



Gambar 4.23 Tampilan Layar Menu Stop word



Gambar 4.24 Tampilan Layar *Menu Add Stop word*



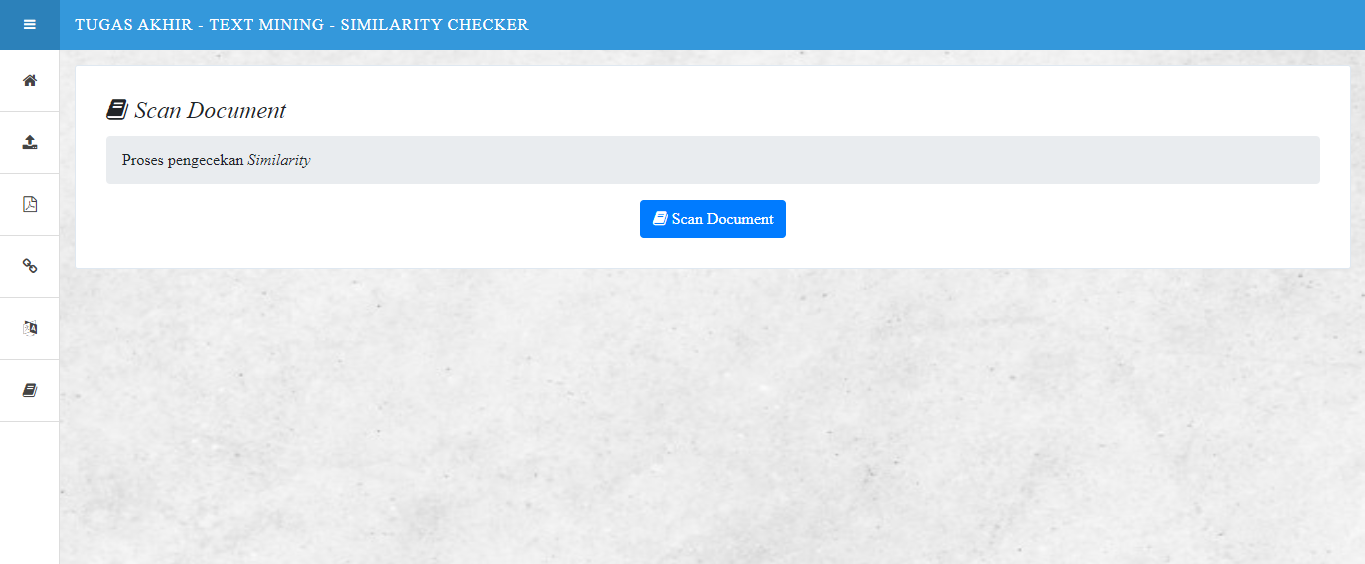
Gambar 4.25 Tampilan Layar *Menu* *Edit Stop word*

### 4.6.6. Tampilan Layar *Similarity Check*

Pada tampilan layar *similarity check* ini menjelaskan proses untuk mengetahui nilai *similarity* dari sebuah dokumen abstrak.

1. Tampilan Layar *Menu Scan Document*

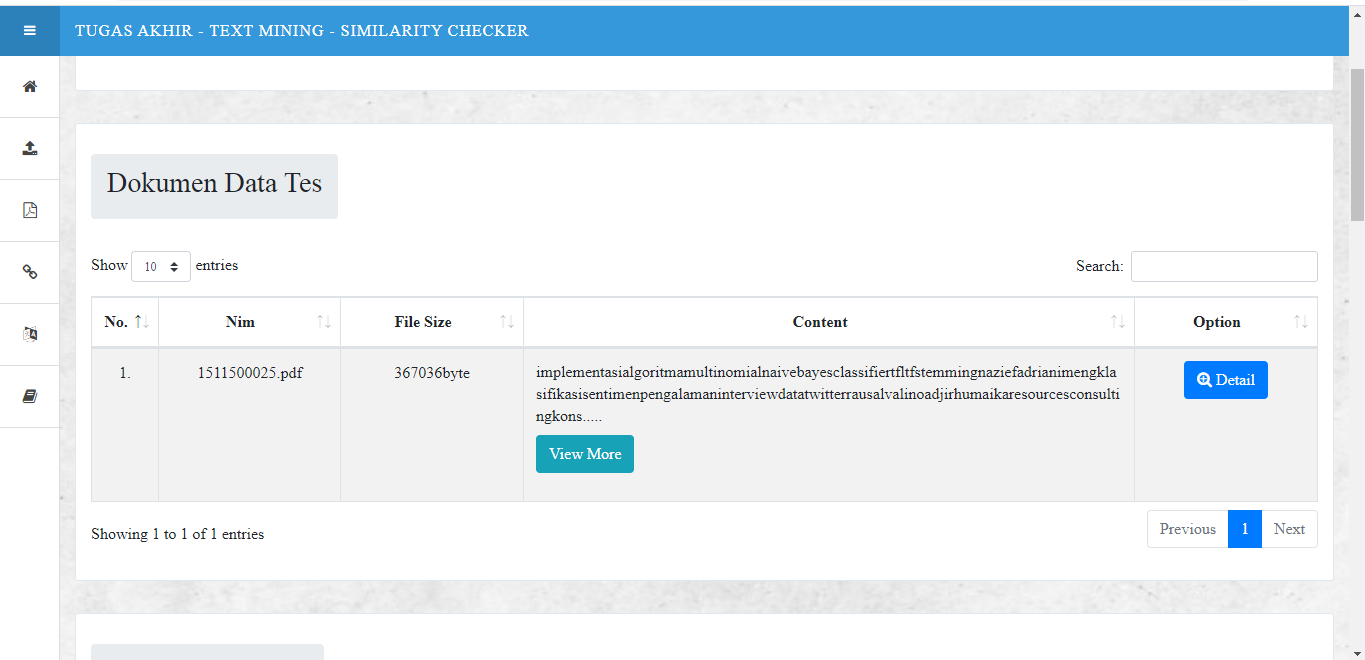
Tampilan layar pada Gambar 4.26 merupakan proses *scan* dokumen abstrak untuk melihat nilai *similarity* dari dokumen tersebut.



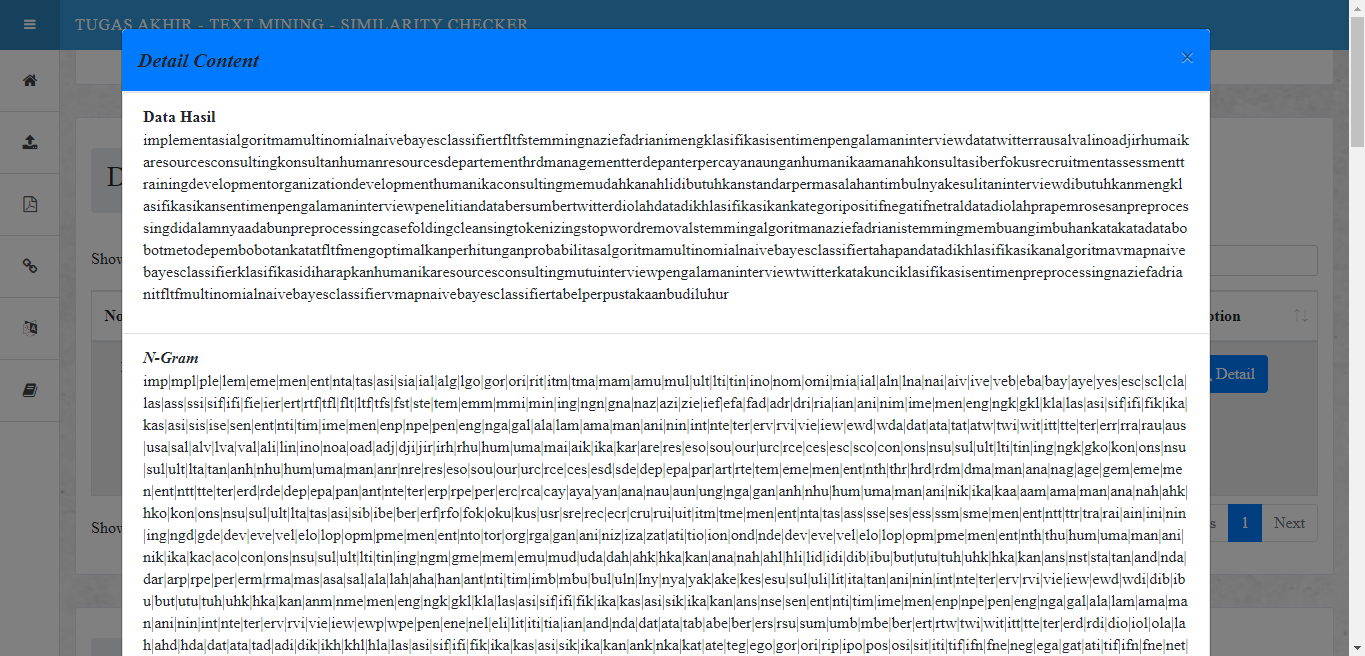
Gambar 4.26 Tampilan Layar *Menu Scan Document*

1. Tampilan Layar *Document* Data Tes

Tampilan layar pada Gambar 4.27 dan Gambar 4.28 merupakan dokumen abstrak data tes yang di*scan* pada proses sebelumnya. Pada gambar ini bisa dilihat teks bersih, potongan *n-gram,* nilai *rolling hash,* pembentukan *window,* serta nilai *fingerprint* dari dokumen tersebut.



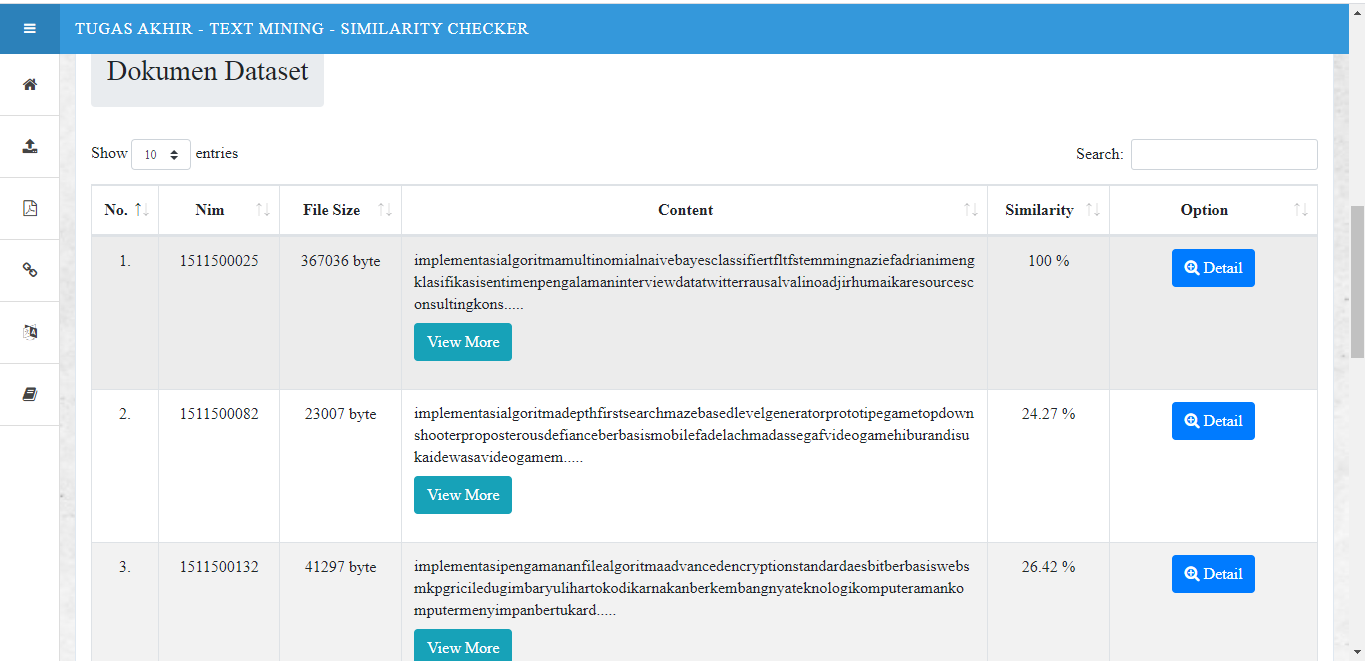
Gambar 4.27 Tampilan Layar *Document* Data Tes



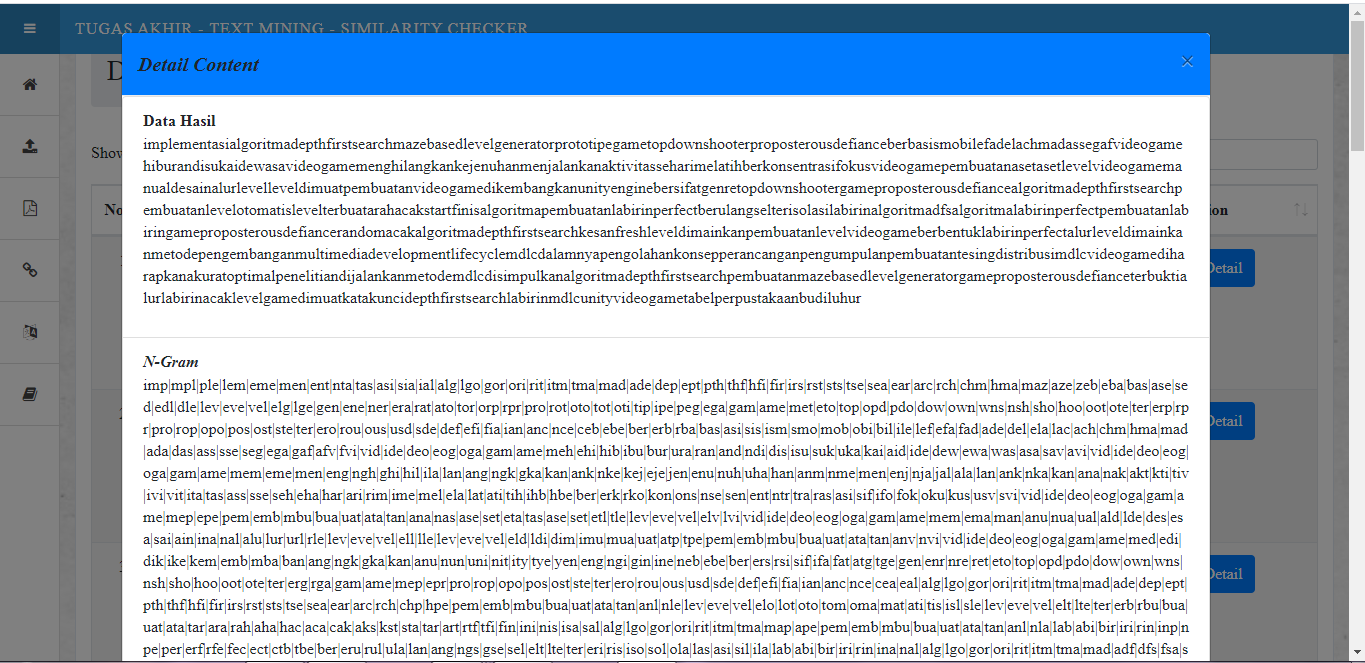
Gambar 4.28 Tampilan Layar *Document* Data Tes (Lanjutan)

1. Tampilan Layar Nilai *Similarity*

Tampilan layar pada Gambar 4.29 dan Gambar 4.30 merupakan dokumen abstrak *dataset* yang diambil dari *database*. Pada gambar ini bisa dilihat teks bersih, potongan *n-gram,* nilai *rolling hash,* pembentukan *window,* serta nilai *fingerprint* dari dokumen *dataset*. Pada tampilan layar ini bisa melihat tingkat *similarity* dari dokumen abstrak yang di*scan* pada proses sebelumnya dengan dokumen *dataset.*



Gambar 4.29 Tampilan Layar Nilai *Similarity*



Gambar 4.30 Tampilan Layar Nilai *Similarity* (Lanjutan)

# bab v

# penutup

## 5.1. Kesimpulan

Berdasarkan hasil evaluasi dari sistem deteksi tingkat *similarity*  pada dokumen abstrak skripsi mahasiswa Budi Luhur, maka dapat disimpulkan bahwa:

1. Sistem ini dapat mendeteksi tingkat *similarity* sebuah dokumen dengan sumber data dokumen abstrak mahasiswa Budi Luhur.
2. Penggunaan metode *n-gram* dan *jaccard similarity* terhadap algoritme *winnowing* cukup baik dengan menghasilkan persentase sampai 100%.
3. Nilai *k-gram* dan *w-gram* mempengaruhi persentase *similarity*, penggunaan *k-gram* dan *w-gram* yang tepat sangat diperlukan.
4. Nilai *k-gram* dan *w-gram* yang terlalu kecil mengakibatkan nilai persentase *similarity* yang besar, begitupun sebaliknya.
5. Metode *jaccard similarity* yang digunakan untuk menghasilkan tingkat *similarity* antar dokumen juga cukup baik karena koefisien ini sederhana dengan mencari item yang sama dari dua dokumen kemudian dibagi dengan total item kedua dari penggabungan dua dokumen.

## 5.2. Saran

Adapun saran yang dapat peneliti berikan sebagai pengembangan lebih lanjut untuk sistem ini agar dapat berjalan dengan sempurna dengan fungsi yang lebih baik adalah sebagai berikut:

1. Pengembangan sistem dapat dipadukan dengan metode lain, agar lebih baik hasil yang didapat.
2. Dapat mengetahui kata atau kalimat mana yang memiliki *similarity.*
3. Dapat melakukan pendeteksian *similarity* tidak hanya di bagian abstrak saja.

# daftar pustaka

Abdullah, I., & Aribowo, E. (2018). Rancang Bangun Aplikasi Pengecekan Kemiripan Judul Skripsi Dengan Metode Cosine Similarity (Studi Kasus : Program Studi Teknik Informatika Uad). *JSTIE (Jurnal Sarjana Teknik Informatika) (E-Journal)*, *6*(2), 43–52. https://doi.org/10.12928/jstie.v6i2.15241

Alamanda, R., Suhery, C., & Brianorman, Y. (2016). Aplikasi pendeteksi plagiat terhadap karya tulis berbasis web menggunakan natural language processing dan algoritme knuth-morris-pratt. *Jurnal Coding, Sistem Komputer Untan*, *04*(1).

Anggara, Y. (2016). Deteksi plagiarisme dokumen bahasa indonesia dengan algoritme jaro-winkler Distance. *Program Studi Teknik Informatika Universitas Muhammadiyah Jember*.

Ariantini, D. A. R., Lumenta, A. S. M., & Jacobus, A. (2016). Pengukuran Kemiripan Dokumen Teks Bahasa Indonesia Menggunakan Metode Cosine Similarity. *Jurnal Teknik Informatika*, *9*(1), 1–8. https://doi.org/10.35793/jti.9.1.2016.13752

Fataruba, F. (2018). Penerapan Metode Cosine Similarity Untuk Pengecekan Kemiripan Jawaban Ujian Siswa. *Jurnal Mahasiswa Teknik Informatika*, *2*(2), 88–95.

Februariyanti, H. (2012). Klasifikasi Dokumen Berita Teks Bahasa Indonesia menggunakan Ontologi. *Teknologi Informasi DINAMIK*, *17*(1), 14–23. Retrieved from http://www.unisbank.ac.id/ojs/index.php/fti1/article/view/1612/594

Filcha, A., & Hayaty, M. (2019). Implementasi Algoritme Rabin-Karp untuk Pendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa. *JUITA : Jurnal Informatika*, *7*(1), 25. https://doi.org/10.30595/juita.v7i1.4063

Fitri, S., Nurjanah, N., & Astuti, W. (2018). Penerapan Data Mining Untuk Evaluasi Kinerja Akademik Mahasiswa (Studi Kasus: Umtas). *Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer*, *9*(1), 633–640. https://doi.org/10.24176/simet.v9i1.2002

Gunadi, G., & Sensuse, D. I. (2012). Penerapan Metode Data Mining Market Basket Analysis Terhadap Data Penjualan Produk Buku Dengan Menggunakan Algoritme Apriori Dan Frequent Pattern Growth ( Fp-Growth ) : *Telematika*, *4*(1), 118–132.

Khidfi, M. N., & Sari, J. Y. (2018). Rancang bangun aplikasi pendeteksian kesamaan pada dokumen teks menggunakan algoritme Enhaced confix stripping dan algoritme winnowing. *Jurusan Teknik Informatika, Fakultas Teknik Universitas Halu Oleo, Kendari*, *4*(2), 1–10. https://doi.org/10.5281/zenodo.1407866

Nurdin, N., Rizal, R., & Rizwan, R. (2019). Pendeteksian Dokumen Plagiarisme dengan Menggunakan Metode Weight Tree. *Telematika*, *12*(1), 31. https://doi.org/10.35671/telematika.v12i1.775

Priambodo, J. (2018). Pendeteksian Plagiarisme Menggunakan Algoritme Rabin-Karp dengan Metode Rolling Hash. *Jurnal Informatika Universitas Pamulang*, *3*(1), 39. https://doi.org/10.32493/informatika.v3i1.1518

Rahmatulloh, A., Kurniati, N. I., Darmawan, I., Asyikin, A. Z., & Witarsyah, J. D. (2019). Comparison between the stemmer porter effect and nazief-adriani on the performance of winnowing algorithms for measuring plagiarism. *International Journal on Advanced Science, Engineering and Information Technology*, *9*(4), 1124–1128. https://doi.org/10.18517/ijaseit.9.4.8844

Rinartha, K. (2017). Pemodelan Penilaian Essay Otomatis Secara Realtime Menggunakan Kombinasi Text Stemming Dan Cosine Similarity. *Konferensi Nasional Sistem & Informatika*, 322–327.

Shu, K., Wang, S., Lee, D., & Liu, H. (2020). Mining disinformation and fake news: Concepts, methods, and recent advancements. *International Journal of Computer Applications*, *68*(13), 13–18. https://doi.org/10.1007/978-3-030-42699-6\_1

Sunardi, S., Yudhana, A., & Mukaromah, I. A. (2018). Implementasi Deteksi Plagiarisme Menggunakan Metode N-Gram Dan Jaccard Similarity Terhadap Algoritme Winnowing. *Transmisi*, *20*(3), 105. https://doi.org/10.14710/transmisi.20.3.105-110

Sunyoto, A., & Informatika, T. (2013). Implementasi Algoritme Rabin Karp untuk Pendeteksian Plagiat Dokumen Teks Menggunakan Konsep Similarity, 23–28.

Tala, F. Z. (2003). A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. *M.Sc. Thesis, Appendix D*, *pp*, 39–46.

Ulfa, N. F., Mustikasari, M., & Bastian, I. (2016). Pendeteksian tingkat similaritas dokumen berbasis web menggunakan algoritme winnowing. *Konferensi Nasional Teknologi Informasi Dan Komunikasi (KNASTIK)*, (November), 194–203.