# 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* kedalam 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, *casefolding,* menghapus karakter selain a-z, mangganti *slangword,* menghapus *stopword.* Berikut bisa dilihat di Tabel 4.1 berikut.

Tabel 1 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 menurus ke depan sejumlah nilai n sampai akhir dokumen. Contoh tahapan pembentukan kgram dengan nilai N=3. Berikut bisa dilihat di Tabel 4.2 berikut.

Tabel 2 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 peritungan ASCII pada setiap karakter. Berikut tahapanya.

### 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 kgram

b = bilangan prima

k = nilai *k-gram*

Tabel 3 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 4.4 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 5 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 6 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 7 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%**

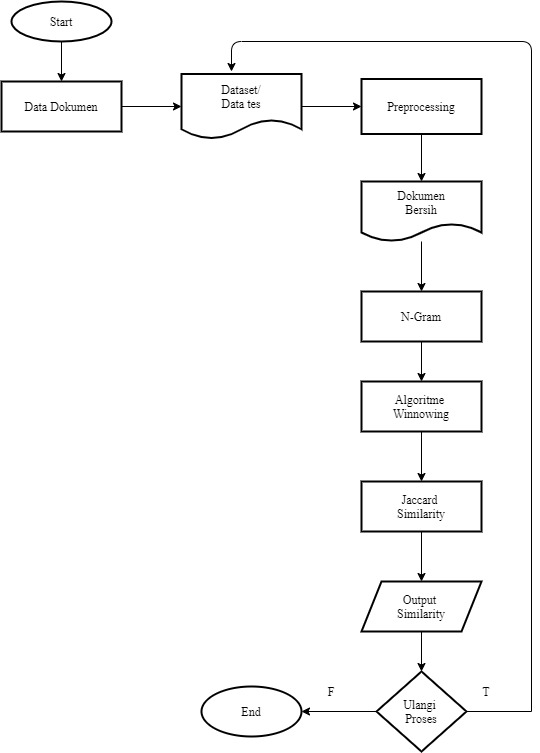
Sehingga tingkat *similarity* dari dokumenadalah 57,1%

## 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.

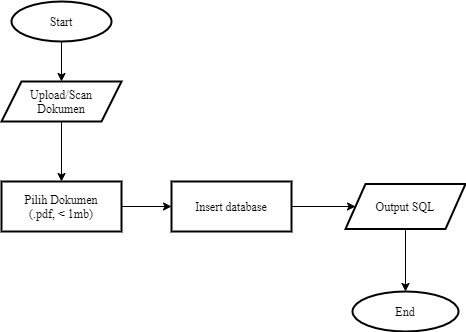


Gambar 1 4.1 *Flowchart* Keseluruhan Sistem

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

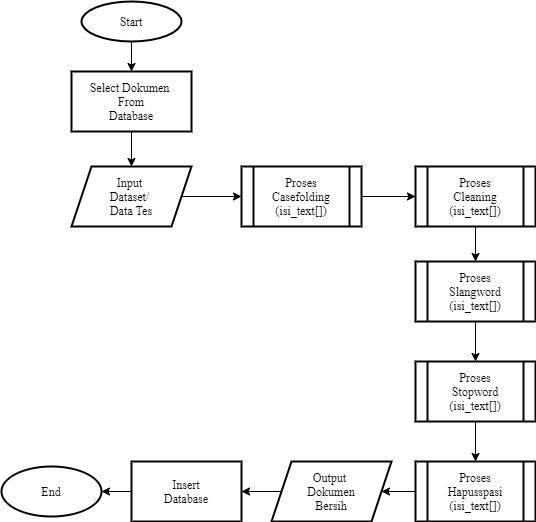
Pada *flowchart* ini, menjelaskan tahapan proses pengumpulan *dataset* serta proses *scan* untuk pengecekan tingkat *similarity* sebuah dokumen. Dokumen yang bisa diproses berupa file *.pdf* serta berukuran <1mb, setelah itu dokumen yang sudah diproses akan disimpan ke *database.* Proses yang sama akan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



Gambar 2 4.2 *Flowchart* Data Dokumen

### 4.3.3. *Flowchart Preprocessing*

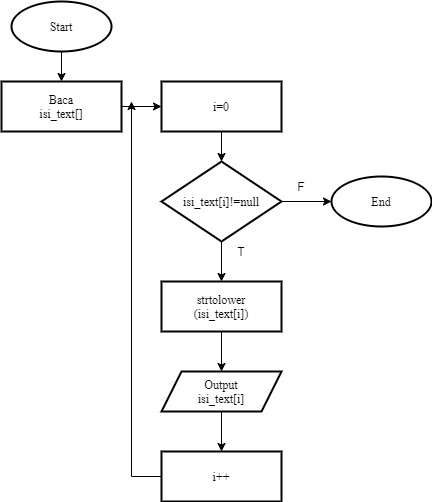
Pada *flowchart* ini, menjelaskan tahapan *preprocessing* data dokumen. Data dokumen hasil *upload/scan* yang sudah di*impor* ke dalam *database* dipilih untuk kemudian dilakukan sub proses *casefolding, cleaning*, *slangword, stopword,* dan penghapusan spasi antar kata. Setelah itu dokumen bersih disimpan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



Gambar 3 4.3 *Flowchart Preprocessing*

### 4.3.4. *Flowchart Casefolding*

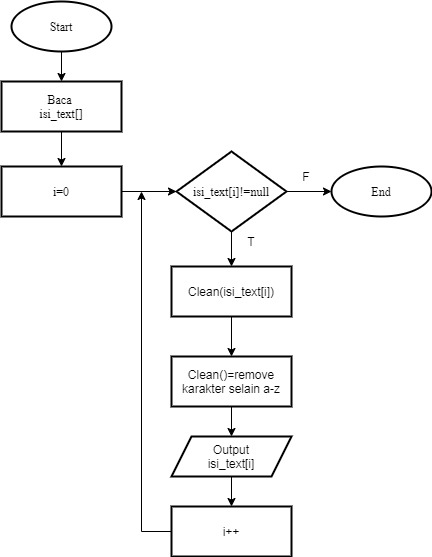
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *casefolding*. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan proses *casefolding* yaitu perubahan isi dokumen menjadi huruf kecil. Dengan fungsi strtolower(). Proses ini dijalankan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



Gambar 4 4.4 *Flowchart Casefolding*

### 4.3.5. *Flowchart Cleaning*

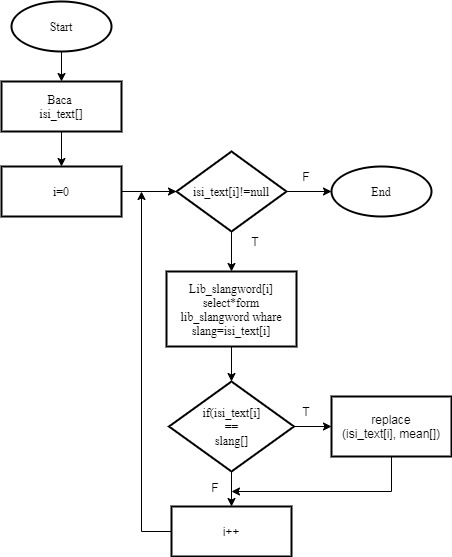
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *cleaning*. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan penghilangan pada karakter selain a-z dengan fungsi clean(). Proses ini dijalankan sampai dokumen habis atau dokumen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



Gambar 5 4.5 *Flowchart Cleaning*

### 4.3.6. *Flowchart Slangword*

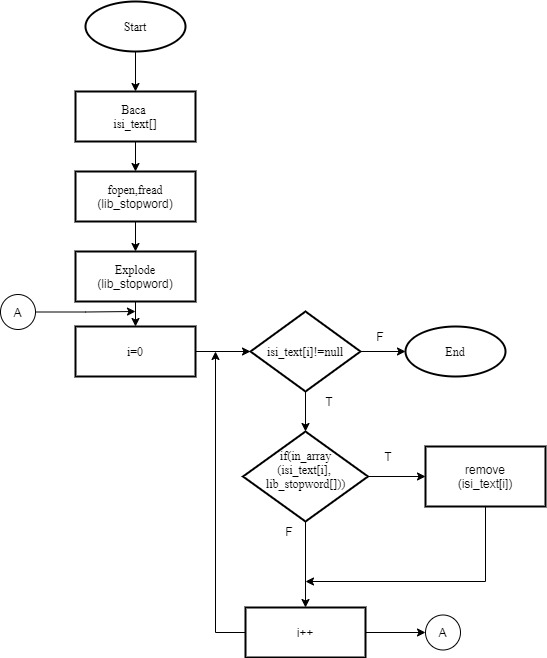
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *slangword.* Proses *slangword* diawali dengan membaca dokumen pada *array* isi\_text[] dan memanggil *library slangword* yang sama pada *array* isi\_text[i], jika ada yang sama maka dilakukan perintah replace(isi\_text[i],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



Gambar 6 4.6 *Flowchart Slangword*

### 4.3.7. *Flowchart Stopword*

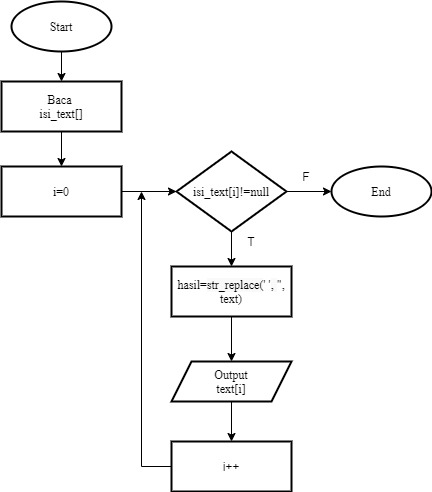
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *stopword,* proses ini diawali dengan membaca *library stopword* yang berisi kamus *stopword.* Kemudian dilakukan pencarian dengan fungsi in\_array (isi\_text[i], lib-stopword[]), artinya jika sebuah kata dalam array isi\_text[i]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



Gambar 7 4.7 *Flowchart Stopword*

### 4.3.8. *Flowchart* Hapus Spasi

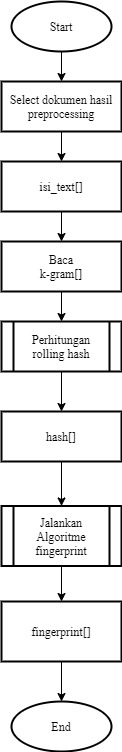
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu hapus spasi. Pada tahapan ini, dokumen yang sudah tersimpan pada *array* isi\_text[] satu per satu dilakukan penghapusan spasi antara satu kata dengan kata lainya. Menggunakan fungsi str\_replace(). Proses ini berulang sampai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



Gambar 8 4.8 *Flowchart* Hapus Spasi

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

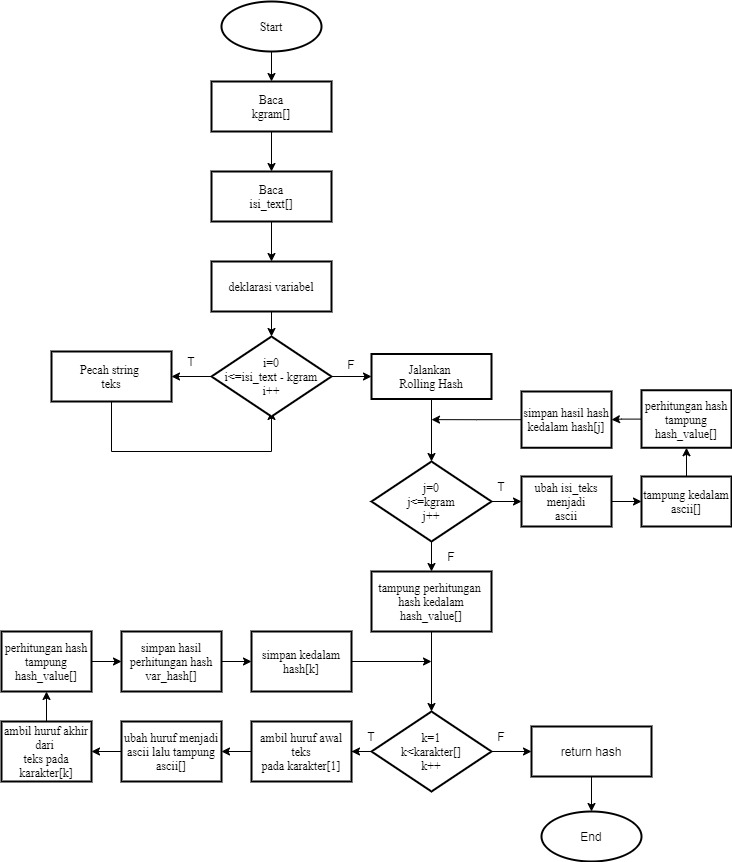
Pada *flowchart* ini, menjelaskan tahapan dari algoritme *winnowing* untuk mencari nilai *hash* dari sebuah dokumen. Setelah dokumen yang diproses pada tahap *preprocessing* dan telah disimpan kedalam *database,* kemudian dicarilah nilai *hash* dari dokumen tersebut, melalui beberapa sub proses *rolling 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 9 4.9 *Flowchart* Algoritme *Winnowing*

### 4.3.10. *Flowchart Rolling Hash*

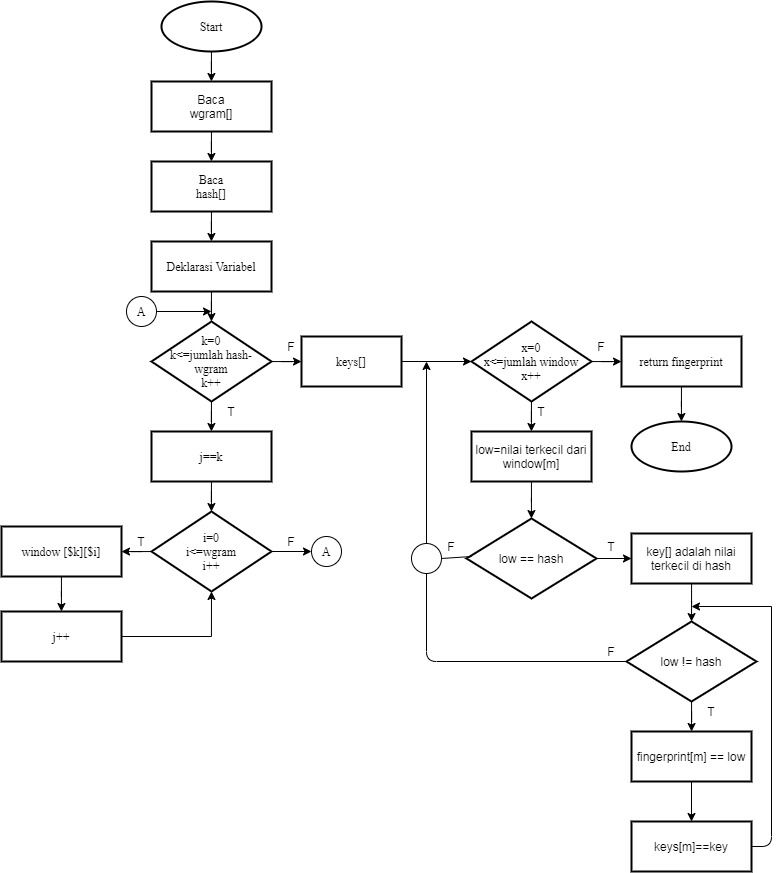
Pada *flowchart* ini, menjelaskan tahapan dari *rolling hash* yang merupakan sub proses dari algoritme *winnowing.* Proses pertama yang dilakukan adalah penentuan nilai *k-gram* untuk menentukan jumlah pecahan teks yang akan diproses, setelah teks dipecah, maka proses selanjutnya mencari nilai ASCII 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 10 4.10 *Flowchart Rolling Hash*

### 4.3.11. *Flowchart Fingerpring*

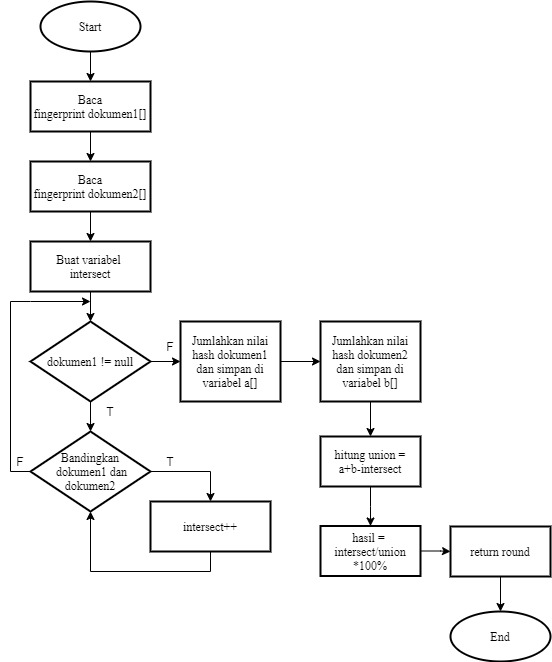
Pada *flowchart* ini, menjelaskan tahapan pencarian nilai *fingerprint* dari proses sebelumnya. Proses pertama yang dilakukan adalah pembentukan nilai *window,* pembentukan ini bertujuan untuk mengelompokan nilai *hash* yang telah didapatkan. Setelah dilakukan pembentukan *window,* maka tahap selenjutnya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



Gambar 11 4.11 *Flowchart Fingerpring*

### 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 12 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,<1mb  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 *casefolding()*  5. Lakukan proses *cleaning text()*  6. Lakukan proses *slangword()*  7. Lakukan proses *stopword()*  8. Lakukan proses penghapusan spasi  9.  *Output :* dokumen bersih  10. Proses simpan ke *database*  11. *End* |

### 4.4.4. Algoritme *Casefolding*

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

**Algoritme 4.4 *Casefolding***

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. Rubah 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 *Slangword*

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

**Algoritme 4.6 *Slangword***

|  |
| --- |
| 1. *Start*  2. Baca isi *array* isi\_text[]  3. i=0  4. *if*(isi\_text[i] ada)  5. *select lib\_slangword 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 *Stopword*

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

**Algoritme 4.7 *Stopword***

|  |
| --- |
| 1. *Start*  2. Baca *array* isi\_text[]  3. Buka & baca *array library stopword*  4. *Explode library stopword*  5. i=0  6. *if*(isi\_text[i] ada)  7. *if*(isi\_text[i] terdapat dalam library stopword[])  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 kedalam *array* hash\_value[]  13. Simpan kedalam *array* hash[j]  14. *Endif*  15. Tampung perhitungan *hash* kedalam *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 kedalam array ascii[]  19. ambil huruf akhir dari teks pada karakter[k]  20. proses perhitungan *hash*, lalu tampung kedalam *array* hash\_value[]  21. simpan hasil perhitungan *hash* kedalam *array* var\_hash[]  22. simpan kedalam 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\_GeryHifaldyPutra\_Abstrak. Hasil pengujian dapat dilihat pada Tabel 4.8.

Tabel 8 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\_GeryHifaldyPutra\_Abstrak. Hasil pengujian dapat dilihat pada Tabel 4.9.

Tabel 9 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\_RausalValinoAdjir\_Abstrak. Hasil pengujian dapat dilihat pada Tabel 4.10.

Tabel 10 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 11 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\_DickyAnanda\_Abstrak. Hasil pengujian dapat dilihat pada Tabel 4.12.

Tabel 12 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

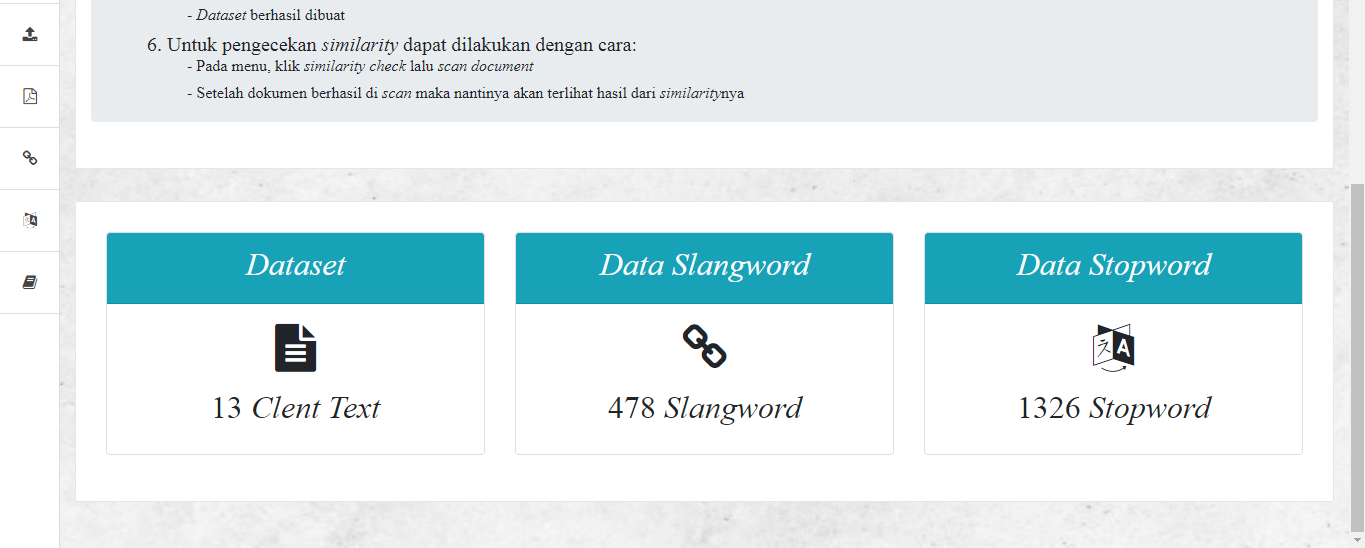
Setelah melewati beberapa tahap seperti implementasi metode, serta tahap pengujian sistem, pada tahap ini akan dijelaskan tampilan layar dari sistem yang telah dibuat.

### 4.6.1. Tampilan Layar *Home*

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



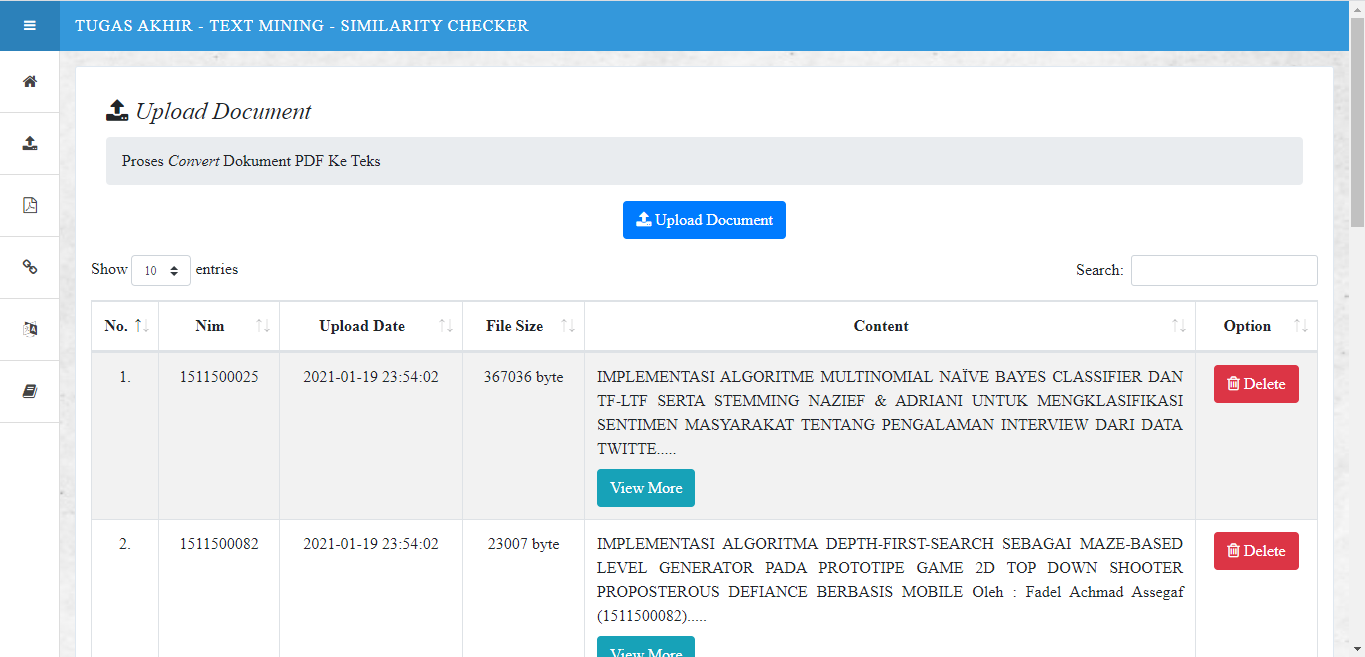
Gambar 13 4.13 Tampilan Layar *Home*



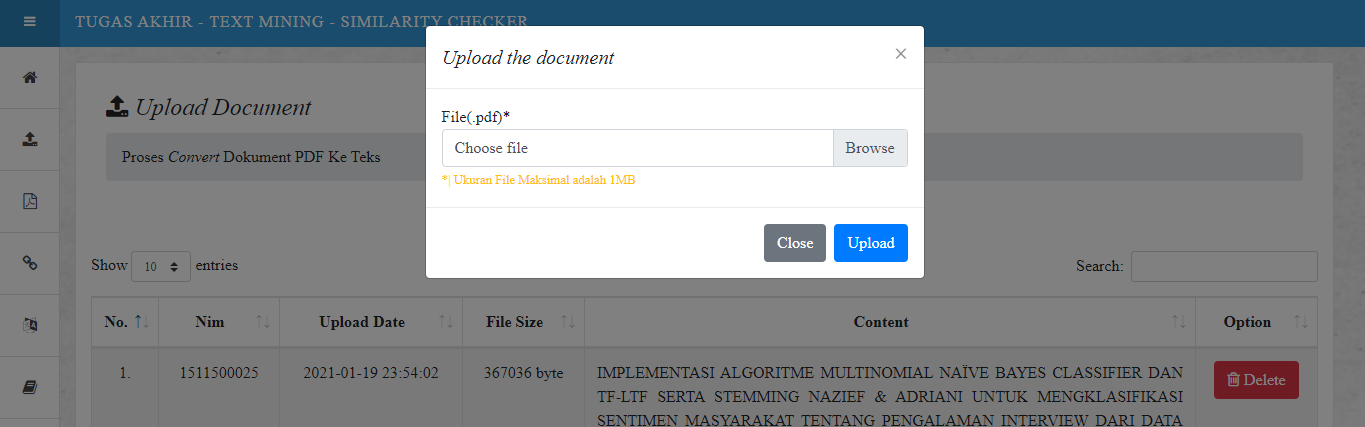
Gambar 14 4.13 Tampilan Layar *Home* (Lanjutan)

### 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 15 4.15 Tampilan Layar *Document Data*



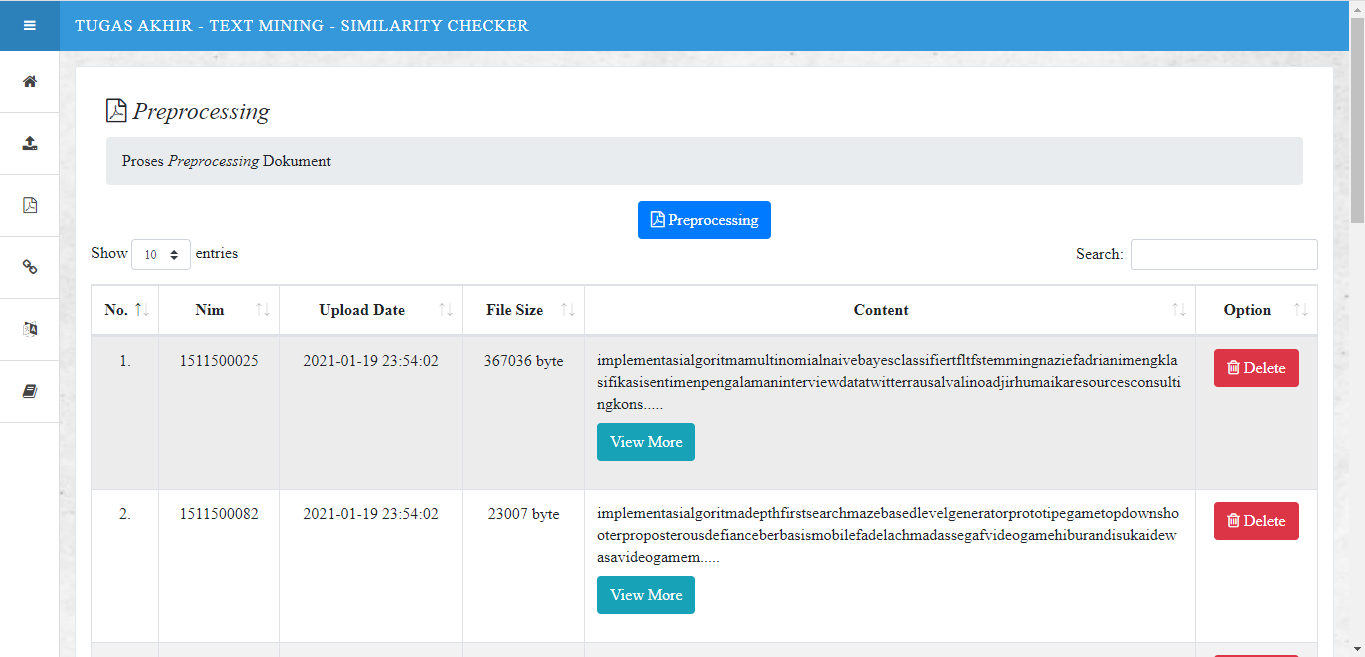
Gambar 16 4.16 Tampilan Layar *Document Data* (Lanjutan)

### 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*

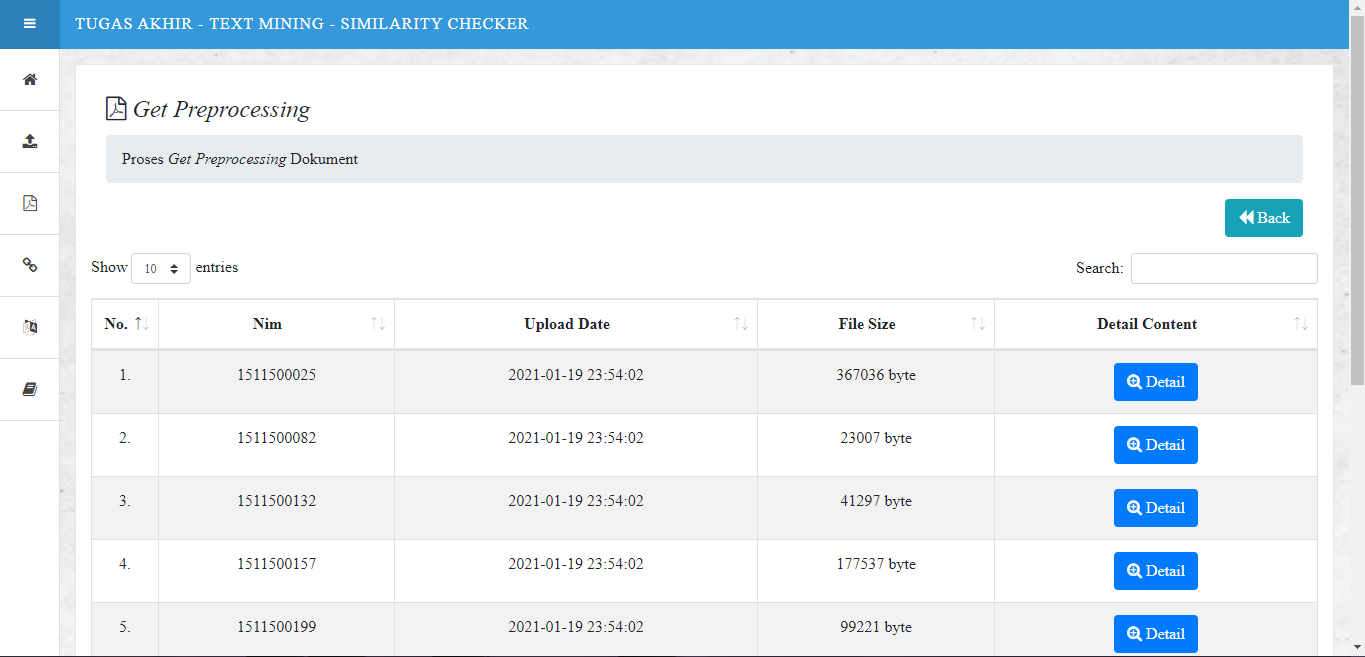
Pada Gambar 4.17 menjelaskan proses *preprocessing,* dari proses *casefolding,* menghilangkan karakter selain a-z, proses *slangword, stopword,* dan penghapusan spasi antar karakter.



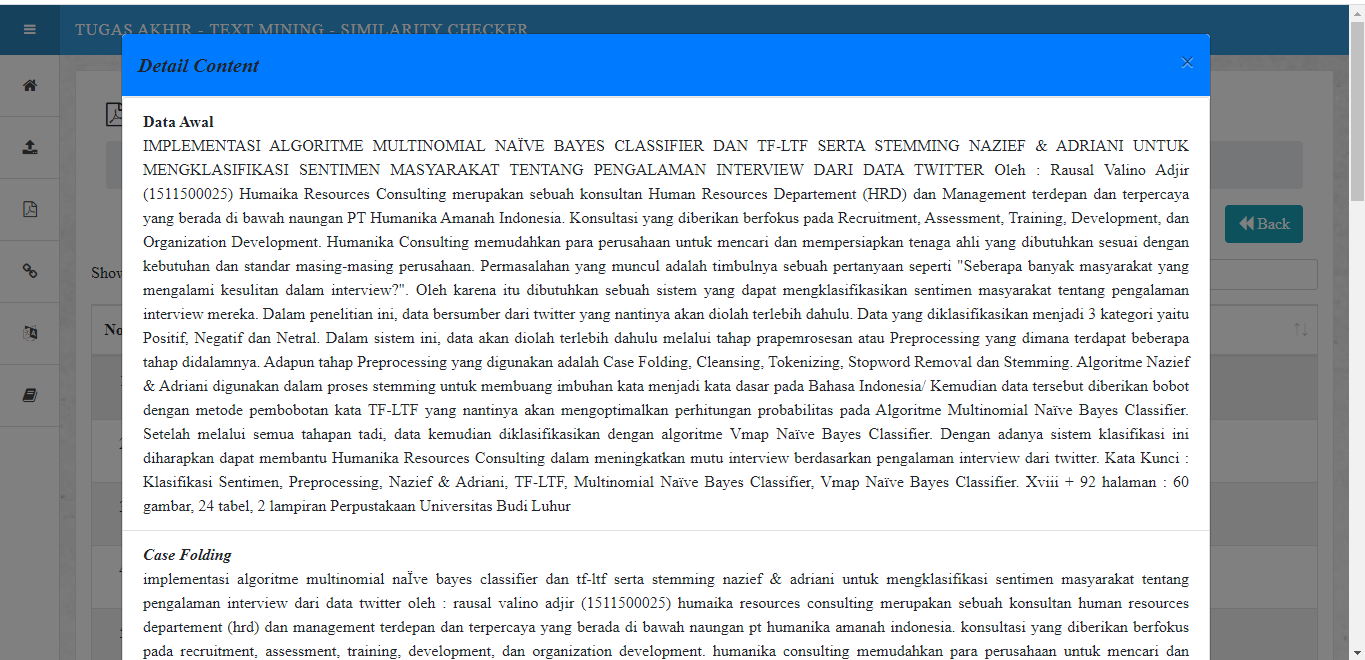
Gambar 17 4.17 Tampilan Layar *Preprocessing*

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.



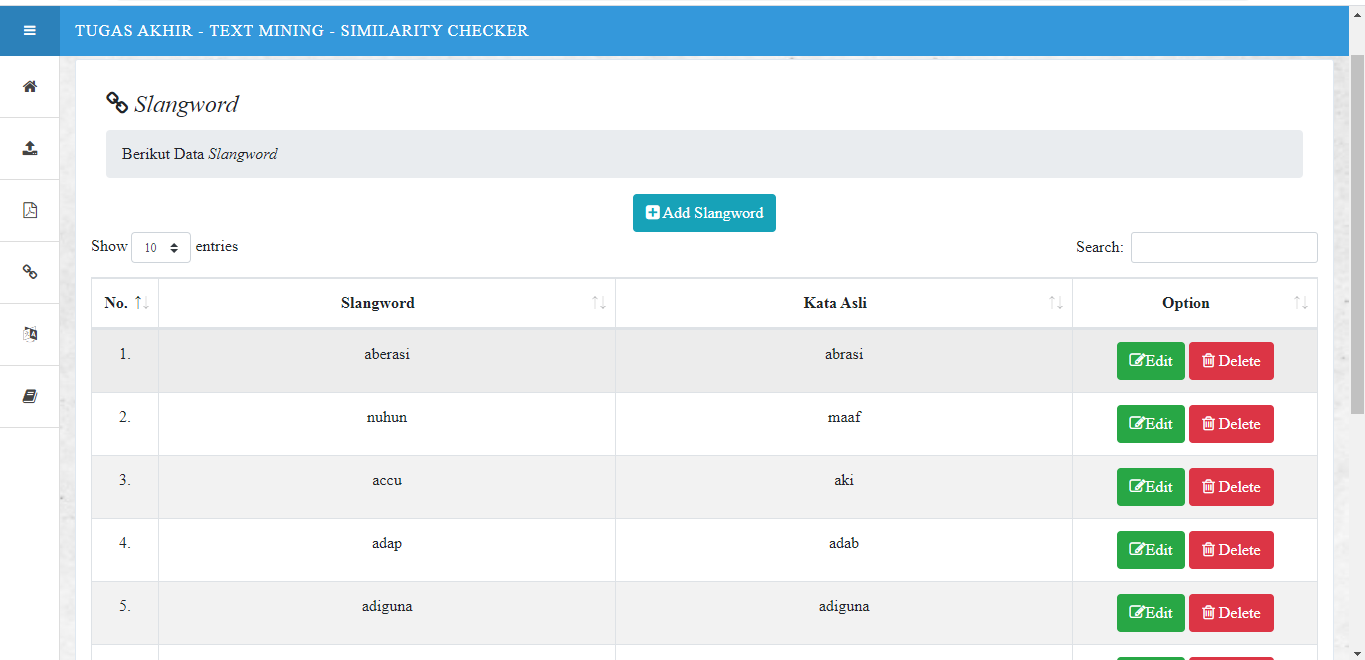
Gambar 18 4.18 Tampilan Layar Detail *Preprocessing*



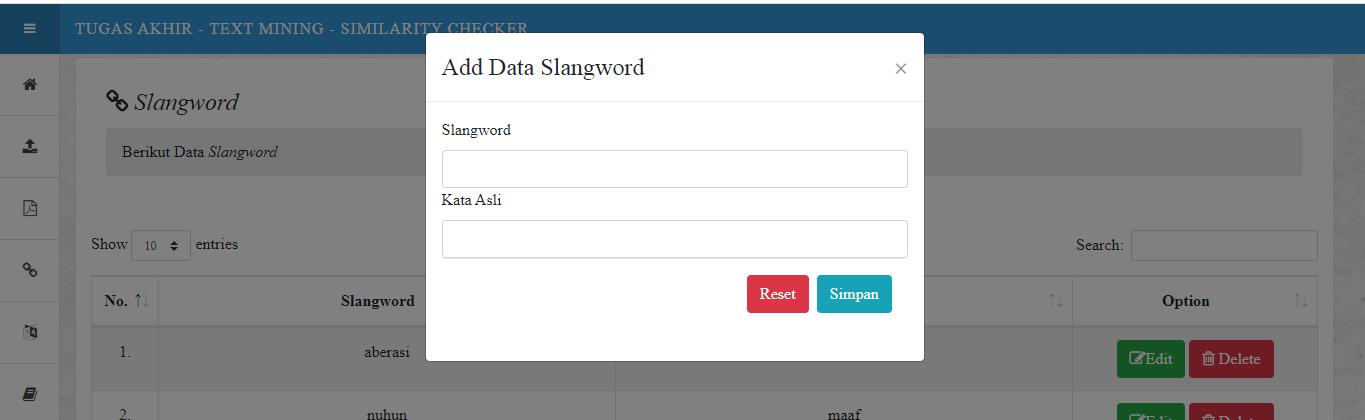
Gambar 19 4.19 Tampilan Layar Detail *Preprocessing* (Lanjutan)

### 4.6.4. Tampilan Layar *Slangword*

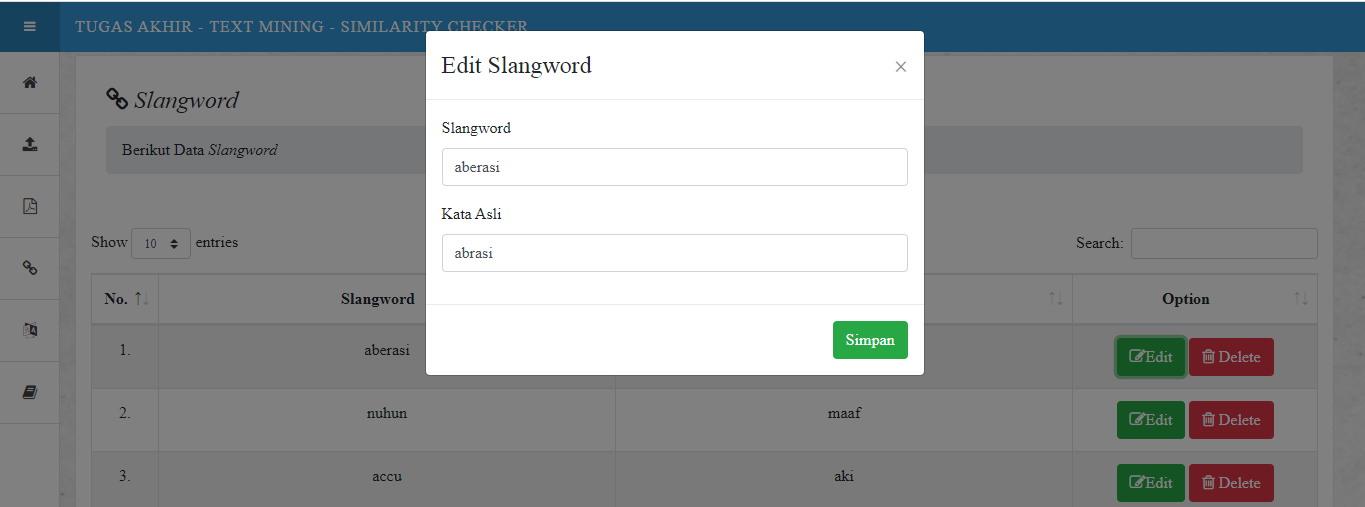
Tampilan layar pada Gambar 4.20, Gambar 4.21, dan Gambar 4.22 ini merupakan *menu slangword,* halaman ini berisi kumpulan data *slangword* yang digunakan untuk proses *preprocessing* nantinya, pada *menu* ini bisa dilakukan proses menambah, menghapus dan edit data *slangword.*



Gambar 20 4.20 Tampilan Layar *Menu Slangword*



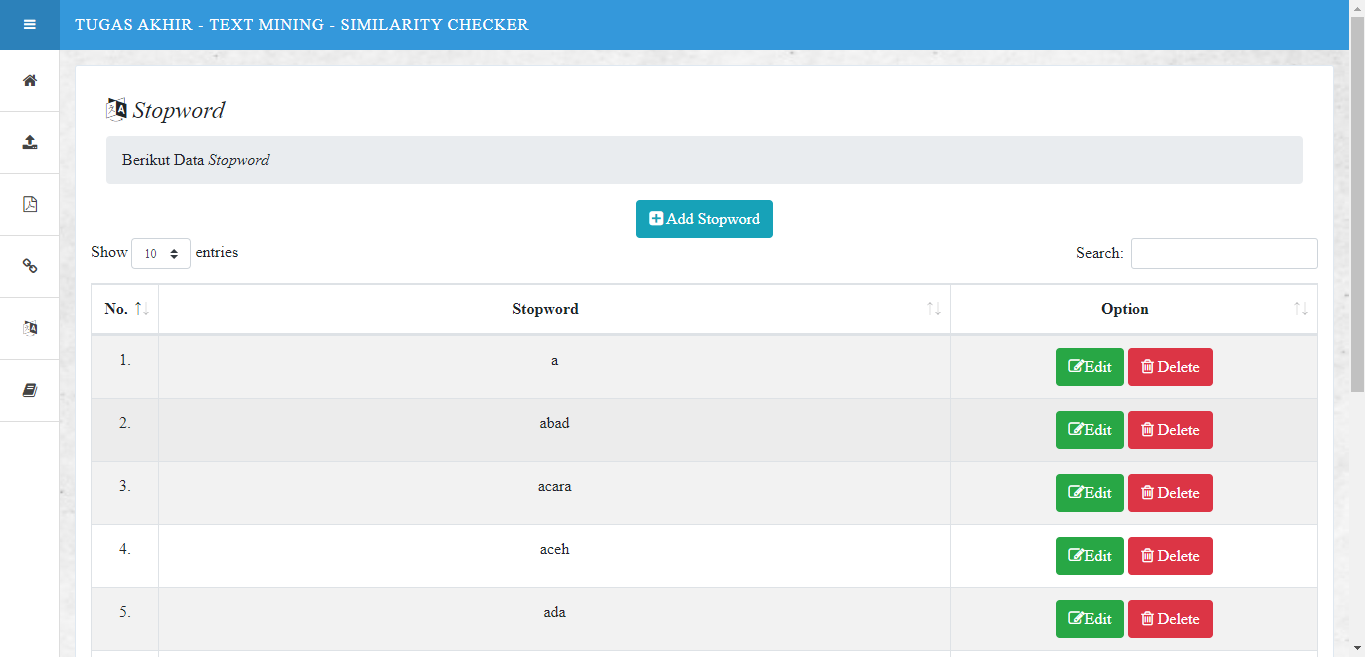
Gambar 21 4.21 Tampilan Layar *Menu Add Slangword*



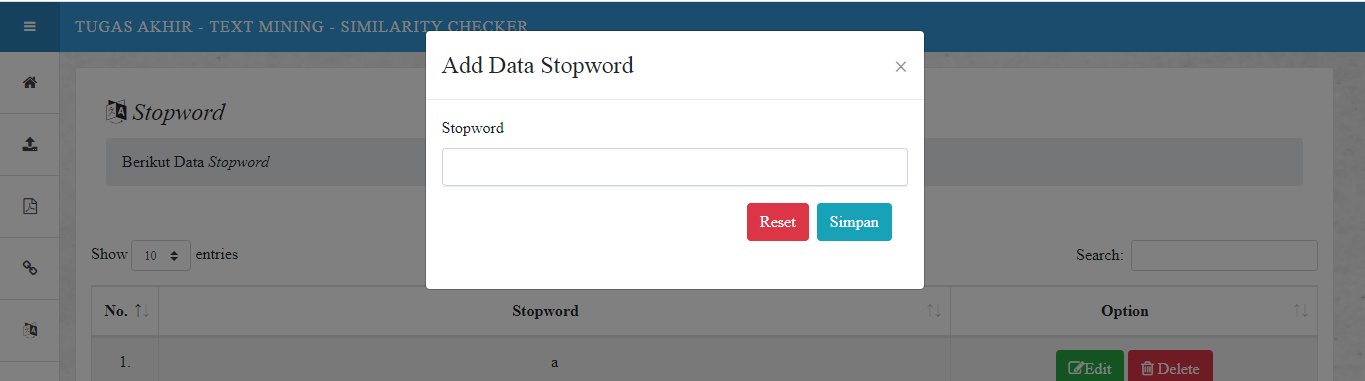
Gambar 22 4.22 Tampilan Layar *Menu Edit Slangword*

### 4.6.5. Tampilan Layar *Stopword*

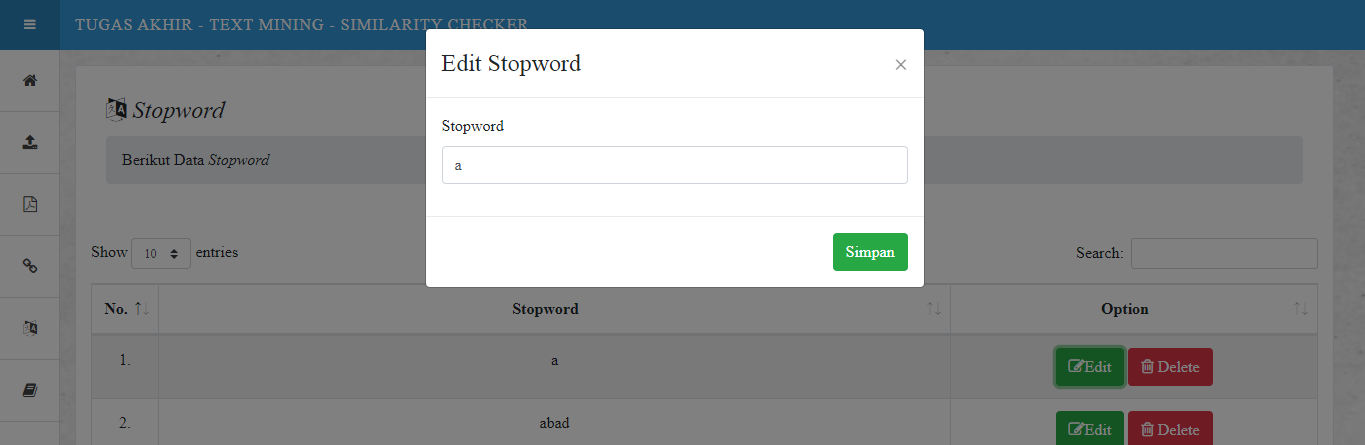
Tampilan layar pada Gambar 4.23, Gambar 4.24, dan Gambar 4.25 ini merupakan *menu stopword,* halaman ini berisi kumpulan data *stopword* yang digunakan untuk proses *preprocessing* nantinya, pada *menu* ini bisa dilakukan proses menambah, menghapus dan edit data *stopword.*



Gambar 23 4.23 Tampilan Layar Menu Stopword



Gambar 24 4.24 Tampilan Layar *Menu Add Stopword*



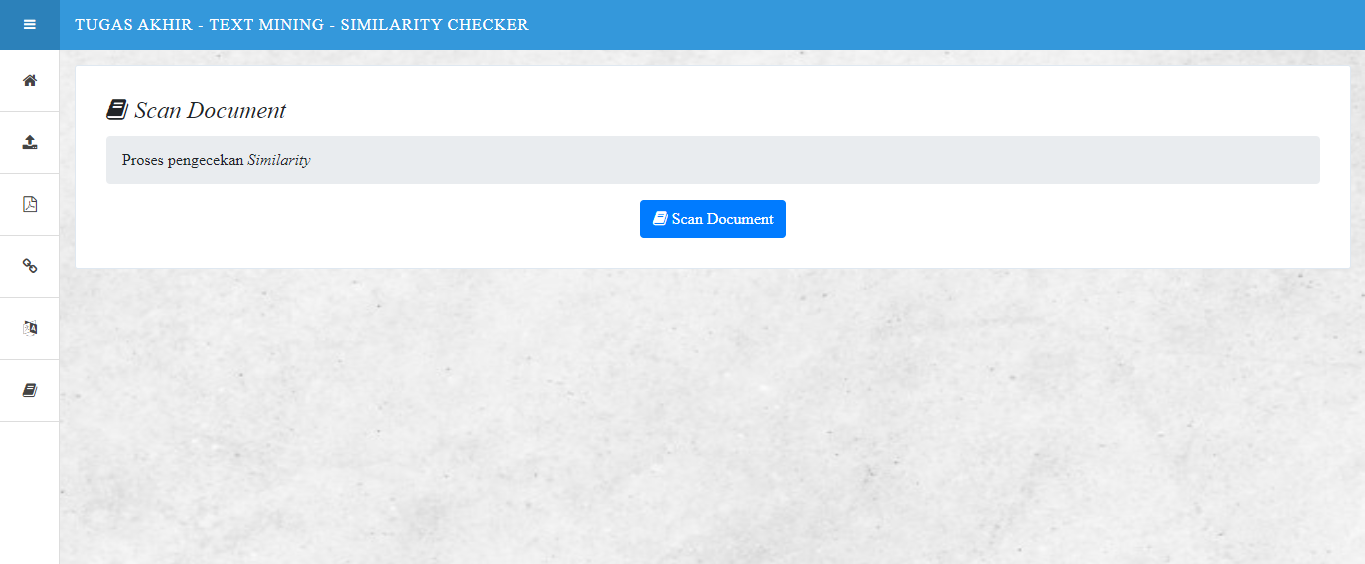
Gambar 25 4.25 Tampilan Layar *Menu* *Edit Stopword*

### 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*

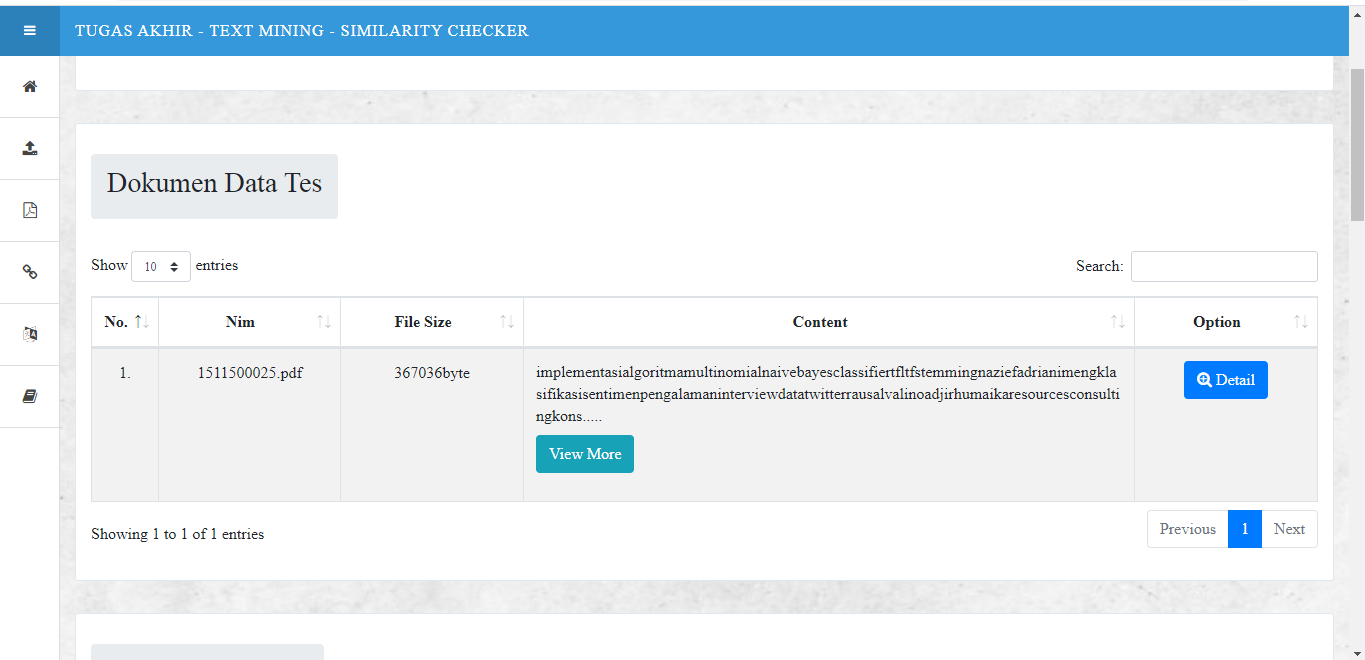
Tampilan layar pada Gambar 4.26 merupakan proses *scan* dokumen abstrak untuk melihat nilai *similarity* dari dokumen tersebut.



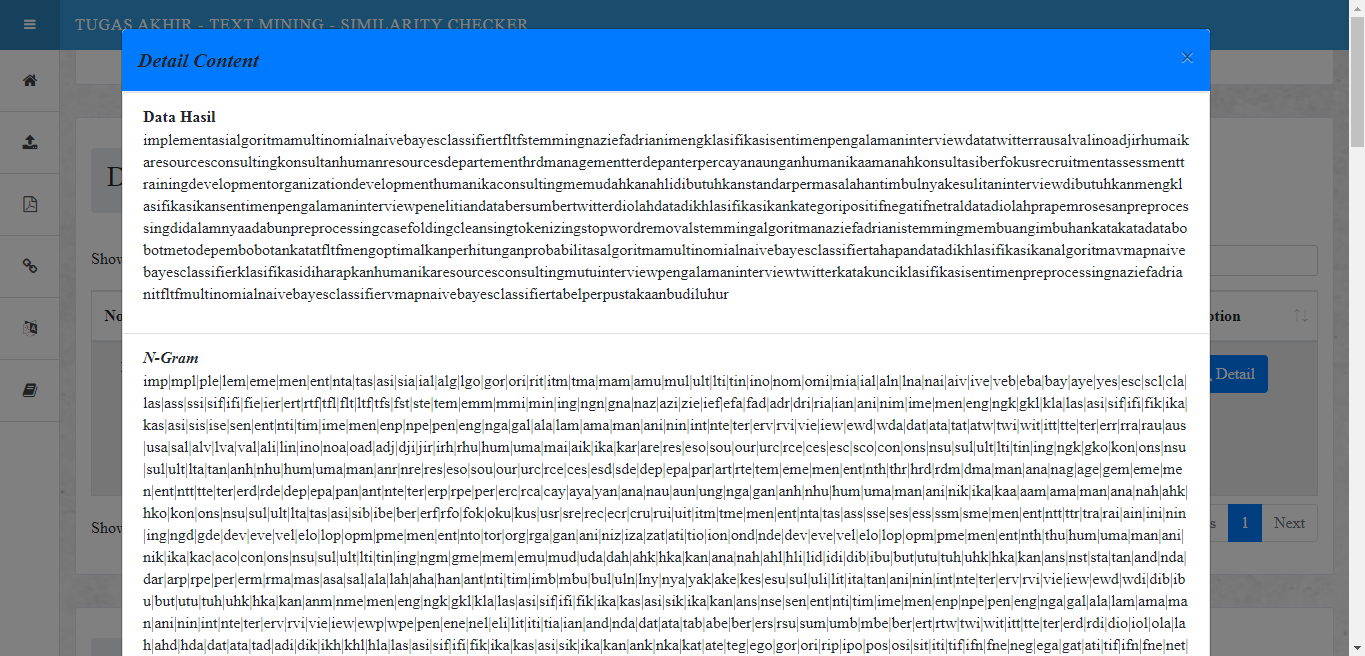
Gambar 26 4.26 Tampilan Layar *Menu Scan Document*

1. Tampilan Layar *Documen* 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.



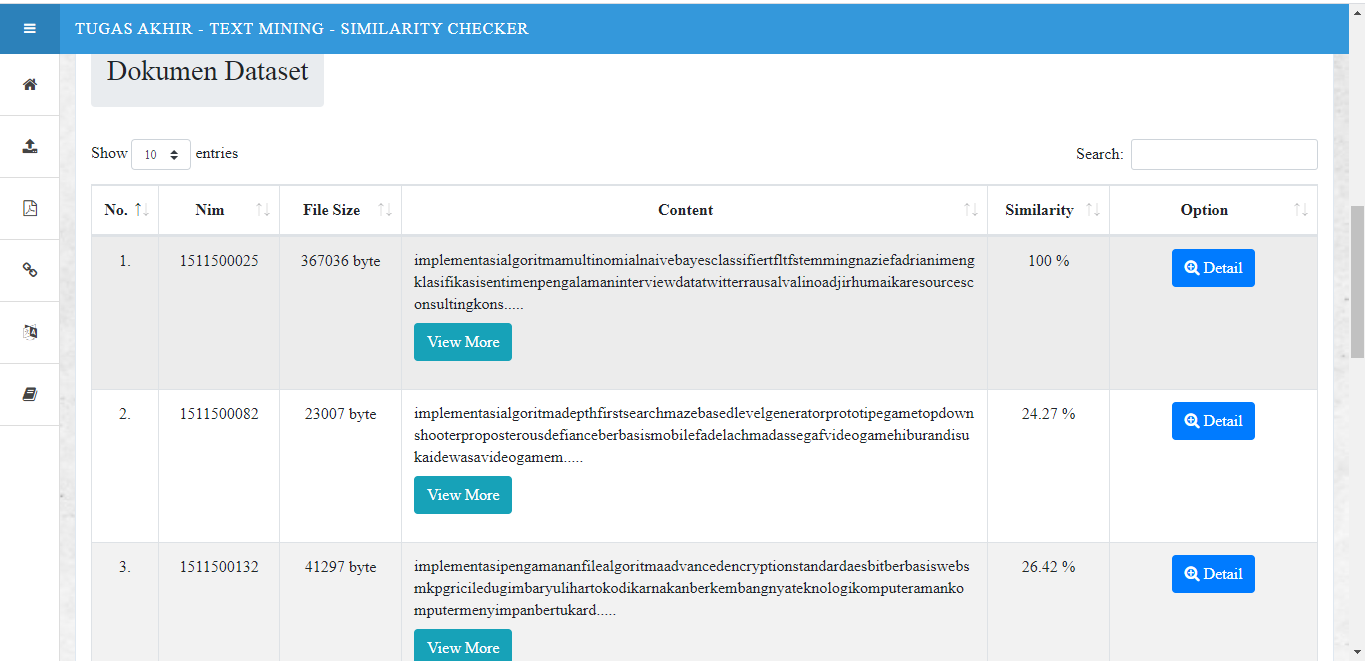
Gambar 27 4.27 Tampilan Layar *Document* Data Tes



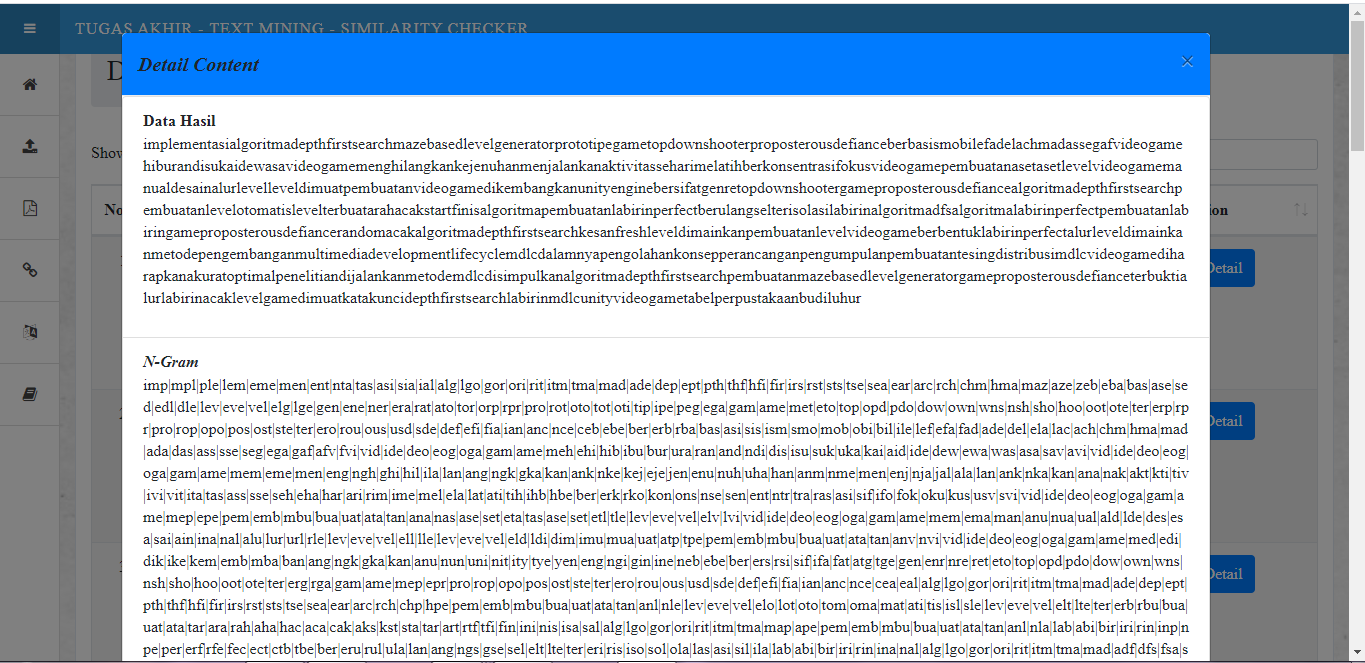
Gambar 28 4.28 Tampilan Layar *Document* Data Tes (Lanjutan)

1. Tampilan Layar Nilai *Similarity*

Tampilan layar pada Gambar 4.29 dan Gambar 4.30 merupakan dokumen abstrak *dataset* yang di ambil 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 29 4.29 Tampilan Layar Nilai *Similarity*



Gambar 30 4.30 Tampilan Layar Nilai *Similarity* (Lanjutan)