# 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. Pada Tabel 4.1 merupakan contoh dokumen abstrak yang sebelumnya berupa file *pdf* kemudian di *convert* menjadi teks. Berikut bisa dilihat di Tabel 4.1 berikut.

Tabel 3 4.1 Dokumen Abstrak

| **No** | **Data Dokumen** |
| --- | --- |
| 1 | IMPLEMENTASI ALGORITME MULTINOMIAL NAÏVE BAYES CLASSIFIER DAN TF-LTF SERTA STEMMING NAZIEF & ADRIANI UNTUK MENGKLASIFIKASI SENTIMEN MASYARAKAT TENTANG PENGALAMAN INTERVIEW DARI DATA TWITTER Oleh : Rausal Valino Adjir (1511500025) Humaika Resources Consulting merupakan sebuah konsultan Human Resources Departement (HRD) dan Management terdepan dan terpercaya yang berada di bawah naungan PT Humanika Amanah Indonesia. Konsultasi yang diberikan berfokus pada Recruitment, Assessment, Training, Development, dan Organization Development. Humanika Consulting memudahkan para perusahaan untuk mencari dan mempersiapkan tenaga ahli yang dibutuhkan sesuai dengan kebutuhan dan standar masing-masing perusahaan. Permasalahan yang muncul adalah timbulnya sebuah pertanyaan seperti "Seberapa banyak masyarakat yang mengalami kesulitan dalam interview?". Oleh karena itu dibutuhkan sebuah sistem yang dapat mengklasifikasikan sentimen masyarakat tentang pengalaman interview mereka. Dalam penelitian ini, data bersumber dari twitter yang nantinya akan diolah terlebih dahulu. Data yang diklasifikasikan menjadi 3 kategori yaitu Positif, Negatif dan Netral. Dalam sistem ini, data akan diolah terlebih dahulu melalui tahap prapemrosesan atau Preprocessing yang dimana terdapat beberapa tahap didalamnya. Adapun tahap Preprocessing yang digunakan adalah Case Folding, Cleansing, Tokenizing, Stopword Removal dan Stemming. Algoritme Nazief & Adriani digunakan dalam proses stemming untuk membuang imbuhan kata menjadi kata dasar pada Bahasa Indonesia/ Kemudian data tersebut diberikan bobot dengan metode pembobotan kata TF-LTF yang nantinya akan mengoptimalkan perhitungan probabilitas pada Algoritme Multinomial Naïve Bayes Classifier. Setelah melalui semua tahapan tadi, data kemudian diklasifikasikan dengan algoritme Vmap Naïve Bayes Classifier. Dengan adanya sistem klasifikasi ini diharapkan dapat membantu Humanika Resources Consulting dalam meningkatkan mutu interview berdasarkan pengalaman interview dari twitter. Kata Kunci : Klasifikasi Sentimen, Preprocessing, Nazief & Adriani, TF-LTF, Multinomial Naïve Bayes Classifier, Vmap Naïve Bayes Classifier. Xviii + 92 halaman : 60 gambar, 24 tabel, 2 lampiran Perpustakaan Universitas Budi Luhur |
| 2 | Implementasi Teknologi Augmented Reality Menggunakan Metode Marker Based Tracking Sebagai Media Briefing Penerimaan Calon Karyawan Magang pada PT. Sewiwi Indonesia Oleh: Deddy Rivaldy (1511500157) Penelitian ini bertujuan untuk mengembangkan aplikasi Augmented Reality pada bidang briefing yang medianya berupa kertas pada PT. Sewiwi Indonesia. Melalui media briefing berupa buku yang hanya berisi tulisan tidak jarang calon karyawan magang masih merasa bingung untuk membayangkan gambaran atau bentuk topologi yang akan dijadikan test perusahaan dan juga mudah rusak jika tetap menggunakan media kertas. Sehingga terkadang calon karyawan magang harus bertanya untuk mendapatkan informasi ataupun gambaran yang lebih jelas. Karena semakin berkembangnya zaman dan teknologi maka ada jalan keluar untuk mengatasi permasalahan tersebut yaitu dengan menggunakan aplikasi Augmented Reality (AR). Aplikasi ini nantinya akan berisi tentang Augmented Reality (AR) topologi jaringan yang akan dijadikan media briefing. Aplikasi ini dibangun dengan menggunakan bahasa pemograman C# dan juga menggunakan metode Marker Based Tracking yang nanti nya ada sebuah topologi jaringan berisi gambar sebagai salah satu marker penanda dan smartphone untuk memindai marker tersebut. Hasil dari penelitian ini menghasilkan aplikasi Augmented Reality (AR) yang dapat menyajikan sebuah gambar 3D dari Topologi Jaringan. Kata kunci: Augmented Reality, Android, Marker Based Tracking XIII+52halaman; 34 Gambar; 12 Tabel; 6 Lampiran Perpustakaan Universitas Budi Luhur |

### 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,* Menghilangkan spasi antar karakter.

1. *Case Folding*

Pada Tabel 4.2 merupakan proses untuk merubah isi teks dokumen menjadi huruf kecil, melalu proses *case folding,* sehingga isi teks menjadi huruh kecil semua*.* Berikut contohnya.

Tabel 4 4.2 Case Folding

|  |  |
| --- | --- |
| **Penjelasan** | **Data Dokumen** |
| Sebelum | IMPLEMENTASI DATA MINING UNTUK MEMPREDIKSI PENERIMAAN PERMOHONAN CALON NASABAH ASURANSI KESEHATAN DENGAN MENGGUNAKAN ALGORITMA NAÏVE BAYES BERBASIS WEB PADA PT. ASURANSI SINARMAS Oleh : Lady Caesar Sevika Detale (1511503581) Kesehatan merupakan bagian yang amat penting bagi kehidupan manusia. PT. Asuransi Sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan. Banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan, membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan. Hal tersebut dikarenakan pemberian jaminan kesehatan memiliki resiko yang cukup tinggi. Di sisi lain, banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi, membuat pihak asuransi mengalami kerugian semakin besar. Maka dari itu, upaya untuk meminimalisir adanya permasalahan tersebut, pihak asuransi harus lebih memperhatikan lagi terhadap penerimaan permohonan calon nasabah. |
| Sesudah | judul : aplikasi augmented reality sspol(solar system planet and orbital lines). sekolah dasar negri cipadu 3 adalah sebuah sekolah yang berada di tangerang selatan. tata surya adalah suatu pelajaran ilmu pengetahuan alam di sdn cipadu 3, media pembelajaran yang digunakan untuk mempelajari planet dan tata surya masih menggunakan globe, presentasi, dan buku. dalam menentukan aplikasi dibutuhkan beberapa tahap yaitu mengumpulkan data dan informasi melalui catatan kuliah, analisis, perancangan aplikasi dan implementeasi aplikasi. dengan menggunakan media pembelajaran menggunakan augmented reality dengan media android dan dengan animasi 3d diharapkan siswa dan siswi jadi menarik untuk belajar, mengerti bentuk dari planet tata surya dan bisa mempelajari planet tata surya dirumah. . menggunakan video siswa-siswi juga dapat melihat sambil mendengarkan video yang disampaikan. augmented reality adalah sebuah teknologi masa kini yang menggabungkan antara dunia maya dan dunia nyata secara realtime. |

1. *Cleaning*

Pada Tabel 4.3 merupakan proses untuk menghapus karakter selain a-z*.* Sehingga isi teks yang bukan huruf a-z akan dihapus, seperti angka, simbol, dan seterusnya. Berikut contohnya.

Tabel 5 4.3 *Cleaning*

|  |  |
| --- | --- |
| **Penjelasan** | **Data Dokumen** |
| Sebelum | implementasi data mining untuk memprediksi penerimaan permohonan calon nasabah asuransi kesehatan dengan menggunakan algoritma naÏve bayes berbasis web pada pt. asuransi sinarmas oleh : lady caesar sevika detale (1511503581) kesehatan merupakan bagian yang amat penting bagi kehidupan manusia. pt. asuransi sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan. banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan, membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan. hal tersebut dikarenakan pemberian jaminan kesehatan memiliki resiko yang cukup tinggi. di sisi lain, banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi, membuat pihak asuransi mengalami kerugian semakin besar. maka dari itu, upaya untuk meminimalisir adanya permasalahan tersebut, pihak asuransi harus lebih memperhatikan lagi terhadap penerimaan permohonan calon nasabah. |
| Sesudah | implementasi data mining untuk memprediksi penerimaan permohonan calon nasabah asuransi kesehatan dengan menggunakan algoritma na ve bayes berbasis web pada pt asuransi sinarmas oleh lady caesar sevika detale kesehatan merupakan bagian yang amat penting bagi kehidupan manusia pt asuransi sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan hal tersebut dikarenakan pemberian jaminan kesehatan memiliki resiko yang cukup tinggi di sisi lain banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi membuat pihak asuransi mengalami kerugian semakin besar maka dari itu upaya untuk meminimalisir adanya permasalahan tersebut pihak asuransi harus lebih memperhatikan lagi terhadap penerimaan permohonan calon nasabah |

1. Mengganti *slang word*

Pada Tabel 4.4 merupakan proses untuk mengganti kata yang tidak baku menjadi kata baku, seperti kata analisa dirubah menjadi analisis*.* Berikut contohnya.

Tabel 6 4.4 Mengganti *slang word*

|  |  |
| --- | --- |
| **Penjelasan** | **Data Dokumen** |
| Sebelum | implementasi data mining untuk memprediksi penerimaan permohonan calon nasabah asuransi kesehatan dengan menggunakan algoritma na ve bayes berbasis web pada pt asuransi sinarmas oleh lady caesar sevika detale kesehatan merupakan bagian yang amat penting bagi kehidupan manusia pt asuransi sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan hal tersebut dikarenakan pemberian jaminan kesehatan memiliki resiko yang cukup tinggi di sisi lain banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi membuat pihak asuransi mengalami kerugian semakin besar maka dari itu upaya untuk meminimalisir adanya permasalahan tersebut pihak asuransi harus lebih memperhatikan lagi terhadap penerimaan permohonan calon nasabah |
| Sesudah | implementasi data mining untuk memprediksi penerimaan permohonan calon nasabah asuransi kesehatan dengan menggunakan algoritma naive bayes berbasis web pada pt asuransi sinarmas oleh lady caesar sevika detale kesehatan merupakan bagian yang amat penting bagi kehidupan manusia pt asuransi sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan hal tersebut dikarenakan pemberian jaminan kesehatan memiliki risiko yang cukup tinggi di sisi lain banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi membuat pihak asuransi mengalami kerugian semakin besar maka dari itu upaya untuk meminimalisir adanya permasalahan tersebut pihak asuransi harus lebih memperhatikan lagi terhadab penerimaan permohonan calon nasabah |

1. Menghapus *stop word*

Pada Tabel 4.5 merupakan proses untuk menghapus kata yang diabaikan dalam tahap *preprocessing,* seperti kata dan, untuk, bagi, dan seterusnya. Berikut contohnya.

Tabel 7 4.5 Menghapus *stop word*

|  |  |
| --- | --- |
| **Penjelasan** | **Data Dokumen** |
| Sebelum | implementasi data mining untuk memprediksi penerimaan permohonan calon nasabah asuransi kesehatan dengan menggunakan algoritma naive bayes berbasis web pada pt asuransi sinarmas oleh lady caesar sevika detale kesehatan merupakan bagian yang amat penting bagi kehidupan manusia pt asuransi sinarmas adalah salah satu asuransi yang menyediakan produk asuransi kesehatan banyaknya masyarakat yang mendaftar dan mengajukan permohonan asuransi kesehatan membuat pihak asuransi harus lebih memperhatikan mengenai penerimaan permohonan asuransi kesehatan hal tersebut dikarenakan pemberian jaminan kesehatan memiliki risiko yang cukup tinggi di sisi lain banyaknya nasabah yang sering terlambat dan kurang peduli membayar premi membuat pihak asuransi mengalami kerugian semakin besar maka dari itu upaya untuk meminimalisir adanya permasalahan tersebut pihak asuransi harus lebih memperhatikan lagi terhadab penerimaan permohonan calon nasabah |
| Sesudah | implementasi data mining memprediksi penerimaan permohonan calon nasabah asuransi kesehatan algoritma naive bayes berbasis web asuransi sinarmas lady caesar sevika detale kesehatan asuransi sinarmas asuransi asuransi kesehatan banyaknya mendaftar mengajukan permohonan asuransi kesehatan asuransi memperhatikan penerimaan permohonan asuransi kesehatan pemberian jaminan kesehatan risiko sisi banyaknya nasabah terlambat peduli membayar premi asuransi kerugian meminimalisir permasalahan asuransi memperhatikan terhadab penerimaan permohonan calon nasabah |

1. Menghilangkan spasi antar karakter

Pada Tabel 4.6 merupakan proses untuk menghilangkan spasi antar karakter. Berikut contohnya.

Tabel 8 4.6 Menghilangkan spasi antar karakter

| **Penjelasan** | **Data Dokumen** |
| --- | --- |
| Sebelum | implementasi data mining memprediksi penerimaan permohonan calon nasabah asuransi kesehatan algoritma naive bayes berbasis web asuransi sinarmas lady caesar sevika detale kesehatan asuransi sinarmas asuransi asuransi kesehatan banyaknya mendaftar mengajukan permohonan asuransi kesehatan asuransi memperhatikan penerimaan permohonan asuransi kesehatan pemberian jaminan kesehatan risiko sisi banyaknya nasabah terlambat peduli membayar premi asuransi kerugian meminimalisir permasalahan asuransi memperhatikan terhadab penerimaan permohonan calon nasabah |
| Sesudah | implementasidataminingmemprediksipenerimaanpermohonancalonnasabahasuransikesehatanalgoritmanaivebayesberbasiswebasuransisinarmasladycaesarsevikadetalekesehatanasuransisinarmasasuransiasuransikesehatanbanyaknyamendaftarmengajukanpermohonanasuransikesehatanasuransimemperhatikanpenerimaanpermohonanasuransikesehatanpemberianjaminankesehatanrisikosisibanyaknyanasabahterlambatpedulimembayarpremiasuransikerugianmeminimalisirpermasalahanasuransimemperhatikanterhadabpenerimaanpermohonancalonnasabah |

### 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 4.7 berikut.

Tabel 9 4.7 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.

* 1. 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 10 4.8 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 4.9 berikut.

Tabel 11 4.9 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 4.10 dan Tabel 4.11 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 4.10 berikut.

Tabel 12 4.10 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 4.11 berikut.

Tabel 13 4.11 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”.

* 1. 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

* 1. 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 4.12 berikut.

Tabel 14 4.12 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%**

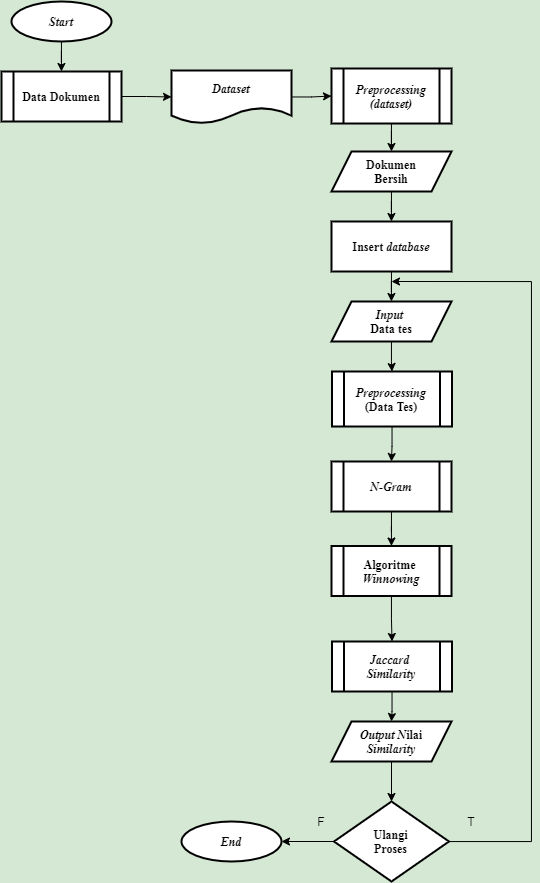
Sehingga tingkat *similarity* dari dokumenadalah 57,1%

## 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.

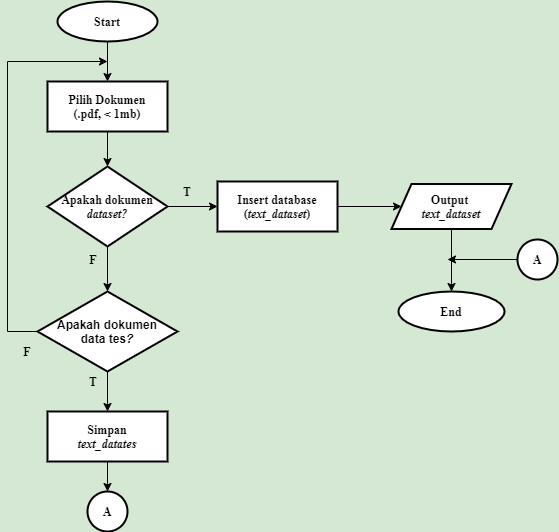


Gambar 3 4.1 *Flowchart* Keseluruhan Sistem

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

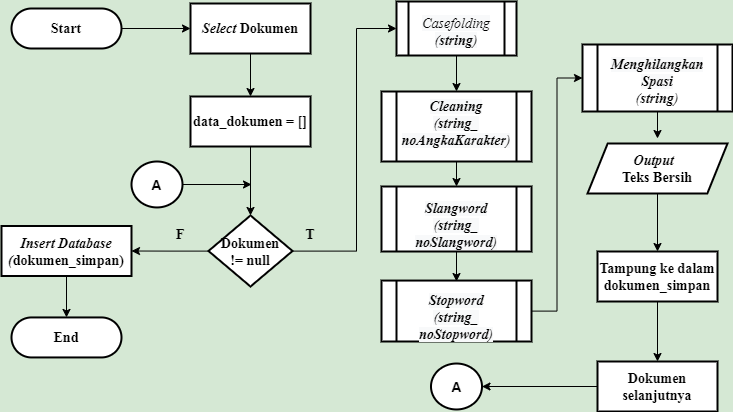
Pada *flowchart* ini, menjelaskan tahapan proses pengumpulan data dokumen *dataset* atau data tes. Dokumen yang bisa diproses berupa file *.pdf* serta berukuran <1 mb, setelah itu dokumen yang merupakan *dataset* akan disimpan kedalam *database,* sedangkan dokumen data tes akan disimpan disebuah array *text\_datates* untuk proses selanjutnya, yaitu pencarian nilai *similarity* dokumen*.* Proses yang sama akan berulang jika ingin memproses dokumen kembali.



Gambar 4 4.2 *Flowchart* Data Dokumen

### 4.3.3. *Flowchart Preprocessing Dataset*

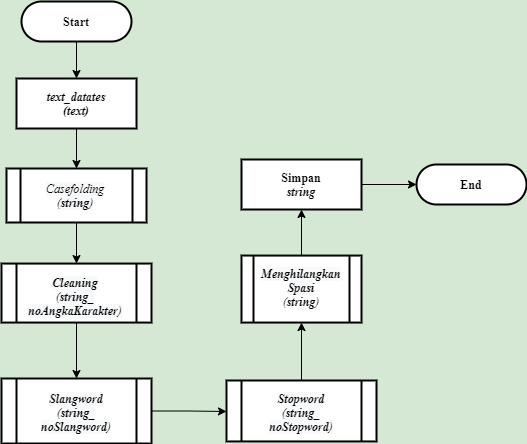
Pada *flowchart* ini, menjelaskan tahapan *preprocessing* *dataset*. Data Dokumen abstrak yang sudah disimpan di *database* yang merupakan *dataset* akan di *select,* kemudian disimpan disebuah *array*, selanjutnya akan dilakukan proses *casefolding, cleaning,* mengganti *slang word,* menghapus *stop word,* dan menghilangkan spasi antar kata, kemudian ditampung disebuah *array* dan di*insert* kedalam *database.* Hasil dari tahapan ini berupa teks bersih, yang digunakan untuk proses pencarian nilai *similarity.*



Gambar 5 4.3 *Flowchart Preprocessing Dataset*

### 4.3.4. *Flowchart Preprocessing* Data Tes

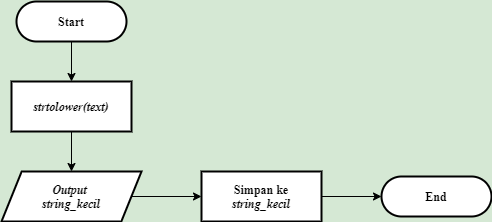
Pada *flowchart* ini, menjelaskan tahapan *preprocessing* data tes. Data Dokumen abstrak yang akan dijadikan data tes disimpan disebuah *array text,* kemudian isi dari *array text* akan dilakukan proses *casefolding, cleaning,* mengganti *slang word,* menghapus *stop word,* dan menghilangkan spasi antar kata, kemudian ditampung disebuah *array* *string.* Selanjutnya digunakan untuk proses pencarian nilai *similarity.*



Gambar 6 4.4 *Flowchart Preprocessing* Data Tes

### 4.3.5. *Flowchart Case folding*

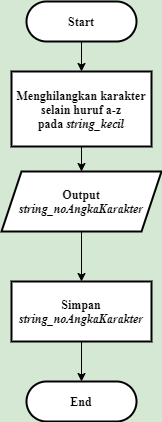
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *case folding*. Pada tahapan ini isi dokumen berupa teks akan dilakukan proses *case folding,* yaitu proses merubah isi teks menjadi huruf kecil, kemudian disimpan didalam *array string\_kecil*, yang akan digunakan untuk proses selanjutnya.



Gambar 7 4.5 *Flowchart Case folding*

### 4.3.6. *Flowchart Cleaning*

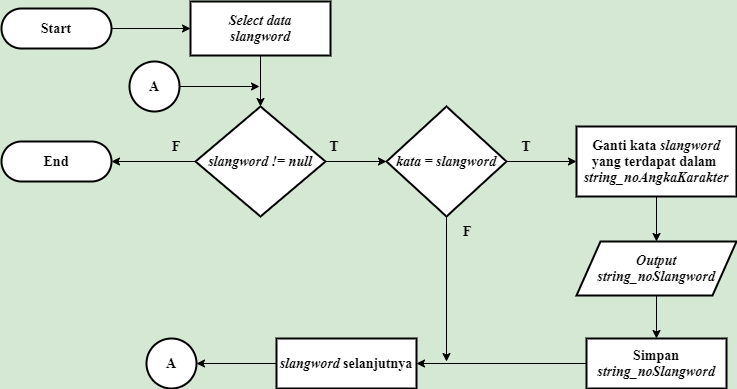
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *cleaning*. Pada tahapan ini isi teks yang telah diproses pada tahap *case folding* akan diproses lagi melalui proses *cleaning,* yaitu menghilangkan karakter selain huruh a-z yang ada didalam *array string\_kecil*, hasil dari proses ini disimpan didalam *array string\_noAngkaKarakter,* yang akan digunakan untuk proses selanjutnya.



Gambar 8 4.6 *Flowchart Cleaning*

### 4.3.7. *Flowchart Slang word*

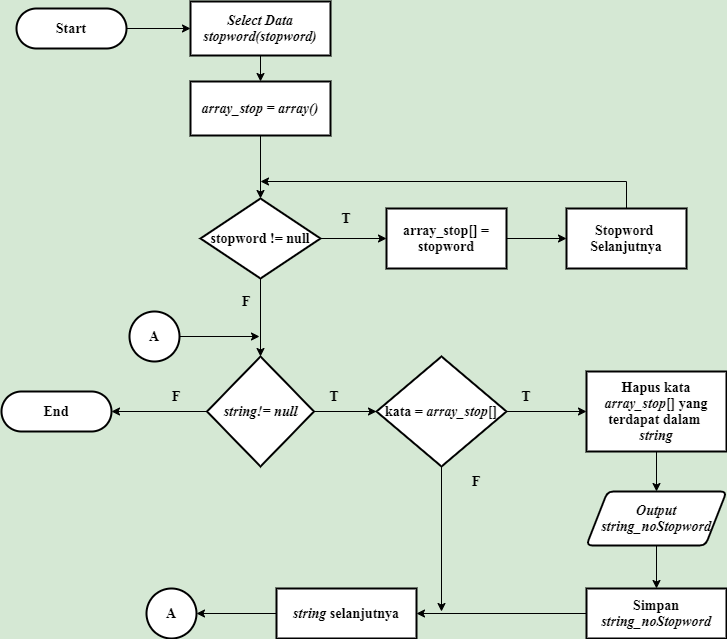
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *slang word.* Proses *slang word* diawali dengan *select* data *slang word* yang ada di *database,* kemudian akan diperiksa teks yang ada di *array string\_noAngkaKarakter,* apakah ada kata yang sama dengan data *slang word,* jika menemukan kata yang sama maka kata tersebut akan diganti dengan data yang ada di data *slang word,* selanjutnya teks yang sudah diproses disimpan didalam *array string\_noSlangword,* yang akan digunakan untuk proses selanjutnya.



Gambar 9 4.7 *Flowchart Slang Word*

### 4.3.8. *Flowchart Stop word*

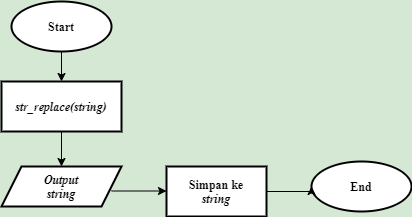
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu *stop word.* Proses *stop word* diawali dengan *select* data *stop word* yang ada di *database,* kemudian akan diperiksa teks yang ada di *array string\_noSlangword,* apakah ada kata yang sama dengan data *stop word,* jika menemukan kata yang sama maka kata tersebut akan dihapusdan teks yang masih ada disimpan didalam *array string\_noStopword,* yang akan digunakan untuk proses selanjutnya.



Gambar 10 4.8 *Flowchart Stop Word*

### 4.3.9. *Flowchart* Hapus Spasi

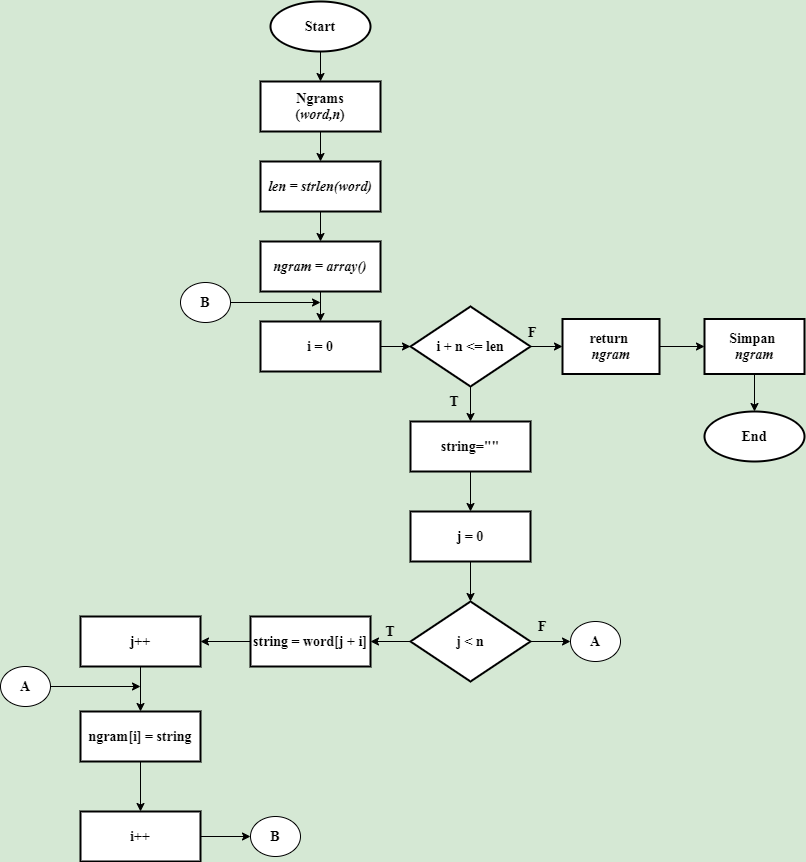
Pada *flowchart* ini, menjelaskan sub proses tahapan dari *preprocessing* yaitu hapus spasi*.* Pada proses ini teks yang sudah tersimpan didalam *array string\_noStopword* akan dihapus spasi antar katanya, sehingga teks ini menjadi satu, sehingga menghasilkan teks bersih, kemudian disimpan kedalam *array string*, yang akan digunakan untuk proses selanjutnya.



Gambar 11 4.9 *Flowchart* Hapus Spasi

### 4.3.10. *Flowchart N-Gram*

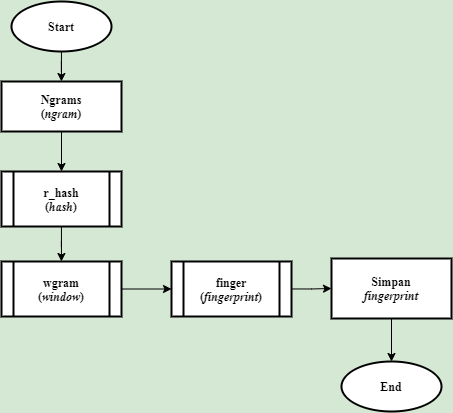
Pada *flowchart* ini, menjelaskan tahapan dari metode *n-gram,* pada tahap ini teks yang sudah di*preprocessing* yang menghasilkan teks bersih akan di proses melalui *n-gram,* yaitu dengan memisahkan *string* sepanjang jumlah n yang sudah ditentukan dan akan dihitung pergeserannya secara terus menurus ke depan sejumlah nilai n sampai akhir isi teks. Kemudian hasilnya akan disimpan didalam *ngram,* hasil *n-gram* ini akan dicari nilai hasnya pada proses selanjutnya.



Gambar 12 4.10 *Flowchart* *N-Gram*

### 4.3.11. *Flowchart* Algoritme *Winnowing*

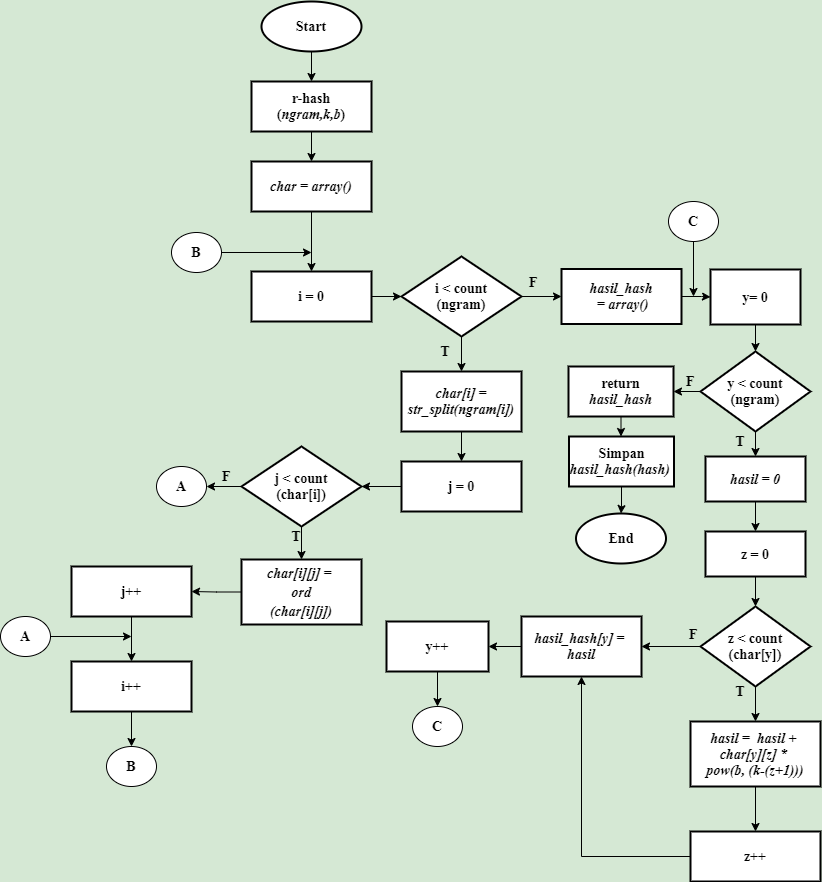
Pada *flowchart* ini, menjelaskan tahapan dari algoritme *winnowing* untuk mencari nilai *hash* dari sebuah dokumen. Setelah dokumen yang diproses pada tahap *n-gram,* kemudian carilah nilai *hash* dari teks tersebut, melalui beberapa sub proses *rolling hash,* pembentukan *window,* serta pencarian *fingerprint.* Setelah mendapatkan nilai *hash* dari dokumen yang diproses, maka setelah itu dapat dilakukan proses pencarian tingkat *similarity* dari sebuah dokumen abstrak.



Gambar 13 4.11 *Flowchart* Algoritme *Winnowing*

### 4.3.12. *Flowchart Rolling Hash*

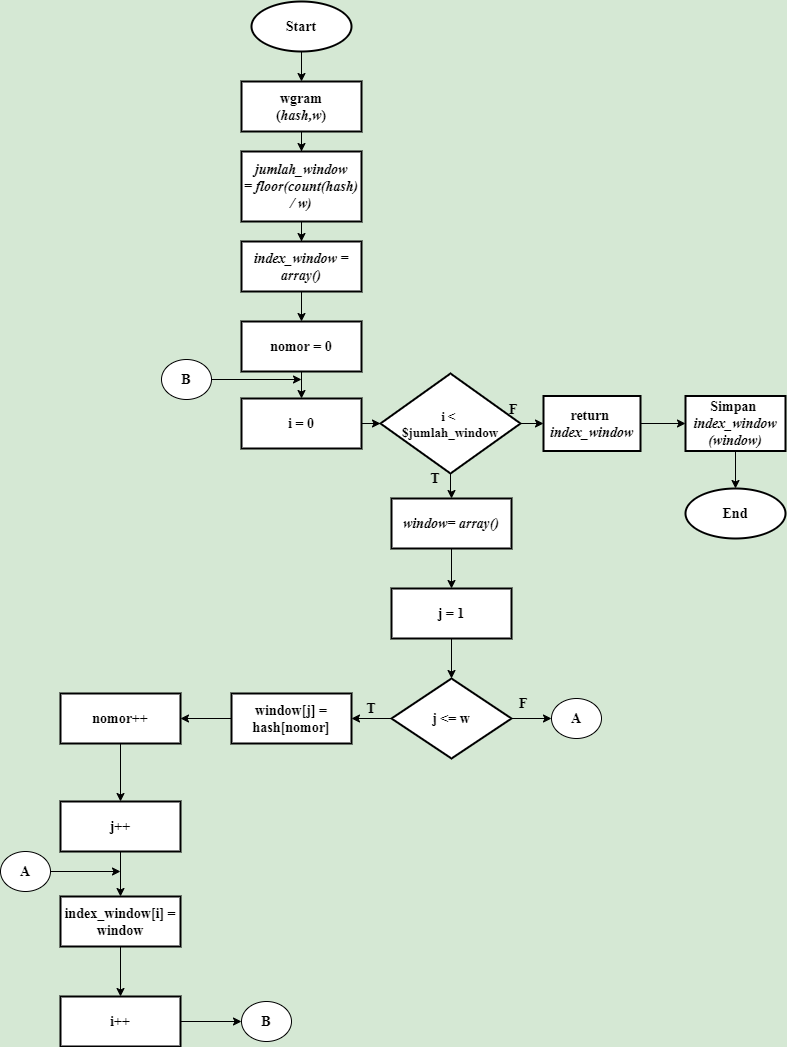
Pada *flowchart* ini, menjelaskan tahapan dari *rolling hash* yang merupakan sub proses dari algoritme *winnowing.* Proses pertama yang dilakukan adalah memanggil isi teks pada *array ngram*, yang sudah diproses ditahap sebelumnya, maka proses selanjutnya mencari nilai ASCII dari setiap huruf yang dipecah, kemudian ditampung di *array char,* setelah mendapatkan nilai ASCII dari setiap huruf yang akan diproses, maka proses selanjutnya mencari nilai *hash* dari teks tersebut, kemudian nilai *hash* tadi akan dilakukan pembentukan *window* pada tahap selanjutnya.



Gambar 14 4.12 *Flowchart Rolling Hash*

### 4.3.13. *Flowchart Window*

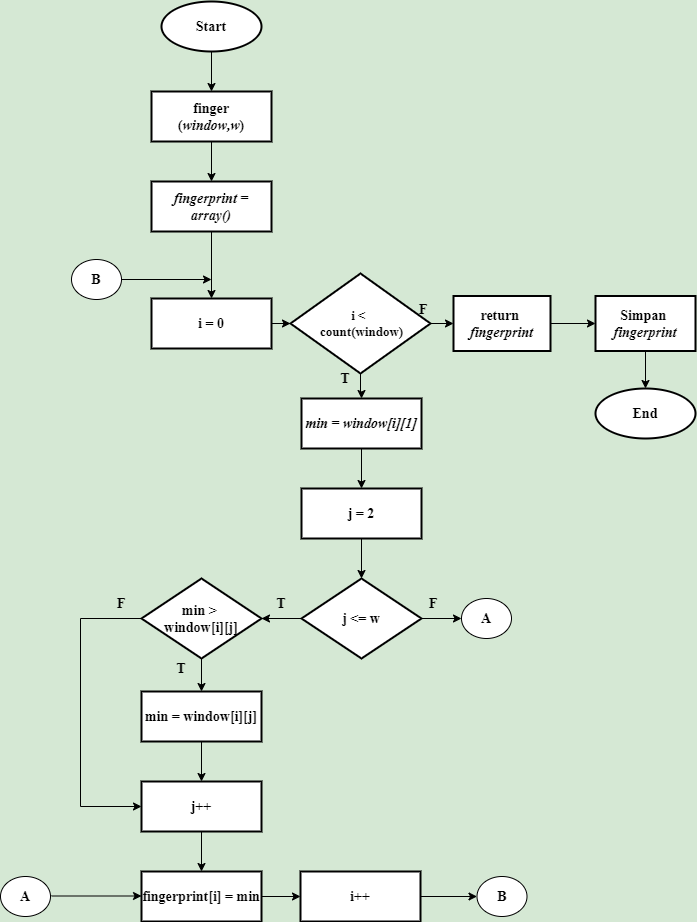
Pada *flowchart* ini, menjelaskan tahapan pembentukan *window* dari nilai *hash* yang sudah diperoleh ditahap sebelumnya, selanjutnya tentukan nilai *window* yang akan dibentuk, setelah itu nilai *hash* tadi akan dikelompokan sesuai dengan *window* yang sudah ditentukan. Kemudian hasil dari *window* akan disimpan didalam *array window,* yang akan digunakan untuk proses selanjutnya.



Gambar 15 4.13 *Flowchart Window*

### 4.3.14. *Flowchart Fingerprint*

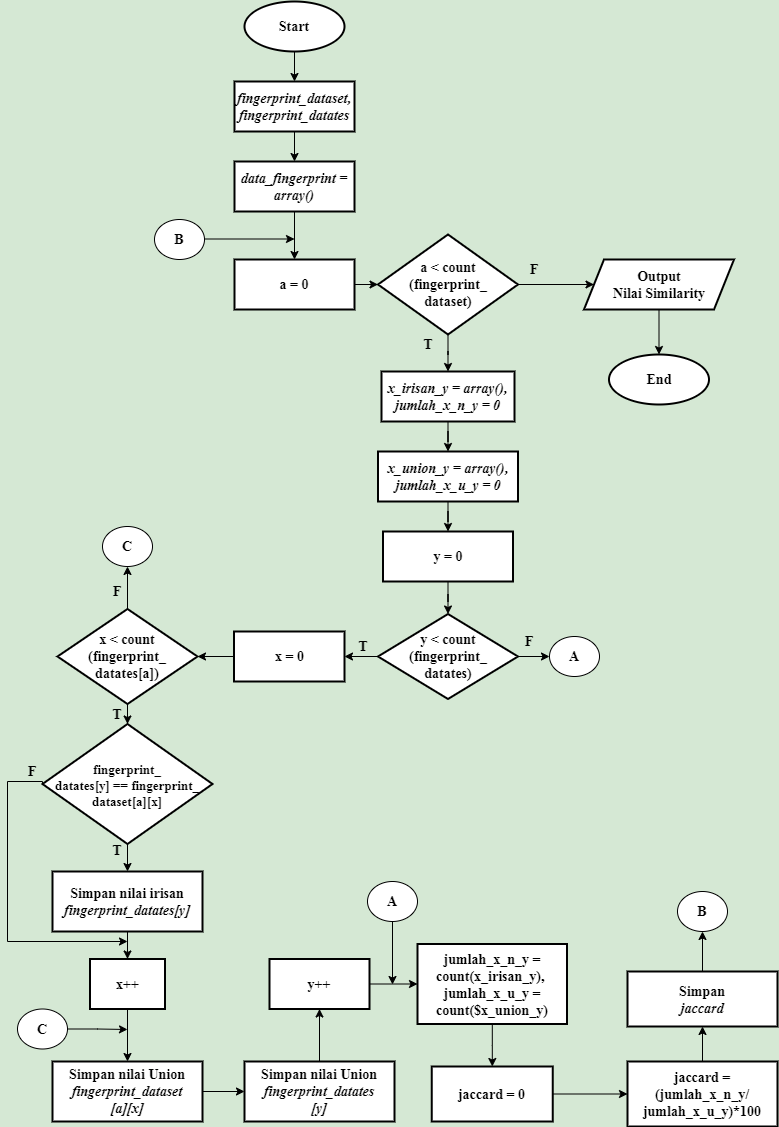
Pada *flowchart* ini, menjelaskan tahapan pencarian nilai *fingerprint* dari proses sebelumnya. Setelah dilakukan pembentukan *window,* maka tahap selanjutnya mencari nilai terkecil dari setiap *window* yang dibentuk, sehingga didapatkanlah nilai *fingerprint* dari setiap *window* yang telah dibentuk.Nilai *fingerprint* ini akan digunakan ditahap *jaccard similarity,* yaitu metode yang digunakan untuk mencari nilai *similarity* dari sebuah dokumen.



Gambar 16 4.14 *Flowchart Fingerprint*

### 4.3.15. *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 17 4.15 *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. Lakukan proses data dokumen  3. Baca dokumen dataset  4. Lakukan preprocessing dataset  5. Hasil dokumen bersih  6. Insert database  7. Input data tes  8. Lakukan preprocessing data tes  9. Lakukan n-gram  10. Lakukan algoritme winnowing  11. lakukan jaccard similarity  12. Output nilai similarity  13. if (ulangi proses)  14. kembali ke nomor 7  15. else  16. Endif  17. 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. Proses dokumen : .pdf,<1mb  3. if (dokumen dataset?)  4. insert database text\_dataset  5. Output : dataset  6. else  7. if (dokumen data tes?)  8. Simpan text\_datates  9. else  10. Endif  11. End |

### 4.4.3. Algoritme *Preprocessing Dataset*

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

**Algoritme 4.3 *Preprocessing Dataset***

|  |
| --- |
| 1. Start  2. Select Dokumen  3. data\_dokumen = []  4. if(Dokumen != null)  5. Lakukan proses Casefolding (string)  6. Lakukan proses Cleaning (noAngkaKarakter)  7. Lakukan proses Slangword (string\_noSlangword)  8. Lakukan proses Stopword (string\_nostopword)  9. Lakukan proses Menghilangkan spasi(string)  10. Output : Teks bersih  11. Simpan string ke dalam list dokumen\_simpan  12. Dokumen selanjutnya  13. Kembali ke nomor 4  14. else  15. Simpan dokumen\_simpan ke dalam database  16. Endif  17. End |

### 4.4.4. Algoritme *Preprocessing* Data Tes

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

**Algoritme 4.4 *Preprocessing* Data Tes**

|  |
| --- |
| 1. Start  2. Proses text\_datates  3. Lakukan proses Casefolding (string)  4. Lakukan proses Cleaning (noAngkaKarakter)  5. Lakukan proses Slangword (string\_noSlangword)  6. Lakukan proses Stopword (string\_nostopword)  7. Lakukan proses Menghilangkan spasi(string)  8. Simpan di variabel string  9. End |

### 4.4.5. Algoritme *Case folding*

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

**Algoritme 4.5 *Case folding***

|  |
| --- |
| 1. Start  2. Proses strtolower(text)  3. Output : string\_kecil  4. Simpan string\_kecil  5. End |

### 4.4.6. Algoritme *Cleaning*

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

**Algoritme 4.6 *Cleaning***

|  |
| --- |
| 1. Start  2. Menghilangkan karakter selain huruf a-z pada string\_kecil  3. Output : string\_noAngkaKarakter  4. Proses simpan string\_noAngkaKarakter  5. End |

### 4.4.7. Algoritme *Slang word*

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

**Algoritme 4.7 *Slang word***

|  |
| --- |
| 1. Start  2. Select data slangword dari database  3. if (slangword!=null)  4. if (kata == slangword)  5. Ubah dengan kata slangword yang terdapat dalam string\_noAngkaKarakter menjadi kata asli  6. Output string\_noSlangword  7. Simpan string\_noSlangword  8. else  9. slangword selanjutnya  10. Kembali ke nomor 3  11. Endif  12. Endif  13. End |

### 4.4.8. Algoritme *Stop word*

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

**Algoritme 4.8 *Stop word***

|  |
| --- |
| 1. Start  2. Select data stopword dari database  3. siapkan array\_stop = array()  4. if (stopword != null)  5. array\_stop[] = stopword  6. stopword selanjutnya  7. kembali ke nomor 4  8. else  9. if (string != null)  10. if(kata = array\_stop[])  11. Hapus kata array\_stop[] yang terdapat dalam string  12. Otrput : string\_noStopword  13. Simpan string\_noStopword  14. string selanjutnya  15. kembali ke nomor 9  16. Endif  17. Endif  18. End |

### 4.4.9. Algoritme Hapus Spasi

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

**Algoritme 4.9 Hapus Spasi**

|  |
| --- |
| 1. Start  2. str\_replace(string)  3. Output : string  4. Simpan string  5. End |

### 4.4.10. Algoritme *N-Gram*

Pada algoritme ini dijelaskan tentang proses tahapan utama pada algoritme *n-gram.*

**Algoritme 4.9 *N-Gram***

|  |
| --- |
| 1. Start  2. Proses Ngrams(word,n)  3. len = strlen(word)  4. ngram = array()  5. i = 0  6. if (i + n <= len)  7. string = ""  8. j=0  9. if(j < n)  10. string = word[j+i]  11. j++  12. else  13. ngram[i] = string  14. i++  15. kembali ke nomor 5  16. Endif  17. else  18. return ngram  19. Simpan ngram  20. Endif  21. End |

### 4.4.11. Algoritme *Winnowing*

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

**Algoritme 4.11 *Winnowing***

|  |
| --- |
| 1. Start  2. Proses Ngrams(ngram)  3. Lakukan proses r\_hash (hash)  4. Lakukan proses wgram (window)  5. Lakukan prose finger (fingerprint)  6. Simpan fingerprint  7. End |

### 4.4.12. Algoritme *Rolling Hash*

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

**Algoritme 4.12 *Rolling Hash***

|  |
| --- |
| 1. Start  2. Proses r\_hash(ngram,k,b)  3. char = array()  4. i = 0  5. if (i < count(ngram))  6. char[i] = str\_split(ngram[i]  7. j = 0  8. if (j < count(char[i]))  9. char[i][j] = ord(char[i][j])  10. j++  11. else  12. i++  13. kembali ke nomor 4  14. else  15. hasil hash = array()  16. y = 0  17. if (y < count(ngram)  18. hasil = 0  19. z =0  20. if (z < count(char[y])  21. hasil = hasil + char[y][z] \* pow(b,(k-(z+1)))  22. z++  23. else  24. hasil\_hash[y] = hasil  25. y++  26. kembali ke nomor 16  27. Endif  28. else  29. return hasil\_hash  30. Simpan hasil\_hash(hash)  31. Endif  32. End |

### 4.4.13. Algoritme *Window*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada algoritme *winnowing* yaitu pembentukan *window*.

**Algoritme 4.13 *Window***

|  |
| --- |
| 1. Start  2. Proses wgram (hash,w)  3. jumlah window = floor(count(hash)/w)  4. index\_window = array()  5. nomor = 0  6. i = 0  7. if (i < jumlah\_window)  8. window = array()  9. j =1  10. if (j <= w)  11. window[j] = hash[nomor]  12. nomor++  13. j++  14. else  15. index\_window[i] = window  16. i++  17. Kembali ke nomor 6  18. Endif  19. else  20. return index\_window  21. Simpan index\_window(window)  22. Endif  23. End |

### 4.4.14. Algoritme *Fingerprint*

Pada algoritme ini dijelaskan tentang proses sub tahapan pada algoritme *winnowing* yaitu pencarian nilai *fingerprint*.

**Algoritme 4.14 *Fingerprint***

|  |
| --- |
| 1. Start  2. Proses finger (window,w)  3. fingerprint = array()  4. i = 0  5. if (i < count(window))  6. min = window[i][1]  7. j = 2  8. if (j <= w)  9. if(min > window[i][j])  10. min = window[i][j]  11. else  12. j++  13. else  14. fingerprint[i] = min  15. i++  16. Kembali ke nomor 4  17. Endif  18. else  19. return fingerprint  20. Simpan fingerprint  21. Endif  22. End |

### 4.4.15. Algoritme *Jaccard Similarity*

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

**Algoritme 4.15 *Jaccard Similarity***

|  |
| --- |
| 1. Start  2. fingerprint\_dataset, fingerprint\_datates  3. data\_fingerprint = array()  4. a = 0  5. if (a < count(fingerprint\_dataset)  6. x\_irisan\_y = array(), jumlah\_x\_n\_y = 0  7. x\_union\_y = array(), jumlah x\_u\_y = 0  8. y = 0  9. if (y < count(fingerprint\_datates)  10. x = 0  11. if (x < count(fingerprint\_datates[a])  12. if (fingerprint\_datates[y] == fingerprint\_dataset[a][x]  13. Simpan nilai irisan fingerprint\_datates[y]  14. else  15. x++  16. else  17. Simpan nilai union fingerprint\_dataset[a][x]  18. Simpan nilai fingerprint\_dataset[y]  19. y++  20. else  21. jumlah\_x\_n\_y = count(x\_irisan\_y), jumlah\_x\_u\_y = count(x\_union\_y)  22. jaccard = 0  23. jaccard = (jumlah\_x\_n\_y/jumlah\_x\_u\_y)\*100  24. Simpan jaccard  25. Kembali ke nomor 4  26. Endif  27. else  28. Output : Nilai Similarity  29. Endif  30. 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. Contoh Pengujian

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 4.13.

Tabel 15 4.13 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%** |
|

1. 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 4.14.

Tabel 16 4.14 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%** |
|

1. 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 4.15.

Tabel 17 4.15 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 4.16.

Tabel 18 4.16 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 4.17.

Tabel 19 4.17 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%** |
|

### 4.5.2. Penjelasan Hasil Pengujian

Hasil uji coba dari pencarian nilai *similarity* menggunakan metode *n*-*gram* dan *Jaccard* *Similarity* terhadap algoritme *winnowing* telah ditampilkan pada tabel 4.13 dengan menggunakan *k-gram* dan *w-gram* yang sama. Tabel 4.13 menjelaskan pengaruh dari penggunaan *k-gram* dan *w-gram* terhadap hasil *similarity*. Hasil pengujian terhadap 1 data tes dengan 13 dokumen *dataset* memiliki derajat kesamaan yang berbeda-beda. Semakin kecil nilai *k-gram* dan *w-gram* maka derajat kesamaan atau *similarity* terhadap suatu dokumen mempunyai derajat kesamaan yang tinggi, sedangkan jika semakin besar nilai *k-gram* dan *w-gram* maka derajat kesamaan suatu dokumen itu memiliki derajat yang rendah, jika nilai *k-gram* dan *w-gram* sama. Pada tabel 4.13 membuktikan *k-gram* dan *w-gram* dengan nilai 2 menghasilkan *similarity* 51.40%, sedangkan *k-gram* dan *w-gram* dengan nilai 5 menghasilkan *similarity* yang rendah yaitu 5.26%.

Penginputan nilai yang berbeda pada nilai *k-gram* dan *w-gram* dapat menghasilkan nilai *similarity* yang tinggi. Semakin kecil nilai *k-gram* dan *w-gram* maka akan semakin sering potongan suku kata akan dicocokan dan sering ditemukan nilai yang sama, jika sebaliknya maka akan semakin jarang data itu dicocokkan atau di temukan nilai yang sama. Pada tabel 4.14 dengan nilai *k-gram* = 2 dan *w-gram* = 5 menghasilkan nilai *similarity* sebesar 45.65%. Pada tabel 4.17 dengan pengujian *dataset* dan data tes yang sama menghasilakan nilai *similarity* sebesar 100%. Dengan demikian, metode *Jaccard Similarity* memiliki kinerja yang baik untuk digunakan dalam pendeteksian nilai *similarity* terhadap suatu dokumen.

## 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*

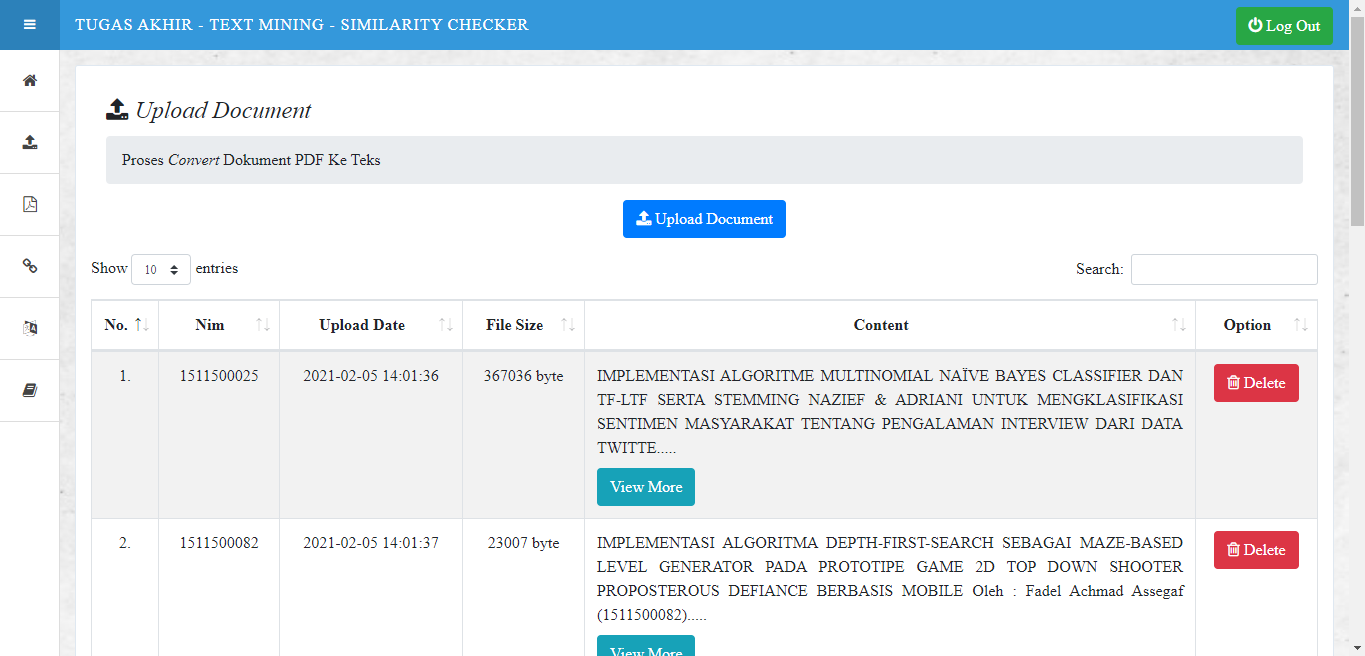
Tampilan layar *home* adalah tampilan saat pertama kali masuk ke dalam sistem. Di*menu home* ini menjelaskan tentang sistem yang dibuat, jumlah *dataset, data slang word, data stop word,* serta terdapat judul dari penelitian yang dibuat*.* Sistem yang dibuat memiliki beberapa sub, diantaranya *document data, document dataset, slang word, stop word,* dan *similarity check.* Tampilan layar *home* bisa dilihat pada gambar 4.16.



Gambar 18 4.16 Tampilan Layar *Home*

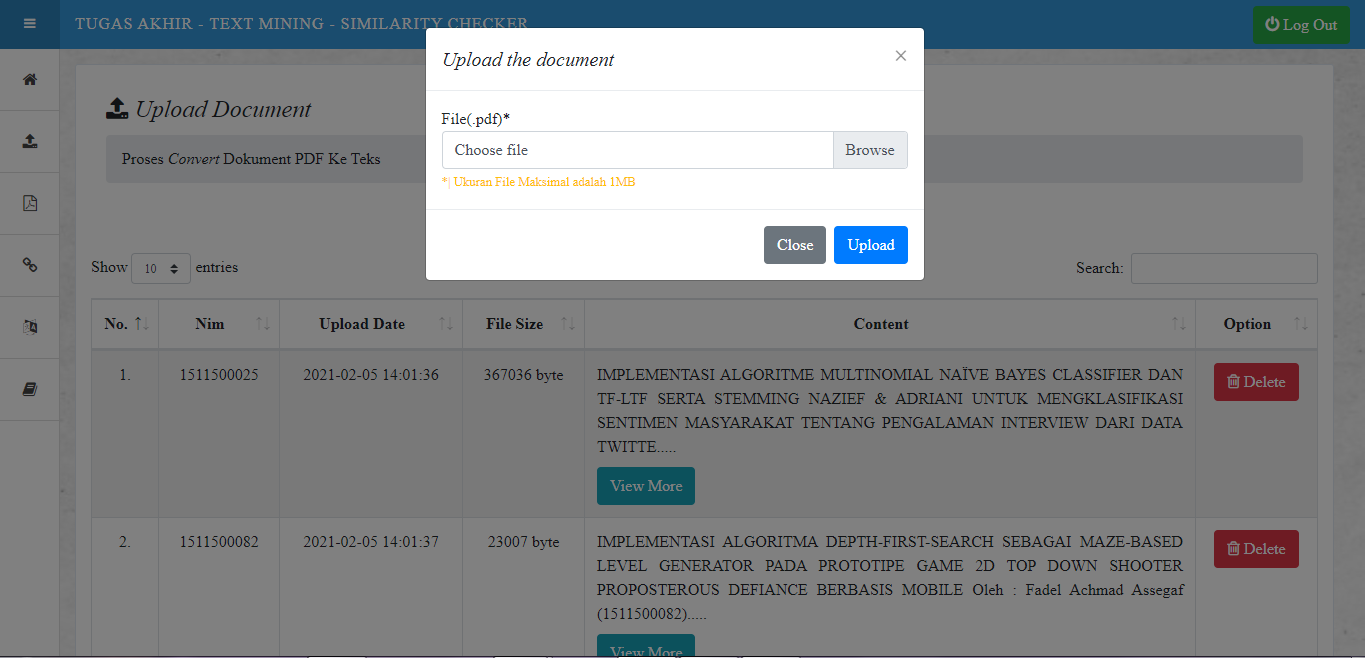
### 4.6.2. Tampilan Layar *Document Data*

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 4.17.



Gambar 19 4.17 Tampilan Layar *Document Data*

Pada Gambar 4.18 merupakan tampilan layar untuk proses *upload* dokumen, pada proses ini *user* akan diarahkan ke *windows explorer* untuk memilih dokumen abstrak yang berbentuk *.pdf,* serta berukuran <1mb. Berikut tampilanya.



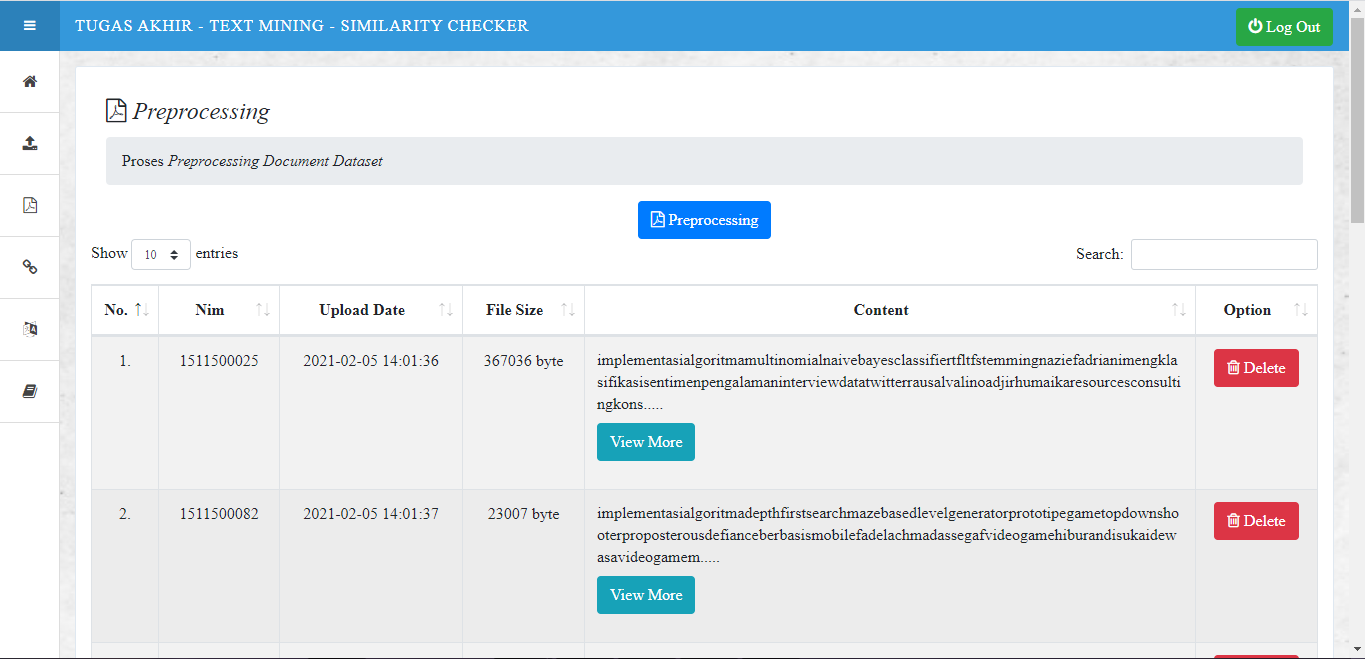
Gambar 20 4.18 Tampilan Layar *Document Data* (Lanjutan)

### 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*

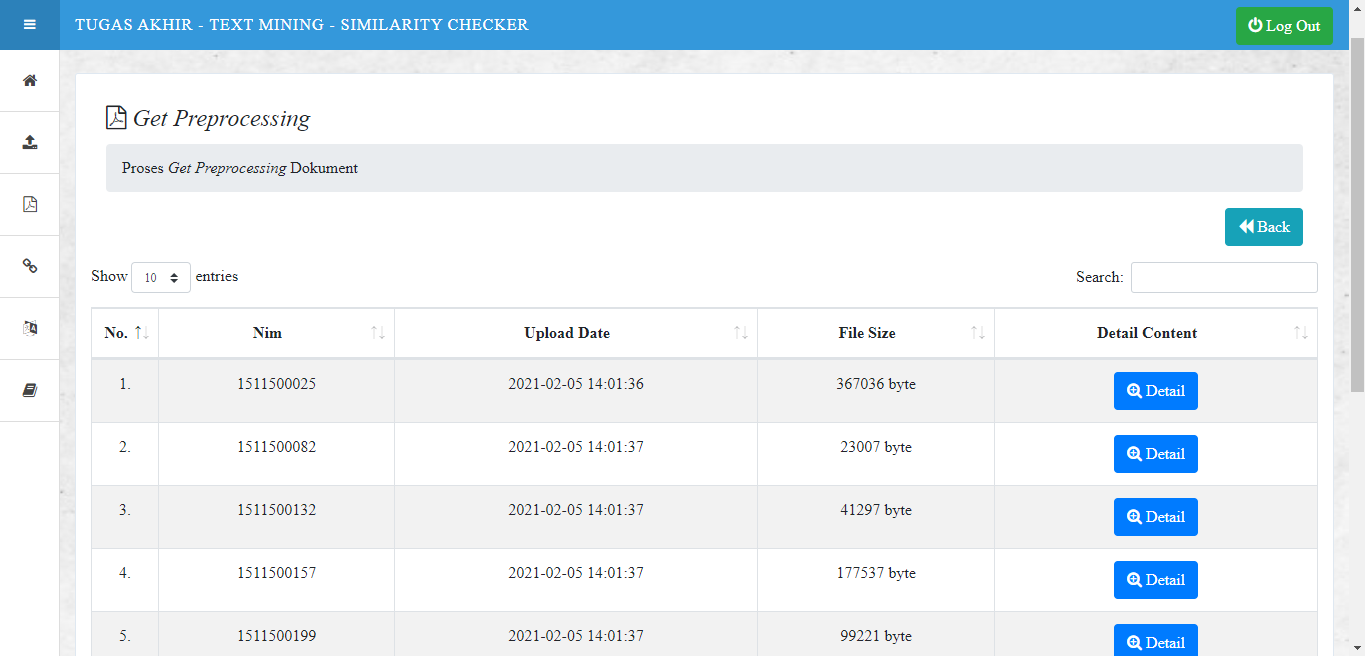
Pada Gambar 4.19 menjelaskan proses *preprocessing,* dari proses *case folding, cleaning*, mengganti *slang word,menghapus stop word,* dan penghapusan spasi antar kata.



Gambar 21 4.19 Tampilan Layar *Preprocessing*

1. Tampilan Layar Detail *Preprocessing*

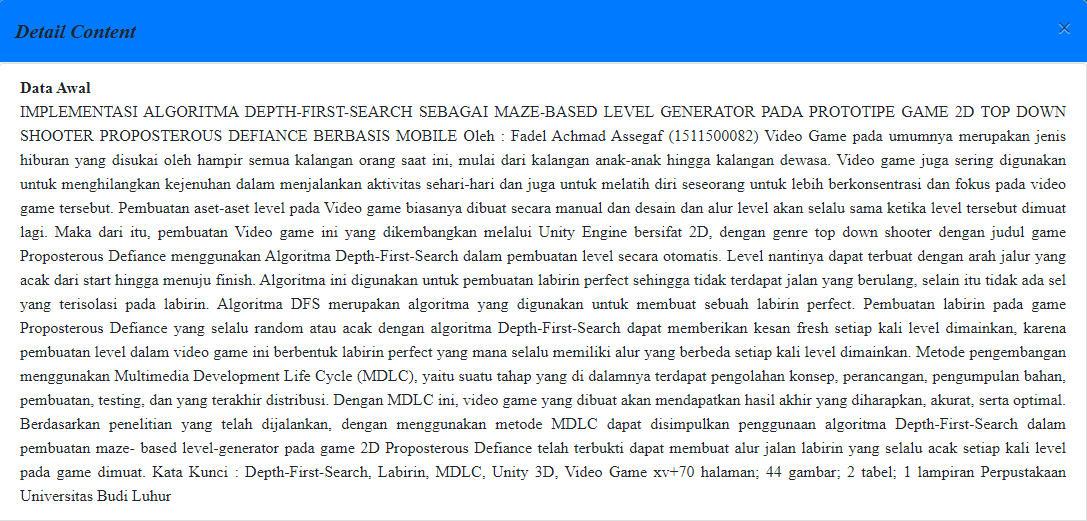
Pada Gambar 4.20 merupakan sub *menu* dari *document dataset.* Tampilan layar ini menjelaskan *detail* proses *preprocessing,* dan bisa dilihat lebih detail lagi pada gambar selanjutnya.



Gambar 22 4.20 Tampilan Layar Detail *Preprocessing*

* Tampilan Layar Data Awal

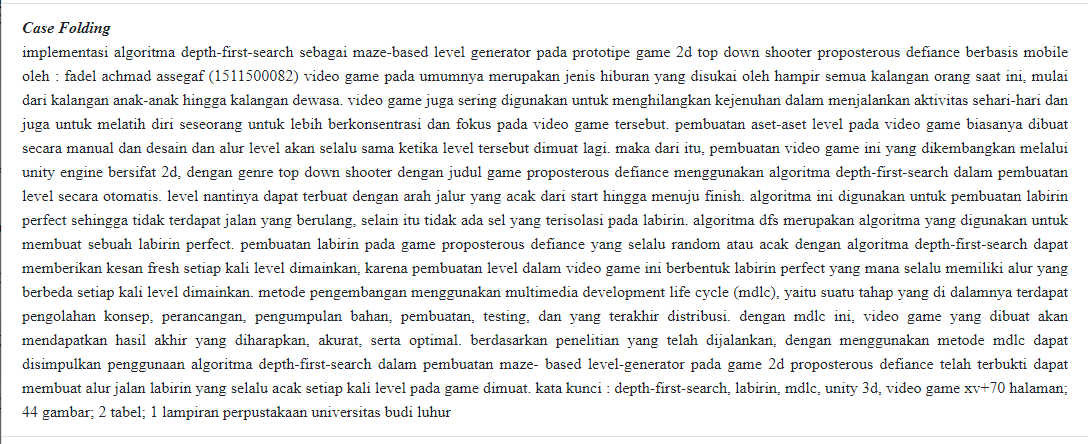
Pada Gambar 4.21 merupakan bentuk data awal dari teks dokumen abstrak, yang akan diproses di tahap *preprocessing.*



Gambar 23 4.21 Tampilan Layar Data Awal

* Tampilan Layar *Case Folding*

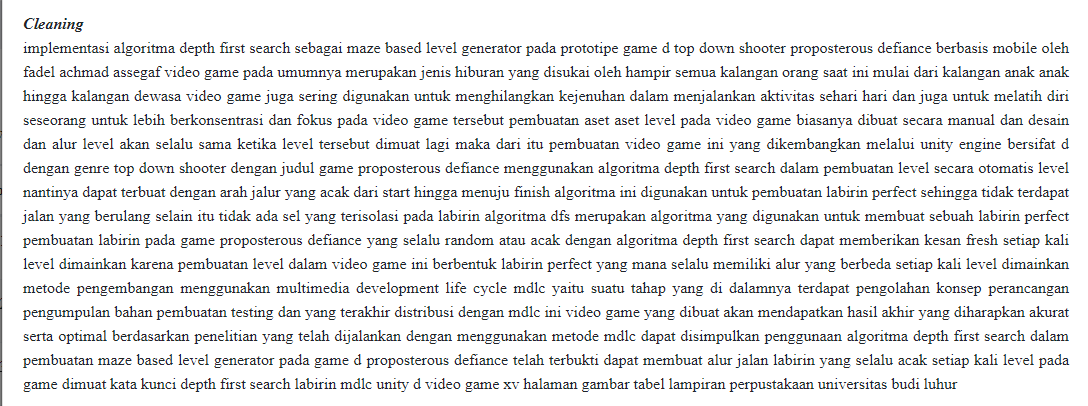
Pada Gambar 4.22 merupakan tampilan layar dari proses *case folding,* yaitu proses merubah teks menjadi huruf kecil.



Gambar 24 4.22 Tampilan Layar *Case Folding*

* Tampilan Layar *Cleaning*

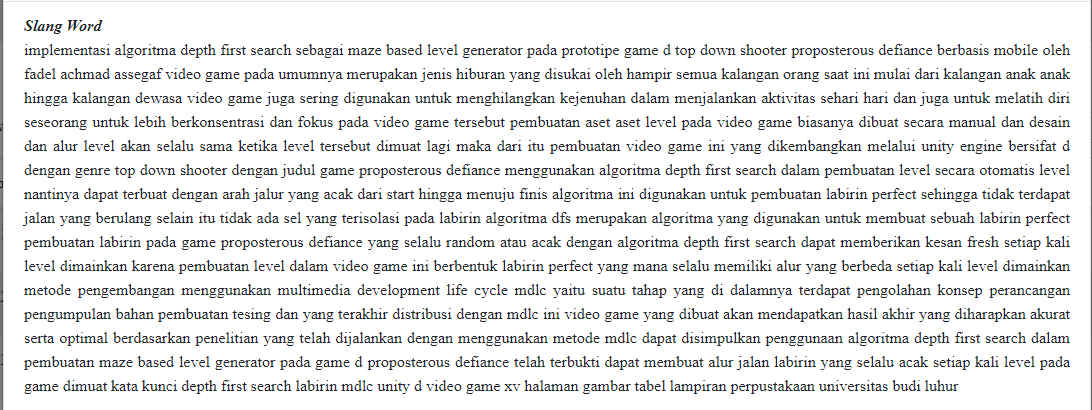
Pada Gambar 4.23 merupakan tampilan layar dari proses *cleaning,* yaitu proses menghapus karakter selain huruf a-z.



Gambar 25 4.23 Tampilan Layar *Cleaning*

* Tampilan Layar *Slang Word*

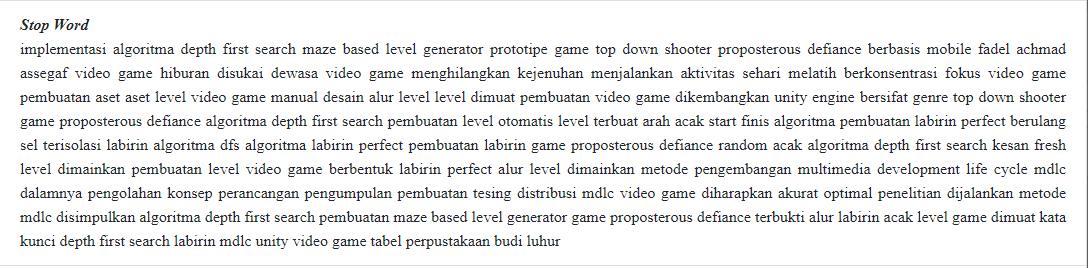
Pada Gambar 4.24 merupakan tampilan layar dari proses *slang word,* yaitu proses mengganti kata yang ada di data *slang word* menjadi kata asli.



Gambar 26 4.24 Tampilan Layar *Slang Word*

* Tampilan Layar *Stop Word*

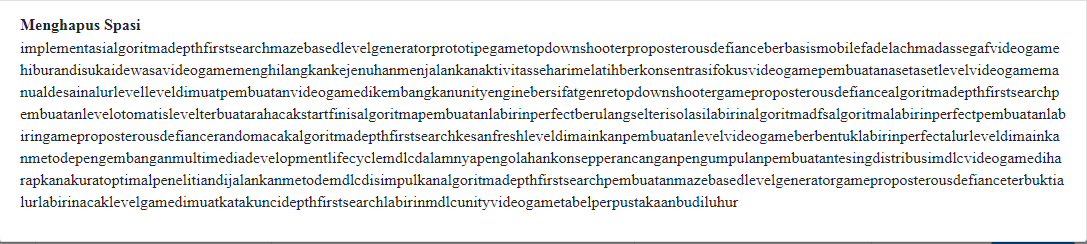
Pada Gambar 4.25 merupakan tampilan layar dari proses *stop word,* yaitu proses menghapus kata yang ada di data *stop word.*



Gambar 27 4.25 Tampilan Layar *Stop Word*

* Tampilan Layar Menghapus Spasi

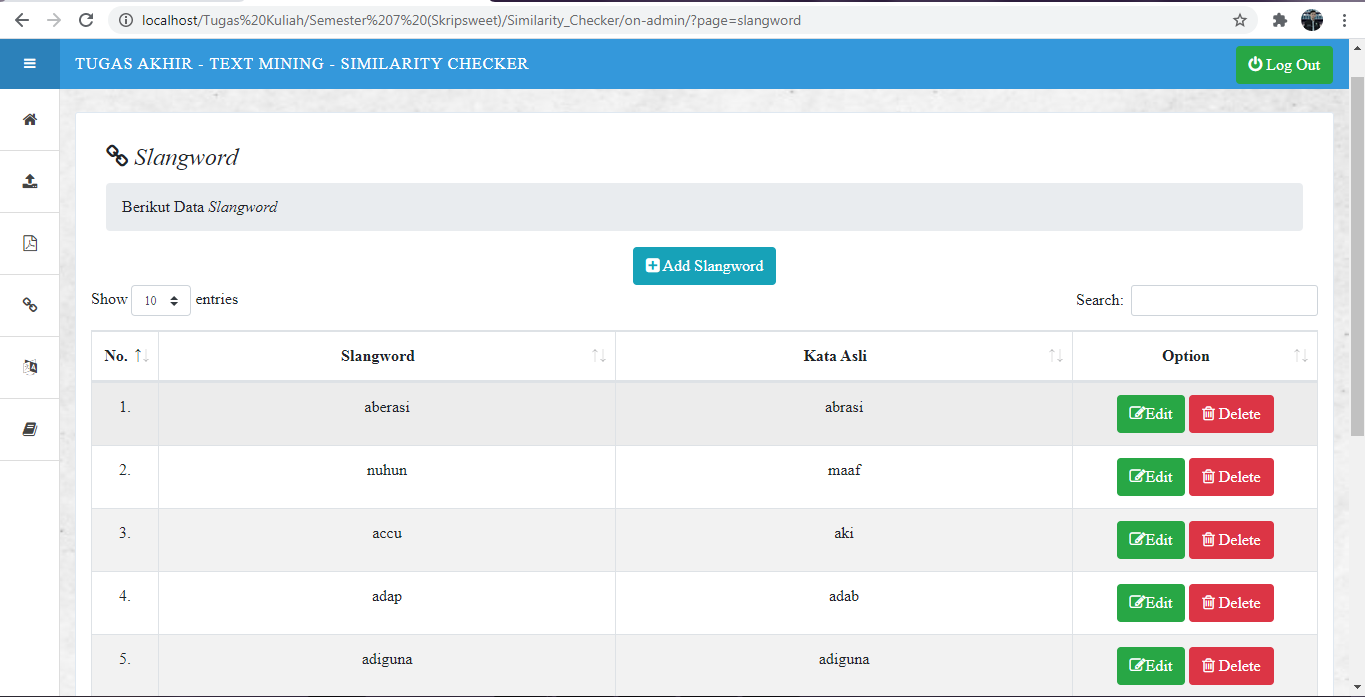
Pada Gambar 4.26 merupakan tampilan layar dari proses menghapus spasi*,* yaitu proses menghapus spasi antar kata di setiap teks.



Gambar 28 4.26 Tampilan Layar Menghapus Spasi

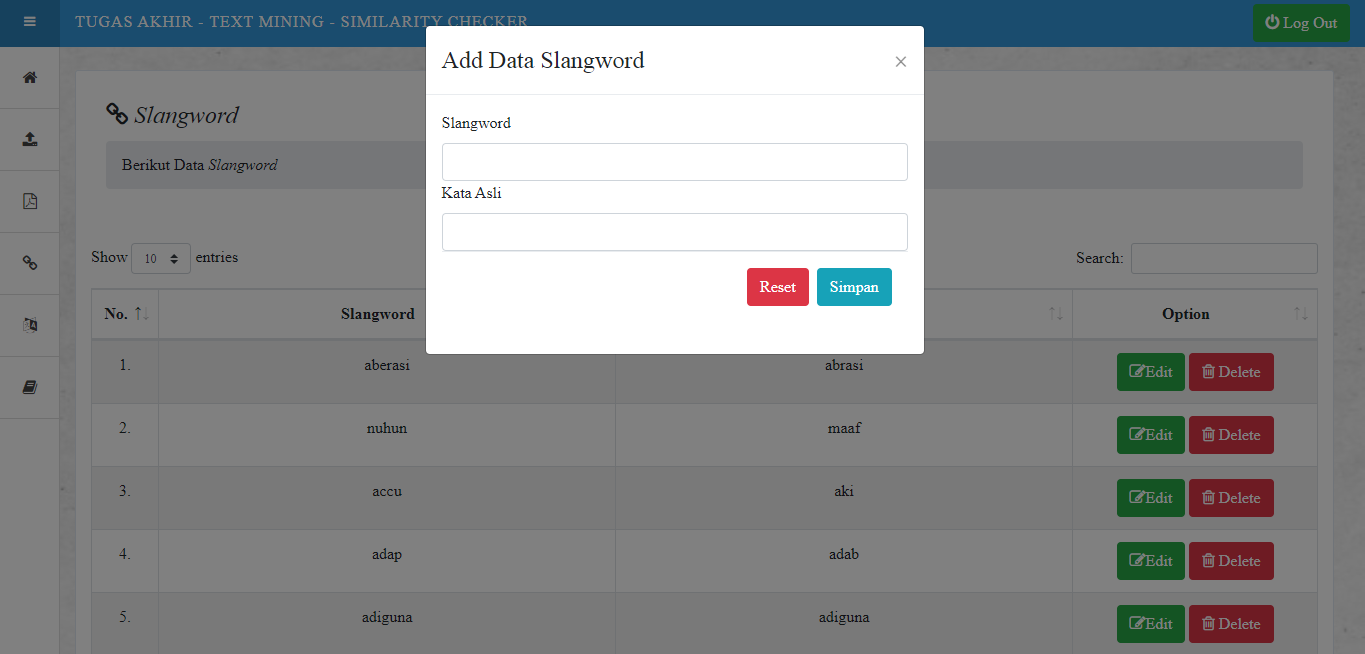
### 4.6.4. Tampilan Layar *Slang word*

Tampilan layar pada Gambar 4.27, Gambar 4.28, dan Gambar 4.29 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.*



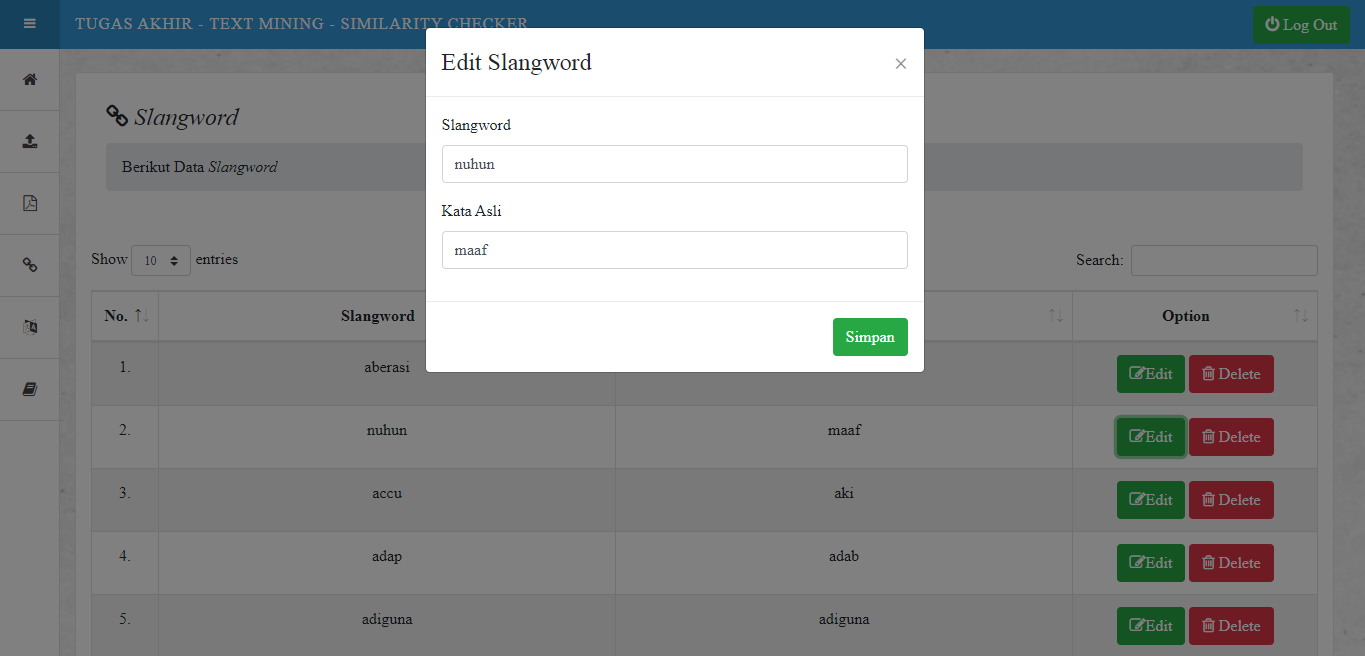
Gambar 29 4.27 Tampilan Layar *Menu Slang word*

Pada Gambar 4.28 merupakan tampilan layar untuk proses menambah data *slang word,* pada proses ini setelah *user* melakukan penambahan data *slang word* maka data tersebut akan disimpan kedalam *database* pada tabel *slang word.*



Gambar 30 4.28 Tampilan Layar *Menu Add Slang word*

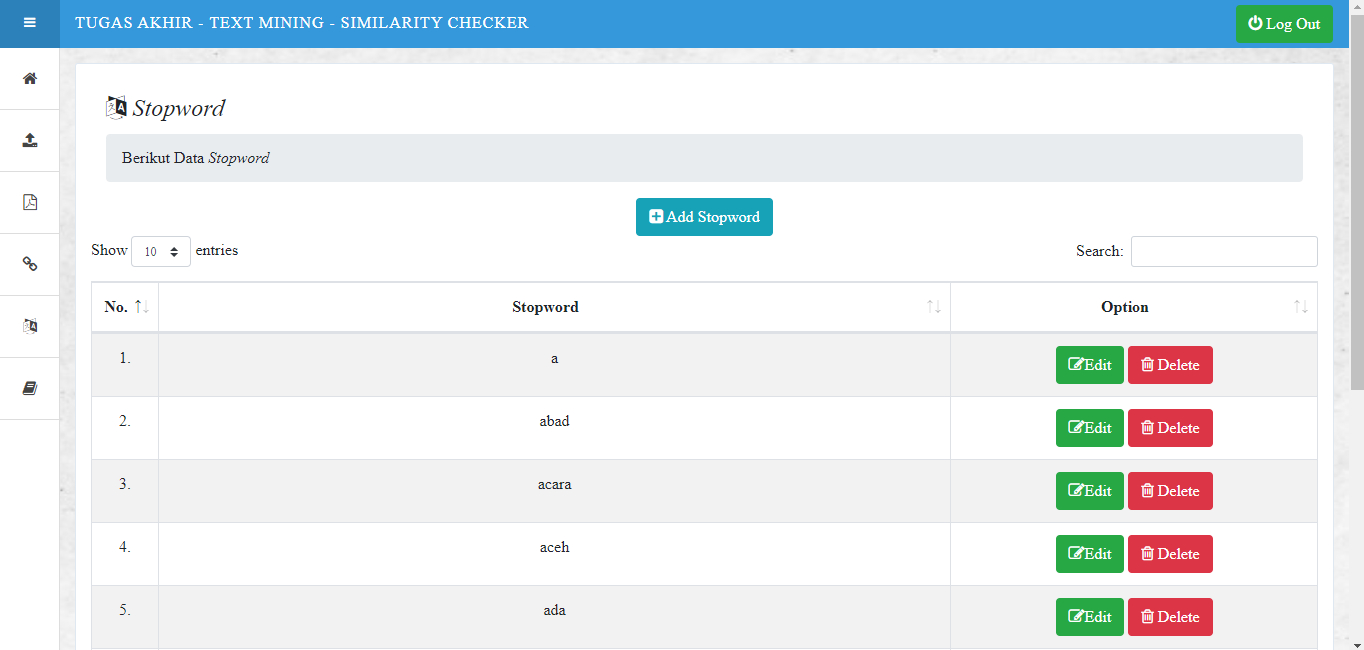
Pada Gambar 4.29 merupakan tampilan layar untuk proses mengedit data *slang word,* pada proses ini setelah *user* melakukan pengeditan data *slang word* maka data tersebut akan di*update* kedalam *database* pada tabel *slang word.*



Gambar 31 4.29 Tampilan Layar *Menu Edit Slang word*

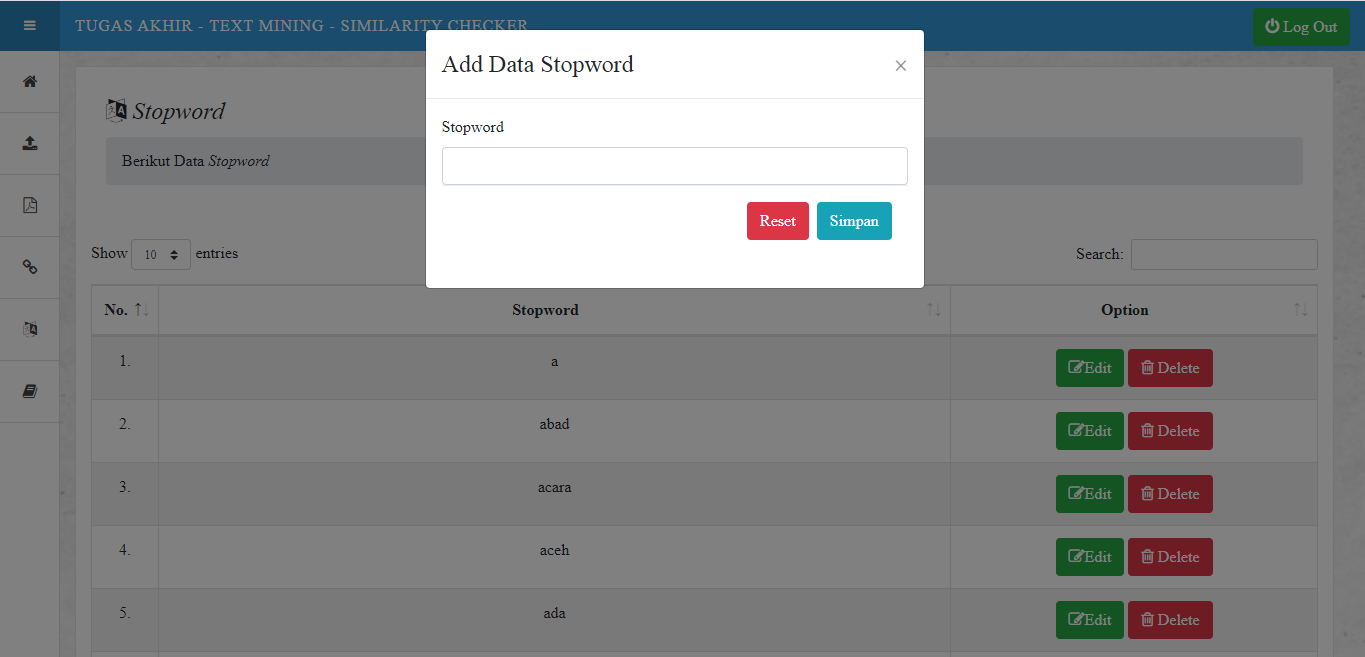
### 4.6.5. Tampilan Layar *Stop word*

Tampilan layar pada Gambar 4.30, Gambar 4.31, dan Gambar 4.32 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.*



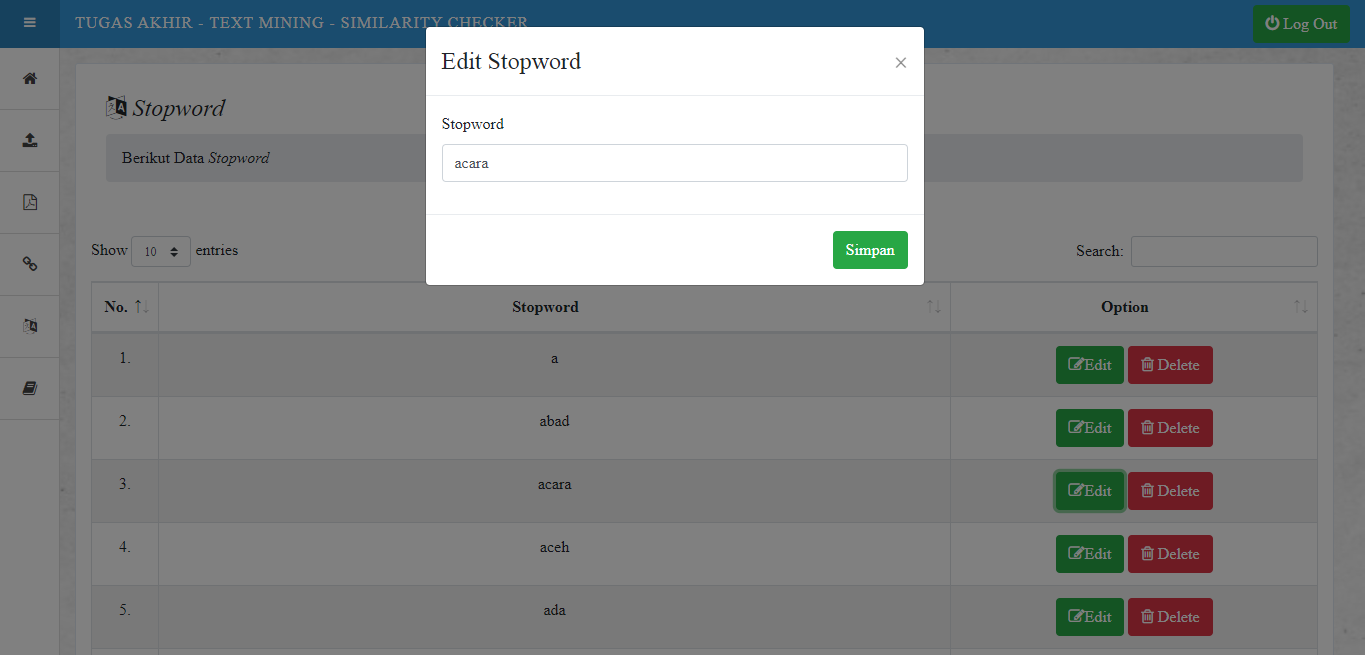
Gambar 32 4.30 Tampilan Layar Menu Stop word

Pada Gambar 4.31 merupakan tampilan layar untuk proses menambah data *stop word,* pada proses ini setelah *user* melakukan penambahan data *stop word* maka data tersebut akan disimpan kedalam *database* pada tabel *stop word.*



Gambar 33 4.32 Tampilan Layar *Menu Add Stop word*

Pada Gambar 4.33 merupakan tampilan layar untuk proses mengedit data *stop word,* pada proses ini setelah *user* melakukan pengeditan data *stop word* maka data tersebut akan di*update* kedalam *database* pada tabel *stop word.*



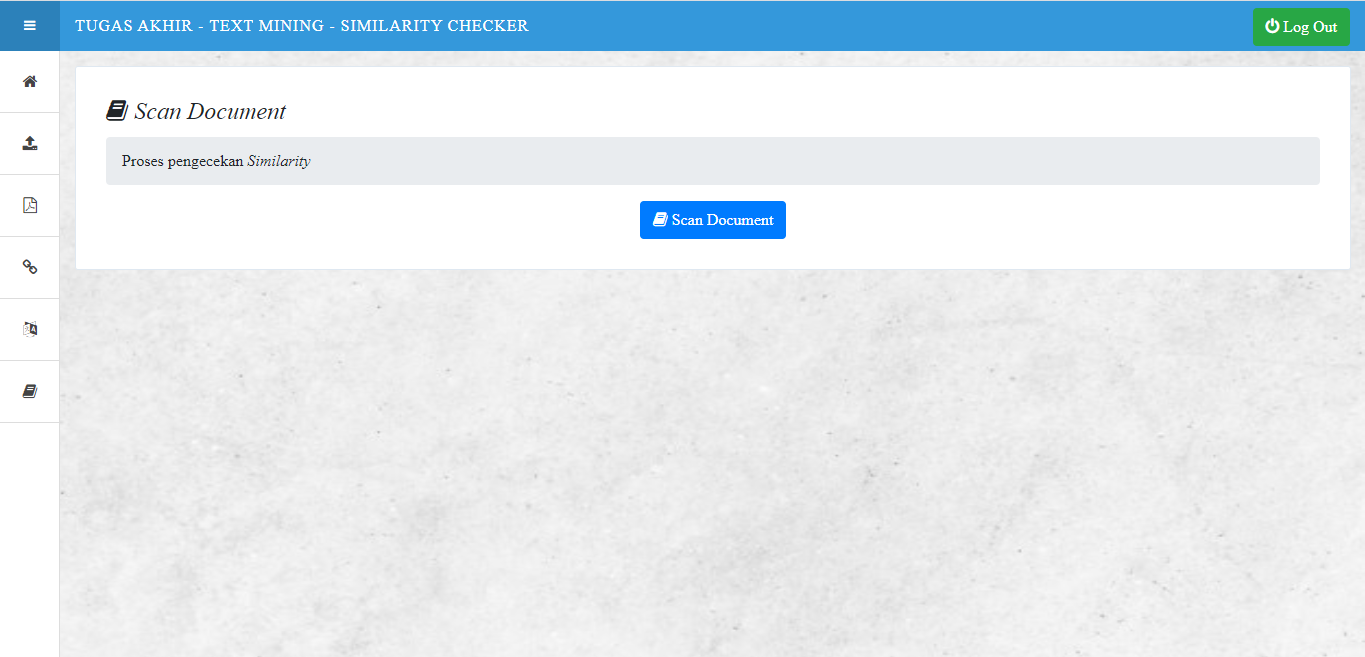
Gambar 34 4.33 Tampilan Layar *Menu* *Edit Stop word*

### 4.6.6. Tampilan Layar *Similarity Check*

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

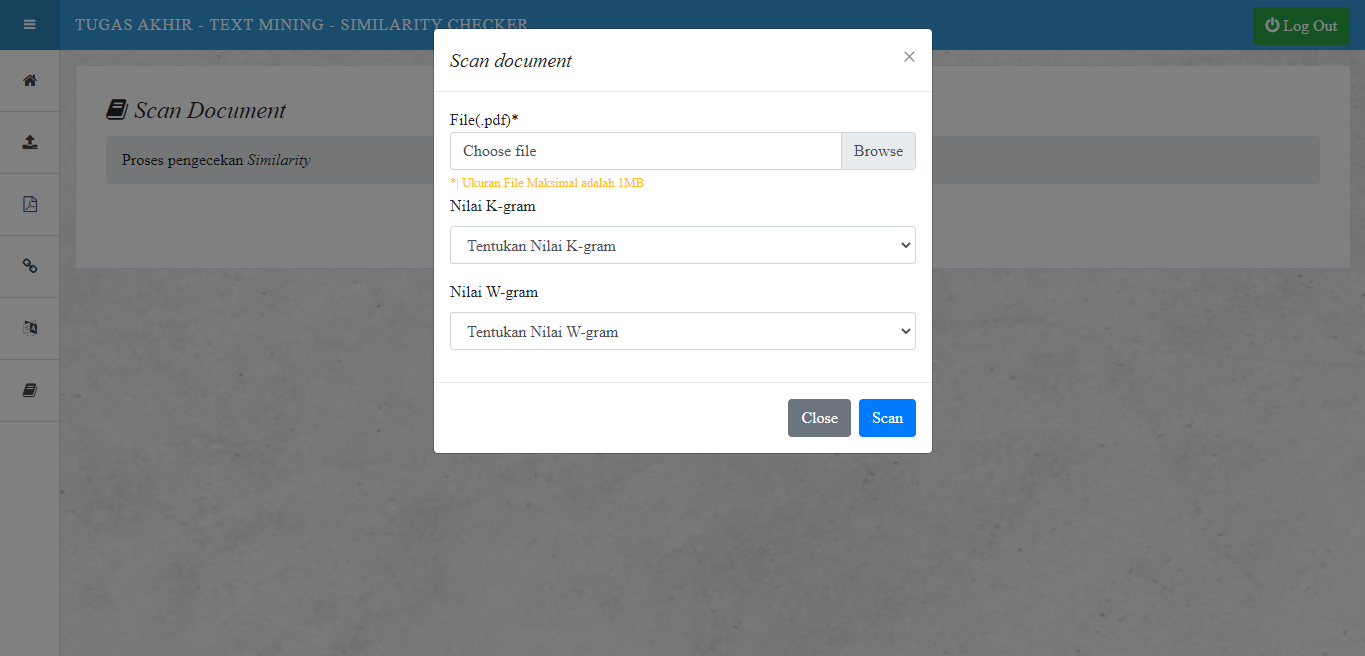
1. Tampilan Layar *Menu Scan Document*

Tampilan layar pada Gambar 4.34 merupakan proses *scan* dokumen abstrak untuk melihat nilai *similarity* dari dokumen tersebut.



Gambar 35 4.34 Tampilan Layar *Menu Scan Document*

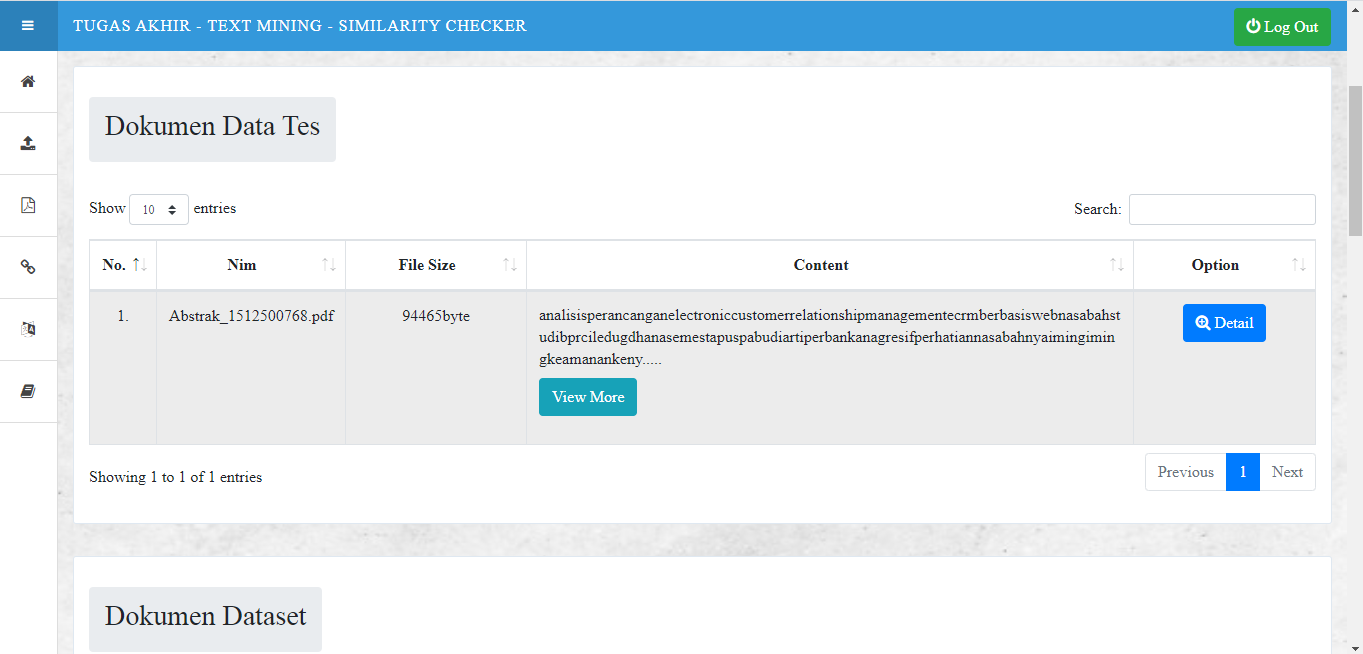
Pada Gambar 4.35 merupakan tampilan layar untuk proses *scan* dokumen data tes, pada tahap ini *user* harus menentukan nilai *k-gram* dan nilai *w-gram* untuk proses pencarian nilai *similarity* dari dokumen yang telah di *scan.*



Gambar 36 4.35 Tampilan Layar *Menu Scan Document* (Lanjutan)

1. Tampilan Layar *Document* Data Tes

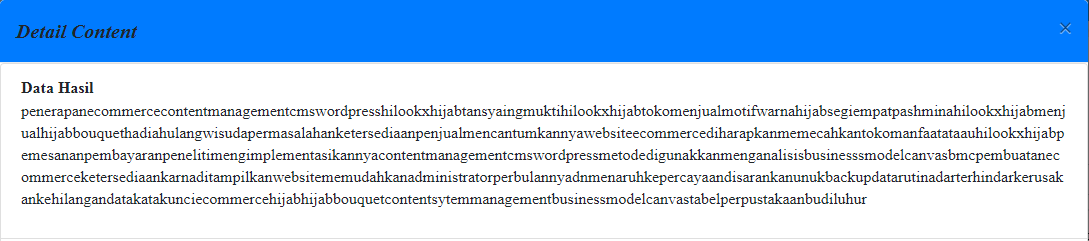
Tampilan layar pada Gambar 4.36 merupakan dokumen abstrak data tes yang di*scan* pada proses sebelumnya. Pada gambar ini bisa dilihat teks bersih, potongan *n-gram,* nilai *rolling hash,* pembentukan *window,* serta nilai *fingerprint* dari dokumen tersebut.



Gambar 37 4.36 Tampilan Layar *Document* Data Tes

* Tampilan Layar Data Hasil

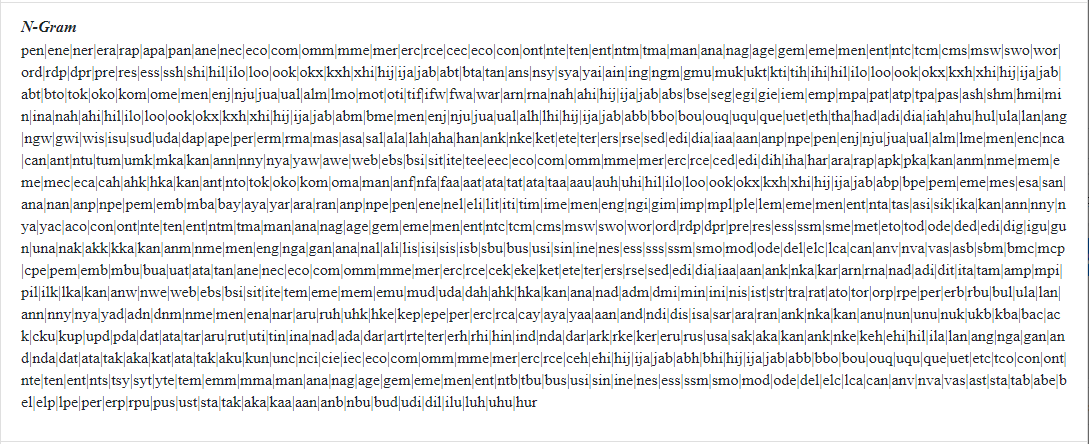
Pada Gambar 4.37 merupakan tampilan layar dari data tes yang sudah melalu tahap *preprocessing,* data ini akan diproses untuk pembentukan *n-gram, rolling hash, window,* dan pencarian *fingerprint.*



Gambar 38 4.37 Tampilan Layar Data Hasil

* Tampilan Layar *N-Gram*

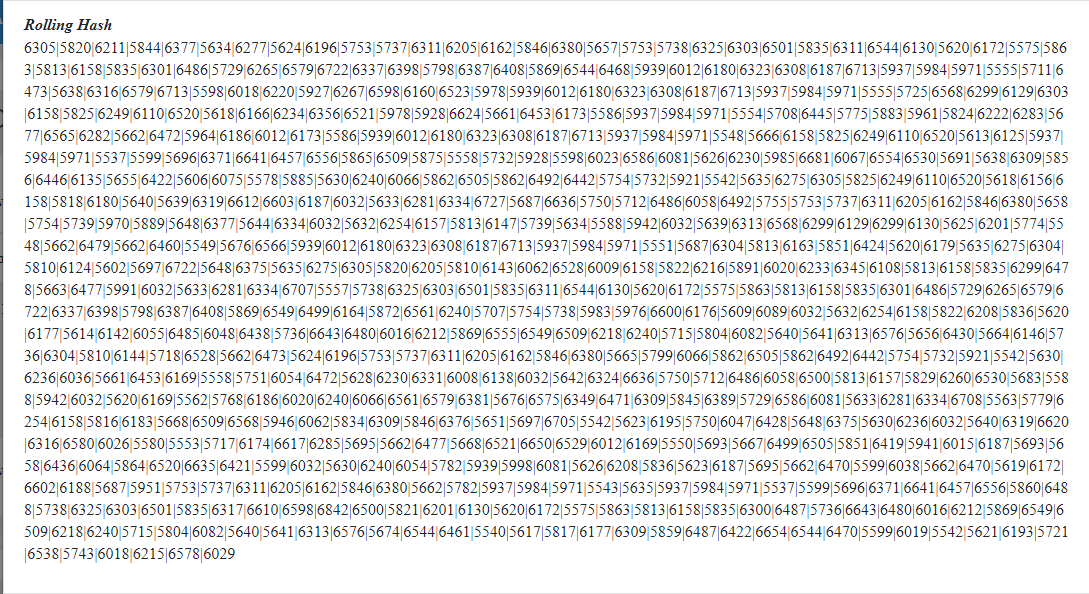
Pada Gambar 4.38 merupakan tampilan layar dari *n-gram,* teks dari data hasil dipotong berdasarkan jumlah nilai *k-gram* yang telah ditentukan pada saat proses *scan* dokumen.



Gambar 39 4.38 Tampilan Layar *N-Gram*

* Tampilan Layar *Rolling Hash*

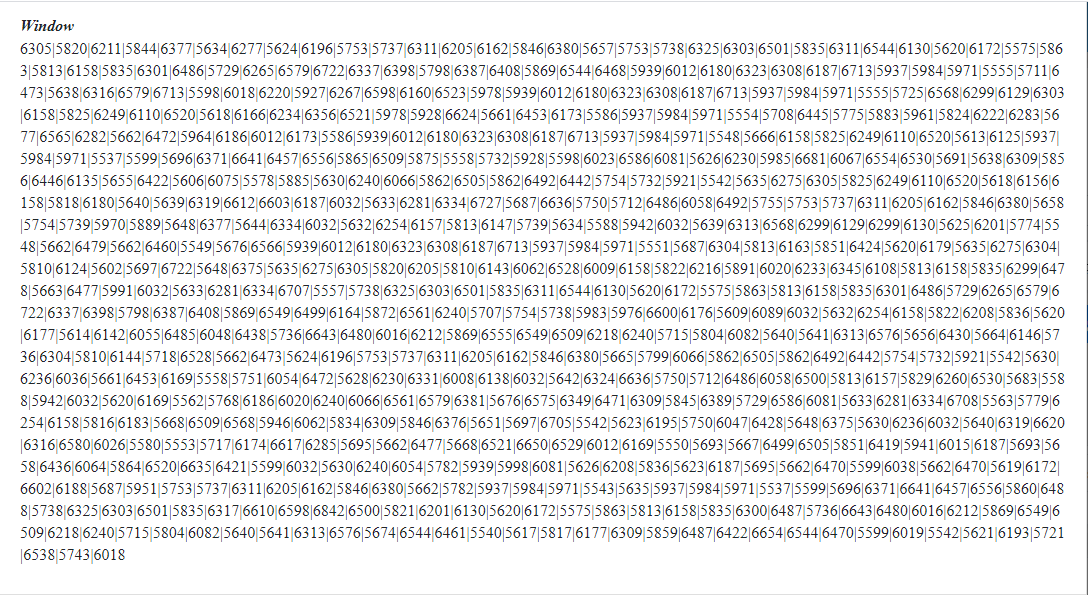
Pada Gambar 4.39 merupakan tampilan layar dari *rolling hash,* pada gambar ini bisa dilihat nilai *hash*  dari ­*n-gram* yang sudah dibentuk.



Gambar 40 4.39 Tampilan Layar *Rolling Hash*

* Tampilan Layar *Window*

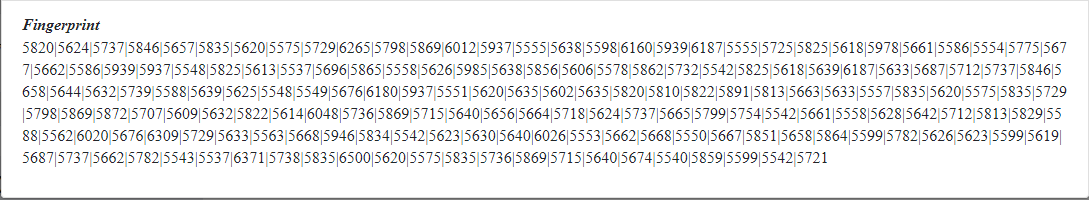
Pada Gambar 4.40 merupakan tampilan layar dari *window,* pada gambar ini nilai *hash* yang sudah didapat pada proses sebelumnya akan dilakukan pembentukan *window* berdasarkan nilai yang telah ditentukan pada saat proses *scan* dokumen.



Gambar 41 4.40 Tampilan Layar *Window*

* Tampilan Layar *Fingerpring*

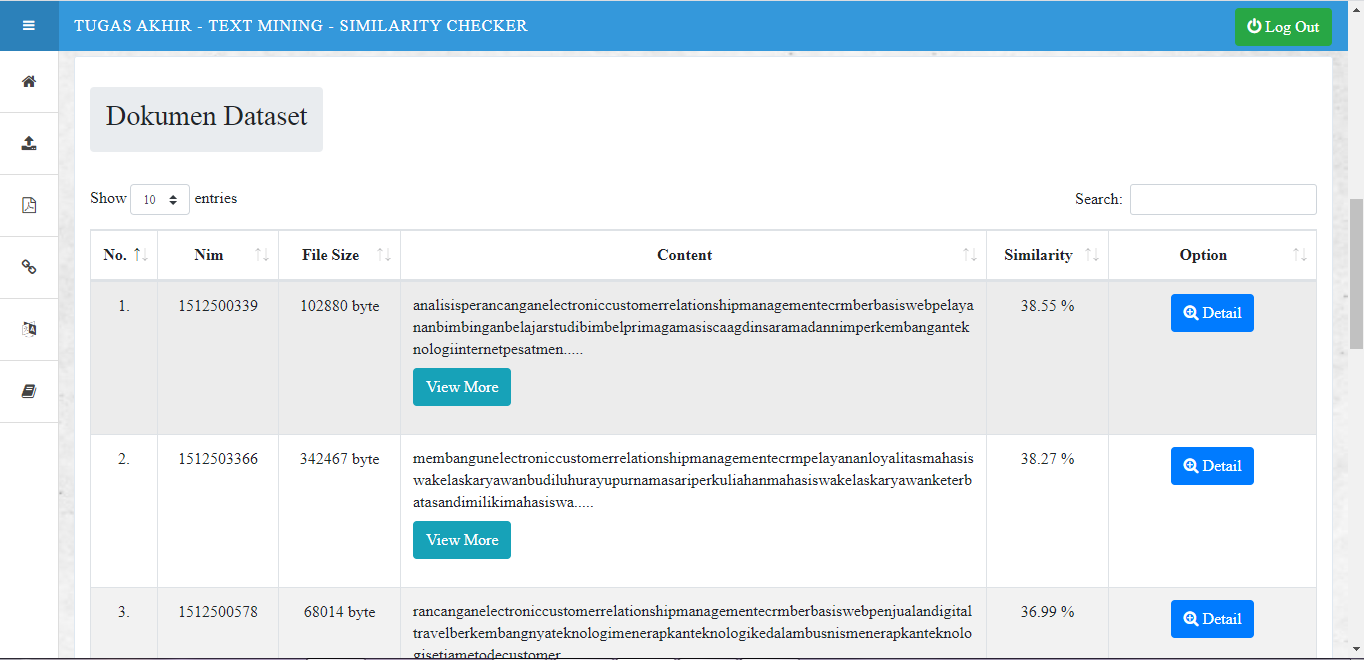
Pada Gambar 4.41 merupakan tampilan layar dari *fingerprint,* pada gambar ini dapat dilihat hasil dari proses pencarian nilai *finger* pada tahap sebelumnya, yaitu pencarian nilai terkecil pada nilai *window* yang sudah dibentuk pada tahap sebelumnya.



Gambar 42 4.41 Tampilan Layar *Fingerprint*

1. Tampilan Layar Nilai *Similarity*

Tampilan layar pada Gambar 4.42 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 43 4.42 Tampilan Layar Nilai *Similarity*