DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN *DISTANCE*

SKRIPSI

Konsentrasi Komputasi Cerdas dan Visualisasi

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Hadi Santoso

NIM. 0910960038

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN *DISTANCE*

SKRIPSI

Konsentrasi Komputasi Cerdas dan Visualisasi

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

Hadi Santoso

NIM. 0910960038

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN DISTANCE

SKRIPSI

Konsentrasi Komputasi Cerdas dan Visualisasi

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh :

HADI SANTOSO

NIM. 0910960038

Skripsi ini telah disetujui oleh dosen pembimbing

pada tanggal 4 April 2014

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Lailil Muflikhah, S.Kom., M.Sc	**Indriati, ST.,M.Kom**
NIP. 19741113 200501 2 001	**NIK. 831013 06 1 2 0035**

LEMBAR PENGESAHAN

DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN DISTANCE

SKRIPSI

Konsentrasi Komputasi Cerdas dan Visualisasi

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

HADI SANTOSO

NIM. 0910960038

Skripsi ini telah diuji dan dinyatakan lulus pada

Tanggal 11 April 2014

Penguji I,	Penguji II,
Imam Cholissodin, S.Si.,M.Kom	**Budi Darma Setiawan, S.Kom, M.Cs**
NIK. 85071916110422	**NIK. 8410150611009**

Penguji III,

Candra Dewi, S.Kom, M.Sc
NIP. 19771114 200312 2 00

Mengetahui,

Ketua Program Studi Teknik Informatika / Ilmu Komputer

Drs. Marji., M.T.
NIP. 19670801 199203 1 001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Hadi Santoso
NIM	: 0910960038
Program Studi	: Informatika / Ilmu Komputer
Jurusan	: Ilmu Komputer
Fakultas	: Program Teknologi Informasi dan Ilmu Komputer
Judul Skipsi	: DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN DISTANCE

Dengan ini menyatakan bahwa:

1. Isi dan skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang tercantum di isi dan tertulis di daftar pustaka dalam skripsi ini.
2. Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya bersedia mmenanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran dan penuh tanggung jawab dan digunakan sebagaimana mestinya.

Malang, 4 April 2014

Mahasiswa,

Hadi Santoso

NIM. 0910960038

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, atas segala rahmat dan karuniaNya penulis dapat menyelesaikan skripsi dengan judul "Deteksi Plagiarisme Dokumen Teks Menggunakan Algoritma Monge Elkan *Distance*".

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Informatika/Ilmu Komputer PTIIK Universitas Brawijaya.

Penulis menyadari tanpa keterlibatan dan sumbangsih dari berbagai pihak, suliat bagi penulis untuk menyelesaikan skripsi ini. Maka dari itu penulis ingin mengucapkan terima kasih kepada :

1. Lailil Muflikhah, S.Kom., M.Sc selaku pembimbing I dan Indriati, ST., M.Kom selaku pembimbing II. Terima kasih atas semua waktu dan bimbingan yang telah diberikan dalam proses penyelesaian skripsi ini.
2. Ibu Rusnanik, Bapak Sunarko Iwan Y, dan seluruh keluarga yang selalu memberikan semangat, motivasi, dan doa restu.
3. Drs. Marji, MT. Selaku ketua program studi ilmu komputer Universitas Brawijaya Malang.
4. Segenap bapak dan ibu dosen PTIIK atas ilmu yang diberikan kepada penulis.
5. Segenap staf dan karyawan PTIIK Universitas Brawijaya yang telah membantu kelancaran pengerjaan skripsi.
6. Teman-teman ilkomers angkatan 2009 yang selalu bertukar semangat dalam pengerjaan skripsi ini.
7. Teman-teman GM dan Fastlap yang selalu memberikan masukkan atau pendapat yang memotivasi penulis.
8. Dan semua pihak lain yang telah membantu terselesaikannya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, oleh karena itu, saran dan kritik membangun dari pembaca senantiasa diperlukan untuk memperbaiki mutu penulisan selanjutnya. Semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, April 2014

Penulis

DETEKSI PLAGIARISME DOKUMEN TEKS MENGGUNAKAN ALGORITMA MONGE ELKAN *DISTANCE*

ABSTRAK

Tindakan plagiarisme sering terjadi pada semua jenjang pendidikan maupun dalam kehidupan bermasyarakat. Tindakan plagiarisme merupakan suatu tindakan yang sebenarnya dapat merusak kreatifitas ilmu pengetahuan dan semakin lama dapat menjadi suatu tindakan yang dianggap biasa karena sudah sering dilakukan oleh kebanyakan orang. Deteksi plagiarisme secara terkomputerisasi diperlukan agar mempermudah dalam proses pendeteksian tindakan plagiarisme. Pada skripsi ini digunakan algoritma Monge-Elkan distance untuk pendeteksian plagiarisme. Algoritma Monge-Elkan distance adalah algoritma perbandingan rekursif untuk dua string panjang yang memiliki dua atau lebih jumlah token/kata. Hasil dari pengujian, didapatkan penurunan nilai prosentase error sebesar 0,61% pada dokumen uji perubahan kata kerja aktif menjadi pasif dan juga sebaliknya setelah dilakukan proses stemming. Pada pengujian tersebut, threshold strukutral kalimat optimal yang digunakan adalah 0,5 yang berarti kalimat yang dibandingkan memiliki nilai struktural kalimat diatas 50% dan threshold internal similarity optimal yang digunakan adalah 0,8 yang berarti nilai kesamaan kata yang digunakan diatas 80%.

Kata Kunci : Monge Elkan distance, Jaro Winkler distance, plagiarisme, string matching

TEXT DOCUMENT PLAGIARISM DETECTION USING MONGE ELKAN DISTANCE ALGORITHM

ABSTRACT

Plagiarism often occurs at all levels of education and in social life. Plagiarism is an action that can ruin a creativity of science and the longer, it can be an action that is considered ordinary because it is often done by most people. Computerized plagiarism detection is needed to simplify the plagiarism detection process. In this paper, Monge-Elkan distance algrorithm is used for plagiarism detection. Monge – Elkan distance algorithm is recursive mathcing scheme for comparing two long string that have two or more tokens/ words. The result of testing, impairment precentage error of 0,61% was found on a test document that active verbs have been changed to passive verbs or vice versa. In that test, optimal threshold of structural sentence is 0,5 which means that compared sentences have values above 50% and optimal threhsold internal similarity is 0,8 which means the value of the similarity of words used above 80%.

Keyword : Monge Elkan distance, Jaro Winkler distance, plagiarism, string matching

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR GRAFIK

DAFTAR KODE SUMBER

DAFTAR RUMUS

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini, perkembangan teknologi sudah mulai berkembang sangat pesat. Hal ini berdampak pada mudahnya mencari informasi yang ada di seluruh dunia dengan hanya mengakses internet. Kemudahan mendapatkan informasi ini mempunyai dampak positif yang cukup besar, tetapi di lain pihak hal ini dapat juga membuat dampak negatif yang cukup besar pula. Dengan mudahnya mendapatkan informasi ini dapat disalahgunakan oleh sebagaian orang untuk melakukan tindakan penjiplakan atau plagiarisme terhadap hasil karya orang lain. Tindakan plagiarisme sering terjadi pada semua jenjang pendidikan maupun dalam kehidupan bermasyarakat. Suatu tindakan yang sebenarnya dapat merusak kreatifitas ilmu pengetahuan ini, semakin lama tidak dipungkiri dapat menjadi suatu tindakan yang dianggap biasa karena sudah terlalu banyak orang yang melakukan tindakan plagiarisme secara sengaja ataupun tidak.

Perilaku plagiarisme di dunia akademik, kadang terjadi karena adanya tekanan tugas yang diberikan oleh pendidik ke peserta didik terlalu berlebihan dan tidak sesuai dengan waktu yang tersedia, sehingga sangat rentan terjadi "gunting tempel" agar tugas yang dikerjakan dapat selesai tepat pada waktunya. Tetapi yang paling sering dilakukan peserta didik adalah meremehkan tugas yang diberikan oleh pendidik, sehingga pada saat waktu pengumpulan tugas semakin dekat, peserta didik akan kewalahan mengerjakan tugas tersebut dan hal terakhir yang biasanya dilakukan adalah melakukan plagiarisme. Padahal, tindakan plagiarisme ini tidak bisa diterima kapanpun dengan alasan apapun. Bagi peserta didik seharusnya sangat diperlukan suatu sikap jujur atau malu untuk melakukan penjiplakan suatu karya yang bukan miliknya, karena dengan cara inilah suatu karya dapat menjadi lebih bernilai.

Plagiarisme berasal dari kata Latin *plagiarius* yang berarti merampok, membajak. Plagiarisme merupakan tindakan pencurian atau kebohongan intelektual. Menurut Oxford Dictionaries (Oxforddictionaries.com) plagiarisme adalah suatu upaya untuk mengambil hasil kerja atau ide milik orang lain dan

mengambil alih hasil kepemilikan tersebut. Dictionary.reference.com mendifinisikan sebagai suatu cara memakai atau minjiplak tulisan atau pemikiran orang lain tanpa izin si pembuat. Menurut Sudigdo (2007) Plagiarisme adalah tindakan menyerahkan (*submitting*) atau menyajikan (*presenting*) ide atau kata/kalimat orang lain tanpa menyebut sumbernya. Plagiarisme dapat terjadi dari berbagai aspek seperti pengutipan suatu tulisan tanpa menyebutkan sumber kutipan, mengakui gagasan orang lain sebagai pemikiran sendiri, dan mengakui temuan orang lain sebagai kepunyaan sendiri. Bahkan, setiap orang yang yang melakukan perubahan beberapa kata dalam kalimat dengan kata-katanya sendiri seperti penambahan imbuhan dan pengacakan urutan kata, tanpa menyebutkan sumber dari penulis asli merupakan bentuk dari plagiarism.

Pada penelitian sebelumnya yang dilakukan oleh Kurniawati dan Puspitodjati (2010), telah melakukan penelitian implementasi algoritma Jaro-Winkler *distance* untuk membandingkan kesamaan dokumen berbahasa Indonesia. Namun, algoritma tersebut masih memperhatikan urutan kata, sehingga dokumen yang memiliki urutan kata berbeda walau isi tekstualnya sama akan sulit untuk dideteksi kemiripannya [KUR-10]. Untuk memperbaiki kekurangan tersebut, pada penelitian Jimenes dan Bacerra (2009), *string* text dibandingkan menggunakan algoritma Monge-Elkan didasari dengan perhitungan kesamaan *internal character-based*, contohnya adalah algoritma perbandingan teks Jaro-Winkler *distance*. Namun, *string* yang digunakan pada penelitian tersebut masih sebatas nama orang dan nama jalan.

Penelitian lain yang membahas deteksi plagiarisme adalah penelitian Aditya (2013). Penelitian tersebut menggunakan algoritma Damerau Levenshtein Distance untuk pencocokan string, serta menggunakan proses *stemming* pada tahap *preprocessing*. Algoritma *stemming* yang digunakan adalah algoritma *confix stripping* dengan refrensi dari algoritma *stemming* Nazief-Adriani. Hasil yang diperoleh dari penelilitan tersebut adalah adanya peningkatan nilai kesamaan antar dokumen yang terindikasi sebagai plagiarisme setelah dilakukan proses s*temming* terlebih dahulu.

Adapun algortima *stemming* selain algoritma *stemming confix stripping*, salah satunya adalah algoritma porter untuk Bahasa Indonesia. Agusta (2009),

dengan penelitian yang berjudul "Perbandingan Algoritma Stemming Porter Dengan Algoritma Nazief & Adriani Untuk Stemming Dokumen Teks Bahasa Indonesia" didapatkan waktu proses algoritma porter yang lebih cepat daripada algoritma Nazief & Adriani dalam proses *stemming.*

Berdasarkan latar belakang diatas, dalam penelitian ini akan dibuat sebuah aplikasi deteksi plagiarisme dokumen teks menggunakan algoritma Monge-Elkan *Distance* perhitungan awal menggunakan algoritma Jaro-Winkler *distance*. Untuk proses *stemming,* algoritma yang digunakan adalah algoritma Porter. Pengujian dilakukan dengan cara menghitung nilai kemiripan antar dokumen yang diindikasi sebagai plagiarisme, apakah dokumen tersebut termasuk plagiarisme ringan, sedang, atau berat.

.

1.2 Rumusan Masalah

Rumusan permasalahan dalam penelitian ini adalah sebagai berikut :

1. Bagaimana menerapkan Algoritma Monge-Elkan *distance* serta algoritma Jaro-Winkler *distance* dalam mendeteksi kemiripan isi dokumen teks?
2. Bagaimana pengaruh *stemming* Porter-Fadillah terhadap hasil akhir nilai *similarity?*

1.3 Batasan Masalah

Untuk memfokuskan penelitian yang akan dilakukan, perlu pembatasan permasalahan sebagai berikut :

1. Data yang digunakan berbentuk dokumen text yang berekstensi .txt
2. Data diperoleh dari artikel berita pada website *www.kompas.com* dan *www.detik.com* dan latar belakang dari makalah ilmiah.
3. Jumlah kata pada data yang digunakan kurang dari lima ratus kata.
4. Dokumen yang digunakan adalah dokumen berbahasa Indonesia.
5. Tidak memperhatikan kesamaan arti kata.

1.4 Tujuan Penelitian

Penelitian ini bertujuan sebagai berikut :

1. Menerapkan Algoritma Monge-Elkan *Distance* dalam mendeteksi kemiripan isi dokumen teks.
2. Mengetahui pengaruh *stemming* (Porter Fadillah) terhadap nilai kemiripan dokumen.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut :

1. Dapat membantu pendidik dalam menentukan kemiripan hasil dari tugas peserta didik.
2. Mengetahui prosentase kemiripan dokumen sehingga dapat digunakan sebagai bahan pertimbangan adanya tindakan plagiarisme.
3. Mengurangi tindakan *repost* (*posting* ulang dari *posting* user lain) pada suatu forum web.

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut :

BAB I PENDAHULUAN

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai pustaka yang digunakan dalam pengerjaan skripsi. Teori – Teori yang terdapat dalam bab ini mencangkup *text minning, text processing (tokenizing, case folding, stemming), stopword removal,* serta metode pencocokan string algoritma Monge-Elkan *Distance* dan Jaro-Winkler.

BAB III METODOLOGI DAN PERANCANGAN SISTEM

Pada bab ini dibahas mengenai urutan langkah – langkah pengerjaan untuk perancangan *user interface,* mengidentifikasi

plagiarism,serta perhitungan manual Algoritma Monge-Elkan *Distance*.

BAB IV IMPLEMENTASI

Pada bab ini menjelaskan tentang implementasi dari Algoritma Monge-Elkan *Distance* dalam perancangan aplikasi deteksi *plagiarism*.

BAB V HASIL DAN PEMBAHASAN

Melakukan analisa dari hasil pengujian algoritma.

BAB VI KESIMPULAN DAN SARAN

Pada bab ini berisi tentang kesimpulan yang didapat dari pembuatan skripsi ini serta saran-saran yang dapat berguna untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini, terdiri dari kajian pustaka dan tinjauan pustaka. Kajian pustaka akan membahas penelitian yang telah ada sebelumnya dan metode yang digunakan. Sedangkan tinjauan pustaka akan dibahas teori pendukung yang berkaitan dengan penelitian yang meliputi tiga pokok bahasan, antara lain *plagiarisme, text mining,*dan *String Matching.*

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahasa penelitian sebelumnya yang berjudul "Implementasi Algoritma Jaro-Winkler *Distance* untuk menbandingkan Kesaman Dokumen Berbahasa Indonesia". Penelitian ini membahas tentang penentuan kemiripan isi dokumen dengan menggunakan algoritma Jaro-Winkler *distance*[KUR-10].

Perbedaan yang dibuat pada penelitian ini adalah penggunaan algoritma yang digunakan. Algoritma yang digunakan adalah algoritma Monge Elkan *distance*. Algoritma Monge Elkan *distance* sebelumya digunakan pada penelitian yang berjudul "*Generalized Mongue-Elkan Method for Approximate Text String Comparison"* [GME-09]. Penelitian ini menggabungkan algoritma Monge Elkan *distance* dengan algoritma *string matching* lainnya seperti algoritma Jaro-Winkler *distance.* Hasil dari perhitungan Jaro-Winkler *distance* selanjutnya akan digunakan pada perhitungan Monge Elkan *distance.* Penggabungan dua algoritma ini bertujuan untuk menutupi kekurangan pada algoritma Jaro-Winkler *distance* yang hanya mengenali kalimat dengan urutan kata yang sama[KUR-10].

2.2 Plagiarisme

2.2.1 Pengertian Plagiarisme

Plagiarisme adalah berbuat sesuatu yang mengganggap karya orang lain berupa fakta, penjelasan, ungkapan, kalimat orang lain seolah-olah milik kita dengan cara yang tidak sah, seperti tanpa mencantumkan sumber. Plagiarisme dianggap tindakan kriminal karena merupakan tindakan mencuri hak cipta orang

lain. Di Indonesia perlindungan hak cipta diatur dalam Undang – Undang Republik Indonesia Nomor 19 Tahun 2002 Tentang Hak Cipta [CSK-13].

Plagiarisme merupakan permasalah yang tidak hanya melanggar hak cipta atau kepemilikan. Hal ini dapat dilihat dari segi pandang pembaca, plagiarism merupakan tindakan yang dapat menimbulkan kesalahpahaman tentang ide ataupun tulisan yang dibaca merupakan hasil penulis tersebut atau bukan. Para pelajar/mahasiswa diperbolehkan untuk menciptakan suatu karya baru dari pengembangan ide orang lain, tetapi yang perlu digarisbawahi adalah bagaimana cara atau etika yang benar untuk mengutip kata, paragraf, ataupun ide dari orang lain agar terhindar dari tindakan plagiarisme.

Plagiarisme dibedakan menjadi beberapa kategori dengan dasar yang berbeda. Kategori yang pertama, berdasarkan aspek yang dicuri yaitu plagiarisme ide, isi (data penelitian), tulisan, dan plagiarisme total [SDS-07].

Kedua, berdasarkan sengaja atau tidaknya plagiarisme. Plagiarisme ini dibedakan menjadi dua jenis plagiarisme yaitu plagiarisme yang disengaja dan yang tidak disengaja. Bila seseorang menggunakan ide, kata, frase, kalimat, atau paragraf orang lain tanpa menyebut sumber, mungkin hal tersebut memang disengaja oleh penulis, namun mungkin juga karena "tidak sengaja", misalnya ia tidak mengetahui bahwa hal tersebut tidak boleh dilakukan. Dengan "ketikdaktahuan" ini penulis tidak bisa dibenarkan melakukan plagiarisme, karena plagiarisme bersifat universal, meskipun belum adanya peraturan di suatu lembaga tentang plagiarisme tidak membuat orang boleh melakukan plagiarisme [SDS-07].

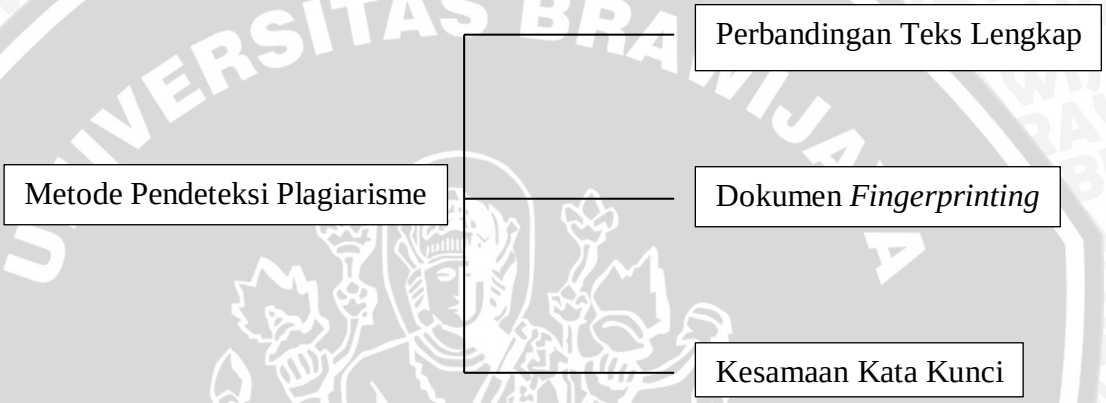
Ketiga, berdasarkan proporsi atau presentasi kata, kalimat, paragraf yang dibajak. Plagiarisme ini dibedakan menjadi tiga jenis plagiarisme yaitu plagiarisme ringan (<30%), sedang (30-70%), dan plagiarisme berat atau total (>70) [SDS-07].

Keempat, berdasarkan pola plagiarisme yaitu plagiarisme kata demi kata (*word for word plagiarism)* dan plagiarisme mosaik. Plagiarisme kata demi kata merupakan plagiarisme yang paling mudah ditentukan, jenis ini dapat merupakan kalimat, paragraf, atau bahkan seluruh tulisan. Plagiarisme mosaik paling susah dideteksi, karena plagiarisme ini mengambil kata, frase, atau kalimat dari orang

lain lalu menggabungkannya dengan kata, frase, atau kalimat dari orang lain yang membuat seolah – olah tulisan tersebut adalah tulisan baru tanpa memberikan rujukan [SDS-07].

2.2.2 Metode Pendeteksi Plagiarisme

Metode Pendeteksi Plagiarisme dibagi menjadi tiga bagian yatu metode perbandingan teks lengkap, metode *fingerprinting*, dan metode kesamaan kata kunci.



Gambar 2.1 Klasifikasi Metode Pendeteksi Plagiarisme

Sumber : [ANK-08]

Berikut ini penjelasan dari masing-masing metode dan algoritma pendeteksi plagiarisme [ANK-08] :

1. Perbandingan Teks Lengkap

   Metode ini diterapkan dengan membandingkan semua isi dokumen. Pendekatan dengan metode ini membutuhkan waktu yang cukup lama tetapi cukup efektif untuk menghitung tingkat plagiarisme. Algoritma yang digunakan dalam metode ini adalah algoritma brute force, edit distance, boyer moore, jaro-winkler, dan algoritma lavenshtein distance.

2. Dokumen *Fingerprinting*

   Dokumen *Fingerprinting* merupakan metode yang mendeteksi keakuratan antar dokumen menggunakan teknik *hashing*. Teknik *hashing* adalah

sebuah fungsi yang mengkonversi setiap string pada dokumen menjadi bilangan. Misalnya Rabin-Karp, Winnowing, dan Manber.

3. Kesamaan Kata Kunci.

   Prinsip dari metode ini adalah mengekstrak kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen yang lain. Pendekatan yang digunakan pada metode ini adalah teknik dot.

2.3 Text Mining

2.3.1 Pengertian *Text Mining*

Text Mining adalah pengambilan data yang berupa teks pada suatu dokumen teks untuk mencari kata – kata yang mewakili isi dokumen tersebut yang dapat berguna untuk menyelesaikan masalah bisnis yang dapat disebut *text analytics*.

Menurut Ranveer Kaur dan Shruti Aggarwal (2013) , *Text Mining* adalah pengambilan suatu pola yang berguna dari sumber yang berbentuk text. *Text Mining* juga dikenal sebagai *Intelligent Text Anlysis, Text Data mining,* dan *Knowledge Discovery in Text (KDT). Text Mining* bekerja dengan mengadopsi beberapa teknik dari bidang lain. Beberapa teknik tersebut adalah *information Retrieval, Natural Language Processing, Information Extraction* dan *Data mining.*

Beberapa penerapan *text mining* dapat dilihat pada pemilihan berita sesuai dengan kategorinya seperti kategori olah raga, kategori ekonomi, dan kategori teknologi. Selain pemilihan kategori tersebut, *text mining* juga dapat dikembangkan dalam perihal pengambilan ide pokok dari sebuah paragraf ataupun artikel.

2.3.2 Information Retrieval

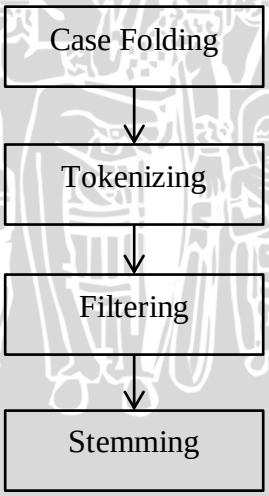
Information Retrieval merupakan bagian dari *computer science* yang berhubungan dengan pengambilan informasi dari dokumen – dokumen yang didasarkan pada isi dan konteks dari dokumen – dokumen tersebut. *Information Retrieval* merupakan suatu pencarian informasi yang didasarkan pada suatu *query* dari inputan user yang diharapkan dapat memberikan informasi dari kumpulan

dokumen yang sesuai dari *query* tersebut. Salah satu aplikasi umum dari *Information Retrieval* adalah *search engine* atau mesin pencarian yang terapat pada jaringan internet [CSK-13].

Information Retrieval System (IRS) tidak memberi tahu pengguna masalah yang ditanyakannya. Sistem tersebut hanya memberitahukan keberadaan dan keterangan dokumen yang berhubungan dengan permintaan pengguna. Penggunaan bahasa natural sebagai bahasa query pada IRS memberikan kemudahan kepada pengguna dalam merepresentasikan kebutuhan informasinya dalam bentuk *query* [USU-09].

2.3.3 Extraksi Dokumen

Pada ekstraksi dokumen teks, tahap yang dilakukan saat mengimplementasikan *text mining* adalah tahap *preprocessing text. Preprocessing* adalah tahapan untuk menyeleksi data yang akan diproses pada setiap dokumen. Proses *preprocessing* meliputi *case folding, tokenizing, filtering, dan stemming.* Tahap *preprocessing* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Tahap *Preprocessing*

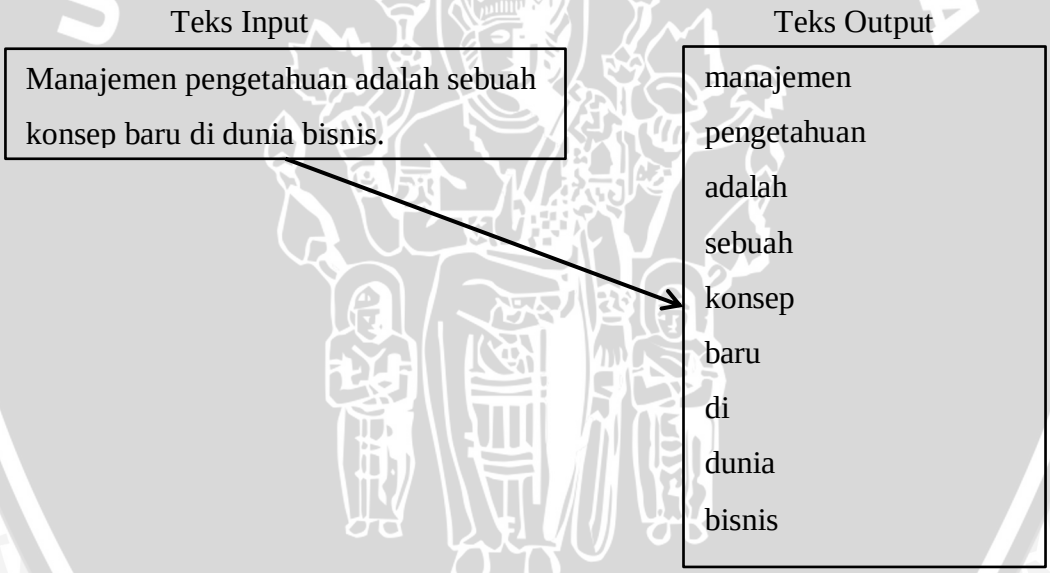
Sumber : [CSK-13]

2.3.4 Case Folding dan Tokenizing

Dokumen teks memilik struktur data yang tidak teratur, seperti pada penggunaan huruf kapital. Dengan adanya *case folding* dengan cara mengkonversi

keseluruhan dokumen teks menjadi dokumen yang memiliki bentuk standar. Bentuk standar disini biasanya merupakan bentuk huruf kecil atau *lowercase.* Dalam sebuah dokumen sering terdapat salah pengetikan huruf besar atau huruf kecil dari suatu kata, contoh penulisan "komputer, "komputEr" dan "kOmputer". Dengan adanya pengkonversian menjadi huruf kecil dua kata tersebut masih tergolong dalam kata yang sama yaitu "komputer". Dalam menerapkan *case folding* huruf yang dikonversi adalah huruf 'a' sampai dengna 'z', selain karakter tersebut dianggap sebagai *delimiter* (pemisah kata).

Tahap *tokenizing* / *parsing* adalah tahap pemotongan string menjadi tiap-tiap kata yang menyusunnya. Tahap ini menggunakakan karakter *delimiter* sebagai pemotong string. Contoh dari tahap *tokenizing* dapat dilihat pada Gambar 2.3.



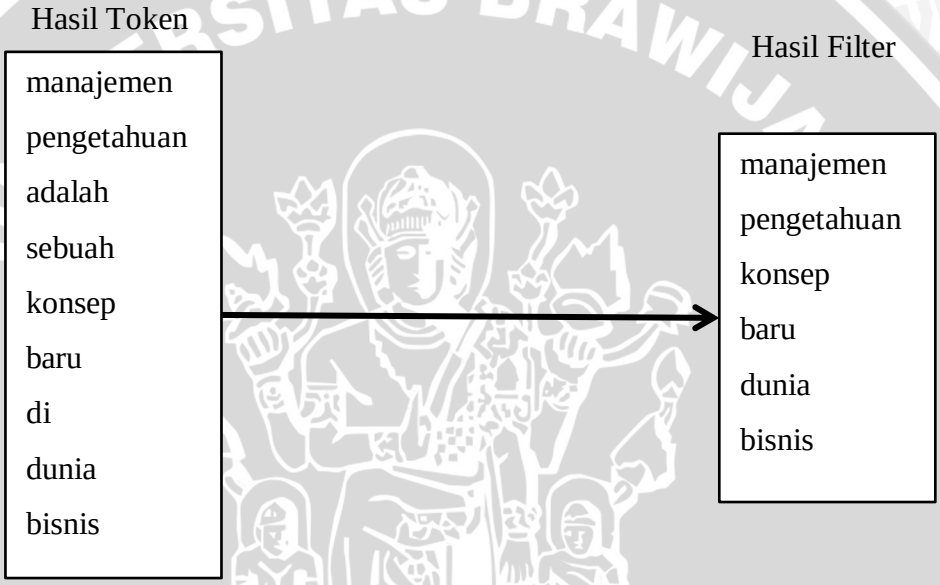
Gambar 2.3 Tahap *case folding* dan *tokenizing*

Sumber : [NUG-11]

Tokenizing secara garis besar adalah memecah suatu string menjadi dalam kata – kata. Dalam proses pemecahan digunakan karakter – karakter tertentu (*delimiter)* sebagai pemisah antar kata. Sebagai contoh karakter spasi, enter, dan tab dianggap sebagai pemisah kata.

2.3.5 Filtering

Setiap kata yang sudah tertoken masih terdapat kata – kata yang bisa dianggap kurang penting yang biasa disebut *stopword*. Contoh *stopwords* adalah "yang", "di", "dan", "ke" dan seterusnya. Data *stopword* yang digunakan pada penlitian ini diambil dari jurnal Fadillah Z tala yang berjudul " *A Study of Stemming Effects on Information Retreival in Bahasa Indonesia".* Sesuai dari *stopwords* tersebut proses filtering dapat dilakukan seperti pada Gambar 2.4.



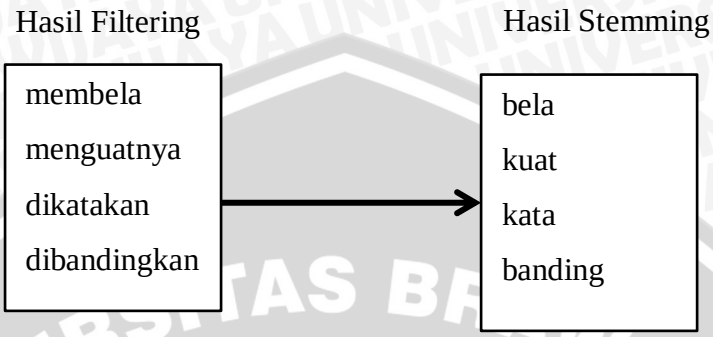
Gambar 2.4 Tahap *filtering*

Sumber : [NUG-11]

2.3.6 Stemming

Stemming merupakan suatu cara untuk meningkatkan hasil dari *Information Retrieval* (IR) dengan cara menghilangkan imbuhan – imbuhan pada suatu kata. Dengan teknik ini nilai kemiripan dari perbandingan kata dapat lebih ditingkatkan, seperti pada kata "bersama", "kebersamaan", dan "menyamai", jika kata tersebut tidak dilakukan teknik *Stemming* maka nilai kesamaan kata akan rendah karena adanya perbedaan kata imbuhan. Lain halnya jika dilakukan teknik *stemming* , ketiga kata tersebut akan memiliki nilai sama karena mempunyai kata dasar yang sama yaitu "sama" *Stemming* untuk kata berbahasa Inggris dengan kata yang berbahasa Indonesia memiliki cara yang berbeda. Untuk kata berbahasa

Inggris yang dhilingkan adalah sufiks, sedangkan pada kata berbahasa indonesia yang dihilangkan adalah sufiks, prefiks, dan konfiks [LEA-09]. Contoh dari tahap ini dapat dilihat pada Gambar 2.5.



Gambar 2.5 Tahap *Stemming*

Sumber : [NUG-11]

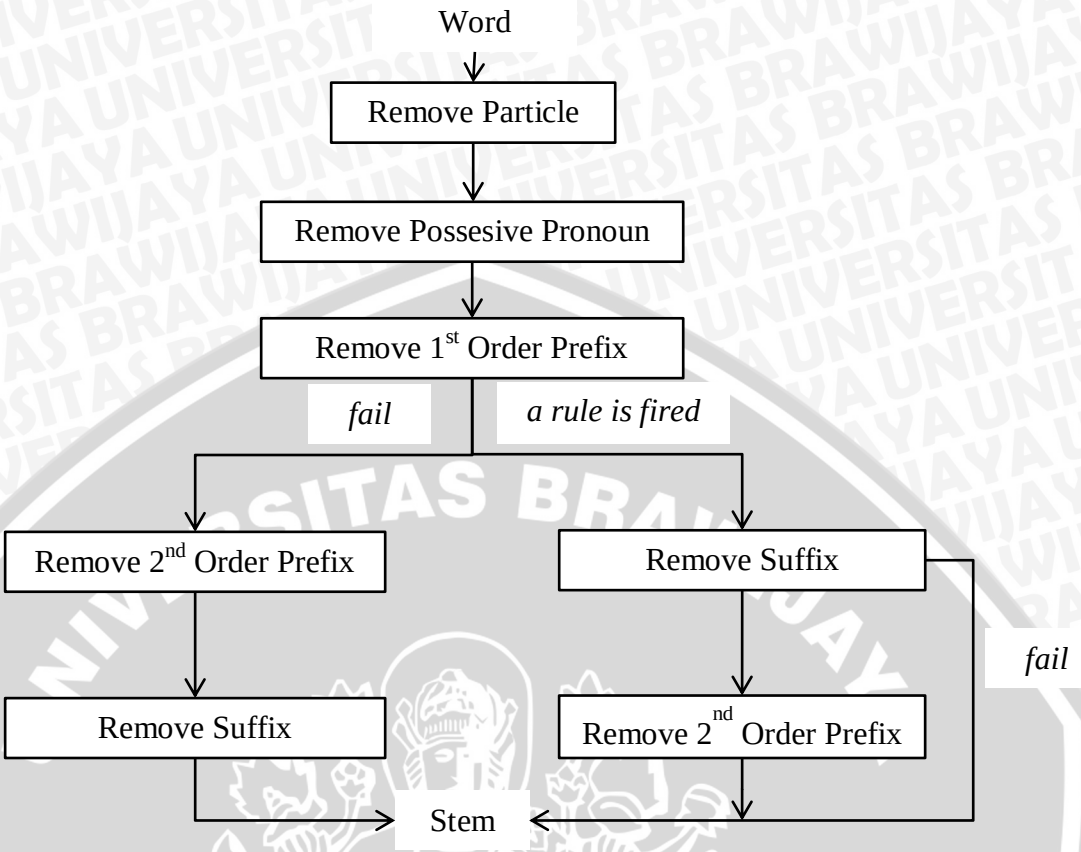
2.3.6.1 Algoritma *Stemming* Porter-Fadillah

Terdapat berapa algoritma *stemming* untuk dokumen Berbahasa Indonesia seperti algoritma Nazief-adriani, Arifin, Vega, dan Porter. Algoritma Porter merupakan algoritma stemming yang dikembangkan oleh Martin Porter pada tahun 1980. Adapun langkah – langkah algoritma porter yang dikembangkan oleh Fadillah untuk *stemming* Bahasa Indonesia yaitu [LEA-09] :

1. Hapus *particle*
2. Hapus *possesive pronoun*
3. Hapus awalan pertama. Jika tidak ada lanjutkan ke langkah 4a, jika ada maka lanjutkan ke langkah 4b.
4. a. Hapus awalan kedua, lanjutkan ke langkah 5a.

   b. Hapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai *root word* / kata dasar. Jika ditemukan maka lanjutkan ke langkah 5b.
5. a. Hapus akhiran, kemudian kata akhir diasumsikan sebagai *root word.*

   b. Hapus awalan kedua, kemudian kata akhir diasumsikan sebagai *root word.*



Gambar 2.6 Bagan Stemming Porter-Fadillah untuk Bahasa Indonesia

Sumber : [FAZ-03]

Terdapat 5 kelompok aturan pada algoritma Porter untuk Bahasa Indonesia. Aturan tersebut dapat dilihat pada Tabel 2.1 sampai 2.6

Tabel 2.1 Aturan Untuk *Inflectional Particle*

Sumber : [FAZ-03]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kah	NULL	2	NULL	bukukah → buku
lah	NULL	2	NULL	adalah → ada
pun	NULL	2	NULL	bukupun → buku

Tabel 2.2 Aturan Untuk *Inflectional Possesive Pronoun*

Sumber : [FAZ-03]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
ku	NULL	2	NULL	bukuku → buku
mu	NULL	2	NULL	bukumu → buku
nya	NULL	2	NULL	bukunya → buku

Tabel 2.3 Aturan Untuk *First Order Derivational Prefix*

Sumber : [FAZ-03]

Prefix	Replacement	Measure Condition	Additional Condition	Examples
meng	NULL	2	NULL	mengukur → ukur
meny	S	2	V…*	menyapu → sapu
men	NULL	2	NULL	menduga → duga
mem	P	2	V…	memaksa → paksa
mem	NULL	2	NULL	membaca → baca
me	NULL	2	NULL	merusak → rusak
peng	NULL	2	NULL	pengukur → ukur
peny	S	2	V…	penyapu → sapu
pen	NULL	2	NULL	penduga → duga
pem	P	2	V…	pemilah → pilah
pem	NULL	2	NULL	pembaca → baca
di	NULL	2	NULL	diukur → ukur
ter	NULL	2	NULL	tersapu → sapu
ke	NULL	2	NULL	kekasih → kasih

* Notasi ini berarti stem diawali dengan huruf vokal.

Tabel 2.4 Aturan Untuk *Second Order Derivational Prefix*

Sumber : [FAZ-03]

Prefix	Replacement	Measure Condition	Additional Condition	examples
ber	NULL	2	NULL	berlari → lari
bel	NULL	2	Ajar	belajar → ajar
be	NULL	2	K*er…	bekerja → kerja
per	NULL	2	NULL	perjelas → jelas
pel	NULL	2	Ajar	pelajar → ajar
pe	NULL	2	NULL	pekerja → kerja

* Notasi ini berarti stem diawali dengan huruf konsonan.

Tabel 2.5 Aturan untuk *Derivational Suffix*

Sumber : [FAZ-03]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kan	NULL	2	Prefix bukan anggota {ke, peng}	tarikkan → tarik mengambilkan → ambil
an	NULL	2	Prefix bukan anggota {di, meng, ter}	makanan → makan perjanjian → janji
i	NULL	2	Prefix bukan anggota {ber, ke , peng}	tandai → tanda (men)dapati → dapat

Tabel 2.6 Contoh suku kata dalam Bahasa Indonesia

Sumber : [FAZ-03]

Measure	examples	Syllables
0	kh, ng, ny	kh, ng , ny
1	ma, af, nya , nga	ma, af, nya , nga
2	maaf, kami, rumpun, kompleks	ma-af, ka-mi, rum-pun, kom-pleks
3	mengapa, menggunung, tandai	meng-a-pa, meng-gu-nung, tan-da-i

Contoh penerapan algoritma porter :

mempermudah**kannyapun** →

mempermudah**kannya** + **pun** (*Particle*) → hapus *particle* **pun**

mempermudah**kan** + **nya** (*Possesive Pronoun*) → hapus *Possesive Pronoun* –**nya**

mem (1st Order Prefix) + **per**mudah**kan** → hapus 1st Order Prefix mem-

per (2nd Order Prefix) + mudah**kan** → hapus 2nd Order Prefix per-

mudah + **kan** (Suffix) → hapus Suffix –kan

mudah → kata dasar / *rootword*

2.4 Suku Kata

Menurut Kamus Besar Bahasa Indonesia suku kata adalah struktur yang terjadi dari satu atau urutan fonem yang merupakan bagian kata. Setiap suku kata ditandai dengan sebuah vokal (termasuk diftong). Fonem adalah dua huruf yang terdiri dari konsonan seperti "kh", "ng", "ny", dan "sy". Sedangkan diftong adalah dua vokal yang membentuk kesatuan bunyi yaitu : "au", "ai", "oi". Bahasa Indonesia mengenal beberapa pola umum suku kata, yaitu [TAB-00] :

a) V a-nak, ba-u
b) VK an-da, da-un
c) KV se-bab, man-di
d) KVK lan-tai, ma-kan
e) KKV pra-ha-rai, sas-tra
f) KKVK frik-si, kon-trak
g) VKK eks, ons
h) KVKK pers, kon-teks
i) KKVKK kom-pleks
j) KKKV in-stru-men
k) KKKVK struk-tur

Untuk memenggal suku kata, dapat digunakan pedoman berikut ini :

a. Kalau di tengah kata terdapat dua vokal berturutan (selain diftong), pemisahan dilakukan di antar kedua vokal tersebut.
b. Kalau di tengah kata terdapat konsonan di antara dua vokal, pemisahan dilakukan sebelum konsonan tersebut
c. Kalau di tengah kata terdapat dua konsonan atau lebih, pemisahan dilakukan setelah konsonan pertama
d. Imbuhan dan partikel yang biasanya ditulis serangkai dengan kata dasar, pada penyukuan dipisahkan

2.5 *String Matching* (Pencocokan Teks)

2.5.1 Pengertian *String Matching*

String adalah sebuah representasi data dalam computer yang berupa kumpulan dari karakter/huruf yang disimpan dalam sebuah array/tabel yang memiliki indeks. Dengan demikian, string dapat dengan mudah diakses karakter – karakternya (dengan mengakses indeks tabel) dan membandingkannya dengan string lain [ARD -08].

String matching atau pencocokan *string* adalah suatu cara untuk membandingkan dua string untuk mendapatkan nilai kesmaaan antar dua string tersebut. Dalam studi ilmu komputer *string matching* merupakan pokok bahasan yang sering digunakan untuk pertukaran informasi seperti pada literatur, karya ilmiah, halaman web, dan sebagainya[NUG-11].

Permasalahan *string matching* sering ditemui dalam dunia informatika. Contoh permasalahan tersebut adalah pencarian teks pada dokumen, pencarian pada *search engine* (Yahoo.com, google.com), serta dapat digunakan pada kamus.

2.5.2 Algoritma Jaro-Winkler

Jaro-Winkler *distance* merupakan pengembangan dari algoritma Jaro *distance* yaitu suatu algoritma untuk mengukur nilai kesamaan antar dua string. Algoritma ini biasanya digunakan untuk mencari kata ataupun mendeteksi duplikasi kata. Semakin tinggi nilai dari Jaro-Winkler *distance*, semakin tinggi pula kemiripan antar dua string atau kata, sebaliknya semakin rendah nilai Jaro-Winkler *distance* maka semakin rendah pula kemiripan antar dua string atau kata. Untuk nilai string yang sama seperti kata “makan” dengan “makan” makan nilai tersebut akan memiliki nilai 1 (nilai tertinggi). Nilai normal dari dari Jaro-Winkler *distance* yaitu 0 sampai 1. Nilai 0 untuk tidak ada kesamaan dan 1 untuk sama persis [KUR-10].

Adapaun dasar dari algoritma Jaro untuk 2 string *s* dan *t* yaitu [SOK-13] :

1. Menghitung panjang string |s| dan |t| ,
2. Menemukan jumlah karakter yang sama di dalam dua string, karakter tersebut *s*[i] dan *t*[j] yaitu *s*[i]=*t*[j] dengan jarak $|i-j| \leq \frac{1}{2} \min\{|s|,|t|\}$
3. Menemukan jumlah transposisi.

Pada algoritma Jaro digunakan rumus untuk menghitung kesamaan antara dua string yaitu *s* dan *t*.

$$sim_{jaro}(s,t) = \frac{1}{3}\left(\frac{c}{|s|} + \frac{c}{|t|} + \frac{c - Tr}{c}\right) \qquad (2.1)$$

c = jumlah karakter yang sama

|*s*| = panjang string pertama

|*t*| = panjang string kedua

Tr = jumlah transposisi

Sedangkan pada algoritma Jaro-Winkler *distance* digunakan nilai *p* yaitu nilai prefix dari kata yang dibandingkan. Algoritma tersebut didefinisikan [WIL-03]

$$sim_{Jw}(s,t) = sim_{jaro}(s,t) + \frac{P'}{10}.\left(1 - sim_{jaro}(s,t)\right) \qquad (2.2)$$

Dimana *P'* adalah panjang karakter prefix yang sama *s* dan *t* dimulai dari awal string sampai maksimal 4 karakter. Berikut ini adalah contoh perhitungan Jaro-Winkler *distance*. Jika string *s* MARTHA dan *t* MARHTA maka [KUR-10] :

c = 6

|*s*| = 6

|*t*| = 6

Karakter yang tertukar hanyalah 'T' dan 'H', maka *Tr* = 1. Maka nilai Jaro *distance* adalah :

$$sim_{jaro}(s,t) = \frac{1}{3}.\left(\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}\right) = 0.944$$

Kemudian perhitungan untuk nilai *P'* pada algoritma Jaro-Winkler dapat diketahui bernilai *P'* = 3. Maka nilai Jaro-Winkler *distance* adalah :

$$sim_{Jw}(s,t) = 0.944 + \frac{3}{10}.(1 - 0.944) = 0.961$$

Hasil diatas mengindikasikan adanya kesamaan teks sebesar 96.1% dari kedua string masukan *s* dan *t*.

2.5.3 Algoritma Monge-Elkan *distance*

Monge-Elkan *distance* merupakan suatu cara untuk membandingkan dua string yang terdiri dari beberapa kata (*tokens*), dengan memakai *internal similarity* (Jaro-Winkler) *sim'(s,t)* dapat memberikan nilai kesamaan antara dua string kata. Diberi dua teks *A* dan *B*, dengan |A| dan |*B*| adalah *tokens* |A| = $s_1 \ldots s_K$ dan |B|=$t_1 \ldots t_L$. Berikut adalah persamaan Monge-Elkan *distance* [WIL-03].

$$sim_{MongeElkan}(A,B) = \frac{1}{K}\sum_{i=1}^{K} \max_{j=1}^{L} sim'(s_i, t_j) \quad (2.3)$$

Dimana :

K = jumlah token string yang pertama

L = jumlah token string yang kedua

s = token string pertama

t = token string kedua

String yang digunakan pada variabel *K* adalah *string* yang memiliki jumlah token lebih banyak antara *string* yang dibandingkan. Pada variabel *L*, string yang digunakan adalah *string* yang memiliki jumlah token lebih sedikit.

Berikut adalah contoh perhitungan Monge-Elkan *distance*.

A ="Lenovo inc."; s_1 ="Lenovo"; s_2 ="inc."

B ="Lenovo corp."; t_1 ="Lenovo"; t_2 ="corp."

$$sim'(s_1, t_1) = 1 - \frac{0}{6} = 1;\ sim'(s_1, t_2) = 1 - \frac{5}{6} = 0.1666$$

$$sim'(s_2, t_1) = 1 - \frac{5}{6} = 0.1666;\ sim'(s_2, t_2) = 1 - \frac{4}{4} = 0$$

$$sim_{MongeElkan}(A, B)$$

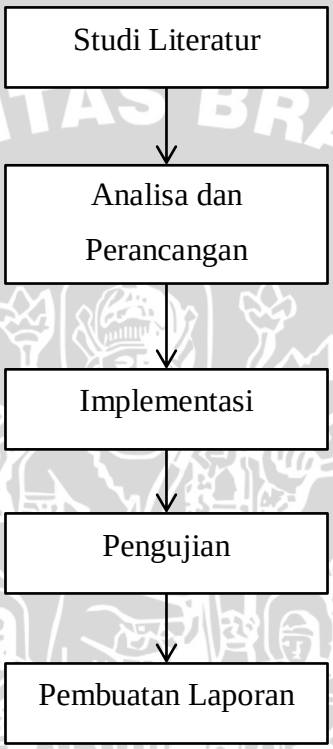
$$= \frac{1}{2}(\max(sim'(s_1, t_1), sim'(a_1, t_2))$$

$$+ \max(sim'(a_2, t_1), sim'(a_2, t_2)))$$

$$sim_{MongeElkan}(A, B) = \frac{1}{2}(1 + 0.1666) = 0.5833$$

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam pengerjaan tugas akhir ini, yaitu studi literatur, penyusunan dasar teori, analisa dan perancangan, uji coba sistem, dan yang terakhir adalah evaluasi dan analisa sistem. Desain penelitian secara umum digambarkan pada Gambar 3.1



Gambar 3.1 Desain Penelitian

Sumber : [Rancangan]

3.1 Studi Literatur

Studi literatur adalah mencari referensi teori yang relefan dengan kasus atau permasalah yang ditemukan. Referensi tersebut berisikan tentang :

1. Metode – metode *string matching*
2. Macam – macam tindakan plagiarisme
3. Metode *stemming* untuk Bahasa Indonesia

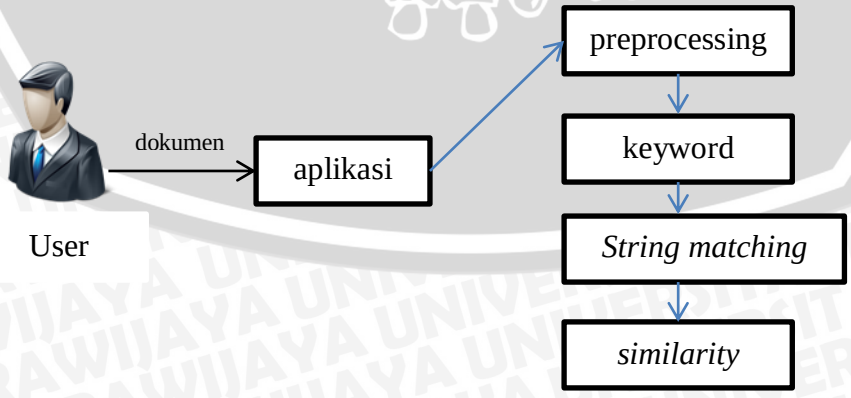
Referensi ini didapat dari buku, jurnal, artikel laporan penelitian, dan situs-situs di internet. Hasil dari studi literatur ini adalah terkoleksinya referensi yang relefan dengan perumusan masalah.

3.2 Analisa Sistem

Analisa kebutuhan sistem bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisa ini dilakukan dengan mengidentifikasi apa saja yang dibutuhkan oleh sistem.

Aplikasi pendeteksian kemiripan dokumen teks ini mempunyai cara kerja yaitu yang pertama user menginputkan dokumen teks asli dan dokumen teks yang ingin diuji, selanjutnya user memasukkan nilai ambang batas (*threshold*). Nilai *threshold* pertama digunakan untuk membatasi nilai minimal kesamaan teks dalam penghitungan bobot plagiarisme. Pada penelitian ini, pemberian nilai *threshold* digunakan untuk menyeleksi kalimat mana (antar dua dokumen) yang nantinya akan diproses pada perhitungan pembobotan plagiarisme atau dapat disebut kesamaan struktural. Perhitungan kesamaan struktural dilakukan dengan cara membangi jumlah kata yang sama dengan jumlah kata terbanyak diantara kedua kalimat. Nilai *threshold* yang digunakan memiliki rentang antara 0.0 sampai 1.0. *Threshold* yang kedua yaitu *threshold* yang digunakan untuk memberi ambang batas pada nilai *string matching*. Nilai *threshold* yang digunakan memiliki rentang antara 0.0 sampai 1.0.

Kemudian sistem akan akan menghitung presentase kimiripan (*similarity)* dan menggunakan algoritma Monge-Elkan *distance*. Gambar 3.2 adalah skema aliran data pada sistem.



Gambar 3.2 Skema aliran data

Sumber : [Rancangan]

Dokumen yang dibandingkan dalam sistem ini adalah dokumen berupa file .txt. Tahap *stemming* pada *preprocessing*, algoritma yang digunakan adalah algoritma Porter oleh Fadillah yang telah disesuaikan untuk Bahasa Indonesia Dari algoritma Porter untuk bahasa Inggris.

3.2.1 Kebutuhan Antar Muka

Kebutuhan antar muka merupakan kebutuhan yang akan ditampilkan oleh aplikasi. Kebutuhan tersebut meliputi kebutuhan input pada sistem dan perancangan *prototype* antar muka.

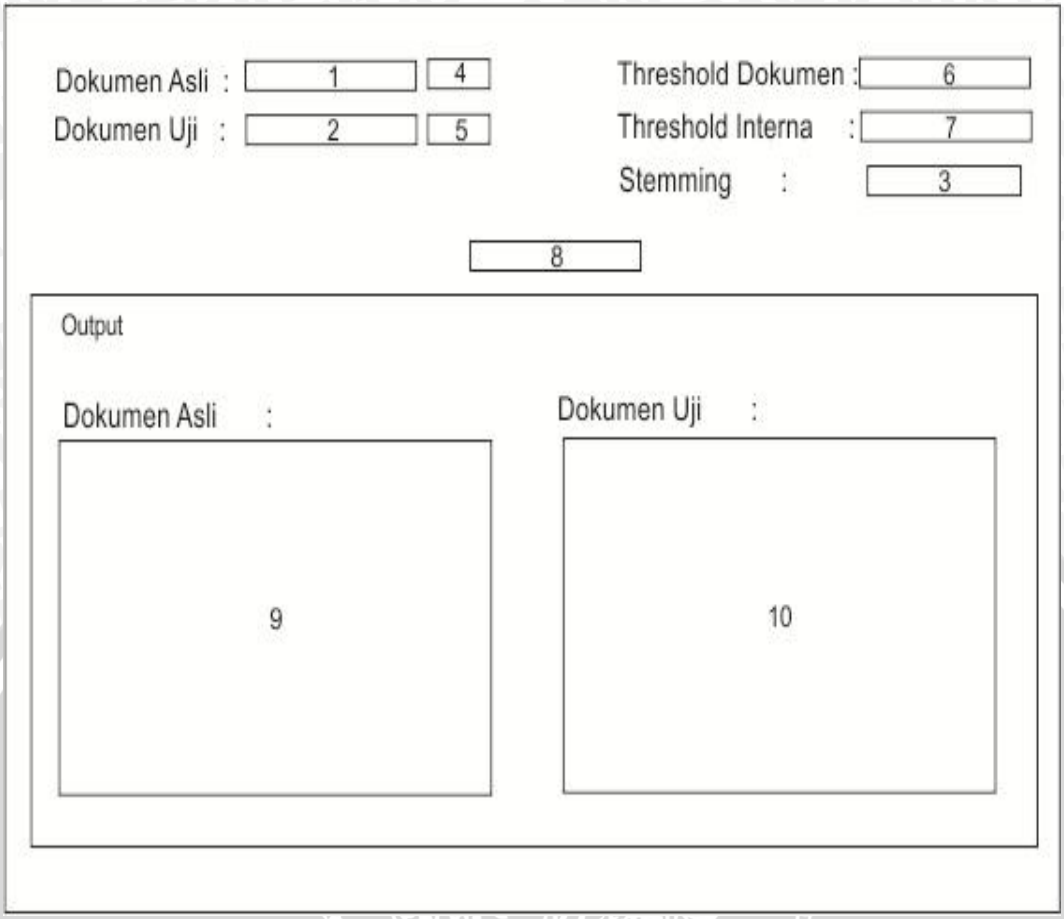
3.2.1.1 Perancangan Input pada Sistem

Pada aplikasi deteksi plagiarisme dokumen terdapat beberapa parameter yang harus diisi oleh user, yaitu :

1. Opsi pilihan penggunaan proses *stemming.*
2. Input dokumen asli
3. Input dokumen uji.
4. Input nilai *Threshold*

3.2.1.2 Perancangan *Prototype*

Aplikasi pendeteksi plagiarisme ini berbasis dekstop dengan menggunakan bahasa pemrograman Java. Sistem ini akan memproses masukan dari user, yaitu dokumen asli dan dokumen uji. Serta user diberi opsi untuk menggunakan *stemming* atau tidak. Setelah data dimasukkan oleh user, aplikasi akan memproses lalu menampilkan informasi-informasi dari dokumen tersebut. Hasil yang diperoleh setelah proses *string matching* akan ditampilkan pada output hasil. Rancangan *user interface* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Perancangan User Interface

Sumber : [Rancangan]

Berdasarkan pada Gambar 3.3. dijelaskan sebagai berikut :

1. *TextField* untuk *path* file dokumen asli
2. *TextField* untuk *path* file dokumen uji
3. *Checklist* proses *stemming*
4. *Button browse* untuk memilih dokumen asli
5. *Button browse* untuk memilih dokumen uji
6. *TextField* input nilai *threshold* struktural kalimat
7. *TextField* input nilai *threshold internal similarity*
8. *Button* untuk memproses perhitungan
9. *Table* dokumen Asli
10. *Table* dokumen uji dan hasil perhitungan nilai *similarity*

3.2.2 Kebutuhan Data

Data yang digunakan untuk diolah oleh perangkat lunak ini adalah data *stoplist/stopword* berisi kata-kata yang kurang memliki makna. *Stopword* ini digunakan untuk memfilter atau menyaring kata-kata pada proses *filtering*. Data *stopword* yang digunakan adalah data dari jurnal Fadillah Z Tala yang berjudul "*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia".*

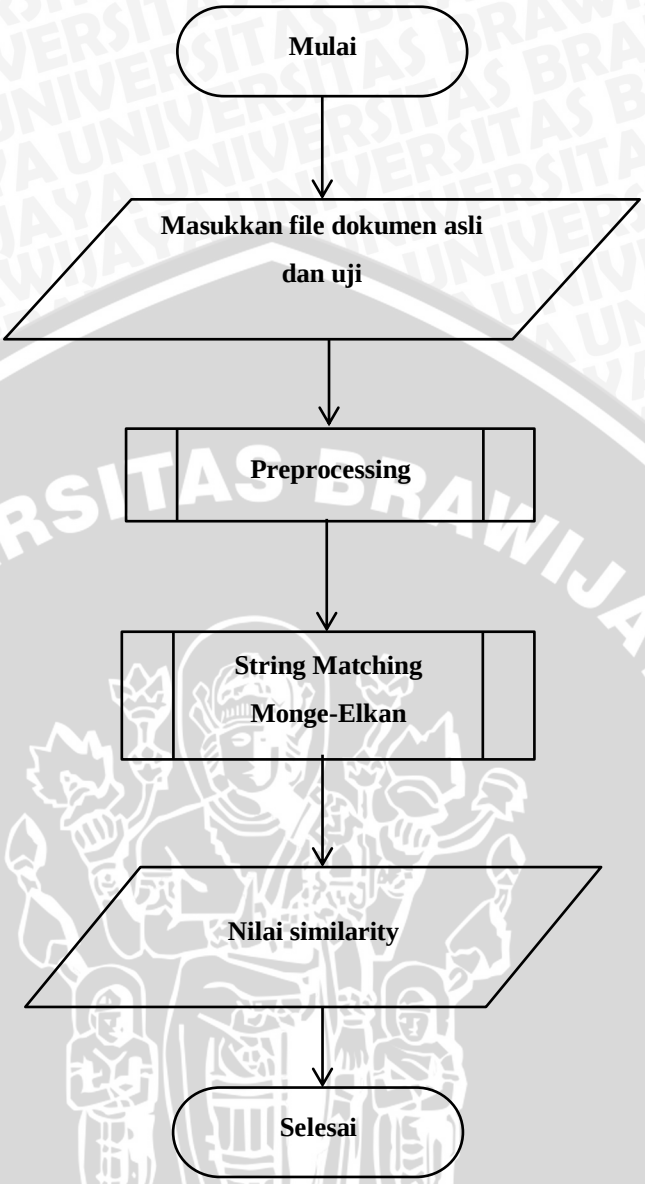
3.2.3 Kebutuhan Fungsional

Fungsi – Fungsi yang dimiliki oleh perangkat lunak ini adalah :

1. Program harus mampu melakukan proses *preprocessing* (*folding, tokenizing, filtering*) dan menghilangkan kata *stopwords* dan melakukan proses *stemming.*
2. Program harus mampu melakukan proses pembandingan dalam setiap tokens menggunakan algoritma Monge-Elkan distance.
3. Program dapat membandingkan perbedaan penggunaan *stemming* atau tidak.

3.3 *Flowchart* Sistem

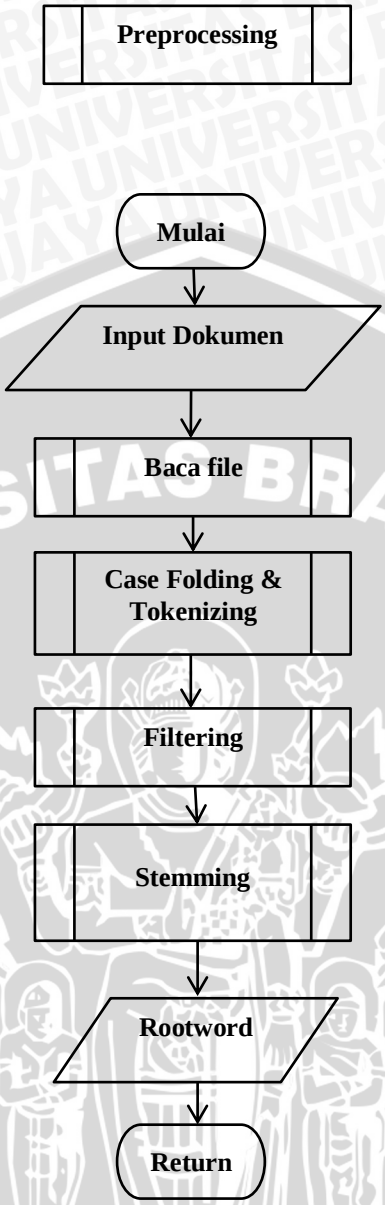
Deteksi plagiarisme pada penelitian ini secara garis besar memiliki dua tahap proses. Tahap pertama adalah tahap *preprocessing* dan tahap kedua adalah tahap *string matching*. Penggunaan *preprocessing* disini bertujuan untuk mengurangi *noise* pada dokumen seperti melakukan perubahan string menjadi huruf kecil, penghilangan simbol-simbol, dan penghilangan kata-kata yang termasuk stopword. Hasil dari *preprocessing* tersebut yakni *token*/kata akan diproses menggunakan pencocokan kata menggunakan algoritma Monge-Elkan distance. Dengan didapatkan nilai dari proses algoritma Monge-Elkan akan diketahui nilai prosentase kesamaan dari dokumen yang dibandingkan.



Gambar 3.4 *Flowchart* Arsitektur Sistem

Sumber : [Rancangan]

Pada proses perhitungan *string mathcing* menggunakan Monge Elkan *distance*, Algoritma Jaro-Winkler digunakan sebagai algoritma *internal similarity*. Setiap hasil dari Jaro – Winkler *distance* akan dibandingkan secara rekursif dengan menggunakan algorima Monge Elkan *distance*. Hasil perhitungan dari Monge Elkan *distance* selanjutnya akan dihitung nilai rata-rata dari nilai *similarity* setiap kalimat.

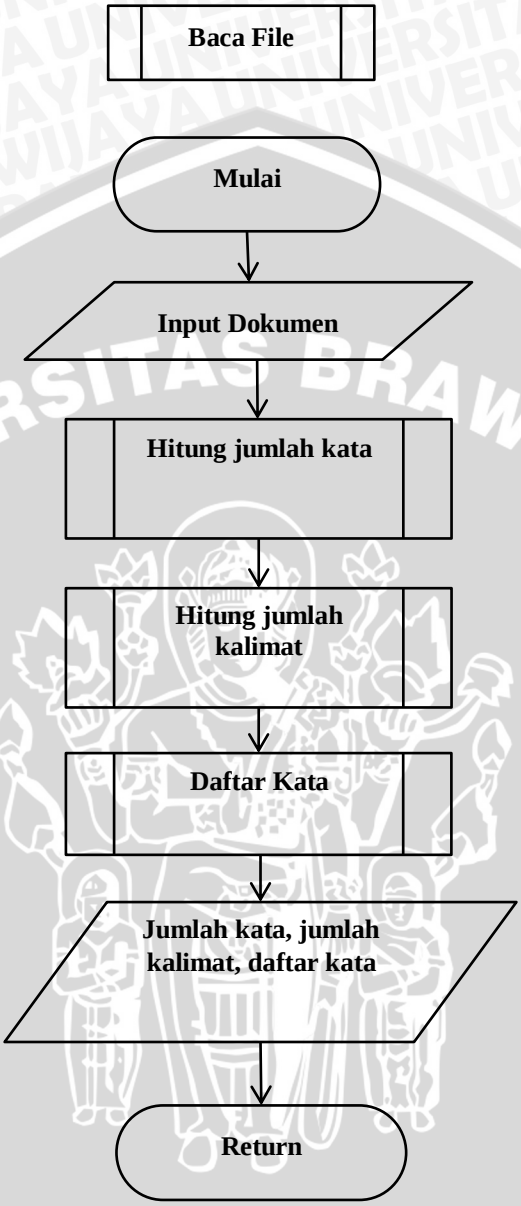


Gambar 3.5 *Flowchart Preprocessing*

Sumber : [Rancangan]

Proses Baca File adalah proses pembacaan file dari inputan user yang akan diproses pada aplikasi ini. Pada tahap ini terdapat suatu fungsi untuk mengambil informasi dari file tersebut. Adapun fungsi-fungsi tersebut yaitu fungsi penghitungan jumlah kata dan jumlah kalimat. Perhitungan jumlah kalimat dilakukan dengan menggunakan karakter titik '.' Sebagai *delimiter*, sedangkan

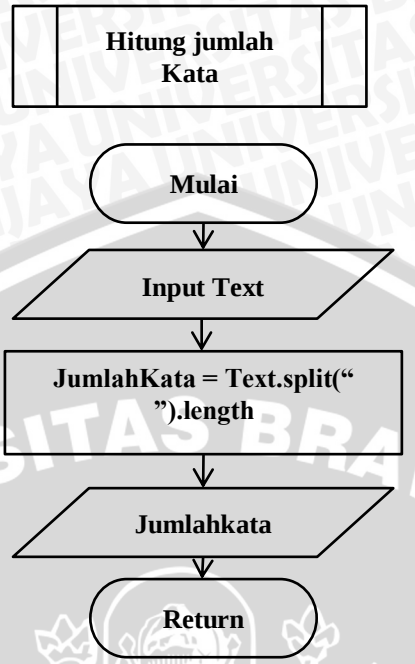
untuk perhitungan jumlah kata digunakan delimiter karakter spasi. *Flowchart* Baca File dapat dilihat pada Gambar 3.6.



Gambar 3.6 *Flowchart* Baca file

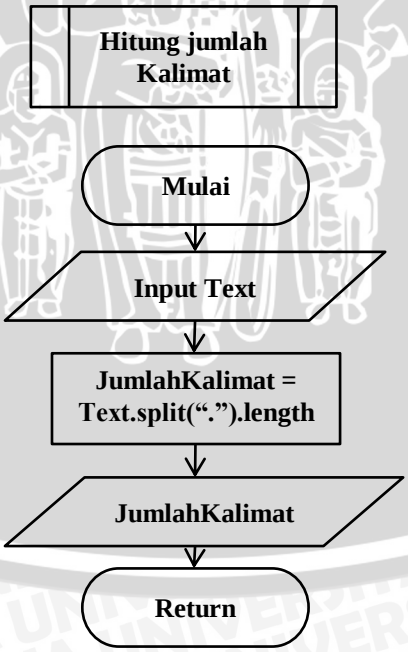
Sumber : [Rancangan]

Penggunaan metode hitung jumlah kata dan jumlah kalimat dapat dilihat pada Gambar 3.7 dan 3.8



Gambar 3.7 *Flowchart* hitung jumlah kata

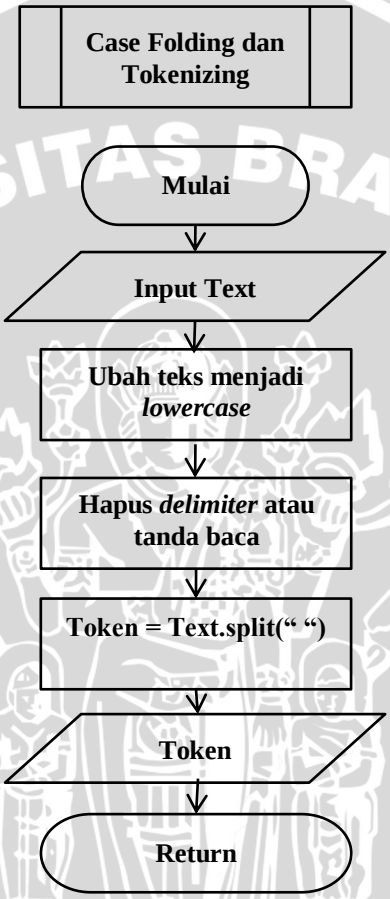
Sumber : [Rancangan]



Gambar 3.8 *Flowchart* hitung jumlah kalimat

Sumber : [Rancangan]

Setelah sistem mendapatkan informasi mengenai dokumen yang diiunputkan user, selanjutnya akan dilakukan proses yang pertama pada *preprocessing* yaitu *casefolding* dan *tokenizing*. *Case folding* adalah merubah kata menjadi huruf kecil. Pada proses *tokenizing* akan dihasilkan token/kata yang berbentuk array. Proses *casefolding* dan *tokenizing* dapat dilihat pada Gambar 3.9.

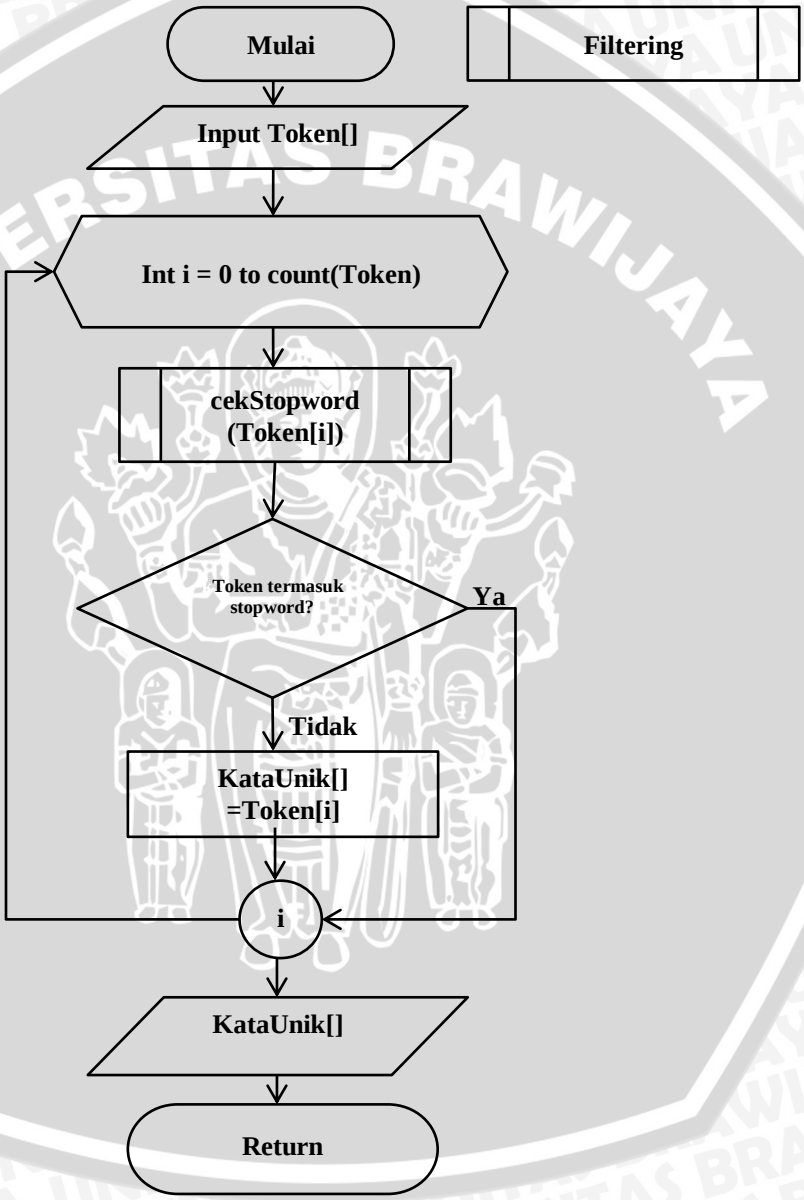


Gambar 3.9 *Flowchart Case Folding* dan *Tokenizing*

Sumber : [Rancangan]

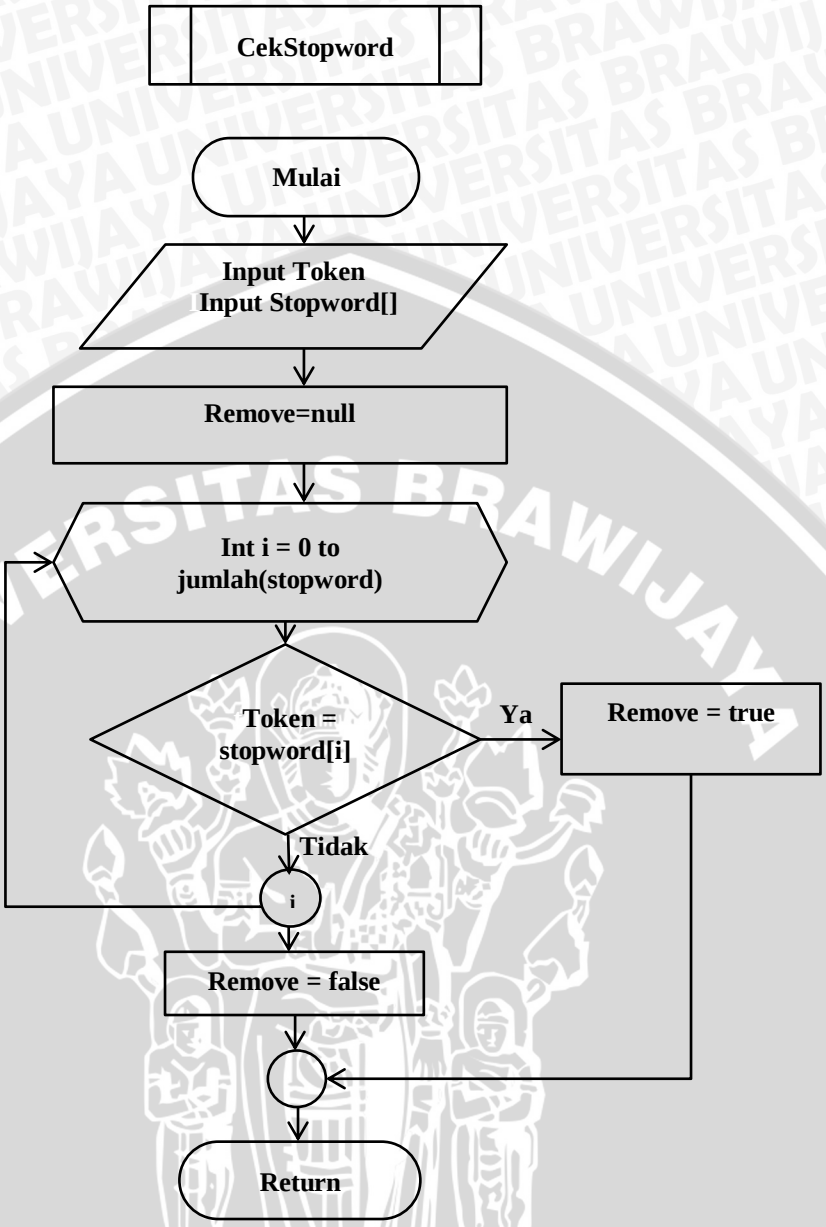
Delimiter yang digunakan pada aplikasi ini adalah semua tanda baca, diantaranya yaitu '.', ',', '"', '"', '-', '/', '{', '}', '+', '_', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '?', '>', '<', '[', ']', '|', '`', '~', ';', ':', '=', '\\' , "\n","\r", untuk mempermudahkan penulisan delimiter yang dipakai adalah karakter selain huruf,angka, dan “.”, penggunaan karakter titik ‘.’ bertujuan untuk memisahkan text menjadi bentuk kalimat. Setelah dilakukan proses *case folding* dan *tokenizing*, selanjutnya

dilakukan proses *filtering* yaitu pembuangan kata yang kurang penting. Adapun kata tersebut seperti "dan", "atau, "di", "adalah', dan seterusnya. Kata – kata kurang penting tersebut disebut *stoplist/stopword*. Pembuangan kata – kata kurang penting ini dilakukan dengan cara menyamakan kata pada *stoplist*. Jika terdapat kata yang sama maka kata tersebut akan dihilangkan. Proses *filtering* dapat dilihat pada Gambar 3.10.



Gambar 3.10 *Flowchart filtering*

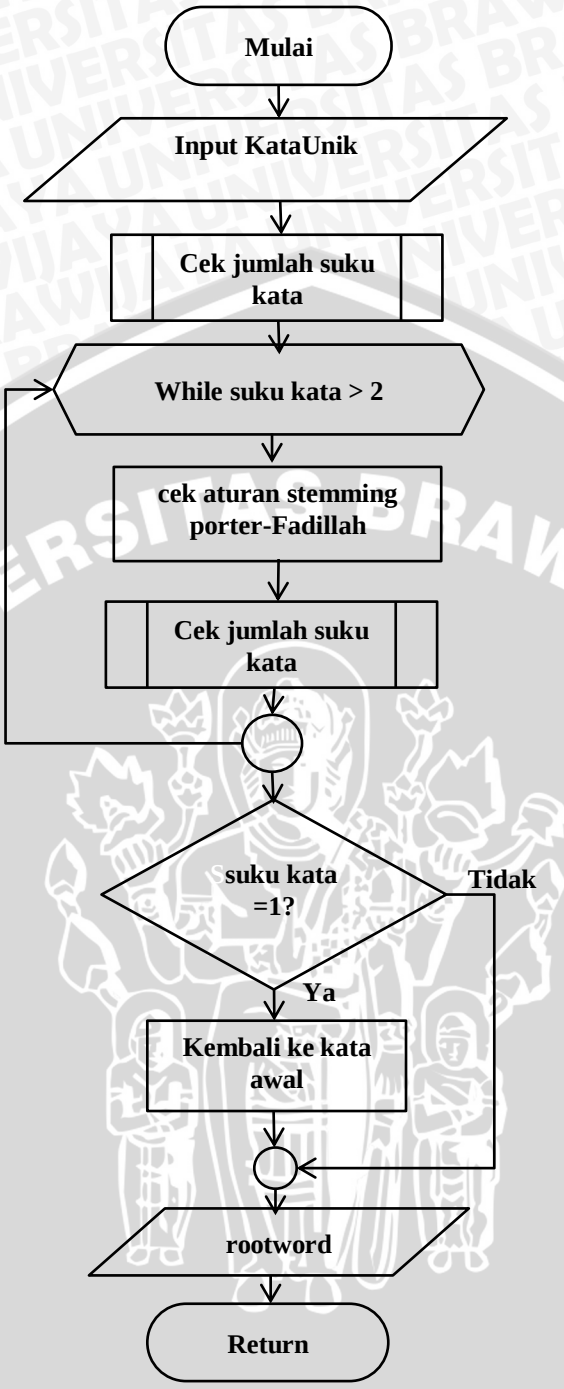
Sumber : [Rancangan]



Gambar 3.11 *Flowchart* Cek *Stopword*

Sumber : [Rancangan]

Setelah proses *filtering*, selanjutnya dilakukan proses stemming. Proses stemming adalah penghilangan imbuhan atau awalan pada suatu kata agar menjadi kata dasar atau *rootword*. Algoritma *stemming* yang digunakan adalah algoritm porter yang sudah dimodifikasi untuk *stemming* berbahasa indonesia oleh Fadillah Z Tala.

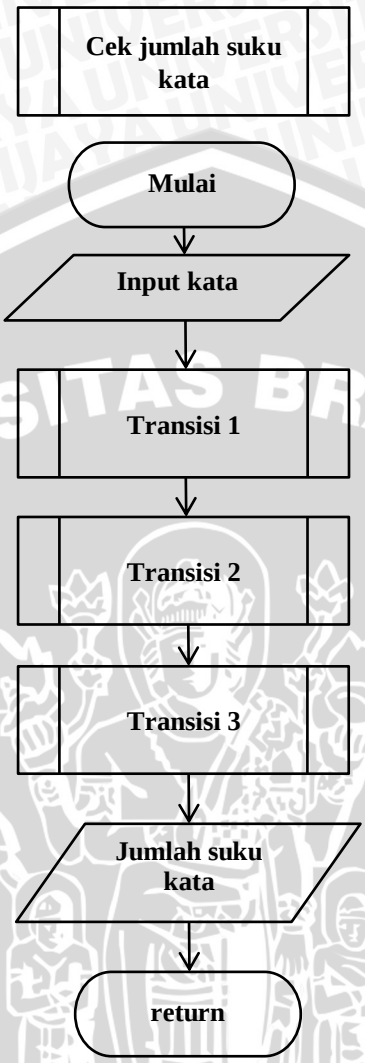


Gambar 3.12 *Flowchart Stemming*

Sumber : [Rancangan]

Cek aturan *stemming* Porter-Fadillah dilakukan dengan cara penghilangan awalan ataupun akhiran pada suatu kata. Proses yang dilakukan sesuai dengan Gambar 2.6 pada bab II. Penggunaan cek suku kata untuk pengecekan jika suku

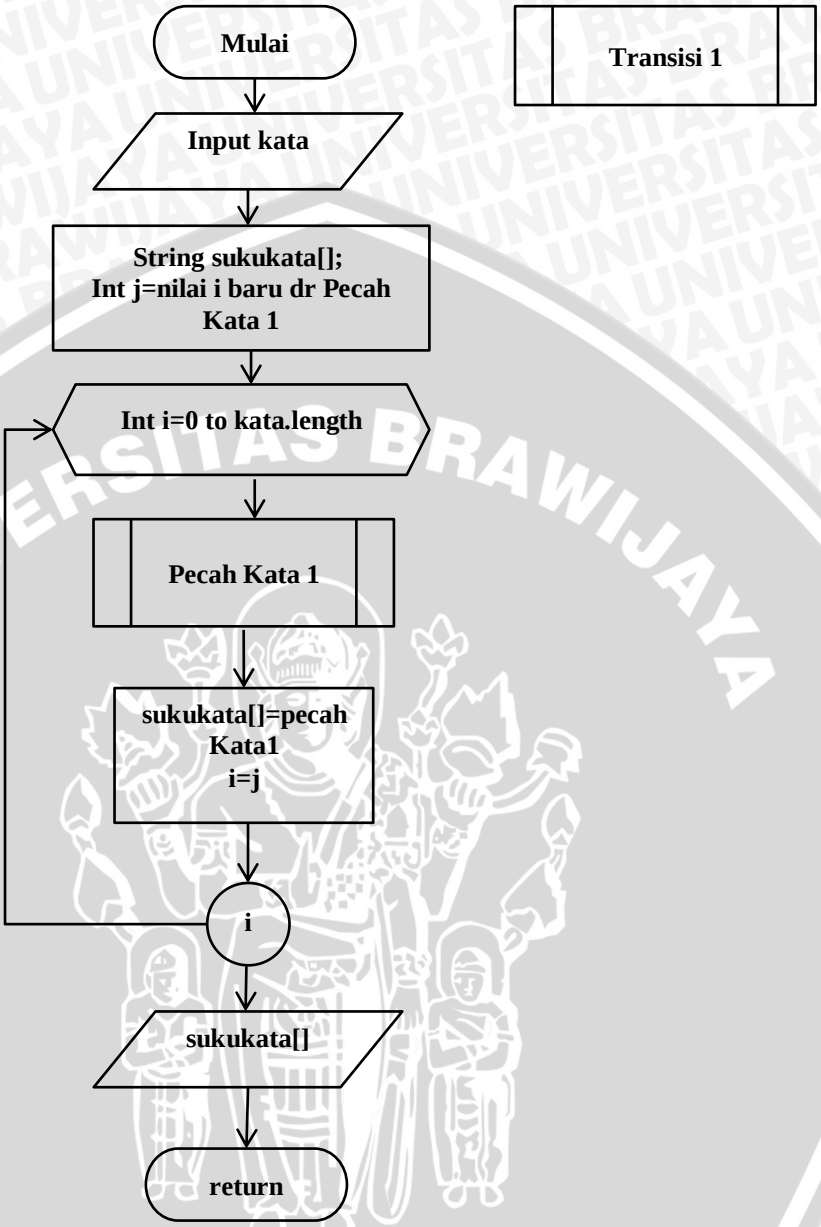
kata sudah mencapai dua, maka proses akan berhenti dan kata tersebut termasuk kata dasar. Cek suku kata ditunjukan pada Gambar 3.13.



Gambar 3.13 *Flowchart* Cek Jumlah Suku Kata

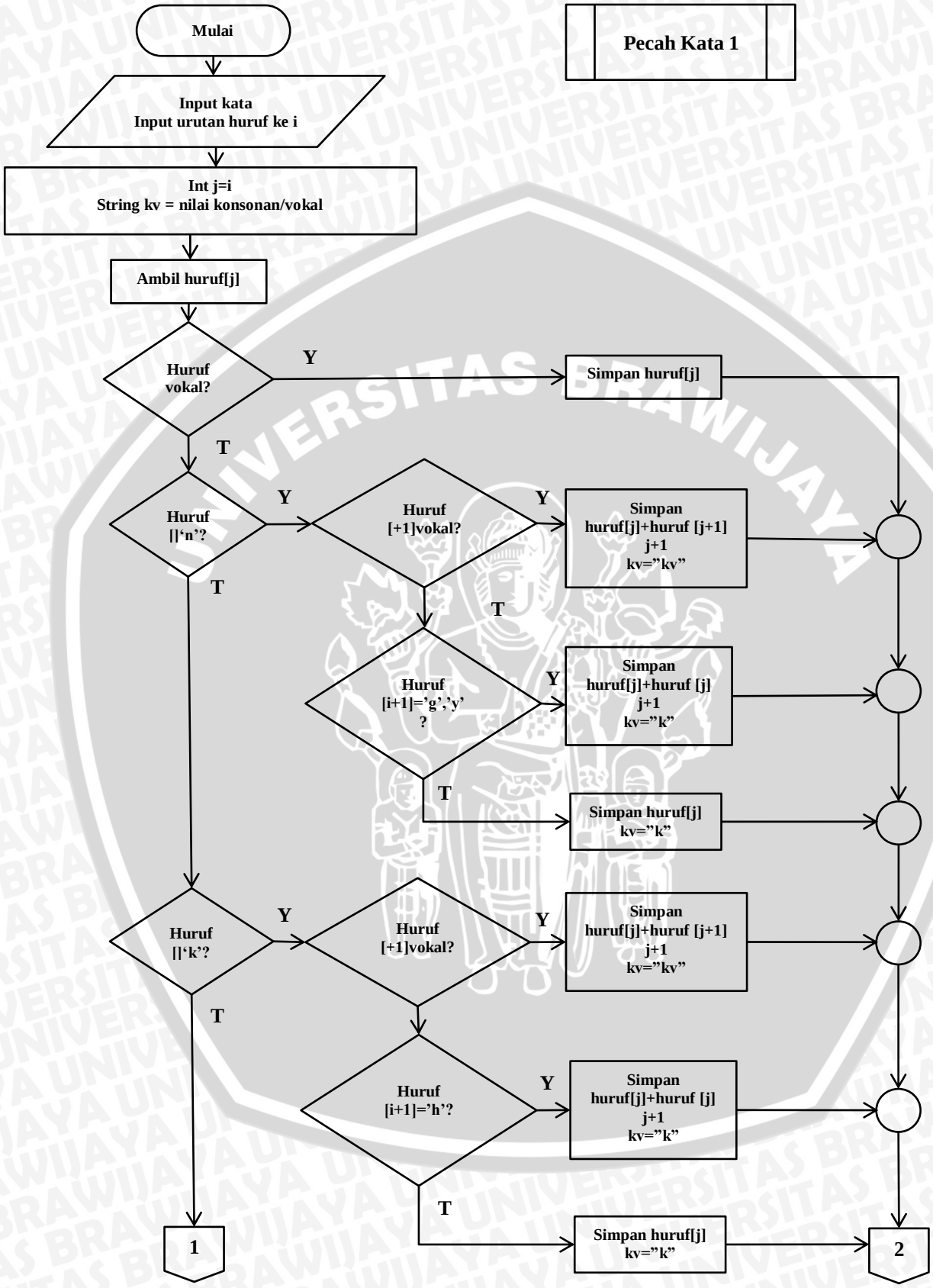
Sumber : [Rancangan]

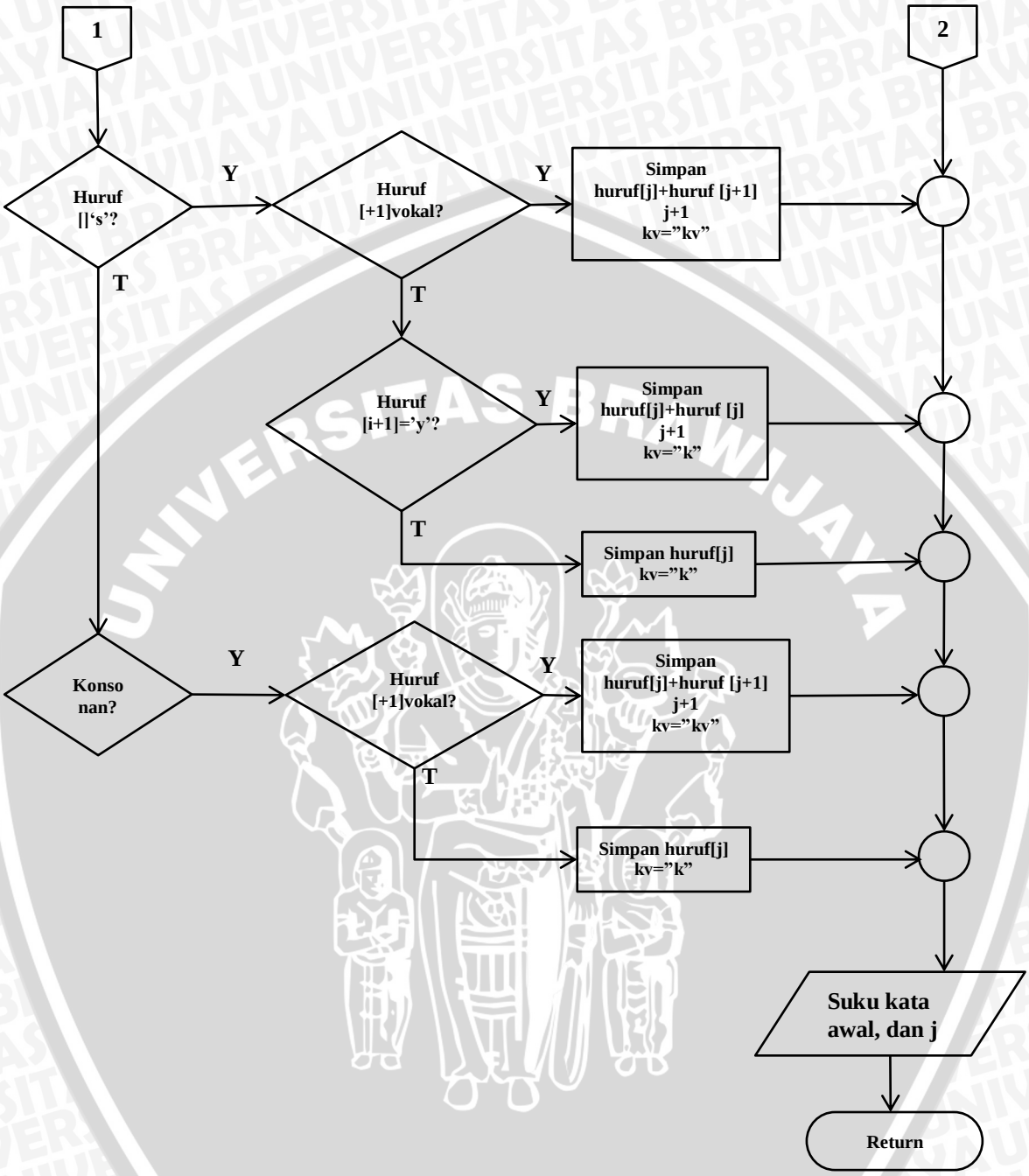
Setiap transisi akan menghasilkan suatu output dengan pola konsonan vokal yang nantinya akan diproses ke transisi lain. Hasil dari ketiga transisi diatas adalah pola pemenggalan suku kata yang sudah dijelaskan pada Bab II.



Gambar 3.14 *Flowchart* Transisi 1

Sumber : [Rancangan]

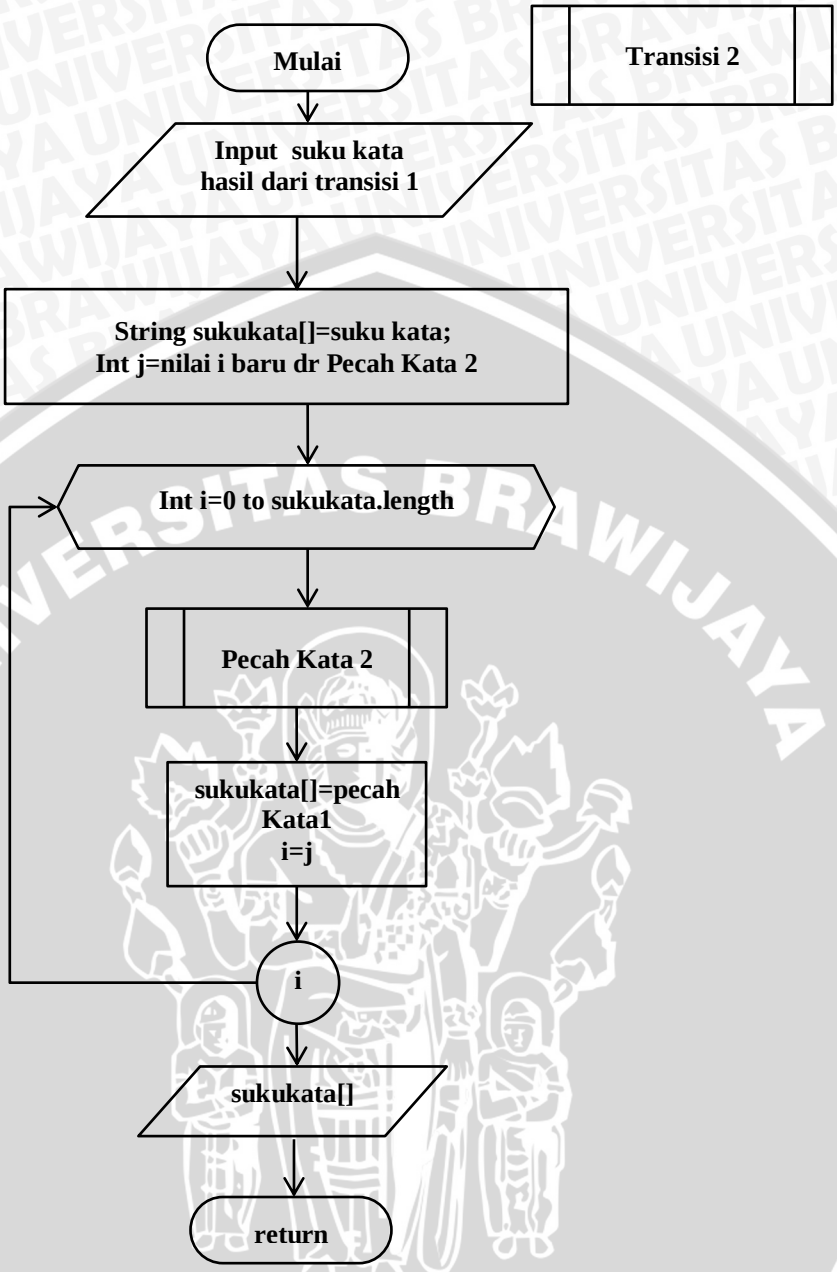

Mulai
Input kata
Input urutan huruf ke i
Int j=i
String kv = nilai konsonan/vokal
Ambil huruf[j]
Huruf vokal?
Y
Simpan huruf[j]
T
Huruf [|'n'?
Y
Huruf [+1]vokal?
Y
Simpan huruf[j]+huruf [j+1] j+1 kv="kv"
T
Huruf [i+1]='g','y' ?
Y
Simpan huruf[j]+huruf [j] j+1 kv="k"
T
Simpan huruf[j] kv="k"
T
Huruf [|'k'?
Y
Huruf [+1]vokal?
Y
Simpan huruf[j]+huruf [j+1] j+1 kv="kv"
T
Huruf [i+1]='h'?
Y
Simpan huruf[j]+huruf [j] j+1 kv="k"
T
Simpan huruf[j] kv="k"
1
2
Pecah Kata 1



Gambar 3.15 *Flowchart* Pecah Kata 1

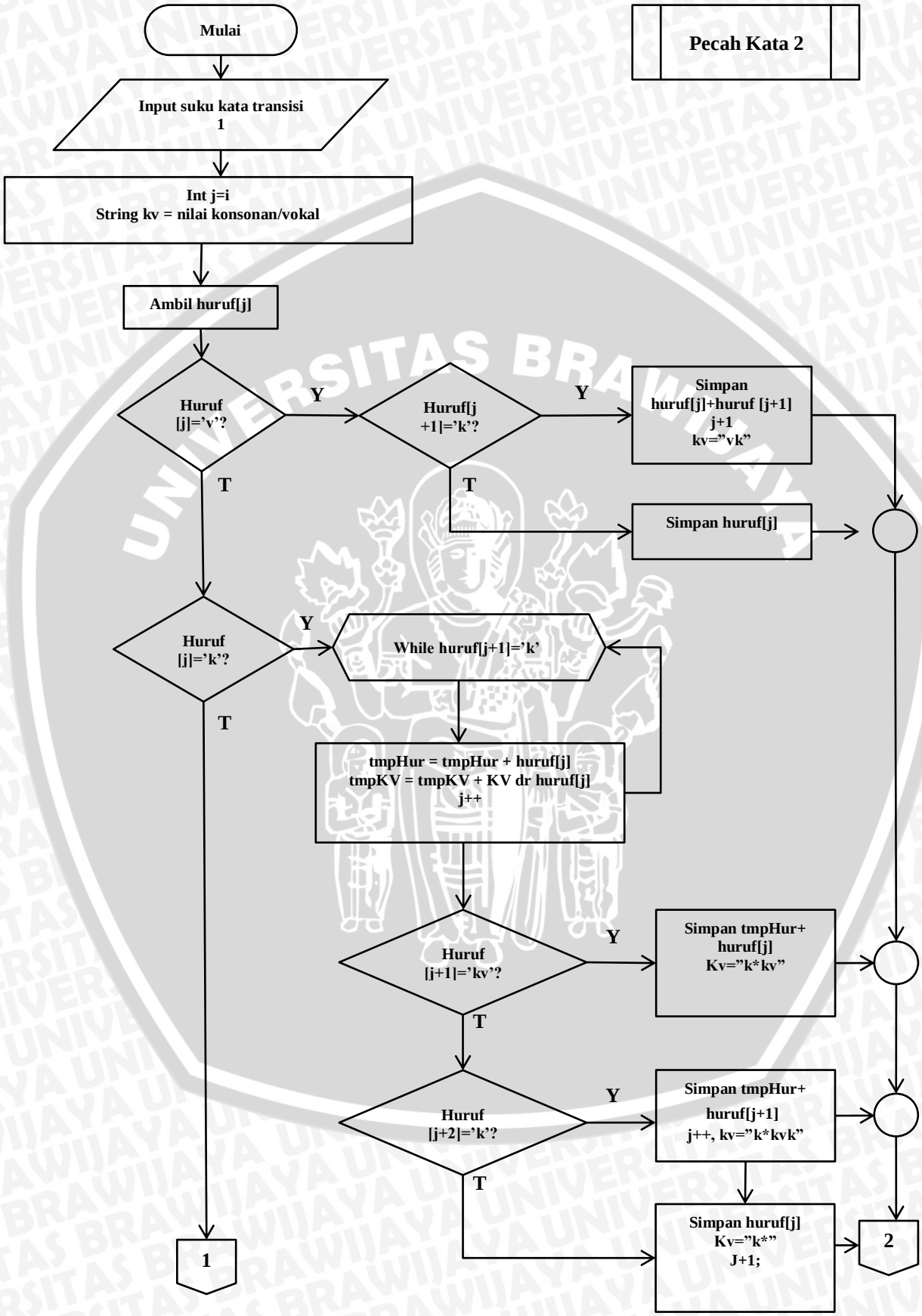
Sumber : [Rancangan]

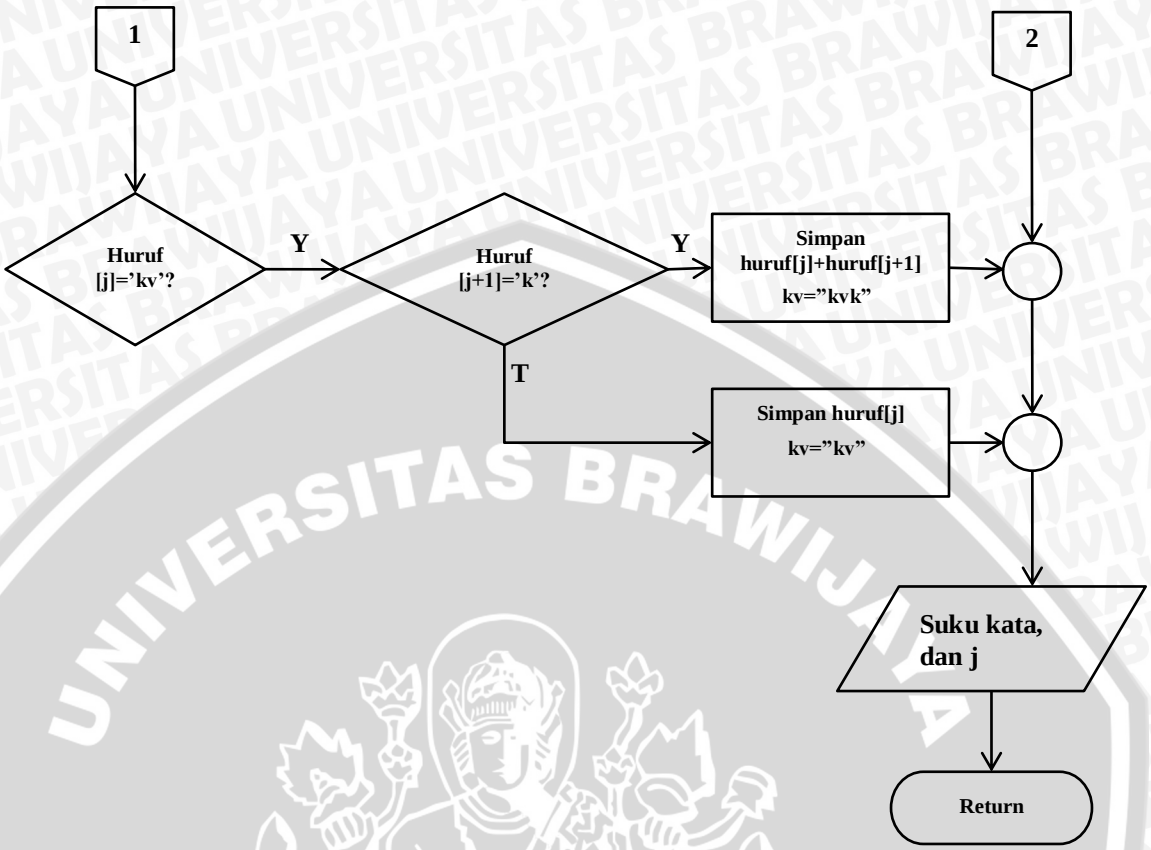
Hasil dari keluaran dari transisi 1 adalah huruf array dengan pola vokal/konsonan V, K, atau KV. Selanjunya hasil dari transisi ini akan diproses pada proses transisi 2. Transisi 2 dapat dilihat pada Gambar 3.16.



Gambar 3.16 *Flowchart* Transisi 2

Sumber : [Rancangan]

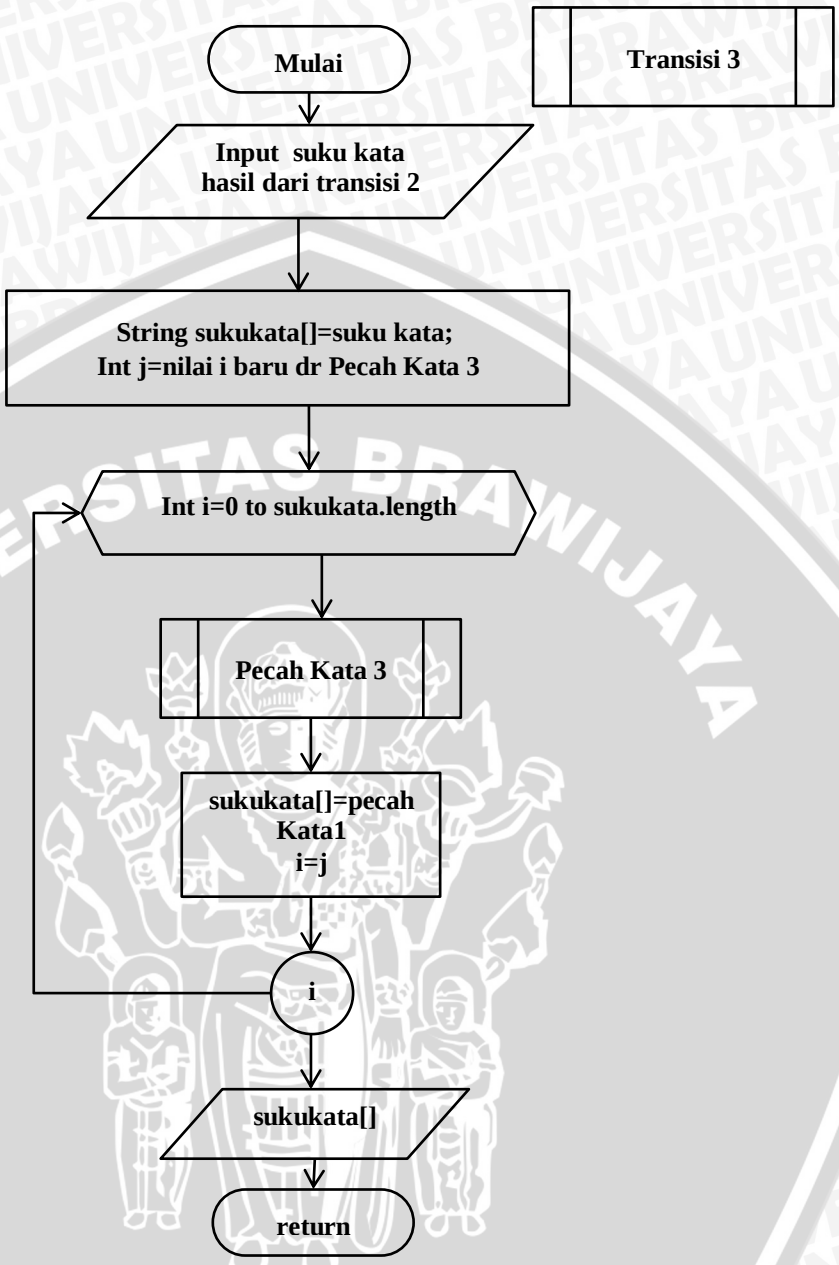

Mulai
Input suku kata transisi 1
Int j=i
String kv = nilai konsonan/vokal
Ambil huruf[j]
Huruf [j]='v'?
Y
T
Huruf [j+1]='k'?
Y
T
Simpan huruf[j]+huruf [j+1] j+1 kv="vk"
Simpan huruf[j]
Huruf [j]='k'?
Y
T
While huruf[j+1]='k'
tmpHur = tmpHur + huruf[j]
tmpKV = tmpKV + KV dr huruf[j]
j++
Huruf [j+1]='kv'?
Y
T
Simpan tmpHur+ huruf[j] Kv="k*kv"
Huruf [j+2]='k'?
Y
T
Simpan tmpHur+ huruf[j+1] j++, kv="k*kvk"
Simpan huruf[j] Kv="k*" J+1;
1
2
Pecah Kata 2



Gambar 3.17 *Flowchart* Pecah Kata 2

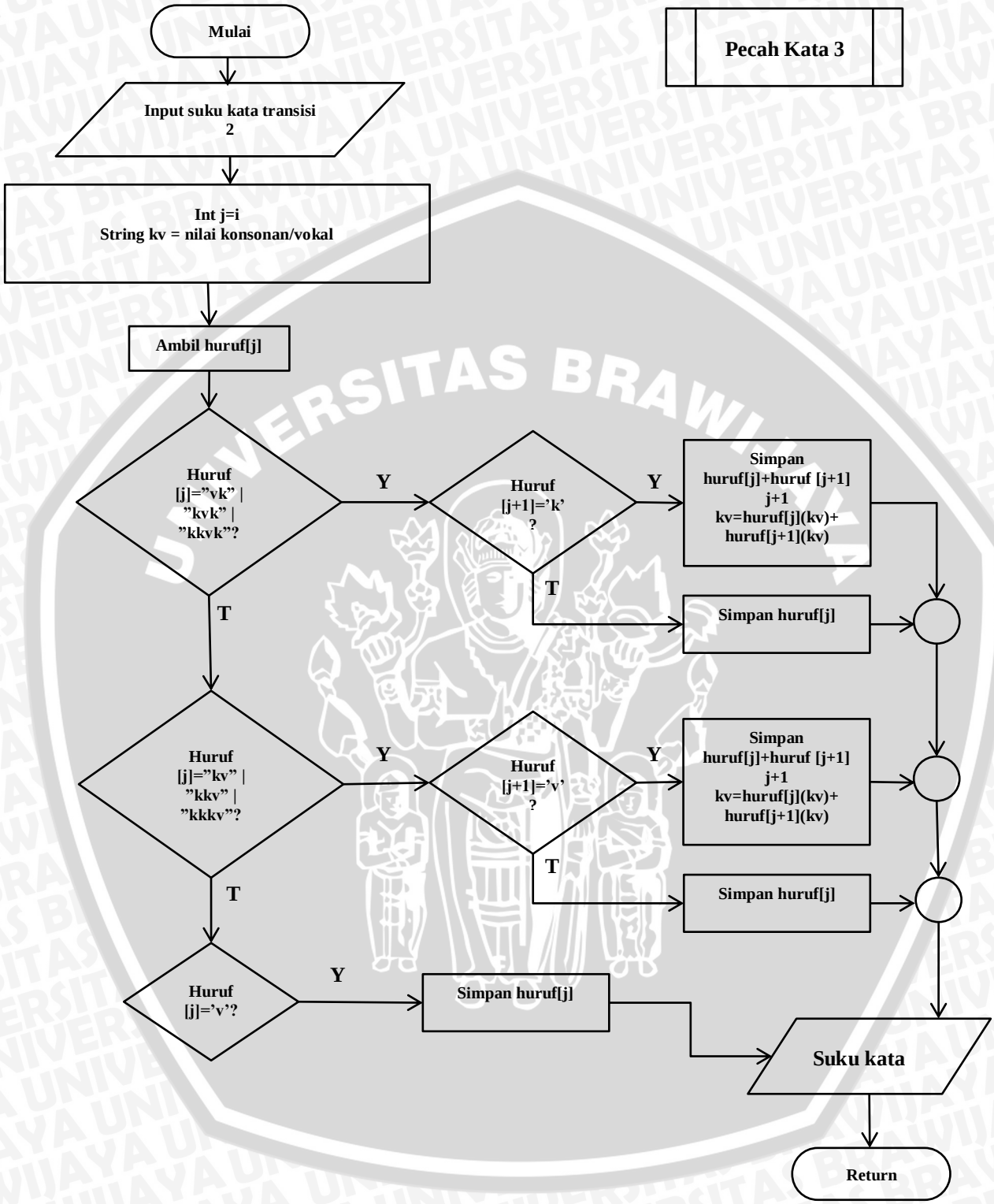
Sumber : [Rancangan]

Keluaran dari proses transisi 2 berupa suku kata yang memiliki pola konsonan/kata VK, KKV atau KKKV, KKVK atau KKKVK, dan KVK. Transisi 3 perlu dilakukan untuk mengenali pola VKK, KVKK, dan KKVKK. Berikut adalah gambar untuk transisi 3.



Gambar 3.18 *Flowchart* Transisi 3

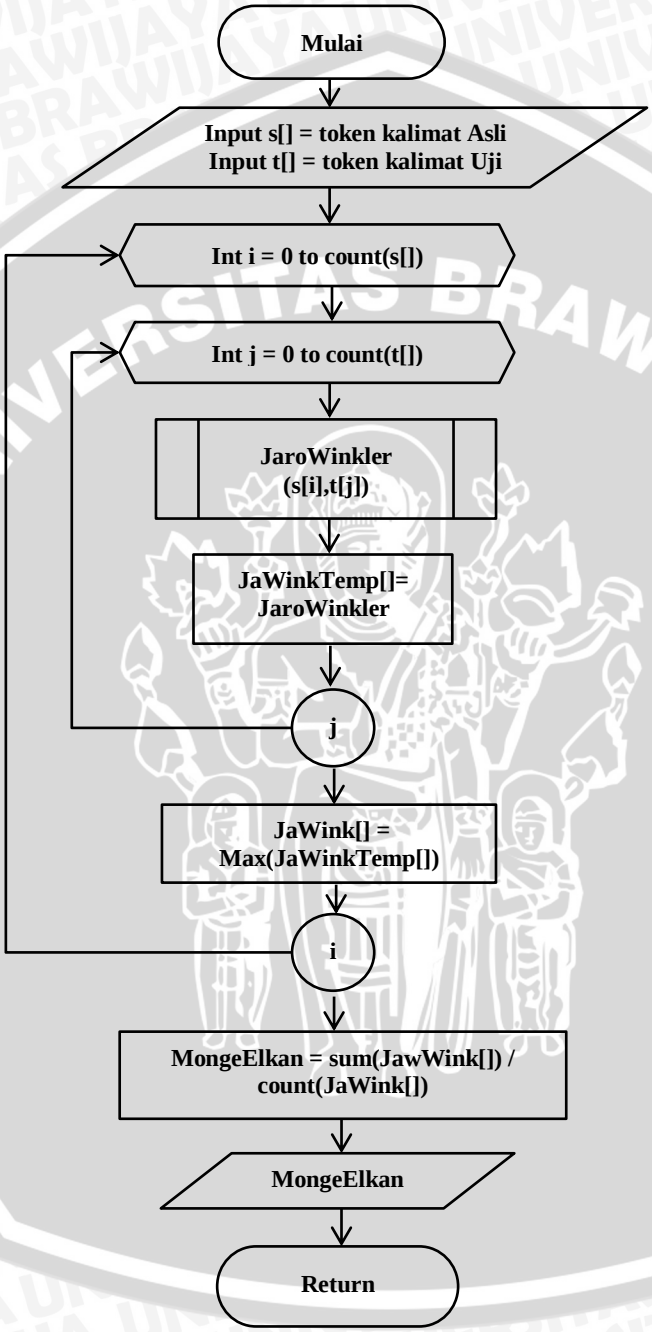
Sumber : [Rancangan]



Gambar 3.19 *Flowchart* Pecah Kata 3

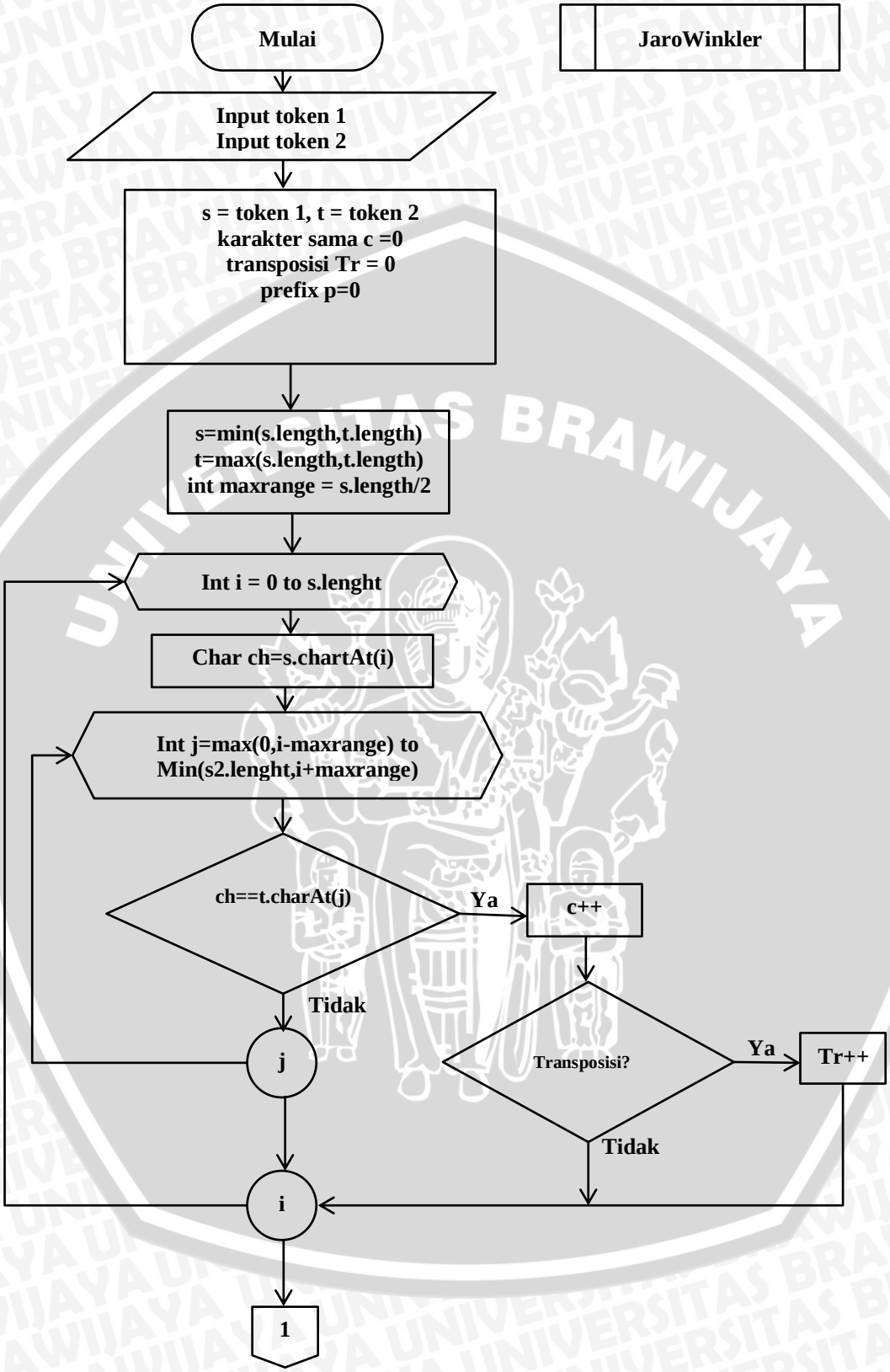
Sumber : [Rancangan]

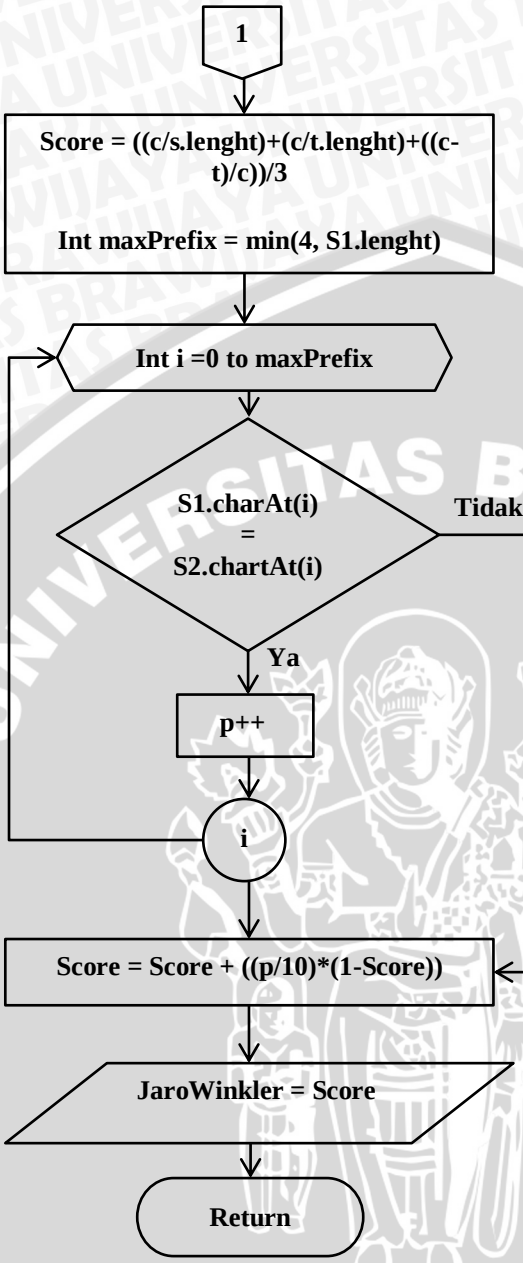
Setelah proses *preprocessing*, selanjutnya dilakukan perhitungan pencocokan string dari dokumen asli dengan dokumen uji menggunakan algoritma Monge-Elkan *distance*. Berikut adalah *flowchart* algoritma Monge-Elkan *distance* (Gambar 3.14) dan Jaro-Winkler *distance* (Gambar 3.15).



Gambar 3.20 *Flowchart* Monge-Elkan *distance*

Sumber : [Rancangan]





Gambar 3.21 *Flowchart* Jaro-Winkler *distance*

Sumber : [Rancangan]

Pada program ini pencocokan string yang diterapkan adalah per kalimat. kalimat. Token yang didapat dari hasil filtering selanjutnya akan dilakukan proses *string matching* menggunakan algoritma Monge-Elkan *distance*.

3.4 Rancangan Sistem

3.4.1 Bahan Pengujian

Data yang diuji pada penelitian ini berupa dokumen teks yang berformat .txt. Penggunaan data diambil dari artikel maupun dari buku studi literatur. Dokumen yang diuji akan diubah sedemikan rupa untuk menguji berapa tingkat plagiarisme dari dokumen uji tersebut.

3.4.2 Tujuan Pengujian

Tujuan dari pengujian sistem untuk mendeteksi plagiarisme ini adalah sebagai berikut :

1. Menganalisa hasil *similarity* yang dihasilkan menggunakan algoritma Monge-Elkan *distance*
2. Menganalisa pengaruh *stemming* Porter-Fadillah terhadap nilai *similarity* yang dihasilkan oleh algoritma Monge-Elkan *distance*.
3. Menganalisa pengaruh *threshold* terhadap nilai tingkat kesamaan dokumen.
4. Menganalisa prosentasi *error* untuk mengetahui galat pada sistem.

3.4.3 Perancangan Tabel Hasil Percobaan

Dengan membandingkan nilai *simalirity* dan wkatu proses antara algoritma Monge-Elkan *distance* dan Jaro-Winkler *distance*, dapat ditentukan seberapa besar pengaruh penggunaan algoritma Monge-Elkan *distance* dengan *internal similarity* Jaro-Winkler *distance*. Berikut adalah rancangan tabel perhitungan.

Tabel 3.1 Perancangan informasi dokumen

Kode	Jumlah Kalimat	Jumlah Kata

Sumber : [Rancangan]

Tabel 3.2 Perancangan pengujian algoritma

Dok. Asli	Dok. Uji	Threshold Struktural	Threshold int. similarity		Monge-Elkan	
				S(%) Non Stemming	S(%) Stemmming	Kategori

Sumber : [Rancangan]

Tabel 3.3 Perancangan pengujian *threshold*

Threshold	Similarity (%)						
	A-20	A-40	A-60	A-TK	A-AP	A-TG	A-KB

Sumber : [Rancangan]

3.4.4 Perhitungan Nilai Kemiripan Teks (*Similarity*)

Nilai *similarity* adalah kemiripan antar kedua dokumen, dokumen asli dan uji. *Stemming* dilakukan untuk mengetahui berapa tingkat nilai *similarity* dengan menggunakan *stemming* atau tanpa *stemming*.

3.4.5 Perancangan Dokumen Uji

Dokumen uji yang digunakan pada penelitian ini ada beberapa jenis dokumen, ketentuan dari dokumen uji yang digunakan adalah sebagai berikut :

1. Dokumen uji dengan pemotongan 20 % kata dari jumlah seluruh kata pada dokumen asli.
2. Dokumen uji dengan pemotongan 40 % kata dari jumlah seluruh kata pada dokumen asli.
3. Dokumen uji dengan pemotongan 60 % kata dari jumlah seluruh kata pada dokumen asli.
4. Dokumen uji dengan penukaran susunan kalimat dari dokumen asli (TK).
5. Dokumen uji dengan pengubahan kata kerja aktif menjadi pasif, begitu pula sebaliknya (AP).
6. Dokumen uji dengan kesalahan ejaan kata (TG)

7. Dokumen uji dengan pemotongan kata 20% dari seluruh kata, penukaran susunan kalimat, dan kesalahan ejaan kata (KB).

Dokumen uji dibuat untuk mengecek sistem yang telah dibuat apakah sudah sesuai. Pemotongan kata dilakukan dengan cara random sesuai dengan jumlah persen pemotongan kata. Berikut ini adalah contoh data dokumen asli dan beberapa dokumen uji:

a. Dokumen asli (A-00)

Kegiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI. Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini.

b. Dokumen uji 20 % pemotongan jumlah kata (A-20)

Kegiatan ini oleh 300 orang yang didominasi perempuan putih dan bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga oleh Wakil Presiden RI Boediono beserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI. Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini.

c. Dokumen uji 40% pemotongan jumlah kata (A-40)

Kegiatan dihadiri oleh 300 yang putih selendang bercorak di hall utama Gedung Induk kanan, Kepresidenan Bogor, Jawa. ini juga Wakil RI Boediono beserta dan muslimah ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, HMI. Bambang Yudhoyono dan Agama RI Ali dalam acara ini.

d. Dokumen uji 60% pemotongan jumlah kata (A-60)

Mereka tapi juga. kita Hari Muslimah Nasional bertepatan Hari Perempuan kata. dihadiri yang didominasi berjilbab bercorak di hall Induk kanan, Kepresidenan Bogor,. Kegiatan ini juga oleh Boediono akan

Muslimat Aisyiyah Muhammadiyah, MUI, dan. dan Agama RI Suryadharma Ali dijadwalkan.

e. **Dokumen uji penukaran susunan kalimat (A-TK)**

Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini. Kegiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI.

f. **Dokumen uji dengan perubahan kata kerja menjadi aktif atau sebaliknya (A-AP)**

300 orang menghadiri kegiatan ini yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI juga menghadiri kegiatan ini. Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini.

g. **Dokumen uji dengan kesalahan ejaan kata (A-TG)**

Kwgiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utwma Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akwn ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI. Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini.

h. **Dokumen uji dengan pemotongan kata 20% dari seluruh kata, penukaran susunan kalimat, dan kesalahan ejaan kata (A-KB)**

Sebelumnya, Preswden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini. Kegiatan dihadiri oleh 300 orang didominasi perempuan putih dan bercorak di hall Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kwgiatan ini juga oleh Wakil Presiden RI Boediono bwserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI.

3.4.6 Perhitungan Manual

Berikut adalah contoh perhitungan manual dari program ini :

Dokumen asli (A-00)

Kegiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI. Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini.

Dokumen uji (U-20)

Kegiatan ini oleh 300 orang yang didominasi perempuan putih dan bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga oleh Wakil Presiden RI Boediono beserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI. Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini.

Hasil parsing dokumen

Dokumen asli (A-00)

1. Kegiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat

2. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI
3. Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini

Dokumen uji (A-20)

1. Kegiatan ini oleh 300 orang yang didominasi perempuan putih dan bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat
2. Kegiatan ini juga oleh Wakil Presiden RI Boediono beserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI
3. Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini

Setelah dilakukan parsing menurut jumlah kalimat, selanjutnya dilakukan perbandingan kata antar kalimat yang dapat disebut *threshold* struktural kalimat. Nilai yang dihasilkan akan dicek apakah sesuai nilai *threshold*. Jika nilai yang didapat lebih dari nilai *threshold*, maka kalimat tersebut akan diproses lebih lanjut untuk menghitung tingkat kemiripan dokumen. Untuk menghitung nilai kesamaan kata antar kalimat digunakan cara membagi jumlah kata yang sama dengan jumlah maksimal kata dari kedua kalimat. Nilai *threshold* yang digunakan pada perhitungan ini yaitu **0.5** yang berarti kesamaan kata pada kalimat berjumlah lebih dari 50%**.**

Kesamaan Struktural Kalimat

A-00 (1) : **Kegiatan ini** dihadiri **oleh 300 orang yang didominasi perempuan** berjilbab **putih dan** selendang biru **bercorak di hall** utama Gedung **Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat**

A-20 (1) : **Kegiatan ini oleh 300 orang yang didominasi perempuan putih dan bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat**

Kesamaan kata = 21/27
= 0,77

A-00 (1) : **Kegiatan ini** dihadiri **oleh** 300 orang **yang** didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat

A-20 (2) : **Kegiatan ini** juga **oleh** Wakil Presiden RI Boediono beserta dan organisasi muslimah **yang** akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI

Kesamaan kata = 4/27
= 0,14

A-00 (1) : Kegiatan **ini** dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat

U-20 (3) : Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam **ini**

Kesamaan kata = 1/27
= 0,03

A-00 (2) : **Kegiatan ini** juga dihadiri **oleh** Wakil Presiden RI Boediono beserta isteri **dan** organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI

A-20 (1) : **Kegiatan ini oleh** 300 orang yang didominasi perempuan putih **dan** bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat

Kesamaan kata = 4/27
= 0,14

A-00 (2) : **Kegiatan ini juga** dihadiri **oleh Wakil Presiden RI Boediono beserta** isteri **dan organisasi muslimah yang akan ikut antara lain Muslimat** NU, Aisyiyah **Muhammadiyah, MUI, ICMI** dan **HMI**

U-20 (2) : **Kegiatan ini juga oleh Wakil Presiden RI Boediono beserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI**

Kesamaan kata = 22/27 = 0,81

A-00 (2) : Kegiatan **ini** juga dihadiri oleh Wakil **Presiden RI** Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI

U-20 (3) : Sebelumnya, **Presiden** Bambang Yudhoyono Menteri Agama **RI** Suryadharma Ali dijadwalkan hadir dalam **ini**

Kesamaan kata = 3/27
= 0,11

A-00 (3) : Sebelumnya, Presiden Susilo Bambang Yudhoyono **dan** Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara **ini**

U-20 (1) : Kegiatan **ini** oleh 300 orang yang didominasi perempuan putih **dan** bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat

Kesamaan kata = 2/21
= 0,09

A-00 (3) : Sebelumnya, Presiden Susilo Bambang Yudhoyono **dan** Menteri Agama **RI** Suryadharma Ali dijadwalkan hadir dalam acara **ini**

U-20 (2) : Kegiatan **ini** juga oleh Wakil Presiden **RI** Boediono beserta **dan** organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI

Kesamaan kata = 4/22
= 0,18

A-00 (3) : **Sebelumnya, Presiden** Susilo **Bambang Yudhoyono** dan **Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam** acara **ini**

U-20 (3) **: Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini**

Kesamaan kata = 13/16
= 0,81

Pada perhitungan kesamaaan kata diatas didapatkan nilai kesamaan kata yang mepresentasikan seberapa besar kesamaan kata antar dua kalimat. Jika *threshold* yang digunakan adalah 0,5 maka kalimat yang akan diproses yaitu kalimat yang memiliki nilai kesamaan kata lebih dari atau sama dengan 0,5.

A-00 (1) : **Kegiatan ini** dihadiri **oleh 300 orang yang didominasi perempuan** berjilbab **putih dan** selendang biru **bercorak di hall** utama Gedung **Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat**

A-20 (1) : **Kegiatan ini oleh 300 orang yang didominasi perempuan putih dan bercorak di hall Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat**

Kesamaan kata = 21/27
= 0,77

A-00 (2) : **Kegiatan ini juga** dihadiri **oleh Wakil Presiden RI Boediono beserta** isteri **dan organisasi muslimah yang akan ikut antara lain Muslimat** NU, Aisyiyah **Muhammadiyah, MUI, ICMI** dan **HMI**

U-20 (2) : **Kegiatan ini juga oleh Wakil Presiden RI Boediono beserta dan organisasi muslimah yang akan ikut antara lain Muslimat Muhammadiyah, MUI, ICMI HMI**

Kesamaan kata = 22/27
= 0,81

A-00 (3) : **Sebelumnya, Presiden** Susilo **Bambang Yudhoyono** dan **Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam** acara **ini**

U-20 (3) **: Sebelumnya, Presiden Bambang Yudhoyono Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam ini**

Kesamaan kata = 13/16
= 0,81

Selanjutnya akan dilakukan proses *filtering* dan proses *stemming*, serta penghilangan kata dilakukan jika terdapat duplikasi kata. Hasil dari proses *filtering* dan *stemming* sebagai berikut :

A-00 (1) : giat hadir 300 orang dominasi perempuan jilbab putih selendang biru corak hall utama gedung induk sayap kanan istana presiden bogor jawa barat

U-20 (1) : giat 300 orang dominasi perempuan putih corak hall induk sayap kanan istana presiden bogor jawa barat

A-00 (2) : giat hadir wakil presiden ri boediono serta isteri organisasi muslimah ikut muslimat nu aisyiyah muhammadiyah mui icmi hmi

U-20 (2) : giat wakil presiden ri boediono serta organisasi muslimah ikut muslimat muhammadiyah mui icmi hmi

A-00 (3) : belum presiden susilo bambang yudhoyono menteri agama ri suryadharma ali jadwal hadir acara

U-20 (3) : belum presiden bambang yudhoyono menteri agama ri suryadharma ali jadwal hadir

Proses *filtering* dan *stemming* menghasilkan token yang berbentuk array yang kemudian akan dilakukan perhitungan menggunakan algoritma Monge-Elkan. Pertama perhitungan dilakukan menggunakan internal *similarity* yaitu algoritma Jaro-Winkler, selanjutnya token dari setiap kalimat akan dibandingkan dengan cara perbandingan rekursif. Perhitungan tersebut dilakukan mulai token "banyak" dengan "banyak", "banyak dengan "teliti", "banyak" dengan dokter, dan seterusnya. Nilai terbesar dari perbandingan tersebut (Jaro-Winkler) akan digunakan pada proses selanjutnya menggunakan algoritma Monge-Elkan. Rumus Jaro-Winkler dapat dilihat pada Rumus 2.2

Jaro-Winkler A-00 (1) : A-20 (1)

giat : giat $= sim_{jaro}(s,t) = \frac{1}{3}\left(\frac{4}{4}+\frac{4}{4}+\frac{4-0}{4}\right) = 1{,}00$

$= sim_{Jw}(s,t) = 1+\frac{4}{10}.(1-1) = 1.00$

giat : 300 $= sim_{jaro}(s,t) = \frac{1}{3}\left(\frac{0}{4}+\frac{0}{3}+\frac{0-0}{0}\right) = 0{,}00$

$= sim_{Jw}(s,t) = 0+\frac{0}{10}.(1-0) = 0{,}00$

giat : orang $= sim_{jaro}(s,t) = \frac{1}{3}\left(\frac{1}{4}+\frac{1}{5}+\frac{1-0}{1}\right) = 0{,}48$

$$= sim_{Jw}(s,t) = 0{,}48 + \frac{0}{10}.(1 - 0{,}48) = 0{,}48$$

giat : dominas $= sim_{jaro}(s,t) = \frac{1}{3}\left(\frac{1}{4} + \frac{1}{7} + \frac{1-0}{1}\right) = 0{,}46$

$$= sim_{Jw}(s,t) = 0{,}46 + \frac{0}{10}.(1 - 0{,}46) = 0{,}46$$

Perhitungan selanjutnya dapat dilihat pada tabel 3.1.

Tabel 3.4 Perhitungan Jaro-Winkler kalimat pertama A-00 (1) : A-20 (1)

Kata	giat	300	orang	dominas	perempuan	putih	corak	hall	induk	sayap	kanan	istana	presiden	bogor	jawa	barat
giat	1.00	0.00	0.48	0.46	0.00	0.47	0.48	0.50	0.48	0.48	0.48	0.64	0.00	0.48	0.50	0.63
hadir	0.47	0.00	0.47	0.56	0.00	0.47	0.43	0.71	0.47	0.47	0.47	0.46	0.44	0.47	0.63	0.60
300.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
orang	0.48	0.00	1.00	0.56	0.47	0.00	0.73	0.48	0.47	0.47	0.60	0.58	0.44	0.60	0.63	0.43
dominas	0.46	0.00	0.56	1.00	0.45	0.45	0.56	0.00	0.45	0.45	0.68	0.75	0.49	0.56	0.46	0.45
rempu	0.00	0.00	0.47	0.45	1.00	0.00	0.47	0.00	0.47	0.47	0.00	0.00	0.55	0.00	0.00	0.47
jilbab	0.61	0.00	0.46	0.54	0.00	0.46	0.46	0.61	0.46	0.46	0.46	0.56	0.43	0.00	0.65	0.46
putih	0.47	0.00	0.00	0.45	0.00	1.00	0.00	0.00	0.47	0.00	0.00	0.46	0.60	0.00	0.00	0.47
selendang	0.46	0.00	0.57	0.55	0.45	0.00	0.45	0.46	0.56	0.45	0.57	0.53	0.60	0.00	0.46	0.45
biru	0.50	0.00	0.48	0.46	0.63	0.47	0.48	0.00	0.63	0.00	0.00	0.47	0.46	0.67	0.00	0.67
corak	0.48	0.00	0.73	0.56	0.47	0.00	1.00	0.48	0.47	0.47	0.47	0.46	0.44	0.60	0.63	0.60
hall	0.50	0.00	0.48	0.00	0.00	0.00	0.48	1.00	0.00	0.48	0.48	0.47	0.00	0.00	0.50	0.48
utama	0.47	0.00	0.60	0.56	0.47	0.60	0.60	0.48	0.00	0.60	0.60	0.70	0.00	0.00	0.63	0.60
gedung	0.52	0.00	0.58	0.54	0.58	0.46	0.00	0.00	0.58	0.00	0.58	0.44	0.63	0.46	0.00	0.00
induk	0.48	0.00	0.47	0.45	0.47	0.47	0.47	0.00	1.00	0.00	0.47	0.51	0.00	0.00	0.00	0.00
sayap	0.48	0.00	0.60	0.45	0.47	0.00	0.60	0.48	0.00	1.00	0.60	0.70	0.00	0.00	0.63	0.60
kanan	0.48	0.00	0.78	0.68	0.00	0.00	0.60	0.48	0.47	0.60	1.00	0.82	0.00	0.00	0.63	0.60
istana	0.64	0.00	0.58	0.75	0.00	0.46	0.46	0.47	0.51	0.70	0.82	1.00	0.53	0.00	0.61	0.59
presiden	0.00	0.00	0.44	0.49	0.55	0.60	0.44	0.00	0.00	0.00	0.00	0.53	1.00	0.00	0.00	0.44
bogor	0.48	0.00	0.60	0.56	0.00	0.00	0.73	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.64
jawa	0.67	0.00	0.63	0.46	0.00	0.00	0.63	0.67	0.00	0.63	0.63	0.61	0.00	0.00	1.00	0.63
barat	0.63	0.00	0.62	0.45	0.47	0.47	0.73	0.48	0.00	0.60	0.60	0.59	0.44	0.64	0.63	1.00

Setelah dilakukan perhitungan menggunakan algoritma Jaro-Winkler selanjutnya dilakukan perhitungan menggunakan algoritma Monge-Elkan *distance*. Rumus dapat dilihat pada Rumus 2.3.

Monge-Elkan *distance* A-00 (1) : U-20 (1)

$$Monge\ Elkan = \frac{1}{22}(1 + 0.7 + 1 + 1 + 1 + 1 + 0 + 1 + 0 + 0 + 1 + 1 + 0.7 + 0 + 1 + 1 + 1 + 1 + 1 + 1 + 1) = \frac{1}{22}.17{,}4 = 0.79$$

Pada perhitungan manual ini *threshold internal similarity* yang digunakan adalah 0,7 sehingga nilai dibawah nilai *threshold* tersebut akan diberi nilai 0.

Kalimat selanjutnya, pada dokumen asli yaitu kalimat kedua dan pada dokumen uji juga kalimat kedua akan dilakukan perhitungan Jaro-Winkler. Berikut tabel perhitungan kalimat kedua.

Tabel 3.5 Perhitungan Jaro-Winkler kalimat kedua A-00 (2) : A-20 (2)

Kata	giat	wakil	presiden	ri	boediono	serta	organisas	muslimah	ikut	muslimat	muhammadiyah	mui	icmi	hmi
giat	1.00	0.47	0.00	0.58	0.00	0.63	0.57	0.00	0.67	0.00	0.44	0.53	0.50	0.53
hadir	0.47	0.60	0.44	0.00	0.55	0.47	0.54	0.44	0.00	0.44	0.52	0.51	0.48	0.72
wakil	0.47	1.00	0.44	0.00	0.44	0.00	0.54	0.38	0.48	0.38	0.43	0.51	0.48	0.51
presiden	0.00	0.44	1.00	0.54	0.62	0.38	0.48	0.50	0.00	0.50	0.31	0.00	0.46	0.00
ri	0.58	0.00	0.54	1.00	0.00	0.00	0.54	0.00	0.58	0.00	0.00	0.61	0.58	0.61
boediono	0.00	0.44	0.67	0.00	1.00	0.44	0.46	0.42	0.00	0.42	0.47	0.00	0.46	0.00
serta	0.63	0.00	0.38	0.00	0.44	1.00	0.54	0.55	0.48	0.55	0.43	0.00	0.00	0.00
ister	0.63	0.00	0.38	0.57	0.44	0.78	0.00	0.44	0.67	0.44	0.00	0.00	0.54	0.00
organisas	0.57	0.54	0.48	0.54	0.46	0.54	1.00	0.35	0.00	0.35	0.42	0.00	0.45	0.00
muslimah	0.00	0.38	0.50	0.00	0.42	0.55	0.35	1.00	0.46	0.95	0.80	0.71	0.58	0.49
ikut	0.67	0.48	0.00	0.58	0.00	0.48	0.00	0.46	1.00	0.46	0.44	0.53	0.55	0.00
muslimat	0.00	0.38	0.50	0.00	0.42	0.55	0.35	0.95	0.46	1.00	0.74	0.71	0.58	0.49
nu	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.54	0.58	0.54	0.53	0.61	0.00	0.00
aisyiyah	0.42	0.55	0.47	0.54	0.50	0.55	0.59	0.75	0.46	0.67	0.68	0.49	0.58	0.49
muhammadiyah	0.44	0.43	0.31	0.00	0.47	0.43	0.42	0.80	0.44	0.74	1.00	0.69	0.44	0.47
mui	0.53	0.51	0.00	0.61	0.00	0.00	0.00	0.71	0.53	0.71	0.69	1.00	0.53	0.78
icmi	0.67	0.48	0.46	0.58	0.46	0.00	0.45	0.58	0.55	0.58	0.44	0.53	1.00	0.72
hmi	0.53	0.51	0.00	0.61	0.00	0.00	0.00	0.49	0.00	0.49	0.47	0.78	0.72	1.00

Monge-Elkan *distance* A-00 (2) : U-20 (2)

$$Monge\ Elkan = \frac{1}{18}(1 + 0.72 + 1 + 1 + 1 + 1 + 1 + 0.78 + 1 + 1 + 1 + 1 + 0 + 0.75 + 1 + 1 + 1 + 1)$$
$$= 0.90$$

Sama halnya dengan kalimat ketiga, proses dilakukan dengan perhitungan Jaro-Winkler selanjutnya dilakukan perhitungan Monge-Elkan. Perhitungan Jaro-Winkler dapat dilihat pada tabel 3.3.

Tabel 3.6 Perhitungan Jaro-Winkler kalimat ketiga A-00 (3) : A-20 (3)

Kata	presiden	bambang	yudhoyono	teri	agama	ri	suryadharma	ali	jadwal	hadir
presiden	1.00	0.42	0.49	0.60	0.00	0.54	0.44	0.00	0.43	0.44
susilo	0.63	0.00	0.52	0.47	0.00	0.00	0.58	0.50	0.44	0.46
bambang	0.42	1.00	0.42	0.00	0.79	0.00	0.49	0.49	0.54	0.45
yudhoyono	0.49	0.42	1.00	0.00	0.00	0.00	0.60	0.00	0.43	0.44
teri	0.60	0.00	0.00	1.00	0.00	0.00	0.45	0.53	0.00	0.63
agama	0.00	0.79	0.00	0.00	1.00	0.00	0.53	0.56	0.70	0.60
ri	0.54	0.00	0.00	0.00	0.00	1.00	0.00	0.61	0.00	0.00
suryadharma	0.44	0.49	0.60	0.45	0.53	0.00	1.00	0.00	0.59	0.43
ali	0.00	0.49	0.00	0.53	0.56	0.61	0.00	1.00	0.50	0.69
jadwal	0.43	0.54	0.43	0.00	0.70	0.00	0.59	0.50	1.00	0.58
hadir	0.44	0.45	0.44	0.63	0.47	0.00	0.43	0.69	0.58	1.00
acara	0.44	0.68	0.00	0.48	0.76	0.00	0.51	0.56	0.70	0.73

Monge-Elkan *distance* A-00 (2) : U-20 (2)

$$Monge\ Elkan = \frac{1}{12}(1 + 0 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0.76) = 0.89$$

Dari hasil Monge-Elkan *distance* diatas, dilakukan perhitungan rata-rata yaitu dengan cara menjumlahkan nilai *string matching* Monge-Elkan kemudian dibagi dengan jumlah kalimat dokumen :

$$Plagiarized\ Value = \frac{0.79 + 0.90 + 0.89}{3} = \mathbf{0.86}$$

Dari perhitungan diatas, berarti dokumen asli A-00 dengan uji A-20 memiliki kemiripan isi teks sebesar **86 %.**

BAB IV
IMPLEMENTASI

Pada bab ini akan dihabas tentang implementasi dari proses perancangan yang telah diuraikan pada bab sebelumnya. Implementasi yang akan dibahas pada bab ini yaitu lingkungan implementasi, implementasi program, implementasi antarmuka, dan implementasi uji coba.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang dibutuhkan agar proses-proses dapat diimplementasikan dan berjalan tepat. Lingkungan implementasi yang akan diuraikan melipat lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan deteksi *plagiarisme* dokumen teks menggunakan algoritma Monge-Elkan Distance adalah sebagai berikut :

1. Processor Pentium Dual-Core CPU B950 @2.10GHZ
2. VGA Nvidia Geforce GT520M 1GB
3. Memory 4.00 GB RAM
4. Hardisk 320 GB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan meliputi :

1. Sistem Operasi Windows 7 64-bit, sebagai *operating system* yang dipakai.
2. Netbeans IDE 7.0.1, sebagai program dalam penulisan kode java.
3. Java Development Kit (JDK) 1.7.0_45

4.2 Implementasi Program

4.2.1 Proses dan Fungsi

Proses input dokumen dilakukan oleh user. Data yang digunakan memiliki tipe data .txt. Data yang dimasukkan akan diolah dari proses infofile, *preprocessing* dan String matching.

Tabel 4.1 Proses dan fungsi

NO.	Proses	Fungsi	Keterangan
1	Info file	BacaFile()	Membaca data text
		hitungKalimat()	Menghitung jumlah kalimat
		hitungKata()	Menghitung jumlah kalimat
2	Preprocessing	CaseFolding()	Merubah text menjadi huruf kecil
		Tokenizing()	Menyimpan tiap kata ke dalam bentuk array
		cekThresholdStruktural()	Mengecek kalimat yang memiliki kesamaan struktural kata lebih dari nilai *threshold* struktural.
		Filtering()	Menghapus kata-kata yang kurang penting (*stopword)*
		Stemming(0	Merubah kata ke bentuk kata dasar
3	String Mathcing	Jarowinkler()	Fungsi untuk menghitung *internal similarity*
		MongeElkan()	Fungsi untuk menghitung *similarity* akhir.

4.2.2 Tahap Pemrosesan

Langkah saat pertama kali aplikasi ini dijalankan adalah membaca dokumen inputan dari user dengan fungsi BacaFile yang terdapat pada Kode Sumber 4.1. Pada fungsi ini aplikasi akan membaca file dari awal sampai akhir. Proses mendapatkan informasi dari dokumen yaitu proses hitung kalimat (Kode Sumber 4.2) dan hitung kata (Kode Sumber 4.3). Proses mendapatkan informasi ini terdapat pada kelas *infofile*.

```
    String text;
    public InfoFile infofile;
    public BacaFile(String s) throws Exception{
        BufferedReader br = new BufferedReader(new FileReader(s));
        try{
            String tmp="";
            String line;
            //membaca setiap baris dari dokumen
            while((line=br.readLine())!=null){
                tmp=tmp+line+"\n";
            }
            text=tmp;
        }catch(FileNotFoundException e){
            System.out.println("File tidak ditemukan");
        }
        //Object baru dari kelas InfoFile
        infofile = new InfoFile(text);    }

    public String getText(){
        return text;
    }
```

Kode Sumber 4.1 *Source code* proses baca file

Pada fungsi hitungKalimat() ini digunakan *delimiter* '.' Untuk menghitung seluruh jumlah kalimat pada dokumen.

```
private void hitungKalimat(String s){
        String[] tmp = s.split("[.]");
        jmlKalimat = tmp.length-1;
    }
public int getJmlKalimat(){
        return jmlKalimat;
    }
```

Kode Sumber 4.2 *Source code* hitung jumlah kalimat

Pada fungsi hitungKata(), *delimiter* yang digunakan adalah spasi (" "). Fungsi ini menghitung jumlah seluruh kata pada dokumen.

```
private void hitungKata(String s){
        String[] tmp = s.split(" ");
        jmlKata = tmp.length;
    }
public int getJmlKata(){
        return jmlKata;
    }
```

Kode Sumber 4.3 *Source code* hitung jumlah kata

Setelah isi dokumen teks berhasil dibaca oleh aplikasi, tahap selanjutnya adalah tahap *preprocessing*. Tahap awal dari *preprocessing* adalah *case folding* yang dapat dilihat pada Kode Sumber 4.4.

```
public CaseFolding(String s){
        String temp=s.toLowerCase();
        //pemberian nilai delimiter
        String delimiter = "[^A-Za-z.]^\\d";
        temp = temp.replaceAll(delimiter, " ").trim();
        temp = temp.replaceAll("\\s+", " ");
        caseFolding=temp;
    }

    public String getCaseFolding(){
        return caseFolding;
    }
```

Kode Sumber 4.4 *Source code case folding*

Tahap *preprocessing* berikutnya adalah *tokenizing* yaitu memparsing text menjadi pecahan kata yang disimpan ke dalam array. Fungsi *tokenizing* dapat dilihat pada Kode Sumber 4.5.

```
    ArrayList<String> token=new ArrayList<String>();
    public Tokenizing(String s){
        String[] temp = s.split(" ");
        token.addAll(Arrays.asList(temp));
    }
    public ArrayList<String> getToken(){
        return token;
    }
```

Kode Sumber 4.4 *Source Code Tokenizing*

Proses selanjutnya adalah menghitung nilai kesamaan struktural kalimat dari dokumen asli dan dokumen uji.

```
private double cekThresholdStruktural(String s, String t){
        double nilai;
        Tokenizing ss = new Tokenizing(s);
        Tokenizing st = new Tokenizing(t);
        ArrayList<String> temp1 = ss.getToken();
        ArrayList<String> temp2 = st.getToken();
        double max=Math.max(temp1.size(), temp2.size()); //Sebagai
pembagi
        double sama=0;
        //mencari kata yang sama
        for(int i=0;i<temp1.size();i++){
            for(int j=0;j<temp2.size();j++){
                if(temp1.get(i).equalsIgnoreCase(temp2.get(j))){
                    sama++;
                }
            }
        }
        nilai = sama/max;
        return nilai;
    }
```

Kode Sumber 4.5 *Source code* proses hitung kesamaan struktural kalimat

Setelah didapatkan kalimat mana saja yang memiliki nilai kesamaan struktural kata, selanjutnya nilai tersebut diseleksi tidak kurang dari nilai *threshold* awal yaitu *threshold* struktural kalimat. Langkah selanjutnya adalah proses *filtering* yaitu menghilangkan kata-kata yang termasuk kata *stopword*.

```
ArrayList<String> KataUnik=new ArrayList<String>();
    Stopwords stop = new Stopwords();
    public Filtering(ArrayList<String> s) {
        for(int i=0;i<s.size();i++){
            if(cekStopword(s.get(i))==false){
                KataUnik.add(s.get(i));
            }
        }
    }

    private boolean cekStopword(String s){
        boolean remove=false;
        if(stop.isStopword(s)){
            remove=true;
        }
        return remove;
    }

    public ArrayList<String> getKataUnik(){
        return KataUnik;
    }
```

Kode Sumber 4.6 *Source code* proses *filtering*

Setiap kata yang tidak ditemukan dalam *stopword* akan dimasukkan ke array KataUnik yang nantinya akan dilakukan proses *stemming*. Proses *stemming* dilakukan dengan menggunakan algortima Porter-Fadillah. Pada Kode Sumber 4.7 merupakan pemanggilan fungsi-fungsi dari algoritma Porter-Fadillah.

```
        String stem;
        stem=hapusawalan1(hapuspp(hapuspartikel(s)));
        //aturan stemming porter-fadillah
        if(stem.equalsIgnoreCase(hapuspp(hapuspartikel(s)))){
            stem=hapusakhiran(hapusawalan2(stem));
        }
        else{
            if(!hapusakhiran(stem).equalsIgnoreCase(stem)){
                stem=hapusawalan2(hapusakhiran(stem));
            }
        }
        return stem;
```

Kode Sumber 4.7 *Source code proses stemming*

Pada kode sumber diatas terdapat fungsi hapusawalan, hapusakhiran, hapusawalan2, hapuspp, dan hapuspartikel. Kode sumber tersebut dapat dilihat pada lampiran.

Setelah dilakukan proses *stemming*, selanjutnya akan dilakukan proses pencocokan string menggunakan algoritma Monge-Elkan *distance*. Dengan *internal similarity* menggunakan algoritma Jaro-Winkler. Kode sumber 4.8 merupakan proses pencocokan string dengan algoritma Jaro-Winkler dan kode sumber 4.9 merupakan pencocokan string dengan algoritma Jaro-Winkler.

```
public double jarowinkler(String s, String t){
        int c=0; //karakter yang sama
        int Tr=0; //transposisi karakter
        int prevpos = -1;
        int p=0; // jumlah prefix
        if (s.length()>t.length()){
            String tmp = s;
            s = t;
            t = tmp;
        }
        int maxrange = s.length()/2;
        for (int i=0;i<s.length();i++){
            char ch=s.charAt(i);
            for(int j=Math.max(0,i-
maxrange);j<=Math.min(t.length()-1, i+maxrange);j++){
                if (ch == t.charAt(j)) {
                    c++;
                    if (prevpos != -1 && j < prevpos &&
ch!=t.charAt(i)) {
                            Tr++;
                    }
                    prevpos = j;
                    break;
                }
            }
        }

        if (c==0){
            return 0.0;
        }
        //Jaro
        double score =
((c/(double)s.length())+(c/(double)t.length())+((c-
Tr)/(double)c))/3.0;
        //JaroWinkler
        int maxPrefix = Math.min(4, s.length());
        for(int i = 0; i<maxPrefix;i++){
            if(s.charAt(i)==t.charAt(i)){
                p++;
```

```
            }
            else break;
        }
        score = score+(p*(1-score)/10);
        return score;
    }
```

Kode Sumber 4.8 Jaro-Winkler *similarity*

Pada fungsi Jarowinkler, setiap kata pada dokumen asli akan dihitung nilai *similarity* kesamaan kata. Dimulai dengan mencari jumlah huruf yang sama, jumlah transposisi huruf, dan jumlah huruf dalam prefix yang sama. Ketiga variable tersebut nanti nya akan diproses sesuai perhitungan matematis algoritma Jaro-Winkler.

Setelah didapat nilai *similarity* dari algoritme Jaro-Winkler, selanjutnya akan digunakan algoritma Monge-Elkan untuk menghitung nilai per kalimat.

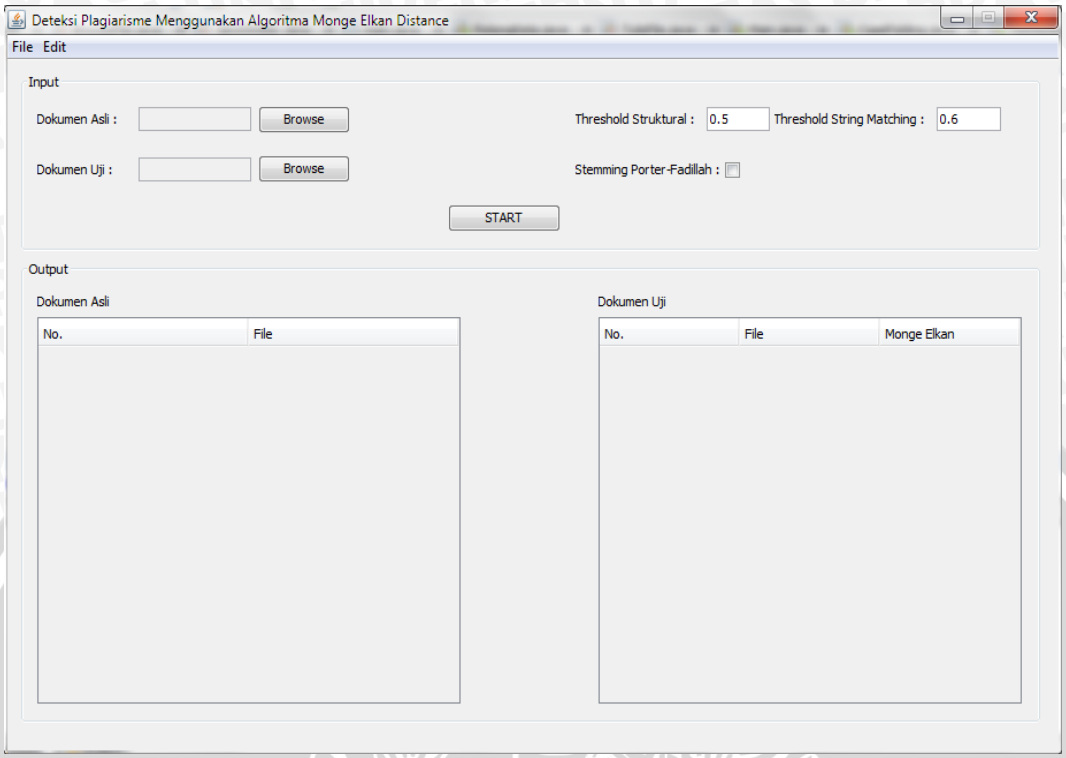
```
private double MongeElkan(ArrayList<String> s, ArrayList<String>
t, double Tr){
        JaroWinkler jw = new JaroWinkler();
        ArrayList<Double> score = new ArrayList<Double>();
        for(int i=0;i<s.size();i++){
            ArrayList<Double> scoretemp = new ArrayList<Double>();
            for(int j=0;j<t.size();j++){
                //Menghitung nilai JaroWinkler
                double nilai = jw.compare(s.get(i), t.get(j));
                scoretemp.add(nilai);
            }
            //Threshold internal similarity
            if(Collections.max(scoretemp)>=Tr){
                //mengambil nilai dengan kesamaan kata terbesar
                score.add(Collections.max(scoretemp));
            }
            else{
                score.add(0.0);
            }
        }
        //rata-rata nilai
        return Average(score);
    }
```

Kode Sumber 4.9 Monge-Elkan *similarity*

4.3 Implementasi User Interface

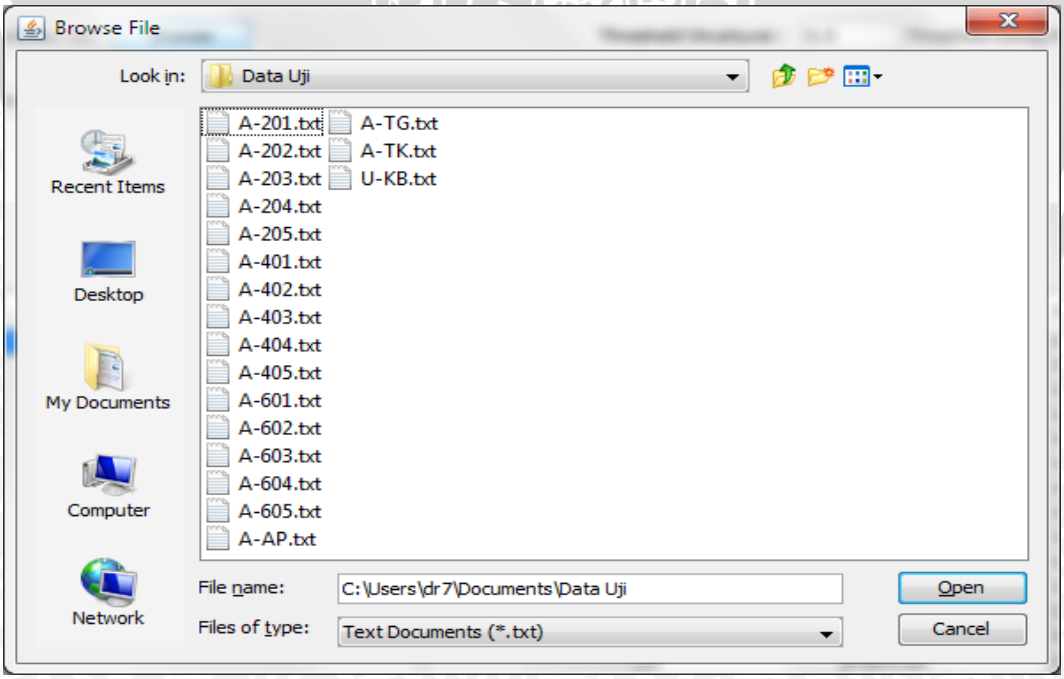
Berdasarkan perancangan yang sudah dijelaskan pada bab 3, pada sub bab ini akan dijelaskan implementasi *user interface*. Pada tampilan awal terdapat beberapa *field* yang harus diisi, yaitu nilai *threshold* struktural kalimat, *threshold internal similarity*, dan penggunaan *stemming* atau tidak. Gambar antarmuka awal

aplikasi ditunjukan pada Gambar 4.1. Pertama kali user melakukan input file dengan menekan tombol *browse.* Input file ditunjukan pada Gambar 4.2.



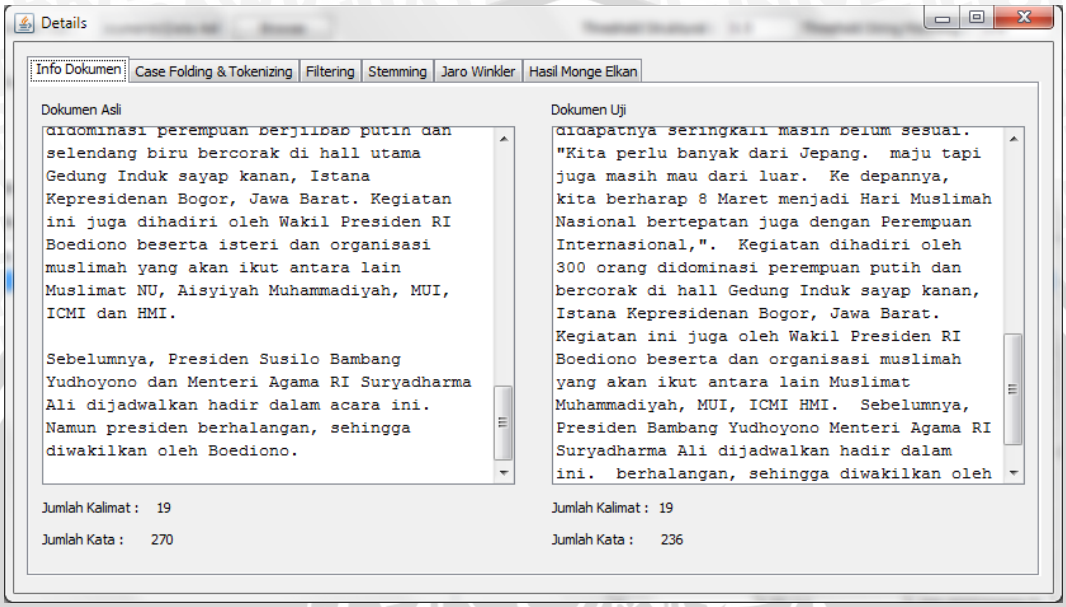
Gambar 4.1 Tampilan halaman utama

Data yang diiputkan adalah data dalam satu folder seperti yang terlihat pada Gambar 4.2



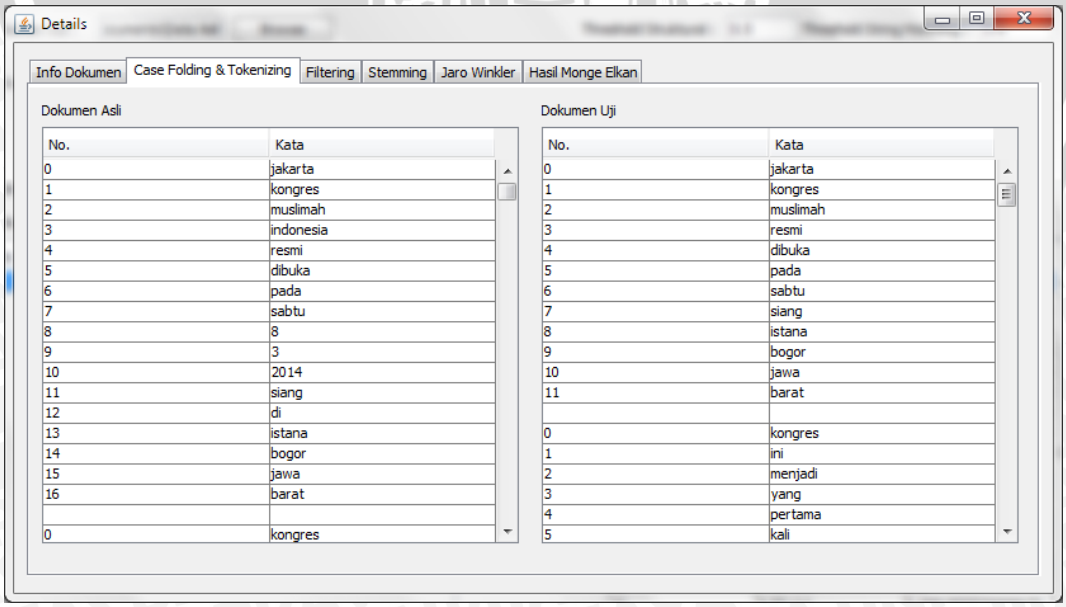
Gambar 4.2 Tampilan *Browse* file

Setelah semua *field* diisi, selanjutnya dilakukan proses perhitungan nilai *similarity* dokumen. Dari hasil nilai *similarity* dapat dilihat detail dari perhitungan tersebut, mulai dari *preprocessing* sampai perhitungan similarity. Detail hasil perhitungan tersebut dapat dilihat pada Gambar 4.3 sampai 4.8.



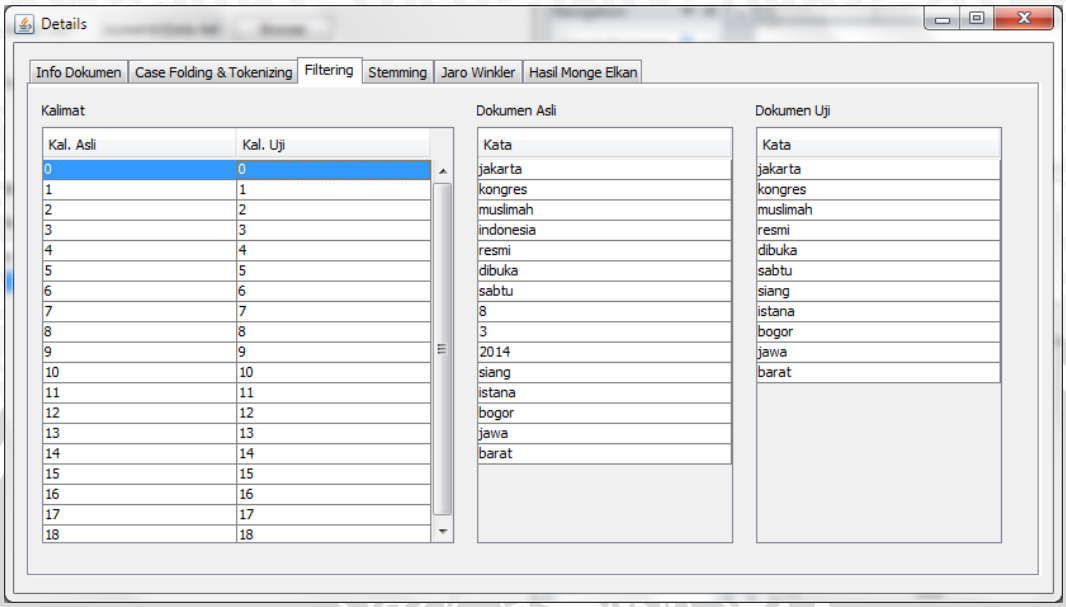
Gambar 4.3 Tampilan Detail perhitungan

Interface *case folding & tokenizing* pada Gambar 4.4. Proses ini menyimpan setiap kata ke dalam bentuk array.



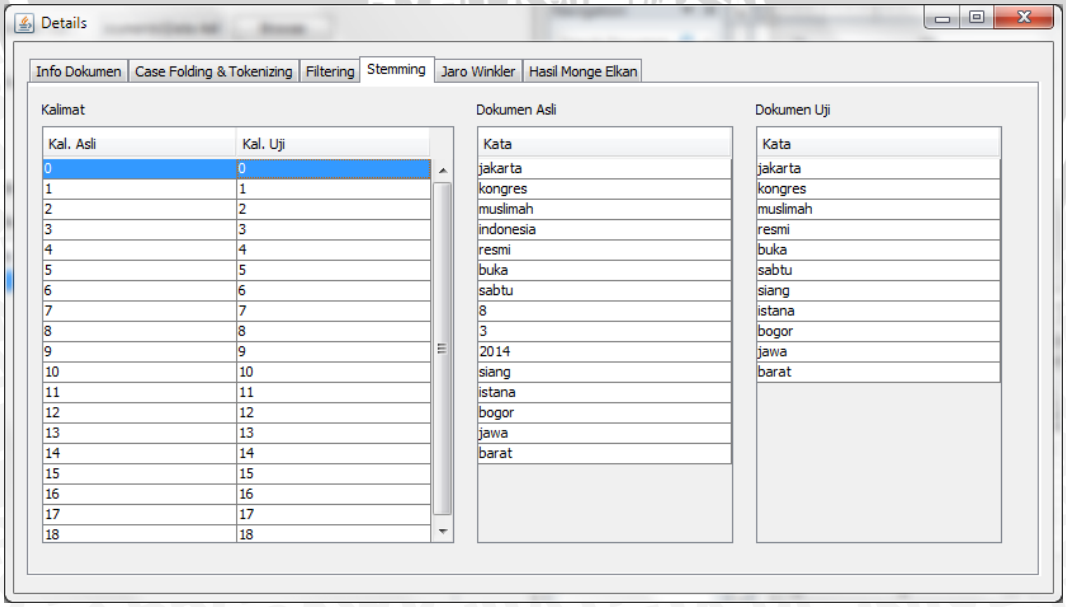
Gambar 4.4 Tampilan *case folding & tokenizing*

Selanjutnya adalah tampilan dari tab *filtering*. Pada tab ini ditampilkan kata-kata yang unik atau kata-kata yang tidak ada pada *stopword*. Tampilan dapat dilihat pada Gambar 4.5



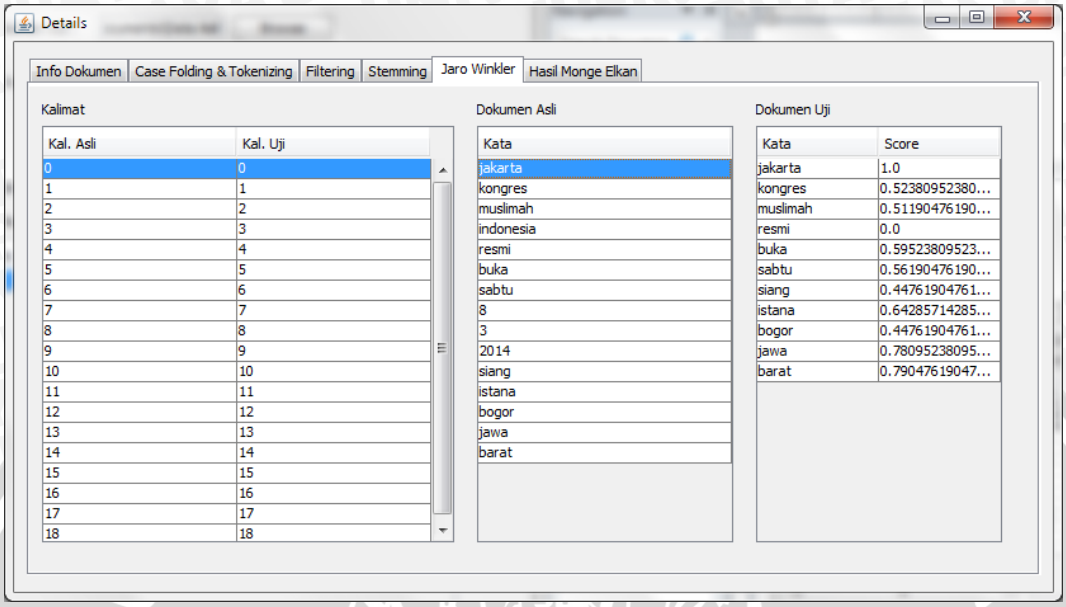
Gambar 4.5 Tampilan *filtering*

Selanjutnya tampilan proses *stemming* yang merubah kata menjadi bentuk kata dasar dapat dilihat pada Gambar 4.6

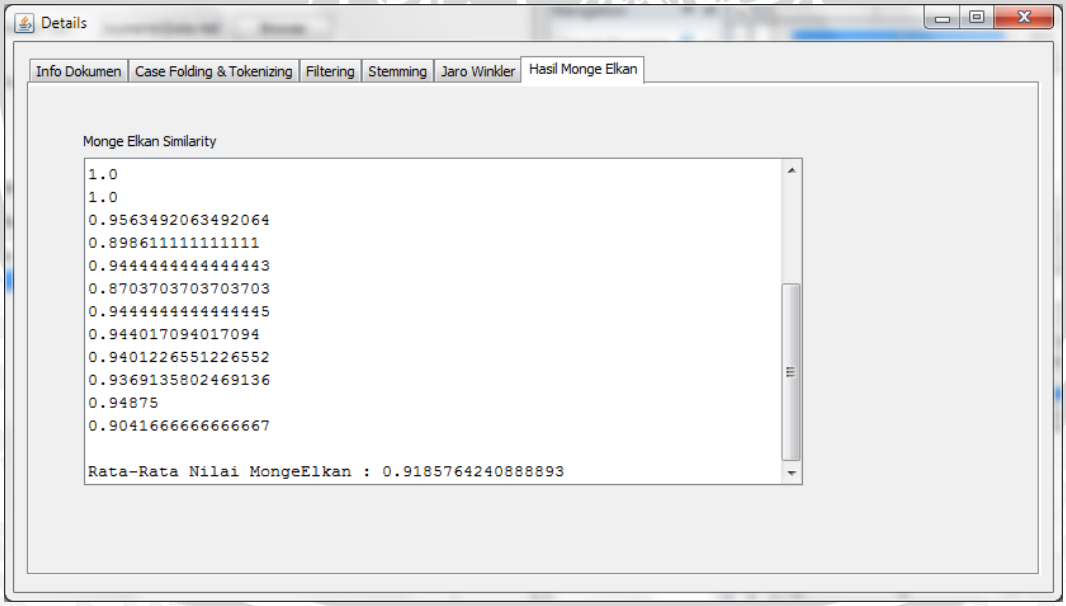


Gambar 4.6 Tampilan *stemming*

Perhitungan nilai *similarity* dari algoritma Jaro-Winkler dan Monge-Elkan dapat dilihat pada Gambar 4.7 dan Gambar 4.8



Gambar 4.7 Tampilan perhitungan Jaro-Winkler



Gambar 4.8 Tampilah hasil perhitungan Monge-Elkan *distance*

BAB V
HASIL DAN PEMBAHASAN

Pada BAB V ini dibahas mengenai hasil pengujian dan pembahasan, sesuai dengan rancangan pengujian pad bab III.

5.1 Hasil Pengujian

Pengujian dilakukan untuk mengetahui kelayakan sistem yang dibuat sudah bekerja dengan baik sesuai dengan spesifikasi aplikasi yang telah ditentukan. Ada beberapa bentuk pengujian pada aplikasi ini yaitu pengujian *threshold* struktural kalimat, *thresold internal similarity*, pengaruh *stemming* pada hasil akhir, dan pengujian prosentase *error*. Adapun informasi data yang digunakan. Berikut adalah seluruh informasi data asli dan data uji yang diujikan pada sistem.

Tabel 5.1 Informasi dokumen asli dan uji

Kode	Jumlah Kalimat	Jumlah Kata
A-00	19	277
A-20	19	220
A-40	19	169
A-60	19	113
A-TK	19	277
A-AP	19	277
A-TG	19	277
A-KB	19	219
B-00	17	246
B-20	17	201
B-40	17	148
B-60	17	97
B-TK	17	246
B-AP	17	246
B-TG	17	246
B-KB	17	200

C-00	13	320
C-20	13	258
C-40	13	194
C-60	13	126
C-TK	13	320
C-AP	13	320
C-TG	13	320
C-KB	13	260

5.1.1 Hasil Pengujian Threshold Struktural Kalimat

Ujicoba dilakukan dengan nilai *Threshold struktural* 0.0. sampai 0.9 dan *threshold internal* 0.6. Pengujian dilakukan sebanyak lima kali dengan pemotongan kata random. Nilai *similarity* diperoleh dengan cara menghitung rata-rata dari lima kali pengujian. Hasil dari pengujian dapat dilihat pada tabel 5.2 sampai tabel 5.6. Setiap tabel adalah hasil dari pengujian untuk setiap dokumen uji.

Tabel 5.2 Hasil pengujian dokumen uji A dengan *threshold struktural* kalimat

Threshold	Similarity (%)						
	A-20	A-40	A-60	A-TK	A-AP	A-TG	A-KB
0	54.49	47.75	37.48	59.33	59.39	59.42	55.19
0.1	65.18	60.44	48.94	69.20	69.23	71.28	66.11
0.2	79.71	72.69	56.66	85.70	85.26	87.26	83.35
0.3	88.56	75.94	54.79	97.81	97.20	96.86	88.57
0.4	88.95	76.78	36.38	100.00	99.37	98.16	89.50
0.5	88.95	76.78	11.53	100.00	99.37	98.16	89.50
0.6	88.95	49.04	1.37	100.00	99.37	98.16	89.50
0.7	88.95	16.29	0.00	100.00	99.37	98.16	85.43
0.8	57.28	0.00	0.00	100.00	99.37	67.34	57.97
0.9	17.71	0.00	0.00	100.00	94.26	15.43	8.94

Tabel 5.3 Hasil pengujian dokumen uji B dengan *threshold struktural* kalimat

Threshold	Similarity (%)						
	B-20	B-40	B-60	B-TK	B-AP	B-TG	B-KB
0	54.21	49.02	39.51	58.84	58.90	58.93	55.46
0.1	66.52	64.82	54.99	70.10	70.11	69.99	67.99
0.2	78.69	74.61	59.92	85.66	85.55	85.38	80.47
0.3	88.17	78.78	58.40	100.00	99.83	99.37	90.39
0.4	88.17	78.78	46.02	100.00	99.83	99.37	90.39
0.5	86.76	77.37	12.30	100.00	99.83	99.37	90.39
0.6	86.76	55.37	0.00	100.00	99.83	99.37	90.39
0.7	86.76	18.97	0.00	100.00	99.83	99.37	84.74
0.8	65.50	3.79	0.00	100.00	99.83	99.37	47.69
0.9	24.01	0.00	0.00	100.00	99.83	81.97	11.20

Tabel 5.4 Hasil pengujian dokumen C dengan *threshold struktural* kalimat

Threshold	Similarity (%)						
	C-20	C-40	C-60	C-TK	C-AP	C-TG	C-KB
0	63.60	58.65	51.15	67.79	67.75	66.73	61.61
0.1	69.23	66.24	59.68	73.11	73.03	74.37	67.98
0.2	80.15	74.71	66.98	83.57	83.55	89.95	80.15
0.3	88.55	80.54	69.77	96.63	96.53	97.95	86.73
0.4	91.50	82.15	63.60	100.00	99.89	97.95	87.85
0.5	91.66	82.13	17.80	100.00	99.89	97.95	87.85
0.6	91.66	72.63	7.84	100.00	99.89	97.95	87.85
0.7	91.66	37.56	4.40	100.00	99.89	97.95	74.14
0.8	91.66	16.75	3.19	100.00	99.89	90.40	21.69
0.9	48.38	6.50	2.27	100.00	99.89	60.48	7.28

Dari percobaan pemberian nilai *threshold* stuktrual diatas pada nilai tertentu nilai *similarity* akan menurun drastis. Untuk *threshold 0.6*, nilai *similarity* pada dokumen uji pemotongan kata sebanyak 60% turun drastis dari *threshold* 0,4, nilai *similarity* rata-rata turun sebesar 45,71% . Begitu juga terjadi pada

dokumen uji 40% dan 20% pemotongan kata pada threshold 0.8, nilai *similarity* relatif turun drastis. Untuk data uji tukar kalimat (TK), perubahan kalimat aktif menjadi pasif begitu pula sebaliknya (AP) memiliki nilai *similarity* tetap pada *threshold* diatas 0,3.

5.1.2 Pengujian Threshold Internal Similarity

Ujicoba pada pengujian ini digunakan nilai *threhsold* struktural kalimat 0.3 dan *thresold internal similarity* rentang 0.5 sampai 1.0. Pengujian dilakukan sebanyak lima kali dengan pemotongan kata random. Nilai *similarity* diperoleh dengan cara menghitung rata-rata dari lima kali pengujian. Pengujian dokumen A dapat dilihat pada Tabel 5.5

Tabel 5.5 Hasil pengujian dokumen uji A dengan *threshold internal similarity*

Threshold	Similarity(%)						
	A-20	A-40	A-60	A-TK	A-AP	A-TG	A-KB
0.5	92.23	83.68	66.38	98.14	97.54	97.21	92.08
0.6	88.56	75.94	54.79	97.81	97.20	96.86	88.57
0.7	84.29	67.27	44.74	97.44	96.51	96.86	83.84
0.8	80.59	60.82	38.86	94.59	93.12	95.32	79.78
0.9	79.52	58.71	37.11	92.69	90.16	87.52	78.39
1	79.14	58.01	36.40	91.60	87.68	77.09	76.71

Selanjutnya, pengujian *threhsold internal similarity* pada dokumen uji B dan C dapat dilihat pada Tabel 5.6 dan Tabel 5.7.

Tabel 5.6 Hasil pengujian dokumen B dengan *threshold internal similarity*

Threshold	Similarity(%)						
	A-20	A-40	A-60	A-TK	A-AP	A-TG	A-KB
0.5	90.85	84.03	67.47	100.00	99.83	99.37	91.88
0.6	88.17	78.78	58.40	100.00	99.83	99.37	90.39
0.7	88.17	78.78	58.40	100.00	99.83	99.37	90.39
0.8	79.16	63.49	40.07	100.00	99.83	99.37	82.46
0.9	78.61	62.03	38.38	100.00	98.94	98.32	81.15
1	78.49	61.73	38.08	100.00	98.94	92.22	76.90

Tabel 5.7 Hasil pengujian dokumen C dengan *threshold internal similarity*

Threshold	Similarity(%)						
	C-20	C-40	C-60	C-TK	C-AP	C-TG	C-KB
0.5	89.90	83.43	75.53	96.63	96.53	97.95	88.69
0.6	88.55	80.54	69.77	96.63	96.53	97.95	86.73
0.7	81.61	70.18	55.33	93.86	93.76	97.76	79.60
0.8	75.06	60.67	45.83	91.50	91.41	96.53	72.97
0.9	71.83	57.28	42.54	89.85	89.34	90.32	66.61
1	70.97	56.97	42.03	89.85	89.34	81.97	58.13

Hasil pengujian menujukkan bahwa semakin tinggi nilai *threshold* nilai *similarity* akan terus menurun. Penurunan nilai terbesar terjadi pada dokumen uji A dengan kesalahan pengejaan kata (TG) yang mencapai 10,43% pada threshold 0,9 ke 1,0.

5.1.3 Hasil Pengujian *Similarity* Dokumen

Ujicoba dilakukan dengan menguji data menggunakan algoritma Monge-Elkan *distance.* Nilai *threshold struktural* yang digunakan adalah 0.5 diasumsikan bahwa yang dicari adalah dokumen uji dengan kemiripan diatas 50 % dan nilai *threshold internal* yang digunakan adalah 0.8. Hasil pengujian pada Table 5.2 adalah rata-rata nilai prosentase kemiripan dokumen dengan pengujian random sebanyak lima kali percobaan.

Table 5.8 Pengaruh *stemming* terhadap hasil pengujian *similarity*

Dok. Asli	Dok. Uji	Threshold Struktural	Threshold int. similarity	Monge-Elkan		
				S(%) Non Stemming	S(%) Stemming	Kategori
A-00	A-20	0.5	0.8	81.91	82.31	Berat
	A-40	0.5	0.8	62.96	63.99	Sedang
	A-60	0.5	0.8	8.59	9.04	Ringan
	A-TK	0.5	0.8	100.00	100.00	Berat
	A-AP	0.5	0.8	98.36	99.94	Berat

	A-TG	0.5	0.8	97.99	93.55	Berat
	A-KB	0.5	0.8	82.48	82.18	Berat
B-00	B-20	0.5	0.8	77.75	78.10	Berat
	B-40	0.5	0.8	62.08	61.82	Sedang
	B-60	0.5	0.8	8.45	8.46	Ringan
	B-TK	0.5	0.8	100.00	100.00	Berat
	B-AP	0.5	0.8	99.83	100.00	Berat
	B-TG	0.5	0.8	99.37	98.54	Berat
	B-KB	0.5	0.8	82.46	82.54	Berat
C-00	C-20	0.5	0.8	83.42	83.38	Berat
	C-40	0.5	0.8	65.18	65.38	Sedang
	C-60	0.5	0.8	12.36	12.75	Ringan
	C-TK	0.5	0.8	100.00	100.00	Berat
	C-AP	0.5	0.8	99.89	99.96	Berat
	C-TG	0.5	0.8	96.53	95.48	Berat
	C-KB	0.5	0.8	76.70	74.84	Berat

Hasil dari pengujian penggunaan *stemming* atau tanpa *stemming* menunjukkan adanya peningkatan nilai *similarity* pada beberapa dokumen seperti pada dokumen uji C-AP, sedangkan beberapa dokumen lain mengalami penurunan nilai *similarity*. Seperti pada hasil dokumen uji dengan kesalahan pengejaan kata (TG) pada setiap dokumen.

5.1.4 Hasil Pengujian Prosentase *error*

Pada pengujian ini setiap data uji akan dilakukan pemotongan kata secara random sebanyak lima kali. Setiap hasil dari pengujian tersebut akan dibandingkan dengan nilai *similarity* yang diharapkan. Nilai yang diharapkan sesuai dengan jumlah kata dari pemotongan kata yaitu untuk dokumen uji pemotongan kata 20%, nilai *similarity* yang diharapkan adalah 80%, pemotongan kata 40%, nilai *similarity* yang diharapkan 60%, dan untuk dokumen uji pemotongan kata 60%, nilai *similarity* yang diharapkan adalah 40% . Adapun dari data uji total (KB) nilai similarity yang diharapkan adalah 80% , karena pada

pengujian ini dokumen tersebut jumlah kata yang dipotong sebesar 20%. Hasil pengujian dapat dilihat pada Tabel 5.9 sampai 5.11

Tabel 5.9 Hasil pegujian prosentase *error non-stemming* dokumen A

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
A-00	A-201.txt	82.55	2.55	3.00
	A-202.txt	77.26	2.74	
	A-203.txt	83.71	3.71	
	A-204.txt	83.69	3.69	
	A-205.txt	82.32	2.32	
	A-401.txt	60.70	0.70	2.96
	A-402.txt	65.18	5.18	
	A-403.txt	64.00	4.00	
	A-404.txt	60.75	0.75	
	A-405.txt	64.14	4.14	
	A-601.txt	6.28	33.72	31.41
	A-602.txt	3.78	36.22	
	A-603.txt	7.74	32.26	
	A-604.txt	14.68	25.32	
	A-605.txt	10.47	29.53	
	A-AP.txt	98.36	1.64	1.64
	A-KB.txt	82.48	2.48	2.48
	A-TG.txt	97.99	2.01	2.01
	A-TK.txt	100.00	0.00	0.00

Nilai *prosentase error* tanpa *stemming* terbesar terjadi pada dokumen uji dengan pemotongan kata sebanyak 60% dari jumlah total kata. Nilai *prosentase error* dari dokumen uji tersebut adalah 31,41%.

Tabel 5.10 Hasil pegujian prosentase *error non-stemming* dokumen B

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
B-00	B-201.txt	82.96	2.96	3.44
	B-202.txt	73.06	6.94	
	B-203.txt	76.07	3.93	
	B-204.txt	79.93	0.07	
	B-205.txt	76.71	3.29	
	B-401.txt	54.15	5.85	4.42
	B-402.txt	63.99	3.99	
	B-403.txt	64.96	4.96	
	B-404.txt	61.10	1.10	
	B-405.txt	66.18	6.18	
	B-601.txt	5.88	34.12	31.55
	B-602.txt	7.86	32.14	
	B-603.txt	10.67	29.33	
	B-604.txt	5.88	34.12	
	B-605.txt	11.96	28.04	
	B-AP.txt	99.83	0.17	0.17
	B-KB.txt	82.46	2.46	2.46
	B-TG.txt	99.37	0.63	0.63
	B-TK.txt	100.00	0.00	0.00

Tabel 5.11 Hasil pegujian prosentase *error non-stemming* dokumen C

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
C-00	C-201.txt	81.56	1.56	3.42
	C-202.txt	84.59	4.59	
	C-203.txt	84.01	4.01	
	C-204.txt	85.99	5.99	

	C-205.txt	80.91	0.91	
	C-401.txt	62.61	2.61	5.18
	C-402.txt	65.56	5.56	
	C-403.txt	64.92	4.92	
	C-404.txt	66.32	6.32	
	C-405.txt	66.50	6.50	
	C-601.txt	8.42	31.58	27.64
	C-602.txt	9.90	30.10	
	C-603.txt	14.43	25.57	
	C-604.txt	17.79	22.21	
	C-605.txt	11.26	28.74	
	C-AP.txt	99.89	0.11	0.11
	C-KB.txt	76.70	3.30	3.30
	C-TG.txt	96.53	3.47	3.47
	C-TK.txt	100.00	0.00	0.00

Pada pengujian prosentase *error non-stemming* diatas dapat dilihat bahwa dokumen uji dengan pemotongan kata 60% memiliki rata-rata prosentase *error* yang cukup besar yaitu 30,2%. Hasil rata-rata prosentasi *error* paling kecil terjadi pada dokumen pengujian dengan penukaran susunan kalimat yaitu sebesar 0%. Berikutnya adalah pengujian prosentase *error* dengan stemming. Pengujian dapat dilihat pada tabel 5.12 sampai 5.14

Tabel 5.12 Hasil pegujian prosentase *error* dengan *stemming* dokumen A

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
A-00	A-201.txt	82.26	2.26	2.89
	A-202.txt	78.55	1.45	
	A-203.txt	83.18	3.18	
	A-204.txt	85.42	5.42	
	A-205.txt	82.13	2.13	

	A-401.txt	62.20	2.20	3.99
	A-402.txt	66.49	6.49	
	A-403.txt	64.26	4.26	
	A-404.txt	62.54	2.54	
	A-405.txt	64.45	4.45	
	A-601.txt	7.77	32.23	30.96
	A-602.txt	3.33	36.67	
	A-603.txt	7.74	32.26	
	A-604.txt	14.91	25.09	
	A-605.txt	11.43	28.57	
	A-AP.txt	99.94	0.06	0.06
	A-KB.txt	82.18	2.18	2.18
	A-TG.txt	93.55	6.45	6.45
	A-TK.txt	100.00	0.00	0.00

Nilai rata-rata prosentase *error* dengan *stemming* pada dokumen A menunjukkan prosentase *error* paling besar terjadi pada dokumen uji pemotongan kata 60% dan dokumen uji tanpa prosentase *error* yaitu dokumen uji tukar kalimat.

Tabel 5.13 Hasil pegujian prosentase *error* dengan *stemming* dokumen B

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
B-00	B-201.txt	83.69	3.69	3.55
	B-202.txt	73.69	6.31	
	B-203.txt	75.50	4.50	
	B-204.txt	80.43	0.43	
	B-205.txt	77.20	2.80	
	B-401.txt	54.95	5.05	3.84
	B-402.txt	63.54	3.54	
	B-403.txt	62.82	2.82	

	B-404.txt	61.22	1.22	
	B-405.txt	66.57	6.57	
	B-601.txt	5.88	34.12	31.54
B-00	B-602.txt	7.93	32.07	
	B-603.txt	10.67	29.33	
	B-604.txt	5.88	34.12	
	B-605.txt	11.96	28.04	
	B-AP.txt	100.00	0.00	0.00
	B-KB.txt	82.54	2.54	2.54
	B-TG.txt	98.54	1.46	1.46
	B-TK.txt	100.00	0.00	0.00

Tabel 5.14 Hasil pegujian prosentase *error* dengan *stemming* dokumen C

Dokumen Asli	Dokumen Uji	Similarity(%)	Prosentasse Error (%)	Rata – Rata (%)
C-00	C-201.txt	80.91	0.91	3.38
	C-202.txt	85.85	5.85	
	C-203.txt	83.70	3.70	
	C-204.txt	85.61	5.61	
	C-205.txt	80.84	0.84	
	C-401.txt	62.62	2.62	5.38
	C-402.txt	64.54	4.54	
	C-403.txt	65.31	5.31	
	C-404.txt	67.74	7.74	
	C-405.txt	66.69	6.69	
	C-601.txt	7.95	32.05	27.25
	C-602.txt	10.44	29.56	
	C-603.txt	15.78	24.22	
	C-604.txt	17.88	22.12	
	C-605.txt	11.70	28.30	

	C-AP.txt	99.96	0.04	0.04
	C-KB.txt	74.84	5.16	5.16
	C-TG.txt	95.48	4.52	4.52
	C-TK.txt	100.00	0.00	0.00

Hasil rata-rata perbedaan dari pengujian posentase *error non-stemming* dan dengan *stemming* dapat dilihat pada tabel 5.15.

Tabel 5.15 Hasil rata-rata prosentase *error*

Dok. Uji	Prosentase *error* non-*stemming* (%)	Prosentase *error* dengan *stemming* (%)
20	3.29	3.27
40	4.19	4.40
60	30.20	29.92
TK	0.00	0.00
AP	0.64	0.03
TG	2.04	4.14
KB	2.75	3.29

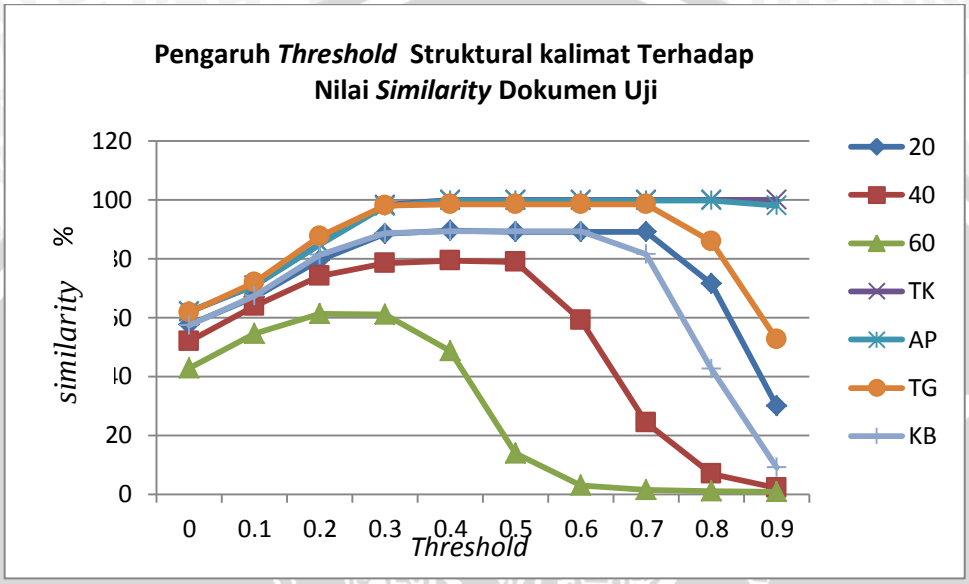
Hasil pengujian prosentase *error* menunjukkan bahwa pada dokumen dengan pemotongan kata 60% memiliki prosentase *error* yang cukup besar yaitu 30,20% untuk dokumen tanpa *stemming* dan 29,92% untuk dokumen menggunakan *stemming*, sedangkan prosentase *error* paling kecil terjadi pada dokumen uji penukaran susunan kalimat. Ada beberapa dokumen uji yang mengalami peningkatan *prosentase* error yaitu dokumen uji kesalahan pengejaan kata (TG) dan dokumen uji perubahan total (KB).

5.2 Pembahasan

Dari hasil pengujian sistem, berikut pembahasan dari tiap-tiap hasil pengujian pada subbab sebelumnya.

5.2.1 *Threshold* Struktural Kalimat

Dari hasil percobaan yang dilakukan terhadap seluruh data latih, didapatkan penggunaan *threshold* struktural kalimat lebih optimal pada nilai di bawah pemotongan kata. Maksudnya, jika pemotongan kata sebanyak 60% maka *threshold* yang optimal adalah di bawah 0,4 atau 40%.



Grafik 5.1 Pengaruh *threshold* struktural kalimat terhadap nilai *similarity* dokumen uji

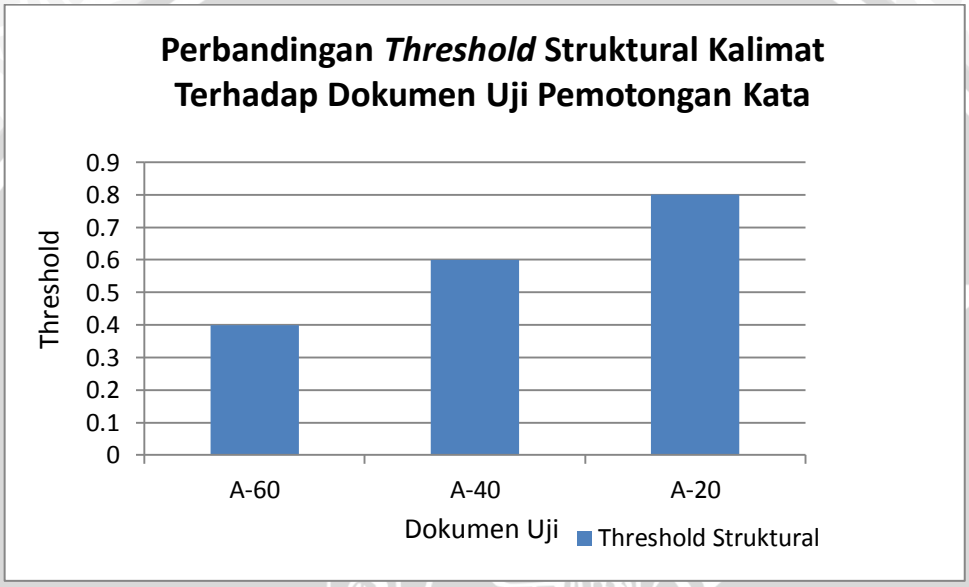
Untuk dokumen uji dengan perubahan susunan kalimat (TK), perubahan kalimat aktif menjadi pasif dan juga sebaliknya (AP), dan dokumen dengan kesalahan ejaan kata (TG) relatif memiliki nilai *similarity* ±100% yang sama hal ini dikarenakan pada dokumen pengujian tersebut tidak dilakukan pemotongan jumlah kata. Dokumen uji TG mengalami penurunan nilai *simiarity* pada *threshold* di atas 0.8, dikarenakan pada dokumen pengujian tersebut terdapat kesalahan pengejaan kata sebesar 20% untuk setiap kalimat yang berakibat penurunan nilai kesamaan struktural kalimat. Berikut contoh dari penurunan tersebut.

Kalimat Asli : Penggunaan *stemming* akan berpengaruh pada hasil nilai *similarity*.

Kalimat Uji (TG) : Penggwnaan *stemming* akam berpengaruh pada hasil nilai *similarity*.

Dari kedua kalimat diatas dapat dihitung jumlah kata yang sama adalah 6 dan jumlah kata total adalah 8. Sehingga hasil nilai kesamaan strukutral kalimat adalah 6/8 atau 0.75.

Berbeda pada dokumen uji perubahan total (KB), similarity mengalami penurunan pada *threshold 0.8* dikarenakan pada dokumen perubahan total tersebut dikurangi 20% kata dari jumlah kata total. Dari hasil pengujian tersebut dapat disimpulkan seperti pada grafik 5.2.

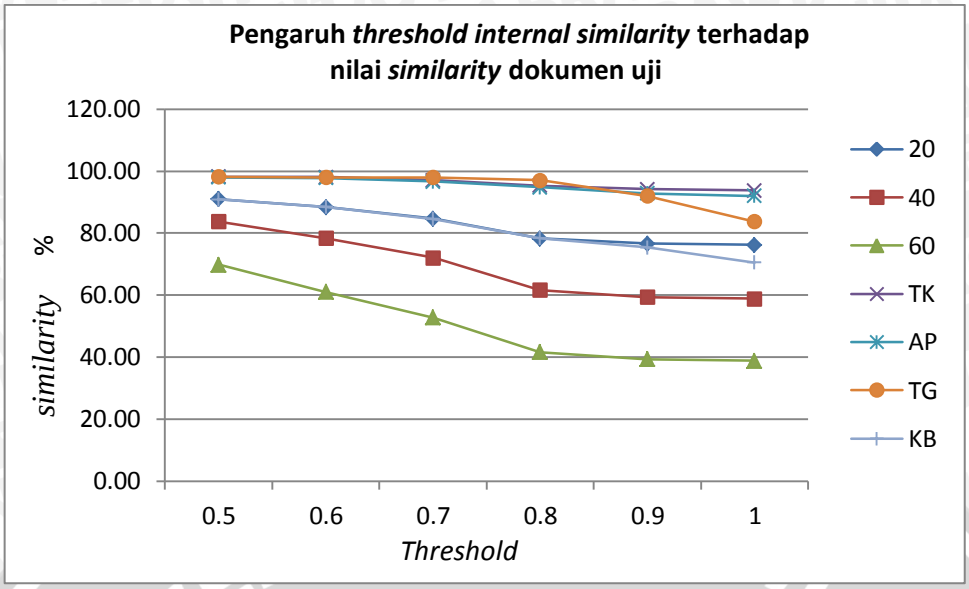


Grafik 5.2 Pebandingan *threshold* struktural kalimat terhadap dokumen uji pemotongan kata

Jika nilai *threshold* yang dimasukkan diatas nilai pemotongan kata, maka hasil *similarity* akan turun drastis dikarenakan banyak kalimat yang akan tersaring atau tidak diproses ke proses selanjutnya.

5.2.2 Threshold Internal Similarity

Pada Grafik 5.3 dapat dilihat bahwa semakin tinggi nilai *threshold internal*, hasil dari nilai *similarity* akan mendekati nilai yang diharapkan.

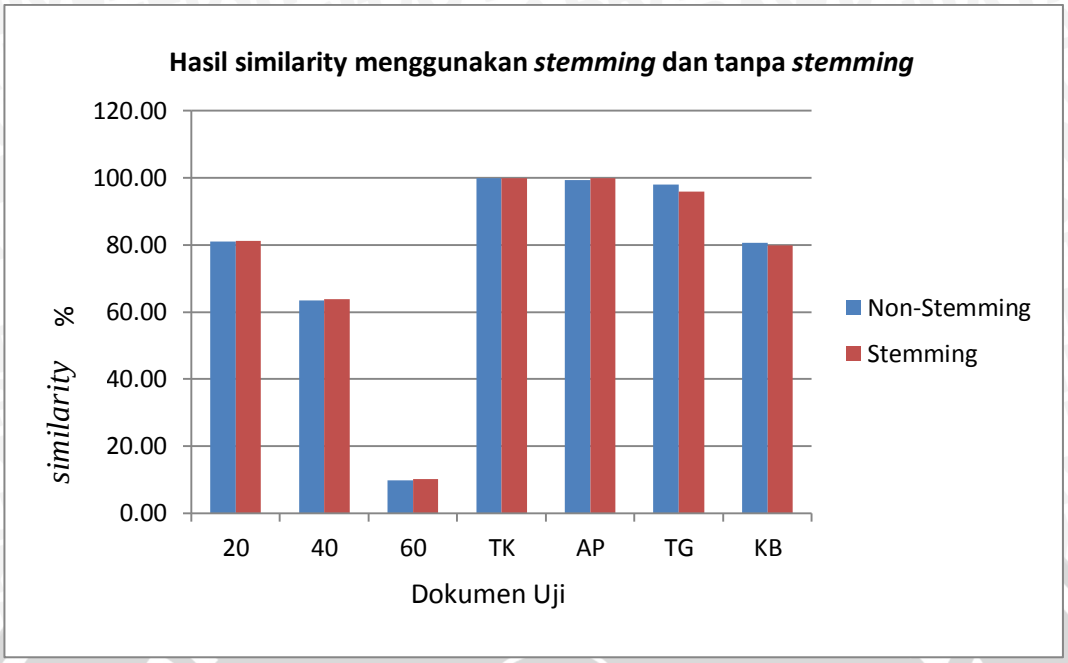


Grafik 5.3 Pengaruh *threshold internal similarity* terhadap nilai *similarity* dokumen uji .

Nilai similarity yang diharapkan adalah nilai yang sesuai dengan jumlah pemotongan kata pada dokumen uji. Untuk data uji TG atau kesalahan pengejaan kata, hasil mengalami penurunan nilai di bawah nilai yang diharapkan (100%) ketika *threshold* diatas 0.8 yang diakibatkan banyak kata pada kesalahan pengejaan diberi nilai nol pada sistem.

5.2.3 Pengaruh *Stemming* pada *Similarity* Dokumen

Selanjutnya pada pengujian similarity dokumen menggunakan *stemming* atau tanpa *stemming* didapatkan hasil yang hampir sama untuk data uji pemotongan ktaa 20%, 40%, dan 60%. Pada dokumen uji kesalahan pengejaan kata terjadi penurunan nilai *similarity* dikarenakan dimungkinkan kesalahan pengejaan kata terjadi pada awalan atau imbuhan. Contoh pada kata “keamanan”, jika kesalahan pengejaan kata menjadi “kramanan”, maka kata tersebut akan gagal untuk dilakukan *stemming* menjadi kata dasar. Hasil nilai *similarity* untuk kata “aman” dan “kraman” (setelah dilakukan *stemming)* akan turun.



Grafik 5.4 Hasil similarity menggunakan *stemming* dan tanpa *stemming*

Pada pengujian dengan dokumen uji pengubahan kalimat aktif menjadi pasif begitu pula sebaliknya (AP), nilai *similarity* menjadi naik yang berarti proses *stemming* berjalan dengan baik. Kenaikkan nilai ini dipengaruhi oleh banyaknya kalimat yang dirubah dari kalimat aktif menjadi pasif atau sebaliknya.

Pada dokumen uji kesalahan pengejaan kata (TG) terjadi penurunan nilai *similarity* dikarenakan adanya kata-kata yang gagal dilakukan *stemming*. Kata-kata tersebut adalah kata yang memiliki kesalahan pengejaan pada awalan atau imbuhan.

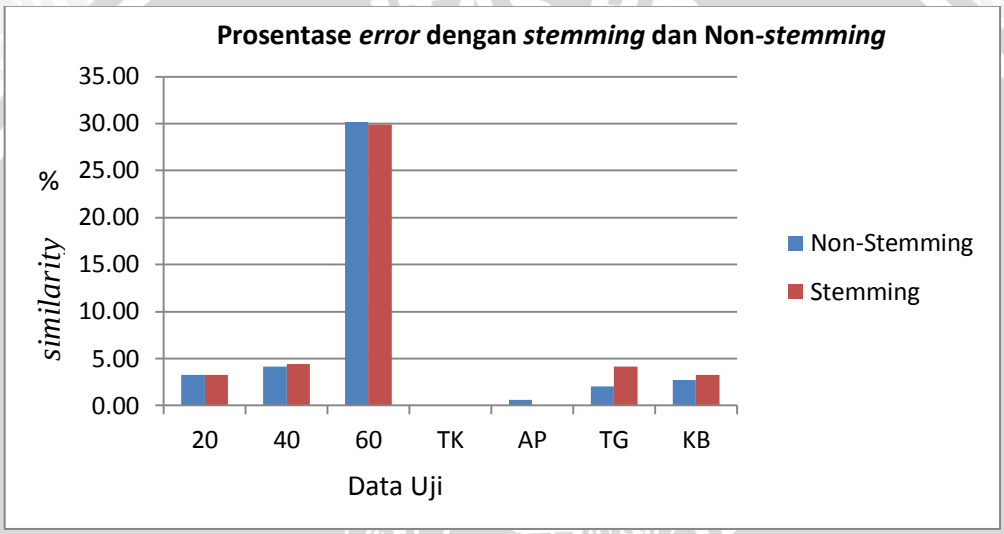
Dokumen uji dengan pemotongan jumlah kata juga terjadi perbedaan nilai *similarity* setelah dilakukan proses *stemming*. Hal ini dikarenakan setelah dilakukan proses *stemming* kata yang tadinya bernilai *similarity* tinggi mejadi bernilai *similarity* rendah. Contoh kata "berjilbab" dan "bercorak", kedua kata tersebut memiliki imbuhan ber- yang sama, sehingga nilai Jaro-Winkler *distance* akan meningkat karena pada algoritma Jaro-Winkler ada penambahan nilai untuk *prefix* yang sama. Berbeda jika kata tersebut telah dilakukan *stemming*. Kata "jilbab" dan "corak" akan memiliki nilai *similarity* kata yang lebih rendah.

Untuk nilai *similarity* yang naik setelah proses *stemming*. Ada beberapa kata yang memiliki nilai *similarity* lebih tinggi setelah dilakukan *stemming*.

Contoh kata "diberi" dan "menteri", kedua kata tersebut setelah dilakukan *stemming* menjada kata "beri" dan "teri" yang memiliki nilai *similarity* yang cukup tinggi.

5.2.4 Pengaruh *Stemming* Pada Prosentase *Error*

Dengan dilakukan proses *stemming* terlebih dahulu, hasil akhir nilai *similarity* akan mengalami perubahan. Hal ini juga akan berpengaruh pada prosentase *error* dari hasil pengujian. Grafik rata-rata prosentase *error* dari hasil pengujian yang telah dilakukan dapat dilihat pada Grafik 5.5.



Grafik 5.5 Prosentase *error* dengan *stemming* dan Non-*stemming*

Pada Grafik 5.5 prosentase *error* mengalami penurunan pada dokumen uji AP dengan menggunakan *stemming*. Untuk dokumen kesalahan pengejaan kata dan dokumen ubah total, terjadi peningkatan prosentase *error*, dikarenakan kesalahan pengejaan kata terjadi pada awalan atau imbuhan yang mengakibatkan kata gagal dirubah menjadi kata dasar.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil pengujian dan pembahasan yang telah dilakukan dapat diambil kesimpulan sebagai berikut :

1. Penerapan algoritma Monge Elkan *distance* dalam mendeteksi plagiarisme teks dilakukan dengan menggunakan pola *word for word plagiarism* atau pencocokan kata per kata dari seluruh teks pada dokumen. Algoritma Monge Elkan *distance* termasuk algoritma *hybrid distance* yang memiliki *internal string matching*, pada penlitian ini *internal string matching* yang digunakan adalah Jaro-Winkler *distance*. Jaro-Winkler *distance* berfungsi untuk menghitung kesamaan setiap kata pada kalimat yang dicocokkan. Hasil dari Jaro-Winkler *distance* selanjutnya dihitung menggunakan algoritma Monge-Elkan *distance* yang mengambil nilai tertinggi pada setiap hasil dari Jaro-Winkler *distance*.
2. Penggunaan *stemming* berpengaruh pada nilai prosentase *error* yang dihasilkan. Hasil dari pengujian yang dilakukan didapatkan penurunan prosentase *error* terbesar setelah dilakukan proses *stemming* terjadi pada dokumen uji perubahan kata kerja aktif menjadi pasif dan juga sebaliknya (AP) dengan penurunan nilai sebesar 0,61%. Hasil tersebut diperoleh saat mengunakan *threshold* struktural kalimat yang optimal sebesar 0,5 (nilai kesamaan stuktural kalimat di atas 50%) dan *threshold internal similarity* yang optimal sebesar 0,8 (nilai kesamaan kata di atas 80%). Penurunan prosentase *error* tersebut menunjukkan bahwa proses *stemming* berjalan baik pada dokumen uji AP.

6.2 Saran

Saran dari hasil penelitian teah dilakukan untuk penelitian lebih lanjut, disarankan ditambahkan variasi data pengujian serta aplikasi tidak hanya dapat mendeteksi kalimat dengan stuktural kalimat berbeda, tetapi juga dapat mengenali makna kalimat yang sebenarnya.

DAFTAR PUSTAKA

[ANK-08] Kurniawati, Ana., Simri Wicaksana, I Wayan. 2008. *Perbandingan Pendekatan Deteksi Plagiarism Dokumen Dalam Bahasa Inggris.* Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma.

[CSK-13] Aditya, Christian Sri Kusuma.2013. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Menggunakan Algoritma Damerau Levenshtein Distance.*Program Studi Ilmu Komputer, Universitas Brawijaya.

[FAZ-03] Talla, Fadillah Z.2003. *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Universiteit van Amsterdam.

[GME-09] Jimenez, Sergio., Becerra Claudia., Gelbukh, Alexander., Gonzalez, Fabio.2009. *Generalized Mongue-Elkan Method for Approximate Text String Comparison.* Springer-Verlag Berlin Heidelberg.

[KUR-10] Kurniawati, Anna., Puspitodjati, Sulistyo., dan Rahman, Sazali. 2010, *Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia*.Universitas Gunadarma.

[LEA-09] Agusta, Ledi. 2009. *Perbandingan Algoritma Stemming Porter Dengan Algoritma Nazief & Adriani Untuk Stemming Dokumen Teks Bahasa Indonesia*. Fakultas Teknologi Informasi. Universitas Kristen Satya Wacana.

[NUG-11] Nugroho, Eko. 2011. *Perancangan Sistem Deteksi Plagiarisme Dokumen Teks Dengan Menggunakan Algoritma Rabin-Karp.* Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijyaya. Malang

[RAK-09] Kaur, Ranverr., Aggarwal, Shruti. 2013. *Techniques for Mining Text Documents. International Journal of Computer Apllications,* Volum 66-No.18, Maret 2013.

[SOK-13] Khatami, Solmaz.2013. *Comparison and Improvement of Basic String Metrics for Surname Matching*. Faculty of Engineering. Islamic Azad University of sarvestan. Iran.

[SDS-07] Sudigdo, Sastroasmoro.2007. *Beberapa Catatan tentang Plagiarisme*. Majalah Kedokteran Indonesia, Volum : 57, Nomor: 8, Agustus 2007.

[TAB-00] A. Basuki, Thomas. 2000. Pengenalan Suku Kata Bahasa Indonesia Menggunakan Finite-State Automata. *Integral*, Vol.5 no.2 Oktober 2000.

[USU-09] Sutisna, Utis.2009. *Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritme Damerau Levenshtein*. Fakultas Matematika dan Ilmu Pengetahuan Alam, Departemen Ilmu Komputer Institut Pertanian Bogor.

[WIL-03] W. Cohen, William., Ravikumar, Pradepp., E.Fienberg, Stephen.2003. *A Comparison of String Distance Metrics for Name-Matching Tasks.*American Association for Artificial Intelligence.

LAMPIRAN

Lampiran 1 : Data Asli, halaman 90-94

Data Asli A

Jakarta - Kongres Muslimah Indonesia resmi dibuka pada Sabtu (8/3/2014) siang di Istana Bogor, Jawa Barat. Kongres ini menjadi kegiatan yang pertama kali setelah proklamasi didengungkan.

"Perlu kami laporkan kongres ini dihadiri 300 cendekiawan muslimah. Kongres ini baru pertama kali dilaksanakan setelah Indonesia merdeka," tutur Ketua Panitia Kongres Muslimah Indonesia, Welya Safitri yang mengenakan baju dan kerudung berwarna biru muda.

Welya mengatakan, kemajuan ilmu pengetahuan dan teknologi tak dapat dipungkiri banyak mendatangkan manfaat bagi masyarakat secara keseluruhan. Tetapi ia mengingatkan, agar hal itu tak membuat orang menjadi terlena hingga mengenyampingkan nilai-nilai keagamaan di dalamnya.

"Kemajuan iptek tentu sangat meningkatkan dunia, dalam hal ini harus diimbangi dengan nilai-nilai moral agar tidak kehilangan. Muslimah sebagai bagian dari masyarakat harus ambil peran. Jangan hanya menonton dan berpangku tangan. Akhlak yang tangguh sangat penting bagi kehidupan berbangsa dan bernegara," lanjutnya.

Menurut perempuan yang peduli akan nasib muslimah ini, banyak kaum perempuan secara tidak sadar mereka dieksploitasi secara ekonomi. Dalam arti mereka bekerja tetapi hasil yang didapatnya seringkali masih belum sesuai.

"Kita perlu banyak belajar dari Jepang. Mereka maju tapi juga masih mau belajar dari luar. Ke depannya, kita berharap setiap 8 Maret menjadi Hari Muslimah Nasional bertepatan juga dengan Hari Perempuan Internasional," kata Welya.

Kegiatan ini dihadiri oleh 300 orang yang didominasi perempuan berjilbab putih dan selendang biru bercorak di hall utama Gedung Induk sayap kanan, Istana Kepresidenan Bogor, Jawa Barat. Kegiatan ini juga dihadiri oleh Wakil Presiden RI Boediono beserta isteri dan organisasi muslimah yang akan ikut antara lain Muslimat NU, Aisyiyah Muhammadiyah, MUI, ICMI dan HMI.

Sebelumnya, Presiden Susilo Bambang Yudhoyono dan Menteri Agama RI Suryadharma Ali dijadwalkan hadir dalam acara ini. Namun presiden berhalangan, sehingga diwakilkan oleh Boediono.

Data Asli B

Merpati Nusantara Airlines (MNA) masih harus memendam hasratnya "mengepakkan sayap" sampai ke Jeddah, bulan ini. Pasalnya, maskapai perintis tersebut belum juga memeroleh izin terbang dari Kementerian Perhubungan (Kemenhub) sejak sertifikat terbangnya atau air operator certificate (AOC) dibekukan awal Februari lalu.

"Sampai sekarang MNA belum jelas business plan-nya sehingga belum bisa diberi izin," kata Plt Kapuskom Kemenhub, Bambang S Ervan, kepada Kompas.com, Minggu (9/3/2014).

Sebelumnya, dalam sebuah jumpa pers, Direktur Utama Merpati, Asep Ekanugraha mengatakan bahwa dari sekian langkah penyelamatan Merpati, umroh flight atau penerbangan ke Jeddah inilah yang paling mendekati realisasi.

Penerbangan ke Jeddah ditargetkan beroperasi pada akhir Februari 2014 atau paling lambat awal Maret 2014. Ini bukan kali pertama Merpati menerbangi luar negeri. Sebelumnya, Merpati sudah pernah melayani penerbangan Denpasar-Los Angeles, Manila, dan Australia.

"Jadi potensi penerbangan ke Jeddah ini bukan dalam rangka sok-sokan. Itu potensi KSO (kerjasama operasi) yang bisa menggerakkan roda Merpati. Kita juga bisa menambahkan pesawat feeder. Finalisasi KSO penerbangan umroh ini ada dalam tahap akhir," terang Asep, pertengahan Februari lalu.

Menanggapi restu dari Kemenhub yang tak kunjung diberikan, Menteri BUMN Dahlan Iskan, hanya mengatakan bahwa dirinya berfikir dengan logika. Jika izin penerbangan domestik saja tidak diberikan, apalagi penerbangan lintas negara (internasional).

"Sama dengan tidak diberi izin," sesal Dahlan.

Namun demikian, mantan Dirut PLN itu punya alasan mengapa Kemenhub perlu memberi izin Merpati terbang ke Jeddah. Tidak seperti penerbangan domestik, di mana kini tinggal dilayani 3 unit pesawat turboprop, penerbangan Jeddah akan menggunakan pesawat KSO, dan seluruh operasionalnya akan ditanggung mitra Merpati.

Data Asli C

Indonesia masuk dalam delapan penerima penghargaan terbaik untuk peserta paling mengesankan dalam Bursa Pariwisata Internasional atau Internationale Tourismus-Borse (ITB) Berlin 2014 dari 11.000 stan yang ikut dalam pameran tersebut.

Deputi Direktur Promosi Internasional untuk Eropa Kementerian Pariwisata dan Ekonomi Kreatif (Kemenparekraf) Agustini Rahayu kepada Antara London, Senin (10/3/2014), mengatakan kriteria yang dinilai antara lain adalah kreativitas, orisinalitas, kualitas layanan, keberlanjutan dan efek khusus.

Agustini mengatakan tahun 2013 Indonesia berperan besar sebagai negara mitra resmi pada penyelenggaraan ITB Berlin tahun 2013. ITB Berlin di Jerman merupakan pameran pariwisata terbesar di dunia untuk kawasan Asia, Australia, Oceania.

Agustini mengatakan Indonesia untuk ke-48 kalinya kembali tampil dalam ITB 2014 di Berlin yang berlangsung 5-9 Maret dengan mengusung kapal Phinisi dalam upaya menjaga konsistensi branding yang ditampilkan tahun 2013.

Wamenparekraf Sapta Nirwandar didampingi Dubes RI di Berlin Fauzi Bowo dan Direktur Jenderal Pemasaran Pariwisata Kemenparekraf Esthy Reko Astuty memberikan informasi kepada lebih dari 80 jurnalis internasional dalam Indonesia Press Conference yang diadakan pada hari pertama ITB Berlin.

Disampaikan selain 16 Kawasan Strategis Pariwisata Nasional, Indonesia memperkenalkan lebih banyak lagi destinasi yang dapat dikunjungi wisatawan terutama jenis wisata adventure atau petualangan dan "niche market".

Kemenparekraf memfasilitasi 88 industri pariwisata yang terdiri dari kalangan perhotelan, biro perjalanan dan asosiasi bidang pariwisata lain menjual berbagai obyek wisata.

Selain itu sejumlah pemerintah daerah (pemda) juga mempromosikan Indonesia di ITB Berlin pemda Jawa Barat, Jawa Tengah, DKI Jakarta, Bali, Nusa Tenggara Barat, Nusa Tenggara Timur, Kalimantan Timur, Sulawesi Utara dan Papua.

Menurut Agustini Rahayu, pada 2014, desain paviliun dibuat lebih terbuka dan bernuansa serba putih berkesan terbuka dan pengunjung lebih nyaman untuk datang.

Kapasitas lahan utama paviliun Indonesia sebesar total sebesar 476 m2 di lantai bawah dengan tambahan 90 m2 di lantai atas dan komposisi 410 m2 untuk aktivitas bisnis di lantai bawah dan 90 m2 pada lantai atas. Sementara 66 m2 yang letaknya di seberang paviliun utama digunakan untuk area kuliner, spa dan meja informasi umum untuk pemda.

Data Asli D

"Plagiarisme akademik merupakan tindakan tercela yang mencoreng nama baik pendidikan kita. Karena itu, para plagiatornya harus diberi sanksi keras. Selain menghukum para penjiplaknya, sanksi itu juga dimaksudkan memberi efek jera kepada para peserta didik agar tidak melakukan perbuatan tercela tersebut di masa-masa mendatang. Namun, pemberikan sanksi tidaklah cukup. Perlu ada regulasi dan tindakan nyata secara kolektif dari para stakeholder pendidikan untuk mencegah tindakan plagiarisme dan kejahatan akademik sejenisnya". Demikian intisari dari tanggapan para pembaca setelah membaca tulisan saya "Plagiarisme Akademik" yang diterbitkan koran ini (Jawa Pos Radar Semarang, 25 Juni 2012). Tulisan itu juga saya upload di Facebook saya dan di salah satu milis masyarakat Flores (NTT). Tujuannya, memberikan pencerahan kepada masyarakat luas, khususnya generasi muda, agar tidak melakukan perbuatan tidak terpuji tersebut. Seperti telah disinggung dalam tulisan sebelumnya, kejahatan akademik berupa plagiarisme sudah menjadi fenomena umum dalam pendidikan kita. Motif dan faktor pemicunya sangat kompleks. Karena itu, solusi untuk mencegahnya harus dilakukan secara sistematis, terintegrasi, komprehensif dan berkesinambungan serta memerlukan komitmen bersama dari semua pihak. Tulisan ini memfokuskan pembahasan pada pemicu dan solusi mencegah plagiarisme. Pemicu plagiarisme Plagiarisme atau penjiplakan karya tulis orang lain dalam dunia akademik dipicu oleh banyak faktor. Faktor pragmatisme mahasiswa dan dosen, lemahnya peraturan, sistem kontrol dan kualitas SDM, industrialisasi pendidikan, lemahnya regulasi dan inkonsistensi kontrol pemerintah dituding sebagai faktor pemicu utamanya. Secara khusus, saya mencatat paling sedikit ada tiga faktor pemicunya. Pertama, penulis (mahasiswa) ingin segera menyelesaikan skripsi, tesis atau

disertasinya agar bisa meraih gelar akademik secepatnya tanpa harus bekerja keras sesuai proses riset dan penulisan ilmiah yang benar. Di sisi lain, dosen pembimbing tidak teliti dalam proses pembimbingan. Kebanyakan dosen pembimbing tidak mau repot membimbing mahasiswa. Mereka hanya berorietansi pada produk skripsi, tesis atau disertasi yang dihasilkan mahasiswa. Demikian pula ketika diuji, para pengujinya juga tidak mau repot mengecek apakah karya mahasiswa yang diuji asli dan sudah bebas dari dosa-dosa etika ilmiah akademik. Akibat sangat longgarnya proses akademik tersebut, banyak skripsi, tesis dan disertasi lolos saringan akademik dan para mahasiswa plagiatornya dinyatakan lulus sarjana. Pasca lulus sebagai sarjana (S1, S2, S3), mereka biasanya bangga karena bisa "mengibuli" dosen pembimbingnya. Biasanya untuk memperlancar proser pembimbingan dan ujian, mahasiswa plagiator memberi "kado spesial" untuk dosen pembimbing dan penguji agar dipermudah proses bimbingan dan kelulusan. Pasca lulus sarjana, sarjana plagiator biasanya merasa sudah "nyaman" perbuatan curangnya tidak akan diketahui pihak lain. Kalaupun nantinya diketahui, tidak ada konsekuensi apapun yang bakal diterima karena sudah lulus. Asumsi itu salah besar. Seiring dengan kemajuan teknologi informasi dan tuntutan keterbukaan informasi akademik, bisa jadi suatu skripsi, tesis dan disertasi yang tercela akan digugat keabsahannya. Risiko terburuknya, selain dicabut gelarnya, sarjana plagiator bisa dipenjarakan. Kedua, penulis (dosen) ingin segera naik jabatan fungsional akademik dan golongan gajinya sehingga bisa menikmati kenaikan tunjangan dan insentif yang besar. Apalagi dalam beberapa tahun terakhir, pemerintah memberi tunjangan profesi kepada para guru dan dosen dalam jumlah yang lumayan besar. Karena motif tersebut, sejumlah dosen lalu menghalalkan segala cara dalam menghasilkan karya-karya akademik dengan melanggar norma-norma akademik. Misalnya, menjiplak karya tulis orang lain (plagiat), melakukan fabrikasi atau mengarang data penelitian yang sebenarnya tidak ada, atau melakukan falsifikasi dalam penulisan karya tulis. Mereka berasumsi, perbuatan curangnya tidak akan diketahui atau dideteksi orang lain. Penulis lupa seiring dengan kemajuan teknologi informasi dan keterbukaan informasi serta adanya keharusan mendigitalisasi dan mengupload karya-karya akademik bagi para guru dan dosen yang mau mengajukan kenaikan jabatan

fungsional akademik maka semua karya akademik akan bisa diakses oleh publik dimana saja dan kapan saja. Dengan begitu, perbuatan curang yang dilakukan para dosen nakal cepat atau lambat pasti akan terungkap dan akan berakibat fatal bagi dirinya. Lolosnya sejumlah karya akademik dosen yang mengandung unsur pelanggaran etika akademik terutama disebabkan karena lemahnya mekanisme kontrol dan penelahaan kritis oleh pengelola jurnal, penerbit, dan institusi pendidikan itu sendiri. Banyak perguruan tinggi juga tidak begitu peduli dan bahkan tidak paham dengan etika penulisan ilmiah. Para mahasiswa dan dosennya dibiarkan melakukan penelitian dan penulisan ilmiah sesuai dengan kehendak masing-masing. Dari pengalaman saya mereview banyak artikel, paper, proposal dan hasil riset, serta berinteraksi dengan banyak dosen dari berbagai perguruan tinggi, ternyata banyak dosen belum memahami etika penulisan ilmiah akademik, terutama terkait plagiasi. Ketiga, penulisnya ingin agar terkenal luas atau tetap terkenal dengan menghasilkan banyak buku, menerbitkan banyak artikel ilmiah, menulis artikel populer di media massa, menghasilkan banyak penelitian dan karya-karya akademik lainnya. Tujuan untuk mendapatkan uang atau insentif dana yang banyak. Karena terdorong oleh motif tersebut, penulis lalu menghalalkan segala cara dalam menulis dan meneliti dengan melanggar norma-norma ilmiah akademik. Dalam sejumlah kasus, tindakan plagiasi justru dilakukan oleh para penulis senior yang sudah terkenal dan para dosen senior yang sudah bergelar doktor dan profesor yang telah memiliki banyak karya tulis. Mereka biasanya melakukan plagiasi parsial, plagiasi terhadap karya sendiri (auto-plagiarism) dan plagiasi antarbahasa. Dalam banyak kasus, para penulis seringkali tidak menyadari apabila karya tulis mereka mengandung unsur-unsur plagiasi. Solusi pencegahan Menurut hemat saya, solusi terhadap tindakan plagiarisme akademik mesti didasarkan pada faktor pemicu utamanya. Pertama, dalam banyak kasus plagiarisme akademik ditemukan para pelakunya ternyata tidak mengetahui bahwa tindakan mereka menyontek hasil karya orang lain menjadi hasil karya sendiri adalah sesuatu yang dilarang. Selain itu, dosen pembimbingnya juga tidak memperingatkan atau mempermasalahannya. Dosen pembimbing ternyata juga tidak paham membedakan mana karya plagiasi dan mana yang bukan karya plagiasi. Bahkan, dalam beberapa kasus plagiarisme sejumlah dosen menyarankan

kepada mahasiswa bimbingannya untuk mengambil tulisan dari suatu jurnal atau working paper untuk dijadikan topik skripsi atau tesis dengan sejumlah pengembangan pada variabel atau metode. Akibatnya, hampir 50 hingga 80 persen tulisan dalam skripsi atau tesis sama persis dengan sumber aslinya. Ternyata dosen penganjur atau pembimbing tidak tahu bahwa hal itu tidak diperbolehkan. Jika hal itu memang benar-benar menjadi realitas "buruk" dalam kehidupan dunia pendidikan kita, maka solusi untuk menghindari plagiasi adalah dengan melakukan proses edukasi secara berkelanjutan tentang norma-norma atau etika penulisan ilmiah kepada para dosen dan mahasiswa, serta kepada para pengelola perguruan tinggi di seluruh Indonesia. Bagi para pelanggarnya sebaiknya tidak diberikan sanksi berat. Proses sosialisasi menjadi sangat krusial dan mendesak. Tujuannya, untuk meningkatkan pengetahuan dan menyamakan persepsi kepada para pelaku pendidikan tentang norma-norma atau etika penulisan ilmiah akademik. Dengan begitu maka tindakan plagiarisme akademik yang dilakukan para mahasiswa dan dosen sendiri akan dapat dicegah atau diminimalisir sekecil mungkin di masa mendatang. Kedua, dalam banyak kasus plagiarisme akademik juga ditemukan para pelakunya paham tentang plagiarisme tapi mereka sengaja melakukannya karena tidak paham cara menulis yang baik dan benar, tidak diberikan ide-ide unik, baru dan menarik dari dosen pembimbingnya sehingga bingung, serta ingin cepat lulus. Para dosen yang melakukan tindakan plagiarisme juga ternyata sadar bahwa perbuatan mereka seharusnya tidak boleh dilakukan. Namun karena ingin naik jabatan dan mendapatkan tunjangan profesi dari pemerintah, mereka terpaksa melakukannya. Jika demikian kenyataannya, maka proses penyadaran dan sosialisasi etika penulisan ilmiah akademik serta kontrol yang ketat dan memberian sanksi yang tegas dan keras kepada para plagiator menjadi solusi yang sangat tepat untuk menghidari dunia akademik dari praktik-praktik plagiasi dan lainnya. Pemerintah melalui Permen Diknas No 17 Tahun 2010 tentang Pencegahan Plagiasi dan Sanksi Plagiasi sebenarnya sudah memberikan arahan yang jelas tentang etika penulisan ilmiah akademik dan sanksi-sanksi yang bakal diberikakan kepada para individu yang melakukan plagiasi. Namun demikian, saya mengusulkan agar ke depan para pelaku dunia pendidikan di Indonesia berkolaborasi dan bergandengan

tangan melakukan aksi-aksi bersama untuk mencegah plasgiasi dan kejahatan-kejahatan akademik lainnya. Dengan begitu, citra dunia pendidikan kita akan meningkat kembali. Peran pendidikan tinggi untuk menghasilkan para SDM yang berkualitas dan unggul juga bisa terwujud.

Data Asli E

Beberapa waktu lalu, muncul lagi berita buruk yang mencoreng wajah dunia pendidikan Indonesia. Seorang dosen dari Universitas Islam Negeri (UIN) Syarif Hidayatullah Jakarta ketahuan melakukan tindakan kejahatan intelektual dalam penulisan artikel ilmiah. Ia melakukan plagiat atau penjiplakan karya skripsi mahasiswa bimbingannya untuk kenaikan jabatan fungsional akademik. Rektor UIN langsung memberi sanksi pemecatan sebagai dosen pada yang bersangkutan.

Sebelum muncul kasus itu, juga muncul sejumlah kasus plagiarisme yang membuat malu dunia akademik kita. Sejumlah dosen dari beberapa perguruan tinggi yang sedang mengajukan jabatan Lektor Kepala dan Guru Besar ke Dikti ketahuan melakukan tindakan plagiasi dalam sejumlah karya ilmiahnya. Dirjen Dikti mengembalikan berkas akademik dari para plagiator tersebut dan meminta pimpinan perguruan tingginya masing-masing untuk memberi sanksi tegas.

Beberapa waktu sebelumnya, masyarakat juga dikejutkan oleh perilaku tidak etis dari beberapa guru besar yang seharusnya menjadi teladan akademik. Seorang profesor dari salah satu PTS terkenal di Bandung melakukan plagiasi dalam penulisan artikel populer di koran nasional. Gelar profesornya dicopot. Ia juga dipecat sebagai dosen. Seorang profesor lainnya dari Sumatera juga ketahuan melakukan plagiasi dalam penulisan buku. Akibatnya, penerbit terpaksa menarik kembali buku itu dan yang bersangkutan juga dikenakan sanksi berat.

Walau tidak terpublikasi ke masyarakat, sejumlah pimpinan perguruan tinggi (PT) yang memiliki komitmen tinggi menegakkan etika akademik juga sudah banyak melakukan tindakan tegas kepada para mahasiswa, alumni dan dosennya yang terbukti melakukan plagiasi. Misalnya, terpaksa mencabut gelar sarjana kepada para alumni yang terbukti melakukan plagiasi karya orang lain dalam penulisan skripsi, tesis dan disertasi. Sejumlah dosen yang terbukti

melakukan pelanggaran etika ilmiah akademik juga diberikan sanksi tegas sesuai dengan kadar dan tingkat pelanggarannya.

1 Tulisan ini sudah dimuat harian Jawa Pos Radar Semarang, 25 Juni 2012.

Dari pengalaman saya menjadi reviewer pada sejumlah jurnal ilmiah nasional juga

ditemukan ada banyak artikel terindikasi mengandung karya plagiarisme dan melanggar etika ilmiah akademik. Kadar pelanggarannya ada yang tergolong berat, sedang dan ringan.

Biasanya untuk artikel yang mengandung pelanggaran berat dan sedang, saya langsung merekomendasikan untuk ditolak karena penulisan artikel tersebut sudah dilandasi oleh niat tidak jujur dari penulisnya. Sementara untuk artikel yang mengandung pelanggaran ringan tapi kualitas cukup baik, saya rekomendasikan untuk diterima dengan sejumlah catatan kritis untuk diperbaiki penulisnya.

Singkat kata, kejahatan intelektual berupa perbuatan plagiarisme sepertinya sudah menjadi fenomena umum dalam dunia pendidikan maupun dalam masyarakat kita. Ada banyak motif dibalik perbuatan tercela tersebut. Penyebab terjadinya kejahatan intelektual tersebut juga sangat kompleks. Karena itu, solusi dan agenda aksi untuk mencegah perbuatan yang tidak beretika tersebut perlu dilakukan secara sistematis, terintegrasi, komprehensif dan

berkesinambungan serta memerlukan komitmen bersama dari semua pihak.

Pada bagian pertama dari tulisan ini saya memfokuskan pembahasan pada esensi dan pola plagiarisme. Pada tulisan berikutnya saya akan memfokuskan pembahasan pada motif, pemicu dan solusi untuk mencegahnya.

Data Asli F

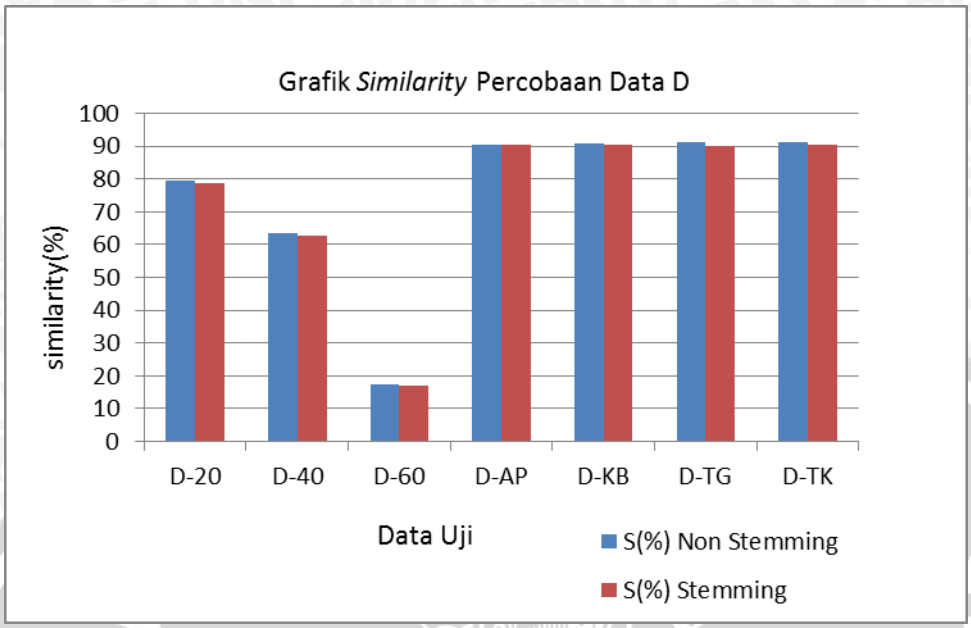
Dengan semakin berkembangnya teknologi informasi, sehingga membuat pembuatan karya tulis semakin mudah dan cepat. Hal tersebut dikarenakan informasi kini tersedia secara melimpah. Akan tetapi dikarenakan kemudahan dalam memperoleh informasi tersebut, pada pembuatan karya tulis sering ditemukan kesamaan dengan karya tulis orang lain sehingga kemudian menimbulkan isu plagiarisme.

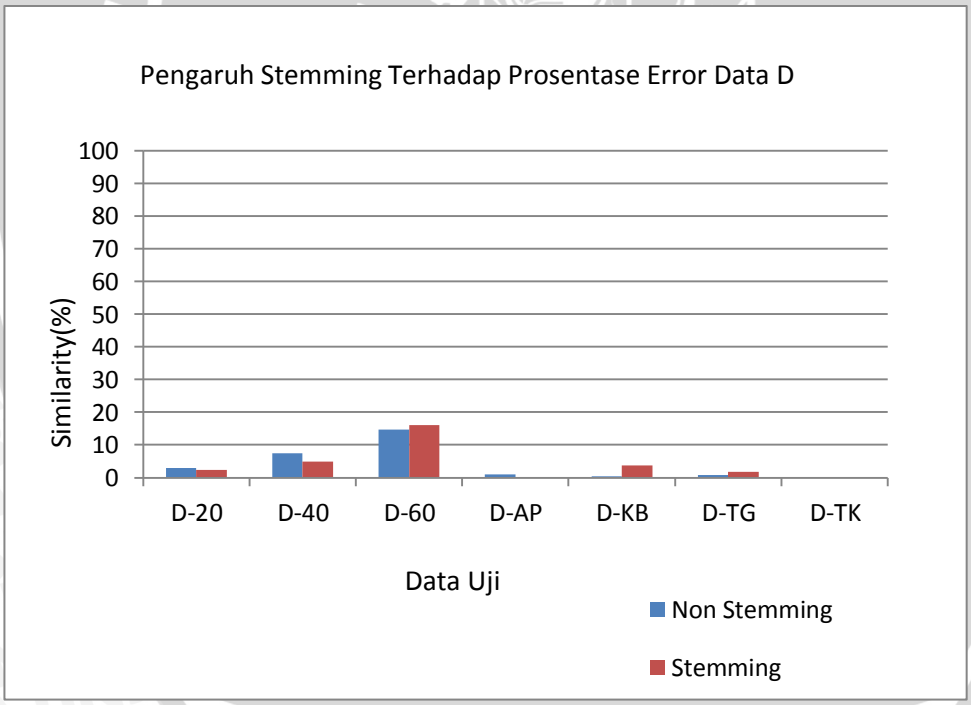
Aksi plagiat dalam karya tulis sangatlah mungkin terjadi, untuk itu perlu dilakukan upaya-upaya sebagai pencegahan maupun pendeteksian, sehingga dalam tulisan ini akan dibahas mengenai pendeteksian plagiarisme dari sebuah dokumen dengan membandingkannya dengan sebuah dokumen lainnya. Karya tulis selanjutnya akan disebut sebagai dokumen teks. Untuk mengetahui seberapa besar kesamaan suatu dokumen teks dengan dokumen teks lainnya dapat dengan menggunakan pendekatan string metric yaitu melakukan perbandingan string dengan memasukkannya ke dalam fungsi matematis tertentu.

Beberapa algoritma yang berdasarkan kepada string metric diantaranya adalah Levenshtein distance, TF/IDF, Needleman-Wunsch distance, Jaro-Winkler distance, dan sebagainya. Dari algoritma yang telah disebutkan di atas Jaro-Winkler distance memiliki ketepatan yang baik di dalam pencocokan string yang relatif pendek. Metode ini dipilih dikarenakan setelah dilakukannya proses tokenizing algoritma ini dapat secara akurat memeriksa salinan antar dokumen. Diharapkan dengan adanya sebuah aplikasi yang mampu mendeteksi kesamaan dokumen maka pada tulisan ilmiah tidak ditemukan lagi kesamaan yang begitu tinggi dengan dokumen lainnya.

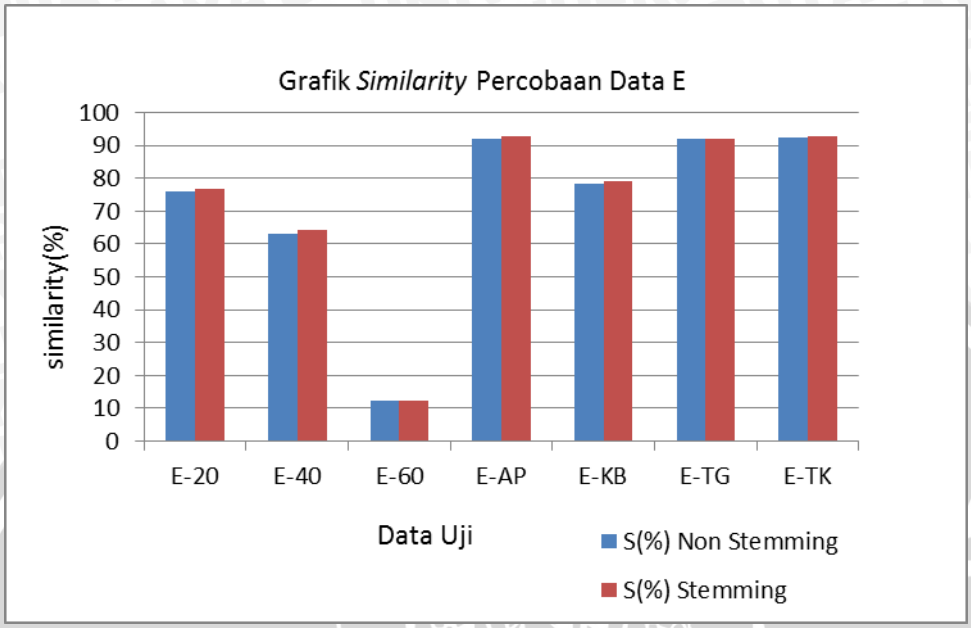
Lampiran 2 : Pengujian Data halaman 101-103

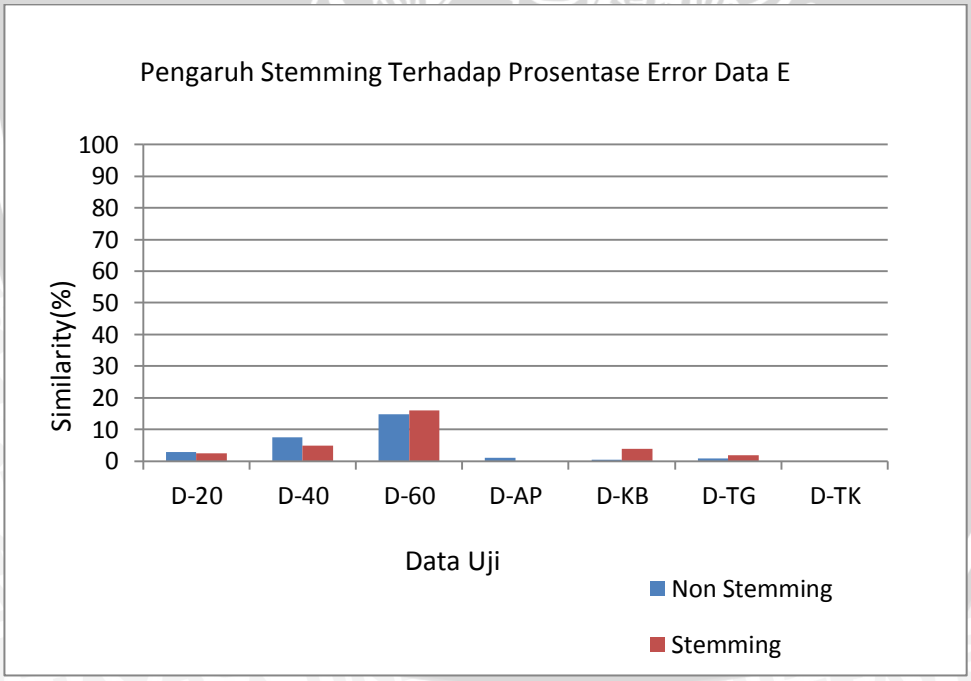
Grafik Pengujian data D



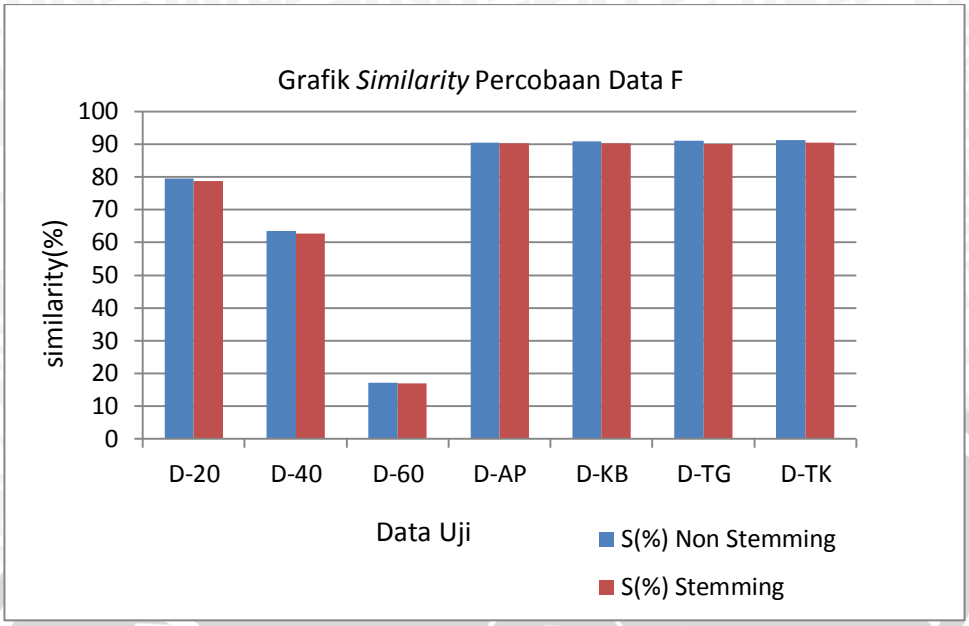


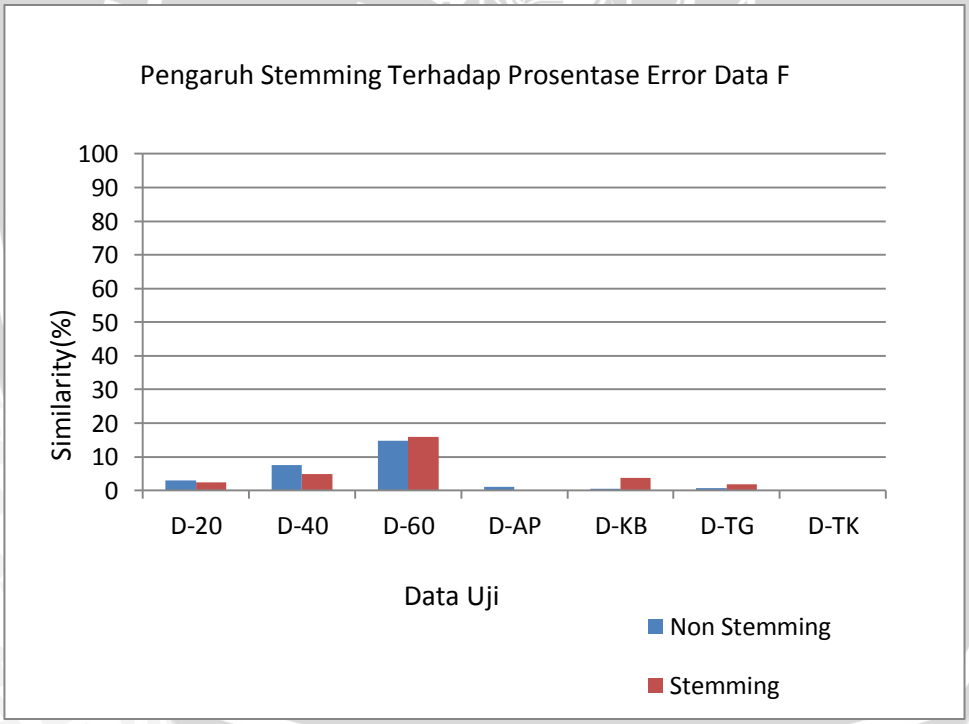
Grafik Pengujian Data E





Grafik Pengujian Data F





Lampiran 3 : Source Code jumlah suku kata, halaman 95-100

```
public class SukuKata {
    private String[][] ambilSuku=null;
    public int pecahKata(String s) {
        Transisi1 tr1 = new Transisi1();
        Transisi2 tr2 = new Transisi2();
        Transisi3 tr3 = new Transisi3();
        String[][] st = tr1.pecahKata(s);
        String[][] st2 = tr2.pecahKata(st);
        String[][] st3 = tr3.pecahKata(st2);
        ambilSuku=st3;
        int jmlh=0;
        for(int i=0;i<st.length;i++){
            if(st3[i][0]!=null){
                jmlh++;
            }
        }
        return jmlh;
    }

    public void getSuku(){
        for(int i=0;i<ambilSuku.length;i++){
            if(ambilSuku[i][0]!=null){
                System.out.print(ambilSuku[i][0]);
                if(ambilSuku[i+1][0]!=null){
                    System.out.print("-");
                }
            }
        }
    }
}
```

```
public class Transisi1 {

    public String[][] pecahKata(String s) {
        String[][] suku = null;

        suku = new String[20][2];
        int p = 0;
        for (int i = 0; i < s.length(); i++) {
            if (cekVokal(s.charAt(i)) == true) {
                suku[p][0] = Character.toString(s.charAt(i));
                suku[p][1] = "v";
                p++;
            } else {
                if (s.charAt(i) == 'n') {
                    if (i == s.length() - 1) {
                        suku[p][0] =
Character.toString(s.charAt(i));
                        suku[p][1] = "k";
                        p++;
                    } else if (cekVokal(s.charAt(i + 1)) == true)
{
                        suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                        suku[p][1] = "kv";
```

```
                        p++;
                        i++;
                    } else if (s.charAt(i + 1) == 'g' ||
s.charAt(i + 1) == 'y') {
                        suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                        suku[p][1] = "k";
                        p++;
                        i++;
                    } else {
                        suku[p][0] =
Character.toString(s.charAt(i));
                        suku[p][1] = "k";
                        p++;
                    }
                } else if (s.charAt(i) == 'k') {
                    if (i == s.length() - 1) {
                        suku[p][0] =
Character.toString(s.charAt(i));
                        suku[p][1] = "k";
                        p++;
                    } else if (cekVokal(s.charAt(i + 1)) == true)
{
                        suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                        suku[p][1] = "kv";
                        p++;
                        i++;
                    } else if (s.charAt(i + 1) == 'h') {
                        suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                        suku[p][1] = "k";
                        p++;
                        i++;
                    } else {
                        suku[p][0] =
Character.toString(s.charAt(i));
                        suku[p][1] = "k";
                        p++;
                    }
                } else if (s.charAt(i) == 's') {
                    if (i == s.length() - 1) {
                        suku[p][0] =
Character.toString(s.charAt(i));
                        suku[p][1] = "k";
                        p++;
                    } else if (cekVokal(s.charAt(i + 1)) == true)
{
                        suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                        suku[p][1] = "kv";
                        p++;
                        i++;
                    } else if (s.charAt(i + 1) == 'y') {
```

```
                            suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                            suku[p][1] = "k";
                            p++;
                            i++;
                        } else {
                            suku[p][0] =
Character.toString(s.charAt(i));
                            suku[p][1] = "k";
                            p++;
                        }
                    } else {
                        if (i == s.length() - 1) {
                            suku[p][0] =
Character.toString(s.charAt(i));
                            suku[p][1] = "k";
                            p++;
                        } else if (cekVokal(s.charAt(i + 1)) == true)
{
                            suku[p][0] =
Character.toString(s.charAt(i)) + Character.toString(s.charAt(i +
1));
                            suku[p][1] = "kv";
                            p++;
                            i++;
                        } else {
                            suku[p][0] =
Character.toString(s.charAt(i));
                            suku[p][1] = "k";
                            p++;
                        }
                    }
                }
            }
            return suku;

        }

        private boolean cekVokal(char c) {
            boolean vokal = false;
            char[] v = {'a', 'i', 'u', 'e', 'o'};
            for (int i = 0; i < v.length; i++) {
                if (c == v[i]) {
                    vokal = true;
                }
            }
            return vokal;

        }
}
```

```
public class Transisi2 {

    public String[][] pecahKata(String[][] s) {
        String[][] suku = new String[20][2];

        int p = 0;
        int length = 0;
        for (int i = 0; i < s.length; i++) {
            if (s[i][0] != null) {
                length++;
            }
        }

        for (int i = 0; i < length; i++) {
            if (s[i][1].equalsIgnoreCase("v")) {
                if (i == length - 1) {
                    suku[p][0] = s[i][0];
                    suku[p][1] = s[i][1];
                } else if (s[i + 1][1].equalsIgnoreCase("k")) {
                    suku[p][0] = s[i][0] + s[i + 1][0];
                    suku[p][1] = s[i][1]+s[i+1][1];
                    //System.out.println(suku[p][0]);
                    p++;
                    i++;
                }
                else{
                    suku[p][0]=s[i][0];
                    suku[p][1]=s[i][1];
                    p++;
                }
            }
            else if (s[i][1].equalsIgnoreCase("k")) {
                if (i == length - 1) {
                    suku[p][0] = s[i][0];
                    suku[p][1] = s[i][1];
                }
                else if (s[i + 1][1].equalsIgnoreCase("k")) {
                    int j = i + 1;
                    String tmpHur = s[i][0];
                    String tmpVK = s[i][1];
                    while (j <
length&&s[j][1].equalsIgnoreCase("k") ) {
                        tmpHur = tmpHur + s[j][0];
                        tmpVK = tmpVK + s[j][1];
                        i++;
                        j++;
                    }
                    if (i + 1 < length && s[i +
1][1].equalsIgnoreCase("kv")) {
                        if (i + 2 < length && s[i +
2][1].equalsIgnoreCase("k") ) {
                            tmpHur = tmpHur + s[i + 1][0] +
s[i+2][0];
                            tmpVK = tmpVK + s[i + 1][1] + s[i +
2][1];
                            suku[p][0] = tmpHur;
                            suku[p][1] = tmpVK;
                            i = i + 2;
                            p++;
```

```
                        }
                        else{
                            suku[p][0]=tmpHur+s[i+1][0];
                            suku[p][1]=tmpVK+s[i+1][1];
                            i++;
                            p++;
                        }
                    }
                    else{
                        suku[p][0]=tmpHur;
                        suku[p][1]=tmpVK;
                        p++;
                    }
                }
                else if(i+1<length &&
s[i+1][1].equalsIgnoreCase("kv")){
                    if (i + 2 < length && s[i +
2][1].equalsIgnoreCase("k")) {
                        suku[p][0] = s[i][0]+s[i + 1][0] + s[i +
2][0];
                        suku[p][1] = s[i][1]+s[i + 1][1] + s[i +
2][1];
                        i = i + 2;
                        p++;
                    }
                    else{
                        suku[p][0] = s[i][0]+s[i + 1][0];
                        suku[p][1] = s[i][1]+s[i + 1][1];
                        i++;
                        p++;
                    }
                }

            } else if (s[i][1].equalsIgnoreCase("kv")) {
                if (i + 1 < length && s[i +
1][1].equalsIgnoreCase("k")) {
                    suku[p][0] = s[i][0] + s[i + 1][0];
                    suku[p][1] = s[i][1] + s[i + 1][1];
                    i++;
                    p++;
                }
                else{
                    suku[p][0]=s[i][0];
                    suku[p][1]=s[i][1];
                    p++;
                }
            }

        }
        return suku;

    }
}
```

```
public class Transisi3 {
    public String[][] pecahKata(String[][] s){
        String[][] suku = new String[20][2];
        int p=0;
        int length=0;
        for(int i=0;i<s.length;i++){
            if(s[i][0]!=null){
                length++;
            }
        }
        for(int i=0;i<length;i++){
            //System.out.println("i : "+i);

if(s[i][1].equalsIgnoreCase("vk")||s[i][1].equalsIgnoreCase("kvk")
|| s[i][1].equalsIgnoreCase("kkvk")){
                if(i+1<length && s[i+1][1].equalsIgnoreCase("k")){
                    suku[p][0]=s[i][0]+s[i+1][0];
                    suku[p][1]=s[i][1]+s[i+1][0];
                    //System.out.println(suku[p][0]);
                    i++;
                    //System.out.println("i2 : "+i);
                    p++;
                }else{
                    suku[p][0]=s[i][0];
                    suku[p][1]=s[i][1];
                    p++;
                }
            }

            else if(s[i][1].equalsIgnoreCase("kv") ||
s[i][1].equalsIgnoreCase("kkv") ||
s[i][1].equalsIgnoreCase("kkkv")){
                if(i+1<length && s[i+1][1].equalsIgnoreCase("v")){
                    suku[p][0]=s[i][0]+s[i+1][0];
                    suku[p][1]=s[i][1]+s[i+1][1];
                    i++;
                    p++;
                }else{
                    suku[p][0]=s[i][0];
                    suku[p][1]=s[i][1];
                    p++;
                }

            }
            else if(s[i][1].equalsIgnoreCase("v")){
                suku[p][0]=s[i][0];
                suku[p][1]=s[i][1];
                p++;
            }
            else{
                suku[p][0]=s[i][0];
                suku[p][1]=s[i][1];
                p++;
            }
        }
        return suku;
    }

}
```