PENGUJIAN DETEKSI TINGKAT PLAGIARISME DENGAN MEMPERTIMBANGKAN MAKNA KATA DENGAN MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

TUGAS AKHIR

Oleh:

DENNY GUNAWAN

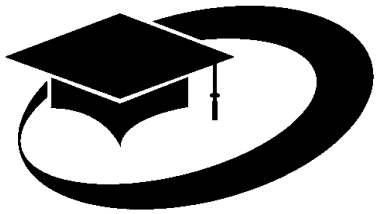
NIM. 12.111.1362

RIVALDI WARMAN

NIM. 12.111.3223

WISELY JANSEN HADI

NIM. 12.111.1044



PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

MIKROSKIL

MEDAN

2016

PLAGIARISM DETECTION TESTING WITH SPECULATING THE MEANING OF THE WORD BY USING LEVENSHTEIN DISTANCE ALGORITHM

FINAL RESEARCH

By:

DENNY GUNAWAN

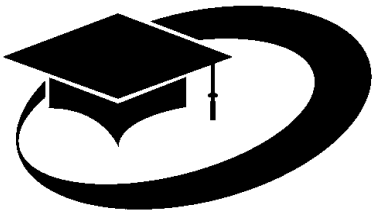
NIM. 12.111.1362

RIVALDI WARMAN

NIM. 12.111.3223

WISELY JANSEN HADI

NIM. 12.111.1044



STUDY PROGRAM OF INFORMATICS ENGINEERING

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER MIKROSKIL

MEDAN

2016

LEMBAR PENGESAHAN

PENGUJIAN DETEKSI TINGKAT PLAGIARISME DENGAN MEMPERTIMBANGKAN MAKNA KATA DENGAN MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

TUGAS AKHIR

Diajukan untuk Melengkapi Persyaratan guna
Mendapatkan Gelar Sarjana Strata Satu
Program Studi Tekni Informatika

Oleh:

DENNY GUNAWAN (12.111.1362)
RIVALDI WARMAN (12.111.3223)
WISELY JANSEN HADI (12.111.1044)

Disetujui Oleh:

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Pahala Sirait, S.T., M.Kom	Sunario Megawan, S.Kom.,M,.Kom.
NIP. 45970117	NIP. 45061039

Medan, 20

Diketahui dan Disahkan oleh:

Ketua Program Studi

Teknik Informatika,

Hardy, S.Kom., M.Sc.

NIP. 45051025

ABSTRAK

Permasalahan terhadap plagiarisme telah ada selama bertahun-tahun. Namum penyebarluasan informasi dan teknologi komunikasi, termasuk salah satunya internet telah memberikan kontribusi untuk kemudahan menjiplak. Oleh karena itu, pendeteksi plagiarisme perlu dilakukan untuk mengurangi tingkat penjiplakan yang semakin marak.

Pada penelitian ini, untuk mendeteksi kata per kata dilakukan dengan menggunakan algoritma *levenshtein distance* dan dalam mempertimbangkan makna kata dilakukan pencocokan dengan databasenya dalam dua buah dokumen yang diuji. Dokumen yang diuji adalah file berformat *.pdf yang berbahasa indonesia. Terdapat 2 tipe pemrosesan dalam proses pengujian, yaitu *preprocessing* dan *nonpreprocessing* yang dapat menampilkan hasil yang berbeda sesuai dengan dokumen yang akan diuji.

Uji coba yang dilakukan dalam penelitian ini akan menampilkan hasil dalam bentuk persentase. Hasil yang diberikan memiliki akurasi yang cukup tinggi dalam mendeteksi plagiat dari segi kemiripan kata maupun makna.

Kata kunci : plagiarisme, *levenshtein distance*, kata, kalimat, *preprocessing*

KATA PENGANTAR

Ucapan syukur kepada Tuhan Yang Maha Esa karena berkat Rahmatnya penulis menyelesaikan tugas akhir yang berjudul, **Pengujian Deteksi Tingkat Plagiarisme dengan Mempertimbangkan Makna Kata dengan Menggunakan Algoritma *Levenshtein Distance,*** sesuai yang direncanakan.

Tugas akhir ini dibuat guna untuk melengkapi persyaratan kurikulum pada Program Studi Teknik Informatika Strata Satu, STMIK Mikroskil Medan. Semoga hasil dari tugas akhir ini ada manfaatnya bagi pihak yang berkepentingan. Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada:

1. Bapak Dr. Pahala Sirait, S.T., M.Kom., selaku pembimbing I yang telah membimbing penulis selama mengerjakan tugas akhir ini.
2. Bapak Sunario Megawan, S.Kom.,M,.Kom., selaku pembimbing II dan Sekretaris Program Studi Teknik Informatika yang telah membimbing penulis selama mengerjakan tugas akhir ini.
3. Bapak Dr. Mimpin Ginting, M.S., selaku Ketua STMIK Mikroskil Medan.
4. Bapak Djoni, S.Kom., M.T.I., selaku Wakil Ketua STMIK Mikroskil Medan.
5. Bapak Hardy, S.Kom., M.Sc., selaku Ketua Program Studi Teknik Informatika.
6. Bapak/Ibu Dosen yang telah membimbing penulis selama pengerjaan tugas akhir ini.
7. Orang tua penulis yang telah memberikan dukungan selama pengerjaan tugas akhir ini.
8. Semua orang yang telah membantu hingga selesainya penulisan tugas akhir ini.

Dalam penulisan tugas akhir ini, penulis menyadari sepenuhnya bahwa dalam penulisan tugas akhir ini masih terdapat banyak kekurangan dan masih jauh dari sempurna. Oleh karena itu setiap saran yang bersifat menbangun akan diterima dengan senang hati. Akhir kata penulis sangat mengharapkan tugas akhir ini dapat memberikan manfaat bagi para pembaca sekalian. Terima kasih

Medan, Agustus 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Permasalahan terhadap plagiarisme telah ada selama berabad-abad. Namun penyebarluasan informasi dan teknologi komunikasi, termasuk internet telah memberikan kontribusi untuk kemudahan menjiplak. Banyak layanan online yang ada untuk memfasilitasi pendeteksian plagiarisme, termasuk database esai dan alat teks *"synonymizer"* seperti synonym.com yang hasil keluaran dari masukan teks berupa daftar-daftar sinomin dari tiap kata. Hasil survei dari 80.000 lebih mahasiswa di amerika serikat dan kanada menunjukkan bahwa sebanyak 62% sarjana dan 59% pascasarjana melakukan plagiarisme *cut and paste* dari sumber tertulis dan internet (McCabe, 2005). Di Indonesia sendiri banyak terjadi kasus plagiarisme salah satunya yaitu dosen di fakultas ekonomi dan bisnis (FEB) Universitas Gadjah Mada, Anggito Abimanyu terhadap artikelnya "Gagasan Asuransi Becana" yang terbit di harian kompas. 10 Februari 2014. Tulisan ini memiliki kesamaan dengan artikel Hotbonar Sinaga dan Munawar Kasan yang Berjudul "Menggagas Asuransi Becana". (Lestarini, Ade Hapsari, 25 Februari 2014)

Penelitian pada sistem untuk menganalisa dan mendeteksi plagiarisme telah dilakukan hampir dua dekade. Sistem pendeteksi plagiarisme saat ini menggunakan pertimbangan teks berbasis karakter yang canggih dan efisien. Sistem ini dapat mendeteksi kata per kata dan penyalinan teks dari sumber yang original. Akan tetapi, penataan ulang kalimat dan penggunaan makna tersirat yang kebanyakan ditemukan dalam penelitian menyebabkan perbandingan kesamaan karakter yang tidak cocok. Oleh karena itu, sistem pendeteksi plagiarisme tidak dapat mendeteksi bahwa dokumen tersebut plagiat atau tidak.

Untuk mengatasi masalah pada sistem pendeteksi kemiripan atau plagiat pada dokumen saat ini, maka digunakan algoritma *levenshtein Distance*. Algoritma *Levenshtein Distance* memiliki 3 proses utama dalam menentukan perbedaan antara

2 *string* (kata) yaitu proses penyisipan *(insertion)*, proses penghapusan *(deletion)* dan proses penggantian (*substitution)* pada huruf yang dianggap sesuai dengan database. Kemudian dari hasil akhir dari proses tersebut maka akan di peroleh nilai kemiripan antar *string* (kata). (adriyani, et al, 2010)

Atas pertimbangan dari paragraf sebelumnya maka diusulkan tugas akhir dengan judul "Pengujian Deteksi Tingkat Plagiarisme dengan Mempertimbangkan Struktur Kalimat dan Makna dengan Menggunakan Algoritma *Levenshtein Distance*".

1.2 Rumusan Masalah

Berdasarkan penjelasan di atas, maka masalah yang diperoleh adalah Kesamaan atau tidak pada makna kalimat yang tersirat antara dokumen yang orisinal dengan dokumen yang plagiat.

1.3 Tujuan dan Manfaat

Tujuan dari penulisan tugas akhir ini adalah sistem yang digunakan dapat melakukan pengujian serta mendeteksi plagiat pada sebuah dokumen dari makna kalimat.

Manfaat dari penulisan tugas akhir ini adalah :

1. Sebuah sistem yang dapat membantu masyarakat khususnya dibidang akademis seperti kampus-kampus agar dapat mengetahui mahasiswa-mahasiswa yang secara sengaja melakukan plagiat terhadap tugas akhir (Skripsi, Tesis, disertasi) dan tugas-tugas kuliah lainnya.
2. Mengurangi tingkat plagiarisme yang terjadi pada masa sekarang dan masa akan datang.

1.4 Batasan Masalah

Adapun batasan masalah dalam tugas akhir ini adalah :

1. Sistem akan melakukan pengecekan terhadap dokumen per kalimat.
2. Sistem memproses dokumen yang berbahasa Indonesia.
3. Sistem mendeteksi dokumen secara *offline*.

4. Dokumen yang akan diproses merupakan dokumen yang berisi teks dan bukan merupakan gambar atau hasil dari *scanning*
5. Dokumen yang akan diproses ber*form*at *.pdf
6. Sistem hanya dapat menguji 2 dokumen

1.5 Metodologi Pengembangan Perangkat Lunak

Langkah-langkah dalam pengerjaan tugas akhir ini, sebagai berikut:

1. Mengumpulkan dan mempelajari materi yang berhubungan dengan topik yang dibahas yaitu mengenai plagiarisme, *paraphrase* dan algoritma *levenshtein distance*.
2. Pengembangan perangkat lunak dengan menggunakan model *waterfall* yang memiliki langkah kerja sebagai berikut:
   a. Memodelkan sistem yang akan dirancang dengan menggunakan alat bantu berupa UML (*Unified Modeling Language*).
   b. Merancang tampilan antarmuka pemakai (*user interface*) perangkat lunak.
   c. Membangun perangkat lunak dari permodelan sistem pada langkah kerja sebelumnya.
   d. Menguji perangkat lunak dan memperbaiki kesalahan yang muncul.
3. Menarik kesimpulan dari pengujian yang dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1 Kata

Kata adalah morfem atau kombinasi morfem yang oleh bahasawan dianggap sebagai satuan terkecil yang dapat diujarkan sebagai bentuk yang bebas. Kata juga merupakan satuan bahasa yang dapat berdiri sendiri, terdiri dari morfem tunggal (misal batu, rumah, datang) atau gabungan morfem (misal pejuang, pancasila, mahakuasa). (KBBI, 2008:633)

Menurut Kamus Besar Bahasa Indonesia (KBBI, 1997) menyebutkan bahwa "kata" adalah (1) elemen terkecil dalam sebuah bahasa yang diucapkan atau dituliskan dan merupakan realisasi kesatuan dan pikiran yang dapat digunakan dalam berbahasa, (2) konversasi atau bahasa, (3) morfem atau kombinasi beberapa morfem yang dapat diujarkan sebagai bentuk yang bebas, (4) unit bahasa yang dapat berdiri sendiri dan terdiri dari satu morfem (mis. kata) atau beberapa morfem gabungan (mis. perkataan).

Menurut Abdul Chaer (2009: 37), secara gramatikal kata mempunyai dua status yaitu sebagai satuan terbesar dalam tataran morfologi, dan sebagai satuan terkecil dalam tataran sintaksis. Sebagai satuan terbesar dalam tataran morfologi, kata dibentuk dari bentuk dasar (yang dapat berupa morfem dasar terikat maupun bebas, atau gabungan morfem) melalui proses morfologi afiksasi, reduplikasi, atau komposisi. Sebagai satuan terkecil dalam sintaksis kata, khususnya yang termasuk kelas terbuka (nomina, verba, dan ajektifa) dapat mengisi fungsi-fungsi sintaksis.

2.1.1 Klasifikasi Kata

Klasifikasi Kata dapat digolongkan menjadi 2 yaitu: (Chaer Abdul, 2007)

1. Klasifikasi Kata Berdasarkan Kelas Kata (Bentuk Kata)

   Dari segi bentuk kata, pengklasifikasiannya terbagi menjadi empat bagian yaitu:

a. Kata Dasar adalah kata asli yang belum diberi imbuhan atau yang belum diberikan awalan, akhiran, sisipan dan penggabungan awalan dan akhiran. Kata-kata seperti baik, getar, kerja, sakit, gunung disebut sebagai kata dasar karena kata-kata itu tidak berimbuhan atau belum diberi imbuhan.
b. Kata Turunan adalah kata yang telah mengalami penambahan atau pengimbuhan. Penambahan atau pengimbuhan disebut juga dengan afiks yang terdiri dari beberapa bentuk, yaitu:
    a) Imbuhan di awal kata (Prefiks atau awalan)
    b) Imbuhan di tengah kata (Infiks atau sisipan)
    c) Imbuhan di akhir kata (Sufiks atau akhiran)

Tabel 2.1 Penjelasan jenis afiks (imbuhan) dan contoh (Chaer Abdul, 2007)

Afiks	Keterangan	Contoh
Prefiks	Afiks yang diimbuhkan di depan bentuk kata dasar	*Ber-, di-, ke-, me-, meng-, mem-, meny-, pe-, pem-, peng- , peny-, per-, se-, ter*
Infiks	Afiks yang diimbuhkan di tengah bentuk kata dasar	*-el-, -er-, -em, -in-*
Sufiks	Afiks yang diimbuhkan di belakang bentuk kata dasar	*-an, -kan, -i, -nya, -pun, -lah, -kah, -wan. –wati, -asi, -isme, -in, -wi, -if, -ik, -is, -er, -al, -in, -us, -or*
Konfiks	Afiks yang berupa morfem yang terbagi yang berada didepan dan belakang bentuk kata dasar	*Ke-an, ber-an, pe-an, per-an, peng-an, peny-an, pem-an, se-an, per-i, di-kan, di-i, me-kan, me-i, ter-kan, ter-I,*

c. Kata Ulang adalah kata yang mengalami proses pengulangan bentuk, baik seluruh kata maupun sebagian. Semua kata ulang wajib ditulis dengan memakai tanda penghubung (-)

d. Kata Majemuk adalah gabungan beberapa kata dasar yang berbeda yang membentuk suatu arti baru. Klasifikasi Kata berdasarkan Kelas Kata (Jenis kata).

2. Dalam tata bahasa baku bahasa Indonesia, kelas kata terbagi menjadi tujuh bagian yaitu:
   a. Kata Benda (Nomina) adalah kata yang mengacu pada benda, orang, konsep ataupun pengertian yang berfungsi sebagai objek dan subjek.
   b. Kata Kerja (Verba) adalah kata yang menyatakan perbuatan atau tindakan. Kata kerja biasanya berfungsi sebagai predikat.
   c. Kata sifat (Adjektiva) adalah kata yang dipakai untuk mengungkapkan sifat atau keadaan sesuatu. Contoh: keadaan orang, binatang dan benda. Kata difat juga berfungsi sebagai predikat.
   d. Kata Keterangan (Adverbia) adalah kata yang memberi keterangan pada verba, adjektiva, nomina predikatif atau kalimat.
   e. Kata ganti (Pronomina) adalah kata yang digunakan untuk mengacu pada nomina lain. Pronomina berfungsi untuk mangganti kata benda atau nomina.
   f. Kata bilangan (Numeralia) adalah kata yang digunakan untuk menghitung banyaknya orang, binatang atau benda.
   g. Kata tugas adalah sejenis kategori kata dalam tatabahasa *form*al bahasa Indoneisa yang berdasarkan peranannya dapat dibagi menjadi lima subkelompok yaitu:
      i. Kata depan (Preposisi) adalah kata yang merangkaikan kata-kata atau bagian kalimat. Preposisi dapat berbentuk kata, contohnya kata di dan untuk, atau gabungan kata, cohtohnya bersama atau sampai dengan.
      ii. Kata sambung (Konjungsi) adalah kata yang menghubungkan dua satuan bahasa; kata dengan kata, frasa dengan frasa, klausa dengan klausa dan seterusnya.
      iii. Kata sandang (Artikula) adalah kata tugas yang membatasi makna nomina.

iv. Kata seru (interjeksi) adalah kata yang mengungkapkan perasaan dan maksud seseorang, misalnya ah atau aduh, atau melambangkan tiruan bunyi, seperti meong.

v. Partikel adalah kategori atau unsur yang bertugas memulai, mempertahankan atau mengukuhkan sebuah kalimat dalam komunikasi. Unsur ini digunakan dalam kalimat tanya, perintah dan pernyataan (berita).

2.2 Kalimat

Secara hierarkis, kalimat merupakan satuan klausa di bawah tataran wacana. Kalimat merupakan konstituen wacana atau pembentuk wacana. Dan disepakati bahwa kalimat sebagai satuan sintaksis terbesar setelah frasa dan klausa. (Dola, 2010:14, 76)

Satuan bahasa yang menjadi inti pembicaraan dalam sintaksis adalah kalimat. Kalimat merupakan satuan di atas klausa dan di bawah satuan wacana. Kalimat adalah satuan sintaksis yang disusun dari konstituen dasar, yang biasany berupa klausa, dilengkapi dengan konjungsi bila diperlukan, serta disertai dengan intonasi final (Chaer Abdul, 2009: 44).

2.2.1 Jenis-jenis Kalimat

Menurut Alwi, dkk. (2003), jenis kalimat dapat ditinjau dari sudut :

1. Berdasarkan jumlah klausa yang dimilikinya, kalimat dibedakan menjadi dua yaitu:
   a. Kalimat Tunggal adalah kalimat yang proposisinya satu sehingga predikatnya pun satu.
   b. Kalimat Majemuk adalah kalimat yang terdiri atas lebih dari satu proposisi sehingga mempunyai paling tidak dua predikat yang tidak dapat dijadikan satu kesatuan. Karena sifat itu maka kalimat majemuk selalu berwujud dua klausa atau lebih. Kalimat majemuk dibagi menjadi 2 yaitu :
      i. Kalimat majemuk setara apabila kalimat itu menyatakan hubungan koordinatif (sejajar/setara).

ii. Kalimat majemuk bertingkat jika hubungan subordinat (bertingkat).

2. Berdasarkan bentuk sintaksis kalimat di bagi menjadi:
   a. Kalimat deklaratif atau kalimat berita

      Kalimat berita merupakan kalimat yang digunakan untuk menginformasikan sesuatu. Biasanya diakhiri dengan tanda titik (.).
   b. Kalimat imperatif atau kalimat perintah

      Kalimat yang bertujuan untuk mengintruksikan seseorang untuk melakukan sesuatu. Kalimat perintah biasanya diakhiri dengan tanda seru. Tapi, jika diakatan langsung atau lisan biasanya ditandai dengan intonasi tinggi
   c. Kalimat interogratif atau kalimat tanya.

      Kalimat tanya mengharapkan jawaban sebagi respon atau reaksi pemberitahuan informasi yang diharapkan, biasanya diakhiri dengan tanda tanya (?). kata tanya yang digunakan bagaimana, mengapa, apa kapan, dimana dsb.
   d. Kalimat ajakan

      Kalimat ajakan merupakan kalimat yang memancing minat lawan bicara. Kata yang sering digunakan adalah Ayo, Mari dsb. Biasanya ada pada iklan.
   e. Kalimat pengandaian

      Kalimat pengandaian menggambarkan keinginan atau tujuan dari penulis atau pembicara yang belum atau tidak kesampaian. Contoh : Andai saja aku bisa jadi dokter bedah.
3. Berdasarkan kelengkapan unsur dibagi menjadi:
   a. Kalimat lengkap atau kalimat major

      Kalimat mayor hanya memiliki subjek dan predikat. Objek, pelengkap dan keterangan boleh ditambahkan sesuka hati. Sama seperti pola dasar pertama.

b. Kalimat tak lengkap atau minor

Kalimat yang memiliki satu inti fungsi gramatikalnya. Bentuk kalimat minor seperti kalimat tambahan, kalimat jawaban, kalimat salam, panggilan maupun judul.

4. Berdasarkan subjek predikat
   a. Kalimat biasa

      Kalimat yang subjeknya mendahului predikatnya. Kalimat berpola dasar.

   b. Kalimat inversi

      Kalimat ini merupakan kebalikan dari kalimat normal. Dimana predikatnya mendahului objek.

5. Berdasarkan pengucapan
   a. Kalimat Langsung

      Kalimat yang secara detai meniru sesuatu yang diujarkan oranglain. Tanda baca kutip tidak luput dalam jenis kalimat langsung. Kutipan dalam kalimat langsung berupa kalimat tanya, kalimat berita ataupun kalimat perintah.

   b. Kalimat tak langsung

      Kalimat yang melaporkan kembali kalimat yang diujarkan orang lain. Kutipan dalam kalimatnya senmuany berbentuk berita.

2.3 Paragraf

Menurut Akhadiah (Nasucha, Rohmadi dan Wahyudi. 2009: 33) Paragraf merupakan inti penuangan buah pikiran dalam sebuah pikiran. Dalam paragraf terkandung satu unit buah pikiran yang didukung oleh semua kalimat dalam paragraf tersebut, mulai dari kalimat pengenal, kalimat utama atau topik, kalimat-kalimat penjelas sampai pada kalimat penutup. Himpunan kalimat ini saling bertalian dalam suatu rangkaian untuk membentuk sebuah gagasan

2.3.1 Jenis-jenis Paragraf

Jenis paragraf dibagi menjadi 3 yaitu:

1. Paragraf pembuka

   Rohmadi dan Nasucha (2010:39) mengemukakan Paragraf pembuka dapat disebut paragraf pendahuluan *(introduction).* Fungsinya sebagai pengantar untuk sampai kepada pokok pembicaraan dalam karangan.

2. Paragraf penghubung

   Paragraf penghubung berisi inti persoalan yang dikemukakan. Oleh karena itu, secara kuantitatif paragraf inilah yang paling panjang, danantara paragraf dengan paragraf saling berhubungan secara logis (Nasucha, Rohmadi dan Wahyudi. 2009: 34).

3. Paragraf penutup

   Paragraf ini berisi kesimpulan dari paragraf penghubung. Paragraf penutup juga dapat berisi penegasan kembali mengenai hal-hal yang dianggap penting dalam paragraf penghubung (Nasucha dkk, 2009:35).

2.3.2 Syarat Pembentukan Paragraf

Dalam pembentukan paragraf, terdapat syarat-syaratnya yang antara lain :

1. Kesatuan

   Paragraf dianggap mempunyai kesatuan, jika kalimat– kalimat dalam paragraf itu tidak terlepas dari topiknya atau selalu relevan dengan topik (Nasucha, Rohmadi dan Wahyudi. 2009: 35). Semua kalimat berfokus pada topik dan mencegah masuknya hal-hal yang tidak relevan.

   Menurut Marsa (2009: 9) kesatuan dalam sebuah paragraf hanyaterbentuk apabila informasi-informasi dalam paragraf itu tetap dikendalikan oleh gagasan utama. Agar kesatuan dapat dicapai penulis senantiasa mengevaluasi kalimat-kalimat yang ditulisnya itu erat hubungannya dengan gagasan utama. Apabila tidak erat hubungannya, kalimat-kalimat itu sebaiknya dihilangkan atau disajikan secara khusus, misalnya menjadi sisipan dalam kalimat lain.

2. Kepaduan

Satu paragraf bukanlah merupakan kumpulanatau tumpukan kalimat yang masing–masing berdiri sendiri atau terlepas, tetapi dibangun oleh kalimat-kalimat yang mempunyai hubungan timbal balik. Pembaca dengan mudah memahami dan mengikuti jalan pikiran penulis tanpa hambatan karena adanya loncatan pikiran yang membingungkan. Urutan pikiran yang teratur, akan memperlihatkan adanya kepaduan. Jadi, Kepaduan atau koherensi dititikberatkan pada hubungan antara kalimat dengan kalimat (Nasucha, Rohmadi dan Wahyudi. 2009: 37)

Kepaduan antar kalimat dalam paragraf itu meliputi dua macam, yaitu kepaduan bentuk dan kepaduan makna. Kepaduan makna adalah kepaduan informasi yang disebut koherensi dan kepaduan dibidang bentuk disebut kohesi.Paragraf yang memiliki kepaduan informasi bersifat kohesi dan kesesuaian di bidang bentuk disebut kohesif. Wacana yang baik dalam sebuah paragraf apabila memiliki dua kepaduan tersebut, yaitu kohesif dan koheren (Rohmadi dan Nasucha, 2010: 46).

3. Kelengkapan

Suatu paragraf dikatakan lengkap, jika berisi kalimat–kalimat penjelas yang cukup untuk menunjang kejelasan kalimat topik dan kalimat utama (Nasucha, Rohmadi dan Wahyudi. 2009: 39). Sebaliknya suatu paragraf dikatakan tidak lengkap, jika tidak dikembangkan atau hanya diperluas dengan pengulangan–pengulangan.

Paragraf dapat dikatakan memiliki kelengkapan, jika kalimat topiknya dapat dikembangkan dengan pendukung yang cukup (Rohmadi dan Nasucha, 2009: 47-48). Istilah cukup adalah relatif, tetapi yang jelas lebih dari satu dan kurang dari sepuluh. Jika didukung oleh satu kalimat maka pengembangannya kering dan jika sangat banyak maka pembaca cepat bosan dan sulit menemukan keutuhan informasi.

2.4 Makna

Menurut teori yang dikembangkan Ferdinand de Saussure, makna adalah pengertian atau konsep yang dimiliki atau terdapat pada sebuah tanda linguistik. Jika tanda linguistik tersebut disamakan identitasnya dengan kata atau leksem, berarti makna adalah pengertian atau konsep yang dimiliki oleh setiap kata atau leksem. Jika disamakan dengan morfem, maka makna adalah pengertian atau konsep yang dimiliki oleh setiap morfem, baik morfem dasar maupun morfem afiks. (Chaer Abdul,2007)

2.4.1 Jenis Makna

Ada banyak jenis makna antara lain (Chaer Abdul, 2007) :

1. Makna Leksikal, Gramatikal dan Konteksual
    a. Makna leksikal adalah makna yang dimiliki atau ada pada leksem meski tanpa konteks apapun.
    b. Makna gramatikal adalah makna yang ada jika terjadi proses gramatikal seperti afiksasi, reduplikasi, komposisi atau kalimatisasi.
    c. Makna kontekstual adalah makna sebuah leksem atau kata yang berada di dalam satu konteks.
2. Makna Referensial dan Non - Referensial

    Sebuah kata atau leksem dikatakan bermakna referensial jika ada referensnya atau acuannya. Ada sejumlah kata yang disebut kata diektik, yang acuannya tidak menetap pada satu wujud.
3. Makna Denotatif dan Makna Konotatif
    a. Makna denotatif adalah makna asli, makna asal atau makna sebenarnya yang dimiliki oleh sebuah leksem. Makna denotatif sebenarnya sama dengan makna leksikal.
    b. Makna konotatif adalah makna lain yang ditambahkan pada makna denotatif yang berhubungan dengan nilai rasa dari orang yang menggunakan kata tersebut.

4. Makna Konseptual dan Makna Asosiatif
    a. Makna konseptual sebenarnya sama dengan makna leksikal, deotatif dan makna referensial.
    b. Makna asosiatif adalah makna yang dimiliki sebuah leksem atau kata bahasa.
5. Makna Kata dan Makna Istilah
    a. Makna yang dimiliki oleh sebuah kata adalah makna leksikal, denotatif atau makna konseptual. Namun, dalam penggunaannya makna kata itu baru menjadi jelas jika kata itu sudah berada di dalam konteks kalimatnya atau konteks situasinya.
    b. Istilah mempunyai makna yang pasti, jelas, tidak meragukan, meskipun tanpa konteks kalimat. Oleh karena itu, istilah sering dikatakan bebas konteks, sedangkan kata tidak bebas konteks.
6. Makna Idiom dan Peribahasa
    a. Idiom adalah satuan ujaran yang maknanya tidak dapat diramalkan dari makna unsur-unsurnya, baik secara leksikal maupun secara gramatikal.
    b. Peribahasa memilliki makna yang masih dapat ditelusuri dari makna unsurnya karena adanya "asosiasi" antara makna asli dengan maknanya sebagai peribahasa.

2.4.2 Relasi Makna

Relasi makna adalah hubungan semantik yang terdapat antara satuan bahasa yang satu dengan yang lain. (Chaer Abdul, 2007). Ada beberapa tipe relasi makna di antaranya yaitu:

1. Sinonim yaitu hubungan semantik yang menyatakan adanya kesamaan makna antara satu satuan ujaran dengan satuan ujaran lainnya. Dua buah ujaran yang bersinonim maknanya tidak akan sama persis. Ketidaksamaan itu terjadi karena faktor : waktu, tempat, ke*form*alan, social, bidang kegiatan, factor nuansa makna

2. Antonim yaitu hubungan semantik antara dua buah satuan ujaran yang maknanya menyatakan kebalikan, pertentangan, atau kontras antara yang satu dengan yang lain.
3. Polisemi yaitu kata yang mempunyai makna lebih dari satu. Dalam kasus polisemi, biasanya makna pertama adalah makna sebenarnya, yang lain adalah makna yang dikembangkan berdasarkan salah satu komponen makna yang dimiliki kata atau satuan ujaran itu
4. Homonim yaitu dua buah kata atau satuan ujaran yang bentuknya "kebetulan" sama dan maknanya berbeda, karena masing-masing merupakan kata atau bentuk ujaran yang berlainan.
5. Hiponimi yaitu hubungan semantik antara sebuah bentuk ujaran yang maknanya tercakup dalam makna bentuk ujaran yang lain. Relasi hiponimi bersifat searah.
6. Ambigiutas atau ketaksaan yaitu gejala dapat terjadinya kegandaan makna akibat tafsiran gramatikal yang berbeda. Ketaksaan terjadi dalam bahasa tulis akibat perbedaan gramatikal karena ketiadaan unsur lisan, karena ketidakcermatan dalam menyusun konstruksi beranaforis.
7. Redudansi yaitu kata yang berlebih-lebihan yang menggunakan unsur segmental dalam suatu bentuk ujaran.

2.5 Plagiat

Plagiat atau plagiarisme adalah salah satu bentuk tindakan kecurangan, tetapi hal tersebut sulit untuk dijelaskan sehingga semua orang bahkan anak-anak bisa melakukannya tanpa menyadari dan memahami bahwa hal tersebut sebenarnya tidak dibenarkan. (Steven, Dowshen,2011)

Plagiat dapat dianggap sebagai tindakan melanggar hukum karena dikategorikan sebagai tindakan mencuri hak cipta seseorang. Di dunia pendidikan, para pelaku plagiat akan mendapatkan hukuman yang cukup berat bila pelaku plagiat adalah seorang mahasiswa atau seorang pendidik (Dosen, Guru) maka pelaku akan dikeluarkan dari universitas/sekolah tempat pelaku memperoleh ilmu atau yang berprofesi sebagai tenaga pendidik. (Steven, Dowshen,2011)

2.5.1 Jenis-jenis Teknik Plagiarisme

Berdasarkan pola, modus dan teknik plagiat dapat dikategorikan menjadi 4 jenis plagiarisme (Andreas Lako,2012) yaitu:

a) Plagiarisme total adalah tindakan plagiat yang dilakukan oleh seorang penulis dengan cara melakukan penjiplakan hasil karya orang lain secara keseluruhan dan kemudian mengklaim bahwa hasil plagiat tersebut sebagai karyanya sendiri.
b) Plagiarisme parsial adalah tindakan plagiat yang dilakukan oleh seorang penulis dengan cara melakukan penjiplakan hanya sebagian dari hasil karya orang lain, biasanya dalam melakukan plagiat seorang penulis hanya mengambil beberapa pernyataan, landasan teori, pembahasan sampai kesimpulan dan tanpa menyebutkan sumber aslinya.
c) Auto-Plagiasi (Self-plagiarisme) adalah tindakan plagiat yang dilakukan oleh seorang penulis terhadap hasil karyanya sendiri, baik hanya sebagian besar dari hasil karyanya atau secara keseluruhan.
d) Plagiarisme antarbahasa adalah tindakan plagiat yang dilakukan oleh seorang penulis dengan cara menerjemahkan suatu karya tulis yang berasal dari negara lain atau karya tulis yang berbahasa asing kemudian diterjemahkan ke dalam bahasa Indonesia.

Selain keempat jenis tersebut plagiarisme juga dapat dikelompokkan atau diklasifikasi lebih detail berdasarkan proporsi atau presentase kata, kalimat atau paragraph yang dibajak (Sastroasmoro Sudigdo , 2007) antara lain :

1. Plagiarisme ringan dengan berkisar antara : < 30%
2. Plagiarisme sedang dengan berkisar antara : 30-70%
3. Plagiarisme berat atau total berkisar antara : > 70%

2.5.2 Taksonomi Plagiarisme

Taksonomi / pengelompokkan plagiarism dibagi menjadi 2 bagian besar yaitu: plagiarisme secara literal (*literal Plagiarism)* dan plagiarisme secara cerdas *(intelligent plagiarism*). (Alzahrani,Salha M. et al 2012)

1. *Literal Plagiarism*

Literal plagiarism/ plagiarisme secara literal adalah suatu tindakan yang paling umum dan paling sering dilakukan dimana seorang plagiat tidak perlu menghabiskan waktu untuk menyembunyikan kejahatan akademik yang telah dilakukan (Alzahrani,Salha M. et al ,2012)

Beberapa teknik *Literal Plagiarism yaitu:*

a) *Extract copy* ; teknik ini melakukan pengambilan / mengcopy secara keseluruhan dari suatu dokumen atau beberapa bagian penting dari dokumen original.
b) *Near copy* ; teknik ini melakukan penambahan, penghapusan, subtitusi, pengabungan atau pemisahan suatu kalimat pada dokumen original.
c) *Modified copy* / *Restructing*; teknik ini melakukan modifikasi dari segi struktur kalimat dari dokumen original dengan penataan kembali frase pada kalimat tersebut.

2. *Intelligent Plagiarism*

Intelligent plagiarism / plagiarisme secara pintar adalah suatu tindakan ketidakjujuran yang merupakan perhatian serius akademik dimana seorang plagiat mencoba untuk menipu pembaca dengan mengganti kontribusi orang lain dan menjadi hasil mereka. Seorang plagiat akan mencoba untuk menyembunyikan, mengaburkan, dan mengubah karya asli dalam berbagai cara yang cerdas, termasuk manipulasi teks, terjemahan dan adopsi ide (Alzhrani, Sallha M. , et al , 2012)

Beberapa jenis teknik dari intelligent plagiarism yaitu:

a) Text Manipulation

Plagiat dapat mengaburkan makna dengan memanipulasi teks dan mengubah semua dari penampilan teks dari dokumen.

Text Manipulation juga dibagi 2 yaitu

i. Paraphrasing

Paraphrasing / *Paraphrase* adalah strategi pemahaman makna suatu bentuk karya sastra dengan cara mengungkapkan kembali karya pengarang

tertentu dengan menggunakan kata-kata yang berbeda dengan kata-kata yang digunakan pengarang. Tujuannya adalah menyederhanakan pemakaian kata atau kalimat (Aminuddin,2010:41).

Teknik *paraphrase* meliputi pengalihan bentuk dari hasil karya orang lain tapi tetap menyertakan sumber karya ilmiah. (1) Perubahan kata/frasa kunci dengan kata lain yang memiliki makna yang mirip atau sama. Proses ini menyangkut pemilihan kata yang memiliki persamaan arti (sinonim). (2) Perubahan bentuk kalimat asal dengan kalimat dengan susunan atau pola berbeda tanpa mengubah maknanya. (3) Perubahan bentuk wacana menjadi uraian yang lebih pendek berupa ringkasan atau rangkuman.

ii. Summarizing

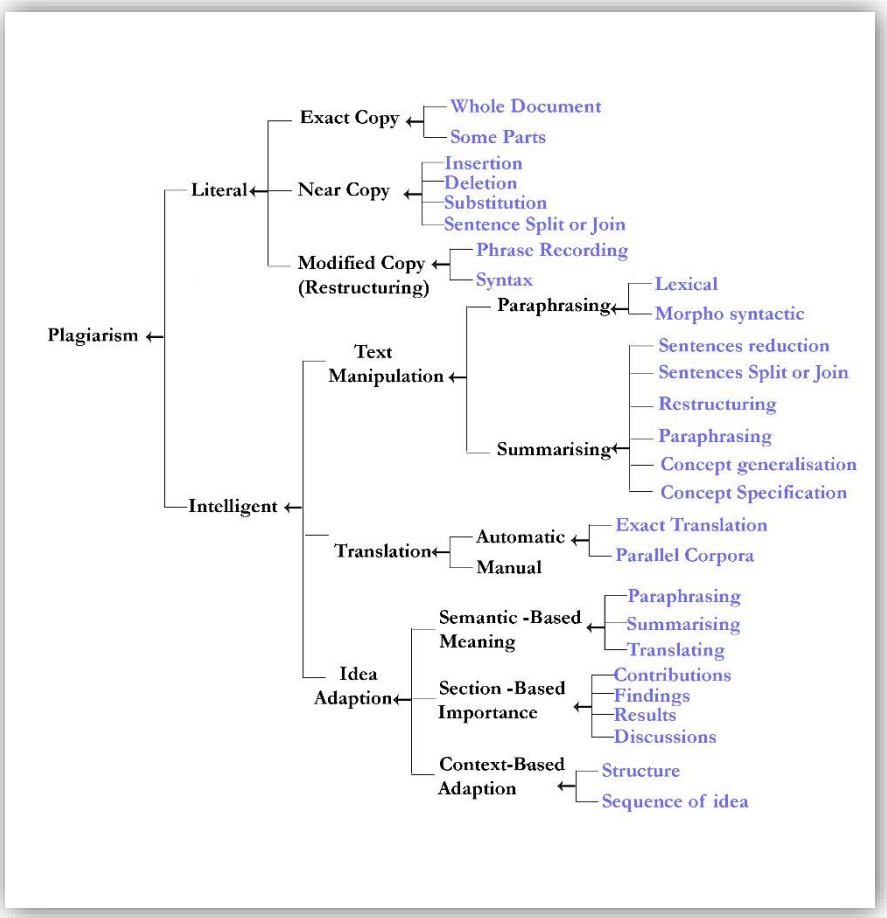
Summary / *summarizing* /ringkasan adalah strategi yang melibatkan ide utama yang akan digunakan untuk plagiat. Ide akan diringkas dari sumber aslinya. Ringkasan secara signifikan lebih pendek dari pada aslinya. (Driscoll, dana lynn & Allen Brizee, 2010)

b) Translation

Translation / menerjemahkan adalah proses mengalihkan satu kebudayaan ke kebudayaan lain atau suatu pengertian dari kebudayaan yang satu ke pengertian kebudayaan yang lain. (mayantara, 23 oktober 2012)

c) *Idea Adoption*

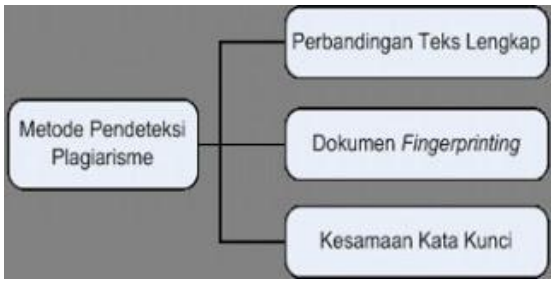
Idea Adoption adalah tindakan plagiarisme yang paling serius mengacu pada penggunaan ide, hasil, konstribusi, temuan, dan kesimpulan tanpa mengutip sumber asli dari ide (Alzhrani,Sallha M., et al, 2012)



Gambar 2.1 Taksnonomi plagiarisme *(Jen Wiss)*

2.5.3 Metode Pendeteksian Plagiarisme

Metode Pendeteksi Plagiarisme dapat dibagi menjadi 3 bagian yaitu metode perbandingan teks lengkap, metode dokumen *fingerprinting* dan metode kesamaa kata kunci. (George R.S Weir,2004)



Gambar 2.2 Metode pendeteksi plagiarisme *(Stein,2006)*

Berikut ini adalah penjelasan dari masing-masing metode dan algoritma pendeteksi plagiarisme (*Stein, B..)*:

1. Perbandingan Teks Lengkap

   Metode ini diterapkan dengan membandingkan semua isi dokumen. Dapat diterapkan untuk dokumen yang besar. Pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif, karena kumpulan dokumen yang diperbandingkan adalah dokumen yang disimpan pada penyimpanan lokal.

   Metode perbandingan teks lengkap tidak dapat diterapkan untuk kumpulan dokumen yang tidak terdapat pada dokumen lokal. Algoritma yang digunakan pada metode ini adalah algoritma *Brute-Force*, algoritma *Edit Distance*, algoritma *Boyer-Moore* dan algoritma *Lavenshtein Distance*.

2. Dokumen *Fingerprinting*

   Dokumen *fingerprinting* merupakan metode yang digunakan untuk mendeteksi keakuratan salinan antar dokumen atau hanya sebagian teks saja. Prinsip kerja dari metode dokumen *fingerprinting* ini adalah dengan menggunakan teknik *hashing*. Teknik *hashing* adalah sebuah fungsi yang mengkonversi setiap *string* 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.

2.6 Preprocessing

Preprocessing adalah proses pembersihan dan persiapan teks untuk tahap selanjutnya. Teks *online* biasanya banyak noise dan bagian informasi tidak berguna seperti HTML *tags*, *scripts* dan iklan. Sebaliknya, pada tingkat kata, banyak kata pada teks tidak berpengaruh pada orientasi umum. Menjaga kata itu akan memyebabkan dimensi masalah tinggi dan kesulitan klasifikasi karena setiap kata dalam teks diperlakukan sebagai satu dimensi. Hipotesis jika dilakukan

preprocessing : mengurangi noise di teks yang membantu meningkatkan per*form*a dan kecepatan pada klasifikasi. (Haddi, emma, ed al, 2013).

2.6.1 Tokenization

Tokenization adalah proses memecahkan setiap susunan kalimat dalam dokumen menjadi beberapa bagian kata yang disebut *token*, dalam waktu yang sama membuang beberapa karakter tertentu yang tidak diperlukan seperti tanda baca dan mengubah semua huruf besar menjadi huruf kecil berikut contoh dari *tokenization* (Manning, Christopher Et al, 2009)

Input : Friends, Romans, Countrymen, Lend me your ears;

Output:

Friends	Romans	countrymen	lend	me	your	Ears

Tokenization berguna baik dalam linguitik (di mana merupakan bentuk dari segmentasi teks), dan dalam dunia komputer *tokenization* merupakan bentuk dari analisis leksikal. Beberapa masalah yang muncul dalam proses *tokenization* adalah seperti penghapusn tanda baca, karakter lain seperti tanda kurung, tanda penghubung dan lain sebagainya yang juga membutuhkan pemrosesan juga. Penggunaan utama *tokenization* adalah mengidentifikasi kata kunci yang memiliki makna. (Gurusamy, Vairaprakash, 2014)

2.6.2 Stemming

Stemming adalah sebuah proses dimana akhiran kata atau imbuhan lain dihilangkan atau diubah agar bentuk kata yang berbeda dalam cara-cara non-relevan dapat mungkin digabung dan diperlakukan setara. Program computer yang melakukan transformasi tersebut disebut *stemmer* atau *stemming algorithm.* Hasil keluaran dari *stemming algorithm* dikenal sebagai *stem.* (liu, ling dan ozsu, tamer, 2009)

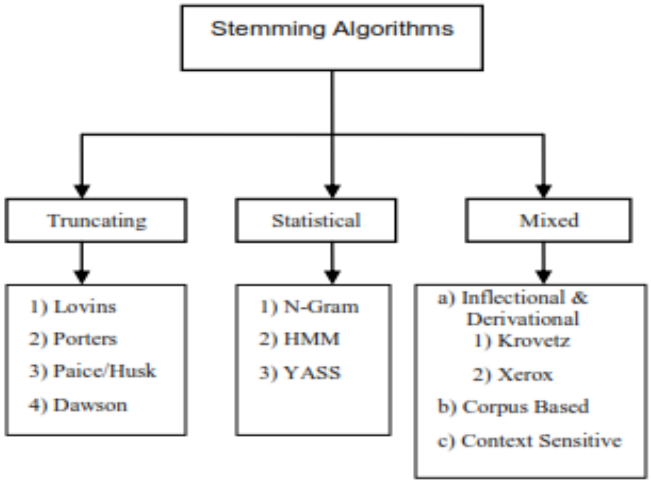
Stemming mengacu pada penghapusan dari setiap imbuhan akhiran (kadang-kadang pada imbuhan lainnya) dari kata yang dimasukkan untuk menghasilkan *stem*. Dua kata akan dianggap setara jika mereka berasal identik (contohnya “memakan” dan “dimakan” berasal dari makan). Metode ini banyak

disukai karena cepat. Dalam hal ini, semua kata diubah menjadi *stem* untuk *input* ke sistem dan selanjutnya dilakukan *string matching* sederhana. (liu, ling dan ozsu, tamer, 2009)

Terdapat 2 jenis kesalahan pada *stemming*, antara lain: (jivani, A. G., 2011)

1. *Understemming*, ketika dua kata yang seharusnya berasal dari kata dasar yang sama tetapi tidak. Hal ini dikenal sebagai *false negative*.
2. *Overstemming*, ketika dua kata yang berbeda *stem* tetapi berasal dari kata dasar yang sama. Hal ini dikenal sebagai *false positive*.

Secara luas, algoritma *stemming* dapat diklasifikasikan menjadi 3 grup yaitu: metode *truncating*, metode *statistikal*, dan metode *mixed*. Setiap dari grup ini mempunyai khas sendiri dalam menemukan stem pada varian kata.



Gambar 2.3 Klasifikasi algoritma *stemming* (jivani, A. G., 2011)

1. Truncating

Seperti namanya yang dijelaskan menunjukan metode ini dengan menghapus akhiran atau awalan (imbuhannya) dari sebuah kata. *Stemming* yang paling dasar yaitu dengan menjaga N huruf dan menghapus sisanya.

Pada metode ini, kata yang lebih pendek dari N akan tetap simpan seperti itu. Peluang *overstemming* akan meningkat jika panjang kata kecil. Contoh algoritmanya yaitu *lovins stemmer, porter stemmer, paice/husk stemmer, dawson stemmer,* dll.

2. *Statistical*

   Stemming ini berdasarkan analisa dan teknik *statistical*. Kebanyakan metode ini menghapus imbuhan awalan tetapi setelah mengimpelmentasikan beberapa prosedur statistikal. Contoh yang menggunakan metode ini adalah N-gram, HMM (*Hidden Markov Model*), YASS (*Yet Another Suffix Striper*), dll.

3. *Mixed*

   Pada metode mixed terbagi 3 pengelompokkan yaitu:

   a. Metode *inflectional* dan *derivational*.
   b. *Corpus based*.
   c. *Context sensitive*

2.6.3 Stopwords

Stopwords adalah kata-kata yang sering muncul dan/atau bertindak sebagai lem untuk membentuk kalimat yang merdu tetapi tidak memiliki kontribusi apapun untuk arti sebuah dokumen (teks). *Stopwords* biasanya kata ganti, kata depan, dll. Menghapus *stopwords* menurunkan *overhead* pengindeksan tanpa mengurangi akurasi pencarian teks (atau sistem serupa – seperti *summarization*, klasifikasi, dll). Pada kebanyakan kasus, *stopwords* merupakan daftar kata-kata biasa yang dalam bentuk inflect yang dihapus pada saat tahap *preprocessing.* (galiotou, eleni , et al, 2013)

2.7 Levenshtein Distance

Levenshtein Distance dibuat oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*. Perhitungan jarak antara dua *string* ini ditentukan dari jumlah minimum operasi perubahan untuk membuat suatu *string* A menjadi *string* B (Adriyani,et al, 2012)

Algoritma ini digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spell checking*), pengenal pembicaraan (*speech*

recognition), pengucapan dialek, analisis DNA, pendeteksi pemalsuan, dan lain-lain. (adiwidya B.M.D, 2009)

Pada dasarnya, algoritma ini menghitung jumlah minimum pentransformasian suatu *string* menjadi *string* lain yang meliputi penggantian, penghapusan, dan penyisipan. Algoritma ini digunakan untuk mengoptimalkan pencarian tersebut karena sangat tidak efisien jika dilakukan pencarian setiap kombinasi operasi-operasi *string* tersebut. Oleh karena itu, algoritma ini tergolong program dinamis dalam pencarian nilai minimal tersebut. (Priayana, et al , 2013)

Ada 3 macam operasi utama yang dapat dilakukan oleh algoritma ini yaitu : (Adriyani NM, et al, 2012)

1. Operasi Pengubahan Karakter. Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain yang dianggap salah atau tidak sesuai.
2. Operasi Penambahan Karakter. Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string*. Penambahan karakter tidak hanya dilakukan di akhir kata, namun bias ditambahkan diawal maupun disisipkan di tengah *string* .
3. Operasi Penghapusan Karakter. Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string* yang dianggap tidak perlu atau tidak sesuai.

Metode *Levenshtein Distance* dapat digunakan untuk melakukan perhitungan beda jarak antara dua *string*. Jarak atau *distance* adalah jumlah minimum dari operasi hapus, *insert* atau substitusi yang dibutuhkan untuk merubah *string* asal (s) menjadi *string target* (t).Sebagai contoh:

1. Jika s adalah “coba” dan t adalah “coba”, maka LD (s,t) = 0, dikarenakan tidak ada transformasi yang dibutuhkan. Kedua *string* adalah identik.
2. Jika s adalah “coba” dan t adalah “coka”, maka LD (s,t) = 1, dikarenakan satu substitusi (merubah “b” ke “k”) dicukupkan untuk mentransformasi s menjadi t.

Perhitungan harga pengeditan pada setiap operasi yang dilakukan adalah sesuai dengan aturan berikut:

a) d (a,ε) = 1 harga untuk menghapus *substring* a

b) d (ε,a) = 1 harga untuk penyisipan *substring* a

c) d (a,b) = 1 harga untuk substitusi *substring* a ke *substring* b

d) d (a,a) = 0

Semakin besar angka yang dihasilkan oleh operasi *Levenshtein Distance* maka semakin besar perbedaan di antara kedua *string* tersebut. (Gilleland,M , people.cs.pitt.edu)

Berikut adalah tahapan-tahapan dari algortima *Levenshtein Distance* (Gilleland M ,et al, people.cs.pitt.edu)

Tabel 2.2 *Tahapan* pseudocode *algoritma* levenshtein distance *(Gilleland M et al, people.cs.pitt.edu*)

Step	**Description**
1	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing 0..m rows and 0..n columns.
2	Initialize the first row to 0..n. Initialize the first column to 0..m.
3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If s[i] equals t[j], the cost is 0. If s[i] doesn't equal t[j], the cost is 1.
6	Set cell d[i,j] of the matrix equal to the minimum of : a. The cell immediately above plus 1: d[i-1,j] + 1. (Deletion) b. The cell immediately to the left plus 1: d[i,j-1] + 1. (Insertion) c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost. (Substitution)
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

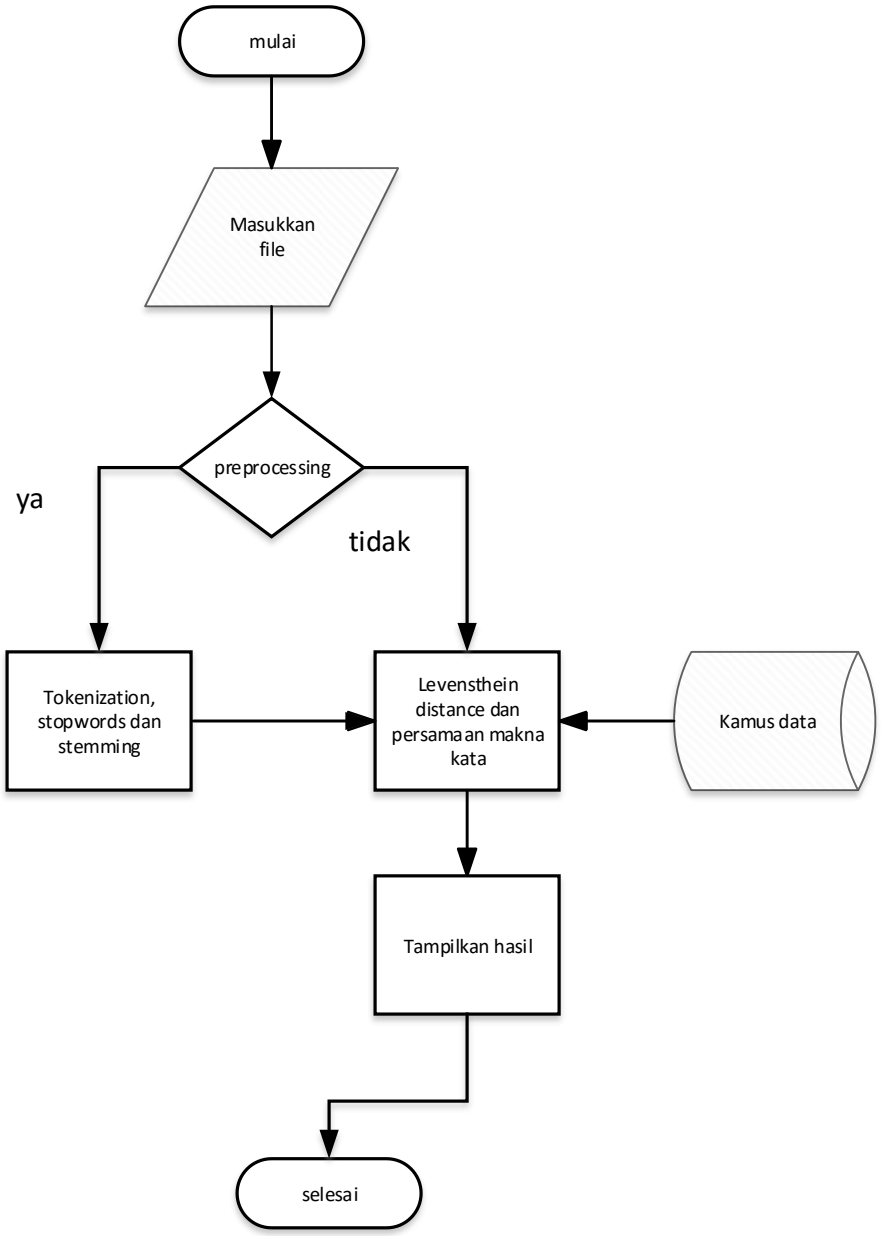
BAB III
METODE PENELITIAN

3.1 Analisis

Pada bagian bab ini akan dijelaskan secara rinci tentang analisis dan perancangan sistem yang akan dikembangkan. Sistem aplikasi yang akan dikembangkan adalah aplikasi yang dapat melakukan pengujian terhadap *file* berupa teks yang dianggap plagiat dan dapat menentukan suatu *file* tersebut plagiat atau tidak.

Tidak hanya mendeteksi plagiat dari segi persamaan kata (sinonim) tapi aplikasi ini juga dapat mendeteksi plagiat dari segi kesamaan kata (*String Matching*). Aplikasi ini juga dapat melakukan 2 tipe pengujian yaitu dengan menggunakan *preprocessing* dan *non preprocessing*. kedua tipe pengujian tersebut akan dapat menghasilkan hasil akhir yang berbeda.

Di bagian hasil akhir terdapat 2 hasil akhir yang dapat menentukan suatu *file* dapat dikatakan plagiat atau tidak. 2 hasil tersebut berupa persentase dari hasil pengujian *file* berupa teks dari segi persamaan kata (sinonim) dan kesamaan kata (*String Matching*). Dalam menentukan hasil akhir digunakan algoritma *levenshtein distance*.



Gambar 3.1 *Flowchart Sistem*

3.1.1 Analisis Proses

3.1.1.1 Leveinsthein distance

Proses levenshtein melibatkan koordinat dari masing masing huruf. Rumus untuk melakukan perhitungan edit levenshtein distance adalah (x, y-1) +1 ke atas, (x-1, y) +1 ke kiri dan (x-1, y-1) + cost dimana cost bernilai 0 jika huruf koordinat X dan Y sama dan bernilai 1 jika huruf koordinat X atau Y tidak sama.

Hasil angka dari rumus tersebut akan selalu diambil yang paling kecil atau minimum. Dibawah ini adalah contoh 2 kalimat dibawah dibawah akan ditunjukkan cara perhitungan levenshtein tersebut. 2 kalimat tersebut akan diproses sedemikian rupa dengan satu kata per kata.

Contoh kalimat 1: Saya benci dia

Contoh kalimat 2: Saya tidak suka dia

Dari contoh kalimat diatas, “Saya benci dia” dan “Saya tidak suka dia”, kata “Saya” di kalimat pertama sebagai acuan kemudian dibandingkan dengan kata “Saya” di kalimat kedua, kemudian dengan kata “Tidak”, kata “Suka” dan kata “Dia”. Setelah semua selesai kata “Saya” di kalimat pertama, maka dilanjutkan acuan berikutnya yaitu kata “Benci”. Kemudian dibandingkan dengan kata “Saya” di kalimat kedua dan seterusnya sampai semua kata telah diproses.

Dalam table tersebut, angka di sudut kanan bawah merupakan hasil angka akhir perbedaan dari dua kata tersebut. Contoh seperti table “Saya” dan “Saya” dibawah ini dimana angka sudut kanan bawah hasilnya nol yang berarti tidak ada perbedaan yang harus diedit karena semua huruf persis sama. Contoh lain adalah table “Saya” dan “Suka” dimana hasil angka akhir adalah dua, yang menjelaskan bahwa ada dua huruf yang tidak sama yaitu huruf “a, y” dan “u, k”. Proses ini akan dijelaskan dengan contoh table “Saya” dan “Suka”.

Berdasarkan table dibawah, baris pertama selalu berisi huruf (horizontal) dan kolom pertama selalu berisi huruf (vertical). Baris kedua dan kolom kedua akan selalu berisi urutan angka dari 0 sampai panjang kolom atau baris. Inisial angka 1 dari kolom dan baris adalah huruf awal dari setiap kata kemudian dilanjutkan sampai akhir. Untuk menentukan angka selanjutnya, rumus tersebut digunakan.

Berikut proses perhitungan levenshtein:

		S	A	Y	A
	0	1	2	3	4
S	1	0			
U	2				
K	3				
A	4				

Gambar 3.2 hasil perhitungan *levenshtein* huruf ‘s’ dengan ‘s’

Angka 0 yang di kotak merah didapatkan dari 1+1 (koordinat Y dari huruf S di kiri), 1 +1 (koordinat X dari huruf S di atas), 0 (angka diagonal) + 0 (cost dimana kasus ini huruf S dan S itu sama maka nilainya 0). Hasilnya adalah 2, 2, 0. Ada 3 hasil angka yang didapat dan yang diambil adalah angka terkecil yaitu 0. Maka didapatkan 0 di kotak merah tersebut.

		S	A	Y	A
	0	1	2	3	4
S	1	0			
U	2	1			
K	3				
A	4				

Gambar 3.3 hasil perhitungan *levenshtein* huruf 's' dengan 'u'

Angka 1 yang di kotak merah didapatkan dari 2+1 (koordinat Y dari huruf U di kiri), 0 +1 (koordinat X dari huruf S di atas), 1 (angka diagonal) + 1 (cost dimana kasus ini U dan S itu tidak sama maka nilainya 1). Hasilnya adalah 3, 1, 2. Ada 3 hasil angka yang didapat dan yang diambil adalah angka terkecil yaitu 1. Maka didapatkan 1 di kotak merah tersebut.

		S	A	Y	A
	0	1	2	3	4
S	1	0			
U	2	1			
K	3	2			
A	4				

Gambar 3.4 hasil perhitungan *levenshtein* huruf 's' dengan 'k'

Angka 2 yang di kotak merah didapatkan dari 3+1 (koordinat Y dari huruf K di kiri), 1 +1 (koordinat X dari huruf S di atas), 2 (angka diagonal) + 1 (cost dimana kasus ini K dan S itu tidak sama maka nilainya 1). Hasilnya adalah 4, 2, 3. Ada 3 hasil angka yang didapat dan yang diambil adalah angka terkecil yaitu 2. Maka didapatkan 2 di kotak merah tersebut.

		S	A	Y	A
	0	1	2	3	4
S	1	0			
U	2	1			
K	3	2			
A	4	3			

Gambar 3.5 hasil perhitungan *levenshtein* huruf 's' dengan 'a'

Angka 1 yang di kotak merah didapatkan dari 4+1 (koordinat Y dari huruf A di kiri), 2 +1 (koordinat X dari huruf S di atas), 3 (angka diagonal) + 1 (cost dimana kasus ini U dan S itu tidak sama maka nilainya 1). Hasilnya adalah 5, 3, 4. Ada 3 hasil angka yang didapat dan yang diambil adalah angka terkecil yaitu 3. Maka didapatkan 3 di kotak merah tersebut.

		S	A	Y	A
	0	1	2	3	4
S	1	0	1		
U	2	1			
K	3	2			
A	4	3			

Gambar 3.6 hasil perhitungan *levenshtein* huruf 'a' dengan 's'

Angka 1 yang di kotak merah didapatkan dari 0+1 (koordinat Y dari huruf S di kiri), 2 +1 (koordinat X dari huruf A di atas), 1 (angka diagonal) + 1 (cost dimana kasus ini S dan A itu tidak sama maka nilainya 1). Hasilnya adalah 1, 3, 2. Ada 3 hasil angka yang didapat dan yang diambil adalah angka terkecil yaitu 1. Maka didapatkan 1 di kotak merah tersebut.

Proses ini berlangsung sampai dapat hasil akhir yaitu:

		S	A	Y	A
	0	1	2	3	4
S	1	0	1	2	3
U	2	1	1	2	3
K	3	2	2	2	3
A	4	3	2	3	2

Gambar 3.7 hasil akhir perhitungan *levenshtein* kata "saya" dengan "suka"

Demikian hasil perhitungan dari 2 kalimat tersebut berada dibawah ini:

		S	A	Y	A
	0	1	2	3	4
S	1	0	1	2	3
A	2	1	0	1	2
Y	3	1	1	0	1
A	4	2	1	1	0

		S	A	Y	A
	0	1	2	3	4
T	1	1	2	3	4
I	2	2	2	3	4
D	3	3	3	3	4
A	4	4	3	4	3
K	5	5	4	4	4

		S	A	Y	A
	0	1	2	3	4
S	1	0	1	2	3
U	2	1	1	2	3
K	3	2	2	2	3
A	4	3	2	3	2

		S	A	Y	A
	0	1	2	3	4
D	1	1	2	3	4
I	2	2	2	3	4
A	3	3	2	3	3

Gambar 3.8 hasil perhitungan levenshtein kata "saya" dengan kata contoh pada kalimat ke-2

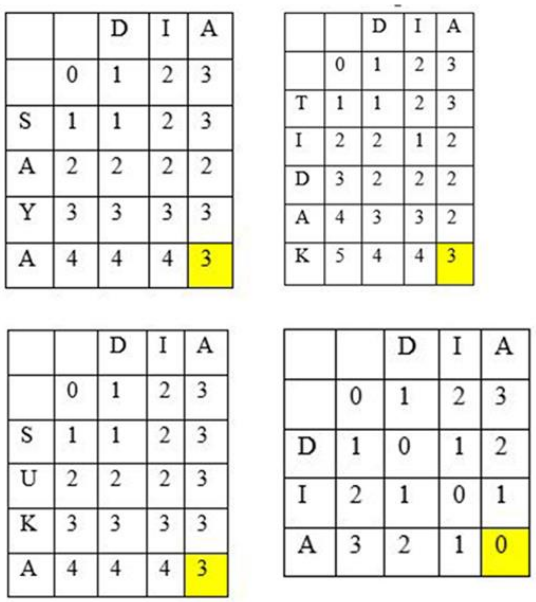
		B	E	N	C	I
	0	1	2	3	4	5
S	1	1	2	3	4	5
A	2	2	2	3	4	5
Y	3	3	3	3	4	5
A	4	4	4	4	4	5

		B	E	N	C	I
	0	1	2	3	4	5
T	1	1	2	3	4	5
I	2	2	2	3	4	4
D	3	3	3	3	4	5
A	4	4	4	4	4	5
K	5	5	5	5	5	5

		B	E	N	C	I
	0	1	2	3	4	5
S	1	1	2	3	4	5
U	2	2	2	3	4	5
K	3	3	3	3	4	5
A	4	4	4	4	4	5

		B	E	N	C	I
	0	1	2	3	4	5
D	1	1	2	3	4	5
I	2	2	2	3	4	4
A	3	3	3	3	4	5

Gambar 3.9 hasil perhitungan *levenshtein* kata "benci" dengan kata pada contoh kalimat ke-2



		D	I	A
	0	1	2	3
S	1	1	2	3
A	2	2	2	2
Y	3	3	3	3
A	4	4	4	3

		D	I	A
	0	1	2	3
T	1	1	2	3
I	2	2	1	2
D	3	2	2	2
A	4	3	3	2
K	5	4	4	3

		D	I	A
	0	1	2	3
S	1	1	2	3
U	2	2	2	3
K	3	3	3	3
A	4	4	4	3

		D	I	A
	0	1	2	3
D	1	0	1	2
I	2	1	0	1
A	3	2	1	0

Gambar 3.10 hasil perhitungan *levenshtein* kata "dia" dengan kata pada contoh kalimat ke-2

Maka kesimpulan dari hasil perhitungan kesamaan kata (*string Matching*) dengan menggunakan algoritma levenshtein distance bahwa dari kedua kata pada contoh di nyatakan tidak sama secara kesamaan kata atau *string*. Tapi pada kenyataannya kedua kalimat memiliki makna yang sama, sehingga perlu adanya proses pengecekan kata atau kalimat dari segi makna atau sinonimnya.

3.1.1.2 Sinonim

Proses pengecekan kata atau kalimat dari segi sinonimnya masih dilakukan dengan cara *brute force* dengan mencocokkan *string* atau kalimat dengan *database* kamus *thesaurus* atau sinonim bahasa Indonesia. Dibawah ini adalah 2 contoh paragraf dari 2 *file* atau dokumen yang akan diuji dari segi sinonim atau kesamaan maknanya. Berikut adalah kalimat dari kedua *file* atau dokumen yang belum mengalami proses *stopwords* maupun *stemming*.

Contoh kalimat dari file asli :

“Memang perkembangan zaman yang modern ini mendekatkan yang jauh tapi menjauhkan yang dekat. Kita semakin jarang untuk berinteraksi dengan sekitar

yang nyata dan yang terlihat oleh mata. Suatu saat nanti perkembangan teknologi yang modern bakal membunuh manusia sebagai penciptanya. Itu terbukti kita seakan mati jika kita kehilangan perangkat yang maju tersebut dibandingkan ditinggalkan oleh orang yang kita sayangi."

Contoh kalimat dari file plagiat:

"Kemajuan di abad yang maju ini mengakrabkan yang jauh tapi menjauhkan yang akrab. Kita semakin kurang berhubungan dengan yang ada sekeliling yang jelas tampak oleh mata. Satu saat manusia akan binasa dan kemajuan teknologi yang maju yang akan melenyapkannya. Itu benar karena orang seakan binasa karena kehilangan alat yang modern dibandingkan ditinggalkan oleh orang yang kita cintai ."

Proses selanjutnya adalah yang akan dilakukan adalah proses *tokenization* untuk memotong atau memisahkan paragraf tersebut menjadi beberapa kalimat. Berikut adalah hasil setelah proses *tokenization*:

Kalimat dari file asli:

Memang perkembangan zaman yang modern ini mendekatkan yang jauh tapi menjauhkan yang dekat
Kita semakin jarang untuk berinteraksi dengan sekitar yang nyata dan yang terlihat oleh mata
Suatu saat nanti perkembangan teknologi yang modern bakal membunuh manusia sebagai penciptanya
Itu terbukti kita seakan mati jika kita kehilangan perangkat yang maju tersebut dibandingkan ditinggalkan oleh orang yang kita sayangi.

Kalimat dari file plagiat:

Kemajuan di abad yang maju ini mengakrabkan yang jauh tapi menjauhkan yang akrab
Kita semakin kurang berhubungan dengan yang ada sekeliling yang jelas tampak oleh mata
Satu saat manusia akan binasa dan kemajuan teknologi yang maju yang akan melenyapkannya
Itu benar karena orang seakan binasa karena kehilangan alat yang modern dibandingkan ditinggalkan oleh orang yang kita cintai .

Proses selanjutnya adalah proses *stemming* dan *stopword* yang akan menghilangkan beberapa kata yang tidak terlalu penting sehingga dapat diproses dengan mudah. Berikut adalah hasil setelah di proses *stopword* dan *stemming.*

asli	
kalimat	1. Kembang zaman modern dekat jauh
	2. Jarang interaksi
	3. Kembang teknologi modern bunuh manusia cipta
	4. Bukti akan mati hilang angkat maju banding tinggal sayang

Plagiat	
kalimat	1. Maju abad maju akrab jauh akrab
	2. Hubung keliling
	3. Maju binasa maju teknologi maju lenyap
	4. Akan binasa hilang alat modern banding tinggal cinta

gambar 3.11 Hasil *preprocessing*

Proses selanjutnya adalah pencocokkan kalimat yang asli dengan yang plagiat. Kalimat – kalimat dari kedua *file* tersebut akan di potong (*split*) per kata untuk proses pencocokkan secara makna. Berikut adalah proses *split* yang dilakukan.

Kembang	Teknologi	Modern	Bunuh	Manusia	cipta

Proses pencocokkan yang dilakukan dengan cara mencocokkan kata dari file asli dan plagiat dan mengecek kedua kalimat tersebut berada dalam satu baris. Jika berada dalam 1 baris maka dianggap sama atau plagiat.

Plagiat					
Maju	Abad	Maju	Akrab	Jauh	akrab

Asli				
Kembang	Zaman	Modern	Dekat	Jauh

Gambar 3.12 Hasil kalimat ke-1 plagiat dengan asli

Plagiat	
Hubung	Keliling

Asli				
Kembang	Zaman	Modern	Dekat	Jauh

Plagiat	
Hubung	Keliling

Plagiat	
Jarang	interaksi

Plagiat	
Hubung	Keliling

Asli					
Kembang	Teknologi	Modern	Bunuh	Manusia	cipta

Plagiat	
Hubung	Keliling

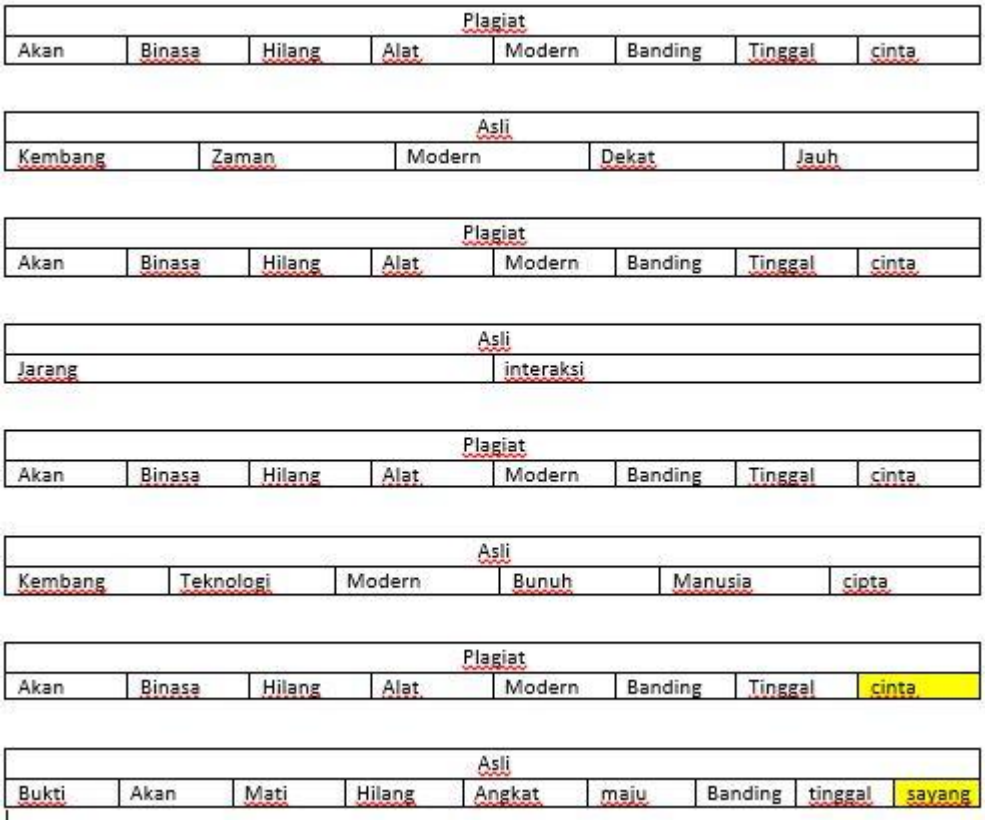
Asli								
Bukti	Akan	Mati	Hilang	Angkat	maju	Banding	tinggal	sayang

Gambar 3.13 Hasil kalimat ke-2 plagiat dengan asli

Plagiat					
Maju	binasa	Maju	teknologi	Maju	lenyap

Asli				
Kembang	Zaman	Modern	Dekat	Jauh

Gambar 3.14 Hasil kalimat ke-3 plagiat dengan asli



Gambar 3.15 Hasil kalimat ke-4 plagiat dengan asli

3.1.2 Analisis Kebutuhan

Bagian ini merupakan penjelasan mengenai kebutuhan-kebutuhan yang diperlukan oleh sistem. Bagian analisis dibedakan atas 2 yaitu analisis kebutuhan fungsional dan analisis kebutuhan non fungsional yang diperlukan untuk mencapai tujuan yang ingin dicapai.

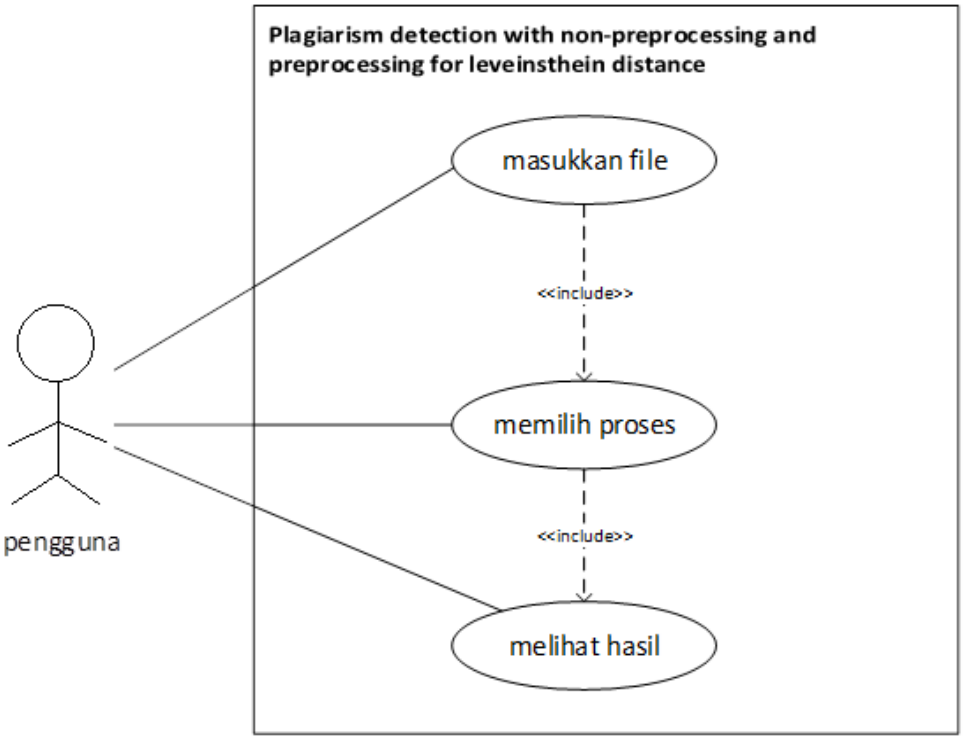
3.1.2.1 Analisis Fungsional

Kebutuhan fungsional adalah pernyataan layanan sistem yang harus disediakan. Bagaimana sistem bereaksi pada *input* tertentu dan bagaimana perilaku sistem pada situasi tertentu. Oleh sebab itu, maka dirumuskan beberapa kebutuhan fungsional pada judul tugas akhir ini:

1. *User* hanya dapat memasukkan *file* berupa pdf pada pengecekan plagiat.

2. *User* hanya bisa memasukkan 2 yaitu 1 dokumen yang original atau asli dan 1 dokumen yang merupakan *file* yang akan diuji.
3. *User* dapat memilih antara 2 pilihan dalam tipe pemrosesan dokumen baik *preprocessing* maupun *non preprocessing*.
4. Sistem dapat berjalan pada dokumen *file* yang berupa pdf dan dokumen berbahasa Indonesia.
5. Sistem tidak akan berjalan pada *file* berupa pdf yang berisi gambar ataupun hasil *scanning*.

Untuk lebih jelasnya kebutuhan fungsional, dapat dilihat *use case* pada gambar berikut ini.



Gambar 3.16 *Use Case*

Penjelasan dari *use case* dari gambar diatas dibuat dalam sebuah narasi yang dapat dilihat pada tabel dibawah ini.

Tabel 3.1 Narasi *Use Case* Masukkan *File*

Nama UseCase	Masukkan *File*.	
Aktor	Pengguna.	
Prakondisi	Pengguna sudah harus memilih *file* yang akan diuji.	
Deskripsi	Proses ini adalah tindakan untuk memasukkan *file* yang akan diuji.	
Sasaran	Isi dari *file* tersebut akan ditampilkan.	
	Aksi Aktor	Respon Sistem
	1. Menekan tombol load. 3. Mencari dan memilih *file* yang akan diuji.	2. Menampilkan tampilan *file* select dialog box. 4. Menampilkan isi *file*.

Tabel 3.2 Narasi Memilih Proses

Nama UseCase	Memilih Proses.	
Aktor	Pengguna.	
Prakondisi	Pengguna sudah dapat memastikan *file* yang akan diuji.	
Deskripsi	Proses ini adalah proses untuk memilih tipe pengecekan atau tipe pengujian.	
Sasaran	Pengguna dapat melihat detil proses yang dilakukan pada saat pengecekan atau pengujian.	
	Aksi Aktor	Respon Sistem
	1. Memilih antara 2 radiobutton yang berupa *Preprocessing* dan *non Preprocessing.* 3. Menunggu proses hingga selesai.	2. Menampilkan tombol proses.

	5. Melihat setiap langkah dalam pengujian.	4. Menampilkan tahapan proses dari tipe pengujian yang telah dipilih.

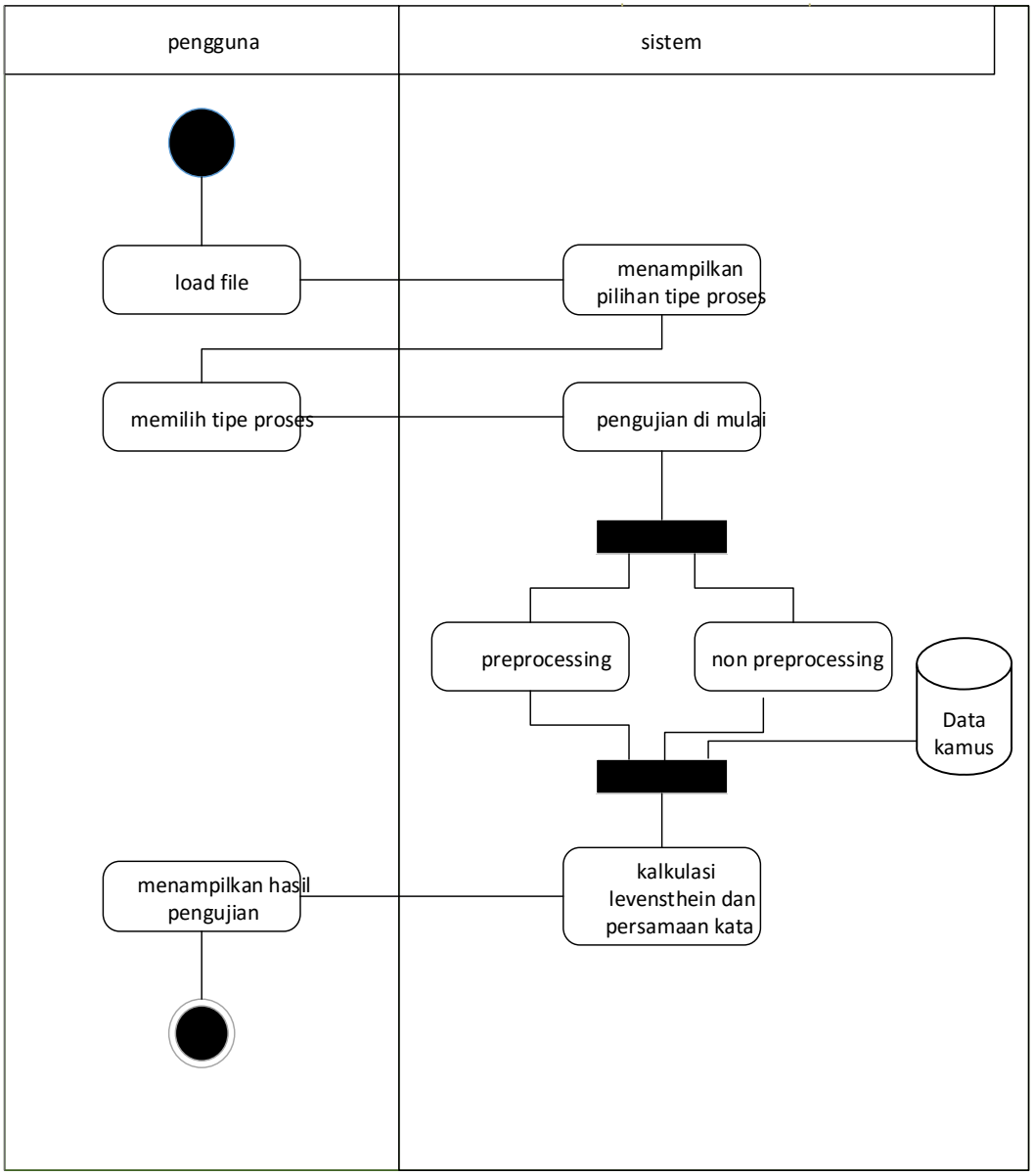
Tabel 3.3 Narasi *Use Case* melihat hasil

Nama UseCase	Melihat Hasil.	
Aktor	Pengguna.	
Prakondisi	Pengguna harus menekan tombol Result.	
Deskripsi	Proses ini adalah proses yang menunjukkan hasil akhir dari pengujian atau pengecekan yang dilakukan.	
Sasaran	Menampilkan hasil akhir dalam bentuk persentase.	
	Aksi Aktor	Respon Sistem
	3. Melihat hasil akhir dari proses pengujian yang dilakukan.	1. Menampilkan persentase nilai dari hasil perhitungan persamaan kata dan perhitungan levenshtein. 2. Menentukan suatu *file* dianggap plagiat atau tidak.

Activity diagram pada gambar 3.17 di bawah ini menjelaskan alur yang dilakukan baik pengguna maupun sistem. Proses dimulai dari pengguna memasukkan dokumen yang akan diuji kemudian sistem akan menampilkan pilihan

tipe pemrosesan yaitu preprocessing dan non preprocessing. Pengguna akan melakukan pemilihan tipe pemrosesan. Kemudian sistem akan memproses dokumen tergantung dari pilihan pemrosesan yang telah dipilih oleh pengguna sebelumnya.

Sistem akan melakukan kalkulasi perhitungan dari segi kesamaan kata *(string matching)* dan segi kesamaan makna atau sinonim. Setelah selesai melakukan perhitungan oleh sistem, maka sistem akan menampilkan hasil akhir perhitungan dan menentukan dokumen tersbut dapat dikatakan plagiat atau tidak.



Gambar 3.17 Activity diagram

3.1.2.2 Analisis Non Fungsional

Kebutuhan non fungsional adalah kebutuhan yang secara tidak langsung memberikan peran dalam fitur tertentu di dalam sistem. Kebutuhan yang tidak memerlukan tahapan peng*input*an, tahapan proses maupun tahapan output kebutuhan non fungsional sebaiknya di penuhi, karena dapat menentukan sistem akan digunakan oleh pengguna atau tidak. Maka harus dilakukan analisis terhadap kinerja, informasi, ekonomis, keamanan, efisiensi dan pelayanan *customer*. Analisis yang digunakan adalah analisis PIECES *(Performance, Information, Economic, Efficiency, and Services).*

1. Performance

Perangkat lunak harus dapat melakukan eksekusi secara langsung setelah adanya aksi dari user. Lamanya proses eksekusi tergantung pada banyaknya kata di dokumen asli dan dokumen plagiat.

2. Information

Perangkat lunak harus mampu menampilkan berbagai informasi yang mendukung bagi pengguna dalam pengisian data *input*.

3. Economic

Perangkat lunak tidak memerlukan perangkat pendukung lainnya dalam proses eksekusinya. Perangkat lunak yang dibutuhkan hanya Microsoft Visual Studio 2013.

4. Control

Perangkat lunak akan menampilkan pesan kesalahan apabila terdapat kesalahan atau kegagalan sistem.

5. Efficiency

Pengguna hanya melakukan 2 proses seperti memasukkan dokumen yang akan diuji dan melakukan pemilihan tipe pemrosesan. Kemudian sistem akan melakukan proses pengecekan dan perhitungan dengan waktu relatif sesuai dengan jumlah kata pada dokumen yang telah dimasukkan.

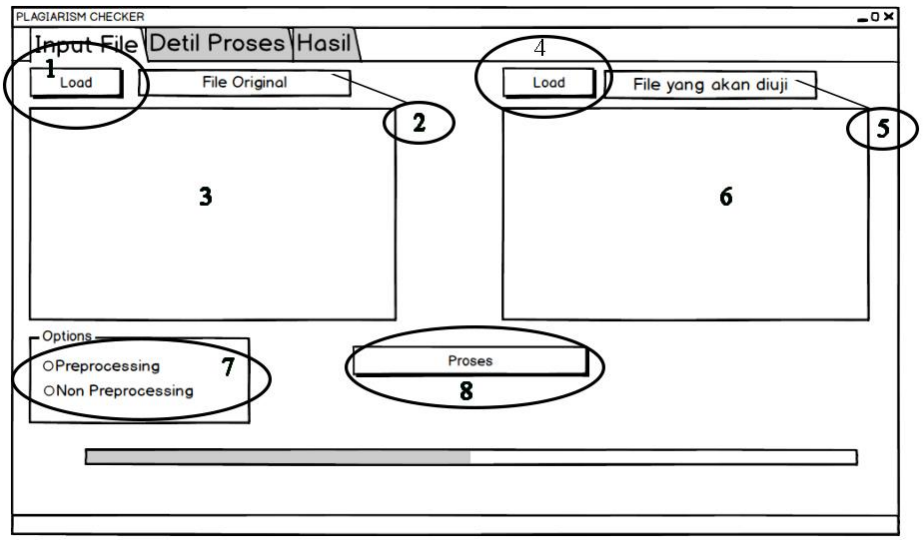
6. Services

Hasil dari perangkat lunak akan membantu pengguna dalam menentukan suatu dokumen merupakan hasil plagiarisme atau tidak.

3.2 Perancangan *User Interface*

User interface merupakan salah satu bagian penting dalam sebuah sistem, karena bagian ini yang menghubungkan pengguna dengan sistem. Tampilan user interface yang baik dan menarik akan mempermudahkan pengguna dalam menggunakan sistem yang akan dibuat.

3.2.1 Form Input File



Gambar 3.18 *Form Input file*

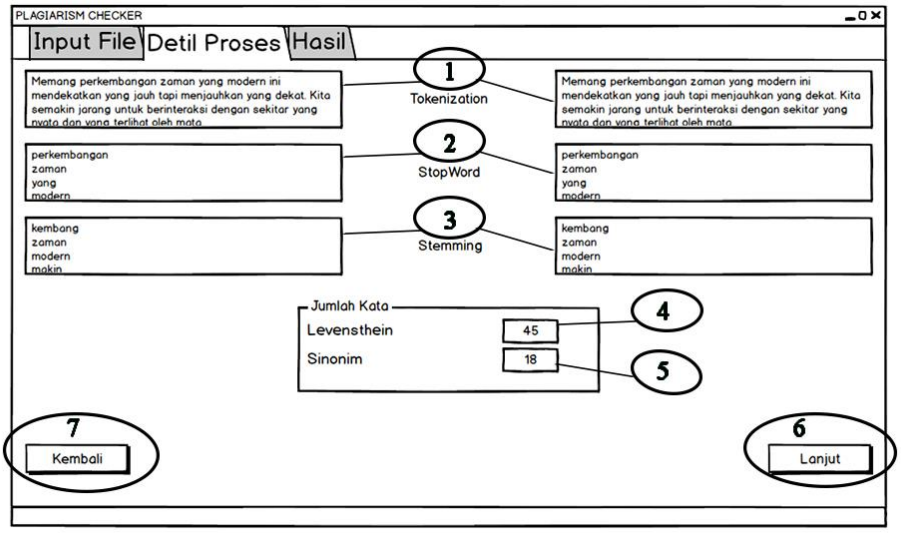
Pada gambar 3.18 yang ada diatas merupakan *form* untuk melakukan peng-*input*-an dokumen yang akan dilakukan pengecekan.

Keterangan :

1. Pada *form* tampilan *input file* merupakan tombol untuk meng-*input* atau memasukkan dokumen. Dokumen yang di-*input*/dimasukkan merupakan dokumen yang menjadi acuan atau dokumen yang original.
2. Pada *form* tampilan *input file* merupakan *textbox* atau tempat di mana informasi dari dokumen yang telah di masukkan sebelumnya ditampilkan.
3. Pada *form* tampilan *input file* merupakan *text area* atau tempat dimana isi dokumen yang berupa teks akan ditampilkan secara rinci.

4. Pada *form* tampilan *input file* merupakan tombol untuk meng*input* atau memasukkan dokumen yang akan diuji.
5. Pada *form* tampilan *input file* merupakan textbox atau tempat di mana informasi dari dokumen ditampilkan.
6. Pada *form* tampilan *input file* merupakan *text area* atau tempat dimana isi dokumen yang berupa teks akan ditampilkan secara rinci.
7. Pada *form* tampilan *input file* merupakan *radiobutton* yang digunakan untuk memilih antara 2 tipe proses pengecekan yaitu proses pengecekan secara *preprocessing* dan *non preprocessing.*
8. Pada *form* tampilan *input file* merupakan tombol untuk memulai proses pengecekan terhadap kedua dokumen yang telah dimasukkan.

3.2.2 *Form* Detil Proses (*Preprocessing*)



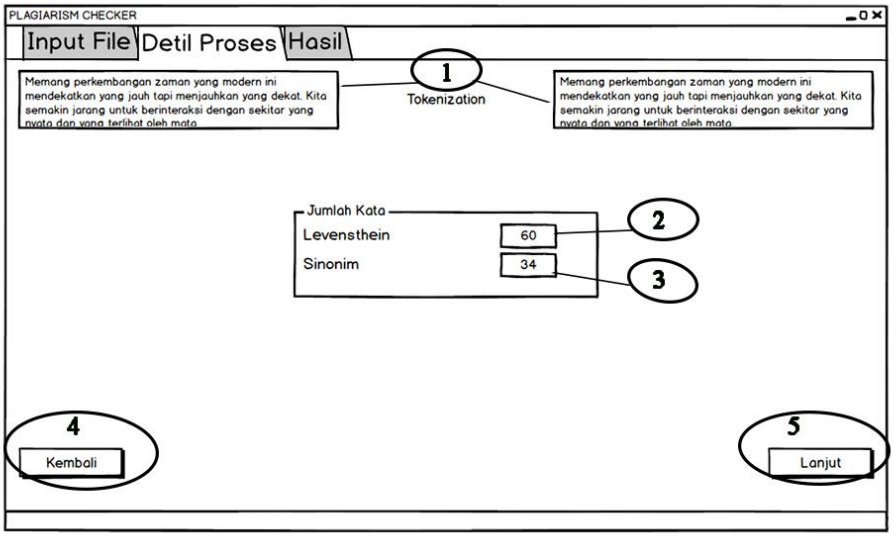
Gambar 3.19 *Form* Detil Proses (*Preprocessing*)

Pada gambar 3.19 yang ada diatas merupakan *form* yang muncul jika pengguna memilih tipe pengecekan dengan preprocessing. *Form* ini menampilkan setiap tahapan proses pengecekan secara detil.

Keterangan :

1. Pada *form* detil proses *(Preprocessing)* merupakan tempat yang menampilkan tampilan hasil *tokenization* atau proses pemotongan sebuah paragraph menjadi per kalimat pada dokumen yang dimasukkan sebelumnya.
2. Pada *form* detil proses *(preprocessing)* merupakan tempat untuk menampilkan tampilan hasil *stopwords* atau proses penghapusan kata-kata yang tidak penting dalam sebuah kalimat.
3. Pada *form* detil proses *(preprocessing)* merupakan tempat untuk menampilkan tampilan hasil *stemming* atau proses penghilangan imbuhan yang terdapat pada setiap kata pada kalimat.
4. Pada *form* detil proses *(preprocessing)* merupakan tempat untuk menampilkan hasil dari proses pengecekan kata yang dianggap plagiat dari segi kesamaan kata atau *string matching.*
5. Pada *form* detil proses *(preprocessing)* merupakan tempat untuk menampilkan hasil dari proses pengecekan kata yang dianggap plagiat dari segi persamaan kata atau sinonim.
6. Pada *form* detil proses *(preprocessing)* merupakan tombol untuk melanjutkan ke *form* selanjutnya yaitu *form* hasil.
7. Pada *form* detil proses (preprocessing) merupakan tombol untuk kembali ke *form input file.*

3.2.3 *Form* Detil Proses (*Non Preprocessing*)



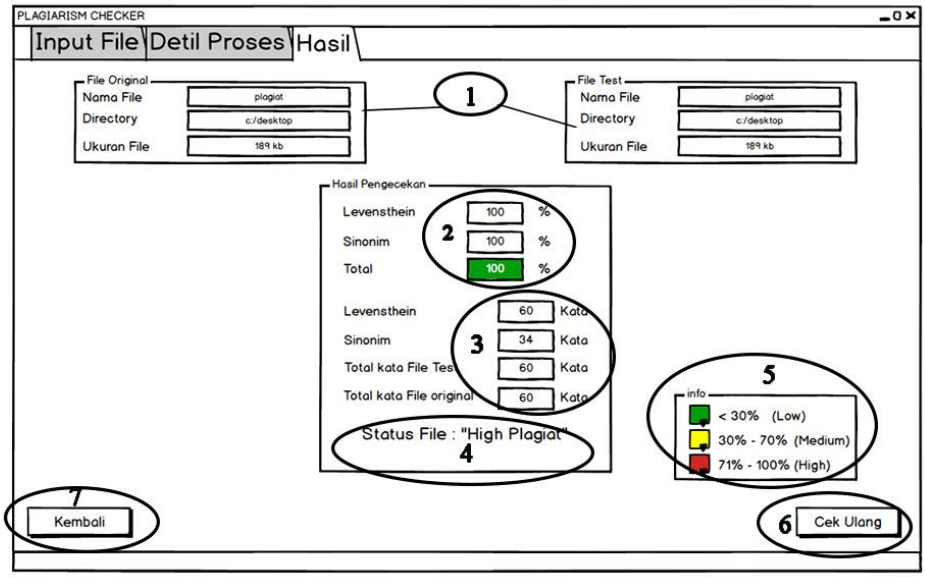
Gambar 3.20 *Form* Detil Proses (*Non Preprocessing*)

Pada gambar 3.20 yang ada diatas merupakan *form* yang muncul jika pengguna memilih tipe pengecekan dengan *non preprocessing*. *Form* ini menampilkan setiap tahapan proses pengecekan secara detil.

Keterangan :

1. Pada *form* detil proses *(non Preprocessing)* merupakan tempat yang menampilkan tampilan hasil *tokenization* atau proses pemotongan sebuah paragraph menjadi per kalimat pada dokumen yang dimasukkan sebelumnya.
2. Pada *form* detil proses *(non preprocessing)* merupakan tempat untuk menampilkan hasil dari proses pengecekan kata yang dianggap plagiat dari segi kesamaan kata atau *string matching.*
3. Pada *form* detil proses *(non preprocessing)* merupakan tempat untuk menampilkan hasil dari proses pengecekan kata yang dianggap plagiat dari segi persamaan kata atau sinonim.
4. Pada *form* detil proses *(non preprocessing)* merupakan tombol untuk melanjutkan ke *form* selanjutnya yaitu *form* hasil.
5. Pada *form* detil proses (*non preprocessing*) merupakan tombol untuk kembali ke *form input file.*

3.2.4 *Form* Hasil



Gambar 3.21 *Form* Hasil

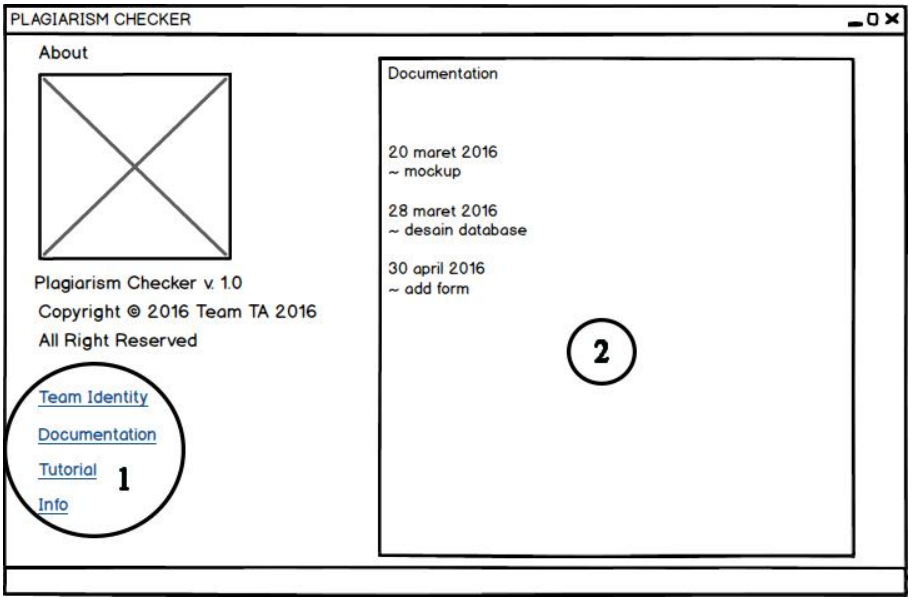
Pada gambar 3.21 yang ada diatas merupakan *form* yang menampilkan hasil pengecekan dokumen.

Keterangan :

1. Pada *form* hasil merupakan tampilan yang menampilkan informasi dari setiap dokumen.
2. Pada *form* hasil merupakan tampilan yang menampilkan hasil pengecekan dalam bentuk persentase.
3. Pada *form* hasil merupakan tampilan yang menampilkan jumlah kata pada setiap dokumen baik dokumen yang original dengan dokumen yang diuji, dan menampilkan jumlah kata menurut kesamaan kata (*string matching*) dan persamaan kata (sinonim)
4. Pada *form* hasil merupakan *label* yang akan menunjukkan status dari dokumen dapat dikatakan plagiat atau tidak
5. Pada *form* hasil merupakan tampilan yang menampilkan informasi tingkatan dari plagiat

6. Pada *form* hasil merupakan tombol untuk melakukan pengecekan ulang terhadap dokumen yang dicek ataupun melakukan pengecekan dokumen lain
7. Pada *form* hasil merupakan tombol untuk kembali ke *form* detil proses

3.2.5 Form about



Gambar 3.22 *Form About*

Pada gambar 3.22 yang ada diatas merupakan *form* yang menampilkan informasi tentang perangkat lunak.

Keterangan :

1. Pada *form about* merupakan *link label* yang menjadi penghubung untuk menampilkan setiap informasi sesuai dengan kategori yang ada.
2. Pada *form about* merupakan *text area* atau bagian yang akan menampilkan setiap informasi yang muncul pada saat *link label* di klik.

BAB IV
HASIL DAN PENGUJIAN

Pada bagian bab ini akan dijelaskan tentang langkah-langkah dalam menggunakan sistem serta melakukan pengujian terhadap kemampuan sistem. Sistem yang dibahas adalah "Sistem Pengujian Deteksi Tingkat Plagiarisme dengan Mempertimbangkan Makna Kata dengan Menggunakan Algoritma *Levenshtein Distance*. Bahan untuk pengujian berupa file dokumen berformat *.pdf.

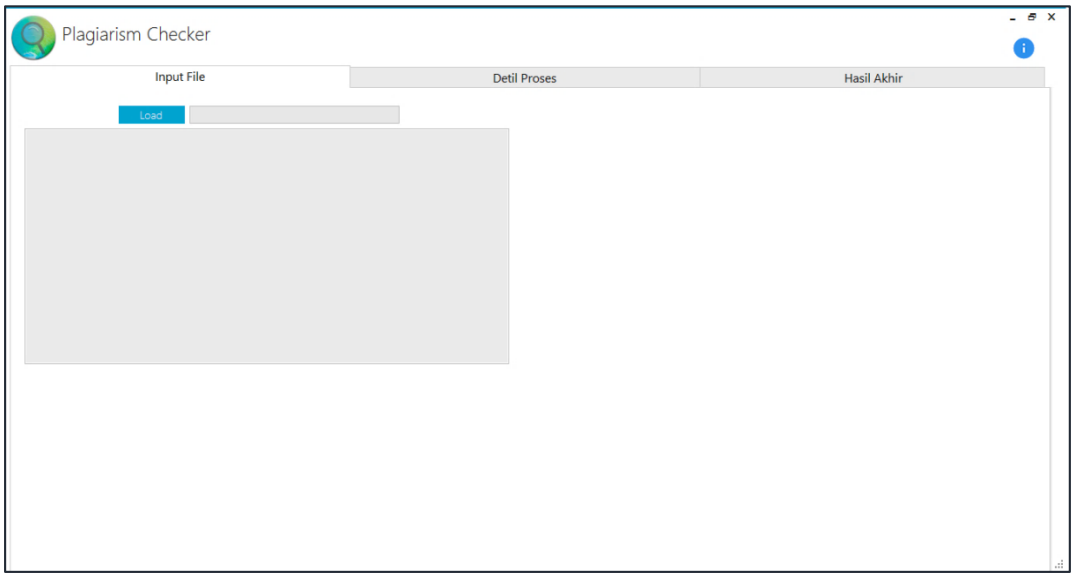
4.1 Hasil

Pada subbab ini akan dijelaskan tampilan dari sistem serta pembahasan mengenai cara menggunakan sistem. Berikut tampilan sistem.

4.1.1 Tampilan pada sistem

1. Tampilan *input file*

Berikut adalah tampilan awal dari aplikasi pada saat di jalankan, dapat dilihat pada gambar 4.1. Pada tampilan ini pengguna harus melakukan klik pada tombol *load* untuk menginput file yang menjadi acuan dalam proses pengecekan yang akan dilakukan.

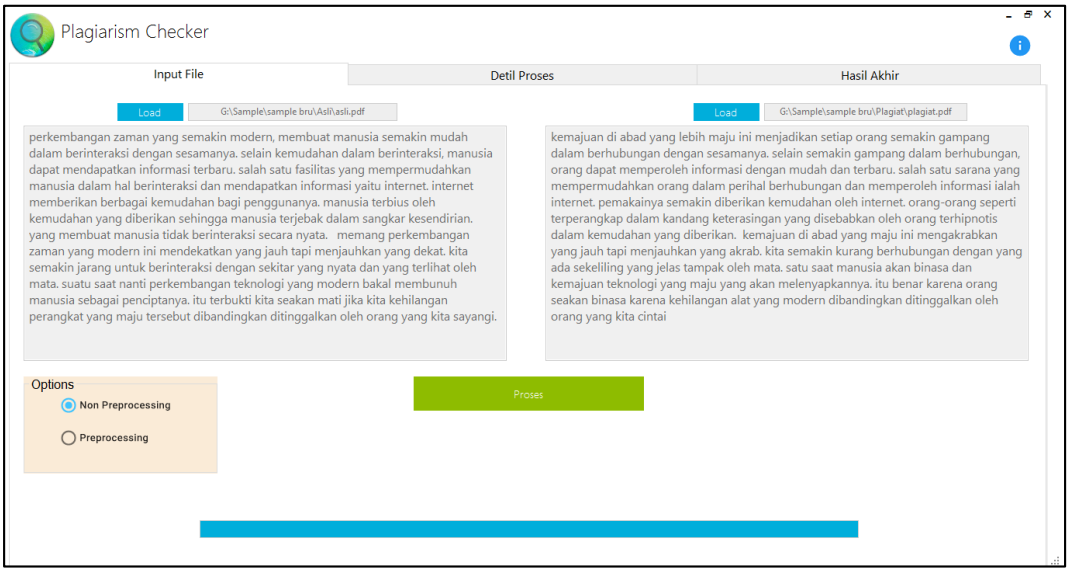


Gambar 4.1 Tampilan *input file*

2. Tampilan proses input file dan pemilihan tipe pengecekan

Berikut adalah tampilan pada saat file telah diinput dan telah melakukan pemilihan tipe pengecekan. Pada tampilan dibawah adalah tampilan pada saat pengguna telah menginput file yang menjadi acuan dan menginput file yang akan diuji.

Pada saat kedua file yang akan dicek telah diinput maka akan keluar pilihan tipe pemrosesan yang terdiri dari tipe *non preprocessing* dan *preprocessing*. Kedua tipe tersebut memiliki tujuan untuk mempersiapkan kedua file yang akan diuji untuk lanjut ke proses selanjutnya yaitu proses *levenshtein* dan sinonim

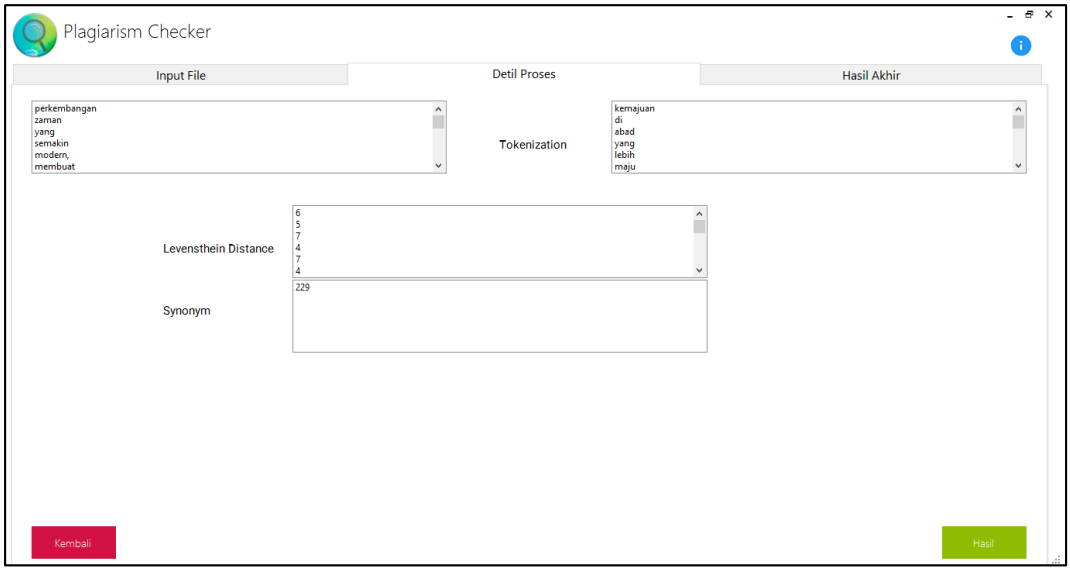


Gambar 4.2 Tampilan proses *input file* dan pemilihan tipe pengecekan

3. Tampilan Detil Proses *(non preprocessing)*

Berikut tampilan detil proses pada saat melakukan pemilihan tipe pengecekan *non preprocessing*. Pada tampilan detil proses yang *non preprocessing* ini akan menampilkan tiap – tiap proses yang seharusnya berada di belakang layar. Tetapi ditampilkan untuk menunjukkan kepada pengguna proses secara detil. Terdapat 2 listbox yang berisi kata – kata, itu merupakan tampilan kalimat yang telah di split atau dipisah per spasi sehingga kata terurut per baris.

Pada bagian bawah terdapat 2 listbox yang berisi angka – angka, itu merupakan hasil dari proses pengecekan dengan menggunakan algoritma levenshtein dan pengecekan secara sinonim. Pada bagian listbox pertama merupakan hasil pengecekan levenshtein yang mengecek string / kata yang sama atau *string matching*. Jika ada kata yang sama secara string maka hasil di listbox tersebut adalah ‘0’ dan jika ada kata yang tidak sama secara string maka hasil di listbox tersebut sesuai jumlah perbedaan huruf pada kata.

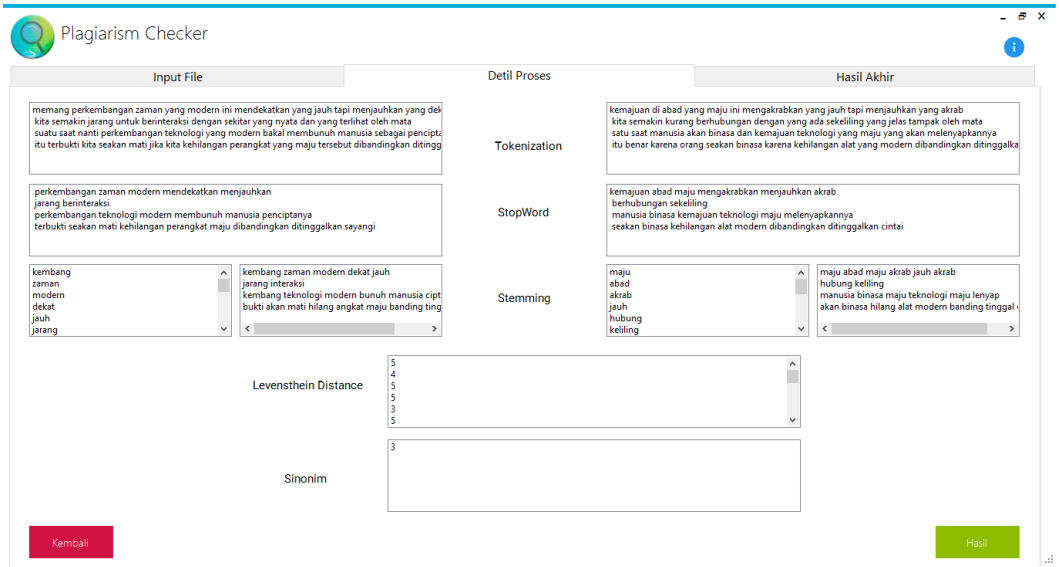


Gambar 4.3 Tampilan Detil Proses *(non preprocessing)*

4. Tampilan Detil Proses *(Preprocessing)*

Berikut tampilan detil proses pada saat melakukan pemilihan tipe pengecekan *preprocessing*. Pada tampilan detil proses ini akan menampilkan tiap – tiap proses dalam pengecekan file. Proses – proses pada preprocessing ditampilkan secara menyeluruh mulai dari proses tokenization, stopword, stemming serta hasil pengecekan awal dari algoritma levenshtein dan sinonim.

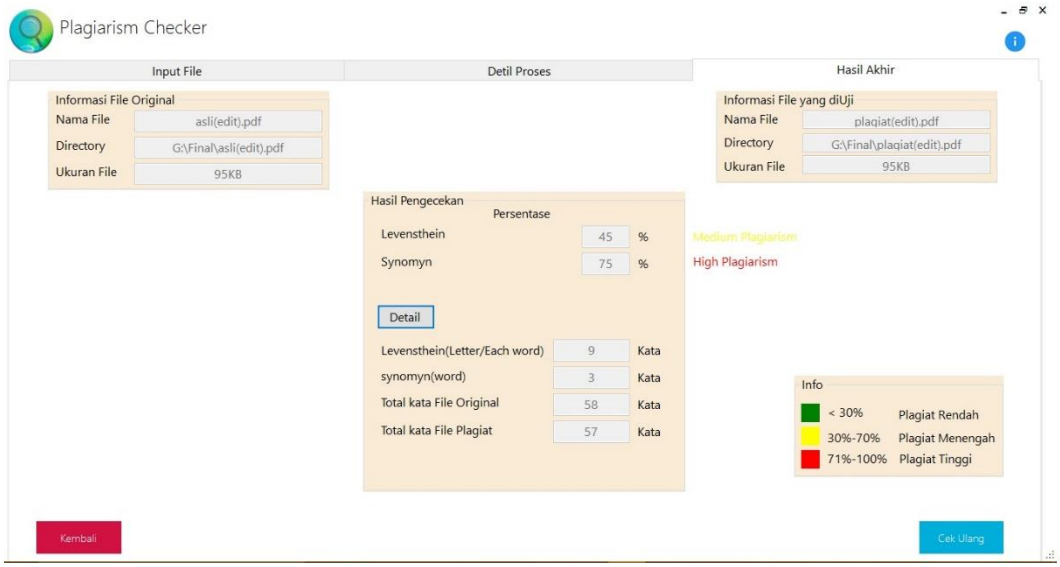
Pada listbox bagian paling atas adalah proses tokenization yang membagi sebuah paragraph menjadi beberapa kalimat. Pada bagian bawahnya merupakan listbox untuk proses stopword yang menghapus atau menghilangkan sebagian kata yang tidak diperlukan. Pada bagian bawah dari listbox stopword adalah listbox untuk proses stemming dibagian terdapat masing – masing 2 listbox antara yang file acuan dan file plagiat. Bagian listbox yang menampilkan kalimat yang di split per kata dan dibuat berurut dimaksudkan untuk proses pengecekan dengan menggunakan algoritma levenshtein untuk mencari kesamaan pada kata atau string. Sedangkan listbox yang menampilkan kalimat dimaksudkan untuk proses pengecekan sinonim.



Gambar 4.4 Tampilan Detil Proses *(preprocessing)*

5. Tampilan hasil akhir

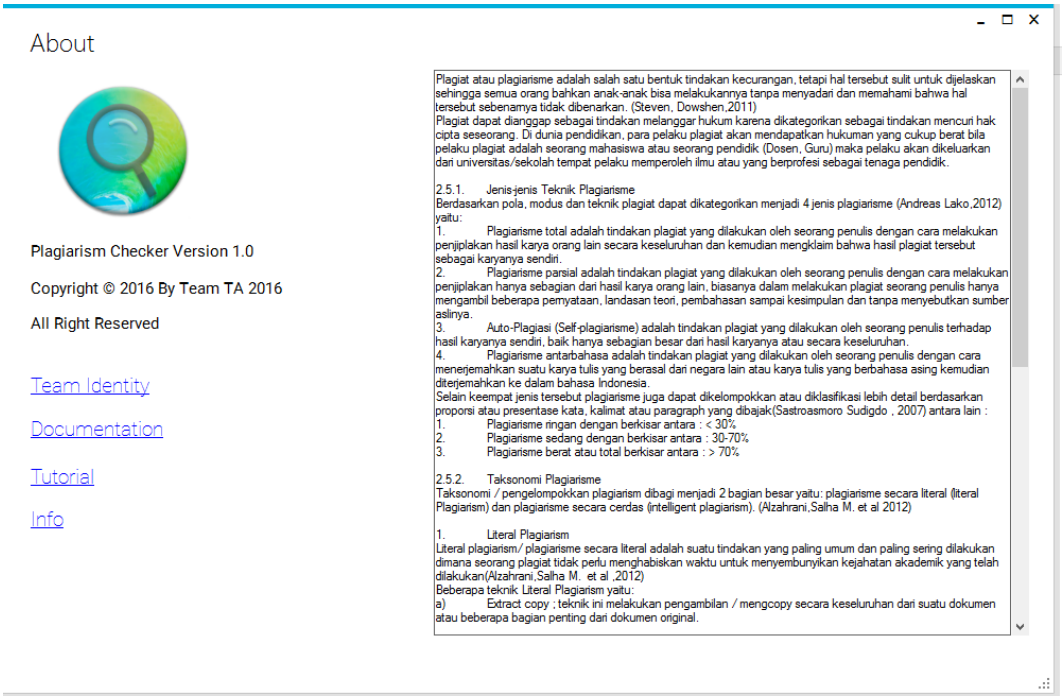
Berikut tampilan hasil akhir dari proses pengecekan yang telah di lakukan. Pada bagian tampilan ini akan menampilkan hasil perhitungan dari proses pengecekan dengan menggunakan levenshtein maupun pengecekan secara sinonim. Pada tampilan ini juga menampilkan informasi dari file yang telah di input, menampikan jumlah kata pada kedua file yang telah di input.



Gambar 4.5 Tampilan hasil akhir

6. Tampilan *about*

Berikut tampilan *about* dari aplikasi, pada tampilan ini akan menampilkan informasi sekilas mengenai plagiarisme, dokumentasi atau track record dalam pengembangan sistem. Selain menampilkan dokumenstasi dan informasi juga dapat menampilkan tutorial atau tata cara penggunaan pada sistem.



Gambar 4.6 Tampilan *about*

4.2 Pengujian

Untuk menguji kemampuan sistem dalam menentukan suatu file di nyatakan sebagai plagiat atau tidak maka di lakukan pengujan dengan menggunakan 10 file dokumen yang original dan 10 file yang dianggap plagiat. Prosedur pengujian adalah memasukkan file yang original dan file yang akan diuji kedalam sistem, Kemudian proses pengecekan akan dimulai. Hasil dari pengecekan dalam bentuk persentase (%). Ada 3 hasil yang akan didapatkan yaitu hasil dari *levenshtein distance*, hasil dari sinonim, dan total.

Berikut ini adalah tabel daftar dokumen asli dan dokumen pembanding (plagiat) yang akan diuji. Proses pengujian dilakukan sebanyak 2 kali yaitu pengujian normal antara dokumen asli dan yang dianggap plagiat dan pengujian secara random antara dokumen asli dan dokumen pembanding.

Tabel 4.1 Daftar dokumen asli dan dokumen plagiat (pembanding)

No.	Dokumen asli	Isi dokumen asli	Dokumen plagiat	Isi dokumen plagiat
1	Asli.pdf	"Dampak dari perkembangan zaman yang modern"	Plagiat.pdf	"kemajuan di abad yang lebih maju"
2	Asli2.pdf	"Guru pahlawan nasional"	Plagiat2.pdf	"pengajar pahlawan bangsa"
3	Asli3.pdf	"Nasionalisme bangsa"	Plagiat3.pdf	"Semangat kebangsaan indonesia"
4	Asli4.pdf	"Akibat dari perang"	Plagiat4.pdf	"Bentrokan senjata "
5	Pidato Michelle Obama (25 agustus 2008)	"Pidato di democratic national convention 2008"	Pidato Melania Trump (18 juli 2016)	"Pidato di republic national convention 2016"
6	Rashomona kutagawa Ryunosuke	"Kota Kyoto yang sepi"	Perempuan Tua dalam rashomon dadang ari murtono	"Kota Kyoto yang ramai dan permai mulai sepi"
7	Menguliti Bakal Calon Presiden Indonesia 2014 (Satriawan kompas, 19 januari 2012)	"Pemilu untuk tahun 2014"	Nanda Dyani Amilla, Menguliti Bakal Calon Presiden 2014 (11 maret 2014)	"Kasak-kusuk tentang siapa calon presiden Indonesia tahun 2014"
8	Berburu cenderamata di bloemenmarkt (kompas 6 agustus 2011)	"Berburu cenderamata di bloemenmarkt"	Novel amsterdam IHJ, hal 238 (arumi, E 2013)	"oleh –oleh dari bloemenmarkt"
9	jalan-jalan di amsterdam (Javamilk, 29 january 2009)	"jalan-jalan di amsterdam"	Amsterdam IHJ, hal 104-105 (arumi,E ,2013)	"perjalanan di kota amsterdam"
10	Keukenhof Secuil Taman Surga Di Daratan Eropa (rohman,2010)	"Keukenhof Secuil Taman Surga Di Daratan Eropa (Belanda)"	Amsterdam Ik Hou Van Je (Arumi E,2013)	Keindahan taman di kota keukenhof

Berikut ini adalah table hasil pengujian dengan menggunakan aplikasi pengujian deteksi tingkat plagiarisme dengan mempertimbangkan makna kata dengan menggunakan algoritma *levenshtein distance*.

Tabel 4.2 Pengujian terhadap dokumen asli dan pembanding

No	File Asli	File Plagiat	Preprocessing		Non Preprocessing	
			Levenshtein (%)	Sinonim (%)	Levenshtein (%)	Sinonim (%)
1	Asli.pdf	Plagiat.pdf	44,117	77,777	42,5	100
2	Asli2.pdf	Plagiat2.pdf	44,736	100	48,888	100
3	Asli3.pdf	Plagiat3.pdf	61,290	100	59,493	100
4	Asli4.pdf	Plagiat4.pdf	47,222	100	43,478	100
5	Pidato Michelle Obama (25 agustus 2008)	Pidato Melania Trump (18 juli 2016)	60,714	0	59,420	100
6	Rashomona kutagawa Ryunosuke	Perempuan Tua dalam rashomon dadang ari murtono	76,119	90	62,616	90
7	Menguliti Bakal Calon Presiden Indonesia 2014 (Satriawan kompas, 19 januari 2012)	Nanda Dyani Amilla, Menguliti Bakal Calon Presiden 2014 (11 maret 2014)	84,905	71,428	76,530	85,714
8	Berburu cenderamata di bloemenmarkt (kompas 6 agustus 2011)	Novel amsterdam IHJ, hal 238 (arumi, E 2013)	83,050	66,666	79,347	100
9	jalan-jalan di amsterdam (Javamilk, 29 january 2009)	Amsterdam IHJ, hal 104-105 (arumi,E ,2013)	70,2702	42,857	75,581	85,714
10	Keukenhof Secuil Taman Surga Di Daratan Eropa (rohman,2010)	Amsterdam Ik Hou Van Je (Arumi E,2013)	88,888	0	72,549	100

Dari tabel 4.2 dapat simpulkan bahwa pada pengujian yang dilakukan dengan menggunakan dokumen asli dan dokumen pembanding (plagiat) menunjukkan hasil yang berbeda sesuai dengan tipe pemrosesan yang digunakan dalam pengecekan. Pada proses *preprocessing*, proses-proses yang dilakukan adalah proses *tokenization*, *stopwords* dan *stemming*. Sedangkan proses *non preprocessing*, proses yang dilakukan hanya *tokenization* untuk memecahkan sebuah paragraf menjadi beberapa kalimat dan juga kata-katanya. Pengujian pada *non preprocessing* menggunakan proses *tokenization* mempermudah pengerjaan proses *levenshtein distance* dan sinonim tanpa menghilangkan informasi yang ada.

Pada tabel pengujian tersebut menunjukkan dengan menggunakan tipe pemrosesan *preprocessing* dokumen yang memiliki kemiripan dari segi kata (*string matching*) menghasilkan nilai persentase menengah ke atas antara 44% hingga 88 persen, sedangkan dokumen yang memiliki kemiripan dari segi makna (sinonim) menghasilkan nilai persentase yang cukup tinggi antara 40% hingga 100% ini menunjukkan bahwa dokumen yang diuji memiliki kesamaan makna kata dalam kalimat cukup tinggi. Adapun dokumen yang menunjukkan nilai persentase 0% itu menunjukkan tidak adanya kata yang memiliki makna yang sama tetapi pada hasil levenshtein distance menunjukkan persentase yang tinggi sehingga dapat disimpulkan penjiplakannya kata-kata hampir semuanya seperti pada pengujian no 8.

Sedangkan dengan menggunakan tipe pemrosesan *non preprocessing* dokumen yang memiliki kemiripan dari segi kata *(string matching)* menghasilkan persentase yang lebih kecil dibanding hasil pada tipe pemrosesan *preprocessing* antara 42% hingga 79% ini disebabkan karena adanya proses-proses untuk mempersiapkan dokumen untuk dilakukan proses perhitungan lebih lanjut. Proses-proses yang dimaksud adalah proses *stopwords* dan *stemming* yang mengakibatkan sebagian kata mengalami penghilangan atau penghapusan sehingga membuat sebagian informasi menghilang, sedangkan dokumen yang memiliki kemiripan dari segi makna menghasilkan persentase yang lebih besar dibanding hasil pada tipe pemrosesan secara *preprocessing* yaitu antara 82% hingga 100% ini disebabkan

tidak adanya kehilangan informasi karena tidak proses *stopwords* dan *stemming* pada tipe pemrosesan *non preprocessing*.

Pada pengujian no 1 sampai no 4 adalah sampel yang dibuat sendiri untuk menguji makna dengan merubah kata yang memiliki makna sama pada file asli ke file pembandingnya. Hasil pengujiannya cukup sesuai yang diharapkan yaitu pada proses *preprocessing* persentasenya 77.77%-100% dan pada proses *non preprocessing* 100%.

Tabel 4.3 Pengujian dengan sample yang di acak

No	File Asli	File Plagiat	Preprocessing		Non Preprocessing	
			Levensthein (%)	Sinonim (%)	Levensthein (%)	Sinonim (%)
1	Asli.pdf	Plagiat 4.pdf	14,285	60	12,5	100
2	Asli 2.pdf	Pidato Melania Trump(18 juli 2016).pdf	13,157	40	18,888	100
3	Asli 3.pdf	Perempuan Tua dalam rashomon dadang ari murtono.pdf	0	30	10,280	90
4	Berburu cenderamata di bloemenmarkt(kompas 6 agustus 2011).pdf	Amsterdam IHJ, hal 104-105(arumi,E ,2013).pdf	3,389	28,571	8,695	85,714
5	jalan-jalan di amsterdam(Javamilk, 29 january 2009).pdf	Nanda Dyani Amilla, Menguliti Bakal Calon Presiden 2014(11 maret 2014).pdf	0	14,285	12,765	85,714
6	Keukenhof Secuil Taman Surga Di Daratan Eropa(rohman,2010).pdf	Novel amsterdam IHJ, hal 238(arumi, E 2013).pdf	3,636	33,333	9,638	66.666
7	Menguliti Bakal Calon Presiden Indonesia 2014(Satriawan kompas, 19 januari 2012).pdf	Amsterdam Ik Hou Van Je(Arumi E,2013).pdf	0	33,333	6,122	66.666
8	Pidato Michelle Obama(25 agustus 2008).pdf	Perempuan Tua dalam rashomon dadang ari murtono.pdf	0	30	5,607	100
9	Rashomona kutagawa Ryunosuke.pdf	Nanda Dyani Amilla, Menguliti Bakal Calon Presiden 2014(11 maret 2014).pdf	1,818	42,857	7,4468	100
10	Asli 4.pdf	Plagiat.pdf	13,888	66,666	14,102	100

Pada tabel 4.3 pengujian dilakukan dengan dokumen yang di acak pengujian ini berbeda dengan pengujian pada tabel 4.2. Pada tabel pengujian 4.3 dapat disimpulkan bahwa meskipun dokumen diuji secara acak masih menghasilkan presentase yang cukup tinggi. Itu terbukti dari pengujian dengan menggunakan tipe pemrosesan *non preprocessing* meskipun dokumen yang diuji berbeda tetap menghasilkan nilai yang tinggi pada pengujian dokumen secara kemiripan makna hingga mencapai persentase 100%. Ini disebabkan karena proses pembobotan atau pemberian nilai pada setiap kalimat dan kebanyakan makna yang sama yaitu kata-kata dari *stopwords*, misalnya "di", "dan", "sebab", "adalah", "dengan", dan lain-lainnya.. Pada saat proses pengecekan sinonim setiap kalimat yang memiliki 1 kata yang makna nya sama dengan kalimat yang diuji akan diberi bobot atau point 1. Sehingga pada saat menggunakan tipe pemrosesan ini, nilai persentase sinonim tinggi. Sedangkan pada nilai dari segi kemiripan kata *(string matching)* menghasilkan nilai yang rendah dikarenakan tidak ada kata yang sama dalam jumlah yang besar.

Pada tipe pemrosesan *preprocessing* persentase dari segi kemiripan kata *(string matching)* tetap rendah dikarenakan isi dari dokumen telah kehilangan sebagian kata karena adanya proses *stopwords* dan *stemming* persentase nilai levensthein berkisar antara 1,8% hingga 14%. Adapun hasil yang mendapatkan nilai 0% karena tidak kata yang sama. Sedangkan nilai persentase dari kemiripan makna (sinonim) juga menghasilkan nilai yang relative tinggi, persentase berkisar antara 14% hingga 60 persen disebabkan karena proses pembobotan pada proses pengecekan kalimat.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan terhadap kinerja sistem, dapat diambil kesimpulan sebagai berikut:

1. Dari pengujian yang dilakukan sistem dapat mengenali berapa banyak kalimat yang memiliki kemiripan makna(sinonim). Serta dapat mengenali dan mendeteksi kata yang mirip(*string matching*).
2. Nilai persentase sinonim yang dihasilkan cukup baik pada pengecekan dokumen secara normal maupun pengecekan secara random, meskipun pada pengecekan random nilai persentase sinonim tetap tinggi dikarenakan pemberian bobot pada kalimat yang memiliki makna yang sama dengan kalimat yang diuji akan diberi bobot 1 sehingga menghasilkan persentase yang tinggi.
3. Nilai persentase *levensthein* yang dihasilkan sangat baik dan sangat sesuai. Meskipun dengan menggunakan tipe pemrosesan yang berbeda hasil yang diberikan cukup stabil.
4. Pengecekan dokumen yang tidak dirandom atau menguji dokumen yang memiliki kesamaan makna maupun isi seperti dokumen ”asli2” dan ”plagiat2” yang memiliki makna yang sama sehingga akan menghasilkan nilai presentase sinonim yang akurat.

5.2 Saran

Adapun saran dari hasil pengujian dan pengembangan dari sistem adalah sebagai berikut:

1. Perlu dioptimasikan proses untuk mendeteksi kata dari segi makna kata. Pada penelitian saat ini pengecekan kata dari segi kata masih menggunakan teknik manual atau *brute force* sehingga membutuhkan waktu yang lama dalam memproses sebuah file atau dokumen dengan jumlah kalimat yang banyak. Perlu adanya algoritma yang dapat mempersingkat proses pencarian makna kata tersebut.

2. Sistem ini masih memiliki kekurangan dalam proses stemming atau penghilangan kata imbuhan (afiks). Yang masih memiliki kesalahan dalam proses penghilang kata imbuhan (afiks) seperti kata “mengerti” menjadi “erti”. Perlu adanya peningkatan dalam hal pengenalan kata yang memiliki imbuhan.
3. Sistem ini juga tidak mengenali sebuah kalimat atau kata yang memiliki makna denotasi, konotasi maupun memiliki makna kiasan. Mungkin diperlukan penambahan beberapa database yang menampung kata-kata yang memiliki makna selain sinonim.
4. Sistem memiliki kelemahan dalam hal menangani kata yang memiliki perulangan seperti “janji-janji”, pada kata tersebut pada di lakukan proses *preprocessing* garis penghubung kata hilang tapi kata menyatu menjadi 1 seperti “janjijanji”. Maka perlu peningkatan dalam menangani kata yang memiliki perulangan agar tidak memunculkan kata yang bias atau kata yang tidak memiliki bahkan bukan termasuk dalam Kamus Besar Bahasa Indonesia (KBBI).
5. Proses pemberian bobot pada proses pengecekan sinonim perlu dioptimalisasi agar dapat menghasilkan nilai persentase yang sesuai.

DAFTAR PUSTAKA

Adiwidya, B.M.D., 2009, Algortima Levenshtein Dalam Pendekatan Approximate String Matching. Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung.

Adriyani, Ni Made Murni, et al., 2012, Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia. Universitas Udayana.

Ariyani, Na'firul Hasna, et al. 2016, "Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levensthein Distance" . Fakultas Teknik, Universitas Halu Oleo, Kendari

Aggarwal, Charu C. dan Zhai, Chen Xiang, 2012, ASurvey of Text Classification Algorithms (Online), Chapter 6, 52 halaman http://link.springer.com/chapter/10.1007%2F978-1-4614-3223-4_6,

Chaer, Abdul, 2007, Linguistik Umum, Jakarta: Rineka Cipta.

Dr. S. Vijayarani ed al., 2011, Preprocessing Techniques for Text Mining - An Overview, International Journal of Computer Science & Communication Networks, Volume 5, halaman 7-16.

Driscoll Lyn, Dana dan Brizee, Allen., 2013, Quaoting, Paraphrasing, and Summarizing. Tersedia pada : https://owl.english.purdue.edu/owl/resource/563/1/, tanggal akses : 18 Mei 2016.

Dola, Abdullah., 2010, Tataran Sintaksis dalam Gramatikal Bahasa Indonesia, Makassar: Badan Penerbit Universitas Negeri Makassar.

Haddi, Emma, ed al., 2013, The Role of Text Pre-processing in Sentiment Analysis.

Gilleland, M dan Merriam Park Software, 2006, Levenshtein Distance, In Three Flavours,http://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Fall2006/Assignments/editdistance/Levenshtein%20Distance.htm.

Gurusamy, Vairaprakash, 2014, Preprocessing Techniques for Text Mining.

Galiotou, Eleni ed al., 2013, On the Effect of Stemming Algorithms on Extractive Summarization: A Case Study.

Jivani , Anjali Ganesh Ms., 2011, A Comparative Study of Stemming Algortihms.

Juri D. Apresjan1, ed al., 2009, Semantic Paraprasing for Information Retrieval and Extraction, tersedia pada : http://link.springer.com/chapter/10.1007/978-3-642-04957-6_44.

Lestarini, Ade Hapsari, 2014, Sederet Kasus Plagiarisme di Kampus, http://news.okezone.com/read/2014/02/25/373/946214/sederet-kasus-plagiarisme-di-kampus, Diakses 28 Maret 2016.

Manning, Christopher D. ed al., 2009, An Introduction to Information Retrieval.

Marsa, A.H. 2009. Ayo Mengenal Paragraf. Jakarta: PT Wangsa Jatra Lestari.

Mayantara, 2012, Menerjemahkan Karya Sastra, http://mayantara.sch.id/artikel/menerjemahkan-karya-sastra.htm#comments, Diakses 18 Mei 2016.

McCabe, Donald L, 2005, Cheating among college and university students:A North American perspective, International Journal of Academic Integrity.

Nasucha, Rohmadi dan Wahyudi, 2009, Bahasa Indonesia Untuk Karya Tulis Ilmiah, Yogyakarta: Media Perkasa.

paice, c. d., 2009. stemming. Dalam : l. liu dan M. T. Özsu, penyunt. *encyclopedia of database systems.* s.l.:springer, halaman 2790-2793.

Pyriana, Dewi Rokhmah, et al., 2013, Program Aplikasi Editor Kata Bahasa Indonesia menggunakan Metode Approximate String Matching dengan Algoritma Levenshtein Distance Berbasis Java, Program Studi Teknik Informatika Universitas Brawijaya.

Rohmadi, Muhammad dan Nasucha, Yakub, 2010, Paragraf Pengembangan dan Implementasi, Yogyakarta: Media Perkasa.

Sastroasmoro, Sudigno, 2007, Beberapa Catatan Tentang Plagiarisme, Departemen Ilmu Kesehatan Anak, Fakultas Kedokteran Universitas Indonesia, Jakarta.

Stein, B., Meyer, S. zu Eissen., 2006, Near Similarity Search and PlagiarismAnalysis, 29th Annual Conference of the German Classification Society (GfKl), Magdeburg, halaman 430 – 437.

George R.S Weir, 2004, Work In progress Technology in plagiarism detection and management, *34th* ASEE/IEEE Frontiers in Education Conference.