**LAPORAN TUGAS BESAR APLIKASI PEMERIKSA PLAGIARISME**

Untuk memenuhi tugas Mata Kuliah Struktur Data dan Algoritma

****

Disusun oleh :

Ahmad Aji Naufal Ali 171524002

Melina Pratiwi Nurliana 171524017

Regawa Rama Prayoga 171524026

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**BANDUNG**

**2018**

# DAFTAR ISI

**DAFTAR ISI** 1

[**BAB I PENDAHULUAN** 2](#_Toc516042489)

[1.1. Latar Belakang 2](#_Toc516042490)

[1.2. Tujuan 3](#_Toc516042491)

1.3. Manfaat 3

1.4. Batasan Masalah 3

[**BAB II DESKRIPSI PROGRAM** 4](#_Toc516042496)

[2.1 Deskripsi Program 4](#_Toc516042497)

[2.2 Definisi Similarity 4](#_Toc516042497)

[2.3 Alur Proses Penggunaan Aplikasi 5](#_Toc516042497)

[2.4 Keluaran 5](#_Toc516042497)

**BAB III DESAIN APLIKASI**  6

3.1 Struktur Data 6

3.2 Desain Proses 6

[**BAB IV TEKNIS KEGIATAN** 6](#_Toc516042498)

[3.1. Pembagian Tugas 6](#_Toc516042499)

[3.2. Jadwal Kegiatan 7](#_Toc516042500)

**BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Seiring dengan kemajuan teknologi sekarang ini, praktik penjiplakan atau praktik plagiarisme sudah sangat mudah ditemukan dalam kegiatan sehari - hari. Hal tersebut dapat dilihat dari beberapa karya tulis, artikel -artikel, dan bahkan skripsi dengan isi yang hampir sama atau bahkan sama persis namun ditulis oleh orang yang berbeda. Hal ini sudah menjadi kebiasaan yang tidak baik dalam kehidupan manusia khusus nya di dalam dunia pendidikan.

Plagiarisme memiliki definisi penjiplakan yang melanggar hak cipta (Pusat Bahasa Departemen Pendidikan nasional, 2002). Sementara itu hak cipta memiliki definisi yaitu hak eksklusif bagi Pencipta atau penerima hak untuk mengumumkan atau memperbanyak ciptaannya atau memberikan izin untuk itu dengan tidak mengurangi pembatasan-pembatasan menurut peraturan perundang-undangan yang berlaku. Ciptaan adalah hasil setiap karya Pencipta yang menunjukkan keasliannya dalam lapangan ilmu pengetahuan, seni, atau sastra (Undang-Undang Republik Indonesia Nomor 19 Tahun 2002).

Dari permasalahan plagiarisme tersebut, diperlukan metode yang tepat untuk mencegah tindakan plagiarisme sejak dini. Maka dibuatlah aplikasi pendeteksi plagiarisme untuk mendeteksi persentase similaritas atau kesamaan antara dua buah teks atau lebih. Tingginya tingkat similaritas antara satu tulisan dengan tulisan lainnya yang akan disajikan oleh aplikasi tersebut dapat dijadikan sebagai indikator terjadinya tindakan plagiarisme.

1. **Tujuan**

Aplikasi ini dibuat dengan tujuan sebagai berikut:

1. Memeriksa presentase plagiarisme dari sebuah teks berbahasa inggris

**1.3 Manfaat**

Adapun manfaat dari aplikasi ini adalah sebagai berikut:

1. Dengan diketahuinya presentase plagiarisme dari sebuah teks maka plagiarisme dapat dikurangi dengan meminimalisir kata yang sama

**1.4 Batasan Masalah**

1. Aplikasi ini hanya mampu bekerja pada teks berbahasa Inggris

**BAB II**

**DESKRIPSI APLIKASI**

* 1. **Deskripsi Aplikasi**

Aplikasi yang akan kami buat adalah aplikasi yang dibuat dengan bahasa C/C++ yang dapat mengeluarkan output persentase plagiarisme dari *input* berupa dokumen-dokumen teks berbahasa inggris. Adapun beberapa fitur yang terdapat pada aplikasi ini adalah sebagai berikut:

1. Pengguna dapat menginput jumlah *file* teks yang akan dibandingkan, setelah itu pengguna memilih *file* teks yang akan dibandingkan dengan cara menginputkan lokasi dari dari tiap tiap *file* teks (contoh: “file1.txt”).
2. Aplikasi ini dapat membandingkan beberapa *file* dan mencari presentase plagiarisme dari tiap *file* ke *file* lainnya. Jika ada 2 file maka akan ada 1 kali perbandingan, jika ada 3 file maka akan ada 3 kali perbandingan. Jadi jumlah file yang dibandingkan dapat dihitung dengan rumus berikut : 1/2 x n(n-1)
3. Aplikasi akan menyimpan statistik kata dan presentase plagiarisme dari *file-file* teks yang diinput ke dalam sebuah *file* teks yang diberi nama “output.txt”.
   1. **Definisi Similarity**

Pada program yang akan kami buat, “similarity” dihitung dari banyaknya kata yang sama pada dua buah dokumen teks bahasa inggris. Adapun tingkatan plagiarisme yang ditentukan adalah sebagai berikut:

1. 0-30% kesamaan kata merupakan plagiarisme rendah.
2. 31-70% kesamaan kata merupakan plagiarisme menengah.
3. 71-100% kesamaan kata merupakan plagiarisme tinggi.

Perhitungan presentase plagiarisme dihitung menggunakan rumus sebagai berikut:

Jumlah frekuensi terkecil dari tiap kata yang sama x Jumlah file yg dibandingkan

X 100%

Jumlah frekuensi kata pada setiap file

* 1. **Teori Stemming**
  2. **Alur Proses Penggunaan Aplikasi**

Adapun alur proses dari aplikasi ini adalah sebagai berikut:

1. Pada saat aplikasi dijalankan akan ditampilkan menu pilihan bagi pengguna, terdapat 2 pilihan yaitu “Cek Plagiarisme” dan “Keluar”.
2. Pilihan “cek plagiarisme” akan menampilkan instruksi “Input jumlah file yang ingin diperiksa :” pada layar konsol. User harus menginput sebuah bilangan yang menyatakan jumlah file yang ingin di cek presentasi plagiarismenya. (Bilangan harus bernilai lebih dari atau sama dengan 2)
3. Selanjutnya aplikasi akan menampilkan instruksi “Input namafile ke-n :” sebanyak n kali. Pengguna harus menginputnama file sesuai dengan jumlah file yang telah ditentukan di awal. Nama file yang diinputkan harus nama file yang berada pada direktori/*folder* dimana aplikasi dijalankan.

Contoh: Jika pengguna menginput nilai 2 pada instruksi pertama maka akan tampil instruksi sebagai berikut.

Input nama file ke-1:

Input nama file ke-2:

1. Program akan memproses semua file yang diberikan. User dapat melihat tampilan loading sambil menunggu proses selesai.
2. Apabila telah selesai aplikasi akan menampilkan “Proses cek plagiarisme selesai”, pengguna dapat melihat file output “output.txt” yang berisikan statistik dan list persentase plagiarisme pada folder Output di dalam folder aplikasi.
3. Pengguna akan kembali ke tampilan awal aplikasi.
   1. **Keluaran**

Aplikasi yang sudah selesai, akan memberikan beberapa keluaran sebagai berikut :

1. Aplikasi dapat mengeluarkan file “output.txt” yang berisikan persentase plagiarisme tiap file beserta statistik kemunculan katanya

**BAB III**

**DESKRIPSI KEBUTUHAN DATA**

**3.1 Struktur Data**

Struktur data yang kami gunakan pada aplikasi ini sebagai berikut :

**typedef struct** NodeTree{ // Node Tree

**char** \*kata;

**struct** NodeTree \*kiri;

**struct** NodeTree \*kanan;

**struct** ListStatistik \*headStatistik;

**int** height**;**

}NodeTree;

Keterangan :

* Kata adalah sub-variabel bertipe pointer yang menampung kata dari sebuah file yang telah dibaca.
* NodeTree kiri adalah sub-variabel bertipe pointer yang menampung
* NodeTree kanan adalah sub-variabel bertipe pointer yang menampung
* ListStatistik adalah
* Height adalah sub-variabel bertipe integer

**typedef struct** TreeStatistik{ // Head tree

**struct** NodeTree \*root;

}TreeStatistik;

Keterangan :

* TreeStatistik adalah
* NodeTree \*root adalah sub-variabel yang menampung

**typedef struct** NodeStatistik{ // Node List

**int** jumlah;

**struct** NodeStatistik \*next;

}NodeStatistik;

Keterangan :

* NodeStatistik adalah
* Jumlah adalah sub-variabel bertipe integer yang menampung
* NodeStatistik \*next adalah sub-variabel bertipe pointer yang menampung

**typedef struct** ListStatistik{ // Head List

**struct** NodeStatistik \*head;

}ListStatistik;

Keterangan :

* ListStatistik adalah
* NodeStatistik \*head adalah sub-variabel bertipe pointer yang menampung

NAMBAH LAGI

**BAB IV**

**RANCANGAN PROSES**

|  |  |
| --- | --- |
| Nama Modul | substr |
| Deskripsi | Modul untuk mengambil substring pada suatu string |
| Jenis | Procedure |
| Parameter | Dest : string, source : string, pos : integer, lenght : interger |
| Initial State | String dari var dest kosong |
| Final State | String dari var dest berisikan sub string dari var source sebanyak posisi *pos* sampai *lenght* |
| Kamus Data | |
| C : integer | |
| Algoritma | |
| begin  c = 0;  while (c < length) {  dest[c] = source[pos+c];  c++;  }  end. | |

|  |  |
| --- | --- |
| Nama Modul | irregularToRegular |
| Deskripsi | Modul untuk mengubah string dest menjadi kata regular sesuai dengan node yang di passing |
| Jenis | Procedure |
| Parameter | dest : string, irr : NodeTree |
| Initial State | String dest berisikan kata yang bersifat irregular |
| Final State | String dest berubah isinya menjadi kata yang bersifat regular |
| Kamus Data | |
| Sub : string  I : integer | |
| Algoritma | |
| begin  int i = strlen(dest);  substr(sub, irr->kata, 0, i);  if(strcmp(dest, sub) == 0){  substr(dest, irr->kata, i+1, strlen(irr->kata));  }  end. | |

|  |  |
| --- | --- |
| Nama Modul | MakeTreeStatistik |
| Deskripsi | Modul untuk menyusun AVL tree yang berisikan statistik huruf |
| Jenis | Procedure |
| Parameter | tree : TreeStatistik, file : FILE, jumlahFile : integer, urutanFile : integer |
| Initial State | TreeStatistik tree kosong |
| Final State | TreeStatistik tree berisikan AVL Tree statistik yang berasal dari kata kata pada file inputnya |
| Kamus Data | |
| i : integer  karakter : char  kata : string | |
| Algoritma | |
| Begin  printf("KATA YANG TERBACA : \n\n");  // TOKENIZING DAN CASE FOLDING  while((karakter = getc(file)) != EOF){    if(karakter >= 'A' && karakter <= 'Z')  karakter +=32 ;    if(karakter >= 'a' && karakter <= 'z'){  kata[i] = karakter;  i++;  } else if(i > 0){  i = 0;  puts(kata); // print word  rootSearch =(NodeTree\*) search(stopwordTree, kata);  if(rootSearch != NULL){  printf("[%s IS A STOPWORD]\n", kata);  printf("[%s WILL BE DELETED]\n\n", kata);  }else{  printf("[%s NOT A STOPWORD]\n", kata);  printf("[PROGRAM WILL CHECK IRREGULARITY OF %s WORD]\n", kata);  rootSearch =(NodeTree\*) searchIrr(irregularTree, kata);  if(rootSearch != NULL){  printf("[%s IS IRREGULAR FORM]\n", kata);  printf("[PROGRAM WILL CHANGE %s TO REGULAR FORM]\n", kata);  printf("[REGULAR FORM OF %s ", kata);  irregularToRegular(kata, rootSearch);  printf("is %s]\n", kata);  }else{  printf("[%s NOT IRREGULAR FORM]\n", kata);  printf("[%s WILL PROCEED TO STEMMING]\n", kata);  printf("[PROGRAM WILL REMOVE PREFIX AND SUFFIX FROM %s]\n", kata);  // Stem  printf("[ROOT WORD FROM %s ", kata);  kata[stem(kata, 0, strlen(kata) - 1) + 1] = '\0';  printf("IS %s]\n", kata);  }  rootStatistik=(NodeTree\*)insert(rootStatistik,kata,jumlahFile,urutanFile);  printf("[%s IS INSERTED TO THE STATISTIC]\n\n", kata);  }  memset(kata, 0, sizeof(kata)); // set array word jadi null / kosong  }  }  if(i>0){  i = 0;  puts(kata); // print word  rootSearch =(NodeTree\*) search(stopwordTree, kata);  if(rootSearch != NULL){  printf("[%s IS A STOPWORD]\n", kata);  printf("[%s WILL BE DELETED]\n\n", kata);  }else{  printf("[%s NOT A STOPWORD]\n", kata);  printf("[PROGRAM WILL CHECK IRREGULARITY OF %s WORD]\n", kata);  rootSearch =(NodeTree\*) search(irregularTree, kata);  if(rootSearch != NULL){    printf("[%s IS IRREGULAR FORM]\n", kata);  printf("[PROGRAM WILL CHANGE %s TO REGULAR FORM]\n", kata);  printf("[REGULAR FORM OF %s ", kata);  irregularToRegular(kata, rootSearch);  printf("is %s]\n", kata);  }else{  printf("[%s NOT IRREGULAR FORM]\n", kata);  printf("[%s WILL PROCEED TO STEMMING]\n", kata);  printf("[PROGRAM WILL REMOVE PREFIX AND SUFFIX FROM %s]\n", kata);  // Stem  printf("[ROOT WORD FROM %s ", kata);  kata[stem(kata, 0, strlen(kata) - 1) + 1] = '\0';  printf("IS %s]\n", kata);  }  rootStatistik=(NodeTree\*)insert(rootStatistik,kata,jumlahFile,urutanFile);  printf("[%s IS INSERTED TO THE STATISTIC]\n\n", kata);  }    memset(kata, 0, sizeof(kata)); // set array word jadi null / kosong  }  end.  saa  end. | |

|  |  |
| --- | --- |
| Nama Modul | createStopwordTree |
| Deskripsi | Modul untuk membuat AVL Tree yang berisikan stopword |
| Jenis | Procedure |
| Parameter | - |
| Initial State | Node StopwordTree kosong |
| Final State | Node Stopword Tree berisikan AVL Tree stopword yang berasal dari infile |
| Kamus Data | |
| infile : FILE, stopword : string | |
| Algoritma | |
| Begin  infile = fopen("stopword-en.txt","r"); // Read file  printf("|| %20s || \n", "DAFTAR STOPWORD");  printf("|| %20s || \n", "stopword-en.txt");  printf("|| %20s || \n", "");  while (fscanf(infile,"%s", stopword)==1){  printf("|| %20s || \n", stopword);  stopwordTree =(NodeTree\*) insert(stopwordTree, stopword,1,1);  }  end. | |

|  |  |
| --- | --- |
| Nama Modul | createIrregularVerbTree |
| Deskripsi | Membuat AVL Tree yang berisikan *irregular* verb dan kata *regularnya* |
| Jenis | ***Procedure*** |
| Parameter | - |
| Initial State | Node irregularTree kosong |
| Final State | Node irregularTree berisikan AVL Tree irregular verb dan perubahannya yang berasal dari infile |
| Kamus Data | |
| Infile : FILE  Irregular : string | |
| Algoritma | |
| Begin  infile = fopen("irregular-en.in","r"); // Read file  printf("|| %20s || \n", "IRREGULAR VERB LIST");  printf("|| %20s || \n", "irregular-en.in");  printf("|| %20s || \n", "");  while (fscanf(infile,"%s", irregular)==1){  printf("|| %20s || \n", irregular);  irregularTree =(NodeTree\*) insert(irregularTree, irregular, 1, 1);  }  end. | |

|  |  |
| --- | --- |
| Nama Modul | showInputDataMenu |
| Deskripsi | Modul untuk menampilkan menu input data pada *console* |
| Jenis | Procedure |
| Parameter | - |
| Initial State | Tidak ada tampilan pada *console* |
| Final State | Muncul tampilan menu input data pada *console* |
| Kamus Data | |
| - | |
| Algoritma | |
| Begin  printf("||%40s||\n", "PILIH PROSES INPUT DATA : ");  printf("||%40s||\n", "");  printf("||%40s||\n", "1. OTOMATIS VIA FILE");  printf("||%40s||\n", "2. MANUAL VIA CONSOLE");  printf("||%40s||\n", "3. PETUNJUK OTOMATIS VIA FILE");  end. | |

|  |  |
| --- | --- |
| Nama Modul | showMainMenu |
| Deskripsi | Modul untuk menampilkan main menu pada *console* |
| Jenis | Procedure |
| Parameter | - |
| Initial State | Tidak ada tampilan pada *console* |
| Final State | Muncul tampilan main menu pada console |
| Kamus Data | |
| - | |
| Algoritma | |
| Begin  printf("||%40s||\n", "MAIN MENU : ");  printf("||%40s||\n", "");  printf("||%40s||\n", "1. DETEKSI PLAGIARISME");  printf("||%40s||\n", "2. KELUAR");  end. | |

|  |  |
| --- | --- |
| Nama Modul | showCredit |
| Deskripsi | Modul untuk menampilkan credit pada *console* |
| Jenis | Procedure |
| Parameter | - |
| Initial State | Tidak ada tampilan pada *console* |
| Final State | Muncul tampilan credit pada console |
| Kamus Data | |
| - | |
| Algoritma | |
| Begin  printf("||%40s||\n", "PENDETEKSI PLAGIARISME");  printf("||%40s||\n", "MADE WITH <3 BY : ");  printf("||%40s||\n", "Ahmad Aji Naufal Ali (171524002)");  printf("||%40s||\n", "Melina Pratiwi (171524017)");  printf("||%40s||\n", "Regawa Rama Prayoga (17154026)");  printf("||%40s||\n", "");  printf("||%40s||\n", "");  printf("||%40s||\n", "");  end. | |

|  |  |
| --- | --- |
| Nama Modul | Choose |
| Deskripsi | Modul untuk membaca input dan mengeluarkan input yang terbaca |
| Jenis | Function |
| Parameter | - |
| Initial State | Var ch kosong |
| Final State | Mengeluaran nilai dari integer ch yang terbaca |
| Kamus Data | |
| ch : integer | |
| Algoritma | |
| Begin  printf("PILIHAN ANDA : ");  scanf("%d", &ch);  printf("\n");  return ch;  end. | |

**BAB V**

**TAMPILAN APLIKASI**

**BAB VI**

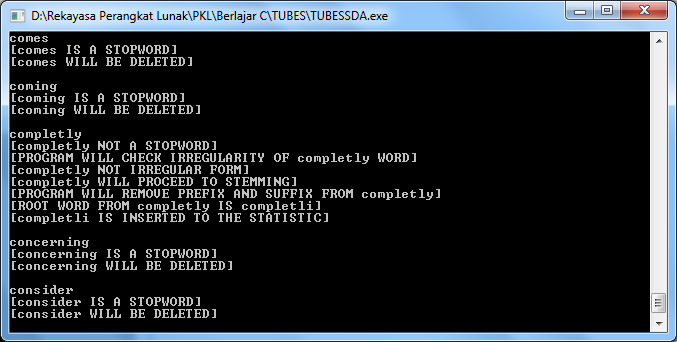
**IMPLEMENTASI DAN UJI COBA**

1. **Uji Coba *Stopword* *Removal***

Untuk menguji kemampuan aplikasi dalam menghilangkan *stopword* berbahasa inggris maka dibuatlah *file* teks yang berisi *stopword* yang akan diinput pada aplikasi cek plagiarism. Isi dari *file* teks tersebut adalah sebagai berikut:

|  |
| --- |
| Uji-stopword.txt |
| A about above according across actual added after against ahead all almost alone along also among amongst an and and-or and/or anon another any are arising around as at award away be because become becomes been before behind being below best better between beyond birthday both but by can certain come comes coming completly concerning consider. |

Hasil pengujian adalah sebagai berikut:



Gambar 1. Proses stopword removal

Kata yang termasuk ke dalam kategori *stopword* akan terdeteksi dan dihapus, sedangkan kata yang tidak termasuk ke dalam kategori *stopword* akan dilanjutkan dengan proses selanjutnya yaitu *Irregural Verb Conversion.*

1. **Uji Coba *Irregular Verb Conversion***

Proses *irregular verb conversion* merupakan sebuah proses untuk mengubah kata *irregular* menjadi kata dasarnya (*ex*: *drank* menjadi *drink*). Untuk menguji kemampuan aplikasi dalam mengkonversi kata *irregular* maka dibuatlah *file* teks uji coba yang berisikan kata *irregular* dan kata yang bukan *irregular*, *file* akan diujicobakan terhadap aplikasi, berikut isi dari teks tersebut:

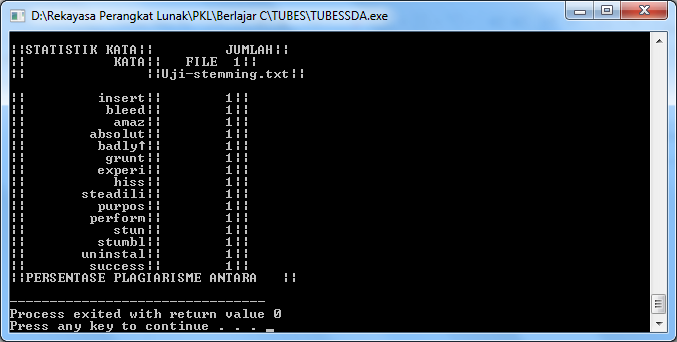
|  |
| --- |
| Uji-irregular.txt |
| Eat ate I want to drink water. Was here before, bought, bad, worse, bit, bled, flew, this is a pencil. You made her cry |

1. **Uji Coba *Stemming***

Proses *stemming* merupakan proses untuk mengubah kata menjadi kata dasarnya dengan menghilangkan prefix dan suffix dari sebuah kata. Untuk menguji ketepatan *stemming* pada aplikasi maka dibuat *file* teks yang berisi kata ber-prefix dan ber-suffix sebagai berikut:

|  |
| --- |
| Uji-stemming.txt |
| Badly, steadily uninstalling inserting purposely absolutely amazing performance stunning hissing stumbled bleeding grunting experiences successful. |

Hasil uji stemming terhadap aplikasi



Gambar 2 Hasil uji stemming

Dari statistik kata yang ditampilkan dapat disimpulkan bahwa hasil stemming dari aplikasi masih kurang baik, hal ini ditandai dengan tidak sesuainya kata hasil *stemming* dengan kata dasarnya (*ex:* absolutely menjadi absolut yang seharusnya adalah absolute).

1. **Uji Coba Perhitungan *Text Similarity***

Untuk menguji ketepatan aplikasi dalam menentukan *text similarity* dari *file-file* teks yang diinput dibuatlah beberapa *file* teks yang berisi data dummy untuk diuji cobakan, berikut isi dara beberapa *file* teks yang akan diujikan pada aplikasi:

|  |
| --- |
| Uji-sim1.txt |
| Video provides powerful way help you prove your point. |

|  |
| --- |
| Uji-sim2.txt |
| Video provides powerful way help you prove your point. |

|  |
| --- |
| Uji-sim3.txt |
|  |

|  |
| --- |
| Uji-sim4.txt |
|  |

|  |
| --- |
| Uji-sim5.txt |
| Video provides |

|  |
| --- |
| Uji-sim6.txt |
| Video provides Video provides |

**BAB VII**

**PENUTUP**

**7.1 Alur Kerja dan Pembagian Tugas**

**7.2 Lesson Learned**