DETEKSI PLAGIARISME PADADOKUMEN SKRIPSI BERDASARKAN TINGKAT KESAMAAN DENGAN MENGGUNAKAN METODE LONGEST COMMON SUBSEQUENCE

Putra Prima Arhandi [1], Faisal Rahutomo[2], ImamNawawi[3]

[1,2,3] Teknologi Informasi, Politeknik Negeri Malang
[3] nawawi932@gmail.com,

Abstrak— **Plagiarisme adalah tindakan menyalin, mengambil karangan atau pendapat orang lain tanpa adanya izin tertulis dan menjadikanya seolah-olah pendapatnya sendiri. Hal ini masih menjadi fenomena yang sering terjadi pada instansi akademik atau non akademik. Namun pada jurusan Teknologi Informasi Politeknik Negeri Malang belum ada aplikasi yang dapat digunakan untuk mendeteksi plagiarisme.**

Berdasarkan permasalahan diatas, maka dibuatlah aplikasi deteksi plagiarisme pada dokumen Tugas Akhir / Skripsi yang bernama *Document plagiarism Detection* (Doristec) dengan menggunakan metode *Longest Common Subsequence (LCS)* dengan membuat modifikasi untuk mencapai hasil yang sesuai dengan perancangan. Aplikasi ini dibuat dengan tujuan untuk mengetahui seberapa besar tingkat plagiarisme pada dokumen Tugas Akhir / Skripsi yang nantinya akan diketahui oleh mahasiswa dan panitia Laporan Akhir dan Skripsi.

Berdasarkan penelitian yang telah dilakukan, Metode *Longest Common Subsequence* dapat digunakan untuk deteksi plagiarisme dengan perbandingan dua atau lebih dokumen. Hal ini dapat menjadi alternatif bagi mahasiswa jurusan Teknologi Informasi Politeknik Negeri Malang dalam melakukan pengujian terhadap penelitiannya dan bagi panitia Laporan Akhir dan Skripsi dapat memonitoring Laporan Akhir dan Skripsi..

Kata kunci—komponen; Kemiripan kata, Similarity, Plagiarisme, Longest Common Subsequence, LCS

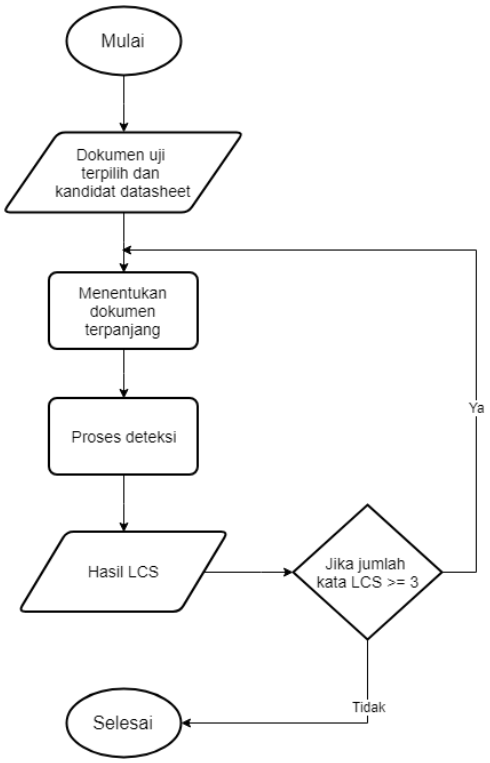
I. PENDAHULUAN

Di era kemajuan teknologi yang semakin pesat saat ini, tingkat publikasi dan jumlah plagiarisme yang terdeteksi semakin meningkat, situasi seperti ini menghambat perkembangan dari aktivitas kreatif mahasiswa maka diperlukan metode yang akurat untuk mendeteksi plagiarisme. Pada Jurusan Teknologi Informasi (JTI) masih belum ada aplikasi yang dapat digunakan oleh mahasiswa JTI untuk mendeteksi adanya plagiarisme pada dokumen skripsi.

Aini, Bariri Isriya Nur (2014) dengan judul "Identifikasi Kemiripan Judul Tugas Akhir PSTI Dan PSMI Di Politeknik Negeri Malang". Dalam penelitianya mengenai judul tugas akhir atau sekripsi yang harus berbeda dari judul – judul yang lain. Algoritma yang digunakan pada penelitian adalah algoritma Nazief & Adriani, algoritma VSM (Vector Space Model) dan algoritma tfidf (Term Frequency – Inversed Document Frequency), dengan algoritma tersebut dihasilkan sebuah aplikasi yang dapat mengidentifikasi dan memberikan informasi kemiripan pada judul tugas akhir atau skripsi.

Widiastuti, Inta (2014) yang berjudul "Aplikasi Pendeteksi Kemiripan Dokumen Menggunakan Algoritma Rabin Karp". Dalam penelitiannya mengenai deteksi kemiripan antara dua dokumen dengan dilakukan perbandingan pattern pada dokumen teks, metode yang digunakan untuk mendeteksi kemiripan dokumen adalah Algoritma Rabin Karp. Penelitian ini berhasil menciptakan aplikasi deteksi plagiarisme dokumen yang bekerja dengan baik serta memperlihatkan hasil similarity-nya.

Dikarenakan pada JTI belum ada aplikasi deteksi plagiarisme dokumen skripsi, oleh karena itu dalam skripsi ini penulis akan membuat sistem deteksi plagiarisme dokumen skripsi menggunakan metode longest common subsequence (LCS). Proses deteksi menggunakan perbandingan pada dokumen yang di upload mahasiswa dengan datasheet yang ada pada database. Proses dari metode ini mengubah teks menjadi angka, dari angka tersebut dapat diketahui deretan angka yang sama dan diambil yang paling panjang dari hasil LCS, kemudian panjang LCS dari dokumen yang di upload dibagi dengan panjang LCS dari datasheet seperti yang ditunjukkan pada gambar 1. Berdasarkan perhitungan menggunakan metode LCS adalah persentase similarity. Dengan adanya aplikasi ini, diharapkan dapat membantu untuk mendeteksi adanya plagiarisme dokumen tugas akhir dan skripsi.



Gambar 1. Flowchart Proses Lcs

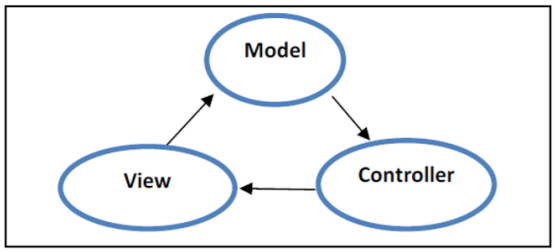
Masalah yang telah dirumuskan yaitu bagaimana menciptakan aplikasi yang dapat mendeteksi adanya plagiarisme pada dokumen skripsi dan bagaimana mendeteksi plagiarisme pada dokumen skripsi menggunakan metode longest common subsequence yang didasarkan bahwa tujuan dari penelitian ini adalah membuat sistem yang dapat mendeteksi plagiarisme pada dokumen skripsi berdasarkan tingkat kesamaan dengan menggunakan metode longest common subsequence dan memberikan kemudahan kepada panitia skripsi dalam menentukan penelitian mahasiswa tidak mengandung plagiarisme terhadap penelitian lainnya.

II. LANDASAN TEORI

Pada bab ini akan diuraikan kajian pustaka dan dasar teori yang mendukung penelitian ini. Dasar teori tersebut diperoleh dari beberapa referensi yang relevan dengan topik yang diangkat dalam penelitian ini. Dalam bab ijni akan dijelaskan mengenai kajian pustaka dan metode *Longest Common Subsequence* (LCS) yang merupakan topik utama diadakannya penelitian ini.

A. Model View Controller (MVC)

Menurut [17], *Model View Controller* (MVC) merupakan *framework* yang menerapkan model MVC, model ini diterapkan guna dapat memberi dampak kedisiplinan dalam menuliskan kode program serta mempermudah dalam pengembangan aplikasi karena telah terstruktur dengan baik, model MVC dibagi menjadi 3 bagian yaitu : model, *view*, dan *controller* seperti pada gambar 2.1.



Gambar 2. Model View Controller

Bagian-bagian tersebut memiliki arti sebagai berikut:

- Model merupakan bagian yang berinteraksi dengan database.
- View merupakan pages yang tampil di halaman web.
- Controller akan mengatur interaksi antar *pages*/ mengkontrol hubungan *view* dan *model*.

Keuntungan MVC adalah sebagai berikut:

- *Division/testability*

  Cotrollers mengatur interaksi antara *User Interface* dan data (model).
- *Flexibility*

  Setiap individual layer mudah dibuang tanpa mempengaruhi layer lainnya karena dapat di *costumize* sehingga *user interface* dapat diganti atau menggunakan *database* yang berbeda.
- *Maintability*

Walaupun dapat di *costumize*, tapi *projects* harus tetap memiliki code yang teratur.

B. Metode Longest Common Subsequence

Longest Common Subsequences merupakan masalah dalam mencari *subsequence* terpanjang dari beberapa *sequence* (biasanya hanya 2 buah *sequence*), dimana sebuah *subsequence* merupakan sekumpulan kata dari *sequence* yang urutan kemunculannya sama [7]. *Subsequence* dari sebuah *string* A adalah sekumpulan kata yang ada pada A dengan urutan kemunculan kata yang sama.

Dalam rangkaian Z yang merupakan *subsequence* dari X (sebagai *sequence*) dimana X = $<x_1,x_2,\ldots,x_m>$, jika terdapat urutan menaik $<i_1,i_2,\ldots,i_n>$ yang merupakan indeks X untuk semua j=1,2,…,k, yang memenuhi $x_i=z_j$. misal pada *common subsequences* dari 2 rangkaian string1 = tadi saya sedang makan dikantin sekolah pada jam istirahat dan string2 = saya sedang mengerjakan tugas seni budaya pada jam istirahat, maka saya sedang, dan pada jam istirahat adalah *common subsequences* dari string1 dan string2. *Longest Common Subsequences* adalah *common subsequences* dari rangkaian hasil yang paling panjang pada 2 rangkaian *string* (string1 dan string2), sehingga *Longest Common Subsequences* yang terpilih adalah pada jam istirahat.

- Mendapatkan Panjang LCS

  Untuk menjabarkan dalam mempermudah dalam menjelaskan, akan dijabarkan sebagai berikut :

  Misal m adalah panjang string1, n adalah panjang string2, dan tab adalah matriks berukuran m x n, dan tab[i,

j] adalah panjang LCS dari subsequences pertama yang terdiri dari i kata pada string1 dengan subsequences kedua yang terdiri dari j kata pada string2, maka untuk $1 \leq i \leq m$ dan $1 \leq j \leq n$, sesuai dengan fungsi rekursif diatas, maka :

$$tab[i, j] = \begin{cases} tab[i-1, j-1] + 1, S1[i] = s2[j] \\ \max(tab[i-1, j], tab[i, j-1]) \end{cases}$$

Teks yang dibentuk yaitu string1 = tadi saya sedang makan dikantin sekolah pada jam istirahat dan string2 = saya sedang mengerjakan tugas seni budaya pada jam istirahat, tabel matrik yang terbentuk berukuran 9 x 7 yang isinya sebagai berikut :

Tabel 1. Hasil Kalkulasi LCS

		1	2	3	4	5	6	7	8	9
		tadi	saya	sedang	makan	dikantin	sekolah	pada	jam	istirahat
1	saya	0	1	1	1	1	1	1	1	1
2	sedang	0	1	2	2	2	2	2	2	2
3	mengerjakan	0	1	2	2	2	2	2	2	2
4	tugas	0	1	2	2	2	2	2	2	2
5	pada	0	1	2	2	2	2	3	3	3
6	jam	0	1	2	2	2	2	3	4	4
7	istirahat	0	1	2	2	2	2	3	4	5

Dan sesuai dengan arti notasi dari setiap tab[*i, j*] yang berisi pada tabel 2.1 bahwa LCS dari string1 dan string2 adalah tab[*m, n*] atau dalam kasus ini adalah ditemukan 5 kata.

- Mendapatkan LCS

Sedangkan untuk mendapatkan LCS yang dimaksud, kita bisa merunut balik pada tabel 2.1 yang dimulai dari tab[m, n]. Dalam runut balik ini, yang kita lakukan sebenarnya hanyalah merunut balik pilihan yang dilakukan pada saat kalkulasi tab[i, j].

Untuk setiap $1 \leq i \leq m$ dan $1 \leq j \leq n$, jika string1[i] sama dengan string2[j], maka kata itu pasti terdapat pada LCS. Selain itu, cek, dari mana mendapatkan LCS saat itu, lakukan pemilihan yang sama. Lakukan hal tersebut dimulai dari i = m dan j = n. Setelah hal itu dilakukan maka akan terbentuk tabel di bawah ini. Di mana cell yang dihitamkan menandakan kata yang masuk ke dalam LCS.

Tabel 2. Hasil Penurutan Balik Untuk Mendapatkan LCS

		1	2	3	4	5	6	7	8	9
		tadi	saya	sedang	makan	dikantin	sekolah	pada	jam	istirahat
1	Saya	0	1	1	1	1	1	1	1	1
2	Sedang	0	1	2	2	2	2	2	2	2
3	Mengerjakan	0	1	2	2	2	2	2	2	2
4	Tugas	0	1	2	2	2	2	2	2	2
5	Pada	0	1	2	2	2	2	3	3	3
6	Jam	0	1	2	2	2	2	3	4	4
7	Istirahat	0	1	2	2	2	2	3	4	5

Berdasarkan pada tabel 2 dapat dilihat bahwa hasil LCS yang ditemukan adalah saya sedang, pada jam istirahat.

C. Codeigniter

CodeIgniter adalah *framework* open source yang powerful yang dibangun menggunakan bahasa pemrograman PHP yang terkenal sebagai PHP *framework* yang mudah dikuasai [15], codeigneter dibangun untuk PHP programmers yang membutuhkan toolkit sederhana dan untuk membuat full-featured web applications. CodeIgniter menggunakan konsep MVC *framework* yang di *design* untuk mempermudah pengguna dan ketika dalam pengembangan aplikasi. Pada penelitian ini codeigniter digunakan untuk membangun aplikasi yang berjudul deteksi plagiarisme pada dokumen skripsi berdasarkan tingkat kesamaan dengan menggunakan metode *Longest Common Subsequence.*

D. Bootstrap

Bootsrap adalah *framework* css yang dapat digunakan untuk membangun tampilan *web* [16]. *Framework* ini menggunakan coding CSS dan JavaScript, namun tetap bisa membuat *website* yang powerfull mengikuti perkembangan *browser*. *Website* yang menggunakan bootstrap akan menjadi *website* yang fleksibel dengan penataan *layout* standard dari bootstrap yang dinamakan grid, sehingga nyaman dan tentu saja cepat baik dalam proses pembuatan maupun ketika *maintenance.* Tidak hanya penataan layoutnya saja yang mudah, bootstrap juga menyediakan komponen-komponen yang siap digunakan untuk mendesain sebuah template.
Komponen yang disediakan sangat banyak dan memiliki beberapa variasi, seperti *dropdowns, button, input, navbar, label, panels, alerts* dan masih banyak lagi yang dapat di *explore* melalui *website* resminya https://getbootstrap.com. Pada penelitian ini desain templatenya menggunakan bootstrap versi 3.

E. MySQL

MySQL adalah *database product* yang bersifat *open source* [19]. MySQL digunakan secara gratis dan hanya menghabiskan sedikit biaya bagi periklanannya namun sangat populer. Tidak seperti *commercial database*, MySQL sangat terjangkau dan mudah digunakan.
Fitur yang terdapat di MySQL antara lain:

- *Openness*
  MySQL sangat terbuka dalam setiap hubungan. SQL dialect yang digunakan adalah ANSI SQL2 sebagai pembangunnya. Database engine menjalankan platform yang tidak terhitung, termasuk Windows 2000, Mac OS X, Linux, FreeBSD, dan Solaris. Jika binary tidak tersedia untuk platform, user dapat mengaksesnya dari source untuk di compile ke platform tersebut.

- *Application Support*
  MySQL memiliki API untuk semua bahasa pemrograman. *User* bisa menulis aplikasi *database* yang mengakses MySQL di C, C++, Eiffel, Java, Perl, PHP, Phyton, dan Tcl.
- *Cross-database joins*
  User bisa mengkonstruksi MySQL queries yang bisa menggabungkan tabel dari *database* yang berbeda.
- *Outer join support*

MySQL mendukung kiri dan kanan outer joins menggunakan ANSI dan sintaks ODBC.

F. *Kalimat*

Menurut [12, pp. 317-319] Kalimat adalah satuan bahasa terkecil, dalam wujud lisan atau tulisan, yang mengungkapkan pikiran yang utuh. Dalam wujud lisan, kalimat diucapkan dengan suara naik turun dan keras lembut, disela jeda dan diakhiri dengan intonasi akhir yang diikuti oleh kesenyapan yang mencegah terjadinya perpaduan ataupun asimilasi bunyi ataupun proses fonologis lainnya.

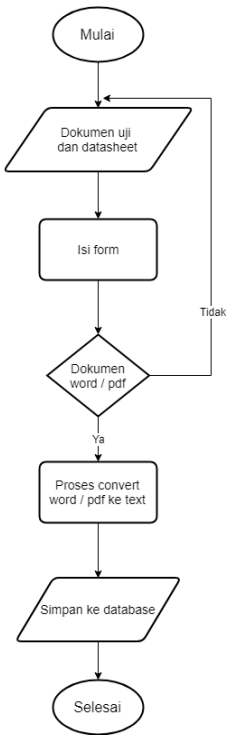
Dilihat dari segi bentuknya, kalimat dapat dirumuskan sebagai konstruksi sintaksis terbesar yang terdiri atas dua kata atau lebih. Hubungan structural antara kata dan kata, atau kelompok kata dan kelompok kata yang lain, berbeda-beda. Sementara itu, kedudukan tiap kata atau kelompok kata dalam kalimat itu berbeda-beda pula. Ada kata atau kelompok kata yang dapat dihilangkan dengan menghasilkan bentuk yang tetap berupa kalimat (1b), dan ada pula yang tidak seperti pada (2b).

1. a. Ibu pergi ke pasar.
   b. Ibu pergi.
2. a. Masalah itu menyangkut masa depan kita.
   b. Masalah itu menyangkut.

III. IMPLEMENTASI

A. *Upload Dokumen*

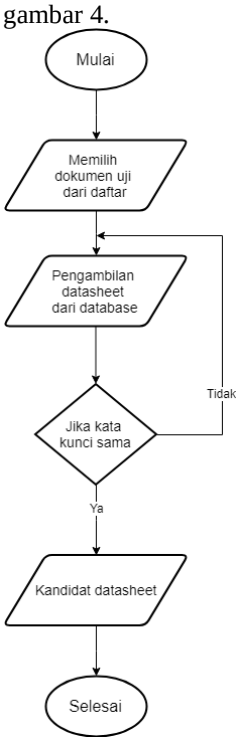
Upload dokumen merupakan proses untuk menyimpan kedalam *database*. Hal yang dilakukan sebelum disimpan terlebih dahulu dokumen word atau pdf dilakukan *convert* untuk mengambil keseluruhan *text* yang ada dalam dokumen, seperti yang ditunjukkan pada gambar 3.



Gambar 3. Upload Dokumen Skripsi Dan Tugas Akhir

B. *Filter Kandidat Pembanding*

Filter Kandidat Pembanding merupakan sebuah fungsi yang terdapat didalam sistem yang bertujuan untuk mencari data-data didalam *dataset*, pencarian dilakukan berdasarkan kata kunci yang sesuai dengan data uji. Sehingga data uji nantinya dilakukan proses perbandingan dengan kandidat pembanding saja, karna data yang tidak memiliki kata kunci yang sama tidak akan diambil menjadi kandidat pembanding. Proses ini memiliki dampak yang besar karna sistem tidak perlu melakukan perbandingan data uji dengan data-data pada *dataset*, serta dapat mempersingkat waktu yang dibutuhkan dalam proses deteksi menggunakan LCS, *filter* ini akan bekerja seperti pada gambar 4.



Gambar 4. Flowchart Filter Kandidat Datasheet

C. *Pencarian Dokumen Terpanjang*

Pencarian dokumen terpanjang merupakan proses yang dilakukan sebelum menjalankan LCS, karena data parameter yang dibutuhkan LCS, parameter 1 harus lebih panjang dari parameter 2, sehingga dibuat sebuah fungsi pencarian data terpanjang untuk mempersiapkan variabel yang akan dimasukkan pada parameter LCS. Sehingga sistem ini benar-benar berjalan secara dinamis.

Jika X > Y		Jika Y > X
Maka	Atau	Maka
S1 = X		S1 = Y
S2 = Y		S2 = X

Keterangan :
S1 = variabel yang menyimpan yang panjang
S2 = variabel yang menyimpan lebih pendek
X = data uji
Y = kandidat pembanding

D. Metode Longest Common Subsequence

Longest Common Subsequences merupakan cara yang digunakan untuk menghitung dan mencari setiap rangkaian kata dalam deteksi plagiarisme. Pencarian ini dilakukan dengan cara membuat tabel matriks yang menyimpan hasil penghitungan LCS, matrik tersebut akan dibuat dengan rumus :

$$tab[i, j] = \begin{cases} tab[i-1, j-1] + 1, S1[i] = s2[j] \\ \max(tab[i-1, j], tab[i, j-1]) \end{cases}$$

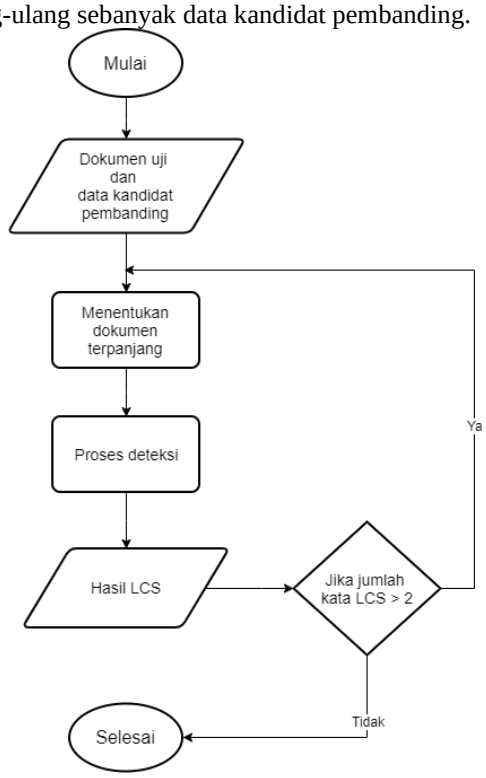
Keterangan :

tab[*i, j*] = panjang LCS yang ditemukan

S1 = data uji

S2 = kandidat pembanding

Karena pada dasar metode LCS hanya mengambil 1 hasil LCS dengan rangkaian kata terpanjang didalam 1 dokumen, maka dibuat sebuah kondisi perulangan untuk menemukan lebih dari 1 rangkaian kata yang plagiat. Kondisi pada perulangan ini dibatasi dengan jumlah kata hasil LCS lebih dari 2 kata, dalam kondisi ini diambil berdasarkan struktur kalimat terbaik adalah kalimat yang memiliki lebih dari 2 rangkaian kata. Dengan adanya kondisi ini akan membuat sistem dapat melakukan pencarian secara berulang-ulang dalam 1 dokumen pembanding, sehingga dalam 1 dokumen pembanding sistem dapat mencari dan mendapatkan lebih dari 1 hasil LCS. Selain itu metode juga dibuat secara dinamis dengan dibuat sebuah kondisi untuk mengetahui jumlah data kandidat pembanding, sehingga sistem dapat berjalan secara berulang-ulang sebanyak data kandidat pembanding.



Gambar 5. Flowchart Proses LCS

Implementasi metode lcs dilakukan menggunakan bahasa python. Berikut *source* pada imlementasi metode LCS ketika mengirim data dan memanggil file python :

```
public function executeCheker()
  {
    if ($this->input->post('submit')) {
      $data1['id_dokumen_user']        =
$this->input->post('id_dokumen_user');
      $data1['id_user']                =
$this->session-
>userdata('logged_in')['id_user'];
      $data1['created_at']             =
date('Y-m-d H:i:s');
      $data1['status']          = 1;

      $id_data      =      $this->input-
>post('id_data');

      $id           =           $this-
>Antrian_dokumen_model->insert($data1);

$data            = array();
      $path                            =
'./assets/python/setup.py';

      foreach ($id_data as $value) {
        array_push($data, array(
          'id_data' => $value,
          'id_antrian_dokumen' => $id,
          'created_at'  =>  date('Y-m-d
H:i:s')
        ));
      }

      $this->Antrian_dataset_model-
>insert_batch($data);
$output  =  passthru("python  ".$path."
".$data1['id_dokumen_user']."    ".$id);
//array  [0]  =  $path,  array  [1]  =
id_dokumen_user,  array  [2]  =  $value
(data yg di uji)
      redirect('tes-plagiasi');
    }else {
      redirect('tes-plagiasi/uji');
    }
  }
```

Implementasi penerimaan data dari PHP di terima oleh python :

```
if __name__== "__main__":
    db = Database()

    id_doc_uji  =  sys.argv[1]  #dokumen
yg di uji
    id_antrian = sys.argv[2] #ambil id
antrian
    doc_uji = (id_doc_uji,) #harus ada
koma agar tidak eror
    antrian = (id_antrian,)
```

```
    q = "SELECT * FROM dokumen_user WHERE id_dokumen_user = %s"
    data = db.queryv(q, doc_uji)

    q = "SELECT * FROM antrian_dataset as ad join dataset as ds on ad.id_data = ds.id_data WHERE id_antrian_dokumen = %s"
    data1 = db.queryall(q, antrian)

for td in data1:
        list_sentence = ""
        angka_sentence = 0

        penelitian = data['penelitian']
        while True:

            common_sentence, persent = LCS.longest_common_sentence(penelitian, td['penelitian'])
            common_words = common_sentence.split(' ')

            #berhenti ketika hasil plagiat kurang dari 3 kata
            if(len(common_words) < 3):
                break
            else:
                list_sentence += '%s###' % common_sentence #mengumpulkan hasil lcs ke 1 variable
                angka_sentence += float(persent)

                buang_plagiarisme = penelitian.split(common_sentence) #membuang kalimat plagiat pada text uji
                penelitian = ' '.join(buang_plagiarisme) #menggabungkan kembali setelah meembuang kalimat plagiat

        q = "UPDATE antrian_dataset SET similarity_angka='%s', similarity_text='%s', status=%d, updated_at='%s' WHERE id_antrian_dataset='%s' "%(angka_sentence, list_sentence, 1, datetime.datetime.now(), td['id_antrian_dataset'])
        db.query(q)
```

Implementasi matrik untuk mendapatkan hasil LCS :

```
def longest_common_substring(s1, s2):
        m = [[0] * (1 + len(s2)) for i in range(1 + len(s1))]
        longest, x_longest = 0, 0

        for x in range(1, 1 + len(s1)):
#ambil per kata dari doc 1, dan x sebagai doc terpanjang
          for y in range(1, 1 + len(s2)): #ambil per kata dari doc 2
            if s1[x - 1] == s2[y - 1]: #cek juka kata dari doc 1 == doc 2
              m[x][y] = m[x - 1][y - 1] + 1
              if m[x][y] > longest:
                longest = m[x][y]
                x_longest = x
            else:
              m[x][y] = 0
        return s1[x_longest - longest: x_longest] #memilih lcs terpanjang


longest_common_sentence(s1, s2):
        s1_arr = []
        s2_arr = []
        s1_words = s1.split(' ') #memisah tiap kata menjadi array
        s2_words = s2.split(' ')

        lg_s1 = len(s1_words) #menghitung panjang
        lg_s2 = len(s2_words)

        if lg_s1 < lg_s2:   #menentukan dokumen terpanjang
            s1_arr = s2_words
            s2_arr = s1_words
        else:
            s1_arr = s1_words
            s2_arr = s2_words

        lcs_arr = LCS.longest_common_substring(s1_arr, s2_arr)
        s1_pesent = (len(lcs_arr)*100)/len(s1_arr)

        lcs = ' '.join(lcs_arr) #menggabungkan array hasil lcs
        return lcs, s1_pesent
```

E. Ranking Pada Hasil Uji Plagiarisme

Data hasil uji yang telah tersimpan pada database akan ditampilkan kembali untuk ditampilkan pada halaman *user* dengan proses pengolahan data untuk mendapatkan ranking nilai plagiarisme tertinggi dan yang terendah dengan menggunakan fungsi `rsort()` pada PHP.

IV. KESIMPULAN

A. *Kesimpulan*

Adapun kesimpulan yang dapat diambil dari hasil penelitian yang dilakukan mengenai deteksi plagiarisme pada dokumen skripsi berdasarkan tingkat kesamaan dengan menggunakan *longest common subsequence* adalah sebagai berikut :

1. Metode Longest Common Subsequence dapat digunakan untuk deteksi plagiarisme dengan perbandingan dua atau lebih dokumen.
2. Sistem ini telah berhasil menerapkan metode Longest Common Subsequence untuk menguji lebih dari satu kalimat dan lebih dari satu kandidat pembanding.
3. Hasil pengujian yang telah dilakukan dengan tiga jenis setudi kasus sesuai dengan yang diharapkan, untuk pengujian accuracy ditunjukkan sebagai berikut :

a. Kasus Pertama

$\frac{84}{84} X\ 100$, maka hasil yang didapatkan adalah 100%.

b. Kasus Kedua

$\frac{44}{262} X\ 100$, maka hasil yang didapat pada kandidat pembanding 2 adalah 17%.
$\frac{41}{262} X\ 100$, maka hasil yang didapat pada kandidat pembanding 3 adalah 16%.
$\frac{29}{262} X\ 100$, maka hasil yang didapat pada kandidat pembanding 1 adalah 11%.

$\frac{44+41+29}{262} X\ 100$, maka total plagiarisme yang didapat pada dokumen uji adalah 43 %.

c. Kasus Ketiga

$\frac{47}{198} X\ 100$, maka hasil yang didapat pada kandidat pembanding 1 adalah 23%.
$\frac{24}{183} X\ 100$, maka hasil yang didapat pada kandidat pembanding 2 adalah 13%.

$\frac{47+24}{183} X\ 100$, maka total plagiarisme yang didapat pada dokumen uji adalah 39 %.

sehingga dapat disimpulkan bahwa deteksi plagiarisme menggunakan metode LCS dapat menghasilkan tingkat akurasi yang tinggi.

B. *Saran*

Saran yang dapat diberikan dari hasil penelitian untuk pengembangan sistem ini ke depan sebagai berikut:

1. Sistem dapat ditambahkan library lain seperti memperluas jangkauan diluar jurusan Teknologi Informasi, *crawling* dari internet.
2. Sistem dapat dikombinasikan dengan algoritma lain untuk mendapatkan tingkat akurasi yang lebih maksimal.

DAFTAR PUSTAKA

[1] Alamsyah, Nur, "Deteksi Plagiarisme Tingkat Kemiripan Judul Skripsi Dengan Algoritma Winnowing," Technologia,Vol.8, No.4, pp.205-213, Desember 2017.

[2] Widiastuti, Inta, Aplikasi Pendeteksi Kemiripan Dokumen Menggunakan Algoritma Rabin Karp, Malang : Politeknik Negeri Malang (2014).

[3] Pasma Cadea Mikha, Pengembangan Sistem Pendeteksi Kemiripan Karya Pada Inaicta 2013, Malang : Politeknik Negeri Malang (2014).

[4] Kharismadita Paratisa, Implementasi Tokenizing Plus Pada Sistem Pendeteksi Kemiripan Jurnal Skripsi Di Program Studi Teknik Informatika Politeknik Negeri Malang, Malang : Politeknik Negeri Malang (2015).

[5] Roshinta Trisna Ali, Analisis Skema-Skema Kemiripan Vektor Pada Sistem Penilaian Essay Online, Malang : Politeknik Negeri Malang (2016).

[6] Nurhadi Riza A, Pengembangan Data Uji Sistem Komputasi Kemiripan Teks Secara Semantik Berbahasa Indonesia, Malang : Politeknik Negeri Malang (2016).

[7] Dominikus Damas Putranto "Pengkajian Masalah *Longest Common Subsequences"* (2008). [Online]. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2007-2008/Makalah2008/MakalahIF2251-2008-047.pdf.

[8] Kensuke Baba, Tetsuya Nakatoh, Toshiro Minami "*Plagiarism detection using document similarity based on distributed representation*" 8th International Conference on Advances in Information Technology, IAIT2016, 19-22 December 2016, Macau, China, doi: https://doi.org/10.1016/j.procs.2017.06.038.

[9] Asad Abdi, Norisma Idris, Rasim M. Alguliyev, Ramiz M. Aliguliyev "*Plagiarism detection using linguistic knowledge*" Expert Systems With Applications (2015), doi: https://doi.org/10.1016/j.eswa.2015.07.048.

[10] Col Jyotindu Debnath "*Plagiarism: A silent epidemic in scientific writing – Reasons, recognition and remedies" Medical Journal Armed Forces India.* (2016), doi: https://doi.org/10.1016/j.mjafi.2016.03.010.

[11] L.K. Burdine, BA, M.B. de Castro Maymone, MD, DSc, N.A. Vashi, MD "*Text recycling: Self-plagiarism in scientific writing" International Journal of Women's Dermatology* (2018), doi: https://doi.org/10.1016/j.ijwd.2018.10.002.

[12] Alwi, dkk, Tata Bahasa Baku Bahasa Indonesia, cet. 8, Jakarta : Pusat Bahasa dan Balai Pustaka, 2010.

[13] Pratama, Rahmatullah Aditya, "Belajar UML - Use Case Diagram," 21 02 2019. [Online]. *Available*: https://www.codepolitan.com/mengenal-uml-diagram-use-case.

[14] Aini, Bariroh Isriya Nur, Identifikasi Kemiripan Judul Tugas Akhir PSTI Dan PSMI Di Politeknik Negeri Malang, Malang : Politeknik Negeri Malang (2014).

[15] Basuki, Alwan Pribadi, "Menguasai CodeIgniter Kasus Membangun Aplikasi Perpustakaan", Yogyakarta, Lokomedia, 2016.

[16] Galuh, Kresna, "Cara Menggunakan Bootstrap 3 untuk Membuat Web" 21 12 2015. [Online]. *Available* : https://www.codepolitan.com/tutorial/cara-menggunakan-bootstrap-3-untuk-membuat-web.

[17] Sidik., Betha, Framework Codeigniter, Bandung : Infomatika (2012).

[18] Solikin, I, Perancangan Sistem Infomasi Penjualan Berbasis Framework Model View Controller (MVC) Pada PT Thamrin Brother Cabang Oki, Jurnal Media Informatika dan Komputer, (2014), 4 (1), pp. 1-13.

[19] Riyanto, Membuat Sendiri Aplikasi *E-commerce* dengan PHP dan MySQL Menggunakan *Codeigniter* dan *JQuery*, Yogyakarta : Andi, (2011).