p-ISSN : 2443-2210
e-ISSN : 2443-2229

Jurnal Teknik Informatika dan Sistem Informasi
Volume 6 Nomor 3 Desember 2020

Analisis Pengaruh Teks *Preprocessing* Terhadap Deteksi Plagiarisme Pada Dokumen Tugas Akhir

http://dx.doi.org/10.28932/jutisi.v6i3.2892

Ariel Elbert Budiman [#1], Andreas Widjaja [✉#2]

[#] *Program Studi Magister Ilmu Komputer, Fakultas Teknologi Informasi*
Universitas Kristen Maranatha
Jl.Prof. drg. Surya Sumantri No.65, Bandung
[1]1879009@maranatha.ac.id
[2]andreas.widjaja@it.maranatha.edu

Abstract — **Final Project Report at a university has the potential for plagiarism. To detect possible plagiarism, String Similarity can be used. Text preprocessing is needed to process words which can make String Similarity results inaccurate. The value of the distribution of the results of the similarity that is getting higher shows the level of accuracy is also getting higher. Reports that contain many words can make it difficult to find plagiarism recommendations. In this study, we try to divide the report into each chapter to provide more detailed recommendation material. By using text preprocessing and comparison methods in the same chapter, can determine the characteristics of each chapter. The discovery of the characteristics of each chapter can be used as plagiarism recommendation material in more detail than a full-text report. The experiment was a comparison of the results of cosine similarity between the same chapters and full text, then combined with preprocessing stopword removal and stemming. The experimental results show that the use of preprocessing stopword removal and stemming can produce the highest distribution value and the similarity ratio in each chapter can show its characteristics. Words that represent the characteristics of a chapter can potentially become a stopword.**

Keywords— **Chapter Characteristic; Cosine Similarity; Document Preprocessing.**

I. PENDAHULUAN

Plagiarisme merupakan tindakan mengambil ide atau kata dari orang lain tanpa memberi kredit [1]. Plagiarisme menjadi salah satu permasalahan dalam bidang edukasi. Di setiap negara memiliki peraturannya masing-masing terkait ini. Di Indonesia, salah satu peraturan yang mengarah plagiarisme adalah Permendiknas No. 17 tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi. Sanksi ditetapkan dari yang paling ringan sampai paling berat. Salah satunya adalah teguran, peringatan tertulis, pembatalan ijazah, pemberhentian tidak dengan hormat dari status mahasiswa.

Walaupun sudah banyak yang menetapkan peraturan dan sanksi, tetap ada kemungkinan mahasiswa melakukan plagiarisme. Salah satu alasan plagiarisme pada mahasiswa salah satunya adalah mereka sedang mengejar waktu [2]. Dikarenakan waktu yang dibutuhkan untuk menyelesaikan studi tidak banyak dengan jumlah pelajaran yang harus dipelajari, mahasiswa terburu-buru mencari nilai agar lolos pada studinya. Dengan adanya internet juga semangkin mudah untuk mengakses informasi dan menyalinnya.

Menurut Jean Liddel [3] ada beberapa poin kenapa mahasiswa melakukan plagiarisme. Pertama, mahasiswa mengira tidak akan ketahuan saat melakukan plagiarisme. Mereka akan cenderung menutupi plagiarisme dengan berbagai cara. Kedua, berusaha sekecil mungkin untuk berhasil melewati mata pelajaran tersebut. Karena kebanyakan dalam system Pendidikan menggunakan tes dan nilai sebagai target kelulusan, maka mahasiswa cenderung fokus untuk mendapatkan nilai yang bagus tanpa melihat usaha dari mahasiswa tersebut.

Penggunaan algoritma *String Similarity* sudah ada digunakan untuk mendeteksi plagiarisme, salah satu contohnya adalah Leonard dan Hansun [4]. Terdapat bermacam-macam algoritma *String Similarity* yang dapat digunakan untuk mendeteksi plagiarisme contoh: *Rabin Karb Greedy String Tiling*, *Cosine Similarity*, *Levenshtein Distance*. Setiap algoritma memiliki hasil yang berbeda dan sama juga dengan *preprocessing*. Contoh *preprocessing* pada *String Similarity* pada umumnya adalah *stemming* dan *stopword*.

Tanpa *preprocessing* pada teks, hasil dari kemiripan tidak akurat karena kata umum tidak diolah atau dihilangkan. Karena kata umum pasti ada pada teks laporan, hasil kemiripan akan selalu tinggi. Dengan adanya *preprocessing*, persebaran angka hasil kemiripan akan membesar. Oleh karena itu akan dianalisis metode *preprocessing* yang menghasilkan persebaran hasil kemiripan tertinggi.

Pada penelitian ini, algoritma *String Similarity Cosine Similarity* dengan kombinasi metode *preprocessing* akan

digunakan. Lalu dianalisis seberapa besar hasil sebaran nilai perbandingan dari hasil *Cosine Similarity* menggunakan *preprocessing* tersebut.

Laporan yang dijadikan sebagai bahan *String Similarity* memiliki jumlah teks yang besar, sehingga mengetahui petunjuk plagiarisme pada laporan tersebut akan sulit. Untuk mengatasinya, akan dicoba melakukan perbandingan dengan memisahkan setiap bab. Dengan adanya perbandingan per bab, dapat dilakukan analisis terhadap karakteristik pada setiap bab.

Hipotesis dari penelitian ini adalah sebagai berikut:

1) Metode *Preprocessing Stopword + Stemming* menghasilkan sebaran nilai *similarity* (*cos θ)* paling tinggi daripada metode tanpa *Preprocessing*, *Stopword*, dan *Stemming*.
2) Menggunakan metode perbandingan bab menghasilkan tingkat *similarity* lebih besar daripada *full text*.

II. KAJIAN TEORI

A. String Plagiarism Detection Document Based

Untuk mendeteksi potensi terjadinya plagiarisme dapat membandingkan karya pada pasangan orang. Dalam dunia akademik, karya yang sering menjadi target plagiarisme merupakan tugas mahasiswa yang berbentuk teks dokumen. Untuk mencari pelaku plagiarisme, dosen akan membandingkan satu per satu pasangan dokumen yang dibuat oleh mahasiswa. Salah satu cara untuk membandingkan adalah dengan melihat kemiripan teks pasangan dokumen tersebut. Namun cara tersebut menghabiskan banyak waktu tergantung dari seberapa banyak teks yang harus dibandingkan dan jumlah mahasiswa, dan akurasi untuk mendeteksi terjadinya potensi plagiarisme disesuaikan dengan keinginan dosen tersebut. Sehingga hasilnya deteksi plagiarisme tidak akurat dan tidak efisien.

Salah satu upaya untuk mendeteksi plagiarisme yaitu menggunakan program untuk mengecek kemiripan teks. Berbagai program untuk mendeteksi plagiarisme telah dibuat, misalnya [5], [6], [7], [8], dan [9]. Dalam [7] melakukan deteksi plagiarisme dengan penambahan metode text *preprocessing*. Teks *preprocessing* yang dilakukan adalah *tokenization*, *stopword removal*, dan *stemming*. Graham dan Starr [9] melakukan survey pada mahasiswa dan staff mengenai penggunaan Turnitin. Turnitin merupakan sistem deteksi plagiarisme yang dipakai oleh 26 juta mahasiswa dan instruktur pada tahun 2014 [10]. Fungsionalitas utama dari Turnitin adalah melakukan perbandingan konten dengan jurnal yang telah diserahkan dan disimpan dalam database milik Turnitin. Dalam survey nya yang menggunakan *Bristol Online Survey service* menunjukan bahwa sebanyak 266 (72.5%) mahasiswa mengenali Turnitin dan 101 (27.5%) mahasiswa tidak. Pada Staff sebanyak 61 (98.4%) Staff mengenali Turnitin dan 1 (1.6%) Staff tidak. Dalam Turnitin tidak menetapkan plagiarisme berdasarkan persentase *threshold.* Melainkan, Turnitin akan melakukan *highlight* pada urutan teks yang sama pada paper yang lainnya sebagai bahan rekomendasi plagiarisme. Berdasarkan Fudge [11], angka *threshold* dalam menentukan plagiarisme tidak dapat ditentukan dengan sebuah angka. Karena terlalu banyak variabel yang perlu ditentukan seperti cara teknis mahasiswa melakukan plagiarisme dan akurasi *detection tools.*

Tokenization adalah proses memisahkan string teks menjadi bagian bagian yang disebut dengan *token* [12]. Kumpulan *token* yang memiliki karakter yang sama disebut dengan *type*. *Type* yang sudah dinormalisasikan disebut dengan *term*. Untuk pemotongan teks, dapat melakukan pemotongan teks pada seluruh karakter yang bukan alfanumerik. Cara ini dapat dilakukan pada teks yang berbahasa Indonesia. Untuk Bahasa Indonesia dapat diperlengkap dengan menambahkan karakter "/" dan "–". Namun, tidak seluruh Bahasa dapat menggunakan cara ini, contohnya adalah Bahasa Jepang.

Stopword merupakan kata yang sering muncul. Dalam *Information Retrieval* dan *text mining*, *stopwords* tidak begitu terpakai karena tidak membawa informasi [12], contohnya adalah kata sambung seperti "untuk", "di", "atau", "dari", dll. Fungsi utamanya dari pembuangan *stopword*s adalah mencegah agar *stopword*s tidak mempengaruhi hasil pada proses selanjutnya.

Stemming dalam *information retrieval* merupakan prosedur untuk mengurangi seluruh kata yang memiliki kata dasar yang sama. Biasanya dilakukan pemotongan pada imbuhan untuk menjadi kata dasar [13]. Salah satu algoritma yang digunakan untuk proses *stemming* Bahasa Indonesia dibuat oleh Tala [14].

Contoh hasil teks *preprocessing* dapat dilihat pada Tabel I.

TABEL I
CONTOH HASIL *PREPROCESSING*

Metode	Teks
Tanpa *Preprocessing*	seiring dengan perkembangan zaman, teknologi pun berkembang semakin pesat. Dengan adanya perkembangan teknologi tersebut, informasi dapat diperoleh dengan lebih efisien dan efektif. Selain itu, perkembangan teknologi juga membuat banyak organisasi menggunakan sistem yang terkomputerisasi, agar informasi dapat diperoleh dengan cepat. Selain itu, komputerisasi juga dapat meminimalisirkan human error yang sering terjadi
Tokenization	seiring;dengan;perkembangan;zaman,;teknologi;pun;berkembang;semakin;pesat;Dengan;adanya;perkembangan;teknologi;tersebut;informasi;dapat;diperoleh;dengan;lebih;efisien;dan;efektif;Selain;i

Metode	Teks
	tu;perkembangan;teknologi;juga;membuat;banyak;organisasi;menggunakan;sistem;yang;terkomputerisasi;agar;informasi;dapat;diperoleh;dengan;cepat;Selain;itu;komputerisasi;juga;dapat;meminimalisirkan;human;error;yang;sering;terjadi;
Stemming	iring;dengan;kembang;zaman;teknologi;pun;kembang;makin;pesat;dengan;ada;kembang;teknologi;sebut;informasi;dapat;oleh;dengan;lebih;efisien;dan;efektif;selain;itu;kembang;teknologi;juga;buat;banyak;organisasi;guna;sistem;yang;komputerisasi;agar;informasi;dapat;oleh;dengan;cepat;selain;itu;komputerisasi;juga;dapat;meminimalisirkan;human;error;yang;sering;jadi
Stopword Removal	iring;kembang;zaman;teknologi;kembang;pesat;kembang;teknologi;informasi;efisien;efektif;kembang;teknologi;organisasi;sistem;komputerisasi;informasi;cepat;komputerisasi;meminimalisirkan;human;error

Sesudah text *preprocessing* dilakukan pada kedua dokumen, algoritma string *similarity* akan dijalankan. Beberapa algoritma *String Similarity* telah diimplementasikan untuk deteksi plagiarisme. Dalam [7], menggunakan algoritma *Rabin Karb Greedy String Tiling* (RKRGST) untuk pengecekan *String Similarity* pada datanya. Sulistiani dan Karnalim [8] menggunakan *Cosine Similarity* dan alternative algoritma yang lain RKRGST, *Local Alignment* dan *Local Alignment with cosine-based filtering.*

B. Cosine Similarity

Algortma *cosine similarity* sering digunakan untuk mengukur kemiripan (*similarity*) antara dua dokumen. Kedua dokumen direpresentasikan sebagai vektor-vektor, yang disebut sebagai vektor-vektor dokumen [15]. *Similarity* kedua dokumen dihitung dari normalisasi dot product dari kedua vektor tersebut, yang sama dengan kosinus sudut yang dibentuk oleh kedua vektor dokumen tersebut, seperti pada persamaan (1) berikut ini:

$$\cos\theta = sim(x,y) = \frac{x.y}{||x||||y||}, \quad (1)$$

Dimana $||x||$ merupakan Euclidian norm dari vektor $x = (x_1, x_2, \dots, x_p)$, yaitu $||x|| = \sqrt{x_1^2 + x_2^2 + \dots + x_p^2}$, yang disebut juga sebagai "Panjang" vektor [16]. Begitu pula dengan $||y||$ merupakan *Euclidian norm* dari vektor y. Nilai $\cos\theta$ yang semangkin mendekati 1 atau semakin kecil sudutnya berarti kedua vektor tersebut semakin mirip. Sebaliknya apabila nilai $\cos\theta$ semangkin mendekati 0 berarti kedua vektor semakin tidak mirip.

C. Pembuatan Vektor Dokumen

Pertama, query akan menghasilkan *term* dari dokumen dimana *term* merupakan setiap katanya [15]. Berikutnya setiap *term* ditentukan nilai weight, total kemunculan *term* dari dokumen tersebut. Cara pemberian weight ini dapat disebut juga dengan *term frequency*. Setelah melakukan *term frequency* pada seluruh *term*, diurutkan dan menjadi model bag of words. Pada titik ini, dokumen "halo salam kenal" dengan dokumen "salam kenal halo" merupakan dokumen yang sama karena memiliki *term frequency* yang sama.

Dari kumpulan *term frequency* dari sebuah dokumen, nilai setiap *term frequency* diubah menjadi satu komponen vektor. Lalu setiap *term* dari kumpulan *term* dari dokumen merepresentasikan satu *axis* dalam *vector space*.

III. TINJAUAN PUSTAKA

Dalam penelitian ini, sistem deteksi plagiarisme laporan tugas akhir menggunakan *Cosine Similarity* dan juga mencari kombinasi teks *preprocessing* yang sesuai untuk sistemnya. Ada beberapa penelitian yang mendukung dalam pembuatan sistem deteksi plagiarisme dan analisis teks *preprocessing*.

Parwitta, Indradewi, dan Wijaya [7] membuat sistem deteksi plagiarisme menggunakan *Rabin Karp Hashing* pada *Hash*. Lalu menggunakan algoritma *Dice's Similarity Coefficients* untuk mengecek kemiripannya. Untuk *preprocessing*nya menggunakan *Tokenization*, *Stopword*s dan *stemming*. Penelitian ini merupakan dasar penelitian yang dipakai.

Mardiana, Adji, dan Hidayah [17] melakukan analisis pengaruh *Stemming* pada *Similarity Detection* pada Bahasa Indonesia. Hasilnya menunjukan bahwa penggunaan *stemming* dan *stopword*s memiliki presisi yang lebih besar pada yang menggunakan *stemming* saja dan tanpa *preprocessing*. Pada fourgram penggunaan *stemming* dan *stopword*s menghasilkan persentase 67% sedangkan yang lain hanya sekitar 30%. Pada penelitian ini menunjukan adanya hasil dari penggunaan *Stemming* dan *stopword* pada abstract. Hasilnya dapat dijadikan kerangka penelitian namun menggunakan data yang lebih besar yaitu keseluruhan isi laporan.

Sulistiani dan Karnalim [8] membuat sistem deteksi plagiarisme dengan beberapa alternative metode *String Similarity*. Metode yang digunakan adalah Rabin Karb

Greedy String Tiling (RKRGST), *Cosine Similarity*, *Local alignment* dan *Local alignment with cosine-based filtering*. Dari ke empat metode tersebut, RKRGST memiliki deteksi yang efektif namun memiliki waktu proses yang sangat lama.

Karena data yang akan digunakan berupa dokumen maka pada penelitian yang akan dilakukan ini perlu menggunakan *Cosine Similarity*. Dalam dokumen memiliki banyak teks, sehingga perlu difokuskan dalam kecepatan komputasinya dan *Cosine Similarity* memiliki komputasi yang cepat dalam teks yang banyak.

IV. IMPLEMENTASI PENELITIAN

A. Implementasi Preprocessing Data

Data yang digunakan berupa laporan Tugas Akhir berbahasa Indonesia dari sebuah universitas sebanyak 24 data. Format dari laporan tersebut berbentuk file PDF yang terbagi setiap bab.

File PDF tersebut di ubah menjadi teks agar dapat digunakan oleh *String Similarity*. Untuk pengubahan file pdf menjadi teks menggunakan algoritma *Optical Character Recognition* (OCR). Fungsi dari OCR untuk mengubah file gambar yang berisi tulisan menjadi file teks.

File PDF pertama akan diubah dulu menjadi gambar dalam bentuk format .jpg. Setiap file akan iterasi setiap halaman menjadi satu .jpg. Lalu setiap gambar diubah menjadi teks dengan bantuan *library* pytesseract [18]. Python-tesseract (pytesseract) merupakan *Optical Character Recognition tool* untuk python. *Library* ini dapat mengenali dan mengubah menjadi teks dari gambar.

B. Implementasi algoritma Cosine Similarity

File yang sudah diubah menjadi teks diproses kembali. Sistem akan mengambil dua dokumen untuk dibandingkan dengan *Cosine Similarity*. Proses *Cosine Similarity* menggunakan bantuan *library* sklearn [19]. Teks diubah menjadi vektor berupa matriks yang berisi token jumlah dari jenis kata dengan bantuan *CountVectorizer* [20]. Lalu vektor tersebut

Selanjutnya untuk perbandingan menggunakan dua cara yaitu dengan per bab (A) atau seluruh bab (B). Pada metode perbandingan per bab akan dibandingkan pada bab yang sama dengan pasangannya.

Untuk *preprocessing* teks akan dikombinasi metode perbandingan bab dengan 3 cara: tanpa *preprocessing* (Z), *Stopword Removal* (S) dan *Stemming* (ST). Lalu sesudah dua dokumen sudah selesai *preprocessing* teks akan dibandingkan dengan *cosine similarity*. Program dengan kombinasi metode akan dijalankan secara terpisah. Proses *Stopword Removal* menggunakan *library* nltk [21] *stopword*s yang Bahasa Indonesia. *Natural Language Toolkit* (nltk) merupakan platform untuk membuat program Python dapat bekerja dengan data bahasa manusia. Nltk memiliki fitur *text processing* untuk *classification*, *tokenization*, *stemming*, *tagging*, *parsing*, *semantic reasoning*. Sedangkan pada proses *Stemming* menggunakan *library* Sastrawi [22] yaitu *library Stemming* untuk Bahasa Indonesia. *Library* ini menggunakan Algoritma *Stemming* dari Nazief dan Adriani [23] untuk Bahasa Indonesia. Kata dasar yang dipakai berdasarkan data kamus kata dasar dari kateglo.com dengan sedikit perubahan.

TABEL II
KOMBINASI METODE *PREPROCESSING*

	Metode Preprocessing				
	Z	**S**	**ST**	**STS**	**SST**
Per Bab (A)	AZ	AS	AST	ASTS	ASST
Seluruh Bab(B)	BZ	BS	BST	BSTS	BSST

Tabel II merupakan kombinasi metode yang akan dilakukan untuk eksperimen. Pada metode Per Bab (A) akan dibandingkan bab yang sama dengan pasangannya, contoh: ID1_Chapter1 dengan ID2_Chapter1. Sedangkan metode Seluruh Bab (B), teks pada setiap bab digabungkan menjadi satu teks. Lalu teks gabungan tersebut di bandingkan dengan teks gabungan lainnya. Setiap metode A dan B dikombinasikan dengan metode *preprocessing* tanpa *preprocessing* (Z), *stopword removal* (S), *stemming* (ST), *stemming + stopword removal* (STS) dan *stopword removal + stemming* (SST).

Kode Program 1. ***Pseudo Code Cosine Similarity***

```
def cosine_simimlarity(a,b):
    return float(dot(a, b) / (norm(a) * norm(b)))
```

Pada Kode Program 1 menunjukan cara menghitung Cosine *Similarity*. Variabel a dan b merupakan vektor berupa *list* dari angka. Fungsi dot(a,b) akan melakukan *dot product* pada vektor a dan b. Fungsi norm(a) berupa *Euclidean norm* dari vektor a, yaitu $||a|| = \sqrt{a_1^2 + a_2^2 + \ldots + a_p^2}$. Begitu pula sama dengan vektor b pada norm(b).

Kode Program 2. ***Cosine Similarity*** **pada Per Bab**

```
chapterList = list of chapter on document
sampleData = list of sample data name
dataPath = path to list of data on folder
allDataList = list of folder names of data

def readDataChapter(currDataPath,dataName):
        listChapterText = list
```

```
        for i in length of chapterList:
                file = open file on currDataPath contains dataName and chapterList[i]
                listChapterText[i] = file.lowerCase()
        return listChapterText

for first in sampleData:
        for second in allDataList:
                if first==second:
                        continue
                firstListText = readDataChapter(dataPath+first,first)
                secondListText = readDataChapter(dataPath+second,second)
                for i in length of chapterList:
                        sentences = [firstListText[i], secondListText[i]]
                        vectors = fit_transform(sentences) [24]
                        print(cosine_similarity(vectors[0],vectors[1]))
```

Kode Program 3. *Cosine Similarity* pada *Full Text*

```
chapterList = list of chapter on document
sampleData = list of sample data name
dataPath = path to list of data on folder
allDataList = list of folder names of data

def readDataFull(currDataPath,dataName):
        string fullChapter
        for i in length of chapterList:
                file = open file on currDataPath contains dataName and chapterList[i]
                fullChapter += file.lowerCase()
        return fullChapter

for first in sampleData:
        for second in allDataList:
                if first==second:
                        continue
                firstListText = readDataFull(dataPath+first,first)
                secondListText = readDataFull(dataPath+second,second)
                sentences = [firstListText, secondListText]
                vectors = fit_transform(sentences) [24]
                print(cosine_similarity(vectors[0],vectors[1]))
```

Variabel *chapterList* merupakan *list* nama bab yang akan dipakai untuk dibandingkan pada laporan tersebut. Variabel *sampleData* berisi *list* sampel laporan yang akan dibandingkan dengan seluruh laporan. Variabel *dataPath* berisi *folder path* seluruh data laporan Tugas Akhir. Variabel *allDataList* berisi list nama folder seluruh data laporan Tugas Akhir.

Pada Kode Program 2 dan Kode Program 3 terdapat *function read data* hasil .txt dari algoritma OCR pada data laporan. *Function readDataChapter* menghasilkan *list* dari *string* teks setiap bab sedangkan pada *readDataFull* menghasilkan gabungan teks seluruh bab menjadi satu *string.*

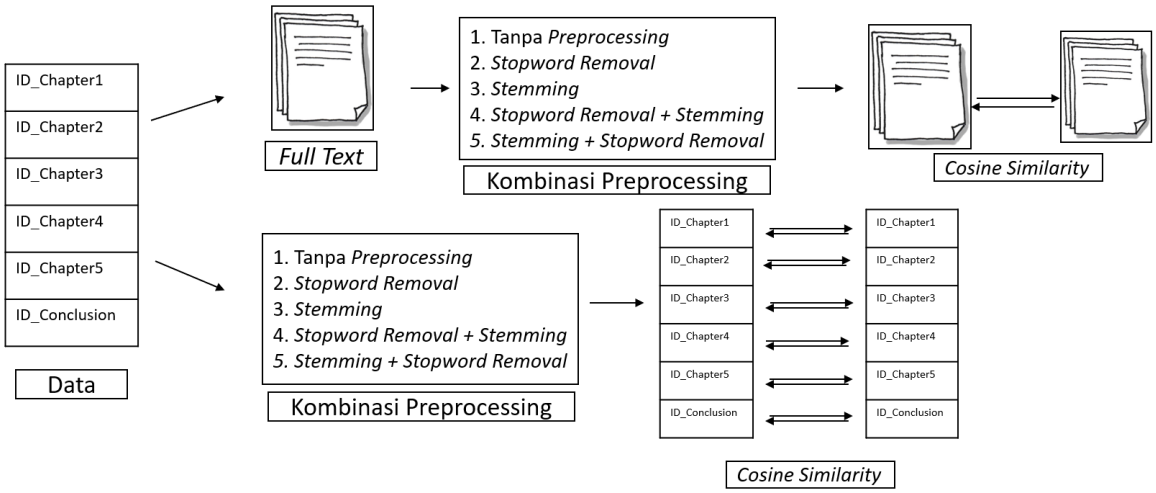
Untuk perhitungan perbandingan laporan menggunakan algoritma *Cosine Similarity*. Pertama, list *sampleData* dibandingkan dengan list *allDataList* tanpa membandingkan laporan yang sama. Pada Kode Program 2, akan dilakukan iterasi pada setiap bab yang terdaftar pada *chapterList*. Teks bab *firstListText* dan *secondListText* dimasukan pada variabel *sentences* yang akan dilakukan *mapping* kamus kata menggunakan fungsi *fit_transform* [24] dari *library CountVectorizer* menghasilkan *list* vektor dua laporan berdasarkan kamus kata tersebut. Lalu vektor akan dihitung nilai perbandingannya menggunakan *cosine_similarity* pada Kode Program 1 [25] dari *library* nltk.

C. Pembuatan Eksperimen

Data yang diuji coba sebanyak 24 data laporan Tugas Akhir program studi teknologi informasi pada sebuah universitas. Setiap data teks laporan tersebut diberi nama Mxx, dimana M merupakan penamaan data teks dan x merupakan nomor data teks mahasiswa. Sehingga terdapat data teks M01 hingga M024.

Lalu dilakukan 3 kali eksperimen dengan data yang berbeda. File teks yang digunakan yaitu: ID_Chapter1.pdf, ID_Chapter2.pdf, ID_Chapter3.pdf, ID_Chapter4.pdf, ID_Chapter5.pdf, dan ID_Conclusion.pdf. Total perbandingan yang dilakukan: 23 (Perbandingan tanpa dengan ID yang sama) x 3 (Case) x 10 (Metode) = 690 perbandingan. Gambaran eksperimen tersebut dapat dilihat pada Gambar 1.

p-ISSN : 2443-2210
e-ISSN : 2443-2229



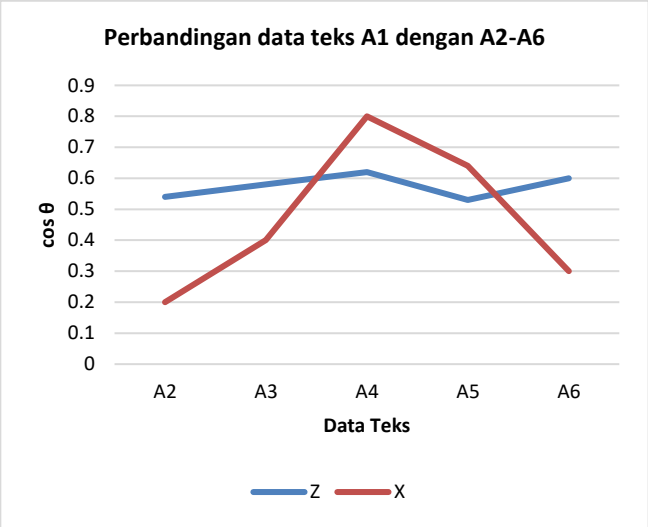
Gambar 1. Metodologi Penelitian

V. HASIL PENELITIAN

A. Penjelasan Hipotesis

Berdasarkan hasil yang didapatkan, terdapat hipotesis sebagai bahan rekomendasi metode yang digunakan untuk program deteksi plagiarisme.

1) Metode preprocessing kata efektif digunakan apabila berhasil membuat nilai perbandingan mempunyai persebaran yang lebih besar. Contoh: Z merupakan hasil perbandingan dengan metode tanpa preprocessing, dan X dengan metode preprocessing. Gambar 2 merupakan contoh grafik dari perbandingan data teks seorang mahasiswa (A1) dengan data teks mahasiswa yang lainnya (A2-A6, karena data teks mahasiswa A1 tidak dibandingkan dengan dirinya sendiri) dengan metode Z dan X.

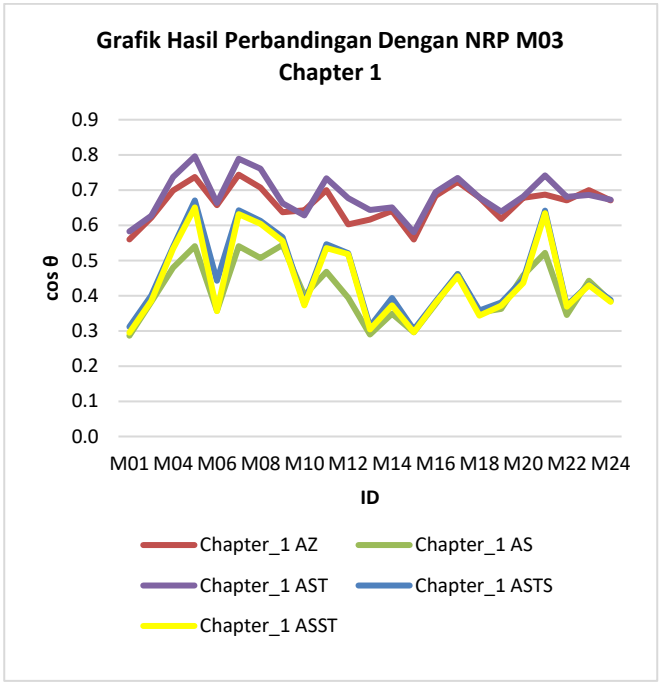


Gambar 2 Contoh Grafik Hipotesis PertamaAnalisis Hasil Dengan Hipotesis Pertama

2) Nilai *similarity* pada perbandingan *full text* akan selalu lebih besar dengan nilai *similarity* pada perbandingan setiap bab yang sama.

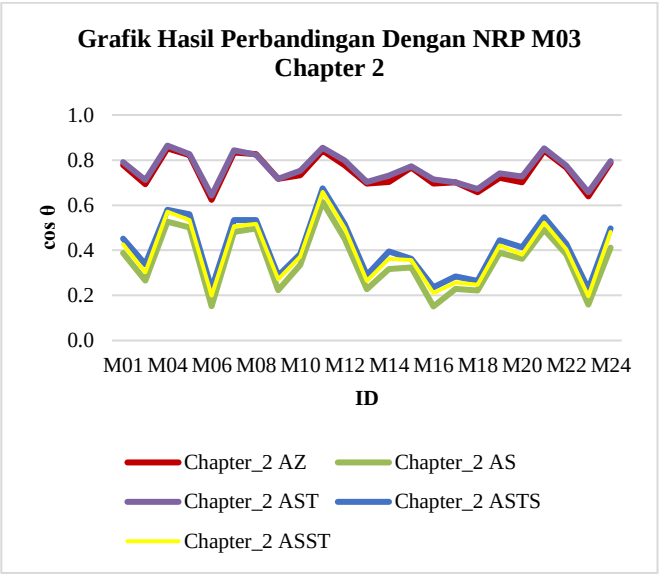
B. Analisis Hasil Dengan Hipotesis Pertama

Pada eksperimen pertama menggunakan data perbandingan ID M03 dengan seluruh ID yang lain menghasilkan Gambar 3 - Gambar 9. Untuk menghitung persebaran pada hasil kemiripan pada laporan, digunakan nilai standar deviasi dari hasil kemiripan kombinasi metode.



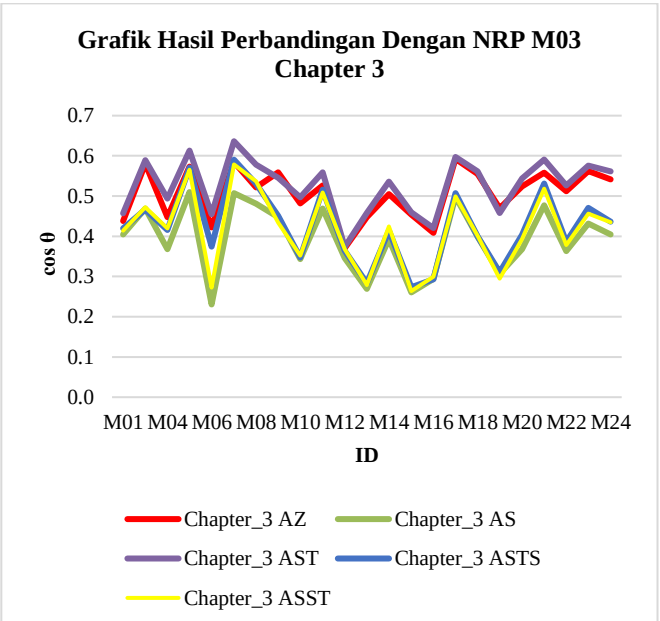
Gambar 3 Grafik Hasil Perbandingan Dengan ID M03 Chapter 1

Pada Gambar 3, metode AZ dengan AST tidak memiliki perbedaan yang signifikan. Metode AS, ASTS, dan ASST juga tidak memiliki perbedaan yang signifikan namun secara kesuluruhan nilai perbandingannya (cos θ) menurun.



Gambar 4 Grafik Hasil Perbandingan Dengan ID M03 Chapter 2

Pada Gambar 4, hasilnya memiliki pola yang sama dengan Gambar 3. Metode AZ dengan AST tidak memiliki perbedaan yang signifikan. Metode AS, ASTS, dan ASST juga tidak memiliki perbedaan yang signifikan namun secara kesuluruhan nilai perbandingannya (cos θ) menurun.

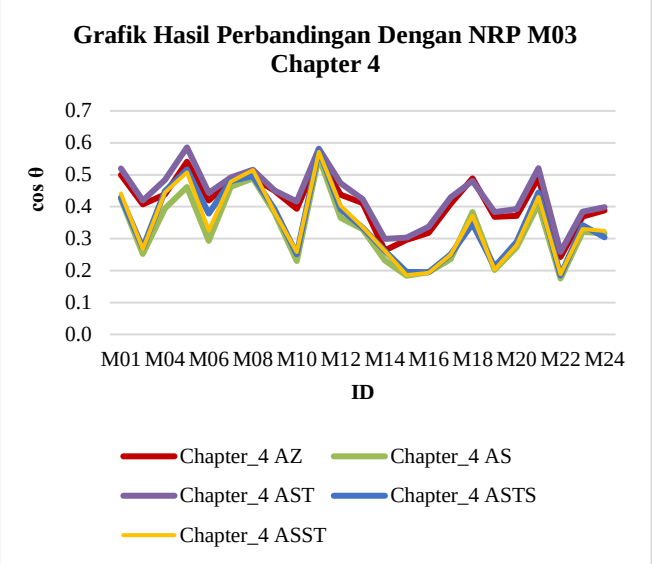


Gambar 5 Grafik Hasil Perbandingan Dengan ID M03 Chapter 3

Pada Gambar 5, memiliki pola yang berbeda dengan Gambar 3 dan Gambar 4. ID M04 - M06, M10, dan M22 seluruh metode nya tidak menghasilkan perbedaan cos θ yang signifikan. Namun secara kesuluruhan metode AZ dan

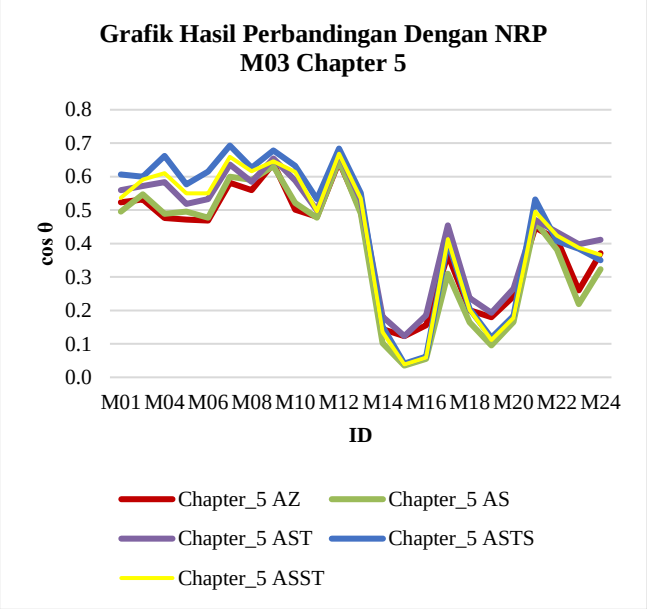
p-ISSN : 2443-2210
e-ISSN : 2443-2229

AST menghasilkan cos θ yang lebih tinggi dari metode AS, ASTS, dan ASST.



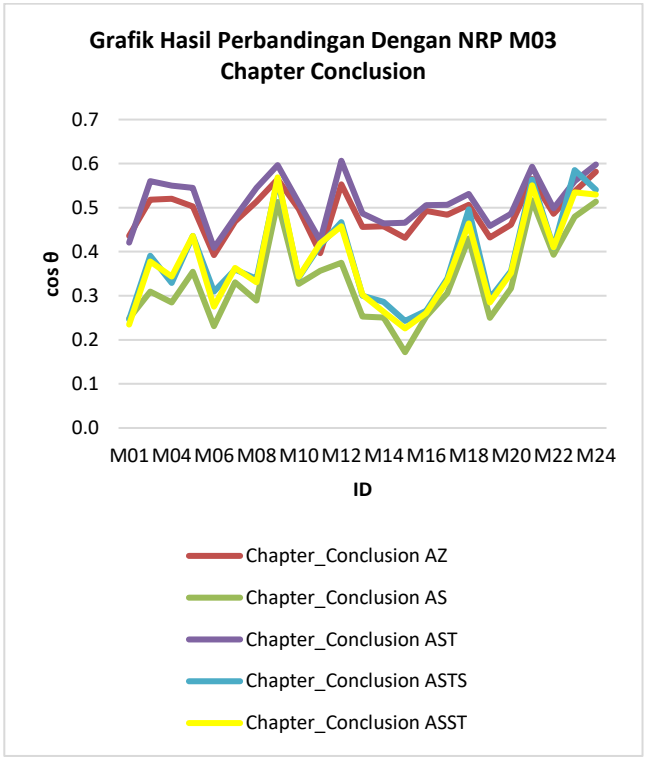
Gambar 6 Grafik Hasil Perbandingan Dengan ID M03 Chapter 4

Gambar 6 menunjukkan metode AZ dan AST dengan AS, ASTS, dan ASST di Chapter 4 tidak memiliki perbedaan yang signifikan. Setelah ditelusuri, data teks pada Chapter 4 memiliki campuran antara teks Bahasa Indonesia dan Bahasa Inggris. Berdasarkan seluruh data, Chapter 4 cenderung memiliki teks kode program. Karena *preprocessing* yang dilakukan hanya pada teks Bahasa Indonesia, perbedaan nilai *similarity* yang dihasilkan tidak signifikan.



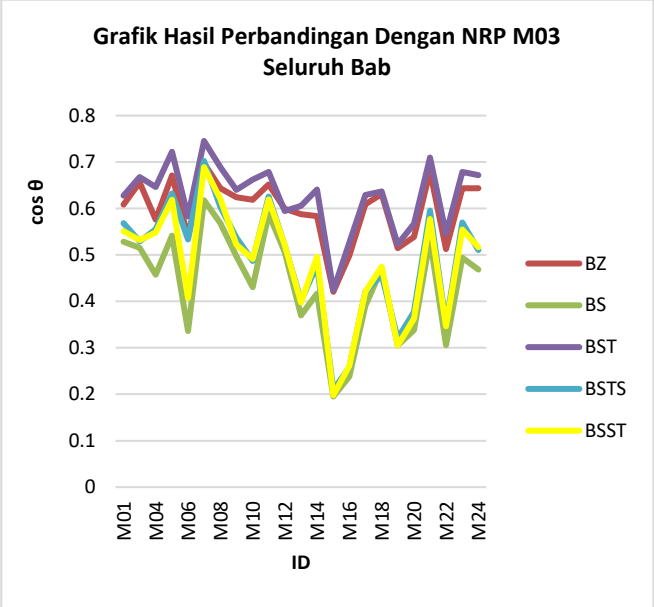
Gambar 7 Grafik Hasil Perbandingan Dengan ID M03 Chapter 5

Berdasarkan Gambar 7, terdapat penurunan nilai kemiripan pada ID M13. Setelah ditelusuri, ID M01 hingga M12 termasuk ID M03 memiliki format laporan yang sama pada Chapter 5. Sedangkan ID M14 hingga M24, menggunakan format yang berbeda beda sehingga menghasilkan nilai cos θ yang bervariasi.



Gambar 8 Grafik Hasil Perbandingan Dengan ID M03 Chapter Conclusion

Berdasarkan Gambar 8, metode AZ dan AST dengan AS, ASTS, dan ASST memiliki perbedaan yang cukup signifikan. Namun pada perbandingan ID M08, M18 dan M22 tidak memiliki perbedaan yang signifikan.



Gambar 9 Grafik Hasil Perbandingan Dengan ID M03 Seluruh Bab

Berdasarkan Gambar 9, metode AZ dan AST dengan AS, ASTS, dan ASST memiliki perbedaan yang cukup signifikan.

Berdasarkan Gambar 3 hingga Gambar 9, metode AZ dan AST menghasilkan nilai cos θ yang mirip. Lalu metode AS, ASTS, dan ASST menghasilkan nilai cos θ yang lebih rendah dari AZ dan AST. Nilai cos θ pada metode AS, ASTS, dan ASST tidak memiliki perbedaan yang signifikan.

TABEL III
JUMLAH STANDAR DEVIASI TERTINGGI DARI 3 EKSPERIMEN

No	ID	Z	S	ST	STS	SST
1	M03	0	1	0	3	3
2	M11	0	4	0	1	2
3	M21	0	4	0	3	0
	Total	0	9	0	7	5

Tabel III merupakan hasil penjumlahan nilai Standar Deviasi tertinggi dari perbandingan nilai *similarity* 3 eksperimen dengan kombinasi *preprocessing*. Berdasarkan Tabel III, dari tiga eksperimen dengan ID M03, M11, dan M21 metode S memiliki total Standar Deviasi Tertinggi. Namun, penggabungan *preprocessing stopword removal* dengan *stemming* memiliki jumlah total yang lebih besar dari metode *stopword*. Untuk menguji seberapa besar pengaruh urutan *preprocessing stopword removal* dan *stemming*, dapat menggunakan uji signifikansi dengan T-Test. Untuk nilai keyakinan / *Alpha* yang ditetapkan sebesar 0.05 atau 5%.

TABEL IV
T-TEST PADA METODE *PREPROCESSING* STS DAN SST

Percobaan ke	Standar Deviasi STS	Standar Deviasi SST
1	0.113200	0.114586
2	0.122843	0.133375
3	0.127292	0.128744
4	0.128570	0.131216
5	0.122292	0.123959
6	0.129229	0.130659
7	0.090304	0.093963
8	0.110409	0.113629
9	0.129395	0.132632
10	0.112983	0.113571
11	0.138067	0.137637
12	0.125985	0.124094
13	0.124189	0.124446
14	0.132236	0.135796
15	0.127823	0.129422
16	0.219121	0.207233
17	0.163111	0.153698
18	0.154109	0.151013
19	0.105973	0.103665
20	0.114740	0.120380
21	0.100381	0.102598
Rata- rata	0.128202	0.128872
T-test	0.93097	
Alpha	0.05	

Berdasarkan Tabel IV, pengujian T-test menghasilkan p-value sebesar 0.93097. Nilai tersebut lebih besar dari *Alpha* yang ditetapkan yaitu 0.05. Oleh karena itu, perbedaan urutan penggabungan metode *preprocessing Stemming* dengan *Stopword Removal* tidak menghasilkan nilai yang signifikan. Sehingga, total nilai tertinggi standar deviasi metode STS dan SST, apabila digabungkan memiliki nilai total yang lebih besar daripada metode S. Kesimpulannya adalah, penggunaan gabungan metode *preprocessing stopword* removal dan *stemming* menghasilkan sebaran nilai *similarity* yang terbesar.

C. Analisis Hasil Dengan Hipotesis Kedua

Untuk melakukan eksperimen hipotesis kedua diuji dengan membandingkan hasil *similarity* pada 3 eksperimen. Hasil *similarity* per bab dibandingkan dengan hasil *similarity full text* dengan metode *preprocessing stopword removal + stemming*. Apabila nilai *similarity* pada per bab lebih besar daripada *full text* akan diberi tanda warna merah pada tabel V, VI, dan VII.

Tabel V menunjukkan nilai *similarity* M03 dengan yang lainnya. Total nilai perbandingan yang lebih besar dari *Full text* pada Chapter_1 sebanyak 11, Chapter_2 sebanyak 6, Chapter_3 sebanyak 5, Chapter_4 tidak ada, Chapter_5 sebanyak 10, dan Chapter_Conclusion sebanyak 7.

p-ISSN : 2443-2210
e-ISSN : 2443-2229

TABEL V
PERBANDINGAN NILAI *SIMILARITY* M03 PADA EKSPERIMEN PERTAMA

ID	Chapter_1 (*cos θ*)	Chapter_2 (*cos θ*)	Chapter_3 (*cos θ*)	Chapter_4 (*cos θ*)	Chapter_5 (*cos θ*)	Chapter_ Conclusion (*cos θ*)	*Full Text*
M01	0.311987	0.452193	0.419967	0.428059	0.606286	0.246161	0.568346
M02	0.400682	0.337463	0.466366	0.270798	0.600001	0.390934	0.529194
M04	0.539122	0.579108	0.416043	0.448489	0.662449	0.329244	0.555510
M05	0.670660	0.559378	0.570961	0.517071	0.576442	0.436025	0.632701
M06	0.442869	0.229232	0.374316	0.377869	0.615098	0.309211	0.533066
M07	0.642418	0.534071	0.590443	0.480036	0.692905	0.359955	0.702468
M08	0.612390	0.534627	0.530185	0.495354	0.626508	0.338260	0.603716
M09	0.565995	0.287333	0.450814	0.390917	0.678672	0.564004	0.537418
M10	0.377356	0.383099	0.346968	0.250974	0.632625	0.341576	0.486966
M11	0.546362	0.674784	0.516028	0.577989	0.532536	0.415033	0.625806
M12	0.520845	0.515697	0.363934	0.391881	0.683792	0.467415	0.520167
M13	0.312362	0.290016	0.283245	0.334616	0.551423	0.300251	0.401957
M14	0.393777	0.395514	0.414049	0.262385	0.151622	0.286171	0.477923
M15	0.303668	0.362375	0.273401	0.195656	0.040908	0.242944	0.209140
M16	0.383533	0.235917	0.292990	0.194462	0.061626	0.265771	0.257912
M17	0.461182	0.284382	0.507540	0.250554	0.405417	0.338987	0.421215
M18	0.357681	0.264515	0.399725	0.346614	0.204548	0.496564	0.457084
M19	0.380433	0.444686	0.310975	0.211634	0.118432	0.294105	0.320249
M20	0.446560	0.412944	0.404046	0.290069	0.183027	0.356285	0.377686
M21	0.641469	0.547065	0.531368	0.443973	0.532278	0.563587	0.594980
M22	0.372365	0.428631	0.386046	0.184445	0.407547	0.409355	0.349399
M23	0.431951	0.227861	0.470273	0.341903	0.384813	0.585328	0.569816
M24	0.387640	0.496662	0.436270	0.304016	0.349637	0.540798	0.510461

TABEL VI
PERBANDINGAN NILAI *SIMILARITY* M11 PADA EKSPERIMEN KEDUA

ID	Chapter_1 (*cos θ*)	Chapter_2 (*cos θ*)	Chapter_3 (*cos θ*)	Chapter_4 (*cos θ*)	Chapter_5 (*cos θ*)	Chapter_ Conclusion (*cos θ*)	*Full Text*
M01	0.248407	0.353756	0.399907	0.531619	0.386653	0.181448	0.471350
M02	0.368510	0.241044	0.395790	0.236049	0.462882	0.209359	0.491090
M03	0.659592	0.409007	0.492426	0.430747	0.605002	0.452099	0.647488
M04	0.546362	0.674784	0.516028	0.577989	0.532536	0.415033	0.625806
M05	0.651523	0.433100	0.563527	0.567907	0.450585	0.423065	0.636424
M06	0.394446	0.211366	0.353973	0.316832	0.403245	0.211779	0.435163
M07	0.610207	0.421385	0.605117	0.494678	0.549949	0.499345	0.643449
M08	0.628168	0.442857	0.696914	0.462762	0.682400	0.406736	0.739498
M09	0.540299	0.240846	0.404558	0.423341	0.354141	0.283112	0.514639
M10	0.457159	0.318565	0.419865	0.257153	0.311773	0.157424	0.515806
M12	0.389607	0.545413	0.628024	0.343894	0.520624	0.288091	0.690269
M13	0.303059	0.231657	0.309505	0.464581	0.401454	0.198085	0.395658
M14	0.424900	0.227589	0.439265	0.254382	0.133220	0.174617	0.465904
M15	0.317885	0.254774	0.346929	0.211003	0.036531	0.169039	0.240094
M16	0.369394	0.233367	0.315697	0.141942	0.106045	0.147316	0.286742
M17	0.396581	0.271961	0.415776	0.236125	0.333613	0.161368	0.406956
M18	0.321170	0.183685	0.531339	0.392898	0.178209	0.260387	0.548636
M19	0.326521	0.316528	0.415059	0.159840	0.181658	0.095364	0.429762
M20	0.421006	0.273305	0.424189	0.227338	0.445865	0.191798	0.511116
M21	0.617354	0.394568	0.573586	0.583260	0.433086	0.420313	0.676971
M22	0.371127	0.339284	0.489186	0.227755	0.289965	0.289499	0.388538
M23	0.430233	0.230406	0.612330	0.371076	0.359905	0.306237	0.657499
M24	0.391840	0.465353	0.624071	0.362993	0.428072	0.288606	0.605010

Tabel VI menunjukkan nilai *similarity* M11 dengan yang lainnya. Total nilai perbandingan yang lebih besar dari *full text* pada Chapter_1 sebanyak 5, Chapter_2 sebanyak 2, Chapter_3 sebanyak 5, Chapter_4 sebanyak 2, Chapter_5 sebanyak 1, dan Chapter_Conclusion tidak ada.

Tabel VII menunjukkan nilai *similarity* M21 dengan yang lainnya. Total nilai perbandingan yang lebih besar dari *full text* pada Chapter_1 sebanyak 7, Chapter_2 sebanyak 4, Chapter_3 sebanyak 5, Chapter_4 sebanyak 0, Chapter_5 sebanyak 2, dan Chapter_Conclusion sebanyak 2.

TABEL VII
PERBANDINGAN NILAI *SIMILARITY* M21 PADA EKSPERIMEN KETIGA

ID	Chapter_1 (*cos θ*)	Chapter_2 (*cos θ*)	Chapter_3 (*cos θ*)	Chapter_4 (*cos θ*)	Chapter_5 (*cos θ*)	Chapter_ Conclusion (*cos θ*)	*Full Text*
M01	0.296902	0.517097	0.397307	0.463084	0.489654	0.211099	0.497558
M02	0.397208	0.383622	0.466059	0.313865	0.563937	0.349812	0.558220
M03	0.645714	0.596102	0.584886	0.561744	0.519489	0.317286	0.703851
M04	0.641469	0.547065	0.531368	0.443973	0.532278	0.563587	0.594980
M05	0.619959	0.615600	0.679673	0.651185	0.458331	0.341693	0.710297
M06	0.427636	0.221232	0.348081	0.396646	0.433211	0.273085	0.464717
M07	0.575484	0.582641	0.738928	0.396877	0.476928	0.430956	0.656291
M08	0.745367	0.602764	0.676292	0.422351	0.501450	0.286848	0.717577
M09	0.559778	0.329393	0.332032	0.363938	0.430969	0.462762	0.453017
M10	0.482115	0.456539	0.340024	0.302858	0.358758	0.341531	0.507833
M11	0.617354	0.394568	0.573586	0.583260	0.433086	0.420313	0.676971
M12	0.541416	0.469734	0.572480	0.444924	0.462734	0.397909	0.665353
M13	0.254160	0.317087	0.251418	0.367391	0.442339	0.298440	0.383876
M14	0.445585	0.365517	0.390882	0.282868	0.128879	0.225165	0.470567
M15	0.335126	0.339487	0.316190	0.211051	0.031702	0.218896	0.266147
M16	0.377274	0.224417	0.311895	0.156239	0.077097	0.240796	0.306047
M17	0.427135	0.328459	0.503703	0.270301	0.384241	0.342894	0.474143
M18	0.377507	0.237501	0.439646	0.373065	0.194359	0.498128	0.534084
M19	0.337823	0.539055	0.432921	0.215518	0.155586	0.223425	0.466251
M20	0.495619	0.430694	0.436145	0.242740	0.287950	0.290565	0.491309
M22	0.374653	0.531698	0.409689	0.212581	0.288797	0.376409	0.369555
M23	0.536267	0.320317	0.522983	0.356440	0.447233	0.462335	0.615058
M24	0.471225	0.269697	0.509806	0.306861	0.441107	0.480101	0.581085

TABEL VIII
JUMLAH DARI TOTAL PERBANDINGAN PER BAB DENGAN *FULL TEXT*

Eskperimen	Chapter_1	Chapter_2	Chapter_3	Chapter_4	Chapter_5	Chapter_ Conclusion
1	11	6	5	0	10	7
2	5	2	5	2	1	0
3	7	4	5	0	2	2
Total	23	12	15	2	13	9

TABEL IX
RATA-RATA NILAI *SIMILARITY* PER BAB DENGAN *FULL TEXT*

Eskperimen	Chapter_1 (*cos θ*)	Chapter_2 (*cos θ*)	Chapter_3 (*cos θ*)	Chapter_4 (*cos θ*)	Chapter_5 (*cos θ*)	Chapter_ Conclusion (*cos θ*)	*Full Text* (*cos θ*)
1	0.456665	0.412068	0.424172	0.347381	0.447765	0.385998	0.488834
2	0.442841	0.335417	0.477090	0.359833	0.373366	0.270875	0.522777
3	0.477512	0.418273	0.468087	0.362598	0.371309	0.350176	0.528904
Rata-rata	0.398968						0.513505

TABEL X
URUTAN KATA TERBANYAK DALAM SETIAP BAB

No	Chapter_1	Chapter_2	Chapter_3	Chapter_4	Chapter_5	Chapter_ Conclusion
1	data	data	data	data	data	aplikasi
2	bab	sistem	gambar	gambar	menampilkan	saran
3	aplikasi	universitas	tabel	halaman	hasil	data
4	sistem	"nama universitas"	menampilkan	tampilan	tabel	simpulan
5	barang	proses	barang	barang	pengujian	sistem

Setelah nilai *similarity* dirata-ratakan menjadi seperti pada Tabel IX, dapat disimpulkan bahwa nilai *similarity* pada setiap bab lebih kecil dari nilai *similarity full text*. Oleh karena itu, penggunaan metode perbandingan per bab dapat menghasilkan nilai *similarity* yang lebih kecil dari *full text*.

Tabel VIII menunjukkan jumlah dari total perbandingan nilai *similarity* setiap bab yang lebih besar dengan nilai *similarity full text*. Nilai Chapter_4 memiliki total paling kecil dan memiliki perbedaan yang signifikan dengan nilai bab yang lain. Setelah ditelusuri dari data teks, Chapter_4 banyak mengandung kode program baik secara tertulis atau gambar. Perhitungan tingkat *similarity* kode program perlu dilakukan penelitian lebih lanjut. Penggunaan *stopword* dan *stemming* dalam penelitian ini dalam berbahasa Indonesia sedangkan kode program biasanya dibuat dalam Bahasa Inggris.

Untuk mencari tahu karakteristik setiap bab, dilakukan *mapping* kamus kata dengan jumlah teks. Berikut pada Tabel IX merupakan hasil *mapping* dengan 5 urutan kata terbanyak pada setiap bab.

Berdasarkan Tabel X, kata "data" dapat dianggap dengan kemunculan kata terbanyak pada laporan Tugas Akhir. Karena data laporan ini diambil dari sebuah universitas yang sama dengan program studi teknologi informasi, kata "data" menjadi karakteristik pada program studi tersebut.

Berikut merupakan analisis karakteristik dari setiap bab:

1) Chapter_1 dengan judul "Pendahuluan" memiliki karakteristik dalam pembahasan rencana pembuatan laporan. Kata "bab" menunjukan penjelasan bab pada laporan tersebut sehingga menjadi karakteristik Chapter_1.
2) Chapter_2 dengan judul "Kajian Teori" memiliki karakteristik dalam membahas teori yang akan digunakan. Namun, kemunculan kata "universitas" dan "nama universitas" tidak menunjukan karakteristik pada Chapter_2. Setelah penelusuran data, penggunaan algoritma OCR dapat membaca format laporan yang dibuat universitas seperti penulisan nama universitas pada Header atau Footer.
3) Chapter_3 dengan judul "Analisis dan Rancangan" memiliki karakteristik memiliki banyak penjelasan pada tabel, diagram, dan gambar. Kata "gambar" dan "tabel" menunjukan karakteristik pada bab tersebut.
4) Chapter_4 dengan judul "Implementasi" memiliki karakteristik dalam penjelasan pembuatan sebuah sistem. Setelah penelusuran data, kata "halaman" menunjukan karakteristik pada Chapter_4. Kata tersebut sering digunakan untuk menampilkan contoh halaman UI.
5) Chapter_5 dengan judul "Pengujian" memiliki karakteristik dalam penjelasan hasil penelitian. Kata "menampilkan", "hasil", "tabel", dan "pengujian" menunjukan karakteristik bab tersebut. Penggunaan kata "tabel" sering digunakan untuk penjelasan dalam tabel di laporan.
6) Chapter_Conclusion dengan judul "Kesimpulan dan Saran" memiliki karakteristik dalam penjelasan kesimpulan hasil penelitian dan saran pengembangan lebih lanjut. Kata "saran" dan "simpulan" menjadi karakteristik bab tersebut.

Penggunaan *stopword* dari *library* Sastrawi dibuat berdasarkan kata umum dari Bahasa Indonesia. Kata yang menjadi karakteristik dari sebuah bab di laporan dapat berpotensi menjadi *stopword*. Karena kemunculan kata karakteristik pada sebuah bab yang terlalu banyak dan memiliki *value* yang kecil terhadap konteks bab tersebut, dapat mengganggu hasil dari *String Similarity*. Oleh karena itu, untuk penyempurnaan penelitian ini, diperlukan analisis lebih lanjut terhadap kata karakteristik yang dapat menjadi *stopword* pada perbandingan *similarity* setiap bab.

VI. DISKUSI

Dengan penggunaan metode per bab pada perbandingan nilai *similarity* dengan *preprocessing* dapat meneliti karakteristik dari setiap bab. Berdasarkan Gambar 7, penggunaan format / *template* pada sebuah bab dapat mempengaruhi nilai *similarity*.

Chapter_4 di laporan Tugas Akhir program studi Teknologi Informasi sering berisi kode program. Karena *library* yang digunakan untuk Bahasa Indonesia dan metode perbandingan yang dilakukan fokusnya pada teks dalam sebuah bab, nilai sebuah tabel, diagram, kode program, dan gambar dengan teks biasa disamakan nilainya. Untuk penelitian lebih lanjut, dapat melakukan klasifikasi pada kode program lalu melakukan integrasi dengan sistem deteksi plagiarisme pada kode program seperti yang dibuat

oleh Sulistiani dan Karnalim [8].

Kegunaan dari sebuah *Stopword Removal* yaitu membuang kata umum dari sebuah teks. Kata umum yang dibuat oleh *library* Sastrawi ditujukan untuk penggunaan secara teks Bahasa Indonesia yang umum. Jika lebih dirinci, kata yang sering muncul pada program studi Teknologi Informasi dapat berpotensi menjadi *Stopword* seperti pada Tabel X.

VII. KESIMPULAN DAN SARAN

A. Kesimpulan

Metode *preprocessing Stemming* dan *Stopword Removal* dapat menghasilkan nilai sebaran hasil *similarity* tertinggi pada algoritma *Cosine Similarity*. Hasil tersebut sesuai dengan hipotesis pertama, yaitu metode *Preprocessing Stopword + Stemming* menghasilkan sebaran nilai *similarity* (*cos θ)* paling tinggi daripada metode tanpa *Preprocessing*, *Stopword*, dan *Stemming*. Urutan penggunaan metode *preprocessing Stemming* dan *Stopword Removal* tidak menghasilkan nilai perbedaan yang signifikan.

Hipotesis kedua yang dibuat pada penelitian ini, yaitu menggunakan metode perbandingan bab menghasilkan tingkat *similarity* lebih besar daripada *full text*, tidak sesuai dengan hasil yang didapatkan. Berdasarkan Tabel IX, nilai *similarity* pada per bab dapat menghasilkan nilai yang lebih kecil dari *full text*, sehingga nilai *similarity* per bab dapat menjadi pertimbangan dalam penentuan plagiarisme. Walaupun hipotesis kedua tidak sesuai, penggunaan metode perbandingan per bab dapat menghasilkan bahan rekomendasi yang lebih banyak daripada perbandingan *full text*.

Teks dari sebuah laporan Tugas Akhir dari program studi Teknologi Informasi dapat menunjukan karakteristik tertentu. Pada program studi Teknologi Informasi, kata "data" menjadi karakteristik pada program studi tersebut. Karakteristik pada setiap bab pada laporan Tugas Akhir program studi Teknologi Informasi dapat dilihat pada Tabel X.

B. Saran

Berdasarkan penelitian yang telah dilakukan, terdapat beberapa saran untuk penelitian selanjutnya:

1) Dikarenakan format data laporan berupa pdf, teknik OCR untuk membuat pdf menjadi teks, tidak memiliki akurasi sebesar 100%. Direkomendasikan menggunakan data berbentuk teks untuk menghasilkan hasil *similarity* yang lebih akurat.
2) Sistem yang dibuat pada penelitian ini hanya membandingkan teks saja, dapat dibuat deteksi kemiripan yang dapat mengklasifikasi gambar, tabel, kode program, dan diagram. Untuk kode program dapat dilakukan pengecekan similaritas seperti pada penelitian yang telah dibuat oleh Sulistiani dan Karnalim [8].
3) Kata yang menjadi karakteristik suatu bab pada laporan dapat menjadi bahan pertimbangan untuk menjadi *stopword* dalam perbandingan antar bab. Kata tersebut perlu dianalisis lebih lanjut mengenai potensi menjadi *stopword*.

DAFTAR PUSTAKA

[1] H. A. Maurer, F. Kappe & B. Zaka, "Plagiarism-A-suervey," *J. UCS*, vol. 12, pp. 1050-1084, Jan. 2006.
[2] S.D. Blum, *My Word!: Plagiarism and College Culture*, Cornell University Press, 2009.
[3] J. Liddel, "A Comprehensive Definition of Plagiarism," *Community & Junior College*, vol. 11, pp. 43-52, Mar. 2003.
[4] B. Leonard & S. Hansun, "Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 5, pp. 462-471, Feb. 2017.
[5] N. Kang, A. Gelbukh & S. Han, "PPChecker: Plagiarism Pattern Checkerin Document Copy Detection," *International Conference on Text, Speech and Dialogue*, 2006, vol. 4188, pp. 661-667.
[6] Z. A. Al-Khanjari, J. A. Fiaidhi, R. A. Al-Hinai and N. S. Kutti, "PlagDetect: A Java Programming Plagiarism Detection Plug-in," *ACM Inroads*, vol. 1, pp. 66-71, Des. 2010.
[7] W. G. S. Parwita, I. G. A. A. D. Indradewi & I. N. S. W. Wijaya, "String Matching based Plagiarism Detection for Document in Bahasa Indonesia," *2019 5th International Conference on New Media Studies (CONMEDIA)*, 2019, pp. 54-58.
[8] L. Sulistiani & O. Karnalim, "ES-Plag: Efficient and sensitive source code plagiarism detection," Computer Applications in Engineering Education, vol. 27, pp. 166-182, Sept. 2018.
[9] L. Graham-Matheson & S. Starr, "Is it cheating or learning the craft of writing? Using Turnitin to helpstudents avoid plagiarism," *Research in Learning Technology*, vol. 21, pp. 1-13, Apr. 2013.
[10] (2000) The Turnitin website. [Online]. Tersedia: http://www.turnitin.org/
[11] T. Fudge, (2018) Percentage Threshold for Determining Plagiarism. [Online]. Tersedia: https://scalar.usc.edu/works/c2c-digital-magazine-fall-2018--winter-2019/should-there-be-percentage-threshold-determining-plagiarism-academic-offense
[12] C. D. Manning, P. Raghavan & H. Schütze, "The term vocabulary and postings lists," *Introduction to information retrieval*, Cambridge University Press, pp. 19-47, 2008.
[13] J. B. Lovins, "Development of a *stemming* algorithm," *Mech. Transl. Comput. Linguistics*, vol. 11, no. 1-2, pp. 22-31, 1968.
[14] F. Tala, "A study of *stemming* effects on information retrieval in Bahasa Indonesia," M.Sc thesis, Institute for Logic, Language and Computation, Armsterdam, Belanda, 2003.
[15] C. D. Manning, P. Raghavan & H. Schutze, "Scoring, Term Weighting and the vector space model," *Introduction to information retrieval*, Cambridge University Press, pp. 109-134, 2008.
[16] J. Han, M. Kamber & J. Pei, *Data mining : concept and techniques*, Elsevier, 2011.
[17] T. Mardiana, T. B. Adji & I. Hidayah, "*Stemming* Influence on *Similarity* Detection of Abstract," *TELKOMNIKA*, vol. 14, no. 1, pp. 219-227, 2016.
[18] M. Lee. (2001) Pytesseract PyPI. [Online]. Tersedia: https://pypi.org/project/pytesseract/.
[19] Scikit-learn. (2007) Sklearn cosine_*similarity* documentation. [Online]. Tersedia: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_*similarity*.html.
[20] Scikit-learn. (2007) Sklearn CountVectorizer documentation. [Online]. Tersedia: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.
[21] NLTK Project. (2001) Natural Language Toolkit - NLTK 3.5 documentation [Online]. Tersedia: https://www.nltk.org/.

[22] A. Librian. (2013) High quality stemmer library for Indonesian Language Bahasa [Online]. Tersedia: https://github.com/sastrawi/sastrawi.
[23] M. Adriani, J. Asian, B. Nazief, S. Tahaghohi & H. E. Williams, "*Stemming* Indonesian: A confix-stripping approach.," *ACM Transactions on Asian Languages Information Processing*, vol. 6, no. 4, 2007.
[24] Scikit-learn. (2007) Feature_extraction documentation. [Online]. Tersedia: https://github.com/scikit-learn/scikit-learn/blob/0fb307bf3/sklearn/feature_extraction/text.py#L1168.
[25] Scikit-learn. (2007) Metric pairwise documentation. [Online]. Tersedia: https://github.com/scikit-learn/scikit-learn/blob/0fb307bf3/sklearn/metrics/pairwise.py#L1144.