JNATIA Volume 1, Nomor 1, November 2022
Jurnal Nasional Teknologi Informasi dan Aplikasinya

# Deteksi Plagiarisme Source Code Tugas Mahasiswa Menggunakan Algoritma *Cosine Similarity* dan Pembobotan TF-IDF

I Gede Ariawan Eka Putra[a1] , I Wayan Supriana[a2]

[a]Program Studi Informatika, Universitas Udayana
Bali, Indonesia
[1]gede.aryawhan@gmail.com
[2]wayan.supriana@unud.ac.id

***Abstract***

This study discusses the detection of plagiarism Source Code student assignments. Detection of the similarity of a program is needed which often arises problems related to plagiarism. This problem often occurs in programmer assignments which are often found in student source code. With the problems that arise, the researchers aim to design a system that can detect source code plagiarism in student assignments. The system is designed using the Term Frequency — Inverse Document Frequency (TF-IDF) weighting method for word weighting and the Cosine Similarity Algorithm to calculate the similarity between documents. The stages are designed to achieve the final results of this research which starts from collecting data obtained from the github.com site. The next stage is Preprocessing which aims to correct and prevent errors in the operation of the algorithm. The weight calculation stage uses the Term Frequency — Inverse Document Frequency (TF-IDF) method and the last one uses the Cosine Similarity algorithm to find similarities between the documents being compared. The results of the study are the scores of each document compared, such as the 4th document with the 7th document, the plagiarism score of 75.28% is obtained, assuming that if the score is above 75%, the document experiences plagiarism between the two documents being compared.

***Keywords:*** *Cosine Similarity, Plagiarisme, TF-IDF, Source Code, student, documents*

## 1. Pendahuluan

Dalam lingkungan informatika, pendeteksian kemiripan suatu program sangat dibutuhkan. Permasalahan yang sering timbul di lingkungan akademis yang berkaitan dengan informatika yaitu terjadinya plagiarisme terhadap tugas-tugas pemrograman. Plagiarisme itu sendiri adalah meniru topik, mengambil hasil penelitian, serta merangkum karya orang lain dengan tidak mencantumkan sumbernya. Tindakan ini sering ditemukan dalam pengerjaan tugas mahasiswa, salah satunya pada tugas praktikum algoritma yang sering ditemui dalam *Source Code* pengerjaan tugas mahasiswa. *Source Code* merupakan sekumpulan deklarasi atau pernyataan pada Bahasa pemrograman yang dapat dibaca serta ditulis manusia. Dengan adanya *Source Code, programmer* dapat berkomunikasi dengan komputer menggunakan beberapa perintah yang telah terdefinisi.

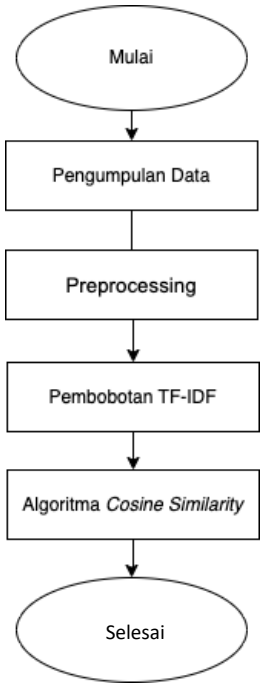
Adapun penelitian terkait yang dilakukan oleh Rito dan dkk pada tahun 2019, penelitian tersebut membahas tentang Deteksi Plagiarisme pada Dokumen Jurnal Menggunakan Metode Cosine Similarity yang dimana penelitian tersebut menggunakan algoritma Cosine Similarity untuk menentukan presentase nilai kemiripan antar dokumen. Berdasarkan skenario yang telah di uji coba dilakukan dengan menghitung jumlah dokumen relevan terambil dibagi dengan jumlah dokumen yang ada dalam database kemudian dikali 100%, maka diperoleh nilai recall pada Aplikasi Deteksi Plagiarisme Menggunakan Metode *Cosine Similarity* yaitu 13%. Sedangkan untuk memperoleh nilai precision dilakukan skenario pengujian dengan menghitung jumlah dokumen relevan terambil dibagi

dengan jumlah dokumen relevan dalam pencarian kemudian dikali 100% diperoleh hasil 8%. Maka dari penelitian yang sudah dilakukan tersebut, algoritma *Cosine Similarity* dapat digunakan dalam pendeteksi plagiarisme antar dua dokumen yang dibandingkan [1].

Dengan adanya permasalahan yang sudah disampaikan, maka dari itu peneliti menulis jurnal dengan judul "Deteksi Plagiarisme Source Code Tugas Mahasiswa Menggunakan Algoritma *Cosine Similarity* dan Pembobotan TF-IDF". Penelitian ini nantinya bertujuan untuk mengetahui tingkat *score* kemiripan antar dokumen dengan menggunakan algoritma *Cosine Similarity* dan pembobotan term *frequency-inverse document frequency* (TF-IDF).

## 2. Metode Penelitian

Penelitian ini akan memakai data *source code* mahasiswa yang diperoleh dari situs github.com. Selanjutnya, data tersebut akan dicari tingkat *score* plagiarisme antar dokumen tugas mahasiswa. Adapun tahapan dari pelaksanaan penelitian ini yang dapat dilihat pada Gambar 1.



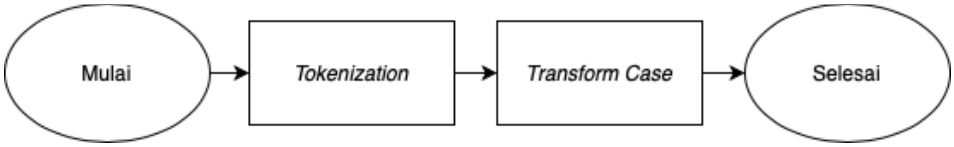

**Gambar 1.** Diagram Alir Penelitian

Penelitian ini dimulai dengan Pengumpulan Data yang diperoleh dari situs github.com, yaitu pada soal *shift-sisop*. Data yang digunakan berjumlah 10 *Source Code* yang nantinya akan digunakan untuk pendeteksian plagiarisme. Selanjutnya, data yang berjumlah 10 *Source Code* akan memasuki tahap *preprocessing* dengan metode *Tokenization* dan *Transform case*. Setelah data berhasil pada tahap *preprocessing* maka selanjutnya, dilakukan pembobotan dari setiap data *Source Code* dengan menggunakan metode term *Frequency-Inverse Document Frequency* (TF-IDF). Tahap terakhir adalah menghitung kemiripan antara data *Source Code* tugas mahasiswa dengan menggunakan algoritma *Cosine similarity*.

## 3. Hasil dan Pembahasan

### 3.1 *Preprocessing*

Tahap *Preprocessing* merupakan tahap pertama yang sangat penting dilakukan untuk memperbaiki kinerja dan mencegah kesalahan di dalam pengoperasian algoritma. Adapun proses yang akan dilakukan dalam *preprocessing*, yaitu *Tokenization* dan *Transform Case*. Berikut adalah alur dari tahapan *preprocessing* yang dapat dilihat pada Gambar 2.




**Gambar 2.** Diagram Alir *Preprocessing*

a. *Tokenization*

Pada tahap *Tokenization*, dilakukan proses pemenggalan yang akan dibagi menjadi beberapa bagian. Hal ini dilakukan untuk memudahkan di dalam proses pembobotan dari setiap kata. Berikut adalah hasil dari pemenggalan salah satu data *Source Code* mahasiswa pada Tabel 1.

**Tabel 1.** Hasil *Tokenization*

| Source Code | Hasil *Tokenization* |
|---|---|
| #!/bin/bash<br><br># 1. A<br>regex='(?<=ERROR )(.*)(?= )'<br>regex2='(?<=\().+?(?=\))'<br><br># 1. B<br>error_message() {<br>grep -oP "$regex" syslog.log \| sort \| uniq -c \| sort -nr \| sed 's/^ *\([0-9]*\) *\(.*\)/\2,\1/' >> error_message.csv<br>}<br><br># 1. C<br>user_statistic() {<br>grep -oP "$regex2" syslog.log \| sort \| uniq \| while read i<br>do<br>echo "$i" \| tr '\n' ','<br>grep "$i" syslog.log \| grep "INFO" \| wc -l \| tr '\n' ','<br>grep "$i" syslog.log \| grep "ERROR" \| wc -l<br>done >> user_statistic.csv<br>}<br><br># 1. D<br>echo "Error,Count" > error_message.csv<br>error_message<br><br># 1. F<br>echo "Username,INFO,ERROR" > user_statistic.csv<br>user_statistic | bash<br>bin<br>Count<br>csv<br>do<br>done<br>echo<br>ERROR<br>error_message<br>log<br>nr<br>oP<br>read<br>regex<br>regex2<br>sed<br>sort<br>syslog<br>tr<br>uniq<br>user_statistic<br>Username<br>wc<br>while |

b. *Transform Case*

Pada tahap *preprocessing*, selanjutnya adalah *Transform Case*. Hasil dari pemenggalan akan dilakukan penyetaraan karakter, yaitu mengubahnya menjadi huruf kecil. Berikut adalah hasil dari *Transform Case*.

**Tabel 2.** Hasil *Transform Case*

| *Tokenization* | Hasil *Transform Case* |
|---|---|
| bash | bash |

| | |
|---|---|
| bin | bin |
| **Count** | **count** |
| csv | csv |
| do | do |
| done | done |
| echo | echo |
| **ERROR** | **error** |
| error_message | error_message |
| log | log |
| nr | nr |
| **oP** | **op** |
| read | read |
| regex | regex |
| regex2 | regex2 |
| sed | sed |
| sort | sort |
| syslog | syslog |
| tr | tr |
| uniq | uniq |
| user_statistic | user_statistic |
| **Username** | **username** |
| wc | wc |
| while | while |

3.2 *Term Frequency — Inverse Document Frequency* (TF-IDF)

Tahap selanjutnya adalah menghitung bobot dari setiap dokumen yang nantinya akan dilakukan perhitungan algoritma *Cosine Similarity*. Untuk mencari bobot pada setiap dokumen dapat dilakukan dengan memakai algoritma *Term Frequency — Inverse Document Frequency* (TF-IDF). Tahap pertama adalah melakukan perhitungan *Inverse Document Frequency* (IDF) dengan menggunakan rumus di bawah ini.

$$\log\left(\frac{n}{df}\right) + 1$$

Keterangan. :
n = Jumlah dokumen
t = Jumlah frekuensi kata terpilih

Setelah melakukan perhitungan *Inverse Document Frequency* (IDF) tahap selanjutnya adalah melakukan perhitungan untuk mencari nilai dari *Term Frequency — Inverse Document Frequency* (TF-IDF). Pada tahap ini akan menggunakan rumus seperti di bawah ini.

$$W_{df} = tf_{dt} * IDF_t$$

Keterangan :
d = Dokumen ke-d
t. = Kata ke-t dari kata kunci
w = Bobot dokumen ke-d terhadap kata ke-t

Pada Tabel 3 merupakan nilai dari perhitungan akhir Term *Frequency — Inverse Document Frequency* (TF-IDF).

**Tabel 3. Hasil** Perhitungan TF-IDF antar dua dokumen

| No | Kata | D1 | D2 |
|---|---|---|---|
| 1 | 1a | 0 | 0.029191416565474984 |
| 2 | 1b | 0 | 0.029191416565474984 |
| 3 | 1c | 0 | 0.029191416565474984 |
| 4 | 1d | 0 | 0.029191416565474984 |
| 5 | 1e | 0 | 0.029191416565474984 |
| 6 | add | 0 | 0.08757424969642494 |
| 7 | bash | 0.06991457568388007 | 0.0207699333103817 |
| 8 | bin | 0.06991457568388007 | 0.0207699333103817 |
| 9 | cat | 0 | 0.029191416565474984 |
| 10 | closed | 0 | 0.08757424969642494 |
| 11 | closedinfo | 0 | 0.08757424969642494 |
| 12 | closing | 0 | 0.05838283313094997 |
| 13 | connect | 0 | 0.029191416565474984 |
| 14 | connectinfo | 0 | 0.08757424969642494 |
| 15 | connection | 0 | 0.05838283313094997 |
| 16 | count | 0.06991457568388007 | 0.0207699333103817 |
| 17 | csv | 0.2796583027355203 | 0.3115489996557255 |
| 18 | cut | 0 | 0.2043399159583249 |
| 19 | db | 0 | 0.08757424969642494 |
| 20 | denied | 0 | 0.08757424969642494 |
| 21 | denyinfo | 0 | 0.08757424969642494 |
| 22 | do | 0.06991457568388007 | 0.0207699333103817 |
| 23 | doesn | 0 | 0.029191416565474984 |
| 24 | done | 0.06991457568388007 | 0.0207699333103817 |
| 25 | echo | 0.2097437270516402 | 0.3323189329661072 |
| 26 | error | 0.2796583027355203 | 0.10384966655190851 |
| 27 | error_message | 0.2796583027355203 | 0.14538953317267192 |
| 28 | exist | 0 | 0.05838283313094997 |
| 29 | existinfo | 0 | 0.08757424969642494 |
| 30 | f1 | 0 | 0.08757424969642494 |
| 31 | f2 | 0 | 0.11676566626189994 |
| 32 | failed | 0 | 0.08757424969642494 |
| 33 | file | 0 | 0.029191416565474984 |
| 34 | found | 0 | 0.029191416565474984 |
| 35 | grep | 0.4194874541032804 | 0.2284692664141987 |
| 36 | info | 0.13982915136776014 | 0.0830797332415268 |
| 37 | information | 0 | 0.17514849939284988 |
| 38 | log | 0.2796583027355203 | 0.29077906634534384 |
| 39 | modified | 0 | 0.08757424969642494 |

| | | | |
|---|---|---|---|
| 40 | modinfo | 0 | 0.08757424969642494 |
| 41 | name | 0 | 0.05838283313094997 |
| 42 | not | 0 | 0.029191416565474984 |
| 43 | nr | 0.09826249667188093 | 0 |
| 44 | op | 0.19652499334376186 | 0 |
| 45 | permission | 0 | 0.08757424969642494 |
| 46 | read | 0.06991457568388007 | 0.0207699333103817 |
| 47 | regex | 0.19652499334376186 | 0 |
| 48 | regex2 | 0.19652499334376186 | 0 |
| 49 | report | 0 | 0.2043399159583249 |
| 50 | retrieving | 0 | 0.08757424969642494 |
| 51 | sed | 0.06991457568388007 | 0.0207699333103817 |
| 52 | sort | 0.2097437270516402 | 0.062309799931145105 |
| 53 | syslog | 0.2796583027355203 | 0.24923919972458042 |
| 54 | the | 0 | 0.08757424969642494 |
| 55 | ticket | 0 | 0.26272274908927484 |
| 56 | timeinfo | 0 | 0.08757424969642494 |
| 57 | timeout | 0 | 0.08757424969642494 |
| 58 | to | 0 | 0.2919141656547498 |
| 59 | totalerr | 0 | 0.029191416565474984 |
| 60 | totalinf | 0 | 0.029191416565474984 |
| 61 | toterror | 0 | 0.05838283313094997 |
| 62 | totinfo | 0 | 0.05838283313094997 |
| 63 | tr | 0.19652499334376186 | 0 |
| 64 | tried | 0 | 0.08757424969642494 |
| 65 | uniq | 0.13982915136776014 | 0.062309799931145105 |
| 66 | updating | 0 | 0.08757424969642494 |
| 67 | user | 0 | 0.11676566626189994 |
| 68 | user_statistic | 0.2796583027355203 | 0.0207699333103817 |
| 69 | username | 0.06991457568388007 | 0.0207699333103817 |
| 70 | was | 0 | 0.08757424969642494 |
| 71 | wc | 0.13982915136776014 | 0.1661594664830536 |
| 72 | while | 0.06991457568388007 | 0.1869293997934353 |

3.3 Algoritma *Cosine Similarity*

Setelah melakukan perhitungan terhadap nilai bobot dari *Term Frequency — Inverse Document Frequency* (TF-IDF). Tahap terakhir adalah menghitung nilai total dari masing-masing dokumen dan dokumen pembanding lalu mengurutkan nilai dari total jumlah dokumen tersebut dengan menggunakan rumus di bawah ini.

$$\boldsymbol{Cos\ a} = \frac{\boldsymbol{A}\ .\ \boldsymbol{B}}{|\boldsymbol{A}||\boldsymbol{B}|} = \frac{\sum_{i=1}^{n} \boldsymbol{A_i}\ \boldsymbol{x}\ \boldsymbol{B}}{\sum_{i=1}^{n} (\boldsymbol{A_i})^2\ \boldsymbol{x}\ \sqrt{\sum_{i=1}^{n} (\boldsymbol{B_i})^2}}$$

Setelah menerapkan metode yang dipakai untuk penelitian ini, maka hasil pengujian terhadap plagiarisme dari tugas *Source Code* mahasiswa memakai algoritma *Cosine Similarity* dan Pembobotan Term *Frequency — Inverse Document Frequency* (TF-IDF), dengan demikian *score* plagiarisme yang didapat dari 10 dokumen yang dibandingkan adalah sebagai berikut. Pada tabel 4 terdapat hasil perbandingan dokumen ke-4 dengan dokumen ke-7 maka diperoleh hasil *score* plagiarisme sebesar 75.28% dengan asumsikan jika *score* diatas 75% dokumen tersebut mengalamin tindakan plagiarisme antara dua dokumen yang dibandingkan.

**Tabel 4.** Hasil *Score Similarity*

| % | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **D1** | 100 | 49.79 | 41.95 | 37.80 | 29.87 | 38.94 | 33.83 | 46.49 | 45.42 | 40.07 |
| **D2** | | 100 | 50.28 | 58.66 | 51.26 | 41.99 | 46.83 | 47.46 | 61.99 | 42.03 |
| **D3** | | | 100 | 65.57 | 40.85 | 47.25 | 61.74 | 59.06 | 50.12 | 48.19 |
| **D4** | | | | 100 | 56.05 | 45.27 | 75.28 | 55.21 | 53.37 | 56.07 |
| **D5** | | | | | 100 | 38.73 | 50.50 | 35.24 | 56.28 | 37.95 |
| **D6** | | | | | | 100 | 41.18 | 38.69 | 51.97 | 33.23 |
| **D7** | | | | | | | 100 | 48.17 | 50.26 | 51.84 |
| **D8** | | | | | | | | 100 | 43.27 | 48.36 |
| **D9** | | | | | | | | | 100 | 32.84 |
| **D10** | | | | | | | | | | 100 |

## 4. Kesimpulan

Dengan adanya hasil pembahasan dan penelitian mengenai “Deteksi Plagiarisme *Source Code* Tugas Mahasiswa Menggunakan Algoritma *Cosine Similarity* dan Pembobotan TF-IDF”. Dengan memakai alur penelitian seperti *preprocessing* untuk memperbaiki kinerja dan mencegah kesalahan di dalam pengoperasian algoritma, perhitungan *Term Frequency — Inverse Document Frequency* (TF-IDF) dengan menghitung bobot dari setiap dokumen serta Algoritma *Cosine Similarity* dalam mencari kemiripan antar dokumen. Dengan demikian dapat ditarik kesimpulan bahwa hasil dari penelitian menggunakan metode tersebut diperoleh plagiarisme berupa *score* kesamaan antar tugas mahasiswa seperti perbandingan dokumen ke-4 dengan dokumen ke-7 diperoleh hasil *score* plagiarisme sebesar 75.28% dengan asumsikan jika *score* diatas 75% dokumen tersebut mengalamin tindakan plagiarisme antara dua dokumen yang dibandingkan. Dengan adanya penelitian ini diharapkan mampu membantu dalam pendeteksian plagiarisme tugas mahasiswa yang berada pada ruang lingkup *Source Code.*

## Referensi

[1] Rito Putriwana Pratama, Muhammad Faisal dan Ajib Hanani, “Deteksi Plagiarisme pada Artikel Jurnal Menggunakan Metode Cosine Similarity”, SMARTICS Journal, vol. 5, No. 1 2019. p22-26, 22, 2019.

[2] Daniel dan Wilda Susanti, “Penerapan Algoritma Cosine Similarity Pada Sistem Pengajuan Judul Tugas Akhir Berbasis Web”, Seminar Nasional Informatika (SENATIKA), vol.-, no.11 , 2021.

[3] Imam Nawawi, Putra Prima Arhandi dan Faisal Rahutomo, “Deteksi Plagiarisme Pada Dokumen Skripsi Berdasarkan Tingkat Kesamaan Dengan Menggunakan Metode Longest Common Subsequence”, vol. 8, no. 3, 2019.

[4] Rio Alamanda, Cucu Suhery dan Yulrio Brianorman, “Aplikasi Pendeteksi Plagiat Terhadap Karya Tulis Berbasis Web Menggunakan Natural Language Processing Dan Algoritma Knuth-Morris-Pratt”, Jurnal Coding, vol. 04. No.1, 2006.

[5] Jurnal Coding, Sam Farisa Chaerul Haviana dan Andika Novianto, “Implementasi Algoritma Cosine Similarity pada sistem arsip dokumen di Universitas Islam Sultan Agung”, Transformtika, vol. 17, No.2, 2020.