Evaluasi Keandalan *Cosine Similarity* dalam Mendeteksi Plagiarisme Kode Program

Andi Farhan Hidayat (13523128)[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523128@std.stei.itb.ac.id, [2]andifarhan1094@gmail.com

***Abstract*— Plagiarisme kode program adalah masalah yang sering dihadapi dalam dunia pemrograman, yang dapat merugikan baik pengembang maupun institusi pendidikan. Untuk mendeteksi plagiarisme, berbagai metode telah dikembangkan, salah satunya adalah metode cosine similarity. Metode ini mengukur kesamaan antara dua vektor berdasarkan sudut antara keduanya, yang digunakan untuk membandingkan kode program. Penelitian ini bertujuan untuk mengevaluasi keandalan cosine similarity dalam mendeteksi plagiarisme kode program, dengan fokus pada berbagai bentuk plagiarisme, seperti plagiarisme yang melibatkan reorganisasi kode, pengubahan variabel, dan plagiarisme langsung. Hasil percobaan menunjukkan bahwa *cosine similarity* efektif dalam mendeteksi plagiarisme, terutama pada kode program yang telah direorganisasi. Namun, metode ini memiliki beberapa keterbatasan, antara lain ketergantungan pada panjang kode program dan kesulitan dalam membedakan kode yang benar-benar berbeda yang menggunakan sintaksis serupa. Meskipun demikian, jika kode program yang dibandingkan cukup panjang, metode ini dapat mendeteksi plagiarisme akibat pengubahan variabel dengan lebih akurat. Penelitian ini juga menyarankan penggabungan *cosine similarity* dengan metode lain untuk meningkatkan akurasi deteksi plagiarisme, terutama pada kode dengan perbedaan minor.**

***Keywords*—Cosine Similarity, Plagiarism.**

I. PENDAHULUAN

Plagiarisme pada kode program merupakan tantangan yang sering dihadapi di dunia pendidikan maupun industri perangkat lunak. Praktik plagiarisme tidak hanya merugikan pengembang dan akademisi yang telah berusaha keras menciptakan kode program, tetapi dalam konteks pendidikan juga dapat menghambat perkembangan kemampuan pelaku plagiarisme dalam memahami dan menyelesaikan masalah pemrograman secara mandiri. Aksi plagiarisme kode program yang merugikan ini tentu perlu untuk dicegah. Salah satu cara untuk mencegah plagiarisme adalah dengan mengetahui terlebih dahulu aksi plagiarisme yang dilakukan sehingga dapat dilakukan aksi preventif lanjutan.

Salah satu metode umum untuk mendeteksi plagiarisme adalah *cosine similarity*. Metode *cosine similarity* mengukur kesamaan dari dua vektor dengan cara membandingkan representasi vektornya pada ruang *inner product* [1]. Metode tersebut umumnya digunakan untuk mengukur kesamaan antara dua dokumen [1]. Dalam konteks kode program, metode ini memiliki kemungkinan untuk digunakan dalam mendeteksi plagiarisme, yaitu dengan cara menemukan kesamaan sintaksis antara dua atau lebih program.

Namun, efektivitas *cosine similarity* dalam mendeteksi plagiarisme kode program masih perlu diuji lebih lanjut. Kode program memiliki karakteristik unik pada struktur, keterkaitan antar baris kode, dan variasi sintaksis yang tidak selalu memengaruhi hasil akhirnya. Oleh karena itu, diperlukan penelitian mendalam untuk mengetahui sejauh mana *cosine similarity* dapat diandalkan sebagai metode deteksi plagiarisme pada kode program.

Makalah ini bertujuan untuk mengevaluasi keandalan *cosine similarity* dalam mendeteksi plagiarisme kode program. Melalui eksperimen dan analisis, diharapkan dapat diperoleh pemahaman yang lebih baik tentang kelebihan dan keterbatasan metode ini, serta rekomendasi untuk pengembangan alat deteksi plagiarisme kode program di masa depan.

II. DASAR TEORI

A. Kode Program

Kode program atau kode sumber adalah aspek penting dalam membangun program komputer, yaitu berupa serangkaian teks kode yang membentuk instruksi atau pernyataan dan ditulis dalam bahasa pemrograman oleh *programmer* [2]. Kode program dibuat dan didesain agar dapat dimengerti oleh pengembang. Kode sumber dibangun dengan bahasa pemrograman tertentu. Setiap bahasa pemrograman memiliki sintaksis dan struktur yang unik, umumnya kode program terdiri dari elemen-elemen seperti variabel, fungsi, metode, kontrol alur, komentar, dan komponen operasional lainnya. Terdapat beragam bahasa pemrograman yang dapat membangun kode program, yaitu Python, Java, C, PHP, Rust, dan masih banyak lagi. Walaupun dengan bentuk dan pola yang variatif, kode program dapat ditulis dalam berbagai cara

yang berbeda untuk mencapai hasil yang sama, sehingga memungkinkan variasi tanpa mengubah keluaran program.

B. Plagiarisme

Menurut Kamus Besar Bahasa Indonesia (KBBI), plagiarisme adalah kegiatan penjiplakan yang melanggar hak cipta [3]. Plagiarisme juga didefinisikan sebagai penyalahgunaan karya orang lain dengan menyalin atau mengambil ide tanpa memberikan atribusi yang tepat [4]. Kata plagiarisme berasal dari bahasa latin "plagiare" yang artinya mencuri. Menurut penulis Ajib Rosidi yang dikutip oleh Teuku Kemal Fasya, plagiarisme adalah apabila seorang ilmuwan atau seniman memberikan kepada masyarakat pengetahuan atau suatu karya seni yang berkaitan dengan seluruh atau sebagian besar karya orang lain, tanpa menyebutkan nama pencipta yang diambil karyanya [5]. Dalam konteks kode program, plagiarisme mencakup penggunaan materi audio atau visual orang lain, materi tes, perangkat lunak, dan kode program tanpa mengungkapkan sumbernya, serta menampilkannya seolah-olah sebagai karya sendiri [6]. Dalam praktiknya, terdapat 4 tipe plagiarisme, yaitu plagiarisme atas sumber (*Plagiarism of Source*), Plagiarisme Kepengarangan (*Plagiarism of Authorship*), Plagiarisme Ide (*Plagiarism of Idea*), dan Plagiarisme Terjemahan (*Plagiarism of Translation*) [5].

C. Vektor dan Ruang Vektor

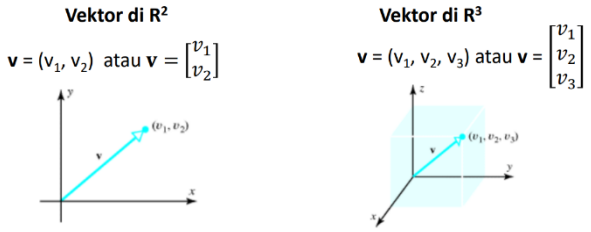
1. Vektor

Vektor adalah representasi data yang memiliki nilai dan arah [7]. Penulisan vektor dilambangkan dengan huruf-huruf kecil yang dicetak tebal atau memakai tanda panah. Dalam geometri, vektor digambarkan sebagai garis berpanah dalam ruang dua dimensi [7]. Vektor sering digunakan untuk menggambarkan berbagai konsep dalam fisika dan matematika.

Gambar 2.1. Vektor di ruang 2D [7]

2. Ruang Vektor

Ruang vektor (*vector space*) atau ruang Euclidean merupakan ruang letak vektor berada atau didefinisikan. Ruang vektor dituliskan sebagai huruf R kapital dengan pangkat sebagai nilai dari dimensi ruang vektor, contoh R^3 artinya merupakan ruang vektor berdimensi tiga, sedangkan R^2 merupakan ruang vektor berdimensi dua [7]. Berikut adalah contoh vektor di R^2 dan R^3.



Gambar 2.2. Vektor di Ruang R^2 dan R^3 [7]

3. Operasi Vektor

Vektor dapat melakukan berbagai jenis perhitungan matematis. Perhitungan matematis dasar dalam vektor meliputi penjumlahan, pengurangan, perkalian skalar, perkalian titik (*dot product*), perkalian silang (*cross product*), panjang vektor, vektor unit, dan proyeksi vektor.

Salah satu perhitungan matematis vektor yang penting dalam *Cosine Similarity* adalah perkalian titik (*dot product*). Perkalian titik adalah operasi aljabar yang mengambil dua vektor dari ruang vektor (Euclidean) dan menghasilkan sebuah skalar. Operasi *dot product* dari dua vektor A dan B didefinisikan sebagai berikut:

$$A \cdot B = \|A\|\|B\| \cos\theta$$

dengan $\|A\|$ dan $\|B\|$ adalah panjang dari vektor A dan B, serta θ adalah sudut antara kedua vektor tersebut. Dalam menghitung panjang vektor, terdapat bentuk perhitungan yang berbeda sesuai dengan jumlah dimensi dari ruang vektor tersebut. Berikut definisi menghitung panjang vektor **v** pada ruang vektor R^n:

$$\|\boldsymbol{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

dengan $v_1, v_2, \ldots, v_n$ adalah komponen-komponen vektor **v** di tiap sumbunya.

Selain itu, perkalian titik pada $\boldsymbol{A} = (a_1, a_2, \ldots, a_n)$ dan $\boldsymbol{B} = (b_1, b_2, \ldots, b_n)$ juga dapat didefinisikan sebagai berikut.

$$\boldsymbol{A} \cdot \boldsymbol{B} = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n$$

Dalam definisi perkalian titik tersebut, tidak digunakan definisi perkalian dengan kosinus sudut antar kedua vektor.

D. Cosine Similarity

Cosine similarity adalah sebuah metrik yang digunakan untuk mengukur seberapa mirip dua vektor dalam ruang Euclidean. Rumus ini sering digunakan dalam berbagai aplikasi seperti pencarian informasi, pengolahan bahasa alami (NLP), pembelajaran mesin, dan pendeteksi plagiarisme. *Cosine similarity* mengukur sudut antara dua vektor dengan tujuan untuk menentukan kesamaan arah antara kedua vektor tersebut tanpa memperhatikan panjangnya. Secara formal, cosine similarity antara dua vektor $\boldsymbol{A} = (a_1, a_2, \ldots, a_n)$ dan $\boldsymbol{B} = (b_1, b_2, \ldots, b_n)$ dihitung dengan menggunakan rumus berikut:

$$sim(\boldsymbol{A}, \boldsymbol{B}) = \cos\theta = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|}$$

dengan $\boldsymbol{A} \cdot \boldsymbol{B}$ adalah perkalian titik vektor **A** dan **B**, serta $\|\boldsymbol{A}\|$ dan $\|\boldsymbol{B}\|$ adalah panjang dari vektor **A** dan **B** [8]. Nilai dari rumus *Cosine similarity* berkisar antara -1 dan 1. Dengan nilai 1 menunjukkan bahwa kedua vektor memiliki arah yang identik, nilai 0 menunjukkan bahwa kedua

vektor tidak memiliki kesamaan arah, dan nilai -1 menunjukkan bahwa kedua vektor memiliki arah yang berlawanan.

III. METODOLOGI DAN PENGUJIAN

A. Rancangan Rencana Pengujian

Dalam menguji keandalan *cosine similarity* dalam mendeteksi plagiarisme dalam kode program yang meliputi plagiarisme dalam bentuk langsung ataupun secara reorganisasi kode, perlu terdapat rancangan pengujian dengan dataset yang berbeda-beda jenisnya untuk mengetahui secara menyeluruh keandalan dari *cosine similarity*. Oleh sebab itu, dalam pengujian ini akan digunakan tiga jenis data uji yang berbeda, yaitu:

- Kode asli: Kode program yang tidak terpengaruh oleh plagiarisme.
- Kode plagiarisme: Kode program yang disalin atau dimodifikasi dari kode asli dengan beberapa variasi, seperti perubahan nama variabel, reorganisasi baris, atau penyamaran kode.
- Kode independen: Kode program yang tidak terkait dengan kode lain, baik itu kode asli maupun kode yang telah disalin dari tempat lain. Kode ini tidak memiliki kemiripan atau hubungan dengan kode lain yang diuji untuk plagiarisme.

Ketiga jenis kode program tersebut akan diuji satu dengan lainnya untuk mengukur keandalan dari *cosine similarity* dalam mendeteksi plagiarisme kode program.

Kemudian, terdapat langkah-langkah pengujian sehingga didapatkan hasil yang dapat mengukur keandalan *cosine similarity*. Berikut adalah langkah-langkah pengujian tersebut:

a. *Preprocessing* Kode: Kode uji ditokenisasi, yaitu mengubah kode program menjadi unit-unit yang lebih kecil dan terstruktur sehingga mempermudah perbandingan antar kode. Tokenisasi kode uji akan menghasilkan kode program dalam bentuk representasi vektor.
b. Perhitungan *Cosine Similarity*: Menghitung *cosine similarity* antara tiap-tiap jenis kode uji untuk mengidentifikasi tingkat kemiripan.
c. Variasi Pengujian: Perhitungan *cosine similarity* diuji pada kode plagiarisme yang lebih variatif, yaitu:
   - Kode yang langsung disalin.
   - Kode salinan dengan hanya mengubah nama variabelnya.
   - Kode salinan yang telah mengalami reorganisasi baris atau struktur logika.

   Selain variasi untuk jenis kode plagiarisme, akan dilakukan pula variasi pasangan pengujian untuk mendapatkan data yang lebih menyeluruh dan dapat dipastikan. Variasi pasangan tersebut sebagai berikut:
   - Kode asli dengan kode asli.
   - Kode asli dengan kode kode independen.
   - Kode asli dengan kode plagiarisme hasil salinan dengan hanya mengubah variabelnya.
   - Kode asli dengan kode plagiarisme hasil salinan dengan reorganisasi.
d. Evaluasi hasil: Membandingkan hasil *cosine similarity* dan mengevaluasi tingkat deteksi plagiarisme serta akurasinya.

Dengan langkah-langkah dan data pasangan uji tersebut, akan didapatkan hasil pengujian yang cukup luas dengan adanya bentuk variasi kode uji untuk dievaluasi.

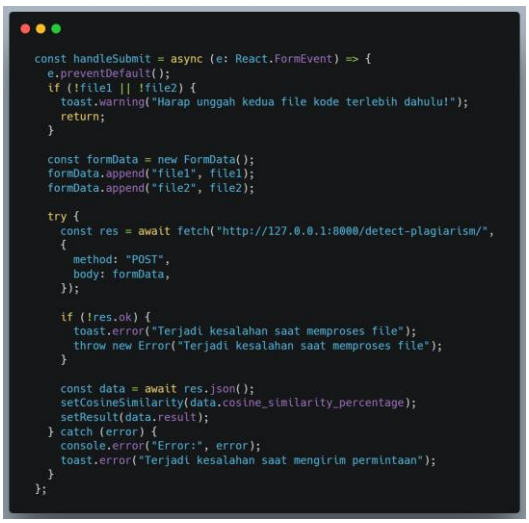
B. Implementasi Kode

a. Teknologi Implementasi

Dalam melakukan implementasi pengujian keandalan *cosine similarity* untuk mendeteksi plagiarisme kode program, berbagai teknologi bantuan digunakan. Pada *front-end* digunakan teknologi Next.Js dengan *App Router* dalam bahasa Typescript, Axios untuk komunikasi dengan endpoint API (Application Programming Interface) *back-end* yang melakukan pemrosesan data, dan teknologi tampilan lainnya seperti DaisyUI serta sonner. Kemudian, dalam pemrosesan data yaitu pada *back-end* digunakan teknologi Fastapi untuk menghasilkan API *endpoint* yang dapat dengan mudah dipanggil oleh *front-end*. Lalu terdapat pula NumPy (Numerical Python) untuk perhitungan numerik dan manipulasi array yang tentu memudahkan perhitungan perkalian dot serta perhitungan panjang vektor yang dibutuhkan dalam operasi *cosine similarity*. Selain itu, terdapat teknologi re (Regular Expression) yang digunakan untuk pencocokan pola pada teks kode program yang meliputi pencarian dan penggantian string sehingga dapat membantu dalam langkah *preprocessing* kode dengan mempersiapkan atau membersihkan data teks kode program untuk mendapatkan hasil *cosine similarity* yang lebih akurat.

b. Implementasi *Preprocessing* Kode

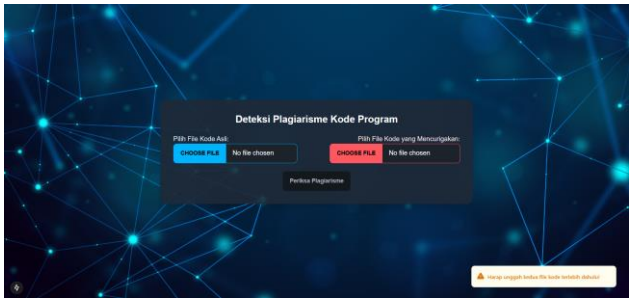
Preprocessing teks kode program dilakukan dalam berbagai langkah, pertama masukan *file* kode program divalidasi terlebih dahulu sehingga memastikan bahwa data yang diproses benar-benar ada. Validasi *file* kode program ini dilakukan pada sisi *front-end*, yaitu pada komponen *DetectPlagiarism* saat formulir akan di-*submit*.



```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  if (!file1 || !file2) {
    toast.warning("Harap unggah kedua file kode terlebih dahulu!");
    return;
  }

  const formData = new FormData();
  formData.append("file1", file1);
  formData.append("file2", file2);

  try {
    const res = await fetch("http://127.0.0.1:8000/detect-plagiarism/",
    {
      method: "POST",
      body: formData,
    });

    if (!res.ok) {
      toast.error("Terjadi kesalahan saat memproses file");
      throw new Error("Terjadi kesalahan saat memproses file");
    }

    const data = await res.json();
    setCosineSimilarity(data.cosine_similarity_percentage);
    setResult(data.result);
  } catch (error) {
    console.error("Error:", error);
    toast.error("Terjadi kesalahan saat mengirim permintaan");
  }
};
```

Gambar 3.1. Validasi *file* kode program saat di-*submit*.

(Sumber: Dokumen Pribadi Peneliti)



Gambar 3.2. Peringatan muncul karena *file* kosong.
(Sumber: Dokumen Pribadi Peneliti)

Selanjutnya, *file* kode program yang sudah divalidasi akan dikirimkan ke sisi *back-end* untuk dilakukan *preprocessing* lanjutan, yaitu penghapusan teks komentar dan baris kosong pada kode program. *Preprocessing* ini menggunakan teknologi re (Regular Expression) dan dieksekusi oleh fungsi *preprocess_code*.

```
def preprocess_code(code):
    """
    Fungsi untuk melakukan preprocessing pada kode program.
    - Menghapus komentar
    - Menghapus baris kosong
    """
    # Menghapus komentar
    code = re.sub(r'#.*', '', code)

    # Menghapus komentar blok
    code = re.sub(r'"""[\s\S]*?"""', '', code)

    # Menghapus baris kosong
    code = "\n".join([s for s in code.splitlines() if s.strip()])

    return code
```

Gambar 3.3. Fungsi *preprocess_code*.
(Sumber: Dokumen Pribadi Peneliti)

Terakhir, pada fungsi *tokenize_code* data *file* kode program yang tadi telah diproses akan ditokenisasi.

```
def tokenize_code(code):
    """
    Fungsi untuk melakukan tokenisasi pada kode program.
    """
    # Split berdasarkan spasi dan karakter khusus
    tokens = re.findall(r'\w+|[^\s\w]', code)
    return tokens
```

Gambar 3.4. Fungsi *tokenize_code*.
(Sumber: Dokumen Pribadi Peneliti)

Dengan begitu, kode program yang awalnya berbentuk blok teks akan diubah menjadi array dengan tiap elemen merupakan kata, simbol, ataupun huruf dari kode program tersebut. Sebagai contoh, kode program “def add(a, b): return a + b” akan diubah menjadi ['def', 'add', '(', 'a', ',', 'b', ')', ':', 'return', 'a', '+', 'b']. Tokenisasi ini bermanfaat untuk memudahkan perhitungan *cosine similarity* di tahap berikutnya.

c. Implementasi Perhitungan *Cosine Similarity*

Perhitungan *cosine similarity* diimplementasikan pada fungsi *calculate_cosine_similarity* dan *cosine_similarity*. Pada *calculate_cosine_similarity*, masukan kode program akan dipecah menjadi token menggunakan fungsi *tokenize_code*. Lalu, frekuensi tiap token dalam kode program masukan dihitung menggunakan Counter. Selanjutnya, dibuat daftar semua token unik dari kedua kode program masukan. Terakhir, hasil daftar semua token unik tersebut dimanfaatkan oleh vektor hasil operasi Counter untuk vektor baru, yaitu vektor frekuensi token untuk kemudian diproses nilai *cosine similarity*-nya oleh fungsi *cosine_similarity*.

```
def calculate_cosine_similarity(code1, code2):
    """
    Fungsi untuk menghitung cosine similarity antara dua kode program.
    """
    tokens1 = tokenize_code(code1)
    tokens2 = tokenize_code(code2)

    # Menghitung frekuensi token
    counter1 = Counter(tokens1)
    counter2 = Counter(tokens2)

    # Gabungan semua token unik
    all_tokens = list(set(counter1.keys()).union(set(counter2.keys())))

    # Membuat vektor frekuensi token
    vec1 = np.array([counter1[token] for token in all_tokens])
    vec2 = np.array([counter2[token] for token in all_tokens])

    return cosine_similarity(vec1, vec2)
```

Gambar 3.5. Fungsi *calculate_cosine_similarity*.
(Sumber: Dokumen Pribadi Peneliti)

Sebagai contoh, terdapat dua kode uji:
- code1 = “def add(a, b): return a + b”
- code2 = “def add_numbers(x, y): return x + y”

Pada fungsi *tokenize_code*, kedua kode uji tersebut akan menjadi:
- code1 = ['def', 'add', '(', 'a', ',', 'b', ')', ':', 'return', 'a', '+', 'b']
- code2 = ['def', 'add_numbers', '(', 'x', ',', 'y', ')', ':', 'return', 'x', '+', 'y']

Kemudian, fungsi Counter akan menghasilkan dua vektor yang disertai data frekuensi tiap token pada kode1 dan kode2, berturut-turut vektor baru tersebut adalah counter1 dan counter2:
- counter1 = {'def': 1, 'add': 1, '(': 1, 'a': 2, ',': 1, 'b': 2, ')': 1, ':': 1, 'return': 1, '+': 1}
- counter2 = {'def': 1, 'add_numbers': 1, '(': 1, 'x': 2, ',': 1, 'y': 2, ')': 1, ':': 1, 'return': 1, '+': 1}

Selanjutnya, daftar token unik dibuat dari kedua vektor counter1 dan counter2.
- all_tokens = ['a', 'b', 'x', 'y', 'return', 'add_numbers', ')', 'add', '(', ':', '+', 'def', ',']

Terakhir, dua vektor baru dibuat dengan all_tokens, counter1, dan counter2 sebagai vektor frekuensi token, yaitu:
- vec1 = [2, 2, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1]
- vec2 = [0, 0, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1, 1]

Kedua vektor tersebut akan digunakan untuk menghitung nilai *cosine similarity*.

Pada implementasi *cosine_similarity*, vektor-vektor hasil pemrosesan fungsi *calculate_cosine_similarity* dihitung nilai *cosine similarity*-nya dengan menghitung nilai dot

kedua vektor dan panjang masing-masing vektor.

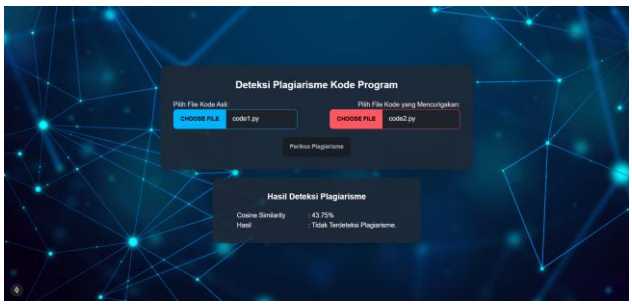
```python
def cosine_similarity(vec1, vec2):
    """
    Fungsi untuk menghitung cosine similarity antara dua vektor.
    """
    dot_product = np.dot(vec1, vec2)
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2)
    return dot_product / (norm_vec1 * norm_vec2)
```

Gambar 3.6. Fungsi *cosine_similarity*.
(Sumber: Dokumen Pribadi Peneliti)

Jika menggunakan contoh hasil kode pada fungsi sebelumnya, dot product dari:

- vec1 = [2, 2, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1]
- vec2 = [0, 0, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1, 1]

adalah 7. Kemudian, panjang vektor vec1 dan vec2 adalah sama, yaitu 4. Dengan begitu, *cosine similarity* dari kedua vektor tersebut adalah 43,75%. Hasil tersebut sesuai dengan hasil ketika dilakukan percobaan dengan sistem yang telah dibuat.



Gambar 3.7. Hasil uji contoh code1 dan code2.
(Sumber: Dokumen Pribadi Peneliti)

d. Integrasi Program Utama

Semua pemrosesan dan fungsi-fungsi sebelumnya khususnya pada sisi *back-end* akan diintegrasikan pada program utama, yaitu pada fungsi *detect_plagiarism*. Fungsi tersebut meliputi integrasi pengambilan *file* kode program masukan, *preprocess_code*, *calculate_cosine_similarity*, perubahan nilai *cosine similarity* menjadi persen, identifikasi plagiarisme berdasarkan nilai *cosine similarity* dengan *threshold* 80%, dan integrasi *endpoint* API untuk mengirimkan hasil perhitungan *cosine similarity* ke sisi *front-end*.

```python
@app.post("/detect-plagiarism/")
async def detect_plagiarism(file1: UploadFile = File(...), file2: UploadFile = File(...)):
    try:
        code1 = (await file1.read()).decode("utf-8")
        code2 = (await file2.read()).decode("utf-8")

        original_code_preprocessed = preprocess_code(code1)
        suspected_code_preprocessed = preprocess_code(code2)

        cosine_sim = calculate_cosine_similarity(original_code_preprocessed, suspected_code_preprocessed)
        percentage = round(cosine_sim, 4) * 100
        threshold = 0.8

        if cosine_sim > threshold:
            result = "Terdeteksi Plagiarisme"
        else:
            result = "Tidak Terdeteksi Plagiarisme."

        return {"cosine_similarity_percentage": percentage, "result": result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Gambar 3.8. Fungsi *detect_plagiarism*.
(Sumber: Dokumen Pribadi Peneliti)

C. Pengujian

a. Kode Asli dengan Kode Asli

1. Berukuran kecil
   Kode asli: a1.py
   Cosine similarity = 100%

2. Berukuran besar
   Kode asli: a2.py
   Cosine similarity = 100%

b. Kode asli dengan kode kode independen

1. Percobaan 1:
   Kode asli: b1-asli.py
   Kode independent: b1-independen.py
   Cosine similarity = 89.13%
2. Percobaan 2:
   Kode asli: b2-asli.py
   Kode independent: b2-independen.py
   Cosine similarity = 86.96%
3. Percobaan 3:
   Kode asli: b3-asli.py
   Kode independent: b3-independen.py
   Cosine similarity = 88.46%
4. Percobaan 4:
   Kode asli: b4-asli.py
   Kode independent: b4-independen.py
   Cosine similarity = 81.11%
5. Percobaan 5:
   Kode asli: b5-asli.py
   Kode independent: b5-independen.py
   Cosine similarity = 87.84%
6. Percobaan 6:
   Kode asli: b6-asli.py
   Kode independent: b6-independen.py
   Cosine similarity = 79.66%
7. Percobaan 7:
   Kode asli: b7-asli.py
   Kode independent: b7-independen.py
   Cosine similarity = 87.98%
8. Percobaan 8:
   Kode asli: b8-asli.py
   Kode independent: b8-independen.py
   Cosine similarity = 87.67%
9. Percobaan 9:
   Kode asli: b9-asli.py
   Kode independent: b9-independen.py
   Cosine similarity = 88.3%
10. Percobaan 10:
    Kode asli: b10-asli.py
    Kode independent: b10-independen.py
    Cosine similarity = 64.60%

c. Kode asli dengan kode plagiarisme hasil salinan dengan hanya mengubah variabelnya

1. Percobaan 1
   Kode asli: c1-asli.py
   Kode plagiarisme: c1-plagiarisme.py
   Cosine similarity = 50%

2. Percobaan 2
   Kode asli: c2-asli.py
   Kode plagiarisme: c2-plagiarisme.py
   Cosine similarity = 50%
3. Percobaan 3
   Kode asli: c3-asli.py
   Kode plagiarisme: c3-plagiarisme.py
   Cosine similarity = 80%
4. Percobaan 4
   Kode asli: c4-asli.py
   Kode plagiarisme: c4-plagiarisme.py
   Cosine similarity = 84.91%
5. Percobaan 5
   Kode asli: c5-asli.py
   Kode plagiarisme: c5-plagiarisme.py
   Cosine similarity = 93.49%
6. Percobaan 6
   Kode asli: c6-asli.py
   Kode plagiarisme: c6-plagiarisme.py
   Cosine similarity = 93.5%

d. Kode asli dengan kode plagiarisme hasil salinan dengan reorganisasi
   1. Percobaan 1
      Kode asli: d1-asli.py
      Kode plagiarisme: d1-plagiarisme.py
      Cosine similarity = 86.6%
   2. Percobaan 2
      Kode asli: d2-asli.py
      Kode plagiarisme: d2-plagiarisme.py
      Cosine similarity = 83.21%
   3. Percobaan 3
      Kode asli: d3-asli.py
      Kode plagiarisme: d3-plagiarisme.py
      Cosine similarity = 85.75%
   4. Percobaan 4
      Kode asli: d4-asli.py
      Kode plagiarisme: d4-plagiarisme.py
      Cosine similarity = 89.41%
   5. Percobaan 5 Kode asli: d5-asli.py
      Kode plagiarisme: d5-plagiarisme.py
      Cosine similarity = 92.30%
   6. Percobaan 6
      Kode asli: d6-asli.py
      Kode plagiarisme: d6-plagiarisme.py
      Cosine similarity = 97.08%

IV. HASIL DAN PEMBAHASAN

A. Hasil

a. Perbandingan Kode Asli – Kode Asli
   Hasil *cosine similarity* yang didapatkan pada kode pendek maupun panjang memiliki nilai yang sama, yaitu 100%.

b. Perbandingan Kode Asli – Kode Independen
   Hasil *cosine similarity* yang didapatkan bervariasi tetapi memiliki kecenderungan untuk bernilai di atas 80%. Rata-rata persentase *cosine similarity* yang didapatkan adalah 84.17%.

c. Perbandingan Kode Asli – Kode Plagiarisme Ubah Variabel
   Hasil *cosine similarity* yang didapatkan sangat bervariasi. Rata-rata persentase *cosine similarity* yang didapatkan adalah 75.32%. Pada percobaan 1-3 persentasenya adalah 60% dan pada percobaan 4-6 persentasenya adalah 90.63%.

d. Perbandingan Kode Asli – Kode Plagiarisme Reorganisasi
   Hasil *cosine similarity* yang didapatkan cenderung tinggi. Rata-rata persentase *cosine similarity* yang didapatkan adalah 89.06%. Pada percobaan 1-3 persentasenya adalah 85.19% dan pada percobaa 4-6 persentasenya adalah 92.93%.

B. Pembahasan

a. Kode Asli – Kode Asli
   Nilai yang didapatkan merupakan nilai yang sudah benar, yaitu 100% karena membandingkan dua kode program yang persis sama. Hal ini menunjukkan bahwa sistem yang dibuat telah bekerja sesuai dengan kenyataan masukan yang ada.

b. Kode Asli – Kode Independen
   Rata-rata nilai *cosine similarity* yang cukup tinggi, yaitu 84.17% menunjukkan bahwa terdapat ketidakandalan pada penggunaan *cosine similarity* dalam mendeteksi plagiarisme kode program. Nilai *cosine similarity* antara kode asli dan kode independen seharusnya memiliki nilai yang rendah sebab memiliki fungsi dan tujuan yang saling lepas. Nilai *cosine similarity* yang cukup besar ini dapat dipengaruhi oleh beberapa hal, seperti sintaksis pada bahasa pemrograman yang umumnya sering berulang dan digunakan pada kode yang bermacam-macam sehingga menyebabkan kurang andalnya *cosine similarity* dalam meninjau kode yang bukan plagiarisme. Selain itu, panjang kode uji juga cukup berpengaruh. Pada percobaan 1-9 digunakan kode uji pendek dan menghasilkan *cosine similarity* yang besar. Sedangkan pada percobaan 10 digunakan kode uji yang lebih panjnag dan hasilnya didapatkan *cosine similarity* yang lebih kecil.

c. Kode Asli – Kode Plagiarisme Ubah Variabel
   Nilai rata-rata *cosine similarity* yang didapatkan kurang sesuai, yaitu 75.32%. Nilai yang didapatkan seharusnya lebih tinggi sebab pada percobaan ini membandingkan kode asli dengan kode hasil plagiarisme. Hal ini dapat disebabkan oleh berbagai faktor seperti perbedaan panjang kode uji. Pada percobaan 1-3 digunakan kode uji yang lebih pendek dibandingkan dengan kode uji pada percobaan 4-6. Hasilnya, nilai *cosine similarity* pada percobaan 1-3 cukup rendah yaitu 60%, dan pada

percobaan 4-6 lebih tinggi yaitu 90.63%. Hal ini menunjukkan bahwa *cosine similarity* sudah cukup andal untuk mendeteksi plagiarisme ubah variabel pada kode yang panjang, tetapi masih belum andal dalam mendeteksi plagiarisme ubah variabel pada kode yang pendek.

d. Perbandingan Kode Asli – Kode Plagiarisme Reorganisasi
Rata-rata *cosine similarity* yang didapatkan cukup besar, yaitu 89.06%. Nilai tersebut telah sesuai karena pada perbandingan kode asli dengan kode plagiarisme nilai plagiarismenya harus tinggi untuk dapat mengetahui hasil pekerjaan plagiarisme. Sama seperti percobaan-percobaan sebelumnya, percobaan 1-3 dilakukan dengan kode yang lebih pendek dengan percobaan 4-6. Walaupun demikian, rata-rata *cosine similarity* yang didapatkan tetap cukup tinggi baik pada percobaan 1-3, yaitu 85.19% maupun pada percobaan 4-6, yaitu 92.93%. Hal ini menunjukkan bahwa metode *cosine similarity* cocok digunakan untuk mendeteksi plagiarisme pada kode hasil reorganisasi. Walau demikian, tetap terlihat bahwa metode *cosine similarity* lebih akurat pada kode program yang lebih panjang.

V. KESIMPULAN DAN SARAN

Penggunaan metode *cosine similarity* dalam mendeteksi plagiarisme kode program memiliki kelebihan dan kekurangan. *Cosine similarity* telah menunjukkan keandalannya dalam mendeteksi plagiarisme yang cukup baik pada kode program hasil plagiarisme dengan reorganisasi. Akan tetapi, *cosine similarity* memiliki beberapa kekurangan seperti bergantung pada panjang kode program dan kesulitan mengidentifikasi dua program yang benar-benar berbeda karena menggunakan sintaksis yang sama. Walau demikian, jika kode program yang dibandingkan cukup panjang, plagiarisme dalam bentuk pengubahan variabel pun dapat dideteksi dengan lebih akurat. Selain itu, dalam membedakan dua program yang benar-benar berbeda juga dapat dideteksi dengan baik oleh *cosine similarity* jika kode program keduanya cukup panjang.

Peneliti memiliki beberapa saran dalam keberlanjutan evaluasi ini, yaitu sebagai berikut:

1. Data percobaan yang dilakukan untuk penelitian lanjutan agar lebih banyak, variatif, dan mempertimbangkan faktor-faktor yang dapat mempengaruhi bias perhitungan.
2. Penggunaan metode yang lebih baik dalam *preprocessing* data program kode sehingga yang dibandingkan benar-benar yang dibutuhkan dan bukanlah kode yang terus berulang karena sintaksis.
3. Kombinasi penggunaan metode *cosine similarity* dengan metode lainnnya untuk mendapatkan hasil pembacaan plagiarisme yang lebih baik.

VI. LAMPIRAN

Kode program implementasi dan data uji percobaan pada makalah ini dapat diakses sepenuhnya pada *link repository* berikut: https://github.com/andi-frame/Code-Plagiarism-Checker

VII. UCAPAN TERIMA KASIH

Peneliti ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, peneliti dapat menyelesaikan makalah ini dengan baik. Peneliti juga ingin mengucapkan terima kasih kepada Arrival Dwi Sentosa, S.Kom., M.T. dan Dr. Judhi Santoso, M.Sc. selaku dosen IF2123 Aljabar Linear dan Geometri yang telah memberikan bimbingan, saran, dan masukan selama masa pembelajaran yang sangat berharga sehingga mendukung proses penulisan makalah ini. Dukungan dan pengetahuan yang diberikan sangat membantu dalam menyelesaikan makalah ini. Terima kasih yang sebesar-besarnya peneliti sampaikan kepada orang tua dan keluarga peneliti yang selalu memberikan dukungan moral dan materiil serta doa yang tiada henti. Tidak lupa, peneliti juga ingin mengucapkan terima kasih kepada teman-teman dan rekan-rekan seperjuangan yang selalu memberikan semangat dan bantuan baik secara langsung maupun tidak langsung dalam penyusunan makalah ini. Akhir kata, peneliti menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat peneliti harapkan untuk perbaikan di masa mendatang. Terima kasih.

REFERENSI

[1] Han, J., Kamber, M., & Pei, J. (2012). Getting to Know Your Data. Data Mining, 39–82. https://doi.org/10.1016/B978-0-12-381479-1.00002-2.

[2] Wallask, S. (2023) What is source code in programming and how does it work?, Search App Architecture. https://www.techtarget.com/searchapparchitecture/definition/source-code. (Diakses: 31 December 2024).

[3] Kamus Besar Bahasa Indonesia (KBBI). KBBI VI Daring (Dalam Jaringan). https://kbbi.kemdikbud.go.id. (Diakses: 31 December 2024).

[4] Risparyanto. (2020). Plagiarisme dan integritas akademik. https://journal.uii.ac.id/Buletin-Perpustakaan/article/download/34333/16776/113779. (Diakses: 31 December 2024).

[5] Budi, H. S. (2011). Plagiarisme: pelanggaran hak cipta dan etika. Kanisius.

[6] Fakultas Teknik Universitas Tanjungpura. (2020). Plagiarisme dalam pendidikan dan industri. Diakses dari https://pwk.teknik.untan.ac.id/files/buku/fullbook-plagiarisme-dan-integritas-akademik-compressed_1706694415.pdf.

[7] Munir, Rinaldi. 2023. "Vektor di Ruang Euclidean (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Aljaba

rGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf. (Diakses pada 31 Desember 2024).
[8] Munir, Rinaldi. 2023. “Aplikasi dot product pada information retrieval”. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf. (Diakses pada 31 Desember 2024).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Januari 2025

Andi Farhan Hidayat
13523128