Penerapan Latent Semantic Analysis (LSA) Berbasis Dekomposisi Nilai Singular untuk Pendeteksian Plagiarisme

Yusuf Faishal Listyardi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524014@std.stei.itb.ac.id, yusuffaishalals@gmail.com

***Abstract*—Pendeteksian plagiarisme saat ini perlu melampaui sekadar pengecekan kata demi kata, mengingat banyaknya manipulasi makna dalam bentuk parafrasa. Sedangkan metode yang umum digunakan tidak efektif dalam mengatasi hal ini. Karena itu Makalah ini mengkaji penggunaan LSA berbasis SVD sebagai metode untuk menemukan kemiripan semantik. Data dokumen akan diolah ke dalam matriks term-document dan matriks TF-IDF untuk menentukan banyak kata pada dokumen beserta pembobotan tiap katanya, lalu dimensinya dipangkas menggunakan SVD agar fokus pada struktur latennya. Tingkat kemiripan dokumen kemudian diukur lewat cosine similarity.Semakin besar nilai similaritas, maka semakin sama makna dari 2 dokumen, begitpun sebaliknya. Hasilnya membuktikan bahwa metode ini efektif mengenali dokumen yang secara bahasa berbeda, namun secara ide atau makna tetap identik. Namun masih memiliki beberapa kekurangan terutama pada dokumen yang banyak mengandung teks dengan kata berarti eksplisit seperti kata majas atau kata-slang.**

***Index Terms*—Latent Semantic Analysis, Singular Value Decomposition, cosine similarity, plagiarisme dokumen**

I. Pendahuluan

Plagiarisme adalah salah satu permasalahan serius dalam dunia akademik yang mengakibatkan penurunan integritas karya ilmiah, terutama dalam lingkungan kampus. Plagiarime tidak hanya terbatas pada penyalinan teks secara identik atau *copy-paste*, tetapi juga mencakup kemiripan ide, struktur kalimat, maupun parafrasa kalimat. Seringkali plagiarisme ini sulit dideteksi menggunakan metode sederhana seperti pencocokan teks sederhana. Oleh karena itu, diperlukan suatu pendekatan yang mampu mendeteksi plagiarisme ini hingga kepada kemiripan makna antar dokumen, bukan hanya sekadar kesamaan kata secara eksplisit.

Metode pendeteksian plagiarisme umumnya menggunakan teknik berbasis seperti pencocokan string atau fingerprinting. Meskipun kedua teknik ini relatif mudah untuk diimplementasikan dan efisien. Ada satu kelemahan utama dalam teknik ini, yaitu keterbatasan dalam menangkap hubungan semantik antar dokumen. Seringkali dua dokumen yang memiliki makna sama namun kosakata berbeda luput dari pendeteksian dan dikategorikan tidak plagiarisme, padahal itu juga termasuk perbuatan plagiarisme dokumen, kalau hal ini tidak diselesaikan dengan sebuah metode yang efektif, para pelaku plagiarisme akan memanfaatkan celah ini untuk meneruskan aksinya tanpa takut akan ketahuan.

LSA atau Latent Semantic Analysis adalah salah satu metode yang dapaat mengatasi keterbatasan pendeteksian kesamaan makna antar dokumen. LSA mempresentasikan dokumen dalam bentuk vektor pada ruang berdimensi tinggi. Dengan pendekatan ini hubungan latentn antara kata dan dokumen dapat dianalisis berdasarkan struktur matematisnya. Dalam penerapannya, LSA memanfaatkan konsep ruang vektor, dekomposisi matriks untuk menangkap pola semantik yang ada pada dokumen.

Makalah ini membahas penerapan Latent Semantic Analysis berbasis Dekomposisi Nilai Singular untuk pendeteksian plagiarisme dokumen. Pembahasan difokuskan pada dasar teori dari LSA seperti matriks term dokumen dan matriks TF-IDF, Dekompsisi Nilai Singular, Cosine SImilarity, dan juga pengaplikasiannya dalam code program bahasa python pada bagian hasil dan pembahasan. Dengan demikian, diharapkan makalah ini dapat menunjukkan bagaimana konsep-konsep aljabar linier dan geometri dapat diterapkan secara nyata dalam menyelesaikan permasalahan praktis di bidang analisis dokumen seperti untuk pengecekan plagriasi dokumen yang marak di era sekarang dan menjadi permasalahan serius.

II. Dasar Teori

A. Matriks

Matriks adalah sekumpulan data berupa bilangan, simbol, ekspresi yang direpresentasikan pada sebuah persegi panjang atau blok berukuran $n \times m$ yang memiliki n baris dan m kolom. Matriks memilki banyak aplikasi terutama dalam bidang aljabar linear dan geometri.

Beberapa operasi penting dalam matriks adalah :

1) Penambahan Matriks
   Penambahan matriks A dan B sebagai berikut :

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nm} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11}+B_{11} & A_{12}+B_{12} & \cdots & A_{1n}+B_{1n} \\ A_{21}+B_{21} & A_{22}+B_{22} & \cdots & A_{2n}+B_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}+B_{n1} & A_{n2}+B_{n2} & \cdots & A_{nn}+B_{nn} \end{bmatrix}$$

2) Perkalian Matrix
Misalkan ada matriks A berukuran $m \times n$ dan matriks B berukuran $n \times p$, maka hasilnya adalah matriks berukuran $m \times p$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix}$$

$$AB = \begin{bmatrix} \sum_{k=1}^{n} a_{1k}b_{k1} & \sum_{k=1}^{n} a_{1k}b_{k2} & \cdots & \sum_{k=1}^{n} a_{1k}b_{kp} \\ \sum_{k=1}^{n} a_{2k}b_{k1} & \sum_{k=1}^{n} a_{2k}b_{k2} & \cdots & \sum_{k=1}^{n} a_{2k}b_{kp} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{mk}b_{k1} & \sum_{k=1}^{n} a_{mk}b_{k2} & \cdots & \sum_{k=1}^{n} a_{mk}b_{kp} \end{bmatrix}$$

3) Transpose Matriks
Operasi pengubahan struktur matriks dengan cara menukar posisi baris menjadi kolom dan kolom menjadi baris.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

B. Nilai Eigen dan Vektor Eigen

Jika A adalah matriks $n \times n$ maka vektro tidak-nol di R^n disebut vektor eigen dari A jika Ax sama dengan perklian suaut skalar λ dengan x, yaitu

$$Ax = \lambda x \tag{1}$$

Skalar λ disebut nilai eigen dari A, dan x dinamakan vektor eigen yang berkoresponden dengan λ. Atau dengan kata lain, nilai eigen menyatakan nilai karakteristik dari sebuah matriks yang berukuran $n \times n$. Bisa diartikan juga bahwa vektor eigen x menyatakan matriks kolom yang apabila dikalikan dengan sebuah matriks $n \times n$ menghasilkan vektor lain yang merupakan kelipatan vektor itu sendiri [1].

C. Matriks Term–Dokumen

Agar bisa diproses dengan menggunakan konsep aljabar linear, maka sebuah dokumen harus kita ubah menjadi sebuah matriks, yaitu matriks terrm–dokumen. Untuk n dokumen dan m term unik, representasi seluruh dokumen dapat disusun dalam matriks term–dokumen $A \in \mathbb{R}^{m \times n}$:

$$A = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mn} \end{bmatrix}, \tag{2}$$

dengan f_{ij} menyatakan frekuensi kemunculan term ke-i pada dokumen ke-j. Matriks ini umumnya berdimensi tinggi dan bersifat jarang (sparse), sehingga diperlukan pembobotan dan reduksi dimensi agar struktur pentingnya lebih menonjol. Sebagai contoh apabila kita memiliki 3 dokumen dengan isi :

1) Dokumen 1 : ”Ayah adalah dosen aljabar linear”
2) Dokumen 2 : ”Materi aljabar adalah materi yang susah”

Maka matriks term document yang terbentuk adalah sebagai berikut :

TABLE I
CONTOH MATRIKS TERM–DOKUMEN

Term / Dokumen	Dokumen 1	Dokumen 2
ayah	1	0
adalah	1	1
dosen	1	0
aljabar	1	1
linear	1	0
materi	0	2
yang	0	1
susah	0	1

D. Pembobotan Term dengan TF–IDF

Matriks term-document hanya berisi frekuensi mentah kemunculan kata dalam dokumen. Diperlukan peningkatan kualitas representasi dari tiap kata kata di dalamnya. Untuk itu, digunakan skema pembobotan Term Frequency–Inverse Document Frequency (TF–IDF). Matriks ini bertujuan memberikan bobot lebih tinggi pada kata-kata yang lebih informatif dan menurunkan bobot kata kata yang umum.

Pertama, Term Frequency (TF) untuk term i pada dokumen j didefinisikan sebagai normalisasi frekuensi terhadap total term pada dokumen tersebut:

$$TF_{ij} = \frac{A_{ij}}{\sum_{i'=1}^{m} A_{i'j}}. \tag{3}$$

Selanjutnya, Inverse Document Frequency (IDF) untuk term i dihitung dari banyaknya dokumen yang memuat term tersebut. Misalkan df_i adalah jumlah dokumen yang mengandung term ke-i, maka:

$$IDF_i = \log_{10}\left(1 + \frac{n}{df_i}\right). \tag{4}$$

Matriks TF–IDF diperoleh dengan mengalikan TF dengan bobot IDF pada setiap baris:

$$W = \text{diag}(IDF) \cdot TF, \tag{5}$$

dengan $W \in \mathbb{R}^{m \times n}$ adalah matriks TF–IDF, dan diag(IDF) adalah matriks diagonal berukuran $m \times m$ yang elemen diagonalnya $[IDF_1, IDF_2, \ldots, IDF_m]$.

Dengan pembobotan ini, term yang informatif (spesifik) cenderung memiliki bobot lebih tinggi, sementara term yang sangat umum memiliki bobot lebih rendah. Matriks W inilah yang digunakan sebagai masukan untuk dekomposisi SVD pada LSA.

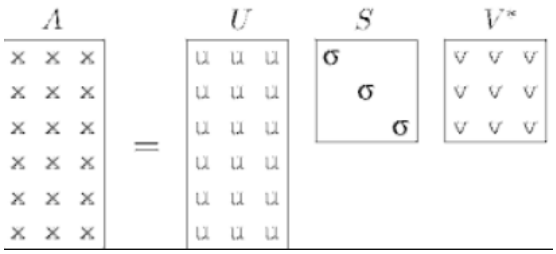
E. Dekomposisi Nilai Singular (SVD)

Dekomposisi Nilai Singular (SVD) adalah teknik faktorisasi matriks berukutan $n \times n$ yang menguraikan matriks berukuran $m \times n$ menjadi tiga matriks komponen: matriks ortogonal U, matriks diagonal Σ (berisi nilai singular), dan transpos matriks ortogonal V^T

$$W = U\Sigma V^T, \tag{6}$$

dengan matriks U berukurang $m \times m$ matriks V berukuran $n \times n$ serta matriks Σ yang berukuran $m \times n$ yang memuat nilai singular $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.

Kolom kolom pada matriks U disebut vektor-vektor singular kiri dari matriks W, sedangkan baris-baris dari matriks V disebut vektor-vektor singular kanan dari matriks W



Gambar 1. Dekomposisi Matriks dengan SVD
Sumber:www.researchgate.net

Pada aplikasi LSA, kita hanya menggunakan k topik teratas dengan singluar values terbesar, kita tidak menggunakan semua komponen dari matriks. Selain lebih efisien, komponen dengan singular value terbesar juga menangkap informasi atau variansi terbesar dalam sebuah data

$$W_k = U_k \Sigma_k V_k^T, \tag{7}$$

di mana matriks U berukuran $m \times k$, matriks V berukuran $k \times n$ serta matriks Σ yang berukuran $k \times k$

F. Latent Semantic Analysis (LSA)

Latent Semantic Analysis adalah salah satu teknik NLP yang menemukan makna tersembunyi atau semantik dalama suuatu teks dengan memanfaatkan kemunculan kata pada setiap dokumen. LSA biasa diaplikasikan pada pencocokan dokumen, pencarian rekomendasi berdasarkan teks, maupun deteksi plagiarisme

Setelah jadi *truncated SVD*, dokumen direpresentasikan dalam ruang semantik laten berdimensi k. Representasi ini diperoleh dengan mengalikan matriks V_k dengan matriks Σ_k

$$D = V_k \Sigma_k, \tag{8}$$

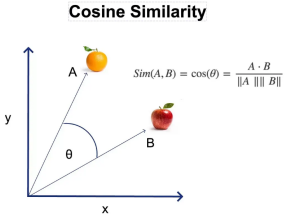
dengan matriks D berukuran $n \times k$, dan setiap d_i merepresentasikan dokumen ke-i dalam ruang semantik laten k-dimensional. Vektor d_i ini disebut sebagai *embedding* dokumen ke-i

G. Cosine Similarity

Cosine similarity merupakan metrik yang digunakan untuk menentukan tingkat kemiripan antara dua vektor non-nol berdasarkan nilai kosinus sudut yang terbentuk. Dalam bidang pembelajaran mesin dan analisis data, teknik ini menjadi instrumen krusial, khususnya untuk keperluan komparasi dokumen, sistem pencarian, serta pengembangan mesin rekomendasi. Untuk dua vektor dokumen a dan b:

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|}. \tag{9}$$

Semakin besar *cosine similarity* maka semakin dekat kemiripan laten kedua dokumen, begitupun sebaliknya. Dengan ini kita bisa menentukan seberapa mirip 2 dokumen dilihat dari maknanya. Nantinya query akan dihitung nilai similaritasnya dengan setiap dokumen pada dataset



Gambar 2. Cosine Similarity
Sumber: https://python.plainenglish.io

III. Hasil dan Pembahasan

A. Skenario Pengujian

Pengujian yang dilakukan oleh penulis adalah sebagai berikut, disediakan 10 file dataset yang masing masing berjumlah kurang lebih 1000 kata. Dokumen pada dataset diseragamkan dengan format **.txt** untuk mempermudah komputasi, ketika program berjalan, program akan load dan proses dataset. Setelah itu kita akan input query, query tersebut akan dibandingkan dengan setiap file pada dataset dengan menghitung *cosine similarity*, lalu program menampilkan nama file pada dataset dengan urutan dari *cosine similairty* terbesar. Dokumen dataset yang ditampilkan lebih dulu mendadakan dia memilki simalaritas makna paling dekat dengan file query ketimbang dokumen lain pada dataset

B. Langkah Langkah dan Code Program

Dalam pengujian LSA berbasis code program ada beberapa langkah yang harus kita lakukan :

1) Siapkan beberapa dataset dan query yang akan diuji, usahakan file-file pada dataset memiliki tema yang mirip namun tidak sama.

```
def load_txt_file(path, max_words=500):
    with open(path, "r", encoding="utf-8",
        errors="ignore") as f:
        text = f.read()
    words = text.split()
    return " ".join(words[:max_words])


def load_documents_from_folder(folder_path,
    max_words=500):
    docs, names = [], []
    files = sorted([f for f in os.listdir(
        folder_path) if f.endswith(".txt")])


    for i, fname in enumerate(files, 1):
        path = os.path.join(folder_path, fname)
        try:
            content = load_txt_file(path,
                max_words)
            wc = len(content.split())

            if content.strip() and wc > 0:
                docs.append(content)
                names.append(fname)
            else:
                print(f"   SKIP: file terbaca
                    tapi kosong setelah
                    diproses")
        except Exception as e:
            print(f"[{i}/{len(files)}] {fname}
                -> ERROR: {e}")

    return docs, names
```

2) Load dataset dengan membentuk matriks term–document untuk menghitung jumlah kata *i* pada dokumen ke *j*.

```
    def build_term_document_matrix(self):
        all_tokens = []
        doc_tokens = []

        for doc in self.documents:
            tokens = self.preprocess(doc)
            doc_tokens.append(tokens)
            all_tokens.extend(tokens)

        self.vocab = sorted(set(all_tokens))
        self.term2idx = {t: i for i, t in
            enumerate(self.vocab)}

        m, n = len(self.vocab), len(self.
            documents)
        A = np.zeros((m, n))

        for j, tokens in enumerate(doc_tokens):
            for t in tokens:
                A[self.term2idx[t], j] += 1

        self.A = A
        return A
```

3) Setelah bentuk matriks term–document, lanjutkan dengan membentuk TF–IDF untuk mengoptimalkan pembobotan tiap kata pada file

```
    def compute_tf_idf(self):
        m, n = self.A.shape
        TF = np.zeros_like(self.A)

        for j in range(n):
            s = np.sum(self.A[:, j])
            if s > 0:
                TF[:, j] = self.A[:, j] / s

        IDF = np.zeros(m)
        for i in range(m):
            df = np.count_nonzero(self.A[i, :])
            IDF[i] = math.log10(1 + n / df)

        self.IDF = IDF
        self.TFIDF = IDF[:, None] * TF
        return self.TFIDF
```

4) Dekomposisi matriks hasil pada langkah ke dua dengan dekomposisi SVD menjadi 3 bagian, ambil k top eigen terbesar saja.

```
    def l2Norm(x):
        return np.linalg.norm(x)

    def dot(u, v):
        return np.dot(u, v)

    def Svd(A, k):
        U, s, Vt = np.linalg.svd(A,
            full_matrices=False)

        kUse = min(k, len(s))
        U_k = U[:, :kUse]
        s_k = s[:kUse]
        V_k = Vt[:kUse, :].T

        return U_k, s_k, V_k
```

5) Ubah matriks menjadi representasi vektor berdimensi k pada ruang semantik laten

```
    def perform_lsa(self, k):
        U, s, V = svd.Svd(self.TFIDF, k)
        self.U = U[:, :k]
        self.s = s[:k]
        self.V = V[:, :k]

        self.doc_embed = self.V * self.s
```

6) Query yang masuk kita proses persis seperti proses pada dataset, kita tidak masukkan hasil dari proses query ke dalam matriks proses dataset, karena kita akan membandingkan matriks proses query dengan matriks proses dataset

```
    def fold_in_query(self, query_text):
        tokens = self.preprocess(query_text)
        q = np.zeros(len(self.vocab))

        for t in tokens:
            if t in self.term2idx:
                q[self.term2idx[t]] += 1

        if np.sum(q) > 0:
            q = q / np.sum(q)

        q_tfidf = q * self.IDF

        q_latent = q_tfidf @ self.U
        q_latent = np.where(self.s > 1e-12,
            q_latent / self.s, 0)
```

```
        return q_latent
```

7) Terakhir, hitung *cosine similarity* dari query denagn tiap file pada dataset

```
    def cosine_similarity(self, a, b):
        na = svd.l2Norm(a)
        nb = svd.l2Norm(b)
        if na == 0 or nb == 0:
            return 0.0
        return float(np.dot(a, b) / (na * nb))
```

C. Pengujian dan Hasil

Penulis menyiapkan 10 dataset file, lalu ada query file yang akan dihitung *cosine similarity* nya dengan setiap file pada dataset. Hasil dari code program ini yang dianalisis oleh penulis. Pada makalah akan ditunjukkan hasil dari pengujian pada 2 query dengan isi yang berbeda.

Penulis memilih menggunakan nilai k = 100, atau dalam arti lain mengambil top 100 eigen value dari matriks. Pemilihan ini didasarkan pada pertimbangan bahwa nilai tersebut sudah cukup besar untuk merepresentasikan semantik laten dari dokumen.

Penulis juga memilih membatasi mengambil kata hingga 1000 kata pertama pada dokumen, hal ini dilakukan untuk menjaga konsistensi panjang dokumen serta mengurangi pengaruh kata-kata yang kurang relevan terhadap representasi matriks. Selain itu dengan mengambil 1000 kata pertama saja akan mengoptimalkan kompleksitas algoritma karena kita tidak tahu seberapa banyak kata dalam sebuah dokumen. Kalau tidak membatasi jumlah kata, untuk *processing* dokumen dengan banyak kata akan memakan waktu yang cukup lama

Dataset terdiri dari 10 dokumen teks bertema AI, machine learning, dan deep learning. Ringkasan tiap dokumen adalah sebagai berikut:

- **doc1.txt:** membahas pengertian dan gambaran umum kecerdasan buatan.
- **doc2.txt:** membahas konsep machine learning
- **doc3.txt:** membahas deep learning dan jaringan saraf berlapis.
- **doc4.txt:** menjelaskan hubungan AI dengan machine learning dalam praktiknya.
- **doc5.txt:** peran data, prapemrosesan, dan pelatihan model pada dunia nyata.
- **doc6.txt:** membahas dasar jaringan saraf tiruan dan mekanisme *learning*.
- **doc7.txt:** mengulas contoh penerapan AI/ML dalam berbagai bidang dan sektor.
- **doc8.txt:** membahas tantangan teknis seperti data, komputasi, dan interpretabilitas.
- **doc9.txt:** membahas aspek etika penggunan AI zaman sekarang seperti privasi, bias, dan dampak sosial.
- **doc10.txt:** membahas tren dan perkembangan terbaru dalam AI.

Lalu untuk query penulis membuat query yang menyangkut 2 tema berbeda.

- **query 1** : membahas penerapan machine learning dan deep learning pada sistem kecerdasan buatan dan jaringan saraf berlapis
- **query 2** : membahas dampak sosial dan bagaimana bertika dalam menggunakan AI di era sekarang

Hasil dari pengujian ditunjukkan pada Gambar 4 untuk query 5 dan Gambar X untuk query 2 Pada tiap query akan ditampilkan

File Dataset	Similarity	Persentase
doc4.txt	0.8404	84.04%
doc3.txt	0.6446	64.46%
doc6.txt	0.4141	41.41%
doc2.txt	0.3978	39.78%
doc5.txt	0.3005	30.05%
doc1.txt	0.1832	18.32%
doc7.txt	0.1589	15.89%
doc8.txt	0.1218	12.18%
doc10.txt	0.0823	8.23%
doc9.txt	0.0351	3.51%

Gambar 3. Hasil Pengujian Query 1
Sumber: Dokumen penulis

=== HASIL KEMIRIPAN ===

File Dataset	Similarity	Persentase
doc9.txt	0.9837	98.37%
doc7.txt	0.2706	27.06%
doc1.txt	0.1870	18.70%
doc10.txt	0.1537	15.37%
doc2.txt	0.1282	12.82%
doc3.txt	0.1033	10.33%
doc5.txt	0.0859	8.59%
doc8.txt	0.0814	8.14%
doc4.txt	0.0785	7.85%
doc6.txt	0.0521	5.21%

Gambar 4. Hasil Pengujian Query 2
Sumber: Dokumen penulis

nama dataset diurutkan dari nilai similaritasnya yang paling tinggi, karena semakin besar nilai similaritas semakin selaras arah vektor semantik kedua dokumen, sehingga kita dapat mengetahui dokumen mana yang paling mirip dan paling tidak mirip berdasarkan maknanya

D. Analisis

1) Query 1: Query 1 berisi teks tentang penerapan machine learning dan deep learning khususnya pada kecerdesan artifisial dan jaringan saraf berlapis. Hasil pada Gambar 4 menunjukkan Query 1 memiliki tingkat similaritas paling tinggi pada **doc4** dan **doc3**. Yang mana kedua dokumen tersebut memang membahas aplikasi machine learning dan deeplearning pada AI dan sebagainya. Sedangkan untuk **doc9** dan **doc10** memiliki similaritas yang kecil dengan Query 1, hal itu sesuai karena **doc9** sendiri membahas tentang etika penggunaan AI, yang mana ini jauh dari isi tema Query 1.

2) Query 2: Query 2 berisi tentang etika dan dampak AI di era sekarang beserta isu isu yang sering terjadi seperti orisinalitas karya. Gambar 5 sebagai hasil pengujian Query 2 menunjukkan bahwa **doc9** memilki nilai similaritas paling tinggi dengan Query 2, ini selaras karena **doc9** membahas tentang Etika AI sama dengan Query 2. Begitupun sebaliknya untuk nilai similaritas paling rendah yaitu **doc6**, karena **doc 6** mekanisme dasar jaringan saraf, sehingga pembahasan Query 2 dengan **doc6** jelas sangat beda.

Analisis keseluruhan 2 query yang sudah penulis uji adalah LSA (Latent Semantic Analysis) mampu mengidentifikasi seberapa mirip sebuah dokumen dengan dokumen lainnya dilihat

dari sisi maknanya. Namun untuk urutan similaritas dari dataset berdasarkan query, memang ada beberapa dataset yang tidak benar urutannya, namun ini anggapan dari penulis, karena makna dari sebuah dokumen bisa berbeda beda interpretasinya berdasarkan orang orang.

Selain itu LSA sendiri memilki beberapa kelemahan yang dapat dimanfaatkan plagiarisme seperti kurang dalam mempertimbangkan makna secara eksplisit (majas, bahasa *slang*, dan lain lain), dan hanya mempertimbangkan makna asli dari dokumen. Tapi dibalik apapun itu, metode LSA ini sangat cocok untuk optimalisasi dalam pengecekan plagiarisme untuk pengecekan kesamaan makna.

IV. Kesimpulan

Berdasarkan hasil pengujian dan pembahasan yang telah dilakukan oleh penulis, dapat disimpulkan bahwa metode Latent Semantic Analysis (LSA) berbasis Dekomposisi Nilai Singular (SVD) efektif dan mampu digunakan untuk mengukur dan mengidentifikasi kemiripan semantik teks pada dokumen. Dengan merepresentasikan dokumen dalam ruang semantik laten berdimensi lebih rendah, LSA dapat menangkap hubungan makna antar dokumen yang tidak bergantung pada kesamaan kata secara eksplisit.

Berdasarkan hasil pengujian pada bagian III, terlihat bahwa dokumen yang memiliki topik mirip atau membahas hal yang mirip dengan *query* memiliki nilai *cosine similarity* yang lebih tinggi dibanding dengan dokumen yang topik pembahasannya beda dari *query*. Kita tahu bahwa semakin tinggi *cosine similarity* maka semakin mirip topik atau semakin tinggi tingkat keselarasan arah vektor semantik dari kedua dokumen. Hal ini membuktikkan bahwa metode LSA efektif dalam mendeteksi kemiripan bukan hanya dari kesamaan teks atau *copy-paste*, namun juga deteksi kemiripan berbasis makna atau tema yang dibahas, termasuk pada kasus parafrase dokumen. Sehingga metode pencocokan teks atau *string* sederhana yang sulit untuk melakukan hal tersebut dapat dioptimalkan dengan penggunaan metode LSA

Disamping LSA memiliki kelemahan terutama pada pertimbangan makna secara eksplit seperti majas atau bahasa *slang*, penulis dapat disimpulkan bahwa pendekatan LSA berbasis SVD merupakan metode yang lanyak digunakan sebagai alat pendukung dalam sistem deteksi plagiarisme dokumen, terutama dalam mendekteksi kemiripan makna antar teks. Penulis harap dokumen ini bisa membantu pembaca yang tertarik pada bidang aljabar linear dan geometri atau tertarik dalam metode pendeteksian plagiarisme dokumen

V. Appendix

Sebagai tambahan material, di bawah ini merupakan link GitHub repository dan link youtube penjelasan :

1) Link reporsitory github yang berisi source code dan file dataset beserta query : https://github.com/ucupsss/Makalah-Algeo-Yusuf-Faishal
2) Link video Youtube : https://youtu.be/1xxxsL-ezu8

VI. Acknowledgment

Segala puji penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan kemudahan-Nya sehingga makalah ini dapat terselesaikan dengan baik. Ucapan terima kasih penulis sampaikan kepada keluarga tercinta atas segala dukungan, semangat, dan doa yang senantiasa mengiringi selama proses penyusunan makalah ini. Penulis juga menyampaikan apresiasi yang sebesar-besarnya kepada Bapak Dr. Ir. Rinaldi Munir, M.T, selaku dosen mata kuliah Aljabar Linear dan Geomteri, atas ilmu, bimbingan, serta arahannya yang sangat bermanfaat selama proses perkuliahan. Penulis berharap makalah ini dapat memberikan kontribusi positif dan menambah wawasan bagi para pembaca, khususnya bagi yang ingin mencoba deteksi plagiarisme atau tertarik pada bidang aljabar linear dan geometri .

References

[1] Rinaldi Munir, "Nilai Eigen dan Vektor Eigen (Bagian 1)" [Daring] Tersedia : https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf

[2] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. dan Harshman, R., "Indexing by latent semantic analysis," J. Amer. Soc. Inf. Sci., vol. 41, no. 6, pp. 391–407, Sept. 1990. [Daring]. Tersedia: https://web.archive.org/web/20241122082607/http://wordvec.colorado.edu/papers/Deerwester_1990.pdf.

[3] Landauer, T. K., Foltz, P. W., Laham, D. "Introduction to Latent Semantic Analysis" [Daring] Tersedia : https://www.researchgate.net/publication/200045222_An_Introduction_to_Latent_Semantic_Analysis.

[4] Shubham Sangole. "Mastering Cosine Metrics: A Data Scientist's Essential Toolkit" [Daring] Tersedia : https://python.plainenglish.io/mastering-cosine-metrics-a-data-scientists-essential-toolkit-3ce95eda91f9

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, 24 Desember 2025

Yusuf Faishal Listyardi 13524014