Konstruksi Ruang Laten Spesifik Domain menggunakan SVD Ko-okurensi untuk Deteksi Sinonim pada E-Commerce Fashion

Dzaki Ahmad Al Hussainy and 13524084[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524084@std.stei.itb.ac.id, [2]dzakiahmada7@gmail.com

***Abstrak*— Pesatnya pertumbuhan e-commerce saat ini berdampak pada ledakan volume data produk yang diiringi dengan tingginya variasi linguistik, khususnya pada domain fashion. Hal ini menimbulkan tantangan dalam sistem pencarian, terutama dalam penggunaan algoritma yang sering kali gagal mengenali sinonim atau istilah spesifik domain. Makalah ini mengusulkan konstruksi ruang laten menggunakan metode Singular Value Decomposition (SVD) yang diterapkan pada matriks ko-okurensi kata. Pendekatan ini bertujuan untuk mengekstraksi kedekatan semantik antar istilah guna meningkatkan akurasi deteksi sinonim pada katalog produk. Melalui transformasi ke ruang laten berdimensi rendah, hubungan semantik yang tersembunyi dapat dipetakan secara matematis. Hasil penelitian ini diharapkan dapat meningkatkan relevansi hasil pencarian pada platform e-commerce melalui pengenalan istilah yang lebih cerdas dan kontekstual**

Keywords*—*e-commerce fashion, deteksi sinonim, matriks ko-okurensi, ruang laten, singular value decomposition

I. Pendahuluan

Perkembangan pesat sektor perdagangan elektronik (*e-commerce*) telah mengakibatkan ledakan data produk secara masif, yang diiringi dengan meningkatnya variasi linguistik dalam deskripsi katalog produk. Fenomena ini sangat terlihat pada domain *fashion*, katalog produk pada *fashion* sering kali direpresentasikan oleh beragam istilah yang memiliki kedekatan semantik. Sebagai contoh, istilah "*jacket*" dan "*coat*" secara visual dan fungsional memiliki sifat laten yang serupa sebagai pakaian luaran (*outerwear*). Namun, dalam sistem pencarian biasa, perbedaan leksikal ini sering kali menjadi kendala.

Permasalahan utama muncul ketika pengguna melakukan pencarian tanpa mempertimbangkan variasi sinonim tersebut. Ketergantungan sistem pada kecocokan kata kunci (*keyword matching*) secara kaku dapat mengakibatkan kegagalan pencarian. Pengguna sering kali dituntut untuk menuliskan spesifikasi yang sangat presisi, padahal produk yang dicari mungkin tersedia di katalog namun dengan label linguistik yang berbeda. Masalah ini menunjukkan perlunya representasi kata yang tidak hanya menangkap bentuk tekstual, tetapi juga makna laten di baliknya.

Untuk mengatasi kendala tersebut, makalah ini mengaplikasikan Aljabar Linier melalui konstruksi ruang laten dari matriks ko-okurensi kata. Dengan menerapkan metode *Singular Value Decomposition* (SVD) dan *Latent Semantic Analysis* (LSA), hubungan semantik antar istilah dapat diekstraksi berdasarkan pola kemunculan kata dalam korpus data *fashion*. Pendekatan ini memungkinkan transformasi data dari ruang dimensi tinggi yang renggang (*sparse*) ke ruang laten berdimensi rendah yang lebih padat. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan sistem pencarian pintar yang mampu mendeteksi sinonim secara otomatis pada katalog produk *e-commerce*.

II. Landasan Teori

A. Representasi Kata dalam Ruang Vektor (Vector Space Model)

Pada tahap awal, kita harus memetakan entitas linguistik (kata) ke dalam entitas matematis (vektor). Secara formal, proses ini disebut sebagai Embedding Awal dalam ruang berdimensi tinggi.

1) Definisi Kosa kata sebagai Basis Standar [3]: Misalkan $V = \{w_1, w_2, \ldots, w_n\}$ adalah himpunan kosa kata unik (vocabulary) dalam domain fashion kita, di mana n adalah ukuran kosa kata Setiap kata $w_i \in V$ direpresentasikan sebagai Vektor Basis Standar (atau One-Hot Encoding) dari adanya kata pada katalog deskripsi tertentu $\mathbf{e}_i$ dalam ruang vektor riil berdimensi-n ($\mathbb{R}^n$):

$$\mathbf{e}_i = [0, 0, \ldots, 1, \ldots, 0]^T$$

Di mana angka 1 berada pada posisi ke-i. Dalam perspektif ini, setiap kata menempati sumbu koordinatnya sendiri yang unik.

2) Ortogonalitas dan Independensi Linier: [2] Secara geometris, karena setiap kata direpresentasikan oleh vektor basis standar, maka berlaku sifat Ortonormalitas:Magnitudo Satuan: Panjang setiap vektor adalah satu, $\|\mathbf{e}_i\| = 1$.Ortogonalitas: Hasil kali titik (dot product) antara dua kata yang berbeda adalah nol:

$$\mathbf{e}_i \cdot \mathbf{e}_j = 0 \quad \text{untuk } i \neq j$$

Artinya, dalam representasi awal ini, semua kata dianggap saling bebas secara linier dan tidak memiliki keterkaitan sama sekali. Secara matematis, sudut θ di antara dua kata mana pun (misal: "tunik" dan "blouse") adalah tepat 90° ($\cos\theta = 0$).

3) Kesenjangan Semantik (The Semantic Gap) [2]: Masalah utama dalam penggunaan representasi basis standar $\mathbf{e}_i$ pada domain e-commerce fashion adalah munculnya Kesenjangan Semantik (The Semantic Gap). Secara linguistik, kata-kata seperti "tunik", "blouse", dan "atasan" memiliki kedekatan makna yang sangat erat. Namun, dalam representasi aljabar linier awal, setiap kata tersebut diperlakukan sebagai dimensi yang saling bebas (linearly independent). Secara formal, jika kita memiliki dua kata sinonim w_i dan w_j, representasi vektor basis standar mereka memenuhi sifat:

$$\mathbf{e}_i^T \mathbf{e}_j = 0$$

Hal ini mengimplikasikan bahwa sudut θ di antara keduanya adalah 90° ($\cos\theta = 0$), yang berarti tidak ada informasi kesamaan yang dapat diekstraksi. Kesenjangan ini terjadi karena model hanya mencatat identitas kata secara diskrit dan mengabaikan hubungan antar-kata yang terdapat dalam struktur teks katalog produk. Secara geometris, semua titik data tersebar di ujung-ujung sumbu koordinat yang berjauhan, sehingga ruang tersebut tidak memiliki struktur kelompok (clustering) yang bermakna.

B. Konstruksi Matriks Ko-okurensi

Untuk menjembatani kesenjangan semantik tersebut, representasi kata harus diubah dari Vektor Identitas menjadi Vektor Distribusi (Distributional Representation). Pendekatan ini didasarkan pada Distributional Hypothesis yang menyatakan bahwa kata-kata yang muncul dalam konteks yang serupa cenderung memiliki makna yang serupa pula. Tujuan utama dari konstruksi matriks ko-okurensi $C \in \mathbb{R}^{n \times n}$ adalah untuk membangun sebuah ruang fitur di mana setiap kata w_i didefinisikan oleh frekuensi interaksinya dengan kata-kata lain dalam jendela konteks tertentu [2]. Secara aljabar, transformasi ini menggeser representasi kata:

- **Dari Basis Standar ke Vektor Konteks:** Setiap baris $\mathbf{c}_i$ dalam matriks C bukan lagi berisi satu angka 1, melainkan berisi distribusi frekuensi kemunculan bersama (co-occurrence counts).
- **Menciptakan Korelasi Non-Zero:** Jika kata "tunik" dan "blouse" sering muncul bersama kata konteks yang sama (misalnya: "wanita", "katun", "kerah"), maka vektor $\mathbf{c}_{\text{tunik}}$ dan $\mathbf{c}_{\text{blouse}}$ akan memiliki nilai non-nol pada dimensi-dimensi yang sama.

Hasilnya, hasil kali titik (dot product) antara dua kata tersebut tidak lagi nol:

$$\mathbf{c}_i^T \mathbf{c}_j > 0$$

Secara geometris, konstruksi ini "menarik" vektor-vektor yang tadinya ortogonal agar mulai mengarah ke arah yang serupa di dalam ruang vektor. Inilah langkah awal untuk mengubah data mentah menjadi informasi semantik sebelum kemudian dioptimasi lebih lanjut menggunakan Singular Value Decomposition (SVD) untuk menghilangkan derau (noise).

C. Transformasi Probabilistik

Meskipun matriks ko-okurensi C telah menangkap hubungan antar-kata, data mentah (raw counts) memiliki kelemahan besar: frekuensi kata yang sangat tinggi (berdasarkan Hukum Zipf) akan mendominasi perhitungan dan menutupi hubungan semantik yang halus. Oleh karena itu, diperlukan transformasi dari frekuensi mentah menjadi kekuatan asosiasi.

1) Konsep Pointwise Mutual Information (PMI): Secara statistik, PMI mengukur seberapa sering dua kata w (word) dan c (context) muncul bersamaan dibandingkan dengan ekspektasi kemunculan mereka secara acak [2]. Rumusnya adalah:

$$PMI(w, c) = \log_2 \left(\frac{P(w, c)}{P(w)P(c)} \right)$$

Di mana:

- $P(w, c)$ adalah probabilitas kedua kata muncul bersama.
- $P(w)$ dan $P(c)$ adalah probabilitas masing-masing kata muncul secara independen.

Interpretasi Aljabar:

- $PMI > 0$: Kedua kata memiliki hubungan semantik yang kuat (muncul lebih sering dari kebetulan).
- $PMI = 0$: Kedua kata bersifat independen.
- $PMI < 0$: Kedua kata saling "menghindar" (muncul lebih jarang dari kebetulan).

2) Transformasi ke PPMI: Dalam praktiknya, nilai PMI negatif sering kali tidak stabil (terutama pada data yang jarang/sparse) dan kurang bermakna dalam pencarian semantik [2]. Oleh karena itu, kita menggunakan PPMI, di mana semua nilai negatif diganti dengan nol:

$$PPMI(w, c) = \max(0, PMI(w, c))$$

D. Singular Value Decomposition (SVD)

1) Singular Value Decomposition Tereduksi: Singular Value Decomposition (SVD) tereduksi merupakan bentuk penyederhanaan dari SVD penuh yang hanya mempertahankan komponen-komponen paling penting dari suatu matriks [5]. Reduksi ini dilakukan dengan memanfaatkan fakta bahwa tidak semua nilai singular memberikan kontribusi signifikan terhadap struktur matriks asal.

Secara umum, SVD penuh memfaktorkan matriks $A \in \mathbb{R}^{m \times n}$ sebagai:

$$A = U\Sigma V^T$$

Namun, dalam banyak aplikasi, hanya k nilai singular terbesar yang dipertahankan, dengan $k < \min(m, n)$. Dengan demikian, diperoleh SVD tereduksi:

$$A \approx U_k \Sigma_k V_k^T$$

Adapun komponen-komponen SVD tereduksi didefinisikan sebagai berikut:

- **Matriks** U_k $(m \times k)$**:** Merupakan matriks yang kolom-kolomnya adalah k *left singular vectors* utama. Vektor-vektor ini berasosiasi dengan k nilai singular terbesar dan merupakan vektor eigen dari matriks AA^T.
- **Matriks** Σ_k $(k \times k)$**:** Merupakan matriks diagonal yang berisi k nilai singular terbesar $\sigma_1, \sigma_2, \ldots, \sigma_k$, dengan urutan menurun ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$). Setiap σ_i merupakan akar kuadrat dari nilai eigen matriks $A^T A$.
- **Matriks** V_k^T $(k \times n)$**:** Merupakan transpos dari matriks V_k yang kolom-kolomnya adalah k *right singular vectors* utama. Vektor-vektor ini adalah vektor eigen dari matriks $A^T A$.

E. Metode Analisis Semantik Laten (Latent Semantic Analysis)

Latent Semantic Analysis (LSA) merupakan metode komputasi yang digunakan untuk mengekstraksi dan merepresentasikan makna kata berdasarkan konteks kemunculannya dalam sekumpulan data teks. LSA bekerja berdasarkan Hipotesis Distribusi, yang menyatakan bahwa kata-kata yang muncul dalam lingkungan konteks yang serupa cenderung memiliki keterkaitan makna (semantik) yang erat [1].

Tujuan utama LSA adalah untuk mengungkap struktur laten (tersembunyi) dalam penggunaan bahasa yang tidak terlihat secara harfiah. Secara matematis, LSA mentransformasikan pencarian berbasis kata kunci (keyword matching) menjadi pencarian berbasis konsep di dalam ruang vektor berdimensi rendah.

Proses implementasi LSA terdiri dari tiga tahapan utama sebagai berikut:

- Konstruksi Matriks Ko-okurensi
  Tahap pertama dalam LSA adalah merepresentasikan data tekstual mentah ke dalam bentuk struktur linear. Matriks ko-okurensi $C \in \mathbb{R}^{m \times m}$ dibangun untuk mencatat frekuensi kemunculan bersama antara kata target dengan kata-kata di sekitarnya dalam jendela konteks tertentu (context window).
- Transformasi Bobot melalui PPMI
  Frekuensi kemunculan mentah seringkali tidak merepresentasikan kekuatan hubungan semantik yang sebenarnya. Oleh karena itu, LSA menerapkan transformasi Positive Pointwise Mutual Information (PPMI) untuk menghitung kekuatan asosiasi antar kata.

  $$PPMI(w, c) = \max\left(0, \log_2 \frac{P(w, c)}{P(w)P(c)}\right)$$

  Transformasi ini berfungsi untuk menormalisasi variansi data dengan cara menekan bobot kata-kata umum yang muncul secara kebetulan dan menonjolkan hubungan antara pasangan kata yang memiliki signifikansi informasi yang tinggi.
- Reduksi Dimensi via Singular Value Decomposition
  Tahap inti dari LSA adalah melakukan proyeksi vektor dari ruang koordinat asli yang luas ke dalam Ruang Laten berdimensi rendah menggunakan Singular Value Decomposition (SVD). Secara formal, matriks didekomposisi menjadi:

  $$A \approx U_k \Sigma_k V_k^T$$

  Melalui proses ini, dimensi-dimensi yang tidak signifikan (mewakili noise) dibuang, sementara struktur utama yang mewakili hubungan semantik dipertahankan. Hasil akhirnya adalah sekumpulan vektor padat (dense vectors) di mana kata-kata yang bersifat sinonim akan terletak berdekatan secara geometris, meskipun kata-kata tersebut tidak pernah muncul bersama dalam satu dokumen yang sama.

F. Ukuran Kemiripan Kosinus (Cosine Similarity)

Setelah vektor kata atau produk diproyeksikan ke dalam ruang laten melalui proses LSA, diperlukan sebuah metrik untuk mengukur tingkat kemiripan antar entitas tersebut. Dalam pengolahan bahasa alami dan sistem pencarian semantik, Cosine Similarity merupakan cara untuk mengukur kesamaan antara dua vektor. Cosine Similarity mengukur kosinus sudut antara dua vektor bukan nol dalam ruang multidimensi. Secara matematis, kemiripan antara vektor $\mathbf{A}$ dan vektor $\mathbf{B}$ dihitung sebagai hasil kali titik (dot product) dibagi dengan perkalian magnitudo (panjang) dari kedua vektor tersebut [2]:

$$\text{similarity}(\mathbf{A}, \mathbf{B}) = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

III. IMPLEMENTASI

A. Arsitektur Implementasi Sistem

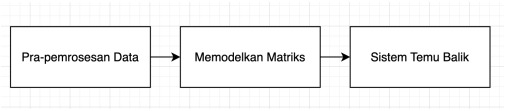
Implementasi sistem dilakukan dengan menggunakan bahasa Python dan menggunakan beberapa library untuk mempermudah implementasi yaitu:

1) Pandas digunakan untuk pengolahan data katalog utama.
2) Numpy digunakan untuk perhitungan matriks.
3) Regular Expression (re) digunakan untuk pengolahan teks dan tokenisasi.
4) csv digunakan untuk melakukan penyimpanan data hasil pembersihan.
5) Scipy digunakan untuk optimisasi perhitungan perkalian matriks.
6) nltk digunakan untuk mendapatkan stopwords.
7) wikipedia digunakan untuk melakukan pencarian definisi kata paling sering muncul.
8) Skitlearn digunakan untuk implementasi SVD tereduksi dan Cosine Similarity.
9) Pickle digunakan untuk penyimpanan hasil dekomposisi.

Memerosesan ini melalui tiga tahapan utama yang dirancang untuk mentransformasi data tekstual mentah menjadi sebuah

ruang vektor laten yang mampu menangkap hubungan semantik antar-kata.

- Pra-pemrosesan Data
  Data yang digunakan dalam penelitian ini diperoleh dari Kaggle yaitu Asos E-Commerce Dataset [6].
  Pada tahap ini, dilakukan serangkaian transformasi teks untuk memastikan kualitas data sebelum masuk ke pemodelan matematis.
- Memodelkan Matriks
  Data yang telah melalui tahap pra-pemrosesan kemudian dimodelkan ke dalam struktur aljabar linier yaitu Matriks Ko-okurensi (C) kemudian Transformasi PPMI (Positive Pointwise Mutual Information) Data yang sudah dijadikan PMMI ini kemudian akan difaktoriasi dengan metode SVD tereduksi dan kemudian akan dibuat similaritas untuk pencarian kueri
- Sistem Temu Balik
  Tahap akhir adalah pembentukan ruang laten untuk mendukung deteksi sinonim dan pencarian produk melalui SVD tereduksi kemudian pengukuran kemiripan dengan (Cosine Similarity)



Gambar III.1. diagram alur proses sistem

B. Pra-pemrosesan Data Spesifik Domain Fashion

Pada tahap ini data mentah akan dilakukan menjadi korpus teks yaitu teks yang sangat besar yang siap dianalisis. Tahapan yang dilakukan meliputi:

1) Pembersihan Teks yaitu menghilangkan karakter khusus seperti angka dan simbol yang tidak relevan
2) Pengayaan Semantik yaitu dalam tahap ini menggabungkan data katalog dengan definisi eksternal untuk memperkuat konteks domain fashion.
3) Normalisasi yaitu mengubah seluruh teks menjadi huruf kecil dan melakukan normalisasi untuk memecah deskripsi produk menjadi unit kata individual
4) Eliminasi Stop-Words yaitu menghapus kata-kata umum dan tidak memiliki makna seperti (”the”, ”of”, dan ”with”) yang tidak memiliki nilai informasi tinggi dalam domain fashion

1) Pembersihan Teks : Sebelum pembersihan teks ini pada kolom warna dan nama yang berisi deskripsi produk akan dikonkatinasi agar mendapatkan data yang berisi deskripsi dan warna. implementasi kode pembersihannya sebagai berikut

```
def smart_append(row):
    name = str(row['name']).lower()
    color = str(row['color']).lower()

    clean_name = re.sub(r'[^a-zA-Z0-9]', ' ', name)
    clean_color = re.sub(r'[^a-zA-Z0-9]', ' ',
        color)

    set_word = set(clean_name.split())

    word_to_add = []
    color_words = clean_color.split()

    for word in color_words:
        if word in ['and', 'or', 'in', 'with']:
            continue
        if word not in set_word:
            word_to_add.append(word)
    if word_to_add:
        return name + " " + " ".join(word_to_add)
    else:
        return name
```

Implementasi Kode 1. Pembersihan Katalog Deksripsi

hasil dari pembersihan ini akan dimasukan ke sebuah kolom baru, bernama processed_text.

2) Pengayaan Semantik: Pada bagian ini sudah didapatkan sebuah korpus teks yang hanya berisi katalog deskripsi produk. Dilakukan infusi pengetahuan dengan menggabungkan definisi istilah fashion ke dalam data pelatihan (korpus), sehingga matriks ko-okurensi yang dihasilkan memiliki kepadatan informasi semantik yang tinggi khusus untuk industri fashion. Secara matematis korpus total akan berisi

$$D_{total} = D_{katalog} \cup D_{eksternal}$$

Di mana D_{total} adalah korpus akhir yang digunakan untuk membangun matriks ko-okurensi.
Pada bagian ini dilakukan pengambilan definisi 600 kata yang paling sering muncul kemudian memasukkan kedalam korpus. Implementasi kode ini menggunakan sedikit data mining dari wikipedia untuk mendapatkan 600 definisi kata teratas.

```
print("\n--- Mulai Mining Wikipedia ---")

for i, word in enumerate(keywords):
    # Cek cache dulu
    if word in cached_results:
      external_corpus.append(cached_results[word])
      print(f"CACHED")
      continue

    summary, status = get_wiki_content(word)

    if summary:
      clean_summary = summary.replace('\n', '
          ').strip().lower()
      external_corpus.append(clean_summary)
      print(f"OK")
    else:
      skipped_words.append({'word': word, 'reason':
          status})
      print(f"Skip ({status})")
```

Implementasi Kode 2. Pengkayaan Semantik

Setelah dilakukan *data mining* ini kemudian ada beberapa data yang tidak ditemukan definisinya jika tidak ditemukan definisinya maka akan dicari secara manual dan ditambahkan ke korpus total.

3) Normalisasi, tokenisasi dan Eliminasi Kata Tidak Bermakna: Pada bagian ini data diambil salah satu bagian korpus kemudian dilakukan normalisasi kemudian dilakukan eliminasi kata-kata yang pendek atau jarang keluar.

```python
def clean_text(text):
    text = str(text).lower()
    # Hapus simbol tapi pertahankan spasi
    text = re.sub(r'[^a-z0-9\s]', '', text)
    tokens = text.split()

    # Filter: Hapus stopwords dan kata terlalu pendek
    en_stopwords = set(stopwords.words('english'))
    tokens = [t for t in tokens if t not in en_stopwords and len(t) > 2]

    return tokens
```

Implementasi Kode 3. Normalisasi dan Eliminasi Kata Stop-Words

Pada bagian ini dilakukan tokenisasi dan pembuatan larik kata yang digunakan. Dibagian ini juga dilakukan optimasi yaitu menghilangkan kata langka atau kata yang selalu sering muncul karena kedua jenis kata ini bisa dikategorikan sebagai noise pada laten simantik yang difokuskan pada domain *fashion*.

```python
vocab = set()
tokenized_sentences = []
for sentence in full_corpus:
    tokens = clean_text(sentence)
    if tokens: # Pastikan tidak kosong
        tokenized_sentences.append(tokens)
        vocab.update(tokens)
sorted_vocab = sorted(list(vocab))
# Filter kata langka dan terlalu sering (keduanya noise)
word_freq = Counter()
for tokens in tokenized_sentences:
    word_freq.update(tokens)

# Hapus kata yang muncul < 3 kali atau > 70% (kemungkinan stopword yang terlewat)
threshold_rare = 3
threshold_frequent = 0.7 * len(tokenized_sentences)
filtered_vocab = [w for w in sorted_vocab if
    threshold_rare <= word_freq[w] <=
    threshold_frequent]
```

Implementasi Kode 4. Tokenisasi dan pembuatan Larik Kata

C. Memodelkan Matriks

Pada bagian ini matriks akan dimodelkan melewati 2 tahap yaitu pembuatan matriks ko-okurensi kemudian melakukan PPMI

1) Konstruksi Matriks Ko-okurensi (C): Membangun matriks yang mencatat frekuensi kemunculan bersama antar-kata dalam jendela konteks tertentu. Matriks ini merepresentasikan hubungan antar-kata berdasarkan pola distribusinya di dalam katalog.

```python
print("--- Building Co-occurrence Matrix ---")
window_size = 5
co_matrix = lil_matrix((vocab_size, vocab_size),
    dtype=np.float32)

for tokens in tokenized_sentences:
    token_ids = [word2id[w] for w in tokens if w in
        word2id]  # Skip words not in filtered vocab
    len_tokens = len(token_ids)

    if len_tokens == 0:
        continue

    for i, target_id in enumerate(token_ids):
        start = max(0, i - window_size)
        end = min(len_tokens, i + window_size + 1)

        for j in range(start, end):
            if i == j: continue
            context_id = token_ids[j]
            co_matrix[target_id, context_id] += 1
co_matrix = co_matrix.tocsr()
```

Implementasi Kode 5. Membuat Matriks Ko-okurensi

Implementasi konstruksi matriks ko-okurensi dilakukan dengan memindai seluruh korpus menggunakan algoritma jendela geser (sliding window). Sebuah jendela konteks simetris dengan ukuran $k = 5$ ditetapkan untuk menangkap hubungan semantik lokal antar-kata. Algoritma iteratif digunakan untuk memetakan setiap kata w_i dan kata konteks w_j ke dalam indeks matriks, di mana setiap kemunculan bersama akan meningkatkan nilai sel C_{ij} secara inkremental. Setelah iterasi korpus selesai, struktur data dikonversi menjadi Compressed Sparse Row (CSR) untuk mengoptimalkan kinerja operasi aljabar linier pada tahap faktorisasi SVD selanjutnya.

2) Transformasi PPMI (Positive Pointwise Mutual Information): Pada tahap ini, matriks ko-okurensi frekuensi mentah ditransformasi menjadi matriks Pointwise Mutual Information (PPMI). Implementasi dilakukan dengan mengiterasi elemen-elemen yang tidak nol untuk efiseiensi komputasi. Digunakan juga parameter *smoothing* (α = 1) pada frekuensi marginal kata untuk menjaga agar tidak terjadi pembagian dengan nol dan hanya mengambil nilai positif.

```python
print("--- Calculating PPMI with Smoothing (Vectorized) ---")
total_sum = co_matrix.sum()
word_counts =
    np.array(co_matrix.sum(axis=1)).flatten()
# tambah smoothing untuk menghindari log(0)
alpha = 1.0
# Hindari pembagian dengan nol
word_counts[word_counts == 0] = 1
# Matriks Index (i, j) dan nilainya (data) dari sparse matrix
rows, cols = co_matrix.nonzero()
data = co_matrix.data
# Rumus PMI dengan smoothing: log2( (count(x,y) + alpha) / ((count(x) + alpha)*(count(y) + alpha)) * total )
pmi_values = []
for i in range(len(data)):
    row_idx = rows[i]
    col_idx = cols[i]
    c_xy = data[i]
    c_x = word_counts[row_idx]
    c_y = word_counts[col_idx]
    # Hitung PMI dengan smoothing
    val = np.log2((c_xy * total_sum) / ((c_x +
        alpha) * (c_y + alpha)))
    pmi_values.append(max(0, val))
ppmi_matrix = csr_matrix((pmi_values, (rows,
    cols)), shape=(vocab_size, vocab_size))
```

Implementasi Kode 6. Transformasi PPMI

Pada bagian akhir matriks baru disimpan dalam format Compressed Sparse Row (CSR) yang kemudian digunakan untuk proses SVD tereduksi.

D. Sistem Temu Balik

Pada implementasi temu balik ini menggunakan metode reduksi ruang vektor asli ke ruang laten berdimensi rendah dari proses ini juga membuang noise dari dimensi-dimensi yang kurang berpengaruh.

1) Faktorisasi Matriks dan Reduksi Dimensi (SVD): Matriks PPMI yang terbentuk merupakan matriks simetris karena misalnya $P(baju, kemeja) = P(kemeja, baju)$ sehingga PPMI merupakan matriks simeteris. Karena PMMI merupakan matriks simetris, hasil dekomposisi matriks tersebut dapat dituliskan sebagai:

$$\mathbf{M} \approx U_k \Sigma_k V_k^T$$

interpretasi adalah

1) U_k dan V_k^T merepresentasikan koordinat kada dalam ruang fitur laten berdimensi k. setiap dimensi merepresentasikan konsep abstrak seperti kategori bahan,gender, atau gaya yang diambil dari statistik data.
2) Σ_k merepresentasikan kekuatan informasi dari setiap dimensi laten.

Implementasi pada bagian ini menggunakan pustaka skitlearn dalam menggunakan SVD tereduksi. Matriks U adalah hasil dari perkalian matriks $U \times \Sigma$ sehingga mendapatkan matriks U yang sudah direduksi dan dihitung kekuatan relevansinya. Pada pustaka skitlearn SVD tereduksi juga dapat dihitung seberapa banyak informasi yang terrangkum oleh SVD dengan perhitungan akumulasi variansi.

```
print("--- Running SVD ---")
n_components = 50
svd = TruncatedSVD(n_components=n_components,
    random_state=42, n_iter=1000)
U_matrix = svd.fit_transform(ppmi_matrix)
explained_var = svd.explained_variance_ratio_.sum()
```

Implementasi Kode 7. Faktorisasi Matriks dan Reduksi Dimensi (SVD)

Hasil dari proses SVD (matriks U dan indeks kosakata) kemudian disimpan menggunakan modul Pickle. Teknik serialisasi biner ini digunakan agar model ruang laten dapat dimuat kembali secara efisien tanpa perlu melakukan komputasi ulang faktorisasi matriks setiap kali sistem dijalankan.

2) Analisis Deteksi Sinonim dan Kedekatan Semantik: Mengingat ruang laten dibangun menggunakan dataset katalog internasional (Asos), sistem memerlukan mekanisme penerjemahan untuk menangani masukan dalam Bahasa Indonesia. Proses deteksi sinonim dilakukan melalui tahapan berikut:

1) Penerjemahan Kueri
   Kata input diterjemahkan ke dalam Bahasa Inggris menggunakan kamus pemetaan (dictionary mapping) sederhana yang telah didefinisikan sebelumnya.
2) Ekstraksi Vektor
   Sistem mengambil vektor koordinat kata yang telah diterjemahkan dari matriks U berdasarkan indeksnya.
3) Perhitungan Similaritas
   Tingkat kedekatan semantik atau sinonim antara dua kata dihitung menggunakan metode Cosine Similarity. Nilai yang mendekati 1 mengindikasikan bahwa kedua kata memiliki makna yang sangat mirip dalam konteks fashion.

```
def get_sim(w1, w2):
    if w1 not in word2id or w2 not in word2id:
        return 0
    v1 = U_matrix[word2id[w1]].reshape(1, -1)
    v2 = U_matrix[word2id[w2]].reshape(1, -1)
    return cosine_similarity(v1, v2)[0][0]
```

Implementasi Kode 8. Deteksi Sinonim dan Kedekatan Semantik

3) Sistem Pencarian Kueri: Dalam sistem pencarian sebuah kueri, teks yang masuk kan ditranlasikan ke dalam bahasa Inggris kemudian dibersihkan dan ditokenisasi menjadi vektor kata. Jika vektor kata tidak terdefinisi dalam kamus bahasa Indonesia ke bahasa Inggris maka tidak akan dicantumkan kedalam vektor kueri vektor kueri ini kemudian akan dihitung rata-ratanya dan dihitung similaritasnya pada matriks katalog produk. kemudian mengambil top_k produk yang memiliki kesamaan yang tinggi.

```
query_en = translate_query(query)
    clean_query = re.sub(r'[^a-z0-9\s]', '',
        query_en).split()

    print(f"[Mencari]  '{query}' -> diterjemahkan:
        {clean_query}")

    vectors = []
    found_words = []

    for word in clean_query:
        if word in word2id:
            word_idx = word2id[word]
            vectors.append(U_matrix[word_idx])
            found_words.append(word)
        else:
            print(f"Kata '{word}' tidak ada di
                vocabulary (diabaikan)")

    if not vectors:
        print("Maaf, tidak ada kata yang dikenali
            sistem.")
        return

    # Buat Vektor Rata-rata Query
    query_vec = np.mean(vectors, axis=0).reshape(1,
        -1)

    similarities = cosine_similarity(query_vec,
        product_matrix)
    top_indices =
        similarities[0].argsort()[-top_k:][::-1]
```

Implementasi Kode 9. Potongan Kode dalam Mencari Kueri

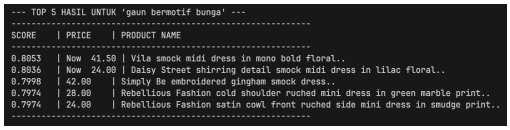
IV. Hasil dan Evaluasi

A. Hasil

Pada melakukan pengujian hasil temu balik sistem dibuat 3 pertanyaan dan menguji apakah sistem menemukan pola tersembunyi, berikut adalah kuerinya:

1) gaun bermotif bunga
2) baju bersantai di siang hari
3) jaket berbahan kulit

Gambar IV.1 memperlihatkan hasil kueri untuk ”gaun bermotif bunga”. Sistem berhasil mengembalikan produk dengan kata kunci seperti floral, dress, dan midi dress. Hal ini

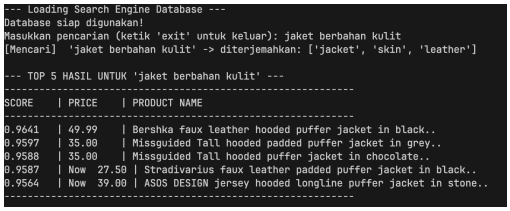


Gambar IV.1. Hasil Kueri Pertanyaan 1

menunjukkan ”bermotif bunga” berhasil dipetakan ke konsep laten floral. Hal ini menunjukkan juga sistem tidak melkaukan pencocokan literal bahasa Indonesia ke bahasa Inggris secara satu-satu namun menggunakan pendekatan vektor semantik.

Gambar IV.2. Hasil Kueri Pertanyaan 2

Pada kueri ini IV.2, sistem menerjemahkan input ke token seperti shirt, in, day, dan afternoon. Meskipun kata bersantai dan afternoon tidak terdapat langsung dalam vocabulary, sistem tetap menghasilkan hasil yang relevan seperti resort shirt, sweatshirt, dan denim top. Hal ini memunjukan kata yang tidak dalam larik kata secara eksplisit diabaikan, representasi laten memungkinkan sisten memahami konteks cual dan daywear meskipun tidak muncul langsung, pada produk pyjama set bisa muncul karena memiliki ko-okurensi tinggi dengan konteks santai dan non-formal.



Gambar IV.3. Hasil Kueri Pertanyaan 3

Pada kueri ini IV.3, sistem menerjemahkan kulit menjadi leather dan berhasil menampilkan produk seperti faux leather jacket dan padded puffer jacket. Produk dengan bahan faux leather muncul dengan skor tinggi, menunjukan sistem memahami kesamaan fungsi dan persepsi material. Item puffer juga muncul karena berko-okurensi dengan kata jacket dan leather dalam konteks fashion musim dingin.

B. *Evaluasi Kualitatif Sistem*

1) Sistem mampu menangani sinonim lintas bahasa dan sinonim implisit.
2) Kesalahan akibat vocabulary gap dapat diminimalkan melalui pendekatan laten.
3) Sistem dapat digunakan sebagai search engine berbasis semantik pada e-commerce fashion.

Namun, keterbatasan tetap ada, seperti:

1) Ketergantungan pada kualitas dan kelengkapan data ko-okurensi.
2) Sinonim yang sangat jarang muncul tetap sulit ditangkap.
3) Translasi bahasa yang terlalu literal dan sulit untuk mendapatkan kalimat translasi yang sepadan.

V. KESIMPULAN

A. *kesimpulan*

Sistem konstruksi ruang laten spesifik domain menggunakan SVD ko-okurensi terbukti efektif untuk deteksi sinonim dan pencarian semantik pada e-commerce fashion. Pendekatan ini mampu menjembatani perbedaan bahasa, variasi istilah, dan deskripsi produk yang tidak seragam, sehingga meningkatkan relevansi hasil pencarian secara signifikan dibanding metode pencocokan kata literal.

B. *Lampiran*

1) Repositori Github : https://github.com/HussainDzaki/fashion-latent-space-svd.
2) Sumber data set : https://www.kaggle.com/datasets/trainingdatapro/asos-e-commerce-dataset-30845-products.
3) Dibuat menggunakan Latex : https://github.com/HussainDzaki/fashion-latent-space-svd/tree/main/Makalah

REFERENCES

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman.1990. ”Indexing by latent semantic analysis,” *J. Amer. Soc. Inform. Sci.*, vol. 41, no. 6, pp. 391–407. [Online]. Available: http://wordvec.colorado.edu/papers/Deerwester_1990.pdf

[2] Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. Morgan & Claypool Publishers.[Online]. Available : https://32931414.s21i.faiusr.com/61/ABUIABA9GAAgxLqCuwYojOPTXQ.pdf, Diakses pada 6 Desember 2025 jam 22.32

[3] Jurafsky, D., & Martin, J. H. (2024). Speech and Language Processing (3rd ed. draft). Chapter 6: Vector Semantics and Embeddings. [Online] Available : https://web.stanford.edu/~jurafsky/slp3/old_feb24/6.pdf, Diakses pada 6 Desember 2025 jam 22.44

[4] Munir, Rinaldi.2025. Singular Value Decomposition (SVD) bagian 1. [Online]. Available : https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-21-Singular-value-decomposition-Bagian1-2025.pdf, Diakses pada 6 Desember 2025 jam 21.44

[5] Munir, Rinaldi.2025. Singular Value Decomposition (SVD) bagian 2. [Online]. Available : https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-22-Singular-value-decomposition-Bagian2-2025.pdf, Diakses pada 6 Desember 2025 jam 21.22

[6] TrainingDataPro, “Asos E-Commerce Dataset - 30,845 Products,” Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/trainingdatapro/asos-e-commerce-dataset-30845-products. [Accessed: 2025].

VI. UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga makalah ini dapat diselesaikan dengan baik. Penulis menyampaikan apresiasi dan terima kasih kepada dosen pengampu mata kuliah Aljabar Linear dan Geometri IF2123, yaitu Bapak Rinaldi Munir, Bapak Rila Mandala, dan Bapak Arrival Dwi Sentosa, atas bimbingan dan ilmu yang telah diberikan. Ucapan terima kasih juga disampaikan kepada para asisten mata kuliah Aljabar Linear dan Geometri IF2123 yang telah membantu dan mendukung kelancaran proses pembelajaran. Akhir kata, penulis berharap makalah ini dapat memberikan manfaat dan menambah wawasan bagi seluruh pembaca.

VII. 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
Dzaki Ahmad Al Hussainy, 13524084