Penerapan Vektor di Ruang Euclidean dan *Cosine Similarity* untuk Koreksi Kata Bahasa Indonesia

Alfian Hanif Fitria Yustanto - 13523073[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523073@itb.ac.id

***Abstract*— Kesalahan dalam penulisan sering kali terjadi, baik berupa salah penulisan kata maupun penggunaan kata yang tidak sesuai. Fitur seperti koreksi kata dapat membantu mengelola teks dan memperbaiki kesalahan tersebut secara otomatis, baik dari segi penulisan maupun penggunaan bahasa. Pada makalah ini, salah satu pendekatan yang digunakan adalah pendekatan metode bigram, dengan merepresentasikan sebuah kata sebagai kumpulan pasangan karakter berurutan, dengan memperhatikan urutan kemunculannya untuk membentuk vektor kata. Kemudian, cosine similarity digunakan untuk menghitung tingkat kemiripan antara vektor kata query dan vektor kata dalam dataset.**

***Keywords*— Bigram, cosine similarity, koreksi kata,vektor.**

I. PENDAHULUAN

Pada masa ini, komputer dan ponsel pintar merupakan benda yang mungkin digunakan setiap hari oleh sebagian besar orang. Kegiatan yang umum dilakukan seperti membuat teks, berbalas pesan dan pencarian di internet tidak akan terlepas dalam satu proses penting yaitu proses mengetik. Walau dilakukan hampir setiap hari, masih sering terjadi kesalahan dalam pengetikan pada komputer dan ponsel pintar. Kesalahan dapat berupa kesalahan penulisan atau kesalahan bahasa. Salah satu cara yang dapat digunakan untuk mengatasi kesalahan-kesalahan tersebut adalah program koreksi kata.

Konsep koreksi kata berfokus pada pengolahan kata dan membandingkannya dengan kumpulan kata benar yang dapat berasal dari KBBI (Kamus Besar Bahasa Indonesia) atau sumber lain. Koreksi kata sudah banyak tersedia di berbagai aplikasi seperti microsoft word, google docs, dan google keyboard. Fitur ini dapat diimplementasikan secara sederhana dengan mencari kemiripan vektor yang dibentuk dari kata masukan pengguna dan kata-kata yang berada pada dataset.

II. DASAR TEORI

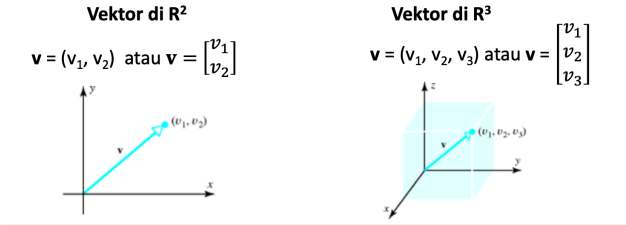
2.1. Vektor di Ruang Euclidean

Vektor merupakan salah satu representasi kuantitas fisik yang memiliki besaran dan arah. Vektor dapat dilambangkan dengan huruf kecil dicetak tebal ($\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$) atau huruf kecil yang memiliki panah ($\vec{u}$, $\vec{v}$, $\vec{w}$) .

Vektor biasa digambarkan sebagai panah dari satu titik ke titik lain. Arah vektor ditentukan dari kedua titik tersebut yang menjadi titik awal dan titik akhir. Panjang sebuah vektor $\boldsymbol{v}$ disebut sebagai norma $\boldsymbol{v}$. Norma vektor di ruang R^2 dapat ditulis dengan,

$$\|v\| = \sqrt[2]{{v_1}^2 + {v_2}^2}$$

Ruang vector merupakan ruang tempat vektor didefinisikan. Ruang vector ini disebut juga ruang Euclidean yang dapat berupa R^2 , R^3 , R^n.



Gambar 2.1.1 Visualisasi R^2 , R^3 , R^n
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf

Dalam pemrosesan teks, ruang vektor digunakan untuk merepresentasikan kumpulan kata sebagai vektor dengan dimensi tertentu. Representasi ini memungkinkan analisis hubungan antar kata melalui operasi matematis pada vektor.

2.2. Bigram

Bigram merupakan salah satu konsep yang sering digunakan dalam pemroresan kata terutama untuk kepentingan *machine learning.* Bigram merupakan bentuk sederhana dari n-gram, yaitu n-gram dengan n = 2. Bigram biasanya digunakan untuk menghubungkan urutan kata

dengan memecah sebuah kalimat menjadi pasangan kata.

```
Kalimat Awal =
" Alfian sedang membuat makalah Algeo "

Representasi bigram =
["Alfian sedang"," sedang membuat","membuat makalah","makalah algeo"]
```

Gambar 2.2.1 visualisasi bigram untuk menghubungkan kata
Sumber : Dokumen penulis

Namun untuk kepentingan koreksi kata , bigram yang biasa digunakan untuk menghubungkan kata saat ini dimodifikasi untuk menghubungkan karakter pada setiap kata.

```
Kata Awal =
" makalah "

Representasi bigram =
["ma" , "ak", "ka", "al", "la", "ah"]
```

Gambar 2.2.2 visualisasi bigram untuk menghubungkan karakter
Sumber : Dokumen penulis

Dengan begini setiap kata dapat direpresentasikan sebagai suatu vektor berdasarkan kemunculan bigram karakternya.

2.3. Information Retrieval

Information Retrieval (IR) merupakan suatu proses menemukan kembali informasi relevan yang relevan terhadap kebutuhan pengguna dari koleksi data yang besar. IR berbeda dengan pencarian di dalam basis data dan umumnya digunakan untuk melakukan pencarian informasi yang tidak terstruktur.

Contoh penerapan IR adalah mesin pencari seperti google dan sistem rekomendasi. Proses IR melibatkan beberapa tahap yaitu tahapan indexing atau membuat representasi data, pemrosesan query , dan penyusunan hasil relevansi.

Salah satu model IR adalah model ruang vektor. Pada makalah ini ruang vektor dibentuk oleh *n* buah bigram unik pada dataset. Kumpulan bigram tersebut akan membentuk ruang vektor berdimensi *n.* Setiap kata pada dataset maupun *query* dinyatakan sebagai vektor. $w = (w_1, w_2, \ldots, w_n)$ di dalam vektor R^n dimana w_i adalah bobot kemunculan setiap bigram di dalam query atau dataset kata.

2.3. Dot Product

Dot product merupakan salah satu operasi aljabar vektor yang menggabungkan dua vektor. *Dot product* digunakan untuk menghitung hubungan antara dua vector seperti perhitungan proyeksi dan sudut diantara dua vektor. Jika terdapat dua buah vektor $\vec{u}$ dan $\vec{v}$ dalam suatu ruang R^n maka *dot product* dapat dihitung dengan persamaan,

$$\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n$$

Secara geometri dot product juga dapat dibuat dengan menghubungkan panjang dari vektor $\vec{u}$ dan $\vec{v}$ serta sudut θ yang berada diantaranya.

$$\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos\theta$$

2.4. Cosine Similarity

Cosine similarity digunakan untuk menentukan kemiripan antara satu vektor dengan vektor yang lain. Pada kasus ini *cosine similarity* digunakan untuk menentukan kemiripan diantara dua vektor kata. Kesamaan antara dua vektor diukur dengan rumu *cosine similarity* yang merupakan bagian dari rumus perkalian titik (dot product) dua buah vektor. Semakin nilai $\cos\theta$ mendekati 1 , semakin mirip pula dua vektor kata yang dibandingkan.

$$\mathbf{Q} \cdot \mathbf{D} = \|\mathbf{Q}\|\|\mathbf{D}\| \cos\theta \quad \Rightarrow \quad sim(\mathbf{Q}, \mathbf{D}) = \cos\theta = \frac{\mathbf{Q} \cdot \mathbf{D}}{\|\mathbf{Q}\|\|\mathbf{D}\|}$$

Gambar 2.3.1 Rumus cosine similarity
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf

2.5. Koreksi Kata

Koreksi kata adalah proses atau teknik untuk memperbaiki kesalahan penulisan kata secara otomatis. Kesalahan tersebut bisa berupa salah ketik , kesalahan tata bahasa, atau ketidaksesuaian dengan kata baku dalam suatu bahasa. Teknologi koreksi kata banyak digunakan dalam aplikasi seperti pengolah kata, mesin pencari, perangkat komunikasi, dan media sosial untuk meningkatkan kualitas teks yang dihasilkan pengguna.

Koreksi kata bekerja melalui beberapa tahapan seperti pendataan kata yang akan menjadi referensi, perhitungan kemiripan , dan penampilan koreksi terbaik.

III. IMPLEMENTASI

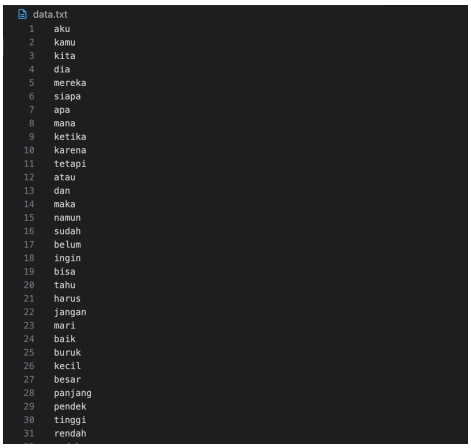
Program ini dibuat menggunakan bahasa Python, yang merupakan salah satu bahasa pemrograman yang umum digunakan dalam pengolahan data secara matematis. Python dipilih karena dianggap sesuai untuk mengimplementasikan fitur koreksi kata. Program ini terdiri dari tiga *file* utama, yaitu main.py, koreksi.py, dan data.txt.

Koreksi.py berisi kumpulan fungsi dan prosedur yang diperlukan untuk melakukan koreksi kata sedangkan *file* main.py berisi simulasi program koreksi kata dalam bentuk *command line interface* (CLI) dan data.txt berisi kumpulan kata dalam bahasa Indonesia yang diguanakan sebagai referensi kata yang dianggap benar sebagai acuan koreksi.

Proses koreksi kata dalam implementasi ini dilakukan secara terstruktur. Langkah pertama adalah membaca data dari *file* data.txt. Pembacaan dilakukan dengan sederhana, yaitu membaca setiap baris dalam *file* data.txt dan menyimpannya ke dalam sebuah array.

```python
# Menyiapkan dataset
with open("data.txt", "r") as file:
    dataset = file.read().splitlines()
```

Gambar 3.1 Pengisian dataset dari file data.txt
Sumber : Dokumen penulis



Gambar 3.2 Contoh sebagian kata pada file data.txt
Sumber : Dokumen penulis

Kumpulan kata yang telah dimasukkan ke dalam array dataset kemudian diproses satu per satu untuk diubah menjadi vektor. Proses pertama yang dilakukan adalah memecah setiap kata menjadi bigram, yaitu pasangan dua karakter yang berurutan. Pemisahan ini dilakukan dengan mengiterasi setiap karakter dalam sebuah kata dan menyusun dua karakter berurutan ke dalam sebuah array.

```python
# Fungsi untuk membangun bigram dari sebuah kata
def createBigram(word):
    return [word[i:i+2] for i in range(len(word) - 1)]
```

Gambar 3.3 Fungsi untuk membuat bigram kata
Sumber : Dokumen penulis

Setelah pasangan bigram dibuat untuk setiap kata dalam array, langkah selanjutnya adalah membentuk vektor kata dengan menghitung jumlah kemunculan setiap bigram dalam sebuah kata. Proses ini dilakukan dengan menganalisis setiap bigram dan mencatat frekuensi kemunculannya, sehingga menghasilkan representasi vektor yang mencerminkan pola bigram dalam kata tersebut. Setiap vektor kata disusun berdasarkan jumlah kemunculan seluruh bigram yang ada dalam dataset, memastikan bahwa seluruh kata memiliki representasi vektor dengan ruang dimensi yang seragam.

```python
# Fungsi untuk membangun vektor bigram dari dataset
def createVektorKata(dataset):
    word_bigrams = {}
    allBigram = set()  # Untuk menyimpan semua bigram unik

    for word in dataset:
        bigrams = createBigram(word)
        word_bigrams[word] = bigrams
        allBigram.update(bigrams)  # Menambah bigram unik ke set

    # Mengonversi bigram unik ke list untuk digunakan dalam vektor
    allBigram = sorted(list(allBigram))
    bigramIdx = {bigram: idx for idx, bigram in enumerate(allBigram)}

    # Membuat vektor bigram untuk setiap kata
    vektorDataset = {}
    for word, bigrams in word_bigrams.items():
        vector = np.zeros(len(allBigram), dtype=int)
        for bigram in bigrams:
            if bigram in bigramIdx:
                vector[bigramIdx[bigram]] += 1
        vektorDataset[word] = vector
    return vektorDataset, bigramIdx
```

Gambar 3.4 Fungsi untuk membuat vektor kata dataset
Sumber : Dokumen penulis

Fungsi createVektorKata akan menghasilkan variabel bigramIdx, yaitu sebuah array yang berisi kumpulan bigram unik beserta indeksnya yang muncul dalam dataset. Variabel ini memastikan bahwa setiap vektor kata memiliki representasi bigram yang terorganisasi pada posisi yang sama dalam ruang vektor. Dengan kata lain, posisi bigram dalam setiap vektor kata konsisten untuk seluruh dataset. Fungsi ini juga mengembalikan variabel vektorDataset, yang merupakan kumpulan vektor kata yang telah diproses berdasarkan frekuensi kemunculan bigram.

Selanjutnya, query kata akan diproses menjadi vektor kata dengan metode yang serupa dengan proses pada dataset. Perbedaannya adalah bahwa proses ini hanya diterapkan pada satu kata, yaitu query yang dimasukkan. Fungsi ini menerima masukan berupa query kata dan bigramIdx yang telah dibuat pada tahap sebelumnya. Dengan menggunakan bigramIdx, fungsi ini memastikan bahwa query kata memiliki representasi vektor yang sesuai dengan struktur ruang vektor dataset.

```python
# Fungsi untuk memproses query dan menghitung vektor bigram
def createVektorQuery(query, bigramIdx):
    # membangun bigram kata
    query_bigrams = createBigram(query)

    # menghitung jumlah kemunculan bigram
    bigram_counts = defaultdict(int)
    for bigram in query_bigrams:
        bigram_counts[bigram] += 1

    # membuat vektor kata
    query_vector = np.zeros(len(bigramIdx), dtype=int)
    for bigram, count in bigram_counts.items():
        if bigram in bigramIdx:
            query_vector[bigramIdx[bigram]] = count
        else:
            print(f"PERINGATAN: Bigram '{bigram}' tidak ditemukan pada bigramIdx")
    return query_vector
```

Gambar 3.5 Fungsi untuk membuat vektor kata query
Sumber : Dokumen penulis

Setelah vektor terbentuk, sekarang menuju proses akhir yaitu perbandingan antara query dengan dataset. Proses perbandingan dilakukan dengan menggunakan formula *cosine similarity*. Fungsi ini digunakan untuk mencari kedekatan anatara dua vektor kata. Fungsi ini menerima dua vektor dan akan mengembalikan nilai kedekatan kedua

vektor tersebut.

```python
# Fungsi untuk menghitung cosine similarity
def cosineSimilarity(vec1, vec2):
    dot_product = np.dot(vec1, vec2)
    norm1 = np.linalg.norm(vec1)
    norm2 = np.linalg.norm(vec2)
    return dot_product / (norm1 * norm2)
```

Gambar 3.6 Fungsi Cosine Similarity
Sumber : Dokumen penulis

Untuk meningkatkan tingkat akurasi dalam proses koreksi kata, diterapkan suatu metode pembobotan. Metode ini memberikan bobot yang lebih tinggi kepada setiap kata yang memiliki panjang yang serupa, dibandingkan dengan kata yang memiliki panjang yang berbeda. Dengan cara ini, kata-kata yang memiliki kesamaan panjang dianggap lebih relevan, sehingga dapat meningkatkan ketepatan hasil koreksi yang dilakukan.

```python
# Fungsi untuk menghitung panjang kata sebagai bobot
def lengthWeight(query, word):
    len_query = len(query)
    len_word = len(word)

    # Jika panjang kata dalam rentang len_query-1, len_query, atau len_query+1
    if abs(len_query - len_word) <= 2:
        return 1  # Bobot yang sama jika panjang kata sama atau hanya berbeda 1
    else:
        return 0.5  # Bobot 0.5 jika panjang kata lebih dari 1 karakter berbeda
```

Gambar 3.7 Fungsi perhitungan bobot panjang kata
Sumber : Dokumen penulis

Setelah mengembangkan fungsi untuk menghitung similaritas dan melakukan pembobotan berdasarkan panjang kata, langkah berikutnya adalah membuat sebuah fungsi gabungan. Fungsi gabungan dirancang untuk menerima tiga input, yaitu query, vektorDataset, dan bigramIdx. Kemudian, fungsi akan memproses ketiga input tersebut dan mengembalikan lima kata dari dataset yang memiliki tingkat similaritas paling tinggi dengan query yang diberikan. Fungsi bertujuan untuk memberikan hasil yang paling relevan berdasarkan perhitungan similaritas dan pembobotan yang telah dilakukan sebelumnya.

```python
# Fungsi untuk mencari 5 koreksi terbaik
def koreksiKata(query, vektorDataset, bigramIdx):
    vektorQuery = createVektorQuery(query, bigramIdx)
    similarities = []
    for word, word_vector in vektorDataset.items():
        # Menghitung cosine similarity
        similarity = cosineSimilarity(vektorQuery, word_vector)
        # Menghitung bobot panjang kata
        length_factor = lengthWeight(query, word)
        # Menggabungkan cosine similarity dan bobot panjang kata
        weighted_similarity = similarity * length_factor
        similarities.append((word, weighted_similarity))
    # Mengurutkan berdasarkan similarity dan mengambil 5 hasil teratas
    similarities.sort(key=lambda x: x[1], reverse=True)
    return similarities[:5]
```

Gambar 3.8 Fungsi koreksi kata
Sumber : Dokumen penulis

File main.py berisi implementasi dari sistem koreksi kata yang akan menerima input berupa query kata dari pengguna. Program kemudian akan memproses query yang diberikan untuk mencari kata-kata dalam dataset yang memiliki tingkat similaritas tertinggi dengan kata yang di-query. Setelah melakukan perhitungan similaritas dan pembobotan, program akan menampilkan lima kata yang paling relevan, berdasarkan nilai similaritas tertinggi dengan query yang dimasukkan.

```python
from koreksi import *
import time

# Menyiapkan dataset
with open("data.txt", "r") as file:
    dataset = file.read().splitlines()

# Membangun vektor bigram untuk dataset
startTime = time.time()
vektorDataset, allBigram = createVektorKata(dataset)
print(f"\n\033[1;32mVektor kata selesai dibangun dalam {time.time() - startTime:.2f} detik.\033[0m")
# Loop untuk input query
while True:
    query = input("\n\033[1;34mMasukkan Kata (atau '\033[1;31mexit\033[1;34m' untuk keluar): \033[0m").strip()
    if query.lower() == 'exit':
        print("\033[1;31mProgram selesai.\033[0m")
        break

    if not query:
        print("\033[1;31mInput tidak boleh kosong.\033[0m")
        continue

    # Mencari 5 koreksi terbaik
    startTime = time.time()
    topKoreksi = koreksiKata(query, vektorDataset, allBigram)
    elapsedTime = time.time() - startTime

    # Menampilkan hasil dalam bentuk tabel sederhana
    if topKoreksi:
        print("\n\033[1;33mTop 5 Kata:\033[0m")
        print("+----+----------------+-----------------+")
        print("| No | Koreksi        | Skor Similarity |")
        print("+----+----------------+-----------------+")
        for i, (word, similarity) in enumerate(topKoreksi, start=1):
            print(f"| {i:<2} | {word:<14} | {similarity:<17.4f} |")
        print("+----+----------------+-----------------+")
    else:
        print("\033[1;31mTidak ada koreksi yang ditemukan.\033[0m")

    print(f"\n\033[1;32mWaktu yang dibutuhkan: {elapsedTime:.4f} detik\033[0m")
```

Gambar 3.9 main.py
Sumber : Dokumen penulis

IV. HASIL DAN PEMBAHASAN

Hasil pengujian diperoleh dari serangkaian pengujian kasus yang mencakup berbagai skenario yang mungkin terjadi dalam proses koreksi kata. Proses pengujian dilakukan untuk menguji keakuratan dan efisiensi fitur koreksi kata dalam berbagai kondisi. Pada sistem koreksi kata seperti yang dirancang, pengguna memiliki kemampuan untuk mengoreksi kata yang salah ketik atau tidak sesuai, sehingga menghasilkan kata yang relevan dengan konteks. Pengguna cukup memasukkan kata yang ingin diperbaiki, dan sistem akan memberikan beberapa saran kata berdasarkan tingkat kemiripan.

Pengujian dilakukan dengan mengisi file data.txt yang akan digunakan sebagai referensi. Untuk pengujian ini, diambil dataset berupa 14557 kata bahasa indonesia yang didapat dari https://github.com/damzaky/kumpulan-kata-bahasa-indonesia-KBBI .

data.txt

```
1  a-beta
2  abad
3  abadi
4  abahan
5  abakus
6  abal
7  abalalu
8  abam
9  abar
10 abdi
11 abdusen
12 abioseston
13 abir
14 abiti
15 abjad
16 ablasi
17 ablatif
```

Gambar 4.1 data.txt
Sumber : Dokumen penulis

Program ini dimulai dengan membaca *file* data.txt yang berisi dataset kata-kata. Setelah itu, program akan memproses data tersebut untuk membentuk vektor dataset, yang terdiri dari kumpulan vektor kata. Proses pembentukan vektor ini melibatkan representasi kata-kata dalam bentuk vektor numerik, yang memungkinkan perhitungan similaritas dilakukan dengan lebih efektif. Setelah vektor dataset terbentuk, program akan menampilkan waktu yang dibutuhkan untuk proses pembentukan tersebut, memberikan informasi mengenai efisiensi waktu dalam menjalankan langkah awal sistem koreksi kata.

```
Vektor kata selesai dibangun dalam 0.04 detik.
```

Gambar 4.2 Pengujian waktu pembangunan vektor dataset
Sumber : Dokumen penulis

Setelah proses pembentukan vektor dataset selesai, program akan meminta masukan dari pengguna berupa kata yang ingin dikoreksi. Kata yang dimasukkan oleh pengguna ini akan diproses menjadi vektor query. Vektor query ini kemudian akan dibandingkan dengan vektor-vektor yang ada dalam dataset menggunakan perhitungan similaritas. Dengan membandingkan vektor query dengan vektor dataset, program akan menentukan kata-kata yang paling mirip dengan kata yang dimasukkan oleh pengguna, yang nantinya akan ditampilkan sebagai hasil koreksi.

```
Vektor kata selesai dibangun dalam 0.04 detik.
Masukkan Kata (atau 'exit' untuk keluar): █
```

Gambar 4.2 Pengujian masukkan kata oleh pengguna
Sumber : Dokumen penulis

Setelah proses input kata dan pembentukan vektor query selesai, ujicoba fitur koreksi kata akan dilakukan dengan memasukkan kata yang memiliki kesalahan penulisan. Kata pertama yang diuji adalah "defnsiens", yang merupakan kata yang salah ketik. Program akan memproses kata tersebut dan membandingkannya dengan vektor-vektor pada dataset. Berdasarkan perhitungan similaritas, hasil yang diharapkan adalah kata "defisiensi", yang merupakan koreksi yang tepat untuk kata yang salah ketik tersebut. Program akan menampilkan lima kata dengan similaritas tertinggi, dan diharapkan "defisiensi" menjadi salah satu dari hasil koreksi yang muncul.

```
Masukkan Kata (atau 'exit' untuk keluar): defnsiens

Top 5 Kata:
+----+------------------+-------------------+
| No | Koreksi          | Skor Similarity   |
+----+------------------+-------------------+
| 1  | defisiensi       | 0.7628            |
| 2  | dependensi       | 0.6139            |
| 3  | menstensil       | 0.6139            |
| 4  | resiliensi       | 0.5721            |
| 5  | tendensius       | 0.5721            |
+----+------------------+-------------------+

Waktu yang dibutuhkan: 0.0788 detik
```

Gambar 4.3 Pengujian query "defnsiens" pada program koreksi kata
Sumber : Dokumen penulis

Program menghasilkan beberapa opsi kata yang memiliki kemiripan dengan "defnsiens". Kata-kata tersebut antara lain defisiensi, dependensi, menstesnsil, resiliensi, dan tendensius. Kemiripan tertinggi terdapat pada kata defisiensi dengan skor similaritas 0.7628. Skor ini diperoleh dengan membandingkan vektor yang dibentuk dari kata "defnsiens" dengan vektor kata lain dalam dataset.

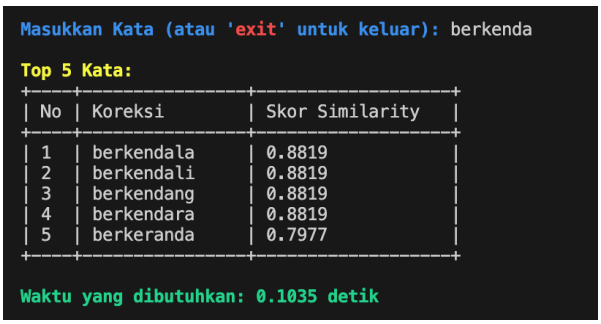
```
Vektor defnsiens = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
```

Gambar 4.4 Vektor kata defnsiens
Sumber : Dokumen penulis

```
Vektor defisiensi =[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
```

Gambar 4.5 Vektor kata defisiensi
Sumber : Dokumen penulis

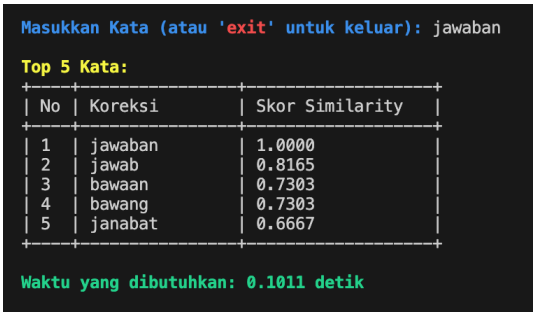
Pengujian selanjutnya akan dilakukan dengan memasukkan query kata yang memiliki kesalahan dalam penulisan. Kata yang akan diuji adalah "berkenda", yang diharapkan memiliki koreksi menjadi kata yang lebih tepat atau lanjutan dari kata tersebut. Program akan memproses kata ini, membandingkannya dengan vektor-vektor yang ada dalam dataset, dan memberikan hasil yang paling mirip. Kata yang diharapkan muncul sebagai hasil koreksi adalah kata yang sesuai dengan konteks "berkenda", seperti "berkendara" atau kata lain yang relevan, tergantung pada hasil perhitungan similaritas yang dihasilkan oleh program.



Gambar 4.6 Pengujian query "berkenda" pada program koreksi kata
Sumber : Dokumen penulis

Program menghasilkan beberapa opsi kata yang memiliki kemiripan dengan "berkenda". Terdapat empat kata yang memiliki skor similaritas yang sama, yaitu berkendala, berkendali, berkendang, dan berkendara. Keempat kata ini memiliki skor similaritas yang identik karena hanya satu bigram yang berbeda antara kata-kata tersebut dengan query. Perbedaan kecil pada satu bigram inilah yang menyebabkan skor similaritas bigram menjadi sama untuk keempat kata tersebut, meskipun kata-kata tersebut memiliki makna yang berbeda.

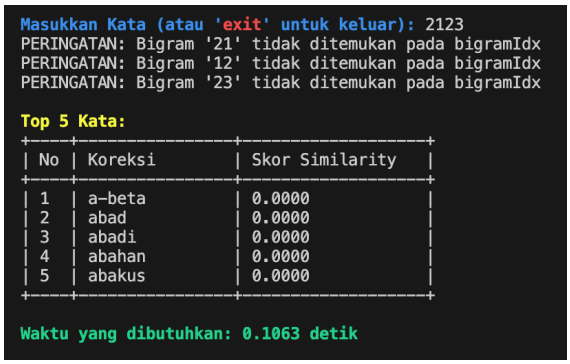
Pengujian selanjutnya dilakukan dengan memasukkan query kata yang sudah ada dalam dataset. Pada pengujian kali ini, kata yang dimasukkan adalah "jawaban", dan diharapkan program menghasilkan kata yang serupa dengan skor similaritas bernilai 1.0000. Hal ini diharapkan terjadi karena kata "jawaban" sudah terdapat dalam dataset, sehingga vektor yang dibentuk dari kata tersebut akan memiliki kecocokan sempurna dengan dirinya sendiri, menghasilkan skor similaritas tertinggi.



Gambar 4.7 Pengujian query "jawaban" pada program koreksi kata
Sumber : Dokumen penulis

Pada hasil kemiripan kata, kata "jawaban" menempati posisi pertama dengan skor similaritas 1.0000. Hal ini terjadi karena vektor yang dibentuk dari query "jawaban" identik dengan vektor yang dibuat oleh kata "jawaban" dalam dataset. Sebagai akibatnya, perhitungan similaritas menghasilkan skor sempurna, yaitu 1.0000, yang menunjukkan kecocokan penuh antara vektor query dan vektor kata yang ada dalam dataset.

Pengujian selanjutnya dilakukan dengan memasukkan query berupa angka. Query yang dimasukkan adalah "2123". Query ini bukan merupakan kata yang valid, melainkan angka yang dimasukkan untuk menguji bagaimana program menangani masukan berupa angka. Pengujian ini bertujuan untuk memastikan bahwa program dapat menangani input non-kata dengan baik, baik dengan memberikan penanganan khusus atau memberikan respons yang sesuai jika input tersebut tidak dapat diproses dalam konteks sistem koreksi kata.



Gambar 4.8 Pengujian query "2123" pada program koreksi kata
Sumber : Dokumen penulis

Program memberikan hasil berupa peringatan bahwa bigram tidak ditemukan. Hal ini terjadi karena tidak ada kata dalam dataset yang menghasilkan bigram '21', '12', dan '23' yang terbentuk dari query "2123". Karena tidak adanya kecocokan bigram dalam dataset, perhitungan similaritas menghasilkan skor similarity bernilai 0.0000, yang menunjukkan bahwa tidak ada kemiripan antara query angka tersebut dengan kata-kata yang ada di dataset. Peringatan ini menandakan bahwa program tidak dapat memproses angka sebagai kata yang dapat dibandingkan dalam konteks koreksi kata.

Pengujian terakhir dilakukan dengan skenario di mana pengguna tidak memasukkan query apapun ke dalam program. Dalam hal ini, program diharapkan memberikan respons yang sesuai, seperti menampilkan pesan peringatan atau meminta pengguna untuk memasukkan query yang valid.

Gambar 4.8 Pengujian query kosong pada program koreksi kata
Sumber : Dokumen penulis

V. KESIMPULAN DAN SARAN

Implementasi sistem koreksi kata berbasis vektor di ruang eucledian dan cosine similarity menunjukkan kemampuan yang baik dalam mengoreksi kesalahan penulisan kata bahasa Indonesia dengan memanfaatkan representasi bigram dan perhitungan tingkat kemiripan antar kata menggunakan *cosine similarity*. Sistem ini relevan dalam menangkap urutan karakter yang salah ketik, memberikan hasil koreksi yang cukup akurat, dan bekerja dengan dataset yang tersedia. Namun program ini

masih menggunakan cara sederhana dan masih dapat dikembangkan lebih lanjut.

VI. Ucapan Terimakasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya yang melimpah dalam proses penulisan makalah ini sehingga dapat diselesaikan tepat waktu. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Rila Mandala, sebagai dosen mata kuliah Aljabar Linear dan Geometri K01. Penulis juga berterima kasih kepada orang tua dan keluarga yang selalu memberikan dukungan moril dan spiritual, serta menjadi sumber inspirasi dan motivasi selama proses penulisan makalah. Semoga makalah ini dapat memberikan manfaat bagi pembaca dan menjadi kontribusi yang berarti dalam bidang terkait.

VII. Lampiran

https://github.com/AlfianHanifFY/KoreksiKata---ALGEO-IF2123.git
(*link* github program koreksi kata)
https://github.com/damzaky/kumpulan-kata-bahasa-indonesia-KBBI
(*link* github dataset pengujian koreksi kata)

References

[1] Samanta, P. dan Chaudhuri, B. B., "A simple real-word error detection and correction using local word bigram and trigram," [Online]. Tersedia: https://aclanthology.org/O13-1022.pdf. [Diakses pada 25 Desember 2024].

[2] Munir, Rinaldi, "Vektor di Ruang Euclidean (Bagian 1)," Bahan kuliah IF2123 Aljabar Linier dan Geometri, [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf. [Diakses pada 26 Desember 2024].

[3] Bekkerman, R. dan Allan, J., "Using Bigrams in Text Categorization," Department of Computer Science, University of Massachusetts Amherst, [Online]. Tersedia: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0af293cfa79fd83bbb120360acf6bf56673563f4. [Diakses pada 26 Desember 2024].

[4] Farid, "Apa itu N-Gram, Bigram, Trigram, dan Seterusnya?," [Online]. Tersedia: https://catatanfarid.wordpress.com/2016/01/28/apa-itu-n-gram-bigram-trigram-dan-seterusnya/. [Diakses pada 2 Januari 2025].

[5] Munir, Rinaldi, "Aplikasi Dot Product pada Sistem Temu-balik Informasi," Bahan kuliah IF2123 Aljabar Linier dan Geometri, [Online].
Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf. [Diakses pada 2 Januari 2025].

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

Alfian Hanif Fitria Yustanto 13523073