

TELAAH PUSTAKA

PERHITUNGAN MATRIKS JARAK PADA DOKUMEN TEKS

DARI TUGAS AKHIR

KOMPARASI AKURASI KLASIFIKASI DATA TEKS
MENGGUNAKAN JARAK MANHATTAN DAN JARAK
JACCARD PADA ALGORITMA KNN

KOMPETENSI STATISTIKA



LUH SUKMA MULYANI
2108541027

PROGRAM STUDI MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN
ALAM
UNIVERSITAS UDAYANA
BUKIT JIMBARAN
2025

LEMBAR PENGESAHAN TELAAH PUSTAKA

Judul : Perhitungan Matriks Jarak pada Dokumen Teks
Kompetensi : Statistika
Nama : Luli Sukma Mulyani
NIM : 2108541027
Tanggal Ujian : 5 Agustus 2025

Disetujui oleh:

Pembimbing II



Ni Ketut Tari Tastrawati, S.Si., M.Si.

NIP 197405282002122002

Pembimbing I



Ir. I Komang Gde Sukarasa, M.Si

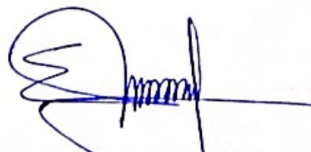
NIP 196501051991031004

Mengetahui:

Komisi Tugas Akhir

Program Studi Matematika FMIPA Unud

Ketua,



I Wayan Sumarjaya, S.Si., M.Stats.

NIP 197704212005011001

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Ida Sang Hyang Widhi Wasa karena berkat rahmat-Nya, penulis dapat menyelesaikan telaah pustaka yang berjudul "Perhitungan Matriks Jarak pada Dokumen Teks" dari tugas akhir yang berjudul "Komparasi Akurasi Klasifikasi Data Teks Menggunakan Jarak Manhattan dan Jarak Jaccard pada Algoritma KNN" tepat pada waktunya. Penyusunan telaah pustaka ini dapat berjalan dengan lancar karena adanya dukungan dan bimbingan dari berbagai pihak baik secara langsung maupun tidak langsung. Oleh karena itu, penulis tidak lupa mengucapkan terima kasih kepada:

1. Ibu I Gusti Ayu Made Srinadi, S.Si., M.Si. selaku Koordinator Program Studi Matematika FMIPA Universitas Udayana.
2. Bapak I Wayan Sumarjaya, S.Si., M.Stats. selaku Ketua Komisi Tugas Akhir Program Studi Matematika FMIPA Universitas Udayana.
3. Dosen Pembimbing I, Bapak Ir. I Komang Gde Sukarasa, M.Si. yang telah membimbing dengan sabar dalam penyusunan tugas akhir ini.
4. Dosen Pembimbing II, Ibu Ni Ketut Tari Tastrawati, S.Si., M.Si. yang telah banyak memberikan dukungan dan arahan.
5. Seluruh dosen Program Studi Matematika FMIPA Universitas Udayana.
6. Keluarga tercinta atas motivasi dan dukungan moril maupun materil.
7. Teman-teman seperjuangan yang telah memberikan semangat dan dukungan dalam penyusunan tugas akhir ini.

Penulis menyadari bahwa penyusunan telaah pustaka ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa mendatang.

Bukit Jimbaran, 21 Mei 2025

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN TELAAH PUSTAKA.....	i
KATA PENGANTAR.....	ii
DAFTAR ISI.....	v
DAFTAR TABEL.....	vi
DAFTAR GAMBAR.....	vii
BAB I PENDAHULUAN.....	1
BAB II TINJAUAN PUSTAKA.....	2
2.1 <i>Text Mining dan Document Similarity</i>	2
2.2 <i>Text Preprocessing</i>	3
2.2.1 <i>Lower Case</i>	4
2.2.2 <i>Cleaning Data</i> (Pembersihan karakter dan <i>Punctuational Removal</i>)	4
2.2.3 Tokenisasi	7
2.2.4 Normalisasi	8
2.2.5 <i>Stopwords Removal</i>	8
2.2.6 <i>Stemming</i>	9
2.3 Representasi Dokumen Teks	10
2.3.1 <i>Bag of Words</i>	10
2.3.2 <i>Term Frequency-Invers Document Frequency</i> (TF-IDF)	11
2.4 Himpunan	14
2.4.1 Operasi Dasar pada Himpunan	15
2.5 Matriks	16
2.5.1 Kolom dan Baris pada Matriks	17
2.5.2 Penjumlahan Matriks	17
2.5.3 Perkalian Skalar dan Matriks	18

2.5.4	Transpos Matriks	19
2.5.5	<i>Document Term</i> Matriks	20
2.6	Vektor	22
2.6.1	Operasi Vektor Pada \mathbb{R}^n	23
2.6.2	<i>Norm, Dot Product</i> dan Jarak pada \mathbb{R}^n	25
2.7	Metrik Jarak	29
2.7.1	Jarak Euclidean	30
2.7.2	Jarak Manhattan	31
2.7.3	Jarak Jaccard	32
2.7.4	<i>Cosine Similarity dan Cosine Distance</i>	33
2.8	Matriks Jarak	35
2.9	Studi Kasus	36
2.9.1	Pra-pemrosesan Teks	36
2.9.2	Representasi Dokumen Teks	38
2.9.3	Perhitungan Matriks Jarak	44
BAB III KESIMPULAN.....		55
3.1	Kesimpulan	55
DAFTAR PUSTAKA.....		56

DAFTAR TABEL

2.1	Contoh Data Setelah Proses <i>Lower Case</i>	36
2.2	Contoh Data Setelah Proses <i>Cleaning Data</i>	37
2.3	Contoh Data Setelah Proses Tokenisasi	37
2.4	Contoh Data Setelah Proses Normalisasi	38
2.5	Contoh Data Setelah Proses <i>Stopwords Removal</i>	38
2.6	Contoh Data Setelah Proses <i>Stemming</i>	38
2.7	Bag of Words Features Matrix	39
2.8	Contoh Kata untuk Perhitungan TF-IDF	40
2.9	<i>Term Frequency</i> (TF) Tiap Dokumen	40
2.10	Contoh Hasil Perhitungan <i>Inverse Document Frequency</i> (IDF)	42
2.11	Nilai Bobot Fitur per Dokumen Sebelum Normalisasi	42
2.12	Nilai Bobot Fitur per Dokumen Setelah Normalisasi	43
3.1	Contoh Data Komentar	57

DAFTAR GAMBAR

2.1	Contoh representasi <i>Bag of Words</i> dari sebuah kalimat.	11
2.2	Ilustrasi Vektor \overrightarrow{AB}	23
2.3	Penjumlahan Vektor	24
2.4	Pengurangan Vektor	25
2.5	<i>Norm</i> Vektor	26
2.6	Panjang Vektor $P_1\vec{P}_2$ adalah d	27
2.7	Ilustrasi Vektor \mathbf{u} , \mathbf{v} , dan $\mathbf{v} - \mathbf{u}$	28
2.8	Ilustrasi Jarak Euclidean	30
2.9	Ilustrasi Jarak Manhattan	31
2.10	Ilustrasi Jaccard Similarity	33
2.11	Ilustrasi <i>Cosine Similarity</i> sebagai sudut antara dua vektor . . .	34

BAB I

PENDAHULUAN

Perhitungan matriks jarak pada dokumen teks merupakan salah satu langkah penting dalam analisis data berbasis teks. Matriks jarak digunakan untuk menggambarkan hubungan antar dokumen dalam bentuk numerik, sehingga memudahkan dalam proses klasifikasi, klusterisasi, atau pencarian informasi. Perhitungan matriks jarak dilakukan dengan merepresentasikan dokumen ke dalam bentuk vektor di ruang fitur, kemudian menghitung tingkat kesamaan atau perbedaan antar vektor tersebut. Ada berbagai teknik dalam melakukan representasi teks antara lain Bag-of-Words, TF-IDF, atau *embedding*.

Setelah melakukan representasi teks ke bentuk numerik, selanjutnya akan dilakukan perhitungan matriks jarak dengan berbagai metode pengukuran jarak seperti Euclidean, Manhattan, Minkowski, Chebyshev, Jaccard, Cosine dan lain sebagainya untuk menghitung seberapa mirip atau berbeda-beda masing-masing dokumen. Pemilihan jarak yang digunakan disesuaikan dengan karakteristik data teks dan tujuan analisisnya. Dalam pengolahan dokumen teks, penggunaan matriks jarak membantu dalam memahami struktur data dan mempermudah dalam proses pengelompokan dokumen berdasarkan kemiripan isi. Selain itu, perhitungan matriks jarak juga berperan dalam meningkatkan hasil akurasi dengan memilih jarak yang sesuai.

BAB II

TINJAUAN PUSTAKA

2.1 *Text Mining dan Document Similarity*

Text mining merupakan salah satu cabang dari data mining yang berfokus pada proses ekstraksi informasi atau pengetahuan tersembunyi dari kumpulan data berbentuk teks yang tidak terstruktur. Menurut Jo (2018), *text mining* didefinisikan sebagai proses untuk mengekstrak pengetahuan yang tidak secara eksplisit tersedia dalam data teks dengan tujuan menghasilkan informasi baru yang dapat digunakan secara langsung dalam pengambilan keputusan. Tugas utama dari *text mining* mencakup klasifikasi, klusterisasi, dan asosiasi yang semuanya memerlukan pengukuran kesamaan antar dokumen. Salah satu konsep fundamental dalam *text mining* adalah *document similarity* yang mengukur tingkat kesamaan atau perbedaan antar dokumen berdasarkan konten tekstualnya (Aggarwal, 2018). Meskipun tidak selalu mencerminkan makna secara semantik, pengukuran ini memungkinkan komputer menilai kedekatan antar dokumen berdasarkan fitur representasi teks, dan menjadi komponen penting dalam berbagai aplikasi seperti sistem rekomendasi, deteksi plagiarisme, dan *information retrieval*.

Pada konteks penelitian ini, *document similarity* atau *document distance* digunakan untuk menyusun matriks jarak, yaitu suatu matriks yang elemennya menunjukkan tingkat kesamaan atau perbedaan antara pasangan dokumen. Namun, untuk dapat mengimplementasikan pengukuran *similarity* ini, dokumen teks harus terlebih dahulu dikonversi dari bentuk asli yang tidak terstruktur menjadi representasi yang dapat diproses secara komputasional melalui proses *text preprocessing*. Oleh karena itu, *text preprocessing* berperan

penting dalam menyusun representasi teks yang bersih dan siap digunakan untuk perhitungan similarity yang selanjutnya dinyatakan dalam bentuk matriks jarak.

2.2 *Text Preprocessing*

Prapemrosesan teks bertujuan untuk menyusun data teks yang awalnya tidak terstruktur menjadi bentuk yang lebih teratur dan mudah dianalisis. Dalam teks sering kali terdapat elemen-elemen yang tidak relevan seperti tag atau tautan, kesalahan ejaan, kata-kata umum yang kurang penting seperti “a”, “an”, dan “the”. Selain itu, variasi kata karena perubahan bentuk kata (seperti plural atau waktu) serta kesalahan ejaan juga dapat memengaruhi analisis teks. Oleh karena itu, prapemrosesan sangat penting dilakukan untuk mempersiapkan teks sebelum dianalisis. Oleh karena itu, prapemrosesan sangat penting dilakukan untuk mempersiapkan teks sebelum dianalisis. Dengan langkah-langkah prapemrosesan meliputi (Aggarwal, 2018):

1. *Case Folding (Lower Case)*: menyesuaikan huruf kapital menjadi huruf kecil untuk konsistensi.
2. *Cleaning Data* (Pembersihan karakter dan *Punctuational Removal*): menangani tanda baca (seperti tanda hubung) dengan hati-hati agar tidak mempengaruhi proses tokenisasi.
3. Tokenisasi: proses memecah teks menjadi unit-unit kata atau token.
4. Normalisasi: proses mengubah kata tidak baku menjadi bentuk baku.
5. *Stop Words Removal*: menghapus kata-kata umum seperti “dan”, “yang”, atau “adalah” karena tidak memberikan nilai yang informatif yang signifikan.

6. *stemming*: untuk menyatukan berbagai variasi kata yang berasal dari akar yang sama.

Setelah proses pra-pemrosesan, dokumen direpresentasikan sebagai matriks istilah dokumen yang jarang (*sparse*) dengan ukuran $n \times d$, di mana n merupakan jumlah dokumen dan d adalah jumlah kata.

2.2.1 *Lower Case*

Proses ini mengubah seluruh huruf dalam teks menjadi huruf kecil untuk menyamakan bentuk kata yang berbeda karena kapitalisasi. Misalnya, “*Happy*” dan “*happy*” akan dianggap sama. Pada implementasinya, proses ini dilakukan dengan menggunakan metode `.str.lower()` pada objek *series* di Python. Sebagai contoh, kode

```
df_copy['content_lowercase'] = df_copy['CONTENT'].str.lower()
```

Baris kode di atas akan menghasilkan kolom baru bernama `content_lowercase` pada data frame `df_copy`, yang berisi hasil konversi semua teks dari kolom `CONTENT` menjadi huruf kecil.

2.2.2 *Cleaning Data (Pembersihan karakter dan Punctuational Removal)*

Pada tahapan ini dilakukan pembersihan karakter khusus yang tidak relevan yang dapat mengganggu proses tokenisasi dan analisis. Tujuan utama dari pembersihan ini adalah untuk menghilangkan “*noise*” atau gangguan dalam teks, sehingga data yang dihasilkan menjadi bersih dan siap diproses pada tahap tokenisasi dan analisis lanjutan.

Proses pembersihan dilakukan menggunakan fungsi `remove noise()`

dan `combine_emojis()` yang terdiri atas beberapa tahapan. Langkah pertama dimulai dengan mengurai entitas HTML seperti `&`, `<`, dan `>` menjadi karakter normal menggunakan `html.unescape`, agar makna asli teks tetap dipertahankan. Selanjutnya, tag HTML dihapus menggunakan ekspresi reguler, begitu pula karakter Unicode tersembunyi atau karakter BOM. Teks kemudian dinormalisasi dari bentuk *full width* (yang umum ditemukan pada karakter Asia) ke bentuk ASCII standar menggunakan `unicodedata.normalize`, agar konsistensi karakter terjaga.

Untuk melindungi informasi sensitif, alamat email digantikan dengan token `EMAILADDRESS`. Selanjutnya, dilakukan penggabungan domain website yang ditulis terpisah, misalnya `nama . com` menjadi `nama.com`, agar dapat dideteksi sebagai satu kesatuan `url`. Deteksi `url` juga dilakukan secara menyeluruh, baik yang ditulis dalam bentuk eksplisit seperti `https://example.com` maupun yang tersamar seperti `bit.ly/abc234` dikonversi menjadi satu bentuk standar `url`. Begitu juga nama domain seperti `youtube . com`, `abc . net` diganti menjadi token `url`.

Tahapan selanjutnya adalah pemrosesan emoji dilakukan melalui fungsi `combine_emojis()`, pertama-tama menambahkan spasi di antara emoji agar dapat terdeteksi sebagai token terpisah. Emoji kemudian diubah ke dalam bentuk deskriptif seperti `smile`. Jika beberapa emoji muncul secara berurutan maka penamaan tersebut juga dipisahkan untuk menjaga keterbacaan. Angka-angka dalam berbagai bentuk seperti bilangan besar, desimal, angka ordinal atau notasi ilmiah diseragamkan dengan token `numeric`, dan apabila token `numeric` muncul secara berulang, hanya disimpan satu kali saja.

Tanda kurung siku seperti `you[tube]` dihapus tanpa menghilangkan isinya. Simbol-simbol khusus yang tidak bermakna secara linguistik seperti `#`, dan karakter yang diulang berlebihan disederhanakan menjadi satu karak-

ter atau dihapus. Proses ini juga menjaga apostrof yang berada dalam kata seperti `don't` agar tidak memecah makna kata. Pembersihan juga mencakup penggabungan huruf yang terpisah strip seperti “d-d-d” menjadi “ddd”, serta huruf yang dipisah spasi seperti “p e a c e” menjadi “peace”. Semua karakter yang bukan huruf, angka, atau spasi termasuk tanda baca seperti titik, koma, dan simbol dihilangkan. Simbol-simbol yang berulang seperti `!!!` disederhanakan menjadi satu karakter. Terakhir, spasi tambahan dibersihkan sehingga teks menjadi rapi dan seragam.

Proses ini dilakukan dengan bantuan fungsi `re.sub()` dari pustaka `re` (*regular expressions*) di Python. Fungsi ini memiliki peran penting dalam mengganti bagian teks yang cocok dengan pola tertentu. Sintaks dasar penggunaannya adalah sebagai berikut:

```
re.sub(pattern, replacement, string)
```

1. `pattern`: pola regex yang ingin dicari dalam string.
2. `replacement`: teks pengganti.
3. `string`: teks yang akan diproses.

Contoh penggunaan:

```
re.sub(r'\b[\w.-]+@[ \w.-]+\.\w+\b', 'email', text)
```

Kode di atas mencari semua bentuk email dan menggantinya dengan token `email`. Penggunaan `re.sub()` memungkinkan pencocokan dan manipulasi teks yang kompleks secara efisien, seperti mendeteksi url, tag HTML, angka, dan simbol yang tidak perlu. Dengan serangkaian tahapan ini, data teks yang sebelumnya tidak terstruktur dan mengandung banyak gangguan dapat dibersihkan menjadi bentuk yang lebih konsisten dan siap untuk proses analisis lanjutan.

2.2.3 Tokenisasi

Tokenisasi merupakan tahapan krusial dalam pemrosesan bahasa alami. Token didefinisikan sebagai unit-unit teks bermakna. Berikut adalah contoh komentar:

“i check back often to help reach 2×10^9 views and I avoid watching Baby”

dapat diuraikan sebagai berikut: “i”, “check”, “back”, “often”, “to”, “help”, “reach”, “ 2×10^9 ”, “views”, “and”, “I”, “avoid”, “watching”, dan “baby”. Adapun tantangan dari tokenisasi, yaitu pada saat terdapat struktur teks yang tidak baku seperti url, ekspresi matematika, atau karakter unicode yang tidak standar.

Dalam penelitian ini, proses tokenisasi akan dilakukan setelah teks dibersihkan dan dikonversi ke bentuk urutan karakter. Untuk melakukan tokenisasi, digunakan pustaka python `nltk` yang menyediakan fungsi `word_tokenize`. Berikut adalah contoh implementasi sederhana untuk tokenisasi menggunakan `nltk`:

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
def word_tokenization(text):
    return word_tokenize(text)
```

Pada kode tersebut, fungsi `word_tokenize` akan memecah teks input menjadi token-token berupa kata atau simbol yang bermakna. Misalnya, jika diberikan kalimat sebagai input, fungsi ini akan mengembalikan daftar kata-kata yang sudah dipisahkan. Token-token yang diperoleh dari tahap ini menjadi dasar dalam representasi teks untuk analisis atau pemodelan berikutnya.

2.2.4 Normalisasi

Setelah data teks melalui proses *lower case*, *cleaning*, dan tokenisasi, langkah selanjutnya adalah normalisasi kata, yaitu proses mengubah kata-kata tidak baku seperti slang, singkatan, atau kesalahan ketik (*typos*), menjadi bentuk kata yang baku dan konsisten (Aleqabie et al., 2024). Normalisasi merupakan tahapan krusial dalam pemrosesan teks informal, seperti komentar pengguna pada platform YouTube, yang banyak mengandung variasi penulisan yang tidak sesuai kaidah bahasa standar. Dalam penelitian ini, proses normalisasi dilakukan menggunakan kamus normalisasi yang dikembangkan secara manual oleh peneliti. Kamus ini disusun berdasarkan observasi langsung terhadap data yang digunakan, dengan mencatat bentuk-bentuk kata tidak baku yang sering muncul, lalu menentukan padanan kata bakunya secara kontekstual. Contoh normalisasi yang dilakukan antara lain "*u*" menjadi "*you*", "*luv*" menjadi "*love*", dan "*plzz*" menjadi "*please*".

Normalisasi dilakukan setelah tokenisasi agar setiap token dapat diperiksa dan dimodifikasi secara individual sesuai kamus. Tujuan dari tahapan ini adalah meningkatkan konsistensi representasi teks serta mengurangi dimensi fitur unik, yang pada akhirnya akan memperbaiki performa dan efisiensi pada tahapan *vectorization* maupun *text classification*. Normalisasi kata juga membantu mengurangi *sparsity* data dan meningkatkan akurasi dalam pemodelan teks berbasis *machine learning*.

2.2.5 Stopwords Removal

Stopwords adalah kata umum dalam bahasa yang kurang memiliki nilai pembeda dalam klasifikasi teks, seperti ("*the*", "*is*", "*in*"). Frekuensi kemunculan kata yang cenderung seragam di berbagai kategori topik. Oleh karena itu,

penghapusan *stopwords* kerap dilakukan untuk meningkatkan efisiensi pemrosesan data dalam analisis.

Pada proses ini, penulis akan menggunakan bantuan pustaka *Natural Language Toolkit* (NLTK) untuk menghapus *stopwords* dari data teks. Berikut adalah contoh implementasi sederhana untuk *stopwords*:

```

nltk.download('stopwords')

def remove_stopwords(text):
    stops = set(stopwords.words("english"))
    if isinstance(text, list):
        text = [w for w in text if w.lower() not in stops]
    return text

```

Fungsi tersebut dimulai dengan mengunduh daftar *stopwords* dalam bahasa Inggris melalui `nltk.download('stopwords')`. Selanjutnya, untuk setiap elemen dalam list input, kata-kata yang termasuk dalam daftar *stopwords* akan dihapus. Dengan cara ini, hanya kata-kata yang dianggap mengandung informasi penting yang dipertahankan, sehingga data teks yang dihasilkan lebih bersih dan siap digunakan pada analisis berikutnya.

2.2.6 *Stemming*

Stemming merupakan proses penggabungan kata-kata yang memiliki akar yang sama. Misalnya, bentuk tunggal, jamak, atau berbagai *tense* dari suatu kata disatukan karena tidak mengubah makna semantisnya dalam konteks *text mining*. Contohnya, kata “*run*”, “*running*”, “*runs*”, dan “*ran*” seluruhnya berasal dari akar kata “*run*” dan sebaliknya dikonsolidasikan menjadi satu istilah.

Dalam penelitian ini, proses *stemming* dilakukan menggunakan pusta-

ka *Natural Language toolkit* (NLTK) dengan metode *Snowball Stemmer* untuk bahasa inggris. Berikut merupakan code yang digunakan:

```
# Mendownload stemmer untuk bahasa Inggris
stemmer = SnowballStemmer("english")

# Fungsi untuk stemming setiap kata dalam list
def stemmed_wrapper(document):
    return [stemmer.stem(term) for term in document]
```

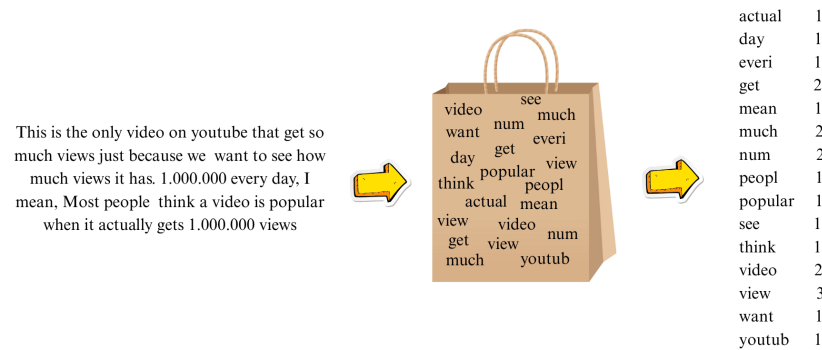
Kode di atas dimulai dengan inisialisasi objek `SnowballStemmer` untuk bahasa inggris. Selanjutnya, dibuat sebuah fungsi bernama `stemmed_wrapper` yang menerima masukan berupa dokumen dalam bentuk daftar kata, kemudian mengembalikan daftar baru berisi kata-kata yang telah melalui proses *stemming*. Dengan demikian, semua bentuk turunan dari suatu kata dapat dikembalikan ke bentuk dasarnya, sehingga membantu mengurangi dimensi dan meningkatkan efisiensi teks selanjutnya.

2.3 Representasi Dokumen Teks

2.3.1 *Bag of Words*

Bag of Words merupakan salah satu pendekatan dasar yang digunakan dalam representasi teks ke dalam bentuk numerik dalam bidang pemrosesan bahasa alami (*Natural Language Processing*). BoW merepresentasikan sebuah dokumen sebagai kumpulan kata-kata, tanpa memperhatikan tata urutan kata dan struktur gramatikal. Dalam model ini, informasi yang dipertahankan hanyalah frekuensi kemunculan kata dalam dokumen (Lane et al., 2019)

Sebagai ilustrasi, Gambar 2.1 menunjukkan bagaimana pendekatan BoW bekerja. Dalam gambar tersebut, kalimat lengkap tidak dianalisis berdasarkan



Gambar 2.1 Contoh representasi *Bag of Words* dari sebuah kalimat.

susunan atau struktur katanya. Sebaliknya, setiap kata yang muncul dari keseluruhan kutipan dihitung berdasarkan jumlah kemunculannya. Misalnya, kata “*view*” muncul sebanyak tiga kali, “*video*”, dan “*get*” masing-masing muncul dua kali, sedangkan kata-kata lain seperti “*actual*”, “*day*”, “*everi*”, “*mean*”, “*much*”, “*num*”, “*peopl*”, “*popular*”, “*see*”, “*think*”, “*want*”, dan “*youtub*” muncul satu kali. Representasi ini ditampilkan dalam bentuk daftar kata disertai frekuensinya dan digunakan untuk mempermudah analisis tanpa mempertimbangkan tata urutan kalimat.

2.3.2 *Term Frequency-Invers Document Frequency (TF-IDF)*

Term Frequency-Invers Document Frequency merupakan salah satu metode representasi teks yang banyak digunakan dalam *text mining* dan *information retrieval* yang digunakan untuk mengukur tingkat kepentingan suatu kata terhadap sebuah dokumen dalam suatu kumpulan dokumen (korpus). Bobot suatu kata dalam dokumen dihitung dari dua komponen utama, yaitu *Term Frequency* (TF) dan *Invers Document Frequency* (IDF) (Lane et al., 2019). *Term Frequency* mengukur seberapa sering suatu kata muncul dalam sebuah dokumen, dan *Invers Document Frequency* mengukur seberapa jarang kata muncul di seluruh dokumen (korpus). Berikut adalah tahapan dalam

melakukan pembobotan kata menggunakan TF-IDF:

Term Frequency (TF) digunakan untuk mengukur seberapa sering suatu kata *term* ke-*i* muncul dalam dokumen *j* relatif terhadap total jumlah kata dalam dokumen tersebut. Nilai TF memberikan informasi mengenai tingkat kepentingan suatu kata dalam sebuah dokumen tertentu. Rumus perhitungan TF adalah sebagai berikut:

$$tf_{ij} = \frac{f_{ij}}{\sum_{k=1}^n f_{kj}} \quad (2.1)$$

dengan, f_{ij} adalah jumlah kemunculan kata ke-*i* dalam dokumen ke-*j*, dan $\sum_{k=1}^n f_{kj}$ adalah total jumlah kata dalam dokumen ke-*j*. Semakin sering suatu kata muncul dalam dokumen, maka semakin tinggi nilai TF nya, yang menandakan bahwa kata tersebut lebih penting dalam dokumen tersebut.

Invers Document Frequency (IDF) digunakan untuk mengukur seberapa penting suatu kata dalam keseluruhan kumpulan dokumen (korpus). IDF memberikan bobot yang lebih tinggi pada kata-kata yang jarang muncul di banyak dokumen, karena kata-kata tersebut dianggap lebih unik dan informatif. Rumus perhitungan IDF adalah sebagai berikut:

$$idf_i = \log \left(\frac{N}{df_i} \right) \quad (2.2)$$

dengan, N adalah jumlah total dokumen dalam korpus, dan df_i adalah jumlah dokumen yang mengandung kata ke-*i*. Jika sebuah kata muncul di hampir semua dokumen, maka nilai IDF-nya akan rendah karena dianggap umum. Sebaliknya, jika hanya sedikit dokumen yang mengandung kata tersebut, nilai IDF akan tinggi.

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan ha-

sil perkalian antara nilai TF dan IDF. Nilai ini digunakan untuk mengukur tingkat kepentingan suatu kata dalam sebuah dokumen relatif terhadap keseluruhan dokumen dalam korpus. TF-IDF memberikan bobot yang tinggi pada kata-kata yang sering muncul dalam sebuah dokumen tetapi jarang ditemukan di dokumen lainnya. Rumus perhitungan TF-IDF adalah sebagai berikut:

$$w_{ij} = tf_{ij} \times idf_i \quad (2.3)$$

dengan, w_{ij} adalah bobot TF-IDF untuk kata ke- i dalam dokumen ke- j , tf_{ij} adalah nilai *Term Frequency* dari kata ke- i dalam dokumen ke- j , dan idf_i adalah nilai *Inverse Document Frequency* dari kata ke- i .

Dalam penelitian ini, proses pembobotan TF-IDF diimplementasikan dengan menggunakan pustaka `scikit-learn` (Raschka et al., 2022). Dimana implementasi TF-IDF dalam `scikit-learn` memiliki perbedaan formula dari TF-IDF standar. Pusta ini menyediakan dua kelas utama, yaitu `TfidfTransformer`, yang mengubah frekuensi kemunculan kata mentah menjadi bobot TF-IDF, dan `TfidfVectorizer`, yang secara otomatis menangani proses tokenisasi, perhitungan TF, serta transformasi IDF dalam satu langkah. Rumus IDF yang digunakan oleh *scikit-learn* adalah:

$$idf(t) = \log \left(\frac{1 + n_d}{1 + df(t)} \right) + 1 \quad (2.4)$$

dimana n_d adalah jumlah total dokumen, dan $df(t)$ adalah jumlah dokumen yang memuat kata ke- t . Penambahan angka 1 pada pembilang dan penyebut dilakukan untuk mencegah pembagian nol. Selanjutnya, bobot TF-IDF dihitung dengan cara mengalikan nilai `tf` dengan `idf`:

$$w(t, d) = tf(t, d) \times idf(t) \quad (2.5)$$

Selain itu, `TfidfTransformer` juga menerapkan normalisasi L2 terhadap vektor hasil TF-IDF. Normalisasi ini mengubah vektor asli menjadi vektor dengan panjang satu dengan cara membaginya terhadap norma L2:

$$v_{\text{norm}} = \frac{v}{\|v\|_2} \quad (2.6)$$

Normalisasi ini penting agar panjang dokumen tidak mempengaruhi perhitungan jarak dalam algoritma seperti K Nearest Neighbors (KNN), sehingga yang lebih diperhatikan adalah bobot relatif kata, bukan jumlah katanya. Dengan pendekatan ini, TF-IDF menghasilkan representasi numerik teks yang efisien, di mana setiap dokumen diubah menjadi vektor berdimensi tinggi. Vektor-vektor tersebut selanjutnya digunakan sebagai masukan utama dalam algoritma klasifikasi KNN untuk mengukur kesamaan antar dokumen, dengan membandingkan performa menggunakan beberapa metrik jarak seperti Manhattan dan Jaccard, yang akan dianalisis lebih lanjut pada penelitian ini.

2.4 Himpunan

Himpunan adalah kumpulan objek yang terdefinisi dengan jelas, di mana setiap objek disebut sebagai anggota atau elemen himpunan tersebut (Rosen, 2000). Notasi standar yang digunakan untuk menyatakan keanggotaan adalah $x \in A$, yang berarti objek x adalah anggota dari himpunan A , dan sebaliknya $x \notin A$, berarti x bukan anggota dari A . Himpunan dapat direpresentasikan dengan beberapa cara. Pertama, dengan Metode Roster, yaitu mendaftarkan semua elemen himpunan di dalam kurung kurawal, misalnya, $S = \{a_1, a_2, \dots, a_n\}$. Kedua, dengan Predikat Pendefinisi, yaitu mendefinisikan himpunan dalam bentuk $S = \{x \mid P(x)\}$, yang berarti S adalah himpunan semua objek x (dalam domain yang relevan) sedemikian sehingga predikat

$P(x)$ bernilai benar. Sebuah himpunan yang tidak memiliki elemen sama sekali disebut himpunan kosong (*null set* atau *empty set*), yang dinotasikan dengan \emptyset atau $\{\}$.

Setelah sebuah himpunan didefinisikan, seringkali penting untuk mengetahui “ukurannya”, sebuah konsep yang diformalkan sebagai kardinalitas. Rosen (2000) mendefinisikan kardinalitas $|S|$ dari sebuah himpunan hingga S sebagai jumlah elemen dalam S . Dengan kata lain, kardinalitas adalah jumlah elemen unik yang terkandung dalam suatu himpunan. Sebagai contoh, jika himpunan $A = \text{jeruk, manggis, anggur}$, maka kardinalitas A , dapat ditulis $|A| = 3$

2.4.1 Operasi Dasar pada Himpunan

Irisan (*Intersection*)

Irisan dua himpunan, A dan B , didefinisikan sebagai himpunan

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$

Defini ini menyatakan bahwa himpunan $A \cap B$ berisi semua elemen dari kedua himpunan A dan himpunan B

Gabungan Himpunan (*Union*)

Gabungan dari dua himpunan, A dan B , didefinisikan sebagai himpunan

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

Berdasarkan definisi tersebut himpunan $A \cup B$ berisi semua elemen x yang merupakan bagian anggota dari himpunan A , atau anggota dari himpunan B ,

atau anggota dari keduanya.

2.5 Matriks

Matriks merupakan susunan bilangan yang ditata dalam bentuk persegi panjang dan ditempatkan di dalam tanda kurung siku (Boyd & Vandenberghe, 2018). Salah satu aspek penting dalam sebuah matriks adalah ukurannya atau dimensinya, yang menunjukkan jumlah baris dan kolom penyusunannya. Sebagai contoh:

$$A = \begin{bmatrix} 2 & 3 & 5 \\ 5 & 1 & 4 \end{bmatrix}$$

Matriks A terdiri atas dua baris dan tiga kolom, sehingga dimensinya adalah 2×3 . Umumnya, sebuah matriks dengan m baris dan n kolom disebut matriks berukuran $m \times n$. Setiap elemen dalam matriks disebut entri, yang terletak pada posisi baris ke- i dan kolom ke- j , serta dinyatakan sebagai A_{ij} , yang berarti elemen pada baris ke- i dan kolom ke- j dari matriks A .

Jika suatu matriks memiliki jumlah baris yang sama dengan jumlah kolom, maka matriks tersebut disebut sebagai matriks persegi. Matriks persegi berukuran $n \times n$ dikatakan memiliki ordo n . Contoh berikut menggambarkan dua matriks persegi:

$$B = \begin{bmatrix} 4 & 1 \\ 5 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} 3 & 1 & 4 \\ 5 & 2 & 9 \\ 2 & 4 & 1 \end{bmatrix}$$

Matriks B memiliki ukuran 2×2 , sedangkan matriks C berukuran 3×3 . Karena jumlah baris dan kolom pada masing-masing matriks tersebut sama, keduanya termasuk dalam kategori matriks persegi.

2.5.1 Kolom dan Baris pada Matriks

Dalam representasi matriks blok, sebuah matriks berukuran $m \times n$ dapat dituliskan sebagai susunan blok yang terdiri atas satu baris dan n kolom. Dalam hal ini, matriks A dapat dinyatakan sebagai:

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

di mana setiap a_j merupakan vektor kolom berdimensi $m \times 1$, yaitu kolom ke- j dari matriks A . Dengan kata lain, sebuah matriks berukuran $m \times n$ dapat dipandang sebagai gabungan dari n vektor kolom.

Sebaliknya, matriks yang sama juga dapat direpresentasikan sebagai matriks blok dengan satu kolom dan m baris blok:

$$A = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

di mana setiap b_i merupakan vektor baris berdimensi $1 \times n$, yaitu baris ke- i dari matriks A . Dalam bentuk ini, matriks dipandang sebagai kumpulan vektor baris yang disusun secara vertikal sebanyak m baris.

2.5.2 Penjumlahan Matriks

Dua buah matriks dapat dijumlahkan apabila memiliki dimensi yang sama, artinya jumlah baris dan kolom dari kedua matriks tersebut harus identik. Proses penjumlahan dilakukan dengan cara menjumlahkan elemen-elemen yang posisinya bersesuaian pada masing-masing matriks. Sebagai ilustrasi, mi-

salkan terdapat dua matriks A dan B , masing-masing berukuran 3×2 , yaitu:

$$A = \begin{bmatrix} 4 & 1 \\ 5 & 2 \\ 3 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 7 \\ 4 & 1 \\ 2 & 8 \end{bmatrix}$$

Penjumlahan antara matriks A dan B dilakukan dengan menjumlahkan elemen pada posisi yang sama dari kedua matriks, sehingga diperoleh:

$$A + B = \begin{bmatrix} 4+3 & 1+7 \\ 5+4 & 2+1 \\ 3+2 & 6+8 \end{bmatrix} = \begin{bmatrix} 7 & 8 \\ 9 & 3 \\ 5 & 14 \end{bmatrix}$$

Dengan demikian, hasil penjumlahan dua matriks berdimensi sama adalah matriks baru yang memiliki dimensi serupa, di mana setiap elemennya merupakan hasil penjumlahan dari elemen-elemen yang bersesuaian.

2.5.3 Perkalian Skalar dan Matriks

Perkalian skalar pada matriks memiliki konsep yang mirip dengan perkalian skalar pada vektor, yakni setiap elemen dalam matriks akan dikalikan dengan suatu nilai skalar tertentu. Jika diberikan sebuah matriks A dan skalar k , maka hasil perkalian skalar kA adalah sebuah matriks baru yang setiap elemennya merupakan hasil perkalian antara elemen matriks A dengan skalar k . Sebagai contoh, misalkan terdapat skalar $k = -3$ dan matriks

$$A = \begin{bmatrix} 1 & 4 \\ 5 & 3 \\ 6 & -2 \end{bmatrix}$$

maka hasil dari kA adalah

$$-3 \times A = \begin{bmatrix} (-3) \times 1 & (-3) \times 4 \\ (-3) \times 5 & (-3) \times 3 \\ (-3) \times 6 & (-3) \times -2 \end{bmatrix} = \begin{bmatrix} -3 & -12 \\ -15 & -9 \\ -18 & 6 \end{bmatrix}$$

Dengan demikian, setiap elemen matriks asli dikalikan dengan nilai skalar sehingga membentuk matriks hasil baru dengan ukuran yang sama.

2.5.4 Transpos Matriks

Jika sebuah matriks A berukuran $m \times n$, maka transpos dari matriks tersebut, yang dilambangkan dengan A^T , merupakan matriks baru berukuran $n \times m$. Elemen-elemen dari matriks A^T diperoleh dengan menukar posisi baris dan kolom dari matriks A , sehingga elemen (i, j) pada A^T sama dengan elemen (j, i) pada A . Dengan kata lain, proses transpose dilakukan dengan memutar posisi baris menjadi kolom dan kolom menjadi baris. Sebagai ilustrasi, misalkan diberikan matriks:

$$A = \begin{bmatrix} 2 & 1 & 4 \\ 3 & 6 & 1 \end{bmatrix}$$

Maka transpos dari matriks A adalah:

$$A^T = \begin{bmatrix} 2 & 3 \\ 1 & 6 \\ 4 & 1 \end{bmatrix}$$

Transpos matriks juga berfungsi untuk mengubah vektor baris menjadi vektor kolom, dan sebaliknya. Notasi a^T sering digunakan untuk menyatakan

an transpos dari vektor kolom a , sehingga menghasilkan sebuah vektor baris. sebagai contoh, baris kedua dari matriks A diatas, yaitu

$$A = \begin{bmatrix} 2 & 1 & 4 \\ 3 & 6 & 1 \end{bmatrix}$$

dapat direpresentasikan sebagai vektor kolom transpos, yakni $(3, 6, 1)^T$.

2.5.5 *Document Term Matriks*

Dalam analisis teks, sebuah korpus terdiri atas N dokumen dapat di-representasikan dalam bentuk matriks dokumen kata. Matriks ini merupakan matriks berdimensi $A_{N \times n}$, di mana n menyatakan jumlah kosakata (kata unik) yang terdapat dalam kamus yang digunakan. Setiap elemen A_{ij} pada matriks tersebut menunjukkan frekuensi kemunculan kata ke- i pada dokumen ke- j .

Setiap baris dalam matriks dokumen kata merepresentasikan satu dokumen dalam bentuk vektor frekuensi kata. Dengan demikian, baris-baris dalam matriks tersebut adalah $a_1^T, a_2^T, \dots, a_N^T$, yang masing-masing menggambarkan sebaran kata dalam dokumen 1 hingga N . Tidak hanya baris, kolom-kolom dalam matriks dokumen kata juga memuat informasi penting. Kolom ke- i menyatakan distribusi kemunculan kata ke- i di seluruh dokumen dalam korpus. Kolom ini berbentuk vektor berdimensi N , yang memperlihatkan bagaimana kata tersebut tersebar pada setiap dokumen. Sebagai ilustrasi, misalkan terdapat korpus yang terdiri atas tiga dokumen dan empat kata dalam kamus,

maka matriks dokumen kata dapat dinyatakan sebagai berikut:

$$A = \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 3 & 0 & 2 \\ 1 & 2 & 0 & 0 \end{bmatrix}$$

Penjelasan dari matriks diatas adalah, sebagai berikut:

1. Baris pertama $(2, 0, 1, 0)$ menunjukkan bahwa dokumen pertama mengandung kata pertama sebanyak dua kali, kata ke tiga sebanyak satu kali.
2. Baris kedua $(0, 3, 0, 2)$ menunjukkan bahwa dokumen kedua mengandung kata kedua sebanyak tiga kali dan kata keempat sebanyak dua kali.
3. Baris ketiga $(1, 2, 0, 0)$ menunjukkan bahwa dokumen ketiga mengandung kata pertama sebanyak satu kali dan kata ke dua sebanyak dua kali.

Adapun informasi yang ditampilkan oleh kolom-kolom dalam matriks tersebut adalah:

1. Kolom pertama $(2, 0, 1)$ menunjukkan bahwa kata pertama muncul dua kali di dokumen pertama, tidak muncul di dokumen kedua, dan muncul satu kali di dokumen ketiga.
2. Kolom kedua $(0, 3, 2)$ menunjukkan bahwa kata kedua tidak muncul di dokumen pertama, muncul tiga kali pada dokumen kedua, dan muncul sebanyak dua kali pada dokumen ketiga.
3. Kolom ketiga $(1, 0, 0)$ menunjukkan bahwa kata ketiga hanya muncul satu kali pada dokumen pertama, dan tidak muncul pada dua dokumen lainnya.

4. Kolom keempat $(0, 2, 0)$ menunjukkan bahwa kata keempat hanya muncul dua kali di dokumen kedua, dan tidak muncul pada dua dokumen lainnya.

Melalui representasi matriks dokumen kata seperti ini, proses analisis teks dan eksplorasi data teks menjadi lebih sistematis dan terarah. Matriks tersebut memungkinkan identifikasi pola kemunculan kata, keterkaitan antar dokumen, serta informasi lainnya yang relevan dalam pengolahan data teks.

2.6 Vektor

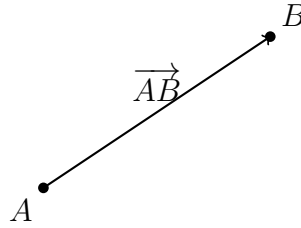
Skalar merupakan besaran yang hanya memiliki nilai saja, tanpa melibatkan arah (Anton & Rorres, 2014). Karena tidak memiliki orientasi, skalar dianggap sebagai besaran yang hanya menunjukkan seberapa besar sesuatu itu. Dalam praktiknya, skalar sering digunakan sebagai konstanta dalam operasi-operasi matematika seperti penjumlahan, pengurangan, perkalian, dan pembagian.

Sementara itu, vektor adalah besaran yang memiliki dua sifat utama, yaitu nilai dan arah. Dalam matematika, vektor biasanya digambarkan sebagai garis lurus yang menunjukkan arah dan panjang tertentu. Notasi untuk vektor umumnya menggunakan huruf tebal seperti **a**, **b**, **v**, **w**, **x**, sedangkan skalar dituliskan dengan huruf miring seperti *a*, *k*, *v*, atau *w*. Jika ingin menyatakan suatu vektor dengan titik awal di *A* dan titik akhir di *B*, maka notasinya dapat dituliskan sebagai:

$$\mathbf{v} = \overrightarrow{AB}$$

Visualisasi dari vektor tersebut dapat digambarkan sebagai berikut

Di mana arah panah menunjukkan arah dari titik *A* menuju titik *B*, dan panjang garis menunjukkan besaran dari vektor tersebut.

Gambar 2.2 Ilustrasi Vektor \vec{AB} .

2.6.1 Operasi Vektor Pada \mathbb{R}^n

Secara umum, \mathbb{R}^n merujuk pada ruang vektor berdimensi n , yang dapat dibayangkan sebagai ruang dengan n sumbu atau arah. Setiap vektor \mathbf{v} yang berada dalam ruang ini dapat dituliskan dalam bentuk:

$$\mathbf{v} = (v_1, v_2, \dots, v_n)$$

Operasi-operasi yang berlaku pada vektor dalam \mathbb{R}^n secara prinsip serupa dengan operasi yang terdapat pada ruang berdimensi tiga, yaitu \mathbb{R}^3 , namun diperluas agar mencakup dimensi yang lebih tinggi. Beberapa jenis operasi yang umum dilakukan baik di \mathbb{R}^n dan di \mathbb{R}^3 antara lain:

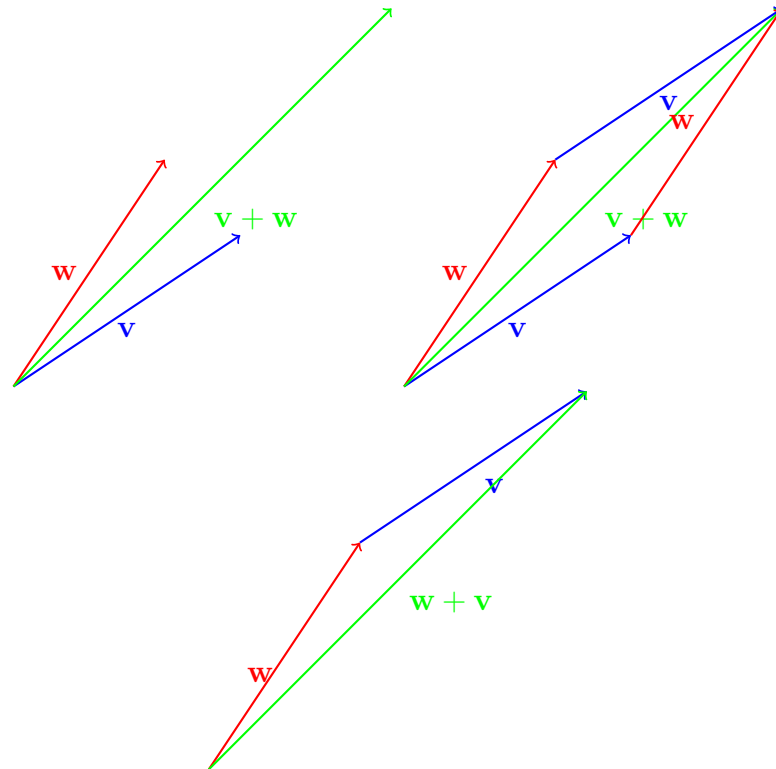
Penjumlahan Vektor

Dua buah vektor dalam ruang berdimensi n dapat dijumlahkan dengan cara menjumlahkan setiap komponen yang bersesuaian dari kedua vektor tersebut. Jika $\mathbf{v} = (v_1, v_2, \dots, v_n)$ dan $\mathbf{w} = (w_1, w_2, \dots, w_n)$, maka:

$$\mathbf{v} + \mathbf{w} = (v_1 + w_1, v_2 + w_2, \dots, v_n + w_n)$$

Dalam konteks geometris, terdapat apa yang disebut aturan segitiga dalam penjumlahan vektor. Aturan ini menyatakan bahwa jika dua buah vektor \mathbf{v} dan \mathbf{w} berada dalam \mathbb{R}^2 atau \mathbb{R}^3 , dan disusun sedemikian rupa sehingga

ujung vektor \mathbf{v} menjadi titik pangkal vektor \mathbf{w} , maka hasil penjumlahan $\mathbf{v} + \mathbf{w}$ dapat direpresentasikan oleh sebuah panah yang menghubungkan titik awal \mathbf{v} langsung ke ujung \mathbf{w} .



Gambar 2.3 Penjumlahan Vektor

Selain sebagai proses penjumlahan, operasi ini juga dapat dipahami sebagai proses *translasi titik*. Translasi merupakan suatu transformasi geometri yang memindahkan posisi suatu titik atau objek dalam ruang ke lokasi baru, tanpa mengubah bentuk, orientasi, maupun ukuran. Dengan kata lain, translasi menggeser titik-titik dalam ruang secara sejajar dan searah sesuai arah serta besar vektor yang digunakan.

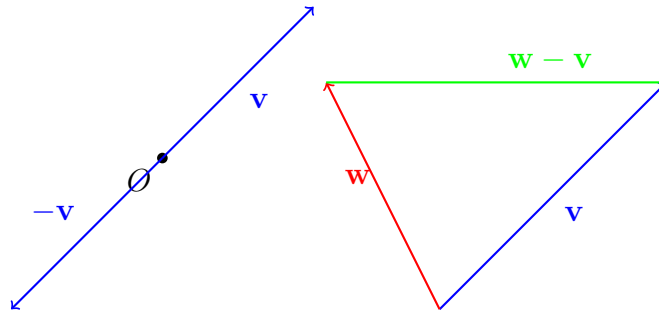
Pengurangan Vektor

Operasi pengurangan antara dua vektor, yang biasa dilambangkan dengan $\mathbf{w} - \mathbf{v}$, pada dasarnya merupakan penjumlahan antara vektor \mathbf{w} dengan

lawan dari vektor \mathbf{v} . Vektor negatif $-\mathbf{v}$ merupakan vektor yang memiliki besar (magnitudo) yang sama dengan \mathbf{v} , tetapi dengan arah yang berlawanan. Oleh karena itu, pengurangan $\mathbf{w} - \mathbf{v}$ dapat dipahami sebagai penjumlahan $\mathbf{w} + (-\mathbf{v})$. Pada ruang vektor berdimensi n , termasuk ruang dua dimensi (\mathbb{R}^2) maupun tiga dimensi (\mathbb{R}^3), pengurangan dilakukan dengan cara mengurangi komponen-komponen yang bersesuaian dari kedua vektor. Secara umum, pengurangan antara dua vektor $\mathbf{v} = (v_1, v_2, \dots, v_n)$ dan $\mathbf{w} = (w_1, w_2, \dots, w_n)$ dituliskan sebagai berikut:

$$\mathbf{v} - \mathbf{w} = (\mathbf{v}_1 - \mathbf{w}_1, \mathbf{v}_2 - \mathbf{w}_2, \dots, \mathbf{v}_n - \mathbf{w}_n)$$

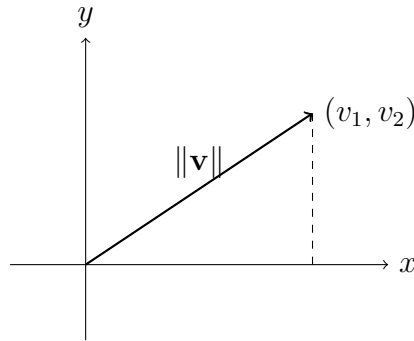
Sebagai ilustrasi, proses pengurangan dua vektor pada ruang dua dimensi (\mathbb{R}^2) dapat divisualisasikan dalam Gambar 2.4, yang menunjukkan bagaimana arah dan besar dari vektor hasil pengurangan diperoleh berdasarkan posisi relatif dari kedua vektor.



Gambar 2.4 Pengurangan Vektor

2.6.2 *Norm, Dot Product* dan Jarak pada \mathbb{R}^n

Panjang suatu vektor \mathbf{v} , yang juga dikenal sebagai norma, magnitudo, atau besar vektor, dilambangkan dengan $\|\mathbf{v}\|$.

Gambar 2.5 *Norm* Vektor

Untuk ruang berdimensi dua (\mathbb{R}^2), norma vektor dapat dihitung menggunakan Teorema Pythagoras sebagai berikut:

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

Pada ruang tiga dimensi (\mathbb{R}^3), besar vektor $\|\mathbf{v}\| = (v_1, v_2, v_3)$ dihitung dengan rumus:

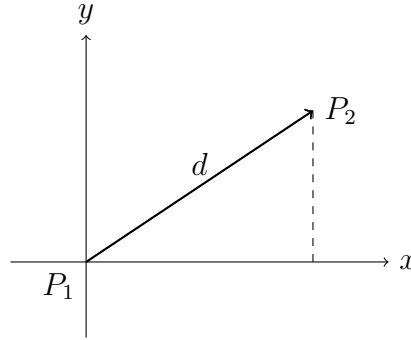
$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + v_3^2}$$

Secara umum, untuk vektor dalam ruang berdimensi n (\mathbb{R}^n), normanya ditentukan melalui:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

Sebuah vektor dikatakan sebagai vektor satuan apabila memiliki norm yang bernilai satu. Vektor jenis ini digunakan untuk menunjukkan arah suatu vektor tanpa memperhatikan panjangnya. Misalnya, jika \mathbf{v} adalah vektor pada \mathbb{R}^2 atau \mathbb{R}^3 dengan panjang 2, maka $\frac{1}{2}\mathbf{v}$ merupakan vektor satuan yang searah dengan \mathbf{v} . Secara umum, vektor satuan \mathbf{u} yang searah dengan \mathbf{v} diperoleh melalui proses normalisasi, yaitu:

$$\mathbf{u} = \frac{1}{\|\mathbf{v}\|} \mathbf{v}$$



Gambar 2.6 Panjang Vektor $P_1\vec{P}_2$ adalah d

Jika terdapat dua titik $P_1(x_1, y_1)$ dan $P_2(x_2, y_2)$ dalam \mathbb{R}^2 , maka panjang dari vektor $P_1\vec{P}_2$ (yakni jarak antara kedua titik tersebut) dapat dihitung menggunakan rumus:

$$d = \|P_1\vec{P}_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Secara lebih umum, jika $\mathbf{u} = (u_1, u_2, \dots, u_n)$ dan $\mathbf{v} = (v_1, v_2, \dots, v_n)$ adalah dua titik dalam \mathbb{R}^n , maka jarak antara keduanya dinyatakan sebagai:

$$d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \dots + (u_n - v_n)^2}$$

Perkalian Titik (*Dot Product*)

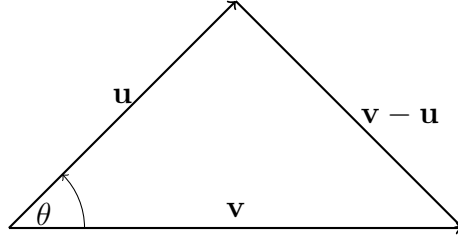
Operasi penting lainnya dalam analisis vektor adalah perkalian titik atau *dot product*. Untuk dua vektor \mathbf{u} dan \mathbf{v} di \mathbb{R}^2 , *dot product* didefinisikan sebagai:

$$\mathbf{u} \cdot \mathbf{v} = u_1v_1 + u_2v_2 \quad (2.7)$$

Jika \mathbf{u} dan \mathbf{v} adalah vektor dalam \mathbb{R}^n , maka *dot product*nya adalah:

$$\mathbf{u} \cdot \mathbf{v} = u_1v_1 + u_2v_2 + \cdots + u_nv_n = \sum_{i=1}^n u_iv_i$$

Dot product juga memiliki representasi geometris yang berkaitan dengan sudut θ antara dua vektor. Jika kedua vektor \mathbf{u} dan \mathbf{v} diposisikan sehingga titik awalnya sama, maka berdasarkan hukum cosinus diperoleh:



Gambar 2.7 Ilustrasi Vektor \mathbf{u} , \mathbf{v} , dan $\mathbf{v} - \mathbf{u}$.

$$\|\mathbf{v} - \mathbf{u}\|^2 = \|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - 2\|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$$

Jika rumus tersebut dikembangkan dan disubstitusi dengan komponen-komponen vektor:

$$(u_1 - v_1)^2 + (u_2 - v_2)^2 = u_1^2 + u_2^2 + v_1^2 + v_2^2 - 2\|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$$

$$-2(u_1v_1 + u_2v_2) = -2\|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$$

$$(u_1v_1 + u_2v_2) = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta \quad (2.8)$$

Dengan menggunakan definisi persamaan 2.7 maka persamaan 2.8 menjadi:

$$\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta \quad (2.9)$$

Dengan demikian, hubungan antara *dot product* dan sudut antara dua vektor dinyatakan sebagai:

$$\cos \theta = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (2.10)$$

2.7 Metrik Jarak

Secara matematis, konsep jarak dinyatakan melalui metrik jarak, yaitu suatu fungsi yang mengukur tingkat kedekatan atau kemiripan antar objek dalam ruang berdimensi banyak. Metrik jarak dinotasikan sebagai $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}$ dan harus memenuhi aksioma dasar untuk memastikan perhitungan kedekatan antar data dapat dilakukan secara matematis dengan benar. Aksioma-aksioma tersebut, sebagaimana dijelaskan oleh (Deza, 2013), meliputi:

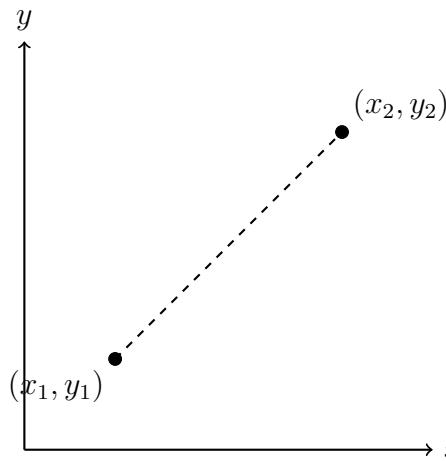
1. *Non-negativity*: $d(x, y) \geq 0$
2. *Identity of Indiscernibles*: $d(x, y) = 0 \iff x = y$
3. *Symmetry*: $d(x, y) = d(y, x)$
4. *Triangle Inequality*: $d(x, y) \leq d(x, z) + d(z, y)$

Dengan memenuhi keempat aksioma tersebut, suatu fungsi dapat dikatakan sebagai metrik jarak yang sah. Dalam konteks analisis dokumen teks, vektor-vektor yang merepresentasikan dokumen diperlakukan sebagai titik-titik dalam ruang berdimensi tinggi, sehingga metrik jarak dapat diterapkan untuk mengukur kedekatan semantik antar dokumen. Pemilihan jenis jarak yang digunakan dalam suatu analisis perlu disesuaikan dengan karakteristik data dan tujuan dari pengukuran, karena penggunaan metrik yang tidak sesuai dapat memengaruhi ketepatan hasil analisis. Pada penelitian ini, peneliti akan menggunakan empat jenis metrik jarak dalam perhitungan matriks jarak

pada dokumen teks, yaitu jarak Euclidean, Manhattan, Jaccard, dan Cosine. Metrik-metrik ini digunakan untuk mengukur tingkat kedekatan antar dokumen yang telah direpresentasikan dalam bentuk vektor numerik.

2.7.1 Jarak Euclidean

Jarak Euclidean merupakan ukuran jarak paling umum yang digunakan dalam ruang berdimensi- n dan dikenal juga sebagai $L2-norm$, merupakan salah satu ukuran jarak dalam ruang \mathbb{R}^n yang digunakan untuk mengukur jarak lurus (garis terpendek) antara dua titik. Jarak ini didasarkan pada teorema Pythagoras, di mana perbedaan nilai setiap koordinat dikuadratkan terlebih dahulu, kemudian dijumlahkan, dan hasilnya diambil akar kuadrat. Jarak ini merepresentasikan "jarak sebenarnya" antara dua titik dalam ruang berdimensi- n , sebagaimana ditunjukkan pada Gambar 2.8



Gambar 2.8 Ilustrasi Jarak Euclidean

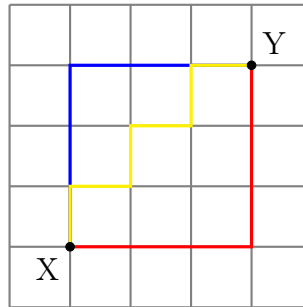
Jarak Euclidean antara dua vektor $x, y \in \mathbb{R}^n$, didefinisikan sebagai berikut:

$$d(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.11)$$

dengan $x_i, y_i \in \mathbb{R}$, di mana x_i menyatakan nilai fitur ke- i dari data uji dan y_i menyatakan nilai fitur ke- i dari data latih.

2.7.2 Jarak Manhattan

Jarak Manhattan, yang dikenal juga sebagai *City-block distance*, *Taxi-cab distance*, atau *L1-norm*, merupakan salah satu jarak dalam ruang \mathbb{R}^n yang digunakan untuk mengukur jarak antara dua titik berdasarkan perbedaan nilai setiap koordinatnya. Jarak Manhattan diasumsikan bahwa perpindahan antara dua titik hanya dapat dilakukan sepanjang sumbu koordinat dalam struktur berbasis grid, sebagaimana ditunjukkan pada Gambar 2.9.



Gambar 2.9 Ilustrasi Jarak Manhattan

Jarak Manhattan antara dua vektor $x, y \in \mathbb{R}^n$, didefinisikan sebagai berikut:

$$d(x, y) = \sum_{i=1}^n |x_i - y_i| \quad (2.12)$$

dengan $x_i, y_i \in \mathbb{R}$, di mana x_i menyatakan nilai fitur ke- i dari data uji dan y_i menyatakan nilai fitur ke- i dari data latih.

2.7.3 Jarak Jaccard

Jarak Jaccard digunakan untuk mengukur tingkat ketidaksamaan (*dissimilarity*) antara dua himpunan. Dalam konteks klasifikasi teks, jarak ini sering digunakan untuk membandingkan dua dokumen dengan menghitung kesamaan kata-kata yang terdapat di dalamnya.

Secara matematis, *Jaccard Similarity* antara dua himpunan X dan Y didefinisikan sebagai

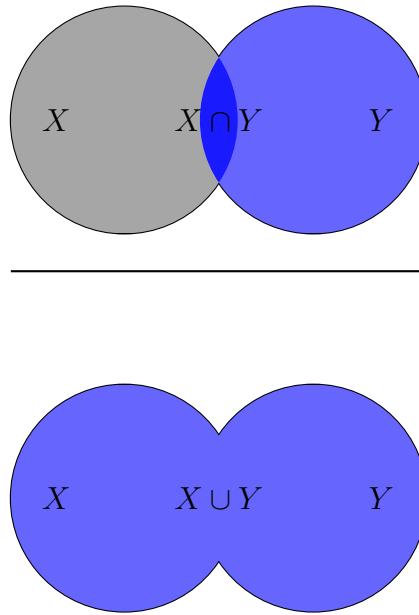
$$Jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \quad (2.13)$$

dengan X dan Y merupakan himpunan kata dari dua dokumen, $|X \cap Y|$ menyatakan jumlah kata yang muncul di kedua dokumen, dan $|X \cup Y|$ menyatakan jumlah total kata unik dalam kedua dokumen.

Jarak ini menghasilkan nilai dalam rentang 0 hingga 1, di mana nilai 1 menunjukkan kesamaan sempurna antara dua dokumen, sedangkan nilai 0 menunjukkan bahwa kedua dokumen tidak memiliki kesamaan kata sama sekali. Konsep *Jaccard Similarity* dapat dilihat pada Gambar 2.10, di mana area yang diarsir menunjukkan bagian irisan, sedangkan seluruh area dalam lingkaran merepresentasikan gabungan.

Gambar 2.10 menunjukkan bahwa semakin besar irisan antara dua himpunan dibandingkan dengan gabungannya, semakin tinggi pula nilai *Jaccard Similarity*. Selanjutnya, Jarak Jaccard merupakan komplemen dari *Jaccard similarity*, yang didefinisikan sebagai:

$$d_{Jaccard}(X, Y) = 1 - Jaccard(X, Y) \quad (2.14)$$



Gambar 2.10 Ilustrasi Jaccard Similarity

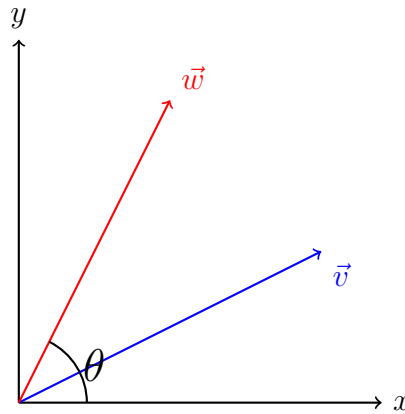
Pada penelitian ini, jarak Jaccard digunakan untuk menghitung kedekatan antar dokumen teks berdasarkan kesamaan kata yang terkandung dalam dokumen tersebut. Nilai jarak Jaccard berada dalam rentang $[0, 1]$, dimana nilai 0 menunjukkan bahwa kedua himpunan sangat identik, sedangkan nilai 1 menunjukkan bahwa kedua himpunan sangat berbeda.

2.7.4 *Cosine Similarity dan Cosine Distance*

Cosine similarity merupakan ukuran kemiripan antara dua vektor dalam ruang berdimensi tinggi yang sering digunakan dalam pengolahan bahasa alami (*Natural Language Processing*), terutama dalam klasifikasi atau pengelompokan dokumen teks. Metrik ini menghitung nilai cosinus dari sudut antara dua vektor yang mewakili fitur suatu dokumen. Secara matematis, *Cosine Similarity* antara dua vektor v dan w didefinisikan sebagai:

$$\text{Cosine}(v, w) = \frac{v \cdot w}{\|v\| \|w\|} \quad (2.15)$$

dengan $v \cdot w$ merupakan hasil perkalian *dot-product* antara dua vektor, dan $\|v\|$, $\|w\|$ masing-masing merupakan norma Euclidean dari vektor v dan w . *Cosine similarity* bernilai antara 0 hingga 1 jika digunakan pada data frekuensi non-negatif, di mana nilai 1 menunjukkan bahwa kedua vektor memiliki arah yang sama (sangat mirip), dan nilai 0 menunjukkan bahwa kedua vektor saling tegak lurus (tidak mirip sama sekali). Ilustrasi *Cosine Similarity* dapat dilihat pada Gambar 2.11, yang menunjukkan sudut antara dua vektor dalam bidang dua dimensi.



Gambar 2.11 Ilustrasi *Cosine Similarity* sebagai sudut antara dua vektor

Gambar 2.11 menunjukkan bahwa semakin kecil sudut θ antara vektor v dan w , semakin besar nilai *cosine similarity* yang dihasilkan. Selanjutnya, *Cosine Distance* merupakan ukuran ketidaksamaan (*dissimilarity*) berdasarkan *cosine similarity*, yang didefinisikan sebagai:

$$d_{\text{Cosine}}(v, w) = 1 - \text{Cosine}(v, w) \quad (2.16)$$

Pada penelitian ini, *cosine distance* digunakan untuk mengukur perbedaan orientasi (arah) antar dokumen teks yang telah direpresentasikan dalam bentuk

vektor berbasis kata (misalnya dengan TF-IDF). Nilai jarak cosine berada dalam rentang $[0, 1]$, di mana nilai 0 menunjukkan kemiripan sempurna dan nilai 1 menunjukkan bahwa kedua vektor tidak memiliki kesamaan arah sama sekali.

2.8 Matriks Jarak

Dalam konteks analisis dokumen teks, setelah setiap dokumen direpresentasikan sebagai vektor numerik, matriks jarak digunakan untuk menyimpan ukuran kedekatan atau ketidaksamaan antara setiap pasangan dokumen dalam suatu korpus. Matriks jarak adalah sebuah matriks persegi berukuran $n \times n$, di mana n adalah jumlah total dokumen yang dianalisis. Setiap entri (i, j) dalam matriks menyimpan nilai jarak $D_{ij} = d(dok_i, dok_j)$ antara dokumen ke- i dan dokumen ke- j , yang dihitung menggunakan metrik jarak tertentu. Elemen diagonal matriks (D_{ii}) selalu bernilai nol karena jarak setiap dokumen dengan dirinya sendiri adalah nol. Struktur matriks jarak dapat direpresentasikan sebagai berikut:

$$D = \begin{bmatrix} d(dok_1, dok_1) & d(dok_1, dok_2) & \cdots & d(dok_1, dok_n) \\ d(dok_2, dok_1) & d(dok_2, dok_2) & \cdots & d(dok_2, dok_n) \\ \vdots & \vdots & \ddots & \vdots \\ d(dok_n, dok_1) & d(dok_n, dok_2) & \cdots & d(dok_n, dok_n) \end{bmatrix}$$

Matriks ini bersifat simetris jika metrik jarak yang digunakan memenuhi sifat simetri, yaitu $d(dok_i, dok_j) = d(dok_j, dok_i)$. Dengan demikian, matriks jarak menyediakan representasi menyeluruh mengenai hubungan kedekatan antar seluruh pasangan dokumen dalam korpus, dan menjadi dasar penting dalam berbagai metode analisis lanjutan, seperti klasifikasi, klasterisasi, dan visual

multidimensi.

2.9 Studi Kasus

Pada bagian ini, akan disajikan sebuah studi kasus untuk mendemonstrasikan proses perhitungan matriks jarak pada dokumen teks. Studi kasus ini menggunakan 10 dokumen teks yang diambil secara acak dari dataset penelitian tugas akhir. Studi kasus ini bertujuan untuk mengilustrasikan bagaimana berbagai metrik jarak, seperti Euclidean, Manhattan, Jaccard, dan Cosine, diterapkan untuk mengukur tingkat kemiripan atau ketidakmiripan antar dokumen teks setelah melalui tahapan pra-pemrosesan dan representasi numerik.

2.9.1 Pra-pemrosesan Teks

Sebelum perhitungan matriks jarak dapat dilakukan, dokumen teks mentah harus melalui serangkaian tahapan prapemrosesan untuk menyusun data teks yang awalnya tidak terstruktur menjadi bentuk yang lebih teratur dan mudah dianalisis, serta menghilangkan elemen yang tidak relevan yang dapat mengganggu hasil analisis. Proses prapemrosesan dimulai dengan *Case Folding (Lower Case)*. Tahapan ini mengubah seluruh huruf dalam teks menjadi huruf kecil untuk menyamakan bentuk kata yang berbeda karena Kapitalisasi, misalnya "Love" dan "love" akan dianggap sama. Contoh hasil *Lower Case* pada dokumen teks dapat dilihat pada Tabel 2.1

Tabel 2.1 Contoh Data Setelah Proses *Lower Case*

Data Asli	Setelah Proses <i>Lower Case</i>
2 billion....Coming soonï»¿	2 billion....coming soonï»¿
Hey everyone. Watch this trailer!!!!!!! http://believemefilm.com?h1r=h2hQBUVB i ï»¿	hey everyone. watch this trailer!!!!!!! http://believemefilm.com?h1r=h2hqbuvb i ï»¿

Setelah mengubah huruf menjadi huruf kecil semua, dilakukan *cleaning data*. Tahapan ini bertujuan untuk membersihkan karakter khusus yang tidak relevan yang dapat mengganggu proses tokenisasi dan analisis. Untuk mengilustrasikan proses *cleaning data*, dapat dilihat pada Tabel 2.2

Tabel 2.2 Contoh Data Setelah Proses *Cleaning Data*

Setelah Proses <i>Lower Case</i>	Setelah Proses <i>Cleaning Data</i>
2 billion....coming soon»¿	numeric billion coming soon
hey everyone. watch this trailer!!!!!! http://believemefilm.com?hlr=h2hqbuvb ï»¿	hey everyone watch this trailer url

Tahapan selanjutnya adalah tokenisasi, yang merupakan tahapan krusial dalam pemrosesan bahasa alami. Token didefinisikan sebagai unit-unit teks bermakna. Proses ini memecah teks yang telah dibersihkan menjadi unit-unit kata (token) sehingga lebih mudah dianalisis dan diproses pada tahap berikutnya. Contoh hasil dari proses tokenisasi teks disajikan melalui Tabel 2.3

Tabel 2.3 Contoh Data Setelah Proses Tokenisasi

Setelah Proses <i>Cleaning Data</i>	Setelah Proses Tokenisasi
numeric billion coming soon	['numeric', 'billion', 'coming', 'soon']
hey everyone watch this trailer url	['hey', 'everyone', 'watch', 'this', 'trailer', 'url']

Tahapan selanjutnya yaitu normalisasi kata, yaitu proses mengubah kata-kata tidak baku menjadi kata baku. Pada penelitian ini, kamus normalisasi disusun mandiri oleh peneliti berdasarkan observasi terhadap data yang digunakan. Contoh hasil dari proses normalisasi kata disajikan melalui Tabel 2.4

Kemudian, tahapan *Stopwords Removal* dilakukan untuk menghapus kata-kata umum dalam bahasa yang kurang memiliki nilai pembeda dalam

Tabel 2.4 Contoh Data Setelah Proses Normalisasi

Setelah Proses Tokenisasi	Setelah Proses Normalisasi
['numeric', 'billion', 'coming', 'soon']	['numeric', 'billion', 'coming', 'soon']
['hey', 'everyone', 'watch', 'this', 'trailer', 'url']	['hello', 'everyone', 'watch', 'this', 'trailer', 'url']

klasifikasai teks (misalnya, "*the*", "*at*", "*is*"). Contoh hasil dari proses *stopwords removal* disajikan melalui Tabel 2.5

Tabel 2.5 Contoh Data Setelah Proses *Stopwords Removal*

Setelah Proses Tokenisasi	Setelah Proses <i>Stopwords Removal</i>
['numeric', 'billion', 'coming', 'soon']	['numeric', 'billion', 'coming', 'soon']
['hello', 'everyone', 'watch', 'this', 'trailer', 'url']	['hello', 'everyone', 'watch', 'trailer', 'url']

Tahapan terakhir dalam prapemrosesan adalah *stemming*, yang merupakan proses penggabungan kata-kata yang memiliki akar kata yang sama. Misalnya bentuk tunggal, jamak, atau berbagai *tense* dari suatu kata disatukan ("*run*", "*running*", "*runs*", dan "*ran*" seluruhnya berasal dari akar kata "*run*"). Proses ini penting karena tidak mengubah makna semantisnya dalam konteks *text mining* dan membantu mengurangi dimensi data. Contoh hasil dari proses *stemming* disajikan melalui Tabel 2.6

Tabel 2.6 Contoh Data Setelah Proses *Stemming*

Setelah Proses <i>Stopwords Removal</i>	Setelah Proses <i>Stemming</i>
['numeric', 'billion', 'coming', 'soon']	['numer', 'billion', 'come', 'soon']
['hello', 'everyone', 'watch', 'trailer', 'url']	['hello', 'everyon', 'watch', 'trailer', 'url']

2.9.2 Representasi Dokumen Teks

Setelah proses prapemrosesan, dokumen teks perlu direpresentasikan ke dalam bentuk numerik agar dapat diolah secara matematis. Dua pendekatan

yang relevan dalam studi kasus ini adalah *Bag of Words*(BoW) biner dan *Term Frequency-Inverse Document Frequency*(TF-IDF).

***Bag of Words*(BoW)**

Pendekatan *Bag of Words*(BoW) merepresentasikan sebuah dokumen sebagai kumpulan kata-kata, tanpa memperhatikan tata urutan kata dan struktur gramatikal. Informasi yang dipertahankan hanyalah frekuensi kemunculan kata dalam dokumen. Untuk keperluan perhitungan jarak Jaccard, dokumen akan direpresentasikan dalam bentuk BoW biner, di mana nilai 1 diberikan jika sebuah kata muncul dalam dokumen, dan 0 jika tidak muncul.

Berikut adalah representasi *Bag of Words* (BoW) biner dari 10 dokumen yang telah melewati tahapan *cleaning*

Tabel 2.7 Bag of Words Features Matrix

Doc ID	awesom	billion	come	danc	everyon	funni	good	happy_emoji	hello	just	lada	laugh	numer	sexi	so	soon	trailer	url	view	watch
1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
2	1	0	0	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0

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

TF-IDF merupakan salah satu metode representasi teks yang banyak digunakan dalam *text mining* untuk mengukur tingkat kepentingan suatu kata terhadap sebuah dokumen dalam suatu kumpulan dokumen (korpus). Bobot suatu kata dalam dokumen dihitung dari dua komponen utama, yaitu *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF). TF mengukur seberapa sering suatu kata muncul dalam sebuah dokumen, dan IDF mengukur seberapa jarang kata muncul di seluruh dokumen. Kombinasi kedua dokumen ini menghasilkan bobot numerik yang mencerminkan pentingnya suatu kata

dalam mendeskripsikan isi dokumen. Pada penelitian ini, pembobotan TF-IDF diimplementasikan menggunakan kelas `TfidfVectorizer` dari pustaka *scikit-learn* di Python, yang secara efisien mentransformasikan data teks yang telah melalui tahap preprocessing menjadi vektor numerik. Sebagai contoh perhitungan untuk memperoleh bobot fitur pada setiap dokumen, digunakan data dokumen dengan ID asli 1,2, dan 8 dari data yang telah melalui proses prapemrosesan. Daftar kata (komentar) dari dokumen-dokumen terpilih disajikan pada Tabel 2.8

Tabel 2.8 Contoh Kata untuk Perhitungan TF-IDF

Doc ID	Komentar Setelah Prapemrosesan
1	['numer', 'billion', 'come', 'soon']
2	['so', 'funni', 'awesom', 'laugh', 'sexi', 'lada', 'happy_emoji']
3	['hello', 'everyon', 'watch', 'trailer', 'url']

Langkah pertama yaitu membangun *term* dari seluruh kata unik yang ada dalam seluruh korpus. *Term* yang terbentuk berisi duapuluh kata, yaitu: *numer, billion, come, soon, so, funni, awesom, laugh, sexi, lada, happy_emoji, url, view, good, hello, everyon, watch, trailer, just, danc*. Tahapan selanjutnya adalah perhitungan *Term Frequency* (TF) menggunakan pendekatan hitung mentah (*raw count*), yaitu dengan menghitung seberapa sering suatu kata ke-*i* muncul dalam dokumen ke-*i*. Misalnya kata *billion* muncul satu kali pada beberapa dokumen, sedangkan kata *awesom* hanya muncul pada dokumen 2. Hasil lengkap dari perhitungan frekuensi kemunculan kata ini disajikan dalam Tabel 2.9, yang menunjukkan frekuensi kemunculan setiap kata per dokumen.

Tabel 2.9 *Term Frequency* (TF) Tiap Dokumen

Doc ID	awesom	billion	come	danc	everyon	funni	good	happy_emoji	hello	just	lada	laugh	numer	sexi	so	soon	trailer	url	view	watch
1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
2	1	0	0	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	1	0	0	0	3	0	1	0	0	1	0	0	0	0	0	0	0

Langkah selanjutnya adalah menghitung nilai *Inverse Document Frequency* (IDF) untuk setiap kata dalam keseluruhan dokumen. Nilai ini dihitung berdasarkan banyaknya dokumen yang mengandung kata tertentu, dan bertujuan untuk mengurangi bobot kata-kata yang terlalu umum. Dalam implementasi *scikit-learn*, IDF dihitung berdasarkan rumus sebagai berikut:

$$idf(t) = \log \left(\frac{1 + n_d}{1 + df(t)} \right) + 1$$

dimana n_d adalah jumlah total dokumen, dan $df(t)$ adalah jumlah dokumen yang memuat kata ke- t . Penambahan angka 1 pada pembilang dan penyebut dilakukan untuk mencegah pembagian nol dan mengurangi dominasi kata yang sangat umum. Sebagai contoh, perhitungan IDF untuk kata `url` adalah:

$$n_d = 10, \quad df_{\text{url}} = 4$$

$$idf_{\text{url}} = \log \left(\frac{1 + 10}{1 + 4} \right) + 1 = \log(2.2) + 1 \approx 1,788$$

Semakin banyak sebuah kata muncul dalam berbagai dokumen, maka nilai IDF-nya akan semakin kecil. Hal ini mencerminkan bahwa kata tersebut bersifat lebih umum dalam korpus, sehingga kontribusinya dalam membedakan isi antar dokumen menjadi lebih rendah. Sebagai contoh, kata `number` juga muncul pada empat dokumen (dokumen 1, 5, 6, dan 10), sehingga memiliki IDF yang sama dengan kata `url`, yaitu sekitar 1,788. Hasil IDF untuk kata lainnya dihitung dengan cara yang serupa, dan dirangkum dalam Tabel 2.10.

Setelah memperoleh hasil dari TF dan IDF, langkah selanjutnya adalah menghitung bobot TF dan IDF menggunakan persamaan 2.5. Contoh perhitungan TF-IDF:

Tabel 2.10 Contoh Hasil Perhitungan *Inverse Document Frequency* (IDF)

Fitur	<i>Inverse Document Frequency</i> (IDF)
awesom	2.704
billion	2.704
come	2.704
danc	2.704
everyon	2.704
funni	2.704
good	2.704
happy_emoji	2.299
hello	2.704
just	2.704
lada	2.704
laugh	2.299
numer	1.788
sexi	2.704
so	2.704
soon	2.704
trailer	2.704
url	1.788
view	2.704
watch	2.704

Perhitungan TF-IDF untuk fitur (happy_emoji):

$$tf(happy_emoji, D10) = 3, \quad idf_{happy_emoji} = 1.299$$

$$tf - idf(happy_emoji, D10) = 3 \times 2.299 = 6.897$$

Seluruh hasil perhitungan TF-IDF disajikan pada Tabel 2.11.

Tabel 2.11 Nilai Bobot Fitur per Dokumen Sebelum Normalisasi

Doc ID	awesom	billion	come	danc	everyon	funni	good	happy_emoji	hello	just	lada	laugh	numer	sexi	so	soon	trailer	url	view	watch
1	0	2.704	2.704	0	0	0	0	0	0	0	0	0	1.788	0	0	2.704	0	0	0	0
2	2.704	0	0	0	0	2.704	0	2.299	0	0	2.704	2.299	0	2.704	2.704	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.788	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.788	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1.788	0	0	0	0	0	2.704	0
6	0	0	0	0	0	0	0	0	0	0	0	2.299	1.788	0	0	0	0	0	0	0
7	0	0	0	0	0	0	2.704	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	2.704	0	0	0	2.704	0	0	0	0	0	0	0	2.704	1.788	0	2.704
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.788	0	0
10	0	0	0	2.704	0	0	0	6.897	0	2.704	0	0	1.788	0	0	0	0	0	0	0

Setelah memperoleh nilai TF-IDF, tahapan selanjutnya adalah melakukan normalisasi menggunakan metode L2 *normalization*, di mana setiap

vektor dokumen disesuaikan agar memiliki panjang (magnitudo) sebesar 1, sehingga perbedaan panjang dokumen tidak mempengaruhi perhitungan jarak. Sebagai contoh, hasil TF-IDF Doc ID 1 sebelum normalisasi:

$$w(D_1) = [2.704, 2.704, 1.788, 2.704]$$

Panjang vektor dihitung sebagai:

$$\begin{aligned} \|D_1\|_2 &= \sqrt{2.704^2 + 2.704^2 + 1.788^2 + 2.704^2} = \sqrt{7.311 + 7.311 + 3.196 + 7.311} \\ &= \sqrt{25.131} \approx 5.013 \end{aligned}$$

Kemudian, setiap nilai TF-IDF dibagi dengan 5.013 sehingga diperoleh vektor hasil normalisasi:

$$\begin{aligned} w_{normalized}(D_1) &= \left[\frac{2.704}{5.013}, \frac{2.704}{5.013}, \frac{1.788}{5.013}, \frac{2.704}{5.013} \right] \\ w_{normalized}(D_1) &\approx [0.539, 0.539, 0.356, 0.539] \end{aligned}$$

Hasil TF-IDF normalisasi untuk dokumen lainnya dihitung dengan cara yang serupa, dan disajikan pada Tabel 2.12

Tabel 2.12 Nilai Bobot Fitur per Dokumen Setelah Normalisasi

Doc ID	awesom	billion	come	danc	everyon	funni	good	happy_emoji	hello	just	lada	laugh	numer	sexi	so	soon	trailer	url	view	watch
1	0	0.539	0.539	0	0	0	0	0	0	0	0	0	0.356	0	0	0.539	0	0	0	0
2	0.393	0	0	0	0	0.393	0	0.334	0	0	0.393	0.334	0	0.393	0.393	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0.551	0	0	0	0	0	0.834	0
6	0	0	0	0	0	0	0	0	0	0	0	0.789	0.613	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0.474	0	0	0	0.474	0	0	0	0	0	0	0	0.474	0.313	0	0.474
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0.334	0	0	0	0.852	0	0.334	0	0	0.221	0	0	0	0	0	0	0

setelah melalui proses pembobotan TF-IDF dan normalisasi, setiap dokumen kini direpresentasikan sebagai vektor berdimensi teap yang mencerminkan pentingnya masing-masing kata relatif terhadap seluruh korpus. Representasi vektor ini tidak hanya menyerderhanakan struktur dokumen, tetapi juga me-

memungkinkan penerapan berbagai teknik analisis numerik berbasis ruang vektor. Salah satu analisis lanjutan dari hasil representasi ini adalah pengukuran kesamaan atau perbedaan antar dokumen. Oleh karena itu, tahapan berikutnya dalam penelitian ini adalah menghitung matriks jarak antar dokumen berdasarkan representasi numerik tersebut.

2.9.3 Perhitungan Matriks Jarak

Dengan setiap dokumen yang telah diubah menjadi vektor fitur numerik melalui proses prapemrosesan dan representasi (TF-IDF dan BoW biner), tahapan selanjutnya adalah mengukur ‘jarak’ antar dokumen tersebut. Matriks jarak adalah sebuah matriks persegi berukuran $n \times n$, di mana n adalah jumlah total dokumen yang dianalisis. Setiap entri (i, j) dalam matrik ini disebut D_{ij} , berisikan nilai jarak antara dokumen ke- i dan dokumen ke- j ($d(dok_i, dok_j)$), yang dihitung menggunakan metrik jarak tertentu. Pada bagian ini, akan dijelaskan penerapan jarak Euclidean, Manhattan, Jaccard, dan Cosine.

Sebagai ilustrasi perhitungan akan difokuskan pada Dokumen 1 dan Dokumen 6 dari dataset. Vektor TF-IDF untuk Dokumen 1 (D1) dan Dokumen 6 (D6) dari Tabel 2.11 adalah sebagai berikut:

- $D1_{\text{TFIDF}} = [0.000, 0.539, 0.539, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.356, 0.000, 0.000, 0.539, 0.000, 0.000, 0.000, 0.000]$
- $D6_{\text{TFIDF}} = [0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.789, 0.0613, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000]$

Vektor *Bag of Words* (BoW) biner untuk Dokumen 1 (BoW1) dan Dokumen 6 (BoW6) dari Tabel 2.7 adalah:

- BoW1 = [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]

- BoW6 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]

Jarak Euclidean

Pada pembahasan ini, digunakan dua komentar yang akan dihitung yaitu komentar pertama (Dokumen 1) dan komentar keenam (Dokumen 6). Jarak Euclidean merupakan ukuran jarak garis lurus antara dua titik dalam ruang Euclidean. Misalkan dua buah vektor dalam ruang berdimensi n , masing-masing dinyatakan sebagai: $X = (x_1, x_2, \dots, x_n)$ dan $Y = (y_1, y_2, \dots, y_n)$. Jarak Euclidean antara dua vektor tersebut fspst dihitung menggunakan persamaan (2.11). Dalam konteks ini, vektor fitur dari komentar pertama (D1) dan komentar keenam (D6) terdiri dari 20 elemen, yang merupakan hasil ekstraksi fitur menggunakan metode TF-IDF. Oleh karena itu, rumus jarak Euclidean antara D1 dan D6 dituliskan sebagai:

$$d_{\text{Euc}}(D_1, D_6) = \sqrt{\sum_{i=1}^n (d_{1,i} - d_{6,i})^2 + (d_{1,2} - d_{6,2})^2 + \dots + (d_{1,20} - d_{6,20})^2}$$

Langkah pertama adalah menghitung selisih kuadrat dari setiap pasangan elemen $d_{1,i} - d_{6,i}$ yang bersesuaian antara kedua dokumen. Beberapa perhitungan yang diperoleh adalah sebagai berikut:

- $(d_{1,2} - d_{6,2})^2 = (0.539 - 0.000)^2 = 0.290521$
- $(d_{1,3} - d_{6,3})^2 = (0.539 - 0.000)^2 = 0.290521$
- $(d_{1,12} - d_{6,12})^2 = (0.000 - 0.789)^2 = 0.622521$
- $(d_{1,13} - d_{6,13})^2 = (0.356 - 0.613)^2 = 0.066049$
- $(d_{1,16} - d_{6,16})^2 = (0.539 - 0.000)^2 = 0.290521$

Jumlah dari kelima komponen tersebut adalah:

$$\begin{aligned}\sum_{i=1}^{20}(d_{1,i} - d_{6,i})^2 &= 0.290521 + 0.290521 + 0.622521 + 0.066049 + 0.290521 \\ &= 1.560133\end{aligned}$$

Dengan demikian, diperoleh jarak Euclidean antara Dokumen 1 dan Dokumen 6 sebagai berikut:

$$d(D_1, D_6) = \sqrt{1.560133} \approx 1.24905284$$

Hasil ini menunjukkan bahwa kedua dokumen memiliki tingkat kemiripan yang sedang, karena nilai jarak tidak terlalu besar namun juga tidak mendekati nol. Selanjutnya, berikut di sajikan matriks jarak Euclidean berukuran 10×10 yang menunjukkan jarak antara setiap pasangan dokumen dalam himpunan data yang dianalisis:

$$E_{10 \times 10} = \begin{bmatrix} 0 & 1.414 & 1.414 & 1.414 & 1.267 & 1.249 & 1.414 & 1.414 & 1.414 & 1.357 \\ 1.414 & 0 & 1.414 & 1.414 & 1.414 & 1.213 & 1.414 & 1.414 & 1.414 & 1.195 \\ 1.414 & 1.414 & 0 & 0 & 1.414 & 1.414 & 1.414 & 1.171 & 0 & 1.414 \\ 1.414 & 1.414 & 0 & 0 & 1.414 & 1.414 & 1.414 & 1.171 & 0 & 1.414 \\ 1.267 & 1.414 & 1.414 & 1.414 & 0 & 1.150 & 1.414 & 1.414 & 1.414 & 1.325 \\ 1.249 & 1.213 & 1.414 & 1.414 & 1.150 & 0 & 1.414 & 1.414 & 1.414 & 1.314 \\ 1.414 & 1.414 & 1.414 & 1.414 & 1.414 & 1.414 & 0 & 1.414 & 1.414 & 1.414 \\ 1.414 & 1.414 & 1.171 & 1.171 & 1.414 & 1.414 & 1.414 & 0 & 1.171 & 1.414 \\ 1.414 & 1.414 & 0 & 0 & 1.414 & 1.414 & 1.414 & 1.171 & 0 & 1.414 \\ 1.357 & 1.195 & 1.414 & 1.414 & 1.325 & 1.314 & 1.414 & 1.414 & 1.414 & 0 \end{bmatrix}$$

Dari matriks di atas, dapat disimpulkan bahwa semakin kecil nilai jarak antara

dua dokumen, maka semakin besar tingkat kemiripan konten di antara keduanya. Sebaliknya, semakin besar nilai jarak, maka kedua dokumen cenderung memiliki perbedaan isi yang lebih signifikan.

Jarak Manhattan

Jarak Manhatta, yang juga dikenal sebagai jarak $L1$ atau jarak *City Block*, merupakan ukuran jarak antara dua titik dalam ruang berdimensi- n yang dihitung berdasarkan jumlah selisih absolut dari setiap koordinat. Misalkan terdapat dua buah vektor, yaitu: $X = (x_1, x_2, \dots, x_n)$ dan $Y = (y_1, y_2, \dots, y_n)$. Jarak Manhattan antara dua vektor tersebut dihitung menggunakan Persamaan (2.12). Pada penelitian ini, digunakan dua komentar, yaitu Dokumen 1 dan Dokumen 6, yang masing-masing direpresentasikan oleh vektor fitur berdimensi 20. Maka, perhitungan jarak Manhattan antara kedua dokumen tersebut dapat ditulis sebagai:

$$d_{\text{Manhattan}}(D_1, D_6) = \sum_{i=1}^n |d_{1,i} - d_{6,i}| + |d_{1,2} - d_{6,2}| + \dots + |d_{1,20} - d_{6,20}|$$

Langkah pertama adalah menghitung selisih absolut dari setiap pasangan elemen $d_{1,i} - d_{6,i}$ untuk semua fitur $i = 1$ sampai 20. Berikut beberapa contoh perhitungannya:

- $|d_{1,2} - d_{6,2}| = |0.539 - 0.000| = 0.539$
- $|d_{1,3} - d_{6,3}| = |0.539 - 0.000| = 0.539$
- $|d_{1,12} - d_{6,12}| = |0.000 - 0.789| = 0.789$
- $|d_{1,13} - d_{6,13}| = |0.356 - 0.613| = 0.257$
- $|d_{1,16} - d_{6,16}| = |0.539 - 0.000| = 0.539$

Total dari selisih absolut kelima komponen tersebut adalah:

$$\sum_{i=1}^{20} |d_{1,i} - d_{6,i}| = 0.539 + 0.539 + 0.789 + 0.257 + 0.539 = 2.663$$

Dengan demikian, diperoleh jarak Manhattan antara Dokumen 1 dan Dokumen 6 sebesar:

$$d_{Manhattan}(D_1, D_6) = 2.663$$

Nilai ini menunjukkan tingkat perbedaan yang lebih besar dibandingkan dengan jarak Euclidean yang telah dihitung sebelumnya, karena metrik manhattan tidak mengkuadratkan selisih, melainkan menjumlahkannya secara langsung. Berikut akan disajikan matriks jarak Manhattan berukuran 10×10 yang menggambarkan jarak antar semua pasang dokumen dalam himpunan data:

$$M_{10 \times 10} = \begin{bmatrix} 0 & 4.613 & 2.974 & 2.974 & 2.647 & 2.663 & 2.974 & 4.187 & 2.974 & 3.275 \\ 4.613 & 0 & 3.639 & 3.639 & 4.024 & 3.372 & 3.639 & 4.851 & 3.639 & 3.712 \\ 2.974 & 3.639 & 0 & 0 & 2.385 & 2.403 & 2 & 2.585 & 0 & 2.742 \\ 2.974 & 3.639 & 0 & 0 & 2.385 & 2.403 & 2 & 2.585 & 0 & 2.742 \\ 2.647 & 4.024 & 2.385 & 2.385 & 0 & 1.685 & 2.385 & 3.598 & 2.385 & 2.686 \\ 2.663 & 3.372 & 2.403 & 2.403 & 1.685 & 0 & 2.403 & 3.616 & 2.403 & 2.703 \\ 2.974 & 3.639 & 2 & 2 & 2.385 & 2.403 & 0 & 3.212 & 2 & 2.742 \\ 4.187 & 4.851 & 2.585 & 2.585 & 3.598 & 3.616 & 3.212 & 0 & 2.585 & 3.955 \\ 2.974 & 3.639 & 0 & 0 & 2.385 & 2.403 & 2 & 2.585 & 0 & 2.742 \\ 3.275 & 3.712 & 2.742 & 2.742 & 2.686 & 2.703 & 2.742 & 3.955 & 2.742 & 0 \end{bmatrix}$$

Seperti halnya pada metrik Euclidean, semakin kecil nilai jarak Manhattan antar dokumen, semakin besar tingkat kemiripan antara kedua dokumen tersebut. Sebaliknya, nilai jarak yang besar menunjukkan perbedaan yang lebih signifikan.

Jarak Jaccard

Indeks Jaccard mengukur kesamaan antara dua himpunan dan didefinisikan sebagai ukuran irisan dibagi dengan ukuran gabungan dari dua himpunan tersebut. Dalam konteks representasi biner, Jaccard Index dirumuskan sebagai:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Jarak Jaccard mengukur ketidaksamaan dan dihitung sebagai:

$$D_J(A, B) = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Dalam konteks representasi biner, jika:

- M_{11} adalah jumlah kata di mana kedua dokumen memiliki nilai 1
- M_{10} adalah jumlah kata di mana Dokumen A = 1 dan Dokumen B = 0
- M_{01} adalah jumlah kata di mana Dokumen A = 0 dan Dokumen B = 1

Maka:

$$J(A, B) = \frac{M_{11}}{M_{11} + M_{10} + M_{01}}$$

$$D_J(A, B) = \frac{M_{10} + M_{01}}{M_{11} + M_{10} + M_{01}}$$

Contoh Perhitungan: Dokumen 1 dan Dokumen 6

- Dokumen 1:

$$\text{BoW1} = [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]$$

Kata yang muncul (bernilai 1): **billion, come, numer, soon**

- Dokumen 6:

$$\text{BoW}_6 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]$$

Kata yang muncul (bernilai 1): **laugh, numer**

Berdasarkan analisis representasi biner:

- $M_{11} = 1$, menunjukkan 1 kata yang muncul di kedua dokumen, yaitu **numer**.
- $M_{10} = 3$, menunjukkan 3 kata yang hanya muncul pada Dokumen 1, yaitu **billion, come, soon**.
- $M_{01} = 1$, menunjukkan 1 kata yang hanya muncul pada Dokumen 6, yaitu **laugh**.

Maka:

$$J(\text{BoW}_1, \text{BoW}_6) = \frac{1}{1 + 3 + 1} = \frac{1}{5} = 0.2$$

$$D_J(\text{BoW}_1, \text{BoW}_6) = 1 - 0.2 = 0.8$$

Atau:

$$D_J(\text{BoW}_1, \text{BoW}_6) = \frac{3 + 1}{5} = \frac{4}{5} = 0.8$$

Hasil perhitungan jarak Jaccard antara Dokumen 1 dan Dokumen 6 menunjukkan nilai sebesar 0.8. Nilai ini mengindikasikan bahwa 80% dari seluruh kata yang muncul di salah satu dari kedua dokumen tersebut tidak dimiliki secara bersama, atau dengan kata lain, hanya 20% kata yang sama. Kondisi ini mencerminkan bahwa Dokumen 1 dan Dokumen 6 memiliki perbedaan

yang cukup besar dalam hal konten kata yang digunakan. Untuk mengilustrasikan hubungan antar dokumen secara menyeluruh, matriks jarak Jaccard berukuran 10×10 dapat disusun. Setiap entri (i, j) dalam matriks ini mewakili nilai $D_j(Dokumen_i, Dokumen_j)$, yang mencerminkan tingkat ketidaksamaan antara dua dokumen berdasarkan representasi binernya.

$$J_{10 \times 10} = \begin{bmatrix} 0 & 1 & 1 & 1.0 & 0.800 & 0.800 & 1 & 1 & 1 & 0.857 \\ 1 & 0 & 1 & 1 & 1 & 0.875 & 1 & 1 & 1 & 0.900 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.8 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.8 & 0 & 1 \\ 0.800 & 1 & 1 & 1 & 0 & 0.666 & 1 & 1 & 1 & 0.800 \\ 0.800 & 0.875 & 1 & 1 & 0.666 & 0 & 1 & 1 & 1 & 0.800 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.8 & 0.8 & 1 & 1 & 1 & 0 & 0.8 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.8 & 0 & 1 \\ 0.857 & 0.900 & 1 & 1 & 0.800 & 0.800 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Cosine Similarity

Cosine Similarity merupakan ukuran kesamaan yang didasarkan pada sudut kosinus antara dua vektor non-nol dalam ruang berdimensi n . Metrik ini sering digunakan untuk mengukur kemiripan teks. Misalkan dua buah vektor dalam ruang berdimensi n , yaitu: $X = (x_1, x_2, \dots, x_n)$ dan $Y = (y_1, y_2, \dots, y_n)$. Menggunakan Persamaan (2.15):

$$S_C(D_1 D_6) = \cos(\theta) = \frac{D_1 \cdot D_6}{\|D_1\| \cdot \|D_6\|} = \frac{\sum_{i=1}^n d_{1,i} d_{6,i}}{\sqrt{\sum_{i=1}^n d_{1,i}^2} \cdot \sqrt{\sum_{i=1}^n d_{6,i}^2}}$$

Nilai *Cosine Similarity* berkisar antara 0 hingga 1 untuk vektor dengan nilai non-negatif seperti TF-IDF. Nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar antara kedua dokumen. Sementara jarak Cosine (*Cosine Distance*) dapat dihitung sebagai:

$$D_C(D_1, D_6) = 1 - S_C(D_1, D_6)$$

Langkah Perhitungan *Cosine Similarity*: Dokumen 1 dan Dokumen 6

Langkah pertama adalah menghitung *dot product* dari setiap pasangan elemen $d_{1,i} \cdot d_{6,i}$ untuk semua fitur $i = 1$ sampai 20:

- $d_{1,2} \cdot d_{6,2} = 0.539 \times 0.000 = 0$
- $d_{1,3} \cdot d_{6,3} = 0.539 \times 0.000 = 0$
- $d_{1,12} \cdot d_{6,12} = 0.000 \times 0.789 = 0$
- $d_{1,13} \cdot d_{6,13} = 0.356 \times 0.613 = 0.218228$
- $d_{1,16} \cdot d_{6,16} = 0.539 \times 0.000 = 0$

Jumlah dari semua *dot product* adalah:

$$D1 \cdot D6 = 0.218228$$

Langkah selanjutnya adalah menghitung Magnitude $\|D1\|$ dan $\|D6\|$:

$$\begin{aligned} \|D_1\| &= \sqrt{0.539^2 + 0.539^2 + 0.356^2 + 0.539^2} \\ &= \sqrt{0.290 + 0.290 + 0.126 + 0.290} \\ &= \sqrt{0.996} \\ &\approx 0.997 \end{aligned}$$

$$\begin{aligned}
\|D_6\| &= \sqrt{0.789^2 + 0.613^2} \\
&= \sqrt{0.622 + 0.375} \\
&= \sqrt{0.997} \\
&\approx 0.998
\end{aligned}$$

Cosine Similarity:

$$S_C(D1, D6) = \frac{0.218}{0.997 \times 0.998} = \frac{0.218}{0.995} \approx 0.219$$

Cosine Distance:

$$D_C(D1, D6) = 1 - S_C(D1, D6) = 1 - 0.219 = 0.781$$

Dengan demikian, hasil perhitungan menunjukkan bahwa *Cosine Similarity* antara Dokumen 1 dan Dokumen 6 adalah 0.219, yang mengindikasikan tingkat kemiripan yang rendah antara kedua dokumen berdasarkan bobot kata-kata yang terkandung di dalamnya. Sebaliknya, jarak Cosine sebesar 0.781 menunjukkan bahwa 78.1% arah vektor antar dokumen berbeda, yang memperkuat interpretasi bahwa kedua dokumen tersebut tidak serupa secara

konteks teks. Berikut disajikan matriks Jarak Cosine 10×10 :

$$C_{10 \times 10} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0.803 & 0.781 & 1 & 1 & 1 & 0.921 \\ 1 & 0 & 1 & 1 & 1 & 0.735 & 1 & 1 & 1 & 0.714 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.686 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.686 & 0 & 1 \\ 0.803 & 1 & 1 & 1 & 0 & 0.661 & 1 & 1 & 1 & 0.878 \\ 0.781 & 0.735 & 1 & 1 & 0.661 & 0 & 1 & 1 & 1 & 0.864 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0.686 & 0.686 & 1 & 1 & 1 & 0 & 0.686 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0.686 & 0 & 1 \\ 0.921 & 0.714 & 1 & 1 & 0.878 & 0.864 & 1 & 1 & 1 & 0 \end{bmatrix}$$

BAB III

KESIMPULAN

3.1 Kesimpulan

Perhitungan matriks jarak pada dokumen teks merupakan tahapan penting dalam analisis data berbasis teks, karena memungkinkan pengukuran tingkat kemiripan atau perbedaan antar dokumen secara numerik. Proses ini diawali dengan tahapan prapemrosesan teks seperti *case folding*, pembersihan karakter, tokenisasi, Normalisasi kata, penghapusan *stopwords*, dan *stemming*, yang bertujuan untuk menyederhanakan struktur data teks mentah menjadi bentuk yang lebih konsisten dan mudah dianalisis. Setelah proses ini, dokumen direpresentasikan dalam bentuk vektor menggunakan metode seperti *Bag of Words* (BoW) atau *Term Frequency-Inverse Document Frequency* (TF-IDF).

Dengan representasi numerik tersebut, jarak antar dokumen dapat dihitung menggunakan berbagai metrik, seperti Manhattan dan Jaccard. Masing-masing metrik memiliki karakteristik tersendiri dalam mengukur kemiripan antar dokumen, sehingga pemilihannya perlu disesuaikan dengan tujuan analisis dan sifat data. Melalui perhitungan matriks jarak, hubungan antar dokumen dapat diidentifikasi secara sistematis, memungkinkan analisis lebih lanjut terhadap struktur dan pola dalam kumpulan teks. Oleh karena itu, pemahaman mendalam terhadap proses ini menjadi fondasi penting dalam berbagai aplikasi pengolahan data teks.

DAFTAR PUSTAKA

- Aggarwal, C. C. (2018). Machine Learning for Text (1st ed.). Springer. Retrieved from <https://link.springer.com/book/10.1007/978-3-319-73531-3>
- Aleqabie, H. J., Sfoq, M., Albeer, R., & Abd, E. (2024, 01). A review of text mining techniques: Trends, and applications in various domains. Iraqi Journal For Computer Science and Mathematics, 5, 125-141. doi: 10.52866/ijcsm.2024.05.01.009
- Anton, H., & Rorres, C. (2014). Elementary Linear Algebra (11th ed.). John Wiley & Sons Incorporated. Retrieved from <https://books.google.co.id/books?id=loRbAgAAQBAJ>
- Boyd, S., & Vandenberghe, L. (2018). Introduction to Applied Linear Algebra. Cambridge University Press & Assessment. doi: 10.1017/9781108583664
- Deza, M. D. E. (2013). Encyclopedia of distances (2nd ed ed.). Springer. Retrieved from <http://gen.lib.rus.ec/book/index.php?md5=6913bc9b68416f1f3333fc5a62a22547>
- Jo, T. (2018). Text mining: Concepts, implementation, and big data challenge. Springer International Publishing. doi: <https://doi.org/10.1007/978-3-031-75976-5>
- Lane, H., Hapke, H., & Howard, C. (2019). Natural language processing in action: Understanding, analyzing, and generating text with python (1st ed.). Shelter Island, NY: Manning Publications. (True PDF, Bookmarked, OCR)
- Raschka, S., Liu, Y. H., & Mirjalili, V. (2022). Machine learning with pytorch and scikit-learn: Develop machine learning and deep learning models with python. Packt Publishing. Retrieved from <https://libgen.li/file.php?md5=c1ae0c9c5eaff8b929972790879197e6>
- Rosen, K. H. (2000). Handbook of discrete and combinatorial mathematics (1st ed.). CRC Press. Retrieved from <http://gen.lib.rus.ec/book/index.php?md5=2de1ed971148eeb7fa41482616d0ab1a>

LAMPIRAN

Tabel 3.1 Contoh Data Komentar

Content	Label
2 billion....Coming soonï»¿	ham
It's so funny it's awesomeness lol aaaaaaa sexy lada??	ham
http://ubuntuone.com/40beUutVu2ZKxK4uTgPZ8KÃrÂzÂ	spam
http://shhort.com/a?r=G8iX5cTKdÃrÂzÂ	spam
2.126.521.750 views!!!!!!!!!!!!!!!!!!!!ï»¿	ham
2:05. Hahahahah ï»¿	ham
Goodï»¿	ham
Hey everyone. Watch this trailer!!!!!!! http://believemefilm.com?h1r=h2hQBUBV ï»¿	spam
https://www.facebook.com/photo.php?fbid=543627485763966&l=0d878a889cÃrÂzÂ	spam
JUST DANCE 3 ??????	ham