Selamat malam, Bapak/Ibu dosen pembimbing serta penguji(bapak sukarasa dan ibu tari selaku dosen pembimbing serta bapak putu dan ibu suciptawati selaku dosen penguji yang telah menyempatkan waktunya pada malam hari ini). Sebelumnya perkenalkan, saya Luh Sukma Mulyani dengan NIM 2108541027. Pada kesempatan kali ini saya akan memaparkan telaah Pustaka dari TA saya yang berjudul "Perhitungan Matriks Jarak Pada Dokumen Teks".

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 KNN.

Untuk menyusun matriks tersebut, terlebih dahulu kita perlu memahami bagaimana dokumen teks direpresentasikan secara numerik, yang dimulai dari proses text mining dan pengukuran document similarity.

Text mining merupakan cabang dari data mining yang fokus pada ekstraksi informasi dari data teks tidak terstruktur. Salah satu konsep penting di dalamnya adalah document similarity, yaitu cara mengukur seberapa mirip atau berbeda isi dua dokumen secara matematis. Dalam penelitian ini, perhitungan similarity ini menjadi dasar dalam penyusunan matriks jarak, dan seluruh prosesnya diawali dengan tahapan text preprocessing agar dokumen dapat direpresentasikan secara numerik.

Sebelum dokumen teks dapat dianalisis secara numerik, perlu dilakukan proses text preprocessing untuk membersihkan dan menyiapkannya dalam bentuk yang lebih terstruktur. Proses ini penting karena dalam teks sering kali terdapat elemen yang tidak relevan, seperti tanda baca, tag HTML, singkatan, atau kata-kata umum yang tidak informatif.

Tahapan preprocessing yang digunakan dalam penelitian ini mengacu pada Aggarwal (2018) dan mencakup enam langkah utama:

- Pertama, case folding, yaitu mengubah semua huruf menjadi huruf kecil untuk menyamakan kata yang secara makna sama tetapi berbeda kapital.
- Kedua, cleaning data, yaitu pembersihan karakter khusus, simbol, HTML, email, URL, angka, bahkan emoji, dengan berbagai fungsi regex di Python agar teks lebih konsisten dan terbaca.
- Ketiga, tokenisasi, yaitu memecah teks menjadi unit-unit kata atau token.
- Keempat, normalisasi, yaitu mengubah kata tidak baku seperti “u” menjadi “you” atau “plzz” menjadi “please”. Proses ini menggunakan kamus normalisasi manual yang disusun berdasarkan data.
- Kelima, stopwords removal, yaitu menghapus kata umum seperti ‘the’, ‘is’, atau ‘in’ karena tidak memberikan informasi yang signifikan.
- Terakhir, stemming, yaitu mengembalikan kata ke bentuk dasarnya, misalnya ‘running’, ‘ran’, dan ‘runs’ menjadi ‘run’, untuk mengurangi variasi kata.

Dengan menyelesaikan keenam tahap ini, dokumen teks yang awalnya tidak terstruktur dapat diubah menjadi representasi numerik yang lebih bersih, konsisten, dan siap digunakan untuk analisis lebih lanjutan.

Setelah teks bersih, langkah selanjutnya adalah merepresentasikannya ke dalam bentuk numerik. Salah satu pendekatan dasarnya adalah Bag of Words (BoW).

BoW merepresentasikan dokumen sebagai kumpulan kata-kata, tanpa memperhatikan tata urutan atau struktur gramatikalnya. Informasi yang dipertahankan hanyalah frekuensi kemunculan kata dalam dokumen. Untuk keperluan penelitian ini, khususnya untuk perhitungan jarak Jaccard, kita akan menggunakan BoW biner, di mana nilai 1 diberikan jika sebuah kata muncul dalam dokumen, dan 0 jika tidak muncul.

Pendekatan representasi dokumen selanjutnya yang saya gunakan adalah Term Frequency-Inverse Document Frequency (TF-IDF).

TF-IDF mengukur tingkat kepentingan suatu kata terhadap sebuah dokumen dalam suatu kumpulan dokumen (korpus). Bobot kata dihitung dari dua komponen utama: Term Frequency (TF), yang mengukur seberapa sering kata muncul dalam dokumen, dan Inverse Document Frequency (IDF), yang mengukur seberapa jarang kata muncul di seluruh korpus. Rumus standar TF adalah $tf\_ij = f\_ij / \sum f\_j$ di mana f_ij adalah jumlah kemunculan kata ke-i dalam dokumen ke-j. Sedangkan untuk IDF, rumusnya adalah $idf\_i = \log(N/df\_i)$, di mana N adalah jumlah total dokumen dan df_i adalah jumlah dokumen yang mengandung kata ke-i.

Dalam penelitian ini, proses pembobotan TF-IDF diimplementasikan menggunakan pustaka Scikit-learn. Library ini menggunakan rumus IDF yang sedikit dimodifikasi dengan penambahan konstanta satu untuk menghindari pembagian nol, yaitu:

$idf(t) = \log(1+n\_d/1+df(t))+1$

Selain itu, hasil vektor TF-IDF juga dinormalisasi menggunakan L2 normalization, sehingga panjang setiap vektor menjadi 1. Ini penting agar panjang dokumen tidak memengaruhi hasil perhitungan jarak, terutama saat digunakan dalam algoritma seperti K-Nearest Neighbors. Dengan TF-IDF, setiap dokumen direpresentasikan sebagai vektor berdimensi tinggi. Vektor-vektor inilah yang menjadi input utama dalam perhitungan matriks jarak, serta digunakan dalam proses klasifikasi KNN untuk mengukur kemiripan antar dokumen berdasarkan bobot kata yang relevan.

Representasi dokumen dalam bentuk numerik ini kemudian kita rangkum dalam sebuah Document Term Matrix (DTM).

DTM adalah matriks berukuran N×n, di mana N adalah jumlah dokumen dan n adalah jumlah kosakata unik. Setiap elemen A _ij pada matriks ini menunjukkan frekuensi kemunculan kata ke-i pada dokumen ke-j. DTM memungkinkan kita mengidentifikasi pola kemunculan kata dan keterkaitan antar dokumen.

Sebagai contoh, matriks A dengan 3 dokumen dan 4 kata unik dapat digambarkan seperti pada slide. Baris pertama (2,0,1,0) menunjukkan dokumen pertama mengandung kata pertama dua kali dan kata ketiga satu kali, dan seterusnya.

Dengan dokumen yang sudah direpresentasikan secara numerik, selanjutnya tentang Metrik Jarak.

Secara matematis, konsep kedekatan antar objek dalam analisis data dinyatakan melalui fungsi yang disebut metrik jarak. Fungsi ini memetakan pasangan titik dalam ruang berdimensi tinggi serta dinotasikan sebagai d: Rn ×Rn →R≥0 dan harus memenuhi empat aksioma dasar: Non-negativity, Identity of Indiscernibles, Symmetry, dan Triangle Inequality.

Dalam konteks analisis dokumen teks, vektor-vektor yang merepresentasikan dokumen diperlakukan sebagai titik-titik dalam ruang berdimensi tinggi. Pemilihan jenis jarak sangat penting dan perlu disesuaikan dengan karakteristik data serta tujuan analisis, karena penggunaan metrik yang tidak sesuai dapat memengaruhi ketepatan hasil.

Pada penelitian ini saya menggunggunakan empat metrik, yaitu Euclidean, Manhattan, Jaccard, dan Cosine distance.

Jarak Euclidean atau L2-norm, ini merupakan ukuran jarak paling umum yang digunakan dalam ruang berdimensi-n dan dikenal juga sebagai L2-norm. Jarak ini mengukur jarak lurus (garis terpendek) antara dua titik, didasarkan pada teorema Pythagoras.

Berbeda dari Euclidean, Jarak Manhattan atau L1-norm menghitung jarak dengan asumsi kita hanya bisa berpindah sepanjang sumbu koordinat, seperti pola jalan di kota berbentuk grid. Karena itu, metrik ini sering juga disebut City Block atau Taxicab Distance.

Jarak Jaccard adalah metrik yang digunakan untuk mengukur tingkat ketidaksamaan antara dua himpunan. Dalam konteks teks, kita menganggap setiap dokumen sebagai himpunan kata. Maka, Jaccard menghitung seberapa banyak kata yang sama di antara dua dokumen. Semakin banyak kata yang sama, semakin tinggi nilainya.

Kemudian Jarak Jaccard adalah kebalikannya dimana rumusnya sebagai berikut
Nilainya selalu antara 0 dan 1.
Nilai 0 berarti dua dokumen sangat mirip
Nilai 1 berarti keduanya benar-benar berbeda

---

selain pendekatan berbasis himpunan, ada juga metrik yang sangat populer untuk data berbasis vektor, yaitu Cosine Similarity.

Cosine Similarity digunakan untuk mengukur kemiripan arah antara dua vektor, tanpa memperhatikan panjang atau magnitude-nya. Metrik ini sangat sering digunakan dalam analisis dokumen teks, terutama ketika menggunakan representasi TF-IDF.

Setelah menghitung jarak antar setiap pasangan dokumen menggunakan metrik-metrik tersebut, hasilnya akan disimpan dalam sebuah Matriks Jarak.

Matriks jarak adalah sebuah matriks persegi berukuran n×n, di mana n adalah jumlah total dokumen yang dianalisis. Setiap entri (i,j) dalam matriks ini, yaitu D _ij, berisikan nilai jarak antara dokumen ke-i dan dokumen ke-j (d(dok_i, dok_j)), yang dihitung menggunakan metrik jarak tertentu. Elemen diagonal matriks (D_ii) selalu bernilai nol karena jarak setiap dokumen dengan dirinya sendiri adalah nol. Matriks ini bersifat simetris jika metrik jarak yang digunakan memenuhi sifat simetri, yaitu d(dok_i,dok_j)=d(dok_j,dok_i). Struktur matriks jarak dapat direpresentasikan seperti pada slide berikut.

Untuk mendemonstrasikan proses perhitungan matriks jarak, berikut di sajikan sebuah studi kasus sederhana.

Alur pemrosesan teksnya adalah dari Raw Text, melalui Preprocessing, kemudian Text Representation (menggunakan Bag of Words biner dan TF-IDF), lalu Pemilihan Metrik Jarak, hingga akhirnya menghasilkan Distance Matrix 10×10 yang mencerminkan kedekatan antar dokumen berbasis representasi numerik teks.

Sebelum melakukan perhitungan matriks jarak antar dokumen, hal yang sangat penting dan tidak bisa dilewati adalah proses prapemrosesan teks. Teks mentah yang kita dapatkan dari korpus, pada dasarnya tidak terstruktur, mengandung banyak noise, seperti tanda baca, huruf kapital, kata-kata tidak baku, bahkan simbol atau link. Oleh karena itu, dalam penelitian ini dilakukan serangkaian tahapan preprocessing

---

- Tahap pertama yang saya lakukan adalah Case Folding, yaitu mengubah seluruh teks menjadi huruf kecil.Tujuannya adalah untuk menyamakan bentuk kata yang secara visual berbeda tapi secara semantik sama, seperti kata “Coming” dan “coming” yang secara makna identik tapi secara komputer dianggap berbeda jika belum diubah.

- Setelah semua huruf sudah berbentuk lowercase, langkah salanjutnya adalah cleaning text, yaitu menghapus karakter-karakter khusus yang tidak relevan. Tujuan tahap ini adalah untuk menghindari gangguan dalam proses tokenisasi dan analisis berikutnya.
- Tahap selanjutnya adalah tokenisasi, yaitu memecah teks menjadi unit kata atau yang disebut token. Ini merupakan tahap krusial dalam Natural Language Processing, karena dari sinilah tiap kata akan diperlakukan sebagai fitur yang berdiri sendiri.
- Selanjutnya dilakukan normalisasi, yaitu mengubah kata-kata tidak baku, typo, atau singkatan menjadi kata yang benar dan baku. Dalam penelitian ini, saya menyusun sendiri kamus normalisasi berdasarkan hasil observasi terhadap teks korpus.
- Setelah normalisasi, selanjutnya dilakukan penghapusan stopwords, yaitu kata-kata umum seperti "the", "is", "this", dan lain sebagainya yang tidak banyak membantu dalam membedakan dokumen.Tujuan tahap ini adalah untuk mengurangi dimensi data dan fokus pada kata-kata yang memang memiliki makna penting.
- Terakhir, saya melakukan stemming, yaitu mengubah kata ke bentuk dasarnya. Tujuan utama dari proses ini adalah untuk menyatukan kata-kata yang secara makna sama, sekaligus menurunkan dimensi data tanpa kehilangan informasi penting.

Setelah proses prapemrosesan teks selesai, langkah berikutnya dalam penelitian ini adalah merepresentasikan dokumen ke dalam bentuk numerik, agar bisa dihitung dan dianalisis secara matematis.

Pendekatan pertama adalah Bag of Words. Dalam metode ini, sebuah dokumen direpresentasikan sebagai kumpulan kata, tanpa memperhatikan urutan ataupun struktur kalimat. Di penelitian ini, saya menggunakan BoW biner.
Misalnya, jika kata “trailer” muncul dalam dokumen ke-8, maka pada posisi kata tersebut di baris dokumen 8 akan bernilai 1.
Contoh hasil BoW biner disajikan dalam slide.

Pendekatan kedua yang digunakan adalah TF-IDF.

Pertama, kita hitung Term Frequency (TF), yaitu frekuensi kemunculan kata ke-i dalam dokumen ke-j. Misalnya, kata ‘url’ muncul satu kali di dokumen D8, sedangkan ‘happy_emoji’ muncul tiga kali di D10. Maka:
tf(‘url’, D8) = 1
tf(‘happy_emoji’, D10) = 3
Berikutnya adalah Inverse Document Frequency (IDF) yang menghitung seberapa penting kata tersebut dalam keseluruhan dokumen. IDF dihitung berdasarkan jumlah dokumen yang mengandung kata tersebut. Misalnya:
Jumlah dokumen, nd = 10
Kata ‘url’ muncul di 4 dokumen, maka dfurl = 4
Rumus IDF-nya adalah:
idf(t) = log((1 + nd) / (1 + df(t))) + 1
→ idf_url = log(11/5) + 1 = log(2.2) + 1 ≈ 1,788

Terakhir, kita kalikan nilai TF dan IDF untuk mendapatkan bobot:
w_url,D8 = 1 × 1,788 = 1,788.

Setelah mendapatkan bobot TF-IDF, kita lakukan normalisasi menggunakan metode L2 untuk menyamakan panjang vektor antar dokumen

---

Misalnya, untuk dokumen D8, telah diperoleh bobot:
w_D8 = [2.704, 2.704, 2.704, 1.788, 2.704]
Langkah pertama adalah menghitung panjang vektornya:
$||D8||_2 = \sqrt{(2.704^2 + 2.704^2 + 2.704^2 + 1.788^2 + 2.704^2)}$
$= \sqrt{32.44} \approx 5.6956$
Kemudian setiap elemen dibagi dengan panjang vektor tersebut, sehingga:
w_normalized(D8) = [2.704 / 5.6956, ..., 1.788 / 5.6956]
≈ [0.474, 0.474, 0.474, 0.313, 0.474]
Hasil ini adalah vektor akhir dokumen D8 setelah normalisasi, yang siap digunakan untuk perhitungan matriks jarak.

Setelah setiap dokumen direpresentasikan dalam bentuk vektor TF-IDF yang telah dinormalisasi, langkah berikutnya adalah menghitung jarak antar dokumen.

Pada slide ini, saya akan menjelaskan perhitungan jarak Manhattan antara dua dokumen, yaitu Dokumen 1 dan Dokumen 6.

Langkah pertama adalah menghitung selisih absolut dari setiap pasangan elemen vektor fitur. Selanjutnya dijumlahkan sehingga memperoleh hasil 2.663. Dimana nilai ini menunjukkan bahwa kedua dokumen memiliki tingkat perbedaan sebesar 2.663 menurut metrik Manhattan.

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.

Jaccard

Langkah Pertama: Hitung Matriks Biner
Dari representasi biner kedua dokumen pada slide sebelumnya, diperoleh:
$M_{11} = 1$, yaitu jumlah kata yang muncul di kedua dokumen.
$M_{10} = 3$, yaitu kata yang hanya muncul di Dokumen 1.
$M_{01} = 1$, yaitu kata yang hanya muncul di Dokumen 6.
Nilai 0.8 ini mengindikasikan bahwa 80% dari seluruh kata unik di 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.

Setelah menghitung selisih kuadrat dari setiap pasangan elemen TF-IDF yang dinormalisasi dan menjumlahkannya, kita mendapatkan nilai 1.560133. Maka, jarak Euclidean antara Dokumen 1 dan Dokumen 6 adalah akar (1.560133) ≈1.24905284.
Nilai ini menunjukkan bahwa kedua dokumen memiliki tingkat kemiripan sedang, tidak terlalu mirip, tapi juga tidak sepenuhnya berbeda.

---
selnjutnya ada cosine similarity dan cosine distance

Pertama, kita hitung Dot Product
Kemudian, panjang vector
Dengan demikian, Cosine Similarity adalah SC(D1,D6)= 0.997×0.998/0.21 ≈0.219. Dan Cosine Distance adalah DC(D1,D6)=1−0.219=0.781
Nilai Cosine Similarity 0.219 mengindikasikan tingkat kemiripan yang rendah, sementara jarak Cosine 0.781 menunjukkan bahwa 78.1% arah vektor antar dokumen berbeda, yang memperkuat interpretasi bahwa kedua dokumen tersebut tidak serupa secara konteks teks.