

# Kopekan UTS Open Book Matkul Text Mining & Natural Language Processing

By Lathif Ramadhan (5231811022)

## Chapter 1: Pengantar Text Mining

### 1. Pendahuluan

- Era Informasi dan Ledakan Data Teks:** Kita hidup di era informasi, di mana volume data yang dihasilkan setiap hari tumbuh secara eksponensial. Sebagian besar dari data ini berbentuk teks, mulai dari dokumen, email, posting media sosial, artikel berita, hingga ulasan pelanggan. Data teks ini mengandung informasi yang sangat berharga yang dapat memberikan wawasan penting bagi organisasi, peneliti, dan individu.
- Tantangan Data Teks:** Tidak seperti data terstruktur yang tersimpan dalam database dengan format yang rapi, data teks seringkali tidak terstruktur, ambigu, dan penuh dengan variasi linguistik. Hal ini membuat analisis data teks menjadi tugas yang kompleks dan menantang.
- Text Mining Sebagai Solusi:** Di sinilah *text mining* hadir sebagai solusi. Text mining adalah bidang interdisipliner yang menggabungkan teknik dari berbagai bidang ilmu untuk mengekstrak informasi berharga dari data teks yang tidak terstruktur.

### 2. Definisi dan Konsep Dasar

- Definisi Text Mining:** Menurut PowerPoint, text mining adalah "Proses ekstraksi informasi dari teks tidak terstruktur dengan menggunakan teknik statistik, pembelajaran mesin, dan pemrosesan bahasa alami yang bertujuan mengubah teks mentah menjadi data yang dapat digunakan untuk analisis lebih lanjut".
- Elaborasi:** Lebih dari sekedar ekstraksi, text mining juga melibatkan penemuan pola, tren, dan pengetahuan baru dari sejumlah besar teks. Ini mencakup proses pembersihan, transformasi, dan analisis teks untuk menghasilkan wawasan yang bermakna.
- Perbedaan dengan Data Mining:** Penting untuk membedakan text mining dengan *data mining*. Data mining berfokus pada ekstraksi informasi dari data terstruktur, sedangkan text mining berurusan dengan data tidak terstruktur atau semi-terstruktur.
- Hubungan dengan NLP:** *Natural Language Processing* (NLP) adalah komponen penting dalam text mining. NLP menyediakan alat dan teknik untuk memahami dan memproses bahasa manusia, yang merupakan langkah penting dalam mengubah teks menjadi format yang dapat dianalisis.

### 3. Natural Language Processing (NLP)

- Definisi NLP:** Seperti yang telah disebutkan, PowerPoint mendefinisikan NLP sebagai "Cabang kecerdasan buatan yang memungkinkan komputer memahami, menginterpretasikan, dan menghasilkan bahasa manusia".
  - NLP adalah bidang interdisipliner yang menggabungkan ilmu komputer, linguistik, dan kecerdasan buatan untuk memungkinkan komputer bekerja dengan data bahasa manusia.
  - Tujuan utama NLP adalah untuk menjembatani kesenjangan antara komunikasi manusia dan pemahaman komputer. Manusia berkomunikasi

menggunakan bahasa alami, yang seringkali ambigu, kontekstual, dan penuh dengan nuansa. Komputer, di sisi lain, membutuhkan data yang terstruktur dan eksplisit. NLP memungkinkan komputer untuk memproses dan menganalisis bahasa alami sehingga dapat melakukan tugas-tugas yang berguna.

- NLP tidak hanya tentang memahami teks, tetapi juga tentang memahami ucapan. Oleh karena itu, NLP mencakup bidang seperti pengenalan ucapan (speech recognition) dan sintesis ucapan (speech synthesis).

#### Komponen Utama NLP:

##### Pemahaman Bahasa Alami (Natural Language Understanding - NLU):

- "Kemampuan mesin untuk menginterpretasikan makna dari teks."
- Detail:** NLU adalah bagian dari NLP yang berfokus pada kemampuan komputer untuk memahami arti dari input bahasa alami. Ini adalah tugas yang sangat kompleks karena bahasa manusia penuh dengan ambiguitas, sinonim, metafora, dan ironi. NLU melibatkan pemetaan teks ke representasi yang bermakna yang dapat dipahami oleh komputer.
- Contoh Tugas NLU:**
  - Analisis Sentimen:** Memahami apakah sebuah kalimat atau dokumen mengungkapkan opini positif, negatif, atau netral.
  - Ekstraksi Informasi:** Mengidentifikasi dan mengekstrak informasi terstruktur dari teks, seperti tanggal, nama, dan lokasi.
  - Pemahaman Pertanyaan (Question Answering):** Memahami pertanyaan yang diajukan dalam bahasa alami dan memberikan jawaban yang relevan.

##### Pembuatan Bahasa Alami (Natural Language Generation - NLG):

- "Kemampuan mesin untuk menghasilkan teks yang terdengar alami dan mudah dipahami."
- Detail:** NLG adalah kebalikan dari NLU. Ini adalah proses mengubah data terstruktur menjadi teks bahasa alami yang dapat dipahami oleh manusia. NLG digunakan dalam berbagai aplikasi, seperti pembuatan laporan, ringkasan teks, dan chatbot.
- Contoh Tugas NLG:**
  - Ringkasan Teks:** Membuat versi ringkas dari dokumen yang lebih panjang.
  - Generasi Teks:** Menghasilkan teks kreatif, seperti puisi atau cerita.
  - Sistem Dialog:** Membangun chatbot atau asisten virtual yang dapat berkomunikasi dengan manusia dalam bahasa alami.

#### Teknik dalam NLP:

- NLP menggunakan berbagai teknik, yang dapat dikelompokkan menjadi pendekatan tradisional dan modern:
- Pendekatan Tradisional:** Berbasis aturan (rule-based) dan statistik, seringkali membutuhkan pengetahuan linguistik yang mendalam.
- Pendekatan Modern:** Berbasis *machine learning* dan *deep learning*, yang mengandalkan data dalam jumlah besar untuk melatih model.

#### Teknik-teknik Utama:

- Analisis Sintaksis:**
  - "Memahami struktur gramatiskal kalimat."
  - Detail:** Analisis sintaksis (atau *parsing*) melibatkan penguraian struktur kalimat berdasarkan aturan tata bahasa. Ini membantu untuk memahami hubungan antar kata dan frasa dalam sebuah kalimat.
  - Contoh:**
    - Part-of-Speech Tagging (POS Tagging):** Menandai setiap kata dalam kalimat dengan label tata bahasa yang sesuai (misalnya, kata benda, kata kerja, kata sifat).
    - Dependency Parsing:** Mengidentifikasi hubungan ketergantungan antara kata-kata dalam kalimat (misalnya, kata mana yang menjadi kepala kata dari kata lain).
- Analisis Semantik:**
  - "Memahami makna kata dan kalimat."
  - Detail:** Analisis semantik melampaui struktur gramatiskal dan berfokus pada pemahaman arti dari teks. Ini melibatkan pemahaman hubungan antar konsep, identifikasi ambiguitas, dan resolusi referensi (menentukan apa yang dirujuk oleh kata ganti).
  - Contoh:**
    - Word Sense Disambiguation:** Menentukan arti yang benar dari sebuah kata dalam konteks tertentu (misalnya, kata "bank" bisa berarti lembaga keuangan atau tepi sungai).
    - Named Entity Recognition (NER):**
    - Pengenalan Entitas Bernama (Named Entity Recognition - NER):**
      - "Mengidentifikasi dan mengklasifikasikan entitas seperti nama orang, organisasi, dan lokasi."
      - Detail:** NER adalah tugas penting dalam NLP yang melibatkan identifikasi kata atau frasa dalam teks yang merujuk pada entitas bernama dan mengklasifikasikannya ke dalam kategori yang telah ditentukan sebelumnya.
      - Contoh:** Dalam kalimat "Apple didirikan oleh Steve Jobs di California," NER akan mengidentifikasi "Apple" sebagai ORGANISASI, "Steve Jobs" sebagai ORANG, dan "California" sebagai LOKASI.
- Analisis Sentimen:**
  - "Menentukan emosi atau opini yang terkandung dalam teks."
  - Detail:** Analisis sentimen (atau *opinion mining*) adalah bidang NLP yang berfokus pada identifikasi dan ekstraksi opini, sikap, dan emosi dari teks. Ini sangat berguna untuk memahami umpan balik pelanggan, memantau sentimen publik, dan menganalisis tren media sosial.
- Tingkat Analisis Sentimen:**
  - Polaritas:** Menentukan apakah sentimen dalam teks bersifat positif, negatif, atau netral.
  - Subjektivitas:** Membedakan antara teks faktual dan teks yang mengungkapkan opini.
  - Emosi:** Mengidentifikasi emosi spesifik yang diungkapkan

- **Peran NLP dalam Text Mining:**
    - "NLP adalah fondasi dari banyak tugas text mining. Teknik NLP digunakan untuk *preprocessing* teks, ekstraksi fitur, dan analisis konten."
    - **Penjelasan Lebih Lanjut:**
      - NLP menyediakan alat dan teknik penting yang memungkinkan text mining untuk bekerja secara efektif dengan data teks yang tidak terstruktur.
      - **Preprocessing:** Teknik NLP seperti tokenisasi, penghapusan *stopword*, dan stemming/lemmatisasi digunakan untuk membersihkan dan menyiapkan teks untuk analisis lebih lanjut.
      - **Ekstraksi Fitur:** NLP membantu dalam mengekstraksi fitur-fitur bermakna dari teks, seperti TF-IDF dan *word embeddings*, yang digunakan untuk merepresentasikan teks dalam bentuk numerik.
      - **Analisis Konten:** Teknik NLP seperti analisis sentimen, NER, dan klasifikasi teks digunakan untuk menganalisis konten teks dan mengekstrak informasi yang berharga.
- 4. Penggunaan NLP dan Text Mining: Elaborasi Mendalam**
- PowerPoint memberikan gambaran umum tentang aplikasi NLP, tetapi kita dapat memperluasnya untuk memberikan pemahaman yang lebih kaya tentang bagaimana teknologi ini mengubah berbagai bidang.
- Ekspansi Aplikasi:**
- a.  **Layanan Kesehatan:**
    - i. NLP tidak hanya menganalisis rekam medis, tetapi juga digunakan untuk:
      1. **Penemuan Obat:** Menganalisis literatur ilmiah dan data pasien untuk mengidentifikasi target obat baru dan memprediksi efektivitas obat.
      2. **Diagnosis Penyakit:** Membantu dokter dalam mendiagnosa penyakit dengan menganalisis gejala yang dijelaskan oleh pasien dalam catatan medis atau transkrip konsultasi.
      3. **Manajemen Kesehatan Populasi:** Mengidentifikasi tren kesehatan dalam populasi besar dengan menganalisis data media sosial atau survei online.
    - ii. **Contoh Tambahan:**
      1. Sistem yang memantau posting media sosial untuk mendeteksi wabah penyakit atau efek samping obat.
      2. NLP digunakan untuk mengekstrak informasi tentang hasil pengobatan dari catatan pasien untuk mendukung penelitian klinis.
  - b.  **Keuangan:**
    - i. Selain deteksi penipuan dan analisis sentimen pasar saham, NLP digunakan untuk:
      1. **Kepatuhan Regulasi:** Menganalisis dokumen keuangan untuk memastikan kepatuhan terhadap peraturan.
      2. **Layanan Pelanggan:** Chatbot yang menjawab pertanyaan pelanggan tentang produk dan layanan keuangan.
    - ii. **Contoh Tambahan:**
      1. Sistem yang menganalisis laporan keuangan untuk mengidentifikasi potensi risiko investasi.
  - c. **E-commerce:**
    - i. NLP memainkan peran penting dalam:
      1. **Personalisasi Pengalaman Belanja:** Merekomendasikan produk berdasarkan riwayat penelusuran dan pembelian pelanggan.
      2. **Manajemen Inventaris:** Memprediksi permintaan produk berdasarkan analisis tren media sosial dan ulasan pelanggan.
    - ii. **Contoh Tambahan:**
      1. Sistem yang mengotomatiskan pembuatan deskripsi produk.
      2. NLP digunakan untuk menganalisis umpan balik pelanggan dari berbagai sumber (ulasan, survei, media sosial) untuk meningkatkan kualitas produk dan layanan.
  - d. **Media Sosial:**
    - i. NLP digunakan secara luas untuk:
      1. **Moderasi Konten:** Mengidentifikasi dan menghapus konten yang melanggar pedoman komunitas (misalnya, ujaran kebencian, pelecehan).
      2. **Analisis Jaringan Sosial:** Memahami struktur dan dinamika jaringan sosial.
    - ii. **Contoh Tambahan:**
      1. Sistem yang mengidentifikasi akun palsu atau bot di media sosial.
      2. NLP digunakan untuk menganalisis percakapan online untuk memahami tren budaya dan opini publik.
  - e. **Pendidikan:**
    - i. NLP memiliki aplikasi dalam:
      1. **Pembelajaran Adaptif:** Menyesuaikan materi pembelajaran dengan kebutuhan individu siswa.
      2. **Dukungan Penulisan:** Memberikan umpan balik tentang tata bahasa, gaya, dan konten tulisan siswa.
    - ii. **Contoh Tambahan:**
      1. Sistem yang mengotomatiskan penilaian tugas esai dan memberikan umpan balik yang dipersonalisasi.
      2. NLP digunakan untuk menganalisis transkrip kuliah dan materi pembelajaran untuk membuat ringkasan dan panduan belajar.
  - f. **Manajemen Sumber Daya Manusia (SDM):**
    - i. NLP membantu dalam:
      1. **Perekrutan:** Menyaring lamaran kerja dan mengidentifikasi kandidat yang paling memenuhi syarat.
      2. **Pengembangan Karyawan:** Menganalisis umpan balik kinerja dan mengidentifikasi area untuk pelatihan dan pengembangan.
    - ii. **Contoh Tambahan:**
      1. Sistem yang menganalisis komunikasi internal untuk mengukur keterlibatan karyawan.
      2. NLP digunakan untuk mengotomatiskan proses onboarding karyawan.
  - g. **Hukum:**
    - i. NLP merevolusi bidang hukum dengan:
      1. **Penelitian Hukum:** Membantu pengacara menemukan kasus dan preseden yang relevan.
      2. **Analisis Kontrak:** Menganalisis kontrak untuk mengidentifikasi klausul penting dan potensi risiko.
    - ii. **Contoh Tambahan:**

1. Sistem yang mengotomatiskan pembuatan dokumen hukum.
2. NLP digunakan untuk menganalisis transkrip pengadilan dan dokumen hukum lainnya untuk mengekstrak informasi penting.

- f) **Visualization:** Menyajikan hasil text mining dalam format visual untuk memudahkan pemahaman dan interpretasi. **Contoh:** Word clouds, grafik batang, scatter plots.
- g) **Text Summarization:** Membuat ringkasan singkat dari dokumen teks yang lebih panjang. **Teknik:** Extractive summarization (memilih kalimat penting dari teks asli), abstractive summarization (menghasilkan ringkasan baru dengan kata-kata sendiri).

## 6. Tantangan dalam Text Mining: Penjelasan Mendalam

Text mining, meskipun sangat kuat, menghadapi sejumlah tantangan yang perlu diatasi untuk menghasilkan hasil yang akurat dan bermakna. Berikut adalah penjelasan yang lebih rinci:

- a) **Ambiguitas Bahasa:** "Kata-kata dapat memiliki banyak arti (polisemi), dan makna kalimat dapat bervariasi tergantung pada konteks." **Detail:**
  1. Bahasa manusia secara inheren ambigu. Kata-kata sering kali memiliki lebih dari satu arti (polisemi), dan arti yang tepat bergantung pada konteks di mana kata itu digunakan.
  2. **Contoh:** Kata "bank" dapat merujuk ke lembaga keuangan atau tepi sungai. Kalimat "Saya melihat kelelawar" bisa berarti saya melihat hewan itu atau saya melihat alat pemukul bola.
  3. Ambiguitas juga dapat muncul pada tingkat kalimat. Struktur kalimat yang sama dapat memiliki arti yang berbeda tergantung pada konteks.
  4. **Contoh:** "Waktu terbang seperti panah" dapat diartikan sebagai waktu berlalu dengan cepat atau mengukur kecepatan waktu.
  5. Mengatasi ambiguitas membutuhkan teknik NLP yang canggih untuk memahami konteks, melakukan *word sense disambiguation*, dan analisis semantik.
- b) **Variasi Linguistik:** "Bahasa memiliki banyak cara untuk mengekspresikan ide yang sama (sinonim, parafrasa)." **Detail:**
  1. Ada banyak cara untuk mengatakan hal yang sama dalam bahasa. Sinonim, parafrasa, dan variasi gaya penulisan dapat membuat sulit untuk mengidentifikasi teks yang memiliki makna yang sama.
  2. **Contoh:** Kata-kata seperti "besar," "agung," "raksasa," dan "luas" semuanya dapat digunakan untuk menggambarkan ukuran. Frasa seperti "membeli mobil" dan "mendapatkan mobil" memiliki arti yang sama.
  3. Variasi linguistik juga mencakup perbedaan regional dan dialek. Orang-orang dari berbagai daerah dapat menggunakan kata-kata dan frasa yang berbeda untuk mengekspresikan ide yang sama.
  4. Untuk mengatasi variasi linguistik, teknik NLP perlu mampu mengenali sinonim, memahami parafrasa, dan menormalkan teks ke bentuk standar.
- c) **Spelling dan Tata Bahasa:** "Teks yang dihasilkan pengguna (misalnya, media sosial) seringkali mengandung kesalahan ejaan dan tata bahasa." **Detail:**
  1. Teks yang dihasilkan pengguna, seperti posting media sosial, komentar online, dan pesan teks, sering kali mengandung kesalahan ejaan, kesalahan ketik, kesalahan tata bahasa, dan penggunaan bahasa informal.
  2. **Contoh:** Kata-kata mungkin salah dieja ("teh" bukan "the"), tata bahasa mungkin salah ("saya pergi ke toko" bukan "saya pergi ke toko"), dan slang serta singkatan mungkin digunakan ("LOL," "BRB").
  3. Noise dalam teks ini dapat menyulitkan algoritma text mining untuk memproses dan menganalisis data secara akurat.
  4. Teknik *preprocessing* teks, seperti koreksi ejaan dan normalisasi teks, dapat membantu mengurangi dampak kesalahan ejaan dan tata bahasa.
- d) **Data yang Bising:** "Teks dapat mengandung informasi yang tidak relevan atau noise (misalnya, iklan, kode HTML)." **Detail:**
  1. Data teks sering kali mengandung informasi yang tidak relevan dengan tugas text mining yang sedang dikerjakan. Informasi ini, yang disebut "noise," dapat mengganggu analisis dan mengurangi akurasi hasil.

- 2. **Contoh:** Teks yang diambil dari halaman web mungkin mengandung kode HTML, iklan, menu navigasi, dan konten lain yang tidak relevan dengan analisis teks. Email mungkin mengandung header dan footer yang tidak relevan.
- 3. **Preprocessing** teks yang tepat diperlukan untuk menghilangkan *noise* dan memfokuskan analisis pada konten yang relevan.

- e) **Skalabilitas:** "Memproses sejumlah besar data teks membutuhkan sumber daya komputasi yang signifikan." **Detail:**

1. Volume data teks yang dihasilkan setiap hari sangat besar, dan terus bertambah. Memproses dan menganalisis data teks dalam skala besar membutuhkan sumber daya komputasi yang signifikan, termasuk daya pemrosesan, memori, dan penyimpanan.
2. Algoritma text mining dapat menjadi intensif secara komputasi, terutama ketika berhadapan dengan data dalam jumlah besar.
3. Teknik seperti komputasi terdistribusi dan *parallel processing* dapat digunakan untuk meningkatkan skalabilitas text mining.

- f) **Privasi dan Etika:** "Text mining harus dilakukan dengan mempertimbangkan masalah privasi dan etika (misalnya, anonimisasi data, bias dalam algoritma)." **Detail:**

1. Text mining menimbulkan sejumlah masalah privasi dan etika yang penting untuk dipertimbangkan.
2. **Privasi Data:** Text mining sering kali melibatkan pengumpulan dan analisis data teks pribadi, seperti email, posting media sosial, dan catatan pelanggan. Penting untuk melindungi privasi individu dengan menganonimkan data, membatasi akses ke data, dan mematuhi peraturan privasi yang relevan.
3. **Bias Algoritma:** Algoritma text mining dapat dipengaruhi oleh bias yang ada dalam data pelatihan, yang mengarah pada hasil yang tidak adil atau diskriminatif. Penting untuk mengevaluasi dan mengurangi bias dalam algoritma untuk memastikan keadilan.
4. **Penggunaan yang Tidak Etis:** Text mining dapat digunakan untuk tujuan yang tidak etis, seperti pengawasan massal, manipulasi opini publik, dan diskriminasi. Penting untuk mengembangkan pedoman etika dan peraturan untuk mengatur penggunaan text mining.
5. **Transparansi:** Penting untuk transparan tentang bagaimana text mining digunakan dan bagaimana keputusan dibuat berdasarkan hasil analisis.

## 7. Kesimpulan

- Text mining adalah bidang yang kuat dan berkembang pesat dengan banyak aplikasi di berbagai industri.
- NLP memainkan peran penting dalam memungkinkan mesin untuk memahami dan memproses bahasa manusia untuk keperluan text mining.
- Proses text mining melibatkan beberapa tahapan, dari pengumpulan data hingga visualisasi hasil.
- Meskipun ada tantangan, text mining menawarkan potensi besar untuk mengungkap wawasan berharga dari data teks yang tidak terstruktur.

## Chapter 2: Persiapan Crawling Data

### 1. Pendahuluan

- Crawling data adalah langkah awal yang krusial dalam banyak proyek text mining dan analisis data. Ini melibatkan pengumpulan data secara otomatis dari berbagai sumber, terutama dari web.
- PowerPoint ini memberikan panduan dasar tentang persiapan untuk crawling data, mencakup tools, library, target crawling, dan penyimpanan data.

- Materi ini akan memperluas konsep-konsep tersebut dan menambahkan detail penting untuk pemahaman yang lebih komprehensif.

### 2. Tools yang Digunakan

PowerPoint menyebutkan Python sebagai tool utama. Ini sangat tepat karena Python memiliki ekosistem yang kaya untuk web crawling dan manipulasi data.

**Python:** Python adalah bahasa pemrograman yang populer untuk data science dan web crawling karena sintaksnya yang mudah dibaca, komunitas yang besar, dan banyak library yang tersedia.

#### 1) Lingkungan Pengembangan:

- a) **Google Colab:** Lingkungan Jupyter Notebook berbasis cloud yang memungkinkan Anda menulis dan menjalankan kode Python melalui browser web. Keuntungannya adalah tidak perlu instalasi, akses mudah dari mana saja, dan integrasi dengan Google Drive.
- b) **Jupyter Notebook:** Aplikasi web open-source yang memungkinkan Anda membuat dan berbagi dokumen yang berisi kode live, persamaan, visualisasi, dan teks naratif. Ideal untuk eksplorasi data, prototyping, dan dokumentasi.

#### 2) Alasan menggunakan Python untuk Crawling:

- a) **Library yang Kuat:** Python memiliki banyak library yang dirancang khusus untuk web crawling dan scraping.
- b) **Mudah Dipelajari dan Digunakan:** Sintaks Python yang sederhana membuatnya relatif mudah dipelajari, bahkan untuk pemula.
- c) **Komunitas Besar:** Komunitas Python yang besar menyediakan banyak sumber daya, tutorial, dan dukungan.
- d) **Cross-platform:** Python dapat dijalankan di berbagai sistem operasi.

### 3. Library yang Digunakan: Penjelasan Mendalam

- Bagian ini sangat penting karena *library* adalah "alat" yang kita gunakan untuk melakukan *crawling*. Ibaratnya, kalau kita mau membangun rumah, *library* ini seperti palu, gerja, dan alat-alat lainnya. PowerPoint sudah menyebutkan beberapa *library* Python yang penting, dan sekarang kita akan bahas lebih dalam.

#### Untuk Crawling Data dari Twitter:

- 1) **Tweepy:** "Library Python untuk mengakses Twitter API."
  - a) **Bahasa Sederhana:** Tweepy ini adalah penghubung antara program Python kita dengan Twitter. Twitter punya "pintu" khusus (API) yang bisa kita gunakan untuk mengambil data, dan Tweepy membantu kita "mengetuk pintu" itu dengan cara yang benar.
  - b) **Lebih Detail:** Dengan Tweepy, kita bisa melakukan banyak hal, misalnya:
    - i) Mengambil tweet berdasarkan kata kunci tertentu.
    - ii) Mendapatkan informasi tentang seorang pengguna Twitter (misalnya, berapa banyak pengikutnya).
    - iii) Mengirim tweet (jika kita ingin membuat program yang bisa nge-tweet otomatis).
    - iv) Mengikuti (*follow*) atau berhenti mengikuti (*unfollow*) akun lain.
  - c) **Kenapa Penting:** Kalau kita ingin menganalisis apa yang orang-orang bicarakan di Twitter, Tweepy adalah *library* yang sangat berguna.
- 2) **csv:** "Modul bawaan Python untuk bekerja dengan file CSV (Comma Separated Values)."
  - a) **Bahasa Sederhana:** CSV ini adalah cara untuk menyimpan data dalam bentuk tabel yang sangat sederhana. Bayangkan seperti tabel di Microsoft Excel, tapi disimpan dalam bentuk teks biasa.
  - b) **Lebih Detail:**
    - i) Setiap baris dalam file CSV adalah satu "data".

- ii) Setiap nilai dalam satu baris dipisahkan oleh tanda koma.
  - iii) Contoh:
    - (1) Nama,Usia,Kota
    - (2) Andi,25,Jakarta
    - (3) Budi,30,Surabaya
- c) **Kenapa Penting:** Setelah kita mengambil data dari Twitter (misalnya) menggunakan Tweepy, kita perlu menyimpannya dalam bentuk yang rapi. CSV adalah pilihan yang baik karena mudah dibaca oleh program lain atau aplikasi seperti Excel.

#### Untuk Scraping Web Umum:

- 1) **Requests:** "Library Python yang sederhana dan elegan untuk membuat permintaan HTTP."
  - a) **Bahasa Sederhana:** Ketika kita membuka sebuah halaman web di *browser*, *browser* kita mengirimkan "permintaan" ke server web untuk mendapatkan informasi halaman tersebut. Requests membantu program Python kita untuk melakukan hal yang sama.
  - b) **Lebih Detail:** Requests memungkinkan kita untuk mengirim berbagai jenis permintaan, yang paling umum adalah:
    - i) **GET:** Untuk mengambil informasi dari server (misalnya, mengambil kode HTML sebuah halaman web).
    - ii) **POST:** Untuk mengirimkan informasi ke server (misalnya, mengirimkan data formulir).
  - c) **Kenapa Penting:** Hampir semua proses *crawling* web dimulai dengan mengirimkan permintaan ke server web, jadi Requests adalah *library* yang sangat fundamental.
- 2) **BeautifulSoup:** "Library Python untuk *parsing* HTML dan XML."
  - a) **Bahasa Sederhana:** Ketika server web mengirimkan balik informasi halaman web, biasanya dalam bentuk kode HTML. Kode HTML ini seperti "bahasa" yang digunakan untuk menyusun halaman web (ada teks, gambar, link, dll.). BeautifulSoup membantu kita "membaca" kode HTML ini dan mengambil bagian-bagian tertentu yang kita inginkan.
  - b) **Lebih Detail:** BeautifulSoup membuat "pohon" dari kode HTML, sehingga kita bisa mencari elemen-elemen tertentu dengan mudah (misalnya, mencari semua judul artikel, atau semua link).
  - c) **Kenapa Penting:** Kode HTML seringkali berantakan dan sulit dibaca oleh manusia. BeautifulSoup membantu kita menavigasinya dengan rapi.
- 3) **Scrapy:** "Framework Python yang kuat untuk web *crawling* dan web *scraping*."
  - a) **Bahasa Sederhana:** Scrapy bukan hanya *library*, tapi sebuah "kerangka kerja". Ini seperti seperangkat alat lengkap untuk melakukan *crawling* web yang kompleks.
  - b) **Lebih Detail:** Scrapy menyediakan fitur-fitur canggih seperti:
    - i) **Konkurenси:** Melakukan banyak permintaan ke server web secara bersamaan untuk mempercepat proses *crawling*.
    - ii) **Manajemen Cookies:** Menangani *cookies* yang digunakan oleh situs web untuk melacak pengguna.
    - iii) **Pipeline Data:** Memproses dan menyimpan data yang di-*crawl* dalam berbagai format.
  - c) **Kenapa Penting:** Untuk proyek *crawling* web yang besar dan kompleks, Scrapy sangat membantu karena menyediakan struktur dan efisiensi.
- 4) **Selenium:** "Alat untuk mengotomatiskan *browser* web."
  - a) **Bahasa Sederhana:** Beberapa situs web menggunakan JavaScript untuk memuat kontennya. Ini bisa membuat *crawling* dengan Requests dan BeautifulSoup saja tidak cukup. Selenium memungkinkan kita untuk membuka *browser* web sungguhan (misalnya, Chrome atau Firefox) melalui program Python, sehingga kita bisa melihat halaman web seperti yang dilihat pengguna biasa.
  - b) **Lebih Detail:** Selenium bisa melakukan banyak hal yang bisa dilakukan pengguna di *browser*, misalnya:

- i) Mengklik tombol.
  - ii) Mengisi formulir.
  - iii) Menggulir halaman.
- c) **Kenapa Penting:** Selenium sangat berguna untuk *crawling* situs web modern yang banyak menggunakan JavaScript.

#### Penjelasan Tambahan tentang Library:

- "Pemilihan *library* tergantung pada kompleksitas tugas *crawling*. Untuk tugas sederhana, Requests dan BeautifulSoup mungkin cukup. Untuk tugas yang lebih kompleks, Scrapy atau Selenium mungkin diperlukan."
- "Penting untuk memahami dokumentasi dan penggunaan yang tepat dari setiap *library*."
- **Intinya:** Kita harus memilih "alat" yang tepat untuk pekerjaan yang tepat.

#### 4. Alamat Crawl yang Dituju: Memilih Target yang Tepat

Bagian ini membahas tentang dari mana kita akan mengambil data. Ini seperti menentukan "lokasi" yang akan kita "gali" untuk mendapatkan informasi.

**1. Platform X (Twitter):** "Twitter adalah sumber data yang kaya untuk analisis sentimen, tren topik, dan opini publik." "Data Twitter dapat diakses melalui Twitter API menggunakan *library* seperti Tweepy."

- **Bahasa Sederhana:** Twitter adalah tempat yang sangat ramai dengan opini dan informasi. Kita bisa mencari tahu apa yang sedang "hot" dibicarakan, bagaimana perasaan orang tentang suatu topik, dan banyak lagi.
- **Lebih Detail:**
  - **Analisis Sentimen:** Mencari tahu apakah orang-orang lebih banyak bicara positif atau negatif tentang suatu produk atau layanan.
  - **Tren Topik:** Menemukan topik apa yang sedang banyak dibicarakan.
  - **Opini Publik:** Memahami pendapat masyarakat tentang isu-isu tertentu.
- **Kenapa Penting:** Twitter memberikan "suara" langsung dari banyak orang, yang bisa sangat berharga untuk riset pasar, analisis sosial, dan lain-lain.

**2. Website E-commerce/Marketplace (Rating Pelanggan):** "Rating pelanggan adalah sumber informasi yang berharga untuk memahami kepuasan pelanggan, mengidentifikasi area perbaikan, dan menganalisis tren produk."

- **Bahasa Sederhana:** Ketika kita berbelanja online, kita sering melihat rating dan ulasan produk. Informasi ini sangat berguna untuk mengetahui apakah pelanggan puas atau tidak.
- **Lebih Detail:**
  - **Kepuasan Pelanggan:** Mengetahui seberapa puas pelanggan dengan produk atau layanan kita.
  - **Area Perbaikan:** Menemukan bagian mana dari produk atau layanan yang perlu ditingkatkan.
  - **Tren Produk:** Melihat produk mana yang sedang populer dan disukai.
- **Kenapa Penting:** Data rating pelanggan membantu bisnis untuk membuat keputusan yang lebih baik tentang produk dan layanan mereka.

#### Target Crawling Lainnya:

- "Situs Berita: Untuk analisis berita, tren topik, dan sentimen publik."
- "Forum Online: Untuk memahami diskusi komunitas dan opini pengguna."

- "Blog: Untuk analisis konten dan identifikasi *influencer*."
- "API: Banyak situs web dan platform menyediakan API yang memungkinkan akses terstruktur ke data mereka. Ini seringkali merupakan cara yang lebih efisien dan legal untuk mengumpulkan data daripada web *scraping* langsung."

**Bahasa Sederhana:** Selain Twitter dan *e-commerce*, kita juga bisa mengambil data dari berbagai tempat lain di internet, seperti situs berita, forum diskusi, dan blog. Banyak juga situs yang menyediakan "pintu" khusus (API) untuk mengambil data dengan lebih mudah.

#### Lebih Detail:

- **Situs Berita:** Menganalisis berita untuk mengetahui isu-isu penting dan bagaimana media memberitakannya.
- **Forum Online:** Memahami apa yang dibicarakan oleh komunitas online tentang topik tertentu.
- **Blog:** Mencari tahu siapa saja orang-orang yang berpengaruh dalam suatu bidang (*influencer*) dan apa yang mereka tulis.
- **API:** Menggunakan API lebih disarankan daripada *web scraping* langsung jika tersedia, karena lebih stabil dan legal.

- **Kenapa Penting:** Internet adalah sumber informasi yang sangat luas, dan kita bisa mendapatkan berbagai jenis data dari berbagai tempat.

#### Pertimbangan dalam Memilih Target Crawling:

- "Relevansi: Apakah data dari target tersebut relevan dengan tujuan analisis Anda?"
- "Ketersediaan: Apakah data tersebut dapat diakses secara publik atau memerlukan otentifikasi?"
- "Struktur Situs Web: Seberapa mudah atau sulit untuk menavigasi dan mengekstrak data dari situs web tersebut?"
- "Legalitas dan Etika: Apakah diperbolehkan untuk melakukan *crawling* data dari situs web tersebut? Pastikan untuk mematuhi *terms of service* dan *robots.txt*."
- **Bahasa Sederhana:** Sebelum kita mulai *crawling*, kita perlu memikirkan beberapa hal penting:
  - Apakah data yang akan kita ambil benar-benar berguna untuk apa yang ingin kita cari tahu?
  - Apakah kita bisa mengambil data tersebut? Apakah data tersebut tersedia untuk umum, atau kita perlu *login* atau izin khusus?
  - Apakah mudah untuk mengambil data dari situs web tersebut? Apakah situs web tersebut terstruktur dengan baik, atau berantakan?
  - Apakah *crawling* data dari situs web tersebut legal dan etis? Kita harus selalu mematuhi aturan situs web tersebut.
- **Kenapa Penting:** Memilih target *crawling* yang tepat sangat penting untuk mendapatkan data yang berkualitas dan relevan.

#### 5. Penyimpanan Data: Merapikan Hasil Crawling

- Setelah kita berhasil mengambil data, kita perlu menyimpannya dalam bentuk yang rapi dan terstruktur. Bagian ini membahas tentang cara menyimpan data hasil *crawling*.
- **Format CSV:**
  - "CSV adalah format file yang umum digunakan untuk menyimpan data tabular."
  - "Data disimpan dalam bentuk teks, dengan setiap baris mewakili satu *record* dan setiap nilai dipisahkan oleh koma."

- "CSV mudah dibaca dan ditulis oleh program komputer dan aplikasi *spreadsheet*."
- Bahasa Sederhana:** CSV ini seperti menyimpan data dalam bentuk tabel yang sederhana. Setiap baris adalah satu "item" data, dan setiap kolom dipisahkan oleh koma.
- Lebih Detail:** Contoh:
  - Nama,Umur,Kota
  - Andi,25,Jakarta
  - Budi,30,Surabaya
  - Bisa dibuka dengan mudah di Excel, Google Sheets, atau program lain.
- Kenapa Penting:** CSV adalah format yang umum dan mudah digunakan untuk menyimpan data tabular.

- Pilihan Penyimpanan Data Lainnya:**

- "Database: Untuk data yang lebih besar dan kompleks, *database relasional* (misalnya, MySQL, PostgreSQL) atau *database NoSQL* (misalnya, MongoDB) mungkin lebih tepat."
- "JSON: Format yang umum digunakan untuk menyimpan data semi-terstruktur."
- "File Teks: Untuk data teks mentah tanpa struktur tabular."
- Bahasa Sederhana:** Selain CSV, ada beberapa cara lain untuk menyimpan data:
  - Database:** Seperti "gudang" yang lebih canggih untuk menyimpan data dalam jumlah besar dan kompleks. Cocok untuk data yang saling berhubungan.
  - JSON:** Format yang lebih fleksibel daripada CSV, cocok untuk menyimpan data yang tidak selalu berbentuk tabel.
  - File Teks:** Hanya menyimpan teks biasa, tanpa ada struktur khusus.

- Lebih Detail:**
  - Database Relasional:** Seperti MySQL dan PostgreSQL, menggunakan tabel dengan baris dan kolom, serta hubungan antar tabel.
  - Database NoSQL:** Seperti MongoDB, lebih fleksibel dalam menyimpan data yang tidak terstruktur.
  - JSON:** Menggunakan format "key-value pairs" seperti ini: {"nama": "Andi", "umur": 25}
- Kenapa Penting:** Ada berbagai cara untuk menyimpan data, dan kita harus memilih yang paling sesuai dengan kebutuhan kita.

- Pertimbangan dalam Penyimpanan Data:**

- "Volume Data: Seberapa banyak data yang akan disimpan?"
- "Struktur Data: Apakah data tersebut terstruktur, semi-terstruktur, atau tidak terstruktur?"
- "Kebutuhan Akses: Bagaimana data akan diakses dan digunakan di masa mendatang?"
- "Skalabilitas: Apakah solusi penyimpanan data dapat menangani pertumbuhan data di masa mendatang?"
- Bahasa Sederhana:** Ketika memilih cara menyimpan data, kita perlu memikirkan beberapa hal:
  - Seberapa banyak data yang akan kita simpan?
  - Apakah data kita rapi seperti tabel, atau lebih berantakan?

- Bagaimana kita akan menggunakan data tersebut nanti? Apakah kita perlu mengaksesnya dengan cepat, atau hanya menyimpannya saja?
- Apakah kita perlu menyimpan lebih banyak data di masa mendatang? Apakah "gudang" kita bisa diperbesar?
- **Kenapa Penting:** Memilih cara penyimpanan data yang tepat sangat penting untuk efisiensi dan kemudahan penggunaan data di masa mendatang.

## 6. Etika dan Legalitas Crawling Data

- Ini adalah aspek penting yang tidak dibahas secara rinci dalam PowerPoint, tetapi sangat krusial.
- Robots.txt:** File yang di-host oleh situs web yang menginstruksikan *web crawler* halaman mana yang boleh dan tidak boleh diakses. Penting untuk menghormati arahan dalam robots.txt.
- Terms of Service:** Perjanjian hukum antara penyedia layanan dan pengguna yang mengatur penggunaan layanan. Pastikan untuk mematuhi *terms of service* dari situs web yang Anda crawl. Beberapa situs web secara eksplisit melarang *web crawling*.
- Hukum Hak Cipta:** Jangan crawl dan menggunakan data yang dilindungi hak cipta tanpa izin.
- Privasi Data:** Berhati-hatilah terhadap data pribadi (misalnya, nama, alamat email). Patuh peraturan privasi yang relevan (misalnya, GDPR).
- Beban Server:** Jangan membebani server web dengan terlalu banyak permintaan dalam waktu singkat. Gunakan teknik seperti *delay* dan *caching* untuk mengurangi dampak pada server.

### Langkah-Langkah Crawling Data: Panduan Komprehensif

Crawling data, juga dikenal sebagai web crawling atau spidering, adalah proses otomatis untuk mengumpulkan data dari berbagai sumber digital, terutama halaman web. Proses ini sangat penting dalam berbagai aplikasi, termasuk text mining, analisis sentimen, dan pengembangan mesin pencari.

#### 1. Persiapan Crawling Data

Sebelum memulai proses crawling, ada beberapa persiapan penting yang perlu dilakukan. Persiapan ini akan memastikan proses crawling berjalan lancar, efisien, dan sesuai dengan tujuan pengumpulan data.

##### a) Menentukan Tujuan Crawling

- Hal pertama yang perlu dilakukan adalah menentukan dengan jelas tujuan dari crawling data.
- Apa jenis data yang ingin dikumpulkan?
- Informasi apa yang dibutuhkan?
- Mengapa data tersebut dikumpulkan?
- Contoh: Mengumpulkan ulasan pelanggan dari situs e-commerce, mengumpulkan posting dari platform media sosial, atau mengumpulkan artikel berita dari situs web berita.

##### b) Identifikasi Sumber Data

- Setelah tujuan ditentukan, langkah selanjutnya adalah mengidentifikasi sumber data yang relevan.
- Sumber data dapat berupa situs web, platform media sosial, forum online, atau sumber digital lainnya.
- Penting untuk membuat daftar alamat web atau URL yang akan di-crawl.

- (4) Contoh: Jika tujuannya adalah mengumpulkan ulasan pelanggan, sumber data mungkin adalah halaman produk di situs web e-commerce.

- c) **Memahami Struktur Situs Web**

- Jika sumber data adalah situs web, penting untuk memahami struktur situs web tersebut.
- Bagaimana halaman-halaman web diatur?
- Bagaimana data yang diinginkan ditampilkan?
- Apakah ada pola URL yang dapat diikuti?
- Pemahaman ini akan membantu dalam merancang strategi crawling yang efektif.

- d) **Pertimbangan Etika dan Legal**

- Crawling data harus dilakukan dengan mematuhi etika dan hukum yang berlaku.
- Periksa robots.txt: Situs web menggunakan file robots.txt untuk memberikan instruksi kepada web crawler tentang halaman mana yang boleh dan tidak boleh di-crawl.
- Hormati batasan kecepatan: Jangan membebani server web dengan permintaan yang berlebihan.
- Gunakan API jika tersedia: Jika situs web menyediakan API, gunakanlah API tersebut untuk mengakses data. API umumnya lebih efisien dan legal daripada crawling langsung.
- Patuhi hak cipta dan lisensi: Pastikan penggunaan data yang dikumpulkan tidak melanggar hak cipta atau lisensi apa pun.

- e) **Pemilihan Tools dan Library**

- Ada berbagai tools dan library yang dapat digunakan untuk crawling data.
- Pemilihan tools tergantung pada jenis data yang akan di-crawl, kompleksitas tugas, dan preferensi pemrograman.
- Python:** Bahasa pemrograman yang populer untuk web crawling karena *simplicity* dan banyak library yang tersedia.
- Library Python:**
  - Requests: Untuk mengirimkan permintaan HTTP ke server web.
  - BeautifulSoup: Untuk *parsing* HTML dan XML.
  - Scrapy: Framework web crawling yang *powerful* untuk tugas crawling yang kompleks.
  - Selenium: Untuk *automating* browser web, berguna untuk situs web yang menggunakan JavaScript untuk memuat konten.
  - Tweepy: Library untuk mengakses Twitter API.
  - csv: Library untuk bekerja dengan file CSV.

- (5) **Tools Lain:**

- Google Colab atau Jupyter Notebook: Lingkungan pengembangan interaktif untuk menulis dan menjalankan kode Python.

- f) **Perencanaan Penyimpanan Data**

- Tentukan bagaimana data yang dikumpulkan akan disimpan.
- Format penyimpanan yang umum adalah file CSV, database, atau file JSON.
- Pilih format yang sesuai dengan kebutuhan analisis data.
- Pastikan tempat penyimpanan data memiliki kapasitas yang cukup.

## 2. Proses Crawling Data

Setelah persiapan selesai, langkah selanjutnya adalah melaksanakan proses crawling data. Proses ini melibatkan beberapa tahapan penting:

- Pengiriman Permintaan (Request)**

- a) Crawler mengirimkan permintaan HTTP ke server web untuk mendapatkan halaman web.
  - b) Permintaan ini mirip dengan ketika Anda mengetikkan URL di browser web Anda.
  - c) Library seperti Requests di Python memudahkan pengiriman permintaan HTTP.
- 2) Penerimaan Respon (Response)**
- a) Server web merespons dengan mengirimkan kode status dan konten halaman web (biasanya dalam format HTML).
  - b) Kode status menunjukkan apakah permintaan berhasil (misalnya, kode 200 OK) atau gagal (misalnya, kode 404 Not Found).
- 3) Parsing HTML**
- a) Crawler perlu mengurai (parse) kode HTML untuk mengekstrak data yang diinginkan.
  - b) HTML adalah bahasa markup yang digunakan untuk membuat halaman web.
  - c) Library seperti BeautifulSoup membantu dalam menavigasi struktur HTML dan mencari elemen-elemen tertentu (misalnya, judul, paragraf, tautan).
- 4) Ekstraksi Data**
- a) Setelah HTML diurai, crawler mengekstrak data yang relevan.
  - b) Data dapat berupa teks, gambar, tautan, atau jenis konten lainnya.
  - c) Crawler menggunakan teknik seperti pemilihan CSS atau XPath untuk menemukan dan mengekstrak elemen HTML tertentu.
- 5) Penyimpanan Data**
- a) Data yang diekstrak disimpan dalam format yang telah ditentukan sebelumnya.
  - b) Data dapat disimpan dalam file CSV, database, atau format lainnya.
- 6) Iterasi dan Navigasi**
- a) Crawler dapat mengikuti tautan di halaman web untuk menemukan halaman lain dan mengulangi proses crawling.
  - b) Ini memungkinkan crawler untuk menjelajahi seluruh situs web atau sebagian dari situs web.
  - c) Crawler harus dikonfigurasi untuk menghindari *loop* tak terbatas dan mengunjungi halaman yang tidak relevan.

### 3. Teknik Crawling Lanjutan

Selain langkah-langkah dasar di atas, ada beberapa teknik crawling lanjutan yang dapat digunakan untuk tugas-tugas yang lebih kompleks:

- 1) Crawling Dinamis**
  - a) Beberapa situs web menggunakan JavaScript untuk memuat konten secara dinamis.
  - b) Crawler tradisional mungkin tidak dapat mengeksekusi JavaScript, sehingga tidak dapat melihat konten dinamis.
  - c) Tools seperti Selenium dapat digunakan untuk mengotomatiskan browser web dan merender halaman web sepenuhnya, termasuk konten dinamis.
- 2) Crawling Terdistribusi**
  - a) Untuk tugas crawling yang sangat besar, crawling dapat didistribusikan di beberapa mesin.
  - b) Ini mempercepat proses crawling dan mengurangi beban pada satu server.
  - c) Framework seperti Scrapy-Redis dapat digunakan untuk mengimplementasikan crawling terdistribusi.
- 3) Manajemen Sesi dan Cookie**
  - a) Beberapa situs web memerlukan *session* atau *cookie* untuk mengakses konten tertentu.
  - b) Crawler perlu dikonfigurasi untuk mengelola *session* dan *cookie* dengan benar.
- 4) Penanganan Anti-Crawling**
  - a) Beberapa situs web menerapkan teknik anti-crawling untuk mencegah web crawler mengakses data mereka.

- b) Teknik ini dapat berupa CAPTCHA, pemblokiran IP, atau deteksi pola crawling.
- c) Crawler yang baik harus dapat mengatasi teknik anti-crawling ini dengan bijaksana (misalnya, dengan menggunakan rotasi IP atau meniru perilaku manusia).

### 4. Ringkasan Langkah-Langkah Crawling Data

- a) Persiapan:**
  - (1) Tentukan tujuan crawling.
  - (2) Identifikasi sumber data.
  - (3) Pahami struktur situs web.
  - (4) Pertimbangkan etika dan legal.
  - (5) Pilih tools dan library.
  - (6) Rencanakan penyimpanan data.
- b) Proses Crawling:**
  - (1) Kirim permintaan (request).
  - (2) Terima respons (response).
  - (3) Parsing HTML.
  - (4) Ekstraksi data.
  - (5) Simpan data.
  - (6) Iterasi dan navigasi.
- c) Teknik Crawling Lanjutan:**
  - (1) Crawling dinamis.
  - (2) Crawling terdistribusi.
  - (3) Manajemen sesi dan cookie.
  - (4) Penanganan anti-crawling.

### 7. Kesimpulan

- Persiapan yang matang sangat penting untuk keberhasilan proyek *crawling* data.
- Memilih *tools* dan *library* yang tepat, menentukan target *crawling* yang relevan, merencanakan penyimpanan data, dan mempertimbangkan etika dan legalitas adalah langkah-langkah penting dalam proses ini.
- Dengan mengikuti panduan ini, Anda dapat membangun dasar yang kuat untuk pengumpulan data yang efektif dan bertanggung jawab.

## Chapter 3: Preprocessing Text

### 1. Pendahuluan: Mengapa Preprocessing Teks Sangat Penting?

PowerPoint Anda dengan tepat menyatakan bahwa "Preprocessing text adalah tahapan krusial dalam text mining dan Natural Language Processing (NLP)." Mari kita dalami mengapa ini *sangat* penting.

#### 1.1 Data Teks Mentah: Kekacauan yang Bermakna

- Data teks mentah, yaitu teks yang kita kumpulkan langsung dari berbagai sumber (misalnya, media sosial, artikel berita, dokumen), seringkali diibaratkan sebagai "kekacauan". Namun, di dalam "kekacauan" ini, terdapat "bermakna" yang sangat berharga.
- **Masalah dengan Data Teks Mentah:**
  - **Tidak Terstruktur:** Teks tidak memiliki format yang rapi seperti tabel dalam database. Teks mengalir bebas dalam kalimat dan paragraf.

- **Noise:** Teks mengandung banyak "noise" atau elemen yang tidak relevan untuk analisis, seperti:
  - Tanda baca yang berlebihan atau tidak konsisten.
  - Karakter-karakter aneh atau tidak terbaca.
  - Kode-kode HTML atau markup lainnya (jika diambil dari web).
  - Informasi yang tidak relevan (misalnya, iklan dalam dokumen).
- **Variasi Bahasa:**
  - Teks dapat ditulis dalam berbagai gaya, dengan tata bahasa yang benar atau salah, menggunakan slang, singkatan, atau bahasa informal.
  - Satu ide yang sama dapat diekspresikan dengan banyak cara yang berbeda.
- **Ambiguitas:**
  - Kata-kata memiliki banyak arti (polisemi).
  - Makna kalimat dapat berubah tergantung pada konteks.

### 1.2 Analogi: Memasak vs. Mengolah Teks

- Untuk memahami pentingnya preprocessing, mari gunakan analogi memasak.
- **Bahan Mentah:** Data teks mentah seperti bahan-bahan mentah seperti sayuran yang kotor, daging yang belum dipotong, dan bumbu yang belum dihaluskan.
- **Preprocessing:** Proses preprocessing teks seperti proses membersihkan, memotong, mengiris, dan membumbui bahan-bahan tersebut.
- **Hidangan Lezat:** Data teks yang sudah dipreprocessing seperti bahan-bahan yang siap dimasak menjadi hidangan yang lezat (analisis yang akurat dan bermakna).
- Tanpa preprocessing yang baik, kita tidak bisa mendapatkan "hidangan" yang baik. Analisis kita akan menjadi "berantakan", tidak akurat, dan sulit diinterpretasikan.

### 1.3 Tujuan Utama Preprocessing Teks

- Tujuan utama preprocessing teks adalah untuk mengubah data teks mentah menjadi format yang lebih bersih, terstruktur, dan sesuai untuk dianalisis oleh komputer.
- **Manfaat Preprocessing Teks:**
  - **Meningkatkan Kualitas Data:** Mengurangi noise dan inkonsistensi.
  - **Memudahkan Komputasi:** Membuat teks lebih mudah diproses oleh algoritma.
  - **Meningkatkan Akurasi Analisis:** Memastikan bahwa analisis kita fokus pada informasi yang relevan.
  - **Mengurangi Dimensi Data:** Menyederhanakan data dengan menghilangkan informasi yang berlebihan.

### 1.4 Hubungan dengan NLP dan Text Mining

- Preprocessing teks adalah langkah *esensial* dalam Natural Language Processing (NLP) dan Text Mining.
- **Natural Language Processing (NLP):** NLP adalah bidang ilmu komputer yang berfokus pada interaksi antara komputer dan bahasa manusia. Preprocessing teks adalah fondasi bagi banyak tugas NLP, seperti:

- Analisis sentimen (menentukan apakah teks bersifat positif, negatif, atau netral).
- Klasifikasi teks (mengategorikan teks ke dalam topik tertentu).
- Ekstraksi informasi (mengambil informasi penting dari teks).
- **Text Mining:** Text mining adalah proses mengekstrak informasi yang berguna dari data teks. tagPreprocessing teks adalah langkah awal yang krusial dalam text mining untuk memastikan bahwa data teks siap untuk ditambah.

## 1.5 Contoh Dampak Preprocessing

Bayangkan kita ingin menganalisis ulasan pelanggan tentang sebuah produk.

- **Tanpa Preprocessing:** Jika kita langsung menganalisis teks mentah, hasilnya mungkin kacau karena ada banyak variasi dalam cara orang menulis, ada kesalahan ejaan, ada kata-kata yang tidak relevan, dll.
- **Dengan Preprocessing:** Jika kita melakukan preprocessing terlebih dahulu, kita bisa mendapatkan hasil yang lebih akurat tentang sentimen pelanggan (apakah mereka puas atau tidak), topik apa yang sering mereka bicarakan, dan lain-lain.

### Kesimpulan:

- Preprocessing teks adalah tahap yang sangat penting dan tidak boleh diabaikan.
- Ini adalah "fondasi" yang kuat untuk analisis teks yang berhasil.
- Dengan preprocessing yang baik, kita bisa mengubah "kekacauan" teks mentah menjadi informasi yang berharga.

Oke, ini dia penulisan ulang bagian "2. Langkah-Langkah Preprocessing Text" dengan format yang lebih rapi dan mudah dicopy ke Word, tanpa indentasi berlebihan:

## 2. Langkah-Langkah Preprocessing Text: Penjelasan Mendalam

Preprocessing text adalah serangkaian teknik yang digunakan untuk membersihkan dan mentransformasi teks mentah menjadi format yang lebih sesuai untuk analisis. Ini adalah langkah penting karena data teks mentah seringkali berantakan, tidak terstruktur, dan mengandung noise yang dapat mengganggu algoritma pemrosesan bahasa alami (NLP) dan text mining. Berikut adalah penjelasan mendalam tentang setiap langkah preprocessing text:

- **Cleaning:** "Pembersihan dari karakter atau elemen yang tidak diperlukan pada teks, seperti tanda baca, angka, dan karakter khusus."

**Penjelasan Lebih Detail:** Cleaning adalah tahap awal yang bertujuan untuk menghilangkan "noise" atau elemen-elemen yang tidak relevan dari teks. Noise ini bisa berupa berbagai macam hal, tergantung pada sumber dan jenis teksnya.

### ● Jenis-jenis Cleaning yang Umum Dilakukan:

- Menghapus Tanda Baca: Tanda baca seperti koma (,), titik (.), tanda seru (!), tanda tanya (?), dan lain-lain, seringkali tidak memberikan kontribusi signifikan terhadap makna teks dalam konteks analisis tertentu. Contoh:
  - Teks Asli: "Halo, apa kabar? Saya baik-baik saja!"
  - Teks Setelah Menghapus Tanda Baca: "Halo apa kabar Saya baik-baik saja"
- Menghapus Angka: Angka mungkin tidak relevan jika fokus analisis adalah pada aspek linguistik atau makna kata. Namun, dalam beberapa kasus (misalnya, analisis numerik), angka mungkin perlu dipertahankan. Contoh:

- Teks Asli: "Harga barang itu Rp 10.000"
  - Teks Setelah Menghapus Angka: "Harga barang itu Rp "
- Menghapus Karakter Khusus: Karakter-karakter seperti simbol mata uang (\$, €, Rp), simbol matematika (+, -, =), dan karakter non-ASCII (misalnya, karakter emoji) mungkin perlu dihilangkan. Contoh:
  - Teks Asli: "Diskon 50% 🎉 untuk semua produk!"
  - Teks Setelah Menghapus Karakter Khusus: "Diskon untuk semua produk"
- Menghapus URL (Uniform Resource Locator): Alamat web (URL) biasanya tidak relevan dalam analisis teks dan dapat dihilangkan. Contoh:
  - Teks Asli: "Kunjungi website kami di ."
  - Teks Setelah Menghapus URL: "Kunjungi website kami di "
- Menghapus HTML Tag: Jika teks diambil dari halaman web, kode HTML (misalnya, <p>, <br>, <a>) perlu dihilangkan agar tidak mengganggu analisis teks. Contoh:
  - Teks Asli: "<p>Ini adalah sebuah paragraf.</p>"
  - Teks Setelah Menghapus HTML Tag: "Ini adalah sebuah paragraf."
- Menghilangkan Mention/User: Dalam teks media sosial, "mention" (misalnya, @username) seringkali mengacu pada pengguna lain dan mungkin tidak relevan untuk analisis konten. Contoh:
  - Teks Asli: "@Andi Hai, apa kabar?"
  - Teks Setelah Menghilangkan Mention: "Hai, apa kabar?"
- Menghilangkan Hashtag: Hashtag (#kata) digunakan untuk menandai topik di media sosial. Tanda pagar (#) dapat dihilangkan, tetapi kata setelahnnya (kata) mungkin relevan. Contoh:
  - Teks Asli: "Saya suka #kucing"
  - Teks Setelah Menghilangkan Hashtag (tanda pagar): "Saya suka kucing"
- Tujuan Cleaning:
  - Mengurangi noise dalam data teks.
  - Memudahkan pemrosesan teks oleh komputer.
  - Meningkatkan akurasi analisis teks.
- Catatan Penting: Jenis cleaning yang diperlukan dapat bervariasi tergantung pada sumber data dan tujuan analisis. Penting untuk mempertimbangkan dengan cermat elemen mana yang perlu dihilangkan dan mana yang perlu dipertahankan.

## 2.2 Case Folding: "Mengubah semua huruf menjadi huruf kecil semua."

**Penjelasan Lebih Detail:** Case folding adalah proses mengubah semua huruf dalam teks menjadi huruf kecil (atau huruf besar, meskipun kurang umum).

- Tujuan Case Folding:
  - Menyeragamkan teks.
  - Mengurangi variasi kata.
  - Contoh: Kata "Indonesia," "INDONESIA," dan "indonesia" akan dianggap sebagai kata yang sama setelah case folding.
- Contoh:
  - Teks Asli: "Saya Ingin Pergi Ke JAKARTA"
  - Teks Setelah Case Folding: "saya ingin pergi ke jakarta"

- Mengapa Case Folding Penting? Komputer membedakan antara huruf kapital dan huruf kecil. Tanpa case folding, kata "Saya" dan "saya" akan dianggap sebagai dua kata yang berbeda, meskipun memiliki arti yang sama. Ini dapat mempengaruhi akurasi analisis teks.
- Pengecualian: Dalam beberapa kasus, case folding mungkin tidak diinginkan. Misalnya:
  - Analisis Sentimen: Huruf kapital dapat menunjukkan intensitas emosi (misalnya, "SENANG" vs. "senang").
  - Named Entity Recognition (NER): Huruf kapital seringkali digunakan untuk mengidentifikasi nama orang, organisasi, dan lokasi.

## 2.3 Tokenizing: "Proses memecah teks menjadi unit-unit kecil seperti kata atau kalimat."

**Penjelasan Lebih Detail:** Tokenizing adalah proses memecah aliran teks menjadi potongan-potongan yang lebih kecil, yang disebut "token". Token ini bisa berupa kata, frasa, simbol, atau elemen lainnya.

- Tujuan Tokenizing:
  - Mempersiapkan teks untuk analisis lebih lanjut.
  - Memudahkan komputer untuk memahami struktur teks.
- Jenis-jenis Tokenizing:
  - Word Tokenizing: Memecah teks menjadi kata-kata individu. Ini adalah jenis tokenizing yang paling umum. Contoh:
    - Teks Asli: "Saya suka makan nasi goreng."
    - Token: ['Saya', 'suka', 'makan', 'nasi', 'goreng', '.']
  - Sentence Tokenizing: Memecah teks menjadi kalimat-kalimat individu. Contoh:
    - Teks Asli: "Saya pergi ke pasar. Saya membeli buah."
    - Token: ['Saya pergi ke pasar.', 'Saya membeli buah.']}
- Tantangan dalam Tokenizing:
  - Menangani tanda baca yang menempel pada kata (misalnya, "dimana?").
  - Menangani kontraksi (misalnya, "isn't," "can't").
  - Menangani kata majemuk (misalnya, "nasi goreng").
- Tools untuk Tokenizing:
  - NLTK (Natural Language Toolkit) menyediakan berbagai tokenizer, termasuk word\_tokenize, sent\_tokenize, dan TweetTokenizer. TweetTokenizer dirancang khusus untuk menangani teks dari media sosial, yang seringkali mengandung emoticon, singkatan, dan bahasa informal.
  - spaCy adalah library NLP lain yang kuat yang juga menyediakan tokenizer yang efisien.

## 2.4 Filtering (Stopword Removal): "Menghapus kata-kata yang tidak memiliki makna dalam teks, seperti 'dan', 'di', 'ke', dll."

**Penjelasan Lebih Detail:** Stopword removal adalah proses menghilangkan kata-kata umum yang dianggap tidak memberikan kontribusi signifikan terhadap makna teks. Kata-kata ini disebut "stopword."

- Contoh Stopword dalam Bahasa Indonesia:

- "dan," "di," "ke," "dari," "yang," "adalah," "untuk," "pada," "dengan," "ini," "itu," "sebuah," "suatu," dll.
- Tujuan Stopword Removal:
  - Mengurangi ukuran data teks.
  - Mempercepat pemrosesan teks.
  - Meningkatkan efisiensi analisis dengan memfokuskan pada kata-kata yang lebih bermakna.
- Mengapa Stopword Removal Penting? Stopword seringkali muncul dengan frekuensi tinggi dalam teks, tetapi mengandung sedikit informasi tentang konten dokumen. Menghapusnya dapat membantu algoritma untuk lebih fokus pada kata-kata penting.
- Perhatian:
  - Daftar stopword dapat bervariasi tergantung pada bahasa dan konteks analisis.
  - Dalam beberapa kasus, stopword mungkin penting (misalnya, dalam analisis gaya penulisan atau analisis sentimen yang sangat halus).
  - NLTK menyediakan daftar stopword untuk berbagai bahasa, termasuk bahasa Indonesia (stopwords\_indonesian).

## 2.5 Stemming: "Mengubah kata menjadi bentuk dasar."

**Penjelasan Lebih Detail:** Stemming adalah proses mengurangi kata-kata ke bentuk dasarnya atau "akar kata" dengan menghilangkan akhiran (suffix) dan/atau awalan (prefix).

- Tujuan Stemming:
  - Menyeragamkan kata-kata yang memiliki arti yang sama tetapi bentuk yang berbeda.
  - Mengurangi ukuran kosakata (jumlah kata unik) dalam teks.
- Contoh:
  - Kata: "berlari," "berlari," "berlarilah"
  - Stem: "lari"
- Algoritma Stemming: Ada berbagai algoritma stemming, yang berbeda dalam aturan dan tingkat agresivitasnya.
  - Porter Stemmer: Algoritma yang umum digunakan untuk bahasa Inggris.
  - Sastrawi Stemmer: Algoritma yang dirancang untuk bahasa Indonesia.
- Contoh (Sastrawi):
  - Kata: "memakan"
  - Stem: "makan"
- Perbedaan dengan Lemmatization: Lemmatization juga bertujuan untuk mengubah kata ke bentuk dasarnya, tetapi dengan cara yang lebih cerdas. Lemmatization mempertimbangkan konteks kata dan menggunakan kamus untuk menghasilkan "lemma" (bentuk dasar yang valid). Contoh:
  - Kata: "adalah," "merupakan," "yaitu"
  - Lemma: "adalah"
- Lemmatization umumnya lebih akurat daripada stemming, tetapi lebih kompleks dan membutuhkan sumber daya komputasi yang lebih besar.
- Mengapa Stemming Penting? Stemming membantu untuk mengelompokkan kata-kata yang memiliki arti yang sama tetapi bentuk yang berbeda, sehingga mengurangi dimensi data dan meningkatkan efisiensi analisis.

Ringkasan:

- Preprocessing text adalah langkah penting untuk menyiapkan data teks untuk analisis.
- Setiap langkah dalam preprocessing memiliki tujuan tertentu dan dapat mempengaruhi hasil analisis.
- Pemilihan langkah-langkah preprocessing yang tepat tergantung pada jenis data teks dan tujuan analisis.

## 3. Implementasi dalam Python: Langkah demi Langkah

Bagian ini menunjukkan bagaimana langkah-langkah preprocessing text yang telah dijelaskan sebelumnya diimplementasikan dalam kode Python. Kode ini menggunakan beberapa library populer untuk data science dan NLP. Berikut adalah penjelasan detail kode yang diberikan dalam PowerPoint:

### 3.1 Import Library

```
import nltk
import string
import re
import pandas as pd
import numpy as np
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
```

Penjelasan:

- nltk:** Natural Language Toolkit. Library Python yang populer untuk NLP. Digunakan di sini untuk tokenisasi dan stopword removal.
- string:** Modul bawaan Python untuk operasi string. Digunakan untuk membersihkan tanda baca.
- re:** Modul bawaan Python untuk *regular expression*. Digunakan untuk mencari dan mengganti pola teks.
- pandas:** Library untuk manipulasi dan analisis data. Digunakan untuk membaca data dari file CSV dan menyimpannya dalam DataFrame.
- numpy:** Library untuk komputasi numerik.
- Sastrawi:** Library untuk stemming bahasa Indonesia.
  - StemmerFactory:** Kelas dari Sastrawi untuk membuat objek stemmer.
  - stemmer = factory.create\_stemmer():** Membuat objek stemmer menggunakan algoritma Sastrawi.
- nltk.tokenize.TweetTokenizer:** Kelas dari NLTK untuk tokenisasi teks, khususnya dirancang untuk teks media sosial.
- nltk.corpus.stopwords:** Modul dari NLTK yang menyediakan daftar stopword untuk berbagai bahasa.
  - stopwords\_indonesia = stopwords.words('indonesian'):** Mengambil daftar stopword untuk bahasa Indonesia.

### 3.2 Membaca Data

```
def load_data():
```

```
data = pd.read_csv('tomlembong.csv')
```

```
return data
```

```
df_twit = load_data()
```

```
df_twit.head(10)
```

Penjelasan:

- load\_data():** Fungsi untuk membaca data dari file CSV (tomlembong.csv) menggunakan pd.read\_csv() dari library pandas.
- df\_twit = load\_data():** Memanggil fungsi load\_data() untuk membaca data dan menyimpannya dalam DataFrame bernama df\_twit.
- df\_twit.head(10):** Menampilkan 10 baris pertama dari DataFrame df\_twit untuk melihat contoh data.

### 3.3 Memilih Kolom Teks

```
df = pd.DataFrame(df_twit[['teks']])
```

```
df.head(10)
```

Penjelasan:

- df = pd.DataFrame(df\_twit[['teks']]):** Membuat DataFrame baru bernama df yang hanya berisi kolom 'teks' dari DataFrame df\_twit. Ini diasumsikan bahwa kolom yang berisi teks yang akan diprocessing bernama 'teks'.
- df.head(10):** Menampilkan 10 baris pertama dari DataFrame df untuk memverifikasi bahwa kolom teks telah dipilih dengan benar.

### 3.4 Menghilangkan Mention/User

```
def remove_pattern(tweet, pattern):
```

```
r = re.findall(pattern, tweet)
for i in r:
    tweet = re.sub(i, '', tweet)
return tweet
```

```
df['remove_user'] = np.vectorize(remove_pattern)(df['teks'],
"@[\w]*")
```

Penjelasan:

- remove\_pattern(tweet, pattern):** Fungsi untuk menghilangkan pola tertentu dari teks.
  - re.findall(pattern, tweet):** Mencari semua kemunculan pattern dalam tweet.
  - Loop for i in r:** Iterasi melalui semua kemunculan pola yang ditemukan.
  - re.sub(i, '', tweet):** Mengganti setiap kemunculan pola dengan string kosong (menghapusnya).
- df['remove\_user'] = np.vectorize(remove\_pattern)(df['teks'], "@[\w]\*"):** Membuat kolom baru bernama 'remove\_user' dalam DataFrame df.

- `np.vectorize(remove_pattern)`: Menerapkan fungsi `remove_pattern` ke setiap elemen dalam kolom 'teks'.
- `@[w]*`: Regular expression untuk mencari "mention" di Twitter (karakter '@' diikuti oleh nol atau lebih karakter word).

### 3.5 Cleaning Teks

```
def tweet_clean(tweet):
    tweet = re.sub('[0-9]+', '', tweet)
    tweet = re.sub(r'\$\\w*', '', tweet)
    tweet = re.sub(r'RT: [\\s]+', '', tweet)
    tweet = re.sub(r'\\https?:\\/\\.*[\\r\\n]*', '', tweet)
    tweet = re.sub(r',', '', tweet)
    tweet = re.sub(r'#', '', tweet)

    emoticons_happy = set([':-)', ':)', ';)', ':o)', ':]', ':3',
    ':c)', ':>', '=]', ':^)', ':-D', ':D', '8-D', '8D', '=-)', '=)',
    '=-3', '=3', ':-))', ":-)", ":", ":", '*', '^*', '>P', 'x-p',
    'xp', 'XP', ':P', ':p', ':<3'])

    emoticons_sad = set([':-l', ':-/', '>:/', ':S', '>:[', ':@',
    ':-(', ':[', ':-||', ':-[', ':-<', '=\\', '=/', '>:(', ':(',
    '>.<', ":-(", ":-<", ':{', '>:\\', ':('])

    emoticons = emoticons_happy.union(emoticons_sad)

    tokenizer = TweetTokenizer(preserve_case=False,
    strip_handles=True, reduce_len=True)

    tweet_tokens = tokenizer.tokenize(tweet)

    tweets_clean = []
    for word in tweet_tokens:
        if (word not in stopwords_indonesia and word not in
            emoticons and word not in string.punctuation):
            stem_word = stemmer.stem(word)
            tweets_clean.append(stem_word)

    return tweets_clean

df['tweet_clean'] = df['remove_user'].apply(lambda x:
    tweet_clean(x))
```

Penjelasan:

- `tweet_clean(tweet)`: Fungsi untuk membersihkan teks dari berbagai elemen.
  - `re.sub('[0-9]+', '', tweet)`: Menghilangkan angka.
  - `re.sub(r'\$\\w*', '', tweet)`: Menghilangkan ticker simbol saham (misalnya, \$GE).
  - `re.sub(r'RT: [\\s]+', '', tweet)`: Menghilangkan teks "RT:" (retweet).
  - `re.sub(r'\\https?:\\/\\.*[\\r\\n]*', '', tweet)`: Menghilangkan URL.
  - `re.sub(r',', '', tweet)`: Menghilangkan koma.
  - `re.sub(r'#', '', tweet)`: Menghilangkan tanda pagar (#).
  - `emoticons_happy, emoticons_sad, emoticons`: Mendefinisikan set emoticon happy dan sad, lalu menggabungkannya.
  - `tokenizer = TweetTokenizer(preserve_case=False,
 strip_handles=True, reduce_len=True)`: Membuat objek TweetTokenizer dengan opsi:
    - `preserve_case=False`: Mengubah teks menjadi huruf kecil.
    - `strip_handles=True`: Menghilangkan `username` Twitter.
    - `reduce_len=True`: Mengurangi pengulangan karakter (misalnya, "haiii" menjadi "haii").
  - `tweet_tokens = tokenizer.tokenize(tweet)`: Tokenisasi teks menggunakan TweetTokenizer.
  - `Loop for word in tweet_tokens`: Iterasi melalui setiap token.
    - `if (word not in stopwords_indonesia and word not in
 emoticons and word not in string.punctuation)`: Menyaring token:
      - `word not in stopwords_indonesia`: Membuang stopword.
      - `word not in emoticons`: Membuang emoticon.
      - `word not in string.punctuation`: Membuang tanda baca.
    - `stem_word = stemmer.stem(word)`: Stemming kata menggunakan Sastrawi stemmer.
    - `tweets_clean.append(stem_word)`: Menambahkan kata yang sudah di-stem ke list `tweets_clean`.
  - `return tweets_clean`: Mengembalikan list token yang sudah dibersihkan dan di-stem.
- `df['tweet_clean'] = df['remove_user'].apply(lambda x: tweet_clean(x))`: Membuat kolom baru 'tweet\_clean' dengan menerapkan fungsi `tweet_clean` ke setiap teks di kolom 'remove\_user'.

### 3.6 Menyimpan Hasil

Kode untuk menyimpan hasil akhir tidak ada di potongan kode yang diberikan, tapi biasanya dilakukan dengan:

- `df.to_csv('teks_bersih.csv', index=False)`: Menyimpan DataFrame df ke file CSV baru.

#### Penjelasan Tambahan:

- Kode ini secara komprehensif melakukan preprocessing teks yang umum untuk teks media sosial.
- Penggunaan regular expression sangat penting untuk membersihkan pola-pola teks yang kompleks.

- Sastrawi adalah library yang berguna untuk menangani stemming dalam bahasa Indonesia, yang memiliki aturan morfologi yang berbeda dengan bahasa Inggris.

### 4. Tantangan dalam Preprocessing Text: Mengatasi Kerumitan

PowerPoint Anda dengan tepat menyebutkan bahwa "Preprocessing text bukan proses yang selalu mudah dan langsung." Ini karena bahasa manusia sangat kompleks dan penuh dengan nuansa. Berikut adalah penjelasan yang lebih mendalam tentang tantangan-tantangan utama dalam preprocessing text:

#### 4.1 Ambiguitas: Ketika Kata Memiliki Banyak Arti

"Beberapa langkah preprocessing dapat menghilangkan informasi yang penting untuk memahami makna teks. Contohnya, penghapusan stopword dapat mengubah makna kalimat dalam beberapa kasus."

**Penjelasan Lebih Detail:** Ambiguitas adalah salah satu tantangan terbesar dalam NLP. Kata-kata sering kali memiliki lebih dari satu arti (polisemi), dan makna yang tepat tergantung pada konteks kalimat. **Contoh:**

- Kata "bisa" dapat berarti "mampu" atau "racun".
- Kalimat "Saya melihat kelelawar" bisa berarti "Saya melihat hewan kelelawar" atau "Saya melihat alat pemukul bola".

**Dampak Preprocessing:** Beberapa langkah preprocessing, seperti penghapusan stopword, dapat memperburuk masalah ambiguitas. Contoh:

- Kalimat asli: "Dia pergi ke bank untuk mengambil uang."
- Setelah menghapus stopword: "Dia pergi bank ambil uang."
- Dalam kalimat yang diprocessing, kata "bank" menjadi lebih ambigu (bisa merujuk ke lembaga keuangan atau tepi sungai).

**Solusi:** Preprocessing yang lebih canggih perlu mempertimbangkan konteks kalimat untuk mengatasi ambiguitas. Teknik seperti *Word Sense Disambiguation* (WSD) berusaha untuk menentukan arti yang tepat dari sebuah kata dalam konteks tertentu.

#### 4.2 Variasi Bahasa: Kekayaan yang Menantang

Bahasa memiliki banyak variasi dalam kata-kata, tata bahasa, dan gaya penulisan. Preprocessing harus mampu menangani variasi ini.

**Penjelasan Lebih Detail:** Bahasa manusia sangat kaya dan beragam. Ada banyak cara berbeda untuk mengekspresikan ide yang sama.

- **Jenis Variasi Bahasa:**
  - **Sinonim:** Kata-kata yang berbeda dengan arti yang sama (misalnya, "besar" dan "agung").
  - **Parafrasa:** Kalimat yang berbeda dengan arti yang sama (misalnya, "Saya membeli mobil" dan "Mobil itu saya beli").
  - **Gaya Penulisan:** Formal vs. informal, naratif vs. deskriptif, dll.
  - **Dialek Regional:** Perbedaan dalam kosakata dan tata bahasa antar wilayah.
- **Tantangan Preprocessing:** Preprocessing harus mampu mengenali bahwa variasi-variasi ini pada dasarnya menyampaikan makna yang sama. Jika tidak, algoritma analisis teks mungkin memperlakukan variasi-variasi ini sebagai hal yang berbeda, yang dapat mengurangi akurasi.

- Solusi:** Teknik seperti lemmatisasi (mengubah kata ke bentuk dasarnya) dan normalisasi teks (mengubah teks ke bentuk standar) dapat membantu mengatasi variasi bahasa.

#### 4.3 Konteks: Makna yang Berubah

- "Makna kata dapat berubah tergantung pada konteksnya. Preprocessing yang ideal harus mempertimbangkan konteks untuk menghindari kesalahan interpretasi."
- Penjelasan Lebih Detail:** Konteks adalah informasi di sekitar kata atau kalimat yang membantu kita memahami maknanya.
- Contoh:**
  - Dalam kalimat "Dia meletakkan buku di atas meja," kata "atas" merujuk ke posisi fisik.
  - Dalam kalimat "Dia adalah atasan saya di kantor," kata "atasan" merujuk ke jabatan.
- Tantangan Preprocessing:** Preprocessing sering kali dilakukan pada tingkat kata per kata, tanpa mempertimbangkan konteks yang lebih luas. Ini dapat menyebabkan kesalahan interpretasi jika makna kata berubah tergantung pada konteks.
- Solusi:** Teknik NLP yang lebih canggih, seperti pemodelan bahasa, berusaha untuk memahami hubungan antar kata dalam sebuah kalimat atau dokumen.

#### 4.4 Bahasa Non-Standar: Tantangan dari Dunia Online

- "Teks dari media sosial atau percakapan online sering kali mengandung bahasa non-standar, seperti slang, singkatan, dan kesalahan ejaan. Preprocessing harus mampu menangani hal ini."
- Penjelasan Lebih Detail:** Teks yang dihasilkan pengguna (User-Generated Content atau UGC), seperti posting media sosial, komentar online, dan pesan teks, sering kali berbeda dari bahasa standar.
  - Karakteristik Bahasa Non-Standar:**
    - Slang:** Kata-kata informal yang digunakan dalam kelompok tertentu (misalnya, "gaul," "baper").
    - Singkatan:** Bentuk pendek dari kata atau frasa (misalnya, "yg" untuk "yang," "km" untuk "kamu").
    - Kesalahan Ejaan:** Kata-kata yang salah ketik atau salah eja.
    - Emotikon dan Emoji:** Simbol-simbol yang digunakan untuk mengekspresikan emosi.
  - Tantangan Preprocessing:** Preprocessing harus mampu mengenali dan menangani variasi-variasi ini, yang mungkin tidak ada dalam kamus atau aturan tata bahasa standar.
  - Solusi:** Preprocessing untuk teks UGC mungkin memerlukan langkah-langkah tambahan, seperti:
    - Normalisasi teks (mengubah kata-kata non-standar ke bentuk standarnya).
    - Koreksi ejaan.
    - Penanganan emoticon dan emoji.

#### 4.5 Efisiensi: Waktu adalah Sumber Daya

- "Preprocessing teks dapat menjadi proses yang memakan waktu, terutama untuk data yang besar. Efisiensi komputasi perlu dipertimbangkan."

- Penjelasan Lebih Detail:** Preprocessing teks bisa menjadi proses yang intensif secara komputasi, terutama jika kita memiliki banyak data teks.

- O Faktor yang Mempengaruhi Efisiensi:**

- Ukuran data teks.
- Kompleksitas langkah-langkah preprocessing.
- Efisiensi algoritma dan implementasi.

- O Tantangan Preprocessing:** Kita perlu menyeimbangkan antara kualitas preprocessing dan efisiensi waktu. Preprocessing yang terlalu kompleks mungkin menghasilkan hasil yang lebih baik, tetapi membutuhkan waktu yang lebih lama.

- O Solusi:**

- Optimasi kode dan algoritma.
- Penggunaan teknik paralel processing (memproses beberapa bagian data secara bersamaan).
- Pemilihan langkah-langkah preprocessing yang relevan dengan tugas analisis (tidak melakukan preprocessing yang tidak perlu).

#### Kesimpulan:

- Preprocessing teks adalah proses yang kompleks dan menantang.
- Kita perlu memahami tantangan-tantangan ini untuk mengembangkan strategi preprocessing yang efektif.
- Tidak ada "satuan ukuran untuk semua" dalam preprocessing teks. Langkah-langkah yang tepat tergantung pada jenis data teks dan tujuan analisis.

#### 5. Kesimpulan

- Preprocessing text adalah tahapan penting dalam text mining dan NLP.
- Langkah-langkah umum dalam preprocessing text meliputi cleaning, case folding, tokenizing, stopword removal, dan stemming/lemmatization.
- Python menyediakan berbagai library yang memudahkan implementasi preprocessing text.
- Preprocessing text yang baik dapat meningkatkan kualitas data teks dan akurasi analisis.
- Tantangan dalam preprocessing text perlu diatasi dengan hati-hati untuk mendapatkan hasil yang optimal.

## Chapter 4: Pembobotan Kata dengan TF-IDF

### 1. Pendahuluan

- Pembobotan kata adalah teknik penting dalam Natural Language Processing (NLP) dan Information Retrieval (IR) untuk memberikan nilai numerik pada kata-kata dalam dokumen.
- TF-IDF adalah salah satu metode pembobotan kata yang paling umum dan efektif.
- PowerPoint ini memberikan pengantar tentang konsep TF-IDF, formula perhitungannya, dan implementasinya dalam Python.
- Materi ini akan memperluas konsep-konsep tersebut dan menambahkan detail penting untuk pemahaman yang lebih komprehensif dan profesional.

**2. Apa Itu TF-IDF?** TF-IDF adalah singkatan dari Term Frequency-Inverse Document Frequency. Ini adalah metode pembobotan kata dalam dokumen yang digunakan dalam NLP dan IR. **Tujuan:** Teknik ini bertujuan untuk menilai seberapa penting sebuah kata dalam konteks dokumen tertentu, relatif terhadap keseluruhan kumpulan dokumen (corpus). **Konsep Dasar:** TF-IDF memberikan bobot tinggi kepada kata-kata yang sering muncul dalam dokumen tertentu tetapi jarang muncul di seluruh corpus. Kata-kata yang umum di seluruh corpus cenderung memiliki bobot rendah karena dianggap kurang informatif dalam membedakan dokumen.

**3. Formula TF (Term Frequency)** TF mengukur seberapa sering sebuah kata muncul dalam satu dokumen.

- Formula:**

$$TF(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)}$$

- Keterangan:**

- $t$  = kata yang sedang dihitung frekuensinya
- $d$  = dokumen tertentu
- $f(t, d)$  = jumlah kemunculan kata  $t$  dalam dokumen  $d$
- $\sum_{t' \in d} f(t', d)$  = total jumlah kata dalam dokumen  $d$

**Contoh:** Jika kata "apel" muncul 5 kali dalam dokumen yang memiliki 100 kata, maka  $TF(\text{"apel"}, \text{dokumen}) = 5/100 = 0.05$

### 4. Formula IDF (Inverse Document Frequency)

**Definisi:** IDF mengukur seberapa jarang atau unik sebuah kata dalam seluruh koleksi dokumen (corpus). **Konsep:**

- Kata yang muncul di banyak dokumen memiliki nilai IDF yang rendah.
- Kata yang jarang muncul memiliki nilai IDF yang tinggi.

- Formula:**

$$IDF(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

- Keterangan:**

- $D$  = jumlah total dokumen dalam kumpulan (corpus)
- $|\{d \in D : t \in d\}|$  = jumlah dokumen yang mengandung kata  $t$
- Penambahan 1 dalam penyebut untuk menghindari pembagian dengan nol jika kata tidak ada di dokumen mana pun.

**Contoh:** Jika kata "apel" muncul dalam 10 dari 100 dokumen, maka  $IDF(\text{"apel"}, \text{Corpus}) = \log(100 / (1 + 10)) \approx 0.95$

### 5. Formula TF-IDF

- Formula:**

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

**Penjelasan:** Nilai TF-IDF untuk sebuah kata dalam dokumen adalah hasil perkalian nilai TF kata tersebut dalam dokumen itu dengan nilai IDF kata tersebut dalam seluruh corpus.

### 6. Jenis-Jenis TF-IDF: Memahami N-gram

PowerPoint menjelaskan bahwa TF-IDF dapat diterapkan dengan mempertimbangkan n-gram.

**Apa itu N-gram?** N-gram adalah urutan  $n$  kata yang berdekatan dalam sebuah teks. N-gram digunakan untuk menangkap konteks dan hubungan antar kata dalam sebuah kalimat. Contoh: \* Dalam kalimat "Saya suka makan nasi goreng", kita bisa mengekstrak unigram, bigram, trigram, dst.

#### Jenis-Jenis TF-IDF Berdasarkan N-gram:

- **TF-IDF Unigram:** Hanya mempertimbangkan kata tunggal sebagai fitur. Setiap kata dalam dokumen dihitung TF-IDF-nya secara individual.
  - Contoh: \* Dalam kalimat "Saya suka makan nasi goreng", unigram adalah "saya", "suka", "makan", "nasi", "goreng".
  - **ngram\_range = (1, 1)** dalam Scikit-learn.
- **TF-IDF Bigram:** Hanya mempertimbangkan pasangan kata yang berurutan sebagai fitur. Berguna untuk menangkap frasa sederhana.
  - Contoh: \* Dalam kalimat yang sama, bigram adalah "saya suka", "suka makan", "makan nasi", "nasi goreng".
  - **ngram\_range = (2, 2)** dalam Scikit-learn.
- **TF-IDF Trigram:** Hanya mempertimbangkan urutan tiga kata yang berurutan sebagai fitur. Menangkap konteks yang lebih panjang, tetapi bisa menghasilkan lebih banyak fitur.
  - Contoh: \* Trigram adalah "saya suka makan", "suka makan nasi", "makan nasi goreng".
  - **ngram\_range = (3, 3)** dalam Scikit-learn.
- **TF-IDF Unigram dan Bigram:** Mempertimbangkan baik kata tunggal maupun pasangan kata yang berurutan sebagai fitur.
  - Mengkombinasikan informasi dari kata individu dan frasa sederhana.
    - Contoh: \* "saya", "suka", "makan", "nasi", "goreng", "saya suka", "suka makan", "makan nasi", "nasi goreng".
  - **ngram\_range = (1, 2)** dalam Scikit-learn.
- **Mengapa N-gram Penting?** N-gram membantu mempertahankan sebagian informasi urutan kata, yang hilang ketika hanya menggunakan unigram. Ini penting karena urutan kata sering kali mempengaruhi makna kalimat. Contoh: \* "bukan main" memiliki arti yang berbeda dari "main bukan". Dengan menggunakan n-gram, model dapat membedakan antara frasa-frasa ini.
- **Pertimbangan dalam Memilih N-gram:**
  - **Komputasi:** Semakin tinggi nilai  $n$ , semakin banyak fitur yang dihasilkan, yang meningkatkan biaya komputasi.
  - **Sparsity:** N-gram yang lebih tinggi (trigram, 4-gram, dst.) cenderung lebih jarang muncul, menghasilkan matriks yang sangat sparse (banyak nilai nol).
  - **Konteks:** Pilihan  $n$  tergantung pada seberapa penting konteks urutan kata untuk tugas yang sedang dikerjakan.

## 7. Library Python yang Digunakan: Scikit-learn

PowerPoint menyatakan bahwa perhitungan TF-IDF di Python umumnya menggunakan library Scikit-learn (sklearn).

- **Scikit-learn (sklearn):** Adalah library machine learning yang populer dan *powerful* di Python. Menyediakan berbagai algoritma untuk klasifikasi, regresi, pengelompokan, dan *dimensionality reduction*, serta alat untuk preprocessing data.

- **TfidfVectorizer:** Scikit-learn menyediakan kelas TfidfVectorizer dalam modul sklearn.feature\_extraction.text untuk menghitung TF-IDF. TfidfVectorizer mengubah koleksi dokumen teks mentah menjadi matriks representasi TF-IDF.
- **Fitur Utama TfidfVectorizer:**
  - **Automatic Calculation:** Secara otomatis menghitung TF dan IDF.
  - **N-gram Support:** Memungkinkan penggunaan n-gram melalui parameter ngram\_range.
  - **Stop Word Removal:** Dapat menghilangkan stop word bawaan atau yang disediakan pengguna.
  - **Vocabulary Control:** Memungkinkan untuk membatasi kosakata berdasarkan frekuensi dokumen.
  - **Normalization:** Melakukan normalisasi vektor TF-IDF.
- **Keuntungan Menggunakan Scikit-learn untuk TF-IDF:**
  - **Kemudahan Penggunaan:** TfidfVectorizer menyediakan antarmuka yang mudah digunakan untuk menghitung TF-IDF.
  - **Efisiensi:** Implementasi yang efisien untuk menangani koleksi dokumen yang besar.
  - **Integrasi:** Mudah diintegrasikan dengan algoritma machine learning lainnya di Scikit-learn.

## 8. Implementasi Python: Kode TF-IDF dalam Praktik

Bagian ini menunjukkan bagaimana menghitung TF-IDF menggunakan Python dan library Scikit-learn. Berikut adalah penjelasan yang lebih detail dan lengkap:

### 8.1 Import Library

- ```
import pandas as pd
import numpy as np
```
- **pandas:** Digunakan untuk membaca dan memanipulasi data dalam format DataFrame, yang sangat efisien untuk data tabular.
  - **numpy:** Digunakan untuk operasi numerik, terutama untuk menangani array dan matriks secara efisien.

### 8.2 Membaca Data

```
dm = pd.read_csv("tomlembong50_clean.csv", usecols=["tweet_clean", "Label"])
```

```
dm.columns = ["tweet_clean", "Label"]
```

```
dm.head(10)
```

Penjelasan:

- ```
pd.read_csv("tomlembong50_clean.csv", usecols=["tweet_clean", "Label"]):
```

 Membaca data dari file CSV bernama "tomlembong50\_clean.csv". Hanya kolom "tweet\_clean" (yang berisi teks yang sudah dipreprocessing) dan "Label" (yang mungkin menunjukkan kategori sentimen atau kelas lainnya) yang dibaca.
- ```
dm.columns = ["tweet_clean", "Label"]:
```

 Memberi nama ulang kolom menjadi "tweet\_clean" dan "Label" untuk kemudahan penggunaan.
- ```
dm.head(10):
```

 Menampilkan 10 baris pertama dari DataFrame dm untuk melihat contoh data.

### 8.3 Menghitung Jumlah Label

`dm['Label'].value_counts()`

**Penjelasan:** `dm['Label'].value_counts()`: Menghitung berapa kali setiap nilai unik muncul dalam kolom "Label". Ini berguna untuk memahami distribusi kelas dalam data (misalnya, berapa banyak tweet positif dan negatif).

## 8.4 Mengonversi String List ke List

```
import ast
def convert_text_list(texts):
    texts = ast.literal_eval(texts)
    return [text for text in texts]
```

```
dm["tekslist"] = dm["tweet_clean"].apply(convert_text_list)
print(dm["tekslist"][23])
print("\ntype: ", type(dm["tekslist"][23]))
```

Penjelasan:

- **import ast:** Mengimpor modul ast (Abstract Syntax Tree). Modul ini membantu dalam mengubah string yang terlihat seperti struktur data Python (seperti list) menjadi struktur data Python yang sebenarnya.
- **convert\_text\_list(texts):** Fungsi ini menerima string yang diformat seperti list Python (misalnya, "[kata1', 'kata2']") dan mengubahnya menjadi objek list Python yang sebenarnya (misalnya, ['kata1', 'kata2']).
  - **ast.literal\_eval(texts):** Dengan aman mengevaluasi string sebagai ekspresi Python literal. Ini penting untuk menghindari risiko eksekusi kode berbahaya.
  - **return [text for text in texts]:** Membuat list baru dari elemen-elemen yang dievaluasi.
- ```
dm["tekslist"] = dm["tweet_clean"].apply(convert_text_list):
```

 Membuat kolom baru bernama "tekslist" di DataFrame dm. Setiap sel di kolom "tweet\_clean" (yang diasumsikan berisi string list) diterapkan fungsi convert\_text\_list() untuk mengubah string list menjadi list Python.
- ```
print(dm["tekslist"][23]):
```

 Mencetak elemen ke-23 dari kolom "tekslist" (indeks dimulai dari 0) untuk memeriksa hasilnya.
- ```
print("\ntype: ", type(dm["tekslist"][23])):
```

 Mencetak tipe data dari elemen ke-23 untuk memastikan bahwa itu adalah list.

## 8.5 Perhitungan TF

```
def calc_TF(document):
    #perhitungan jumlah kata
    TF_dict = {}
    for term in document:
        if term in TF_dict:
            TF_dict[term] += 1
        else:
```

```

TF_dict[term] = 1

#perhitungan tf

for term in TF_dict:
    TF_dict[term] = TF_dict[term] / len(document)

return TF_dict

dm["TF_dict"] = dm['tekslist'].apply(calc_TF)

dm["TF_dict"].head()

index = 23

print("%20s % "term", "\t", "%\n")

for key in dm[ "TF_dict" ][index]:
    print("%20s % key, "\t", dm["TF_dict" ][index][key])

```

Penjelasan:

- calc\_TF(document)**: Fungsi ini menghitung Term Frequency (TF) untuk setiap dokumen.
  - TF\_dict = {}**: Inisialisasi dictionary untuk menyimpan hitungan kata.
  - Loop pertama (**for term in document**): Menghitung jumlah kemunculan setiap kata dalam dokumen.
  - Loop kedua (**for term in TF\_dict**): Menghitung TF dengan membagi jumlah kemunculan kata dengan total jumlah kata dalam dokumen.
  - return TF\_dict**: Mengembalikan dictionary yang berisi nilai TF untuk setiap kata dalam dokumen.
- dm["TF\_dict"] = dm['tekslist'].apply(calc\_TF)**: Membuat kolom baru "TF\_dict" di DataFrame dm dengan menerapkan fungsi calc\_TF ke setiap list kata di kolom "tekslist".
- dm["TF\_dict"].head()**: Menampilkan beberapa baris pertama dari kolom "TF\_dict" untuk melihat hasilnya.
- Bagian kode setelahnya mencetak nilai TF untuk dokumen pada indeks 23. Ini berguna untuk memeriksa hasil perhitungan TF untuk dokumen tertentu.

#### 8.6 Perhitungan IDF

```

def calc_DF(tfDict):
    count_DF = {}

    for document in tfDict:
        for term in document:
            if term in count_DF:
                count_DF[term] += 1
            else:
                count_DF[term] = 1

    return count_DF

```

```

DF = calc_DF(dm["TF_dict"])

#menghitung idf

n_document = len(dm)

def calc_IDF(n_document, DF):
    IDF_Dict = {}

    for term in DF:
        IDF_Dict[term] = np.log_(n_document / (DF[term] + 1))

    return IDF_Dict

#penyimpanan kamus idf

IDF = calc_IDF(n_document, DF)

Penjelasan:

● calc_DF(tfDict): Fungsi ini menghitung Document Frequency (DF) untuk setiap kata di seluruh corpus.
            

- count_DF = {}: Inisialisasi dictionary untuk menyimpan jumlah dokumen yang mengandung setiap kata.
- Loop bertingkat: Menghitung berapa banyak dokumen yang mengandung setiap kata.
- return count_DF: Mengembalikan dictionary yang berisi nilai DF untuk setiap kata.


● DF = calc_DF(dm["TF_dict"]): Menghitung DF dengan menerapkan fungsi calc_DF ke kolom "TF_dict".
● calc_IDF(n_document, DF): Fungsi ini menghitung Inverse Document Frequency (IDF).
            

- n_document = len(dm): Menghitung total jumlah dokumen.
- Loop: Menghitung IDF untuk setiap kata menggunakan formula IDF.
- IDF_Dict[term] = np.log_(n_document / (DF[term] + 1)): Formula IDF. Penambahan 1 pada penyebut untuk menghindari pembagian oleh nol.
- return IDF_Dict: Mengembalikan dictionary IDF.


● IDF = calc_IDF(n_document, DF): Menghitung IDF menggunakan fungsi calc_IDF.

8.7 Perhitungan TF-IDF

def calc_TF_IDF(TF):
    TF_IDF_Dict = {}

    for key in TF:
        TF_IDF_Dict[key] = TF[key] * IDF[key]

    return TF_IDF_Dict

#penyimpanan variabel TF-IDF

dm["TF-IDF_dict"] = dm["TF_dict"].apply(calc_TF_IDF)

```

Penjelasan:

- calc\_TF\_IDF(TF)**: Fungsi ini menghitung TF-IDF untuk setiap kata dalam setiap dokumen.
  - Loop**: Mengalikan nilai TF dan IDF untuk setiap kata.
  - return TF\_IDF\_Dict**: Mengembalikan dictionary TF-IDF.
- dm["TF-IDF\_dict"] = dm["TF\_dict"].apply(calc\_TF\_IDF)**: Membuat kolom baru "TF-IDF\_dict" dengan menerapkan fungsi calc\_TF\_IDF.

#### 8.8 Hasil Perhitungan TF-IDF

```

# memunculkan nilai TF-IDF

index = 23

print( '%20s % "term", "\t", '%10s % "TF", "\t", "%20s % "TF-IDF\n" )

for key in dm["TF-IDF_dict"] [index]:
    print('%20s % key, "\t", dm["TF_dict"][index][key] , "\t", dm["TF-IDF_dict"][index][key])
```

Penjelasan: Kode ini mencetak nilai TF dan TF-IDF untuk dokumen pada indeks 23. Ini memungkinkan pemeriksaan nilai TF-IDF yang telah dihitung.

#### 8.9 Matriks TF-IDF

```

#matrik tf-idf

# pengurutan descending berdasarkan nilai DF

sorted_DF = sorted(DF.items(), key=lambda kv: kv[1], reverse=True)[:30]

# pembuatan list kata dari pengurutan 'sorted_DF'

unique_term = [item[0] for item in sorted_DF]

def calc_TF_IDF_Vec(_TF_IDF_Dict):
    TF_IDF_vector = [0.0] * len(unique_term)

    # For each unique word, if it is in the review, store its TF-IDF value.

    for i, term in enumerate(unique_term):
        if term in _TF_IDF_Dict:
            TF_IDF_vector[i] = _TF_IDF_Dict[term]

    return TF_IDF_vector
```

```

dm["TF_IDF_Vec"] = dm["TF-IDF_dict"].apply(calc_TF_IDF_Vec)

print("tampil baris pertama matrix TF_IDF_Vec Series\n")

print(dm["TF_IDF_Vec"])[0]
```

```
print("\nukuran matrix : ", len(dm["TF_IDF_Vec"])[0]))
```

Penjelasan:

- sorted\_DF = sorted(DF.items(), key=lambda kv: kv[1], reverse=True) [:30]:** Mengurutkan kata-kata berdasarkan Document Frequency (DF) dalam urutan menurun dan mengambil 30 kata teratas. Ini mungkin dilakukan untuk memilih kata-kata yang paling relevan.
- unique\_term = [item[0] for item in sorted\_DF]:** Membuat list unique\_term yang berisi 30 kata teratas. Ini akan menjadi kolom dalam matriks TF-IDF.
- calc\_TF\_IDF\_Vec\_(TF\_IDF\_Dict):** Fungsi ini mengubah dictionary TF-IDF menjadi vektor.
  - TF\_IDF\_vector = [0.0] \* len(unique\_term):** Membuat vektor dengan panjang sama dengan jumlah kata unik, diinisialisasi dengan 0.
  - Loop: Mengisi vektor dengan nilai TF-IDF yang sesuai untuk setiap kata unik.
  - return TF\_IDF\_vector:** Mengembalikan vektor TF-IDF.
- dm["TF\_IDF\_Vec"] = dm["TF-IDF\_dict"].apply(calc\_TF\_IDF\_Vec):** Membuat kolom "TF\_IDF\_Vec" yang berisi vektor TF-IDF untuk setiap dokumen.
- Kode print menampilkan baris pertama dari matriks TF-IDF dan ukurannya.

## 8.10 Menampilkan ke dalam List

```
#menampilkan top 30 term tf-idf
# konversi ke dalam List
TF_IDF_Vec_List = np.array(dm["TF_IDF_Vec"].to_list())
sums = TF_IDF_Vec_List.sum(axis=0)
data = []
for col, term in enumerate(unique_term):
    data.append((term, sums[col]))
ranking = pd.DataFrame(data, columns=['term', 'rank'])
ranking.sort_values('rank', ascending=False)
```

Penjelasan:

- TF\_IDF\_Vec\_List = np.array(dm["TF\_IDF\_Vec"].to\_list()):** Mengubah kolom "TF\_IDF\_Vec" (yang berisi vektor TF-IDF) menjadi NumPy array.
- sums = TF\_IDF\_Vec\_List.sum(axis=0):** Menghitung jumlah dari setiap kolom (yaitu, jumlah dari setiap kata TF-IDF di semua dokumen).
- Loop: Membuat list data yang berisi pasangan (kata, jumlah TF-IDF).
- ranking = pd.DataFrame(data, columns=['term', 'rank']):** Membuat DataFrame ranking dari list data.
- ranking.sort\_values('rank', ascending=False):** Mengurutkan DataFrame berdasarkan jumlah TF-IDF dalam urutan menurun. Ini memberikan peringkat kata-kata berdasarkan kepentingan mereka di seluruh corpus.

## 9. Kesimpulan

- TF-IDF adalah metode yang efektif untuk pembobotan kata dalam teks.

- TF-IDF membantu dalam analisis teks dengan memberikan bobot yang sesuai pada kata-kata penting dalam dokumen.
- Python dan Scikit-learn menyediakan alat yang mudah digunakan untuk menghitung TF-IDF.
- Hasil TF-IDF dapat digunakan sebagai fitur dalam berbagai tugas NLP, seperti klasifikasi teks, pengelompokan teks, dan Information Retrieval.

## Langkah-Langkah Pembobotan Kata dengan TF-IDF: Panduan Lengkap

Pembobotan kata adalah teknik penting dalam Natural Language Processing (NLP) dan Information Retrieval (IR) untuk memberikan nilai numerik pada kata-kata dalam dokumen. TF-IDF (Term Frequency-Inverse Document Frequency) adalah salah satu metode pembobotan kata yang paling umum dan efektif.

### 1. Term Frequency (TF)

- Definisi:** TF mengukur seberapa sering sebuah kata muncul dalam satu dokumen.
- Proses Perhitungan:**
  - Hitung jumlah kemunculan setiap kata dalam dokumen.
  - Bagi jumlah kemunculan setiap kata dengan total jumlah kata dalam dokumen.
- Tujuan:** Untuk mengetahui seberapa penting sebuah kata dalam konteks dokumen tersebut.

### 2. Document Frequency (DF)

- Definisi:** DF mengukur dalam berapa banyak dokumen sebuah kata muncul.
- Proses Perhitungan:**
  - Hitung berapa banyak dokumen yang mengandung kata tertentu.
- Tujuan:** Untuk mengidentifikasi kata-kata yang umum di seluruh dokumen.

### 3. Inverse Document Frequency (IDF)

- Definisi:** IDF mengukur seberapa jarang atau unik sebuah kata dalam keseluruhan kumpulan dokumen (corpus).
- Proses Perhitungan:**
  - Hitung total jumlah dokumen dalam corpus.
  - Bagi total jumlah dokumen dengan jumlah dokumen yang mengandung kata tertentu (DF).
  - Hitung logaritma dari hasil pembagian tersebut.
- Tujuan:** Untuk memberikan bobot lebih tinggi pada kata-kata yang jarang muncul di seluruh corpus, karena dianggap lebih informatif.

### 4. TF-IDF

- Definisi:** TF-IDF adalah hasil perkalian antara nilai TF dan nilai IDF untuk sebuah kata dalam dokumen.
- Proses Perhitungan:**
  - Kalikan nilai TF dari sebuah kata dalam dokumen dengan nilai IDF dari kata tersebut.
- Tujuan:** Untuk menilai seberapa penting sebuah kata dalam konteks dokumen tertentu, relatif terhadap keseluruhan corpus.

### 5. Implementasi dalam Python (Scikit-learn)

- Scikit-learn menyediakan kelas TfidfVectorizer untuk menghitung TF-IDF secara efisien.

- Langkah-langkah umum:**
  - Import library TfidfVectorizer dari sklearn.feature\_extraction.text.
  - Buat objek TfidfVectorizer.
  - Gunakan metode fit\_transform() untuk menghitung matriks TF-IDF dari kumpulan dokumen.

## 6. Interpretasi Hasil TF-IDF

- Nilai TF-IDF yang tinggi menunjukkan bahwa kata tersebut penting dalam dokumen tertentu dan jarang muncul di dokumen lain dalam corpus.
- Nilai TF-IDF yang rendah menunjukkan bahwa kata tersebut umum di seluruh corpus dan kurang informatif dalam membedakan dokumen.

## Ringkasan Langkah-Langkah Pembobotan Kata dengan TF-IDF

- Hitung TF:** Hitung frekuensi kemunculan kata dalam setiap dokumen.
- Hitung DF:** Hitung jumlah dokumen yang mengandung setiap kata.
- Hitung IDF:** Hitung inverse document frequency untuk setiap kata.
- Hitung TF-IDF:** Kalikan nilai TF dan IDF untuk setiap kata dalam setiap dokumen.
- Representasi Matriks:** Ubah hasil TF-IDF menjadi matriks, di mana baris mewakili dokumen, kolom mewakili kata, dan nilai sel mewakili nilai TF-IDF.

Langkah Langkah Crawling data

Langkah Langkah pembobotan kata dengan tf idf

Materi dari tugas

Membuatkan Michael contekan