Nama : **Rizka Nugraha**

NIM : **A11.2022.14119**

Kelompok : **A11.4703**

Mata Kuliah : **Sistem Temu Kembali Informatika**

1. Definisi STKI, Peran Indeks, dan Ranking

Sistem Temu Kembali Informasi (STKI), atau Information Retrieval (IR), adalah bidang ilmu yang berfokus pada pencarian dan penemuan informasi dari sekumpulan data. Berbeda dengan sistem database, data dalam STKI umumnya bersifat tidak terstruktur (seperti dokumen teks, halaman web, atau gambar). Tujuan utama STKI adalah untuk memenuhi kebutuhan informasi (information need) pengguna, yang seringkali diekspresikan melalui kueri yang ambigu (misalnya, kata kunci).

Perbedaan mendasar antara STKI dan Database Retrieval terletak pada tiga hal:

- Struktur Data: Database bekerja dengan data terstruktur yang tersimpan dalam tabel dengan skema yang kaku (baris dan kolom). STKI bekerja dengan data tidak terstruktur seperti teks bebas.
- Kueri: Kueri database (seperti SQL) bersifat tegas, formal, dan spesifik (misal: SELECT * FROM users WHERE id = 10). Kueri STKI bersifat ambigu (misal: "berita varian delta").
- Hasil: Hasil database bersifat cocok (match) atau tidak. Hasil STKI bersifat probabilistik; sistem mengembalikan dokumen yang kemungkinan relevan, sehingga membutuhkan perankingan.

Dalam arsitektur STKI, dua komponen krusial adalah Indeks dan Ranking.

- Peran Indeks: Indeks adalah struktur data inti yang memetakan konten ke lokasinya dalam dokumen. Tujuannya adalah untuk mempercepat proses pencarian secara drastis. Tanpa indeks, sistem harus memindai setiap dokumen satu per satu (sequential scan) untuk setiap kueri. Dalam proyek ini, Soal 03 mengimplementasikan Inverted Index (Indeks Terbalik). Ini adalah struktur data dictionary yang memetakan setiap term (kata) unik ke sebuah postings list (daftar dokumen yang mengandung kata tersebut).

- Peran Ranking: Ranking adalah proses mengurutkan dokumen yang telah diambil (retrieved) berdasarkan skor relevansinya terhadap kueri. Ini adalah pembeda utama dari model boolean murni. Dalam proyek ini, Soal 04 mengimplementasikan perankingan menggunakan Vector Space Model (VSM). Dokumen dan kueri direpresentasikan sebagai vektor TF-IDF, dan skor relevansinya dihitung menggunakan Cosine Similarity. Dokumen dengan skor cosine tertinggi dianggap paling relevan.

2. Garis Besar Arsitektur Search Engine Klasik

Arsitektur search engine klasik, seperti yang diimplementasikan dalam proyek ini, mengikuti alur yang logis untuk mengubah kueri pengguna menjadi daftar hasil yang terurut.

Alur lengkapnya adalah sebagai berikut:

1. Query: Pengguna memasukkan kueri (misal: "kasus varian delta") melalui antarmuka, yang dalam proyek ini adalah aplikasi Streamlit (app/main.py).
2. Preprocess: Kueri tersebut harus melalui proses pembersihan yang sama dengan dokumen. Ini ditangani oleh modul src/preprocess.py , yang melakukan case-folding, tokenisasi, stopword removal, dan stemming.
3. Retrieve: Sistem menggunakan kueri yang sudah bersih untuk mengambil dokumen dari indeks.
    - Pada Model Boolean, ini berarti mengambil postings list dari Inverted Index.
    - Pada VSM, ini berarti merepresentasikan kueri sebagai vektor TF-IDF.
4. Rank: Dokumen yang diambil kemudian diberi skor dan diurutkan.
    - Pada Model Boolean, tidak ada perankingan; hasilnya adalah himpunan dokumen yang cocok.
    - Pada VSM, skor dihitung menggunakan Cosine Similarity antara vektor kueri dan setiap vektor dokumen.
5. Present: Hasil yang sudah terurut (biasanya Top-K, misal k=3) ditampilkan kepada pengguna, lengkap dengan snippet (potongan teks), melalui antarmuka Streamlit.

3. Sketsa Arsitektur Retrieval: Boolean vs. VSM

Proyek ini mengimplementasikan dua model retrieval klasik dengan arsitektur yang fundamentalnya berbeda.

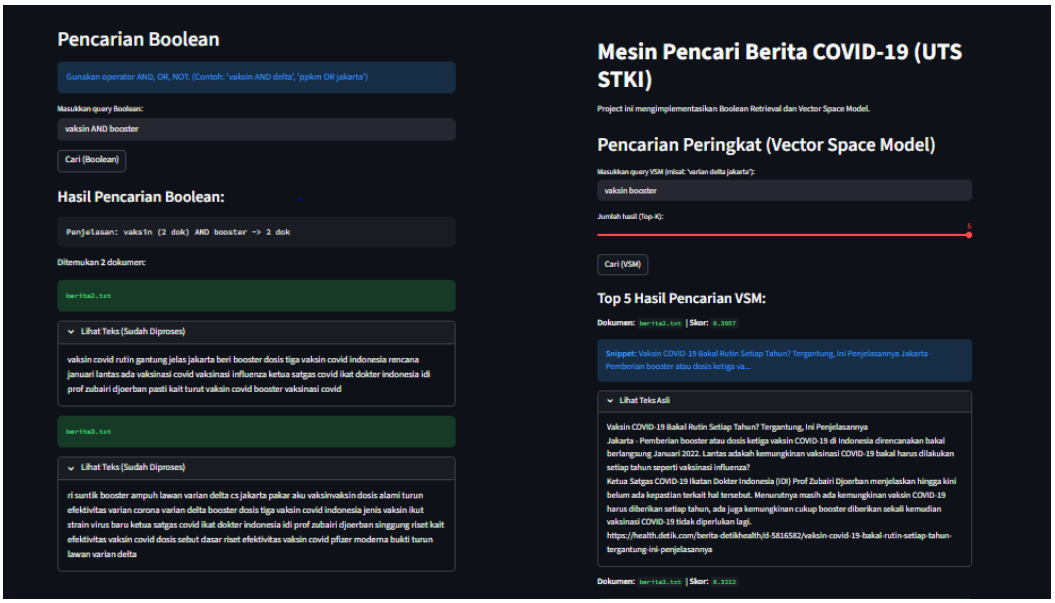
Fitur	Model Boolean (Implementasi Soal 03)	📐 Vector Space Model (VSM)
Representasi Data	**Inverted Index** (Sebuah dict Python yang memetakan *term* ke set ID dokumen).	**TF-IDF Matrix** (Sebuah *sparse matrix* di mana baris adalah dokumen dan kolom adalah *term*).
Logika Retrieval	**Operasi Himpunan (Set)**. Menggunakan AND (irisan), OR (gabungan), dan NOT (komplemen) untuk menemukan dokumen yang cocok secara eksak.	**Aljabar Vektor**. Menghitung jarak sudut (Cosine Similarity) antara vektor kueri dan semua vektor dokumen.
Hasil Akhir	Sebuah himpunan dokumen yang **cocok** (match). Hasilnya **tidak terurut** (unranked).	Sebuah daftar dokumen yang **terurut (ranked)** berdasarkan skor relevansi.

4. **Demo Singkat (Screenshot Arsitektur IR)**

   Untuk mendemonstrasikan kedua arsitektur ini secara visual, jalankan aplikasi Streamlit (app/main.py) dan ambil screenshot yang menunjukkan kedua hasil pencarian.

1. **Lakukan Pencarian VSM:** Masukkan kueri seperti "varian delta jakarta" di bagian VSM. Hasilnya akan muncul **terurut** berdasarkan skor *cosine*.
2. **Lakukan Pencarian Boolean:** Masukkan kueri seperti "varian AND delta" di bagian Boolean. Hasilnya akan muncul sebagai daftar yang **tidak terurut**.

   Screenshot ini secara efektif mendemonstrasikan perbedaan output fundamental antara arsitektur VSM (ranked) dan Boolean (unranked).



5. **Peta Kaitan Materi Proyek ke RPS**

   Proyek UTS ini dirancang untuk secara langsung mengimplementasikan konsep-konsep yang ada di RPS, seperti yang terlihat pada pemetaan Soal 02 hingga Soal 05 ke Sub-CPMK.

- **Soal 02 - Document Preprocessing**
  - **Kaitan RPS:** Soal ini secara langsung mengimplementasikan **Sub-CPMK10.1.2** ("Mampu menjelaskan Document Preprocessing dan tahapannya"). Modul src/preprocess.py yang kami buat menerapkan seluruh tahapan yang diminta: tokenisasi, case-folding, stopword removal, dan stemming pada korpus dokumen.

- **Soal 03 - Boolean Retrieval Model**
    - **Kaitan RPS:** Soal ini adalah implementasi praktis dari **Sub-CPMK10.1.3** ("Mampu menjelaskan Pemodelan... Boolean Retrieval Model"). Kami membangun *inverted index* dan parser kueri untuk mendukung operasi AND, OR, dan NOT.
- **Soal 04 - Vector Space Model & Ranking**
    - **Kaitan RPS:** Soal ini juga mengimplementasikan **Sub-CPMK10.1.3** ("Mampu menjelaskan Pemodelan... Vector Space Model"). Kami berhasil merepresentasikan dokumen sebagai vektor menggunakan TF-IDF dan mengimplementasikan perankingan menggunakan *cosine similarity*.
- **Soal 05 - Term Weighting, Search Engine, dan Evaluasi**
    - **Kaitan RPS:** Ini adalah soal *capstone* yang mencakup **Sub-CPMK10.1.4** ("Mampu menjelaskan konsep Term Weighting, Search Engine dan Evaluasi Model").
    - **Term Weighting:** Kami membandingkan dua skema (TF-IDF standar vs. Sublinear TF).
    - **Search Engine:** Kami membangun *orchestrator* CLI (search.py) dan antarmuka Streamlit (app/main.py).
    - **Evaluasi Model:** Kami menghitung metrik Precision/Recall/F1 dan MAP@k untuk mengukur kualitas model.