Nama : Ramadhika Surya Purmiadanu

NIM : A11.2022.14116

Kelompok : A11.4703

PENDAHULUAN

a) Tujuan

Tujuan utama dari proyek ini untuk merancang, mengimplementasikan, dan mengevaluasi dua model fundamental Sistem Temu Kembali Informasi (STKI) pada korpus dokumen teks berukuran kecil.

Secara spesifik :

1. Mengimplementasikan Model Boolean secara manual, lengkap dengan struktur data *inverted index* dan operasi pencarian (AND, OR, NOT).
2. Mengimplementasikan Model Ruang Vektor (VSM) menggunakan pembobotan TF-IDF dan perankingan (ranking) berbasis Cosine Similarity.
3. Mengevaluasi secara kuantitatif kedua model tersebut menggunakan metrik standar (Precision, Recall, F1-Score untuk Boolean; P@k dan MAP@k untuk VSM).
4. Menganalisis dan membandingkan dampak dari skema pembobotan yang berbeda (TF-IDF standar vs. Sublinear TF-IDF) terhadap kualitas hasil pencarian VSM.

b) Ruang Lingkup Proyek

Ruang lingkup (scope) proyek ini mencakup keseluruhan alur kerja dasar sebuah sistem temu kembali informasi, mulai dari pengumpulan data hingga evaluasi. Alur kerja tersebut meliputi :

- Korpus Data: Penggunaan 5 dokumen teks statis sebagai sumber data.
- Preprocessing: Penerapan langkah-langkah *preprocessing* standar (tokenisasi, case-folding, stopword removal, dan stemming).
- Implementasi Model: Pembangunan dua model, yaitu:
    1. Boolean: Dibuat secara manual.
    2. VSM: Dibuat menggunakan *library* scikit-learn (TfidfVectorizer).

- Evaluasi Model: Pengujian model menggunakan *ground truth* (kunci jawaban) yang dibuat manual dan diukur dengan metrik yang sesuai (P/R/F1 dan MAP@k).
- Antarmuka: Pembuatan antarmuka pengguna (UI) sederhana menggunakan Streamlit untuk mendemonstrasikan fungsionalitas model VSM.

c) Kontribusi Proyek vs. Sub-CPMK

Proyek ini dirancang untuk secara langsung memenuhi dan mendemonstrasikan pemahaman atas beberapa Sub-Capaian Pembelajaran Mata Kuliah (Sub-CPMK) STKI, yaitu:

- Sub-CPMK10.1.2: mengimplementasikan fungsi preprocess_text yang mencangkup tokenisasi, case-folding, stopword removal, dan stemming.
- Sub-CPMK10.1.3: mengimplementasikan (Model Boolean dan inverted_index) dan mengimplementasikan (VSM dengan TF-IDF dan Cosine Similarity).
- Sub-CPMK10.1.4: menganalisis Term Weighting (TF-IDF Standar vs. Sublinear TF), melakukan Evaluasi Model (menggunakan F1-Score dan MAP@k), membangun Search Engine (antarmuka Streamlit app/main.py)

DATA & PRE-PROCESSING

a) Korpus Data

Korpus yang digunakan dalam proyek ini terdiri dari 5 dokumen teks (format .txt). Dokumen-dokumen ini berisi teks contoh (dummy text) yang relevan dengan tema akademik dan universitas, mencakup topik seperti (1) Fakultas Informatika, (2) Kegiatan Mahasiswa, (3) Informasi Beasiswa, (4) Perpustakaan, dan (5) Sistem Informasi.

b) Metodologi Preprocessing

Untuk mengubah teks mentah (*raw text*) menjadi data yang bersih dan terstruktur, sebuah fungsi preprocess_text diterapkan pada setiap dokumen. Fungsi ini melakukan empat langkah utama secara berurutan:

1. Tokenisasi: Memecah kalimat menjadi kata-kata (token) individual.

2. Case-Folding: Mengubah semua huruf menjadi huruf kecil (lowercase) untuk menyeragamkan kata (misal: "Mahasiswa" dan "mahasiswa" akan dianggap sebagai kata yang sama).
3. Stopword Removal: Menghapus kata-kata umum yang tidak memiliki makna signifikan (seperti "dan", "di", "yang", "adalah") menggunakan daftar stopword Bahasa Indonesia dari *library* Sastrawi.
4. Stemming: Mengubah kata-kata ke bentuk dasarnya (kata dasar) menggunakan *stemmer* Sastrawi (misal: "pembelajaran" diubah menjadi "ajar", "mengikuti" menjadi "ikut").

c) Contoh Hasil ”Before” vs ”After”

Berikut adalah contoh perbandingan hasil "Before" (sebelum) dan "After" (sesudah) proses *preprocessing* diterapkan pada salah satu dokumen:

- Contoh "Before" (Teks Mentah - korpus_mentah[0]): "Fakultas Informatika di Universitas Teknologi adalah pusat pembelajaran ilmu komputer. Mahasiswa baru mengikuti program orientasi universitas untuk mengenal lingkungan akademik."
- Contoh "After" (Teks Bersih - korpus_bersih[0]): "fakultas informatika universitas teknologi pusat ajar ilmu komputer mahasiswa baru ikut program orientasi universitas kenal lingkung akademik"

Seperti yang terlihat, teks "After" jauh lebih bersih, seragam, dan hanya berisi kata-kata inti yang penting untuk proses temu kembali informasi.

METODE TEMU KEMBALI INFORMASI (IR)

Proyek ini mengimplementasikan dua model fundamental dalam STKI: Model Boolean dan Model Ruang Vektor (VSM). Kedua model ini memiliki pendekatan yang berbeda secara fundamental dalam memproses *query* dan menentukan relevansi dokumen.

a) Model Boolean Retrieval

Model Boolean adalah model STKI klasik yang paling sederhana. Model ini memperlakukan setiap dokumen sebagai sekumpulan *term* (kata).

1. Struktur Konseptual: Incidence Matrix

Secara konseptual, Model Boolean dapat direpresentasikan oleh Incidence Matrix. Ini adalah sebuah matriks biner (0/1) yang besar di mana:

- Barisnya adalah semua *term* unik dalam *vocabulary* (total 51 *term*).
- Kolomnya adalah setiap dokumen dalam korpus (5 dokumen).
- Nilai 1 menandakan sebuah *term* ada di dalam dokumen, dan 0 menandakan *term* tersebut tidak ada.

2. Struktur Praktis: Inverted Index Model ini bergantung pada struktur data Inverted Index (Indeks Terbalik), yang juga telah kami implementasikan (create_inverted_index). Ini adalah sebuah struktur data (kamus) yang memetakan setiap *term* unik dalam korpus ke *postings list*, yaitu daftar ID dokumen yang mengandung *term* tersebut.
    - Contoh : {'informatika': {0, 2}, 'mahasiswa': {1, 2, 4}}
3. Logika Query Pencarian dilakukan dengan menerapkan operator logika himpunan (set theory) pada *postings list* dari *term* yang ada di *query*. Model ini menghasilkan jawaban yang absolut (YA/TIDAK)—sebuah dokumen dianggap relevan atau tidak sama sekali.
    - Query AND (Irisan): Menggunakan operator &. Mencari dokumen yang mengandung semua *term*.
    - Query OR (Gabungan): Menggunakan operator |. Mencari dokumen yang mengandung salah satu *term*.
    - Query NOT (Selisih): Menggunakan operator -. Mencari dokumen yang mengandung *term* pertama TAPI BUKAN *term* kedua.

b) Model Ruang Vektor (Vector Space Model - VSM)

Berbeda dengan Boolean, VSM adalah model *ranked retrieval* (pencarian berperingkat). Model ini tidak hanya menentukan apakah dokumen relevan, tetapi juga seberapa relevan dokumen tersebut.

1. Konsep Vektor Dokumen dan *query* direpresentasikan sebagai vektor dalam sebuah ruang multidimensi. Setiap dimensi dalam ruang ini mewakili satu *term* unik dari seluruh korpus (kamus).
2. Struktur Data: Matriks TF-IDF Nilai dari setiap komponen dalam vektor dokumen adalah sebuah bobot (weight) yang menunjukkan seberapa penting *term* tersebut di

dalam dokumen itu. Bobot yang paling umum digunakan adalah TF-IDF. Hasilnya adalah sebuah matriks TF-IDF, di mana baris adalah dokumen dan kolom adalah *term*.

3. Perankingan (Ranking) Peringkat relevansi dihitung berdasarkan "kedekatan" antara vektor *query* dengan setiap vektor dokumen. Kedekatan ini diukur menggunakan Cosine Similarity.

c) Formula TF, IDF, TF-IDF, dan Cosine

1. Term Frequency (TF)

   Mengukur seberapa sering sebuah *term* (t) muncul dalam sebuah dokumen (d).

   - Standar: $\text{tf}(t,d) = (\ j\text{umlah kemunculan term } t \text{ di dokumen } d\ )$
   - Sublinear TF-Scaling (Logarithmic): Formula ini meredam efek dari *term* yang muncul terlalu sering. $\text{tf}_{\text{sublinear}}(t,d) = 1\ + \log\big(\text{tf}(t,d)\big)(\text{jika tf} > 0)$.

2. Inverse Document Frequency (IDF)

   Mengukur seberapa unik atau langka sebuah *term* (t) di seluruh korpus (D). *Term* yang langka lebih informatif dan mendapat skor IDF tinggi.

   - Formula: $\text{idf}(t,D) = \log\left(\frac{N}{\text{df}(t)}\right)$

     Di mana:

     - N = Jumlah total dokumen dalam korpus.
     - df(t) = *Document Frequency*, yaitu jumlah dokumen yang mengandung *term t*.

3. TF-IDF

   Bobot akhir yang menggabungkan kedua perhitungan di atas. Sebuah term dianggap penting jika ia sering muncul di satu dokumen (TF tinggi) TAPI jarang muncul di dokumen lain (IDF tinggi).

   - Formula: $\omega(t,d) = \text{tf}(t,d) \times \text{idf}(t,D)$

4. Cosine Similarity

   Mengukur kemiripan antara dua vektor (misalnya, vektor *query* q dan vektor dokumen d) dengan menghitung kosinus sudut di antara keduanya. Skor 1.0 berarti identik, dan 0.0 berarti tidak ada kemiripan.

- Formula: $cosine(q, d) = \frac{(q \cdot d)}{\|q\| \times \|d\|} = \frac{\sum_i^n q_i d_i}{\sqrt{\sum_{i=1}^n q_i^2} \sqrt{\sum_{i=1}^n d_i^2}}$

  Di mana:

  - $q \cdot d = Dot\ product$ (produk titik) dari kedua vektor.
  - $\|q\| = Magnitude$ (panjang) dari vektor q.
  - $\|d\| = Magnitude$ (panjang) dari vektor d.

ARSITEKTUR SEARCH ENGINE

Arsitektur search engine yang diimplementasikan dalam proyek ini adalah Model Ruang Vektor (VSM) yang disajikan melalui antarmuka web. Arsitektur ini berfokus pada ranked retrieval, di mana dokumen diurutkan berdasarkan relevansinya terhadap query pengguna. Diagram alir di bawah ini mengilustrasikan proses query (pencarian) pada sistem VSM yang telah dibangun:

1. Input Pengguna (Query Mentah): Pengguna memasukkan *query* (misal: "informasi mahasiswa") melalui antarmuka (UI).
2. Preprocessing Query: *Query* mentah tersebut diproses menggunakan fungsi preprocess_text yang sama dengan yang digunakan untuk dokumen. Ini untuk memastikan konsistensi *term* (tokenisasi, fold, stopword, stem).
3. Vectorization Query: *Query* yang sudah bersih kemudian diubah menjadi vektor TF-IDF (vektor *query*) menggunakan vectorizer.transform(). Vektor *query* ini memiliki dimensi yang sama dengan matriks dokumen.
4. Perhitungan Kemiripan (Similarity): Vektor *query* dibandingkan dengan semua vektor dokumen yang ada di tfidf_matrix menggunakan Cosine Similarity. Langkah ini menghasilkan sebuah skor kemiripan (antara 0.0 hingga 1.0) untuk setiap dokumen.
5. Perankingan (Ranking): Skor kemiripan tersebut diurutkan dari yang tertinggi ke terendah. Sistem kemudian mengambil Top-k (misal, k=3) dokumen dengan skor tertinggi.
6. Presentasi Hasil: Hasil Top-k (peringkat, nama dokumen, skor, dan *snippet*) ditampilkan kepada pengguna melalui antarmuka Streamlit (app/main.py).

EKSPERIMEN & EVALUASI

a) Skenario Eksperimen

1. **Skenario 1 (Evaluasi Model Boolean):** Menguji akurasi Model Boolean dalam mengambil dokumen yang relevan secara eksak. Tiga *query* (AND, OR, NOT) dijalankan dan hasilnya (Dokumen Ditemukan) dibandingkan dengan Ground Truth (kunci jawaban).
2. **Skenario 2 (Perbandingan Skema VSM):** Menguji performa Model Ruang Vektor (VSM) dalam melakukan *ranking* (peringkat). Eksperimen ini juga membandingkan dua skema pembobotan TF-IDF yang berbeda:

- Skema 1: TF-IDF Standar.
- Skema 2: TF-IDF dengan *Sublinear TF-Scaling* (sublinear_tf=True).

b) Metrik Evaluasi

Untuk Model Boolean (Skenario 1):

- **Precision:** Seberapa banyak dokumen yang ditemukan sistem yang benar-benar relevan.
- **Recall:** Seberapa banyak dokumen relevan yang berhasil ditemukan oleh sistem.
- **F1-Score:** Rata-rata harmonik dari Precision dan Recall.

Untuk Model VSM (Skenario 2):

- **Precision@k (P@k):** Proporsi dokumen relevan yang ditemukan dalam $k=3$ hasil teratas.
- **Mean Average Precision@k (MAP@k):** Metrik utama untuk evaluasi *ranking*. Ini adalah rata-rata dari *Average Precision* (AP) untuk sekumpulan *query*, yang memperhitungkan posisi (urutan) dari dokumen relevan yang ditemukan.

c) HASIL DAN ANALISIS

1. Hasil Model Boolean

Query	Ditemukan (Sistem)	Relevan (Ground Truth)	Precision	Recall	F1-Score
'univ' DAN 'mhs'	{1,2}	{1}	0.50	1.00	0.67
'fakultas' ATAU 'mhs'	{0,1,2,3,4}	{0,1}	0.40	1.00	0.57
'univ' BUKAN 'mhs'	{0}	{0}	1.00	1.00	1.00
Rata-Rata (Macro)			0.63	1.00	0.75

Analisis:

- Model ini mencapai Recall sempurna (1.00), yang berarti sistem berhasil menemukan *semua* dokumen yang relevan untuk setiap *query*.

- Namun, Precision-nya rendah (rata-rata 0.63). Ini terlihat jelas pada *query* OR dan AND, di mana sistem juga mengembalikan dokumen yang tidak relevan (misal: dokumen {2} untuk *query* AND, dan {2, 3, 4} untuk *query* OR).
- Ini menyoroti kelemahan utama Model Boolean: model ini "kaku" dan tidak bisa membedakan *tingkat* relevansi, sehingga menghasilkan banyak *false positives* (terutama pada *query* OR).

2. Skenario 2: Hasil Perbandingan Skema VSM

Kedua skema VSM dievaluasi menggunakan 3 query yang sama terhadap ground truth VSM, dengan k=3.

Table 1 Hasil VSM (TF-IDF Standar)

Query	Dok. Ditemukan (Top 3)	P@3	AP@3
'informasi mahasiswa'	[1 2 4]	1.0000	0.7500
'fakultas informatika'	[0 4 3]	0.3333	1.0000
'informasi beasiswa'	[1 2 4]	0.3333	0.5000
Rata-Rata (Macro)		0.5556	0.7500

Table 2 Hasil VSM (Sublinear TF-IDF)

Query	Dok. Ditemukan (Top 3)	P@3	AP@3
'informasi mahasiswa'	[1 2 4]	1.0000	0.7500
'fakultas informatika'	[0 4 3]	0.3333	1.0000
'informasi beasiswa'	[2 1 4]	0.3333	1.0000
Rata-Rata (Macro)		0.5556	0.9167

Analisis:

1. Performa Keseluruhan: Seperti yang ditunjukkan oleh MAP@3, skema Sublinear TF-IDF (MAP@3 = 0.9167) terbukti unggul signifikan dibandingkan skema TF-IDF Standar (MAP@3 = 0.7500).
2. Akar Peningkatan: Peningkatan performa ini hanya terjadi pada *query* "informasi beasiswa".

- Pada skema Standar, dokumen relevan ({2}) berada di Peringkat 2 (hasil: [1 2 4]), sehingga skor AP@3-nya adalah 0.5.
- Pada skema Sublinear, *ranking* berubah. Dokumen relevan ({2}) berhasil naik ke Peringkat 1 (hasil: [2 1 4]). Ini memberikan skor AP@3 yang sempurna (1.0) untuk *query* tersebut.

3. Kesimpulan: Penerapan *logarithmic scaling* (sublinear_tf) terbukti efektif. Ini meredam bobot *term* yang terlalu sering muncul, sehingga mampu mengubah urutan *ranking* menjadi lebih akurat dan relevan, yang berdampak langsung pada peningkatan skor MAP.

DISKUSI

a) Kelebihan

Proyek ini berhasil mengimplementasikan dan mengevaluasi dua model IR fundamental, dengan beberapa temuan positif:

1. Validasi Model Boolean: Implementasi inverted_index manual terbukti 100% fungsional untuk *query* logis. Model ini berhasil mencapai Recall 1.00, yang menunjukkan kemampuannya untuk menemukan *semua* dokumen yang relevan.
2. Efektivitas VSM: Model VSM terbukti jauh lebih superior daripada Model Boolean dalam hal menyajikan hasil yang relevan. Kemampuannya untuk memberikan skor peringkat (bukan hanya jawaban YA/TIDAK) memecahkan masalah presisi rendah yang dialami Model Boolean.
3. Peningkatan Performa yang Terukur: Eksperimen Soal 05 secara kuantitatif membuktikan bahwa pemilihan skema pembobotan (Term Weighting) memiliki dampak langsung pada kualitas *ranking*. Penggunaan Sublinear TF-IDF secara jelas meningkatkan MAP@3 dari 0.7500 menjadi 0.9167, sebuah peningkatan yang signifikan.

b) Keterbatasan

1. Ukuran Korpus yang Sangat Kecil: Keterbatasan terbesar adalah penggunaan korpus yang hanya terdiri dari 5 dokumen. Hasil dan kesimpulan (terutama keunggulan Sublinear TF) mungkin tidak dapat digeneralisasi ke korpus yang lebih besar dan lebih beragam.

2. Simplisitas *Ground Truth*: Kunci jawaban (ground truth) dibuat secara manual untuk 3 *query* sederhana. Dalam sistem nyata, relevansi bersifat jauh lebih subjektif dan kompleks.
3. Keterbatasan Presisi Boolean: Seperti yang ditunjukkan oleh hasil evaluasi, Model Boolean menderita Precision yang rendah (0.63). Ini adalah kelemahan inheren dari model tersebut, yang tidak dapat membedakan tingkat kepentingan dokumen.
4. Ketergantungan pada *Stemmer*: Kualitas hasil *preprocessing* sangat bergantung pada *stemmer* (Sastrawi). Kesalahan *stemming* dapat menyebabkan *mismatch* antara *query* dan dokumen.

c) Saran Pengembangan

1. Ekspansi Korpus: Langkah paling krusial adalah memperluas korpus data secara signifikan (misalnya, menjadi 100+ dokumen) untuk menguji skalabilitas dan keandalan model.
2. Implementasi Model BM25: Sebagai langkah selanjutnya dari VSM, mengimplementasikan model Okapi BM25 (yang juga disebutkan di soal sebagai opsi bonus). BM25 adalah model probabilistik yang seringkali memberikan hasil lebih baik daripada TF-IDF murni.
3. Ekspansi *Query* (Query Expansion): Mengimplementasikan teknik di mana sistem secara otomatis menambahkan sinonim atau *term* terkait ke *query* pengguna untuk meningkatkan *recall* tanpa mengorbankan presisi.
4. Analisis *Error*: Melakukan analisis *error* yang lebih mendalam pada hasil pencarian (misalnya, mengapa *query* 'fakultas informatika' hanya memiliki P@3 sebesar 0.33) untuk mengidentifikasi kelemahan spesifik pada *preprocessing* atau pembobotan.

KESIMPULAN (CAPAIAN SUB-CPMK)

a) Sub-CPMK10.1.1: Mampu menjelaskan konsep-konsep STKI.

- Capaian: Tercapai melalui Bab 1 (Pendahuluan) dan Bab 4 (Arsitektur). Laporan ini telah menjelaskan konsep inti STKI, perbedaannya dengan *database retrieval*, serta memaparkan diagram alir arsitektur untuk Model Boolean dan Model Ruang Vektor (VSM).

b) Sub-CPMK10.1.2: Mampu menjelaskan Document Preprocessing.

- Capaian: Tercapai melalui Bab 2 (Data & Preprocessing). Kami telah berhasil mengimplementasikan *pipeline preprocessing* lengkap (tokenisasi, case-folding, stopword removal, dan stemming Sastrawi) dan menunjukkan hasilnya secara nyata melalui contoh "Before vs After".

c) **Sub-CPMK10.1.3: Mampu menjelaskan Pemodelan... Boolean... dan Vector Space Model.**
   - Capaian: Tercapai melalui Bab 3 (Metode IR) dan Bab 5 (Eksperimen).
     1. Model Boolean: Kami berhasil mengimplementasikan inverted_index manual dan fungsi pencarian search_and/or/not.
     2. Model VSM: Kami berhasil mengimplementasikan VSM menggunakan TfidfVectorizer dan cosine_similarity untuk menghasilkan sistem pencarian berperingkat (ranking).

d) **Sub-CPMK10.1.4: Mampu menjelaskan konsep Term Weighting, Search Engine dan Evaluasi Model.**
   - Capaian: Tercapai melalui Bab 5 (Eksperimen) dan Bab 6 (Diskusi).
     1. Evaluasi Model: Kami berhasil mengevaluasi kedua model menggunakan metrik yang tepat (Precision/Recall/F1-Score untuk Boolean dan MAP@k untuk VSM).
     2. Term Weighting: Kami berhasil membandingkan dua skema *term weighting* (TF-IDF Standar vs. Sublinear TF) dan membuktikan secara kuantitatif bahwa Sublinear TF (MAP@3 = 0.9167) lebih unggul.
     3. Search Engine: Seluruh komponen telah disatukan dalam sebuah aplikasi antarmuka (UI) Streamlit (app/main.py), yang membuktikan fungsionalitas *search engine* secara *end-to-end*.