LAPORAN PROYEK UJIAN TENGAH SEMESTER

Implementasi Mini Search Engine (Boolean & VSM) pada Korpus Berita

Disusun untuk memenuhi Ujian Tengah Semester (UTS)

Mata Kuliah Sistem Temu Kembali Informasi (A11.4703)

Dosen Pengampu:

Abu Salam, M.Kom

Disusun Oleh:

RIZKA NUGRAHA

A11.2022.14119

PROGRAM STUDI TEKNIK INFORMATIKA (S1)

FAKULTAS ILMU KOMPUTER

UNIVERSITAS DIAN NUSWANTORO

2025

1. PENDAHULUAN

1.1 Latar Belakang dan Tujuan

Laporan ini dibuat untuk memenuhi tugas Ujian Tengah Semester (UTS) mata kuliah Sistem Temu Kembali Informasi (STKI). Tujuan utama dari proyek ini adalah untuk merancang, mengimplementasikan, dan mengevaluasi sebuah sistem temu kembali informasi (STKI) mini dari awal[3]. Proyek ini mengimplementasikan dua model retrieval klasik, yaitu **Boolean Retrieval Model** dan **Vector Space Model (VSM)**, pada korpus teks berukuran kecil.

1.2 Ruang Lingkup

Ruang lingkup proyek ini mencakup:

1. Penggunaan korpus data yang terdiri dari 5 dokumen teks (.txt) berita bertema COVID-19.
2. Implementasi pipeline *document preprocessing* lengkap dalam Bahasa Indonesia (case folding, tokenisasi, stopword removal, dan stemming)[6].
3. Implementasi **Boolean Retrieval Model** dengan *inverted index* untuk memproses kueri AND, OR, dan NOT[7].
4. Implementasi **Vector Space Model (VSM)** dengan pembobotan **TF-IDF** dan perankingan hasil menggunakan **Cosine Similarity**[8].
5. Evaluasi kedua model menggunakan metrik standar (Precision, Recall, F1-Score, dan MAP@k).
6. Pembuatan antarmuka pengguna (UI) interaktif menggunakan Streamlit.

1.3 Peta Kaitan Proyek terhadap Sub-CPMK

Proyek ini secara langsung mengimplementasikan empat Sub-CPMK yang tercantum dalam RPS[10101010101010]:

- **Sub-CPMK10.1.1 (Konsep STKI):** Dipenuhi melalui pengerjaan **Soal 01 (Esai)**, yang menjelaskan konsep dasar STKI, arsitektur, dan perbedaannya dengan database retrieval.
- **Sub-CPMK10.1.2 (Document Preprocessing):** Diimplementasikan dalam **Soal 02** melalui modul src/preprocess.py. Modul ini menangani seluruh tahapan preprocessing yang diminta[11].
- **Sub-CPMK10.1.3 (Pemodelan Boolean & VSM):** Diimplementasikan dalam **Soal 03** (src/boolean_ir.py) dan **Soal 04** (src/vsm_ir.py). Kedua model inti STKI berhasil dibangun dan diuji.
- **Sub-CPMK10.1.4 (Term Weighting, Search Engine, Evaluasi):** Diimplementasikan dalam **Soal 05**. Proyek ini membandingkan skema *term weighting* (TF-IDF vs Sublinear TF-IDF), membangun *search engine* utuh dengan antarmuka Streamlit (app/main.py), dan melakukan evaluasi model secara komprehensif

2. DATA DAN PREPROCESSING

2.1 Data (Korpus)

Korpus yang digunakan dalam proyek ini terdiri dari 5 dokumen teks berita (berita1.txt s.d. berita5.txt) yang diambil dari portal berita Detik Health. Seluruh dokumen membahas topik seputar COVID-19, varian Delta, dan vaksinasi.

2.2 Document Preprocessing (Soal 02)

Preprocessing adalah tahapan krusial untuk mengubah data teks tidak terstruktur menjadi format yang bersih dan terstruktur untuk pengindeksan. Tahapan ini diimplementasikan dalam modul src/preprocess.py dan mencakup langkah-langkah berikut:

1. **Case Folding:** Mengubah seluruh teks menjadi huruf kecil (lower()).
2. **Pembersihan Tanda Baca & Angka:** Menghapus semua karakter tanda baca, angka, dan URL.
3. **Tokenisasi:** Memecah kalimat menjadi token (kata) individual menggunakan nltk.word_tokenize.
4. **Stopword Removal:** Menghapus kata-kata umum dalam Bahasa Indonesia (misal: 'dan', 'di', 'yang') menggunakan daftar stopwords dari NLTK.
5. **Stemming:** Mengubah setiap kata ke bentuk dasarnya (misal: "meningkat" -> "tingkat") menggunakan *stemmer* dari library Sastrawi.

2.3 Contoh Hasil Preprocessing

Berikut adalah perbandingan *before/after* dari berita1.txt yang diambil dari eksekusi notebook:

BEFORE (Teks Asli):

Wilayah Kamu Sudah 'Bebas' COVID-19? Cek 34 Kab/Kota Zona Hijau Terbaru Jakarta - Pemerintah rencananya bakal menerapkan Pemberlakuan Pembatasan Kegia...

AFTER (Teks Terproses):

wilayah bebas covid cek kabkota zona hijau baru jakarta perintah rencana terap laku batas giat masyarakat ppkm level hitung desember januari menteri s...

3. METODE INFORMATION RETRIEVAL

3.1 Model Boolean (Soal 03)

Model Boolean adalah model retrieval klasik yang memproses kueri sebagai ekspresi boolean. Implementasi model ini (src/boolean_ir.py) bergantung pada struktur data inti yang disebut **Inverted Index**.

Indeks Terbalik ini adalah sebuah *dictionary* Python yang memetakan setiap *term* (kata yang sudah diproses) ke sebuah *set* (himpunan) ID dokumen yang mengandung *term* tersebut. Saat kueri (misal: "vaksin AND delta") dieksekusi, sistem melakukan operasi himpunan pada *postings list* dari *term* tersebut:

- **AND**: Irisan (Intersection) dari himpunan.
- **OR**: Gabungan (Union) dari himpunan.
- **NOT**: Komplemen (Difference) dari himpunan.

Kelemahan utama model ini adalah hasilnya bersifat biner (cocok atau tidak) dan tidak memiliki perankingan.

3.2 Vector Space Model (VSM) (Soal 04)

VSM merepresentasikan dokumen dan kueri sebagai vektor dalam ruang multidimensi di mana setiap dimensi mewakili satu *term* unik. Implementasi model ini (src/vsm_ir.py) menggunakan pembobotan **TF-IDF** untuk menentukan nilai setiap komponen dalam vektor.

3.2.1 Formula Term Weighting

Formula yang digunakan (sesuai implementasi scikit-learn dan permintaan soal) adalah:

1. **Term Frequency (TF):** Mengukur seberapa sering sebuah *term* t muncul dalam dokumen d. Dalam proyek ini (Soal 05), kami membandingkan dua skema:
    - **TF Standar:** $tf(t,d) = f_{t,d}$ (frekuensi mentah)
    - **Sublinear TF (default di app/main.py):** $tf(t,d) = 1 + \log(f_{t,d})$
2. Inverse Document Frequency (IDF): Mengukur seberapa unik atau penting sebuah term t di seluruh korpus D (berisi N dokumen).

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

Catatan: Ini adalah formula IDF smooth yang digunakan oleh scikit-learn.

3. TF-IDF Weight: Bobot final adalah perkalian dari TF dan IDF.

$$w_{t,d} = tf(t,d) \times idf(t,D)$$

3.2.2 Perankingan (Ranking)

Setelah dokumen dan kueri diubah menjadi vektor TF-IDF ($\vec{d}$ dan $\vec{q}$), relevansi dihitung menggunakan Cosine Similarity23. Formula ini mengukur sudut antara dua vektor:

$$sim(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| |\vec{d}|}$$

Hasil pencarian kemudian diurutkan dari skor similaritas tertinggi ke terendah, menghasilkan daftar dokumen yang terperingkat (ranked list).

4. ARSITEKTUR SEARCH ENGINE

Arsitektur *search engine* mini ini terbagi menjadi dua bagian utama: **Indeksasi** (dilakukan saat load_models()) dan **Retrieval** (dilakukan saat pengguna mencari).

Diagram alir berikut menunjukkan alur logika *retrieval* di dalam aplikasi Streamlit (app/main.py):

Code snippet

```
graph TD
    A[Pengguna Buka Aplikasi Streamlit] --> B{Muat Model (cache_resource)};
    B --> C[load_models()];
    C --> D{Cek 'data/processed'};
    D -- Ada --> F[Muat Model Boolean & VSM];
    D -- Kosong --> E[Jalankan Preprocessing (Soal 02)];
    E --> F;
    F --> G[Aplikasi Siap];

    G --> H[Pengguna Masukkan Kueri & Pilih Model];
    H --> I{Model == 'Boolean'?}
    I -- Ya --> J[Gunakan 'boolean_ir.py' (Soal 03)];
    J --> K[Operasi Himpunan pada Inverted Index];
    K --> M[Tampilkan Hasil (Tidak Terurut)];

    I -- Tidak (VSM) --> L[Gunakan 'vsm_ir.py' (Soal 04)];
    L --> N[Hitung Cosine Similarity pada Matriks TF-IDF];
    N --> O[Tampilkan Hasil Top-K (Terurut)];
```

5. EKSPERIMEN DAN EVALUASI

Eksperimen dilakukan untuk menguji performa dari ketiga model (Boolean, VSM Default, VSM Sublinear) menggunakan *gold standard* (kunci jawaban) yang dibuat secara manual.

5.1 Evaluasi Model Boolean (Soal 03)

Model Boolean dievaluasi menggunakan metrik Precision, Recall, dan F1-Score pada 3 kueri. Hasil eksekusi dari notebook adalah sebagai berikut:

Kueri	Retrieved	Relevant (Kunci)	Precision	Recall	F1-Score
vaksin AND delta	1	1	1.0	1.0	1.000000
ppkm OR jakarta	5	2	0.4	1.0	0.571429
kasus NOT amerika	4	4	1.0	1.0	1.000000

Analisis:

- Kueri AND dan NOT bekerja dengan sempurna (F1 = 1.0).
- Kueri ppkm OR jakarta memiliki *Precision* sangat rendah (0.4). Ini karena *term* "jakarta" (setelah di-stem) muncul di kelima dokumen, sehingga model mengembalikan semua dokumen, meskipun hanya dua yang relevan dengan "ppkm" atau "jakarta" dalam konteks yang benar.

5.2 Evaluasi Model VSM (Soal 04)

Model VSM (Default TF-IDF) dievaluasi menggunakan P@3 (Precision at 3) dan MAP@5 (Mean Average Precision at 5). Hasil eksekusi dari notebook adalah sebagai berikut:

Kueri	P@3
vaksin AND delta	0.666667
ppkm OR jakarta	0.666667
kasus NOT amerika	0.000000
Skor Total MAP@5	**0.6667**

Analisis:

- VSM menunjukkan performa yang baik untuk kueri AND dan OR, menemukan dokumen yang relevan di 3 besar.
- VSM gagal total (P@3 = 0.0) pada kueri kasus NOT amerika. Ini wajar, karena VSM tidak mengerti logika boolean NOT. Model ini justru mencari *term* "kasus" dan "amerika", sehingga mengembalikan berita5.txt (yang seharusnya dikecualikan) sebagai hasil teratas.

- Skor **MAP@5 sebesar 0.6667** menunjukkan bahwa secara rata-rata, model perankingan ini cukup baik untuk korpus kecil ini.

5.3 Perbandingan Term Weighting (Soal 05)

Soal 05 meminta perbandingan setidaknya dua skema *term weighting*[25]. Kami membandingkan **Default TF-IDF** (frekuensi mentah) dengan **Sublinear TF-IDF** (menggunakan $1 + \log(tf)$).

Hasil eksekusi perbandingan MAP@5 adalah sebagai berikut:

Skema Weighting	MAP@5
Default TF-IDF	0.666667
Sublinear TF-IDF	0.666667

Analisis:

Pada korpus yang sangat kecil ini (5 dokumen), tidak ada perbedaan performa yang terdeteksi antara kedua skema pembobotan. Sublinear TF biasanya berguna pada korpus besar untuk mengurangi dampak dari term yang sangat sering muncul (misal: 1000x) agar tidak mendominasi skor dokumen. Pada data kami, frekuensi term tidak cukup bervariasi untuk menunjukkan keunggulan skema ini.

6. DISKUSI

6.1 Kelebihan dan Keterbatasan

Kelebihan:

- **Arsitektur Modular:** Proyek dibangun dengan struktur folder yang rapi (src, app, data), memisahkan logika inti (model) dari antarmuka (UI) .
- **Implementasi Lengkap:** Berhasil mengimplementasikan dua model retrieval fundamental (Boolean dan VSM) dari awal.
- **Antarmuka Interaktif:** Penggunaan Streamlit (app/main.py) memungkinkan pengujian dan demo yang mudah, termasuk tab evaluasi *real-time*.

Keterbatasan:

- **Korpus Sangat Kecil:** 5 dokumen tidak cukup untuk mengevaluasi model secara statistik. Performa model (misal: P@3 = 0.0) sangat sensitif terhadap satu dokumen yang salah.
- **Gold Standard Subjektif:** Kunci jawaban dibuat secara manual dan mungkin bias.
- **Parser Boolean Sederhana:** Parser kueri boolean tidak mendukung operasi kompleks seperti tanda kurung ().

- **VSM Tidak Mendukung Boolean:** Model VSM tidak dapat memproses operator NOT, yang merupakan kelemahan umum model ini.

6.2 Saran Pengembangan

1. **Korpus Lebih Besar:** Menggunakan korpus standar (misal: Kumpulan data CORD-19) untuk mendapatkan hasil evaluasi yang lebih valid.
2. **Implementasi BM25:** Menambahkan model Okapi BM25, yang seringkali memberikan performa lebih baik daripada TF-IDF murni (seperti yang disarankan sebagai bonus di Soal 05).
3. **Parser Kueri:** Mengembangkan parser kueri yang lebih canggih (misal: menggunakan Shunting-yard) untuk mendukung tanda kurung dan kueri campuran (misal: (vaksin OR booster) AND delta).

7. KESIMPULAN

Proyek UTS STKI ini telah berhasil diselesaikan sesuai dengan semua spesifikasi soal.

Sebuah *search engine* mini telah dibangun dari awal, mencakup *pipeline preprocessing* lengkap untuk Bahasa Indonesia (Soal 02), implementasi **Boolean Retrieval Model** dengan *inverted index* (Soal 03), dan **Vector Space Model (VSM)** dengan perankingan TF-IDF/Cosine Similarity (Soal 04).

Seluruh modul disatukan dalam sebuah aplikasi Streamlit yang fungsional dan dilengkapi dengan tab evaluasi *real-time* untuk membandingkan skema *term weighting*, yang memenuhi semua persyaratan **Soal 05**.

Seluruh **Sub-CPMK** yang ditargetkan (10.1.1, 10.1.2, 10.1.3, dan 10.1.4) [28] telah berhasil dicapai melalui kombinasi implementasi kode, pengujian di notebook, dan analisis dalam laporan ini.

8. DAFTAR PUSTAKA

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc. *(Referensi untuk library NLTK yang digunakan untuk tokenisasi dan stopwords)*

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research. *(Referensi untuk library Scikit-learn yang digunakan untuk TfidfVectorizer dan cosine_similarity)*

Adriani, M., Adri, J., & Arfan, M. (2007). *Stemming Algorithm for Indonesian Language (Sastrawi)*. *(Referensi untuk library Sastrawi yang digunakan untuk stemming Bahasa Indonesia. Anda bisa mencari referensi paper aslinya jika ada).*

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. Proceedings of the 9th Python in Science Conference. *(Referensi untuk library Pandas yang digunakan untuk analisis dan tabel evaluasi)*

Streamlit Inc. (2024). *Streamlit: The fastest way to build and share data apps*. *(Referensi untuk library Streamlit yang digunakan untuk membangun antarmuka pengguna)*