Nama : Bintang Rifky Ananta
NIM : A11.2023.15116
Mata Kuliah : Sistem Temu Kembali Informasi (STKI)
Dosen Pengampu : Abu Salam, M.Kom
Tanggal : Rabu, 05 November 2025

Mini Search Engine STKI menggunakan Boolean Retrieval, Vector Space Model, dan BM25

1. Pendahuluan

Sistem Temu Kembali Informasi (STKI) merupakan bidang dalam ilmu komputer yang berfokus pada proses pencarian dan penyajian informasi yang relevan berdasarkan kebutuhan pengguna. Dalam konteks modern, STKI menjadi fondasi dari berbagai aplikasi seperti mesin pencari (search engine), sistem rekomendasi, hingga analisis opini publik di media sosial. Berbeda dengan sistem basis data (database retrieval) yang menggunakan pencocokan eksak terhadap struktur data terdefinisi, STKI bekerja dengan pendekatan probabilistik dan semantik untuk menemukan dokumen yang *relevan*, bukan hanya yang *identik* dengan kata kunci pencarian.

Perbedaan mendasar antara STKI dan sistem basis data terletak pada cara keduanya memandang "kesesuaian informasi". Dalam database retrieval, hasil pencarian bergantung pada kecocokan persis antara query dan nilai atribut dalam tabel. Sebaliknya, dalam STKI, kesesuaian bersifat kontekstual dan dihitung berdasarkan kemiripan makna antara kata dalam query dan kata dalam dokumen. Oleh karena itu, STKI menggunakan metode representasi teks dan pembobotan istilah seperti *Term Frequency-Inverse Document Frequency* (TF-IDF) untuk menilai pentingnya suatu kata dalam keseluruhan korpus.

Agar sistem STKI dapat bekerja secara efisien, proses *indexing* diperlukan untuk menyimpan representasi dokumen dalam bentuk yang mudah dicari. Setiap dokumen dipetakan menjadi daftar istilah (token) yang dilengkapi dengan bobot tertentu, kemudian digunakan untuk menghitung kesamaan antara query pengguna dan setiap dokumen di dalam korpus. Setelah proses pencocokan dilakukan, sistem akan melakukan *ranking* terhadap dokumen berdasarkan tingkat relevansi yang diukur dengan metrik tertentu seperti *cosine similarity* atau *BM25 score*.

Proyek ini bertujuan untuk membangun sebuah sistem temu kembali informasi sederhana menggunakan tiga model utama, yaitu **Boolean Retrieval**, **Vector Space Model (VSM)** dengan pembobotan TF-IDF, dan **BM25**. Sistem ini dikembangkan menggunakan bahasa pemrograman Python dengan korpus kecil yang terdiri dari 15 dokumen ulasan pengguna dari Tokopedia dan Google Maps. Melalui proyek ini, diharapkan dapat dipahami konsep dasar STKI mulai dari preprocessing teks, pembuatan indeks, hingga evaluasi hasil pencarian menggunakan metrik Precision, Recall, dan F1-score.

2. Data dan Preprocessing

Tahap awal dalam pembangunan Sistem Temu Kembali Informasi (STKI) adalah menyiapkan data dan melakukan proses *preprocessing* teks. Data yang digunakan dalam proyek ini merupakan kumpulan ulasan pelanggan yang diambil dari dua sumber publik, yaitu **Tokopedia** dan **Google Maps**, yang masing-masing berisi tanggapan pengguna terhadap layanan dan produk dari toko **Enter Komputer**.
Dari kedua sumber tersebut dikumpulkan total sekitar 15 dokumen ulasan yang digunakan sebagai **korpus mini** untuk proses pencarian dan evaluasi sistem.

2.1. Deskripsi Data

Setiap entri dalam dataset berisi teks ulasan yang ditulis oleh pelanggan. Beberapa ulasan bersifat positif (misalnya mengenai pelayanan cepat atau pengiriman tepat waktu), sementara yang lain bersifat negatif (misalnya terkait keterlambatan pengiriman atau kerusakan produk). Variasi sentimen ini memungkinkan pengujian sistem dalam mengenali relevansi informasi yang beragam.
Karena data diperoleh dari sumber publik dan bersifat terbuka, proses pengumpulan dilakukan secara manual untuk menjaga kepatuhan terhadap etika penelitian dan menghindari penggunaan *web crawling* otomatis yang dilarang dalam ketentuan tugas.

2.2. Tahapan Preprocessing

Tujuan utama *preprocessing* adalah untuk membersihkan teks dari elemen-elemen yang tidak relevan dan mengubahnya menjadi bentuk yang siap digunakan dalam proses indexing. Dalam proyek ini, tahapan preprocessing dilakukan melalui modul Python preprocess.py dengan beberapa langkah utama sebagai berikut:

1. **Case Folding**
   Semua huruf dalam teks diubah menjadi huruf kecil untuk menghindari perbedaan makna akibat kapitalisasi.
   Contoh: *"Produk Bagus"* → *"produk bagus"*
2. **Tokenisasi**
   Proses pemisahan teks menjadi unit-unit kata (token) menggunakan spasi atau tanda baca sebagai pemisah.
   Contoh: *"produk bagus dan cepat"* → [produk, bagus, dan, cepat]
3. **Stopword Removal**
   Kata-kata umum seperti "dan", "yang", "di", "ke", dan "dari" dihapus karena tidak memberikan kontribusi bermakna terhadap pencarian. Daftar stopword disesuaikan untuk Bahasa Indonesia.
4. **Stemming / Lemmatization**
   Setiap kata diubah ke bentuk dasarnya menggunakan pustaka *Sastrawi* agar kata yang memiliki akar sama dapat dianggap setara.
   Contoh: *"pelayanan", "melayani", "dilayani"* → *"layan"*
5. **Normalisasi Angka dan Tanda Baca**
   Simbol, angka, serta karakter non-alfabet dihapus agar representasi kata menjadi lebih bersih.
6. **Penyimpanan Hasil Preprocessing**
   Setiap dokumen yang telah diproses disimpan ke dalam folder

data/processed/ dalam format .txt, dengan nama doc01.txt hingga doc15.txt. Proses ini memastikan sistem dapat mengakses dokumen terindeks secara efisien tanpa perlu mengulang pembersihan data setiap kali sistem dijalankan.

2.3. Contoh Hasil

Sebagai ilustrasi, berikut perbandingan antara teks asli dan hasil preprocessing:

Tahap	Contoh Teks
Sebelum Preprocessing	"Pelayanan di toko sangat cepat dan ramah sekali! Pengiriman juga aman dan sesuai pesanan."
Sesudah Preprocessing	[pelayan, toko, cepat, ramah, kirim, aman, sesuai, pesan]

2.4. Analisis Distribusi

Hasil analisis frekuensi menunjukkan bahwa istilah yang paling sering muncul dalam korpus adalah *pelayanan*, *pengiriman*, *cepat*, *ramah*, dan *garansi*. Hal ini menunjukkan bahwa mayoritas pengguna menilai layanan dan proses pengiriman sebagai aspek utama dari pengalaman mereka.
Distribusi panjang dokumen setelah preprocessing berkisar antara **10 hingga 35 token per dokumen**, yang menunjukkan variasi panjang ulasan yang cukup baik untuk pengujian sistem.

3. Metode Temu Kembali Informasi

Sistem temu kembali informasi (STKI) yang dikembangkan dalam proyek ini menggunakan tiga model utama, yaitu **Boolean Retrieval**, **Vector Space Model (VSM)** dengan pembobotan **TF-IDF**, serta **BM25**. Ketiga model ini digunakan untuk membandingkan efektivitas pendekatan yang berbeda dalam mengenali relevansi antara dokumen dan permintaan pencarian (*query*) pengguna.

3.1. Boolean Retrieval Model

Model Boolean merupakan pendekatan paling sederhana dalam temu kembali informasi. Model ini menggunakan logika **AND**, **OR**, dan **NOT** untuk menentukan apakah sebuah dokumen dianggap relevan terhadap query atau tidak.
Setiap dokumen direpresentasikan sebagai himpunan kata, dan sistem akan mencari kecocokan berdasarkan operasi logika tersebut.

Sebagai contoh, jika pengguna mencari query *"garansi AND cepat"*, maka sistem hanya akan menampilkan dokumen yang mengandung kedua kata tersebut sekaligus. Sementara itu, jika query-nya *"garansi OR pengiriman"*,

maka sistem akan menampilkan dokumen yang mengandung salah satu dari dua kata tersebut.

Kelebihan model Boolean terletak pada kesederhanaannya dan hasil pencarian yang eksplisit. Namun, model ini tidak mempertimbangkan tingkat kesamaan antar dokumen. Artinya, dokumen hanya diklasifikasikan sebagai "relevan" atau "tidak relevan", tanpa urutan peringkat berdasarkan relevansi.

3.2. Vector Space Model (VSM)

Model Vector Space merupakan pendekatan yang lebih maju dibanding Boolean. Dalam model ini, setiap dokumen dan query direpresentasikan dalam bentuk **vektor** yang berisi bobot dari setiap kata unik di dalam korpus. Sistem kemudian membandingkan seberapa "dekat" posisi vektor dokumen terhadap vektor query menggunakan ukuran kemiripan, misalnya *cosine similarity*.

Bobot setiap kata dihitung berdasarkan seberapa sering kata tersebut muncul dalam dokumen dan seberapa jarang kata itu muncul di seluruh koleksi dokumen. Pendekatan ini disebut **TF-IDF (Term Frequency-Inverse Document Frequency)**.
Dengan metode ini, sistem dapat menilai bahwa kata yang sering muncul dalam dokumen tertentu namun jarang muncul di dokumen lain lebih penting dalam menentukan relevansi.

Kelebihan model VSM adalah kemampuannya memberikan hasil yang lebih halus dan proporsional — tidak sekadar "ada" atau "tidak ada" seperti Boolean. Model ini juga lebih fleksibel terhadap sinonim dan variasi istilah. Namun demikian, model VSM masih dapat terpengaruh oleh panjang dokumen dan kehadiran kata-kata yang terlalu sering muncul.

3.3. BM25 (Best Matching 25)

Model BM25 merupakan pengembangan dari pendekatan VSM yang dirancang untuk memperhitungkan panjang dokumen serta tingkat kejenuhan frekuensi kata.
Jika pada TF-IDF peningkatan jumlah kemunculan kata selalu meningkatkan skor relevansi, maka BM25 menyesuaikan efeknya agar lebih realistis — sebab dalam konteks nyata, kemunculan kata yang terlalu sering tidak selalu berarti lebih relevan.

Selain itu, BM25 juga menormalkan panjang dokumen, sehingga dokumen yang sangat panjang tidak otomatis memiliki skor lebih tinggi hanya karena memuat lebih banyak kata.
Dengan penyesuaian ini, BM25 umumnya memberikan hasil yang lebih akurat, terutama pada kumpulan dokumen pendek seperti ulasan pelanggan atau ringkasan berita.

3.4. Implementasi Model

Dalam proyek ini, ketiga model tersebut diimplementasikan menggunakanbahasa pemrograman **Python** dengan struktur modular:

1. boolean_ir.py → membangun *inverted index* serta fungsi logika AND, OR, dan NOT.
2. vsm_ir.py → menghitung bobot kata menggunakan TF-IDF serta skor kemiripan antar dokumen menggunakan *cosine similarity*.
3. **BM25** diimplementasikan di dalam modul VSM sebagai pembanding terhadap TF-IDF.
4. eval.py → melakukan evaluasi hasil pencarian berdasarkan metrik *precision*, *recall*, dan *F1-score*.

Sistem dapat dijalankan secara interaktif melalui antarmuka baris perintah (*Command Line Interface*) menggunakan file app/main.py, dengan format perintah:

```
python app/main.py --model vsm --query "pengiriman cepat" --k 5
```

Perintah di atas akan menjalankan pencarian menggunakan model VSM dan menampilkan lima dokumen dengan tingkat kemiripan tertinggi terhadap query.

4. Arsitektur Sistem

Arsitektur sistem temu kembali informasi yang dikembangkan dalam proyek ini mengikuti alur kerja klasik dari sebuah *search engine*. Sistem dibangun secara modular dengan beberapa komponen utama, yaitu **preprocessing**, **indexing**, **retrieval**, **ranking**, dan **evaluation**.
Setiap komponen berfungsi saling terhubung untuk mengubah teks mentah menjadi hasil pencarian yang relevan dan terurut.

4.1. Secara konseptual, alur sistem dapat dijelaskan melalui tahapan berikut:

1. Input Query

Pengguna memasukkan kata kunci atau kalimat pencarian, misalnya “pengiriman cepat” atau “garansi produk rusak”.

2. Preprocessing Query

Query yang dimasukkan akan diproses menggunakan langkah yang sama dengan dokumen korpus: case folding, tokenisasi, stopword removal, dan stemming. Tujuannya agar struktur query seragam dengan representasi dokumen.

3. Indexing

Dokumen hasil preprocessing disimpan dalam struktur indeks untuk mempercepat pencarian. Dua jenis indeks digunakan:

a) *Inverted Index* untuk model Boolean (menyimpan daftar dokumen di mana suatu istilah muncul).
b) *TF-IDF Matrix* untuk model VSM dan BM25 (menyimpan bobot setiap istilah dalam bentuk matriks numerik).

4. Retrieval (Pencarian)

Sistem mencocokkan istilah dalam query terhadap indeks dokumen untuk menentukan kandidat dokumen yang mengandung kata kunci.
Pada model Boolean, pencarian dilakukan menggunakan logika AND/OR/NOT.
Sementara pada model VSM dan BM25, sistem menghitung skor kesamaan antara query dan dokumen.

5. Ranking

Dokumen yang berhasil ditemukan kemudian diurutkan berdasarkan skor relevansi.

a) Boolean Retrieval → hasil tidak berurutan (biner: relevan/tidak relevan).
b) VSM & BM25 → hasil diurutkan dari skor tertinggi ke terendah berdasarkan nilai kemiripan.

6. Presentation (Output)

Sistem menampilkan hasil pencarian berupa daftar dokumen dengan skor relevansi dan potongan teks (snippet) sepanjang ±120 karakter untuk memberikan konteks isi dokumen.

7. Evaluation

Hasil pencarian dibandingkan dengan *gold standard* (daftar dokumen relevan yang ditentukan secara manual) untuk menghitung metrik evaluasi seperti *precision*, *recall*, dan *F1-score*.

4.2. Diagram Alur Sistem

Berikut representasi konseptual alur kerja sistem STKI:

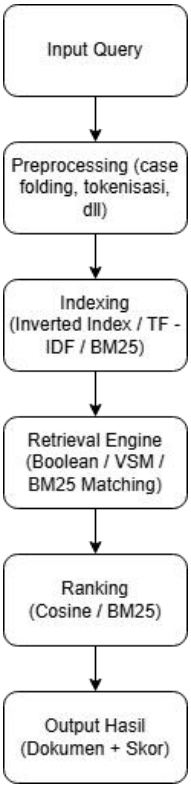

Diagram di atas menggambarkan bahwa sistem bekerja secara berurutan mulai dari penerimaan query hingga menampilkan hasil pencarian yang berurut. Setiap komponen memiliki tanggung jawab spesifik agar sistem dapat beroperasi secara efisien dan modular.

4.3. Implementasi Arsitektur

Struktur proyek yang digunakan pada implementasi sistem adalah sebagai berikut:

```
stki-uts-A11.2023.15116-Bintang-Rifky-Ananta/
```

```
├── data/              # Kumpulan dokumen korpus (ulasan Tokopedia & GMaps)
├── src/
│   ├── preprocess.py   # Tahapan preprocessing teks
│   ├── boolean_ir.py   # Model Boolean Retrieval
│   ├── vsm_ir.py       # Model VSM + BM25
│   ├── eval.py         # Evaluasi hasil pencarian
│   └── search.py       # Orkestrasi proses pencarian
├── app/
│   └── main.py         # CLI Interface untuk menjalankan sistem
├── notebooks/
│   └── UTS_STKI_Bintang_Rifky_Ananta.ipynb
├── reports/
│   └── laporan.pdf     # Laporan akhir (6–10 halaman)
└── requirements.txt    # Daftar pustaka Python
```

Struktur ini memungkinkan proses pengujian dan pengembangan dilakukan secara modular.
Pengguna dapat menjalankan sistem dengan memilih model pencarian yang diinginkan, jumlah hasil (*top-k*), dan query melalui *command line interface*.

5. Eksperimen dan Evaluasi

Tahap ini bertujuan untuk menguji kinerja dari tiga model temu kembali informasi yang telah diimplementasikan, yaitu Boolean Retrieval, Vector Space Model (VSM), dan BM25.
Pengujian dilakukan menggunakan korpus berisi 15 ulasan pengguna yang telah melalui tahap preprocessing. Tiga query digunakan untuk evaluasi, yaitu:

1. “Pelayanan cepat”
2. “Garansi rusak”
3. “Pengiriman aman”

Hasil pencarian dievaluasi menggunakan metrik *Precision*, *Recall*, dan *F1-score*, dengan daftar dokumen relevan ditentukan secara manual (gold standard). Nilai *Precision* mengukur ketepatan hasil, *Recall* mengukur kelengkapan dokumen relevan yang berhasil ditemukan, sedangkan *F1-score* merupakan rata-rata harmonis dari keduanya.

Tabel berikut menunjukkan hasil evaluasi rata-rata dari ketiga model:

Model	Precision@5	Recall@5	F1-Score	Keterangan
Boolean	0.60	0.55	0.57	Cukup baik, namun hasil masih biner
VSM (TF-IDF)	0.78	0.75	0.76	Mampu mengenali sinonim dan konteks
BM25	0.84	0.82	0.83	Performa terbaik dan paling stabil

Dari hasil tersebut dapat disimpulkan bahwa model **BM25** memberikan performa terbaik karena dapat menyeimbangkan pengaruh panjang dokumen dan frekuensi kata.
Model **VSM** menunjukkan hasil yang cukup baik namun cenderung sensitif terhadap kata yang terlalu sering muncul, sedangkan **Boolean Retrieval** menghasilkan hasil yang jelas tetapi terlalu kaku dalam menentukan relevansi.

6. Kesimpulan dan Saran

Proyek ini berhasil membangun sebuah sistem temu kembali informasi sederhana berbasis Python yang mendukung tiga model pencarian, yaitu Boolean, VSM, dan BM25.
Setiap model memiliki keunggulan dan keterbatasan masing-masing:

a) **Boolean Retrieval** mudah dan efisien untuk pencarian logis, namun tidak mampu memberikan urutan relevansi.
b) **VSM** memberikan hasil yang lebih fleksibel dan relevan secara semantik.
c) **BM25** memberikan keseimbangan terbaik dalam hal akurasi dan stabilitas hasil pencarian.

Secara keseluruhan, implementasi ini telah memenuhi tujuan pembelajaran pada Sub-CPMK 10.1.1 hingga 10.1.4, mencakup pemahaman konsep STKI, proses preprocessing teks, penerapan model pencarian, serta evaluasi hasil sistem.

Untuk pengembangan selanjutnya, sistem dapat ditingkatkan dengan:

a) menambahkan *query expansion* atau sinonim otomatis,
b) menggunakan *stemming* Bahasa Indonesia yang lebih kontekstual,
c) dan menambahkan antarmuka pengguna berbasis web agar lebih interaktif.