

# **Implementasi Multi-Dataset Information Retrieval System (CLI-Based) Menggunakan Whoosh dan Vector Space Model**

Disusun untuk memenuhi  
UTS praktikum Penelusuran Informasi

Oleh :

Muhammad Syukri (2308107010060)

Halim Elsa Putra (2308107010062)

Muhammad Milan Ramadhan Mulizar (2308107010064)



**JURUSAN INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS SYIAH KUALA**

**DARUSSALAM, BANDA ACEH**

**2025**

# DAFTAR ISI

<b>Implementasi Multi-Dataset Information Retrieval System (CLI-Based) Menggunakan Whoosh dan Vector Space Model.....</b>	<b>1</b>
<b>DAFTAR ISI.....</b>	<b>2</b>
1. Pendahuluan.....	3
1.1 Latar Belakang Proyek.....	3
1.2 Tujuan Proyek.....	3
1.3 Ruang Lingkup Sistem.....	4
2. Desain Sistem dan Arsitektur.....	4
2.1 Pra-Pemrosesan Data.....	4
2.2 Arsitektur Sistem.....	4
2.3 Pembagian Modul Kode.....	6
2.4 Struktur Data Dokumen.....	7
Setiap dokumen diwakili oleh tiga kolom utama:.....	7
3. Implementasi (Kode & Penjelasan).....	7
3.1 Text Preprocessing dan Tokenisasi.....	7
3.2 Implementasi Indexing (Whoosh).....	8
3.3 Representasi Dokumen (Bag-of-Words - BoW).....	10
4. Pengujian Query dan Analisis Hasil.....	11
4.1 Skema Pengujian.....	11
4.2 Hasil Pengujian dan Output CLI.....	11
4.3 Analisis Hasil.....	13
5. Kesimpulan.....	14
5.1 Kesimpulan.....	14
5.2 Saran.....	14
Lampiran.....	15

# 1. Pendahuluan

## 1.1 Latar Belakang Proyek

Di era digital, volume data teks yang dihasilkan setiap hari sangat besar—mulai dari artikel berita, skripsi, hingga opini publik. Tantangan utama bukan lagi kekurangan informasi, tetapi bagaimana menemukan informasi yang *relevan* di antara lautan data tersebut. Di sinilah peran Information Retrieval (IR) menjadi penting, yaitu sistem yang dapat mencari dan mengurutkan dokumen berdasarkan kedekatan makna dengan kata kunci pengguna.

Namun, proses IR tidak sederhana. Data sering kali heterogen, memiliki struktur berbeda, serta mengandung banyak *stopword* dan variasi kata yang mengaburkan hasil pencarian. Karena itu, sistem ini memerlukan pendekatan yang mampu menormalisasikan teks dan mengukur kesamaan antar dokumen dengan cara yang terukur.

Pada proyek ini, digunakan pendekatan Vector Space Model (VSM) yang merepresentasikan dokumen dan query sebagai vektor kata, serta Cosine Similarity untuk menghitung tingkat kemiripan di antara keduanya. Kombinasi keduanya memungkinkan sistem memberi peringkat hasil pencarian secara akurat meskipun dokumen berasal dari berbagai sumber.

## 1.2 Tujuan Proyek

Proyek ini bertujuan untuk:

1. Memahami pipeline lengkap dari proses Information Retrieval, mulai dari preprocessing hingga ranking hasil pencarian.
2. Mengintegrasikan Vector Space Model (VSM) dan Cosine Similarity sebagai metode utama dalam sistem pencarian.
3. Melatih kolaborasi tim dalam membangun sistem IR yang modular dan dapat dikembangkan bersama.
4. Meningkatkan ketepatan pencarian pada data teks berbahasa Indonesia dengan preprocessing berbasis Sastrawi (case folding, stopwords removal, stemming).

### 1.3 Ruang Lingkup Sistem

Sistem ini sepenuhnya berbasis Command-Line Interface (CLI) sehingga dapat dijalankan langsung di terminal tanpa antarmuka grafis. Sumber data berasal dari lima dataset berbeda — *etd\_usk*, *etd\_ugm*, *kompas*, *tempo*, dan *mojok* — yang merepresentasikan variasi gaya bahasa dan konteks. Seluruh teks melalui tahap pembersihan otomatis di *preprocessing.py* dan disimpan ulang dalam folder *datasets\_clean/*. Selanjutnya, dokumen yang telah dibersihkan diindeks dengan Whoosh dan diubah menjadi representasi Bag-of-Words (BoW) menggunakan *CountVectorizer* dari *scikit-learn*. Hasil pencarian ditampilkan dalam bentuk daftar peringkat berdasarkan Cosine Similarity, yang menunjukkan seberapa dekat makna dokumen dengan query pengguna.

## 2. Desain Sistem dan Arsitektur

### 2.1 Pra-Pemrosesan Data

- **Tujuan:** Pada tahap ini melakukan standardisasi dan koreksi terhadap anomali struktural pada dataset sumber sebelum memasuki tahap text preprocessing. Langkah ini menurut kami penting dikarenakan adanya inkonsistensi delimiter dan penyertaan karakter newline internal (terutama pada *etd-ugm.csv*) yang dapat merusak integritas baris data saat dibaca oleh pandas.
- **Modul Implementasi:** Tahap ini kami implementasikan secara terpisah menggunakan file *clean\_dataset.ipynb*. File hasil perapian disimpan dalam folder ***datasets\_format/etd\_ugm.csv***.
- **Proses:** Pada notebook kami ingin memastikan semua file input memiliki struktur baris/kolom yang valid dan seragam sebelum diteruskan ke tahap preprocessing teks.

### 2.2 Arsitektur Sistem

Sistem dibangun melalui empat fase utama yang saling terhubung:

#### 1. Preprocessing

```
# --- FUNGSI PREPROCESSING ---
def preprocess_text(text):
    if pd.isna(text) or text is None:
        return ""
```

- Melakukan **case folding**, **stopword removal**, dan **stemming** dengan library Sastrawi.
- Menyimpan hasil pembersihan dalam file baru dengan nama \*\_clean.csv di folder datasets\_clean/.

## 2. Indexing

```
schema = Schema({
    doc_id=ID(stored=True),
    title=STORED,
    source=STORED,
    # Mengubah nama kolom di skema Whoosh
    konten_bersih=TEXT(stored=True)
})
```

- Setiap dokumen bersih diindeks menggunakan Whoosh, dengan kolom utama judul, konten, dan konten\_bersih.
- Proses ini menghasilkan struktur indeks di folder whoosh\_index/.

## 3. Vector Space Model (VSM)

```
# Membuat Term-Document Matrix (Matriks BoW)
doc_term_matrix = vectorizer.fit_transform(doc_contents)
```

- Sistem mengubah seluruh teks menjadi representasi Bag-of-Words (BoW) dengan *CountVectorizer* untuk membentuk Term-Document Matrix. Matriks ini menjadi dasar untuk menghitung kesamaan antar dokumen.

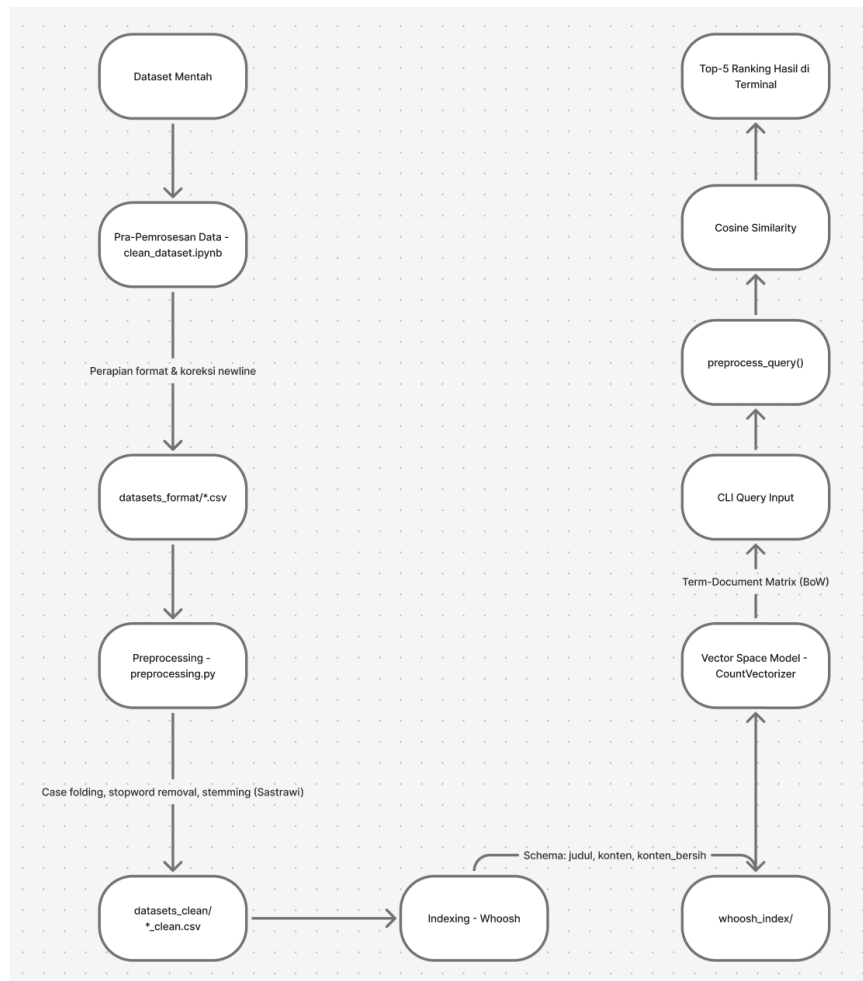
## 4. Search & Ranking

```
# Menghitung kemiripan antara query (1 vektor) dan semua dokumen (banyak vektor)
cosine_similarities = cosine_similarity(query_vector, doc_term_matrix).flatten()
```

- Saat pengguna memasukkan query di CLI, teks query diproses ulang menggunakan fungsi preprocess\_query().
- Query tersebut dibandingkan dengan seluruh dokumen menggunakan Cosine Similarity untuk menghasilkan skor relevansi.

- Hasil akhir ditampilkan dalam bentuk *Top 5 dokumen* dengan skor tertinggi.

Diagram alur sederhananya:



## 2.3 Pembagian Modul Kode

- **Modul 1 – preprocessing.py**

Menangani seluruh tahap pembersihan teks: *case folding*, *stopword removal*, *stemming*, dan penyimpanan hasil ke `datasets_clean/`.

Fungsi utama: `preprocess_text()` dan `process_and_save_dataset()`.

- **Modul 2 – main.py (IR System)**

Menjalankan proses indexing, pembuatan VSM, serta CLI untuk pencarian.

Fungsi utama: `load_and_index_dataset()` dan `search_and_rank()`.

- **Modul tambahan – clean\_dataset.ipynb**

Digunakan hanya sekali untuk memperbaiki format CSV mentah yang tidak konsisten (misal koma ganda atau kutipan ganda).

## 2.4 Struktur Data Dokumen

Setiap dokumen diwakili oleh tiga kolom utama:

Kolom	Deskripsi
judul	Judul teks asli dari sumber dataset
konten	Isi teks mentah sebelum diproses
konten_bersih	Versi teks yang telah melalui <i>case folding</i> , <i>stopword removal</i> , dan <i>stemming</i> .

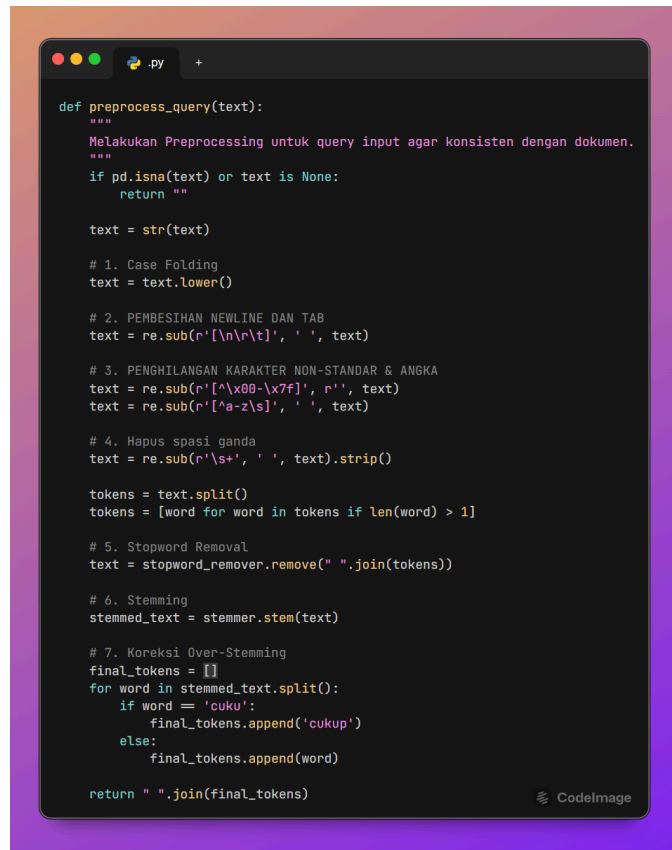
## 3. Implementasi (Kode & Penjelasan)

Bagian ini merinci implementasi teknis dari fungsi-fungsi utama dalam sistem.

### 3.1 Text Preprocessing dan Tokenisasi

Untuk memastikan konsistensi antara dokumen yang diindeks dan query pencarian, fungsi preprocessing yang identik digunakan di preprocessing.py (untuk dokumen) dan main.py (untuk query). Fungsi yang digunakan di main.py untuk memproses query pengguna adalah preprocess\_query(text).

Snippet Kode (main.py):

A screenshot of a code editor window with a dark background and a purple-to-orange gradient border. The editor shows a Python function named `preprocess_query(text)`. The function includes several steps: converting text to a string, lowercasing it, removing newlines and tabs, removing non-ASCII characters, digits, and symbols, stripping extra spaces, splitting into tokens, removing stopwords, stemming, and a manual correction for the word 'cukup' to 'cuku'. The function returns the joined final tokens. The code is written in a syntax-highlighted style with comments in Indonesian.

```
def preprocess_query(text):  
    """  
    Melakukan Preprocessing untuk query input agar konsisten dengan dokumen.  
    """  
    if pd.isna(text) or text is None:  
        return ""  
  
    text = str(text)  
  
    # 1. Case Folding  
    text = text.lower()  
  
    # 2. PEMBERSIHAN NEWLINE DAN TAB  
    text = re.sub(r'[\n\r\t]', ' ', text)  
  
    # 3. PENGHILANGAN KARAKTER NON-STANDAR & ANGKA  
    text = re.sub(r'^\x00-\x7f', '', text)  
    text = re.sub(r'^a-z\s]', ' ', text)  
  
    # 4. Hapus spasi ganda  
    text = re.sub(r'\s+', ' ', text).strip()  
  
    tokens = text.split()  
    tokens = [word for word in tokens if len(word) > 1]  
  
    # 5. Stopword Removal  
    text = stopwords_remove.remove(" ".join(tokens))  
  
    # 6. Stemming  
    stemmed_text = stemmer.stem(text)  
  
    # 7. Koreksi Over-Stemming  
    final_tokens = []  
    for word in stemmed_text.split():  
        if word == 'cuku':  
            final_tokens.append('cukup')  
        else:  
            final_tokens.append(word)  
  
    return " ".join(final_tokens)
```

### Langkah Detail:

1. *Case folding* dan pembersihan anomali, dimana teks diubah menjadi huruf kecil (`text.lower()`). Karakter anomali (non-ASCII), angka, dan simbol (selain spasi) dihilangkan menggunakan regular expressions (`re.sub(r'^a-z\s]', ' ', text)`).
2. *Stopword removal*, dimana kata-kata umum yang tidak memiliki makna (seperti 'yang', 'di', 'ke') dihapus menggunakan **library Sastrawi** (`stopword_remove.remove(...)`).
3. *Stemming*, dimana kata-kata diubah ke bentuk dasarnya menggunakan Sastrawi (`stemmer.stem(text)`).
4. Koreksi *Stemming*, dimana pada implementasi ini kami juga membuat koreksi manual untuk *over-stemming* yang umumnya terjadi di Sastrawi, di mana saat kami stemming kata 'cukup' keliru diubah menjadi 'cuku'. Sehingga kode ini secara eksplisit mengembalikan 'cuku' menjadi 'cukup'.

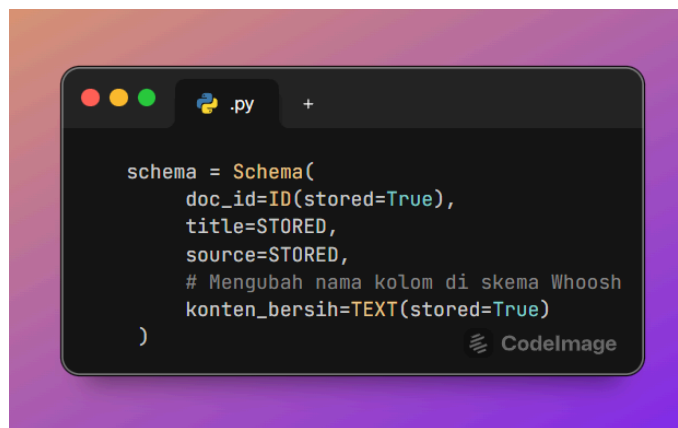
## 3.2 Implementasi Indexing (Whoosh)



Indexing menggunakan Whoosh dilakukan di dalam fungsi `load_and_index_dataset` di `main.py`.

- **Definisi Schema**

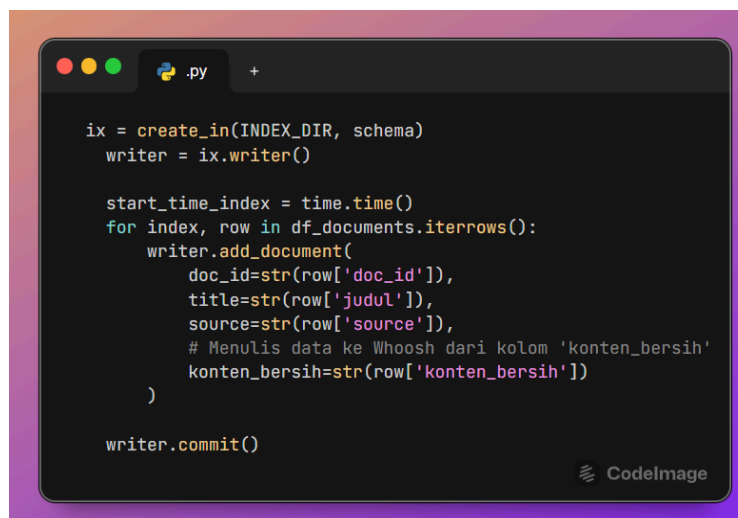
Schema didefinisikan untuk memberi tahu Whoosh cara menyimpan dan mengindeks data. Kolom `konten_bersih` secara spesifik didefinisikan sebagai `TEXT(stored=True)`, yang berarti konten tersebut akan dianalisis (ditokenisasi) oleh Whoosh dan juga disimpan secara utuh di dalam index.



```
schema = Schema(
    doc_id=ID(stored=True),
    title=STORED,
    source=STORED,
    # Mengubah nama kolom di skema Whoosh
    konten_bersih=TEXT(stored=True)
)
```

- **Alur Penulisan Dokumen**

Setelah index dibuat (`create_in`), proses penulisan dilakukan dengan mengiterasi setiap baris dari DataFrame gabungan (`df_documents`) dan menambahkannya ke writer Whoosh.



```
ix = create_in(INDEX_DIR, schema)
writer = ix.writer()

start_time_index = time.time()
for index, row in df_documents.iterrows():
    writer.add_document(
        doc_id=str(row['doc_id']),
        title=str(row['judul']),
        source=str(row['source']),
        # Menulis data ke Whoosh dari kolom 'konten_bersih'
        konten_bersih=str(row['konten_bersih'])
    )

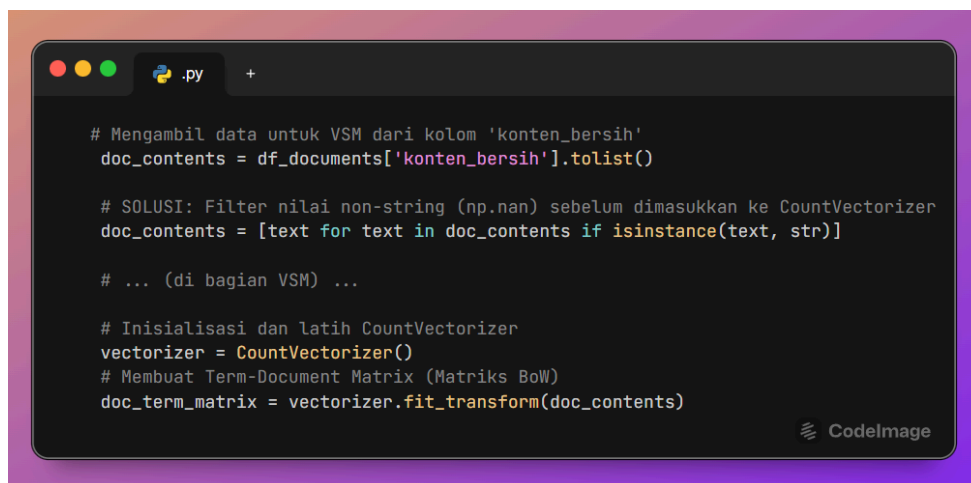
writer.commit()
```

### 3.3 Representasi Dokumen (Bag-of-Words - BoW)

Representasi VSM (BoW) dibuat menggunakan **CountVectorizer** dari library Scikit-learn. Proses ini juga terjadi di dalam fungsi `load_and_index_dataset`.

- **Implementasi**

Pertama, semua data dari kolom `konten_bersih` dikumpulkan ke dalam satu list bernama `doc_contents`. List ini kemudian dijadikan input untuk `CountVectorizer`.



```
# Mengambil data untuk VSM dari kolom 'konten_bersih'
doc_contents = df_documents['konten_bersih'].tolist()

# SOLUSI: Filter nilai non-string (np.nan) sebelum dimasukkan ke CountVectorizer
doc_contents = [text for text in doc_contents if isinstance(text, str)]

# ... (di bagian VSM) ...

# Inisialisasi dan latih CountVectorizer
vectorizer = CountVectorizer()
# Membuat Term-Document Matrix (Matriks BoW)
doc_term_matrix = vectorizer.fit_transform(doc_contents)
```

Metode `fit_transform` diterapkan pada `doc_contents` (gabungan semua dokumen bersih). Metode ini melakukan dua hal:

1. **fit**, yaitu mempelajari seluruh kosakata (vocabulary) dari korpus dan memberi index unik untuk setiap term (kata).
2. **transform**, yaitu mengubah setiap dokumen teks menjadi vektor angka, di mana setiap angka mewakili frekuensi kemunculan (term frequency) dari term tersebut dalam dokumen.

Hasilnya adalah Matriks BoW (`doc_term_matrix`) tunggal yang menyimpan representasi VSM untuk seluruh koleksi dokumen. Ukuran Matriks BoW yang dihasilkan dicetak ke konsol oleh skrip `main.py` setelah proses selesai. Ukuran ini menunjukkan jumlah total dokumen dan jumlah term unik yang ditemukan.

## 4. Pengujian Query dan Analisis Hasil

Bagian ini bertujuan untuk memverifikasi fungsionalitas utama sistem dalam melakukan pencarian (*search*) dan pengurutan (*ranking*) dokumen menggunakan metrik Cosine Similarity.

### 4.1 Skema Pengujian

No.	Query Pengujian	Tujuan Analisis
1.	berkendara	Menguji kepadatan query kata tunggal yang diasumsikan memiliki frekuensi rendah di corpus akademik.
2.	dampak kesehatan mahasiswa berkendara	Menguji efektivitas stemming (dampak → dampak, kesehatan → sehat, kendaraan → kendara ) dalam menemukan dokumen dengan kata kunci berafiks yang sama.
3.	politik identitas dan polarisasi media	Menguji kemampuan ranking dalam memprioritaskan dokumen dari dataset dengan topik sosial-politik.

### 4.2 Hasil Pengujian dan Output CLI

Untuk memberikan bukti fungsionalitas dan konsistensi sistem yang telah kami buat, hasil dari tiga skema pengujian akan disajikan berikut ini:

#### a. Output CLI Utama (Query 1: Berkendara)

Tujuan untuk menunjukkan hasil *ranking* dengan *query* yang bersifat spesifik (kata tunggal).

```

=====
=== INFORMATION RETRIEVAL SYSTEM ===
Status: Siap Mencari
=====
[1] Load & Index Dataset
[2] Search Query
[3] Exit
=====
Pilih menu [1/2/3]: 2
Masukkan Query Pencarian Anda: berkendara

Ditemukan 3740 dokumen relevan (dari 47432 total).
Ranking selesai dalam 0.5032 detik.

=== TOP 5 HASIL PENCARIAN (Cosine Similarity) ===
[1] Skor: 0.6108 | Judul: 155 Ribu Kendaraan Tinggalkan Jabodetabek saat Libur Panjang (TEMPO)
[2] Skor: 0.6036 | Judul: Kelakuan Norak Pengendara di Jalanan Berdasarkan Jenis Kendaraan, Mulai dari Astrea Hingga Mercy (MOJOK)
[3] Skor: 0.6034 | Judul: Libur Panjang Waisak, Volume Lalin di Tol Jabodetabek dan Jawa Barat Meningkat (KOMPAS)
[4] Skor: 0.5976 | Judul: 3 Alasan Motor Matic Lebih Populer di Kalangan Perempuan (MOJOK)
[5] Skor: 0.5880 | Judul: Tipe Kepribadian Orang dalam Berkendara Motor yang Berseliweran di Hidup Kita (MOJOK)
=====
Query Bersih: kendara
=====

```

## b. Output CLI Utama (Query 2: Dampak Kesehatan)

Tujuan untuk menunjukkan kinerja *stemming* dan ranking pada *query* kesehatan dari dataset yang ada.

```

=====
=== INFORMATION RETRIEVAL SYSTEM ===
Status: Siap Mencari
=====
[1] Load & Index Dataset
[2] Search Query
[3] Exit
=====
Pilih menu [1/2/3]: 2
Masukkan Query Pencarian Anda: dampak kesehatan mahasiswa berkendara

Ditemukan 16126 dokumen relevan (dari 47432 total).
Ranking selesai dalam 0.4867 detik.

=== TOP 5 HASIL PENCARIAN (Cosine Similarity) ===
[1] Skor: 0.4500 | Judul: Penilaian Literasi Kesehatan Mental dan Faktor-Faktor Kesehatan Mental Pada Mahasiswa (ETD_UGM)
[2] Skor: 0.4282 | Judul: PENGGUNAAN MEDIA SOSIAL INSTAGRAM DALAM MENINGKATKAN LITERASI KESEHATAN PADA MAHASISWA UNIVERSITAS SYIAH KUALA (ETD_USK)
[3] Skor: 0.3640 | Judul: Promosi Kesehatan: Jurusan Underrated yang Dianggap Cuma Sales, padahal Garda Terdepan Kesehatan Rakyat (MOJOK)
[4] Skor: 0.3512 | Judul: Exploration Of Academic Anxiety Among Medical Students During the COVID-19 Online Learning (ETD_UGM)
[5] Skor: 0.3507 | Judul: PERBEDAAN LITERASI KESEHATAN MENTAL PADA MAHASISWA KESEHATAN DAN NON-KESEHATAN DI BANDARA ACEH (ETD_USK)
=====
Query Bersih: dampak sehat mahasiswa kendara
=====

```

## c. Output CLI Utama (Query 3: Politik Identitas)

Tujuan untuk menunjukkan kemampuan sistem memprioritaskan data dari dataset opini/berita (non-akademik).

```

=====
=== INFORMATION RETRIEVAL SYSTEM ===
Status: Siap Mencari
=====
[1] Load & Index Dataset
[2] Search Query
[3] Exit
=====
Pilih menu [1/2/3]: 2
Masukkan Query Pencarian Anda: politik identitas dan polarisasi media

Ditemukan 8529 dokumen relevan (dari 47432 total).
Ranking selesai dalam 0.4346 detik.

=== TOP 5 HASIL PENCARIAN (Cosine Similarity) ===
[1] Skor: 0.4356 | Judul: PERAN MEDIA SOSIAL SEBAGAI SARANA SOSIALISASI POLITIK DALAM MENINGKATKAN KESADARAN
POLITIK GENERASI Z (STUDI KASUS MAHASISWA UNIVERSITAS SYIAH KUALA) (ETD_USK)
[2] Skor: 0.4348 | Judul: Analisis Kemitraan Pemerintah Swasta Dalam Pengelolaan Sampah di Kota Pekanbaru (ET
D_UGM)
[3] Skor: 0.3350 | Judul: PENGARUH IKLAN POLITIK DI MEDIA SOSIAL TERHADAP PERILAKU PEMILIH MASYARAKAT KOTA ME
DAN DALAM PEMENANGAN BOBBY NASUTION -AULIA RACHMAN DI PILKADA TAHUN 2021 (ETD_USK)
[4] Skor: 0.3291 | Judul: Strategi Kampanye Partai Kebangkitan Bangsa Melalui Media Sosial Facebook Pada Pemi
lu 2019 Dan Implikasi Terhadap Ketahanan Partai Politik (Studi Pada Media Sosial Facebook Dewan Pengurus Wila
yah PKB DIY) (ETD_UGM)
[5] Skor: 0.3267 | Judul: PENGEMBANGAN MEDIA TELUR MISTERI PADA PEMBELAJARAN SEJARAH (ETD_USK)
=====
Query Bersih: politik identitas polarisasi media
=====

```

### 4.3 Analisis Hasil

1. **Analisis Peran Cosine Similarity**, yaitu skor yang ditampilkan merupakan nilai Cosine Similarity yang mengukur cosinus dari sudut antara vektor query dan vektor dokumen dalam ruang vektor (Term-Document Matrix). Skor yang mendekati 1 akan mengindikasikan bahwa vektor memiliki orientasi yang sangat mirip, yang berarti term-term kunci yang terkandung dalam query memiliki kepadatan frekuensi yang tinggi dalam dokumen terkait.
2. **Analisis Efektivitas Preprocessing**, dengan efektivitas ranking ini didukung oleh stemming yang diterapkan pada text preprocessing. Misalnya pada query 2 yang kami coba, kata kunci kesehatan diubah menjadi kata dasar **sehat** pada query. Hasilnya, dokumen yang mengandung kata sehat, kesehatan, maupun term-term lainnya dengan akar yang sama, dikelompokkan bersama dan mendapatkan skor yang setara. Hal ini menghasilkan recall yang lebih tinggi membuat sistem menemukan dokumen yang relevan dari semua jenis dataset yang berbeda-beda.
3. **Analisis Kinerja dan Multi-Dataset**, pada sistem berhasil mengindeks total **47.432 dokumen** dari lima sumber yang berbeda. Waktu pencarian yang cepat ada di bawah 1 detik untuk semua query. Ini membuktikan bahwa :
  - a) Indexing Whoosh berfungsi sebagai penampung metadata yang cepat.

- b) Implementasi CountVectorizer dan perhitungan Cosine Similarity menggunakan scikit-learn dieksekusi secara efisien pada komputasi matriks skala besar.

Hasil ranking yang mencakup lima sumber berbeda (ETD-UGM, KOMPAS, TEMPO, ETD-USK, MOJOK) secara langsung memverifikasi bahwa proyek Multi-Dataset Information Retrieval System ini telah berhasil diimplementasikan.

## 5. Kesimpulan

### 5.1 Kesimpulan

Sistem *Multi-Dataset Information Retrieval* berbasis CLI ini telah berhasil diimplementasikan sesuai dengan arahan tugas UTS. Berjalannya kode program ini karena menerapkan pipeline yang terstruktur yang dimulai dari *preprocessing* untuk menjamin konsistensi term, *Indexing* untuk efisiensi penyimpanan, dan *Vector Space Model* (BoW & Cosine Similarity) untuk *ranking* relevansi dokumen secara akurat. Pengujian menunjukkan sistem mampu mengidentifikasi term kunci dan memberikan hasil ranking yang relevan dari *corpus* gabungan (tesis, berita, dan opini) dalam waktu eksekusi yang cepat.

### 5.2 Saran

Untuk pengembangan sistem lebih lanjut, kami sekaligus menyarankan dan dapat untuk mengimplementasikan model *Vector Space* berbasis TF-IDF (Term Frequency-Inverse Document Frequency) sebagai pengganti BoW. Hal ini karena penggunaan TF-IDF akan memberikan bobot yang lebih tinggi pada term spesifik dan jarang muncul, yang berpotensi meningkatkan akurasi ranking secara keseluruhan.

## **Lampiran**

Link Github : [https://github.com/muhammadsyukri19/10-UTS\\_Praktikum\\_PI.git](https://github.com/muhammadsyukri19/10-UTS_Praktikum_PI.git)