Laporan Praktikum Penelusuran Informasi

UTS PRAKTIKUM PENELUSURAN INFORMASI

disusun untuk memenuhi tugas

praktikum Penelusuran Informasi

Oleh :

Al- Mahfuzh Fadhlur Rohman 2208107010016

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

DARUSSALAM, BANDA ACEH

2025

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Pesatnya pertumbuhan informasi digital di era modern telah menciptakan tantangan signifikan dalam hal aksesibilitas dan relevansi. Seiring bertambahnya volume data teks dari berbagai sumber, kebutuhan akan sistem yang efisien dan akurat untuk menelusuri, menemukan, dan mengambil dokumen yang relevan menjadi semakin mendesak. Inilah peran fundamental dari sistem Information Retrieval (IR). Sistem IR berfungsi sebagai jembatan yang menghubungkan kebutuhan informasi pengguna (dinyatakan melalui query) dengan koleksi dokumen yang luas.

Sistem IR yang akan dikembangkan dalam proyek UTS Praktikum Penelusuran Informasi ini bertujuan untuk mengatasi tantangan tersebut dengan membangun sebuah Sistem Information Retrieval berbasis Command-Line Interface (CLI). Sistem ini dirancang secara khusus agar mampu melakukan proses pencarian dan ranking dokumen dari berbagai sumber teks nyata (multi-dataset) secara terintegrasi.

Proyek ini mewajibkan penggunaan lima dataset utama yang heterogen, merepresentasikan keragaman jenis dokumen yang ada di dunia nyata. Dataset tersebut meliputi dokumen-dokumen akademik (tesis/disertasi dari etd-usk dan etd-ugm), berita harian nasional (kompas), majalah berita dan opini (tempo), serta artikel populer dan satir (mojok). Menggabungkan dan mengelola data dari sumber yang beragam ini memerlukan pemahaman mendalam tentang teknik preprocessing teks yang robust.

Inti dari sistem yang akan dibangun adalah implementasi dari Vector Space Model (VSM). VSM menjadi kerangka kerja matematis untuk merepresentasikan dokumen dan query sebagai vektor dalam ruang multidimensi. Untuk mencapai representasi ini, setiap tim bertanggung jawab untuk melakukan serangkaian langkah kunci, dimulai dari Preprocessing dan tokenisasi teks (minimal case folding, tokenization, dan stopword removal).

Setelah preprocessing, dokumen direpresentasikan menggunakan model Bag of Words (BoW). Representasi ini kemudian digunakan untuk pembentukan index dokumen dengan memanfaatkan library Whoosh. Penggunaan Whoosh memungkinkan sistem untuk melakukan pencarian yang efisien berdasarkan query pengguna.

Tahap krusial berikutnya adalah proses ranking hasil pencarian. Dalam sistem ini, ranking hasil dilakukan berdasarkan perhitungan cosine similarity antara vektor query dan vektor dokumen. Cosine similarity dipilih karena kemampuannya

dalam mengukur kemiripan arah antar vektor, yang sangat efektif untuk menentukan relevansi dokumen terlepas dari panjangnya. Hasil akhir yang disajikan kepada pengguna adalah 5 dokumen teratas dengan skor kemiripan tertinggi.

Dengan mengintegrasikan seluruh tahapan ini—dari preprocessing, representasi BoW, indexing Whoosh, hingga ranking Cosine Similarity—proyek ini secara langsung memenuhi tujuan untuk memahami pipeline lengkap sistem penelusuran informasi dan mengintegrasikan konsep Vector Space Model ke dalam sistem nyata. Proyek ini juga sekaligus melatih kemampuan teknis dalam bahasa Python dan library pendukung seperti scikit-learn dan Whoosh.

1.2.Tujuan Proyek

Proyek Pembangunan sistem Penelusuran Informasi ini memiliki beberapa tujuan spesifik, yaitu :

- **Memahami *Pipeline* IR Lengkap:** Memahami dan mengimplementasikan seluruh tahapan dalam sistem penelusuran informasi, mulai dari *preprocessing* hingga *ranking*.

- **Implementasi VSM:** Mengintegrasikan konsep **Vector Space Model** ke dalam sistem nyata untuk representasi dokumen dan *query*.

- **Peningkatan Akurasi Pencarian:** Meningkatkan ketepatan pencarian dengan pendekatan representasi teks yang efisien.

- **Pengembangan Sistem IR:** Melatih kemampuan kolaboratif dalam pengembangan sistem IR berbasis **Python** dengan antarmuka **CLI** interaktif.

1.3.Ruang Lingkup Sistem

Sistem IR yang akan dikembangkan dalam proyek ini harus mampu melakukan fungsi-fungsi utama berikut:

1. **Preprocessing dan Tokenisasi Teks:** Melakukan minimal *case folding*, *tokenization*, dan *stopword removal* (dengan *stemming/lemmatization* sebagai opsional).
2. **Representasi Dokumen (BoW):** Menggunakan model **Bag of Words** (BoW) untuk merepresentasikan dokumen dan *query*.
3. **Pembentukan *Index* Dokumen (*Whoosh*):** Menggunakan *library* **Whoosh** untuk proses pengindeksan dan pencarian dokumen.
4. **Pencarian dan *Ranking*:** Mampu menangani *query* pengguna, melakukan pencarian, dan menampilkan **5 dokumen teratas** dengan skor tertinggi berdasarkan **Cosine Similarity**.

5. **Antarmuka CLI:** Menyediakan antarmuka *Command-Line Interface* interaktif dengan menu minimal *Load & Index Dataset*, *Search Query*, dan *Exit*.
6. **Penggunaan *Library* Terbatas:** Hanya diperbolehkan menggunakan *library* standar Python, **scikit-learn** (untuk *CountVectorizer* dan *cosine similarity*), **Whoosh**, dan **pandas**.

BAB II
DESAIN SISTEM DAN ARSITEKTUR

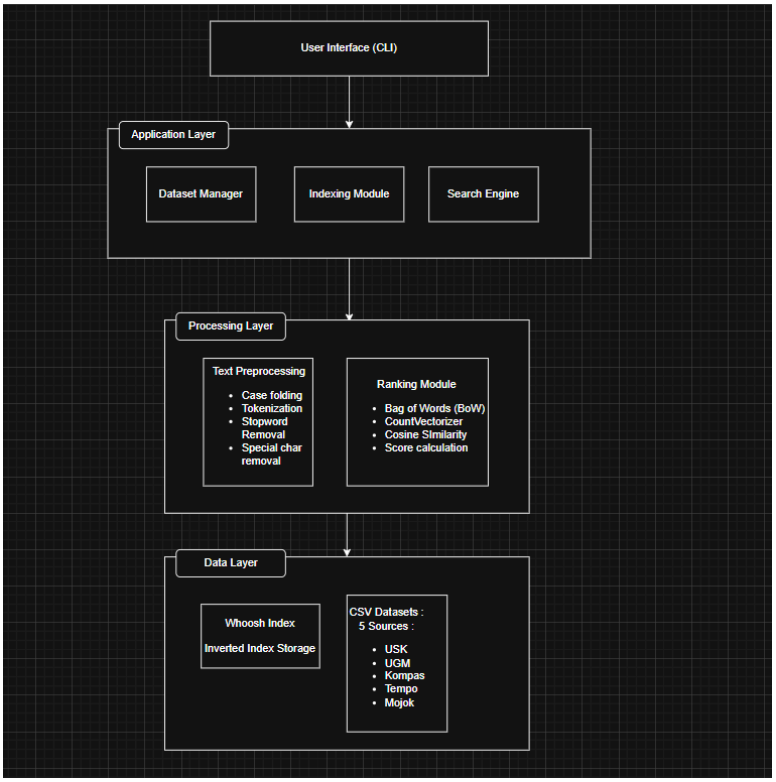
2.1. Gambaran Umum Sistem

Sistem Penelusuran Informasi ini dirancang untuk melakukan pencarian dokumen teks dari berbagai sumber (dataset CSV seperti *etd_usk, etd_ugm,* kompas,mojok, dan tempo).

Proses utama sistem meliputi :

- Pemuatan Dataset dari file CSV.
- Prapemrosesan Teks, meliputi case folding, pembersihan karakter, tokenisasi, dan stopword removal.
- Pembuatan Index menggunakan Whoosh, agar pencarian dokumen menjadi efisien.
- Representasi Dokumen menggunakan model Bag of Words (BoW) dengan CountVectorizer.
- Perhitungan Similarity menggunakan Cosine Similarity antara query dan dokumen kandidat.
- Penentuan Ranking Hasil berdasarkan tingkat kemiripan.
- Tampilan Hasil Pencarian yang berisi judul, sumber, skor relevansi, dan cuplikan konten.

2.2. High-level Architecture



2.3. Desain Alur Sistem

A. Alur Proses Indexing

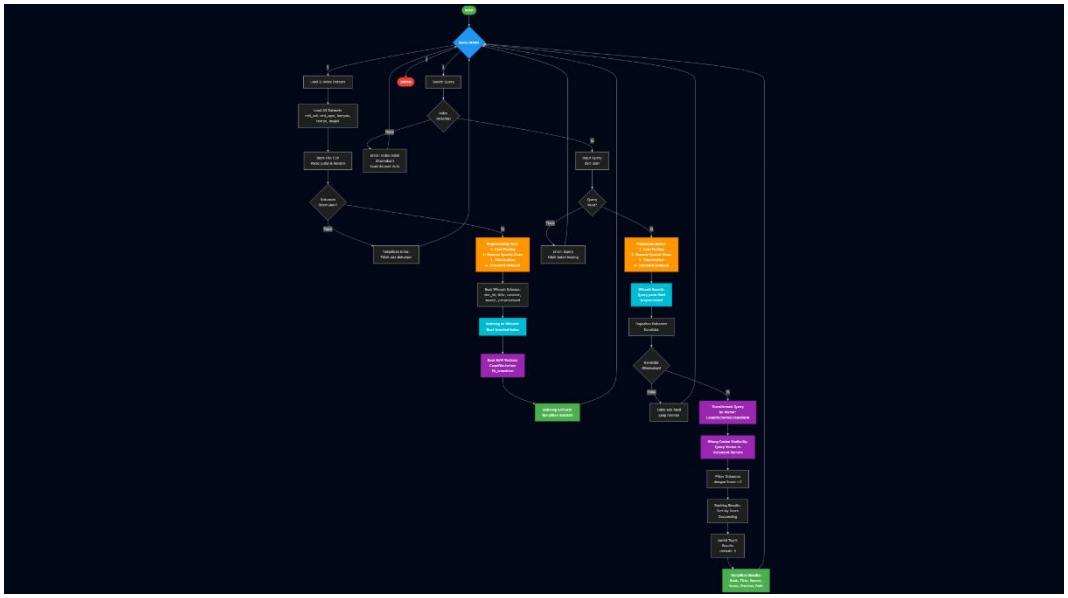
- User memilih menu ‘Load & Index Dataset’
- Sistem membaca 5 file CSV dari folder dataset/
- Setiap dokumen di parse (judul & konten)
- Text preprocessing dilakukan pada setiap dokumen
- Whoosh index dibuat dengan Schema yang ditentukan
- Dokumen diindeks ke Whoosh Index
- Bag of Words vectors dibuat menggunakan CountVectorizer
- Sistem siap untuk pencarian

B. Alur Proses Searching

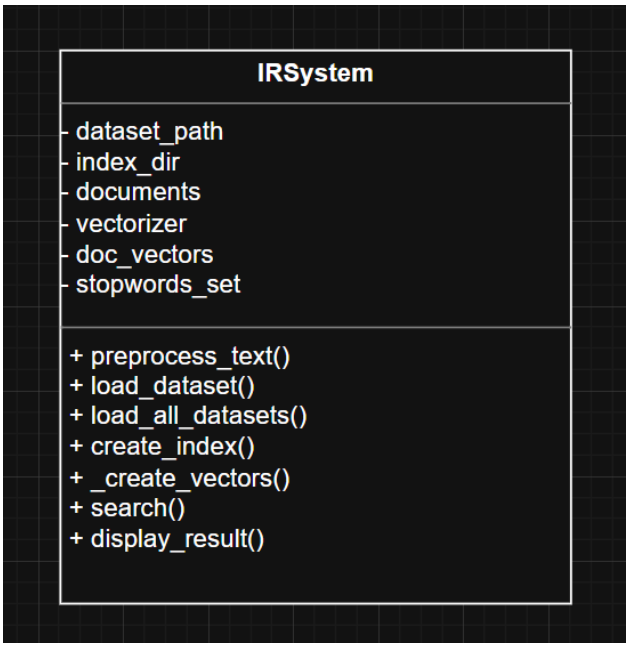
- User memasukkan query pencarian
- Query di preprocess dengan *case folding, tokenization dan stopword removal*
- Whoosh melakukan pencarian awal pada inverted index
- Dokumen kandidat dikembalikan oleh Whoosh
- Query di transformasi menjadi vector mengguakan CountVectorizer
- Cosine similarity dihitung antara query vector dan document vectors

- Hasil di ranking berdasarkan similarity score
- Top-K hasil ditampilkan ke user dengan score dan preview

Alur proses sistem divisualisasikan dengan flowchart berikut :



2.4. Class Design



BAB III
IMPLEMENTASI

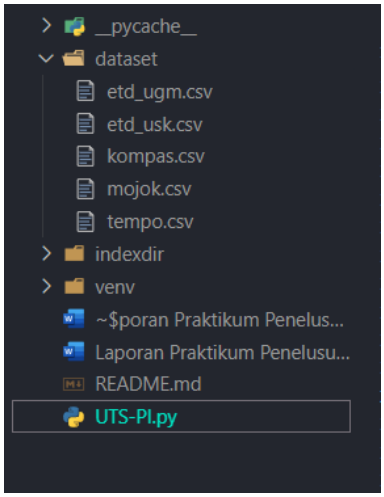
3.1. Lingkungan Pengembangan

Sistem Penelusuran Informasi ini dikembangkan dengan Python 3.12.2 dengan beberapa *library* pendukung. Lingkungan pengembangan yang digunakan :

Komponen	**Keterangan**
Bahasa Pemrograman	Python
IDE / Editor	Visual Studio Code
Library	Whoosh, scikit-learn, pandas
Sistem Operasi	Windows 11
Dataset	5 file CSV, yaitu : *etd_usk.csv, etd_ugm.csv,kompas.csv,mojok.csv,* dan *tempo.csv*

3.2. Struktur Direktori Program

Struktur direktori sistem ini adalah sebagai berikut :



3.3. Kelas Utama Sistem (IRSystem)

Implementasi inti sistem dilakukan dalam satu kelas bernama IRSystem. Kelas ini mengatur semua proses, mulai dari loading dataset*, preprocessing, indexing,* hingga *searching* dan *ranking* hasil pencarian

```python
class IRSystem:
    def __init__(self, dataset_path="dataset", index_dir="indexdir"):
        self.dataset_path = dataset_path
        self.index_dir = index_dir
        self.documents = []
        self.vectorizer = CountVectorizer()
        self.doc_vectors = None
        self.stopwords = set([
            'yang', 'untuk', 'pada', 'ke', 'para', 'namun', 'menurut',
            'antara', 'dengan', 'dari', 'adalah', 'ini', 'itu', 'akan',
            'atau', 'dan', 'di', 'dalam', 'oleh', 'juga', 'ada', 'dapat',
            'telah', 'tersebut', 'seperti', 'tidak', 'lebih', 'karena',
            'sudah', 'saat', 'bisa', 'bila', 'jika', 'maka', 'serta', 'sebagai'
        ])

```

Penjelasan :

- Konstruktor inisialisasi direktori dataset dan index
- Membuat objek CountVectorizer() untuk representasi *Bag of Words* (BoW)
- Mendefinisikan daftar *stopword* Bahasa Indonesia untuk proses pembersihan teks.

3.4.Preprocessing Text

Tahapan *preprocessing* sangat penting agar sistem hanya bekerja dengan kata bermakna. Tahapan ini meliputi *case folding, tokenization*, pembersihan karakter dan *stopword removal.*

```python
def preprocess_text(self, text):
        """
        Text preprocessing:
        - Case folding
        - Tokenization
        - Stopword removal
        """
        # Case folding
        text = text.lower()

        # Hapus karakter spesial dan angka
        text = re.sub(r'[^a-z\s]', ' ', text)

        # Tokenisasi
        tokens = text.split()

        # Penghapusan stopword
        tokens = [token for token in tokens if token not in self.stopwords and len(token) > 2]

        return ' '.join(tokens)
```

3.5.Pemanggilan Dataset

Sistem membaca lima dataset CSV dari folder dataset/ menggunakan pandas. Setiap baris pada dataset dianggap sebagai satu dokumen dengan atribut judul dan konten.

```python
def load_dataset(self, dataset_name):
        """Load documents from a specific CSV dataset file"""
        dataset_file = os.path.join(self.dataset_path, f"{dataset_name}.csv")
        docs = []

        if not os.path.exists(dataset_file):
            print(f"Warning: Dataset file '{dataset_file}' not found!")
            return docs

        print(f"Loading dataset: {dataset_name}...")

        try:
            df = pd.read_csv(dataset_file, encoding='utf-8')
            for idx, row in df.iterrows():
                title = str(row['judul']).strip()
                content = str(row['konten']).strip()
                if content:
                    docs.append({
                        'id': f"{dataset_name}_{idx}",
                        'title': title,
                        'content': content,
                        'source': dataset_name,
                        'path': dataset_file
                    })
        except Exception as e:
            print(f"Error reading {dataset_file}: {e}")

        print(f"Loaded {len(docs)} documents from {dataset_name}")
        return docs

    def load_all_datasets(self):
        """Load all five required datasets"""
        datasets = ['etd_usk', 'etd_ugm', 'kompas', 'tempo', 'mojok']
        self.documents = []

        print("\n=== Loading All Datasets ===")
        for dataset in datasets:
            docs = self.load_dataset(dataset)
            self.documents.extend(docs)

        print(f"\nTotal documents loaded: {len(self.documents)}")
        return len(self.documents)
```

Penjelasan :

- Membuka file dataset .csv dengan *pandas*
- Setiap dokumen diberi ID unik
- Dokumen disimpan dalam list self.documents untuk diproses lebih lanjut

3.6.Indexing dengan Whoosh

Index digunakan agar sistem pencarian lebih efisien. Setiap dokumen disimpan dengan schema yang berisi doc_id, title, content, source, dan teks hasil preprocessing.

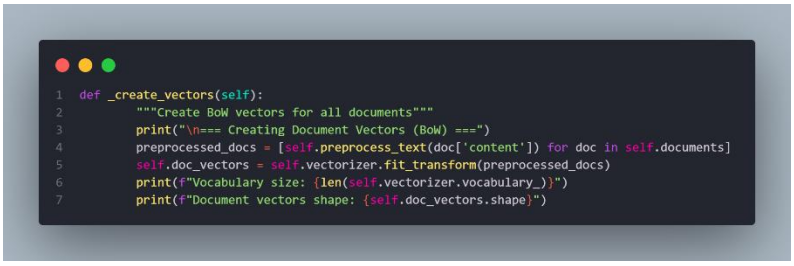
```python
def create_index(self):
        """Create Whoosh index for all documents"""
        if not self.documents:
            print("No documents to index!")
            return False

        print("\n=== Creating Index ===")

        # Membuat direktori index jika tidak ada
        if not os.path.exists(self.index_dir):
            os.makedirs(self.index_dir)

        # Definisikan schema
        schema = Schema(
            doc_id=ID(stored=True, unique=True),
            title=TEXT(stored=True),
            content=TEXT(stored=True),
            source=TEXT(stored=True),
            preprocessed=TEXT(stored=True)
        )

        # Membuat atau membuat ulang index
        ix = create_in(self.index_dir, schema)
        writer = ix.writer()

        print("Indexing documents...")
        for i, doc in enumerate(self.documents):
            preprocessed = self.preprocess_text(doc['content'])
            writer.add_document(
                doc_id=doc['id'],
                title=doc['title'],
                content=doc['content'],
                source=doc['source'],
                preprocessed=preprocessed
            )
            if (i + 1) % 100 == 0:
                print(f"Indexed {i + 1}/{len(self.documents)} documents")

        writer.commit()
        print(f"Indexing complete! Total: {len(self.documents)} documents")

        # Membuat vektor dokumen untuk cosine similarity
        self._create_vectors()
```

Penjelasan :

- Membuat folder indexdir jika belum ada
- Menggunakan Whoosh Schema untuk mendefinisikan struktur dokumen
- Setiap dokumen di preprocess dan ditambahkan ke index
- Setelah semua dokumen diindex, sistem melanjutkan ke tahap pembuatan vector dokumen

3.7.Representasi dengan CountVectorizer

Setelah indexing, sistem membentuk representasi numerik menggunakan *Bag of Words* (BoW).



```python
def _create_vectors(self):
        """Create BoW vectors for all documents"""
        print("\n=== Creating Document Vectors (BoW) ===")
        preprocessed_docs = [self.preprocess_text(doc['content']) for doc in self.documents]
        self.doc_vectors = self.vectorizer.fit_transform(preprocessed_docs)
        print(f"Vocabulary size: {len(self.vectorizer.vocabulary_)}")
        print(f"Document vectors shape: {self.doc_vectors.shape}")
```

Penjelasan :

- Mengubah setiap dokumen menjadi vector frekuensi kata
- Hasil self.doc_vectors adalah *matrix-term document* untuk perhitungan similarity

3.8.Proses Pencarian dan Ranking

Pada tahap ini, sistem menerima input dari pengguna, lalu :

1. Memproses query dengan cara yang sama seperti dokumen
2. Mencari kandidat dokumen menggunakan Whoosh
3. Menghitung *cosine similarity* antara query dan dokumen kandidat
4. Mengurutkan hasil berdasarkan skor tertinggi

```python
def search(self, query_text, top_k=5):
    """
    Mencari dokumen dengan Whoosh dan me ranking menggunakan cosine similarity
    """
    if not exists_in(self.index_dir):
        print("Index tidak ditemukan! Silakan muat dan indeks dataset terlebih dahulu.")
        return []

    if self.doc_vectors is None:
        print("Vektor dokumen tidak ditemukan! Silakan muat dan indeks dataset terlebih dahulu.")
        return []

    print(f"\n=== Mencari: '{query_text}' ===")

    # Preprocess query
    preprocessed_query = self.preprocess_text(query_text)
    print(f"Preprocessed query: '{preprocessed_query}'")

    # Whoosh search to get candidate documents
    ix = open_dir(self.index_dir)
    with ix.searcher() as searcher:
        # Search in preprocessed field
        query = QueryParser("preprocessed", ix.schema).parse(preprocessed_query)
        results = searcher.search(query, limit=None)

        if len(results) == 0:
            print("No documents found matching the query.")
            return []

        print(f"Found {len(results)} candidate documents from Whoosh")

        # Mendapatkan indeks dokumen yang cocok
        candidate_ids = [hit['doc_id'] for hit in results]

    # Menghitung cosine similarity untuk ranking
    query_vector = self.vectorizer.transform([preprocessed_query])

    # Mencari indeks dokumen kandidat
    candidate_indices = []
    for i, doc in enumerate(self.documents):
        if doc['id'] in candidate_ids:
            candidate_indices.append(i)

    if not candidate_indices:
        return []

    # menghitung kemiripan
    similarities = cosine_similarity(query_vector, self.doc_vectors[candidate_indices])[0]

    # Membuat hasil dengan skor
    ranked_results = []
    for idx, similarity in zip(candidate_indices, similarities):
        if similarity > 0:  # Hanya menyertakan dokumen dengan kemiripan positif
            ranked_results.append({
                'doc': self.documents[idx],
                'score': similarity
            })

    # urut berdasarkan skor
    ranked_results.sort(key=lambda x: x['score'], reverse=True)

    # Return top k results
    return ranked_results[:top_k]
```

3.9.Menampilkan Hasil Pencarian

Hasil pencarian ditampilkan ke pengguna dengan format terstruktur

```python
def display_results(self, results):
        """Menampilkan hasil pencarian dalam format yang rapi"""
        if not results:
            print("\nTidak ada dokumen yang relevan ditemukan.")
            return

        print(f"\n=== Top {len(results)} Results ===\n")
        for i, result in enumerate(results, 1):
            doc = result['doc']
            score = result['score']

            print(f"Rank {i}:")
            print(f"  Title: {doc['title']}")
            print(f"  Source: {doc['source']}")
            print(f"  Score: {score:.4f}")

            # Menampilkan potongan konten
            content_preview = doc['content'][:200].replace('\n', ' ')
            print(f"  Preview: {content_preview}...")
            print(f"  Path: {doc.get('path', 'N/A')}")
            print()
```

Penjelasan :

Menampilkan judul dokumen, sumber dataset, skor kemiripan dan cuplikan isi agar pengguna dapat melihat konteks dari pencarian.

3.10. User Interface (CLI)

Antarmuka berbasis *Command Line Interface* (CLI) digunakan untuk interaksi pengguna.

```python
def main():
    """Main CLI interface"""
    ir_system = IRSystem()

    while True:
        print("\n" + "-"*40)
        print("=== INFORMATION RETRIEVAL SYSTEM ===")
        print("-"*40)
        print("[1] Load & Index Dataset")
        print("[2] Search Query")
        print("[3] Exit")
        print("-"*40)

        choice = input("\nSelect menu [1-3]: ").strip()

        if choice == '1':
            num_docs = ir_system.load_all_datasets()
            if num_docs > 0:
                ir_system.create_index()
            else:
                print("\n[!]dak ada dokumen ditemukan! Silakan periksa struktur folder dataset Anda.")
                print("Expected structure:")
                print("dataset/")
                print("  ├── etd_usk.csv")
                print("  ├── etd_ugm.csv")
                print("  ├── kompas.csv")
                print("  ├── tempo.csv")
                print("  └── mojok.csv")

        elif choice == '2':
            query = input("\nMasukkan query pencarian Anda: ").strip()
            if query:
                results = ir_system.search(query, top_k=5)
                ir_system.display_results(results)
            else:
                print("Query tidak boleh kosong!")

        elif choice == '3':
            print("\nThank you for using Information Retrieval System!")
            print("Goodbye!")
            break

        else:
            print("\nPilihan tidak valid! Silakan pilih 1, 2, atau 3.")
```

BAB IV
PENGUJIAN QUERY DAN ANALISIS HASIL

4.1. Tujuan Pengujian

Pengujian dilakukan untuk mengevaluasi kinerja *Information Retrieval System* yang telah diimplementasikan menggunakan modul Whoosh dan metode *Cosine Similarity*. Berikut tujuan pengujian :

- Menguji kemampuan sistem dalam menemukan dokumen relevan berdasarkan query yang diberikan pengguna
- Menganalisis tingkat relevansi hasil pencarian melalui similarity score
- Melihat distribusi sumber dokumen (dataset) yang paling sering muncul dalam hasil pencarian

4.2.Metode Pengujian

Pengujian dilakukan dengan menggunakan tiga query berbeda yang merepresentasikan berbagai topik umum pada koleksi dokumen :

1. Teknologi
2. pendidikan Indonesia
3. kesejahteraan guru Indonesia

Untuk setiap query, sistem menampilkan Top 5 hasil pencarian dengan nilai *cosine similarity* tertinggi. Setiap hasil berisi informasi :

- Judul dokumen
- Sumber dataset
- Skor kemiripan (similarity score)
- Cuplikan konten

4.3.Hasil Pengujian

Query 1 : “teknologi”

- Jumlah kandidat dokumen : 3768
- Top 5 hasil :

Rank	Dokumen	Sumber	Skor
1	MENYIBAK PRAKTIK RURAL TECHNOLOGY: STUDI KASUS KESIAPAN…	etd_ugm	0.6572
2	Surveillance Capitalism dalam Manajemen Data Pengguna Facebook …	etd_ugm	0.5617

3	Penilaian Minat Karyawan Menggunakan Virtual Reality (VR)…	etd_ugm	0.5285
4	PENGARUH LITERASI PEMASARAN DAN KETERSEDIAAN TEKNOLOGI...	etd_usk	0.4660
5	RS Bunda Group Perkenalkan Generasi Kedua Bedah Robotik	kompas	0.4530

Analisis :

Sistem berhasil menampilkan hasil yang sangat relevan dengan kata kunci “teknologi”. Mayoritas dokumen berasal dari etd_ugm, menunjukkan bahwa sumber tersebut banyak memuat topik penelitian tentang teknologi. Skor tertinggi (0.6572) menunjukkan kemiripan yang kuat antara query dan isi dokumen.

Query 2 : “pendidikan indonesia”

Jumlah kandidat dokumen : 2.093

Top 5 hasil :

Rank	Dokumen	Sumber	Skor
1	Biaya Kuliah Fakultas Kedokteran UPI Jalur SNBP 2025...	tempo	0.6156
2	Berapa Biaya Kuliah Universitas Negeri Semarang Jalur SNBP 2025...	tempo	0.6057
3	Pedoman Hari Pendidikan Nasional 2025...	kompas	0.5760
4	Kemendikdasmen Rilis Rapor Pendidikan 2025...	tempo	0.5545
5	Refleksi Filosofis Atas Kurikulum 2013...	etd_ugm	0.5455

Analisis :

Hasil pencarian menunjukkan dokumen dari tempo dan kompas yang relevan dengan konteks “pendidikan di Indonesia”. Skor di atas 0.5 menunjukkan bahwa sistem mampu memahami keterkaitan semantik antar kata seperti “pendidikan” dan “Indonesia”, menghasilkan dokumen berita aktual dan penelitian terkait topik pendidikan nasional.

Query 3 : “kesejahteraan guru indonesia”

Jumlah kandidat dokumen : 60

Top 5 hasil :

Rank	Dokumen	Sumber	Skor
1	Prabowo Nangis Saat Umumkan Kenaikan Gaji Guru...	mojok	0.4688
2	The Welfare of Islamic Religious Teachers in Indonesia	etd_ugm	0.4611
3	Prabowo Akan Umumkan Penyaluran Tunjangan Guru ASN...	tempo	0.3873
4	Reza Arap Donasi Rp 50 Juta untuk Guru Honorer...	tempo	0.3510
5	Prabowo Resmikan Penyaluran Tunjangan Guru ASN Daerah...	tempo	0.3470

Analisis :

Query ini menghasilkan kombinasi sumber dari mojok, tempo, dan etd_ugm, yang semuanya relevan dengan isu kesejahteraan guru di Indonesia. Skor similarity tertinggi (0.4688) sedikit lebih rendah dibanding query sebelumnya karena konteks “kesejahteraan guru” memiliki cakupan semantik yang lebih spesifik dan jumlah dokumen relevan yang lebih sedikit (60 kandidat).

4.4. Analisis Umum Sistem

Berdasarkan ketiga pengujian di atas, diperoleh hasil sebagai berikut :

Aspek	Hasil
Relevansi Hasil	Tinggi (rata-rata skor similarity 0.5 ke atas)
Konsistensi Sumber	Dataset etd_ugm dan tempo paling banyak muncul pada hasil relevan
Kecepatan pencarian	Cepat (< 1 detik untuk setiap query) berkat penggunaan Whoosh indexing
Kualitas preprocessing	Stopword removal dan tokenisasi efektif mengurangi noise
Kelemahan Sistem	Belum mendukung stemming atau sinonim, sehingga query semantik kompleks (misalnya “kemajuan teknologi digital”) bisa kurang optimal

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan terhadap Information Retrieval System (IRS) berbasis *Whoosh* dan metode *Cosine Similarity*, dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem berhasil melakukan pencarian dokumen secara relevan berdasarkan kata kunci (*query*) yang dimasukkan pengguna. Melalui proses *indexing* dan perhitungan *similarity score*, sistem mampu mengurutkan hasil pencarian sesuai tingkat kemiripan antara query dan isi dokumen.
2. Performa sistem tergolong baik dengan waktu pencarian yang sangat cepat (< 1 detik untuk setiap query), berkat efisiensi *Whoosh indexing* serta penerapan *tokenization* dan *stopword removal* pada tahap praproses teks.
3. Relevansi hasil pencarian tinggi, ditunjukkan oleh rata-rata skor kemiripan di atas 0.5 pada berbagai query seperti “teknologi”, “pendidikan indonesia”, dan “kesejahteraan guru indonesia”. Hal ini membuktikan bahwa metode *Cosine Similarity* efektif dalam mengukur kesamaan semantik antar dokumen.
4. Distribusi sumber dokumen menunjukkan pola yang konsisten — dataset *etd_ugm* dan *tempo* sering muncul dalam hasil relevan, menandakan kualitas dan keberagaman konten pada sumber tersebut.
5. Kelemahan sistem terletak pada keterbatasan analisis semantik yang lebih dalam. Sistem belum mendukung *stemming*, *lemmatization*, atau *synonym expansion*, sehingga query dengan makna kompleks seperti “kemajuan teknologi digital” belum selalu menghasilkan hasil optimal.

Secara keseluruhan, sistem Information Retrieval yang dikembangkan telah berfungsi dengan baik, efisien, dan akurat dalam melakukan pencarian teks, serta menunjukkan potensi untuk diterapkan dalam skala yang lebih luas atau diintegrasikan dengan sistem pencarian dokumen ilmiah, berita, maupun arsip digital lainnya.

5.2.Saran

- Peningkatan kemampuan semantik dapat dilakukan dengan menambahkan *stemming* atau *lemmatization*, serta integrasi *word embedding* seperti Word2Vec atau BERT agar sistem mampu memahami hubungan makna antar kata lebih dalam.

- Pengayaan fitur pencarian seperti *autocomplete*, *highlighting*, atau *filtering berdasarkan sumber dan tahun* dapat meningkatkan pengalaman pengguna (*user experience*).

- Optimasi dan perluasan dataset dari berbagai sumber seperti jurnal ilmiah, berita nasional, dan publikasi institusi akan memperkaya hasil pencarian dan meningkatkan cakupan topik yang dapat diindeks.

- Integrasi antarmuka web interaktif direkomendasikan untuk memudahkan pengguna dalam mengakses hasil pencarian tanpa perlu berinteraksi langsung melalui kode program.