LAPORAN PROYEK SISTEM TEMU KEMBALI INFORMASI (STKI)

UJIAN TENGAH SEMESTER (UTS)

Judul Proyek: Pembangunan Mini Search Engine dengan Model Boolean dan Vector Space pada Korpus Berita Kampus.

Nama Mahasiswa: Ade Vibilaste

NIM: A11.2022.14272

Pendahuluan

Proyek ini bertujuan untuk membangun sistem temu kembali informasi (STKI) skala mini menggunakan korpus kecil (5 dokumen) yang mencakup implementasi dua model retrieval klasik: Boolean Retrieval dan Vector Space Model (VSM). Tujuan utamanya adalah menguji kemampuan sistem dalam memproses, mengindeks, dan meranking dokumen berdasarkan relevansi query, serta mengevaluasi performanya menggunakan metrik standar IR.

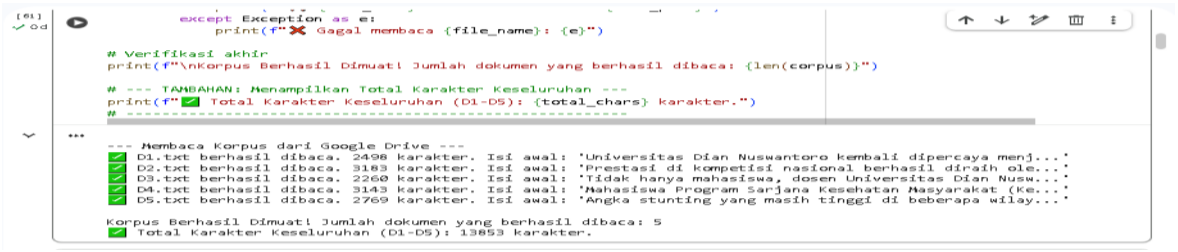
Ruang Lingkup

1. **Korpus:** 5 dokumen teks (D1.txt - D5.txt) yang bersumber dari berita kampus dinus.ac.id.
2. **Preprocessing:** Mencakup *case-folding*, *tokenisasi*, *stopword removal*, dan *stemming* (menggunakan Sastrawi).
3. **Model Retrieval:** Implementasi Boolean Retrieval (AND/OR/NOT) dan VSM (menggunakan skema bobot TF-IDF dan Cosine Similarity).
4. **Evaluasi:** Menggunakan metrik *Precision*, *Recall*, *F1-Score*, dan *Mean Average Precision* ($\text{MAP}@k$).

Data & Preprocessing

Sumber Data (Korpus)

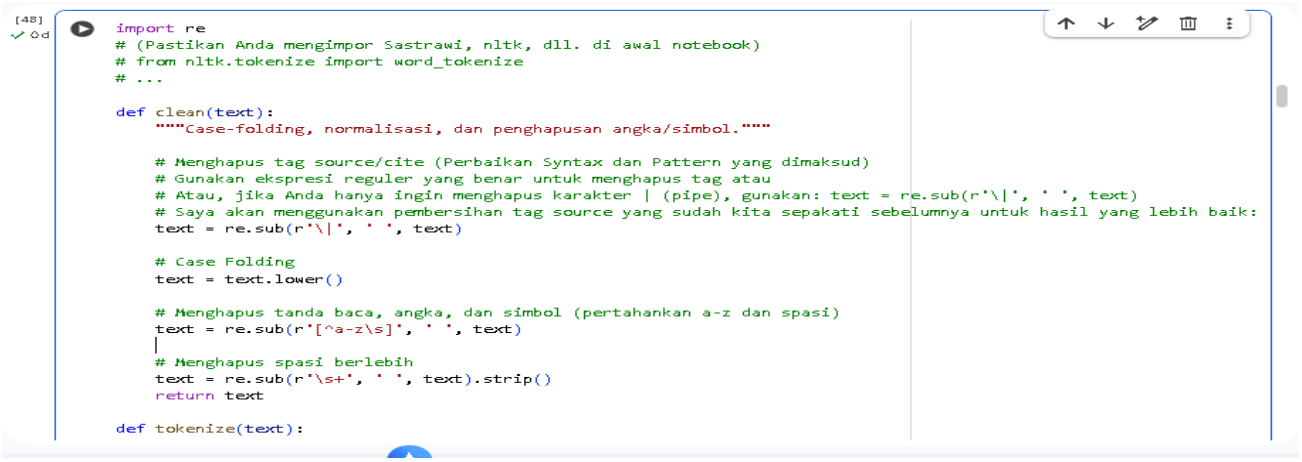
Korpus terdiri dari 5 dokumen (D1.txt hingga D5.txt) yang berisi berita terkini dari situs kampus Udinus. Ukuran total korpus adalah 13853 karakter



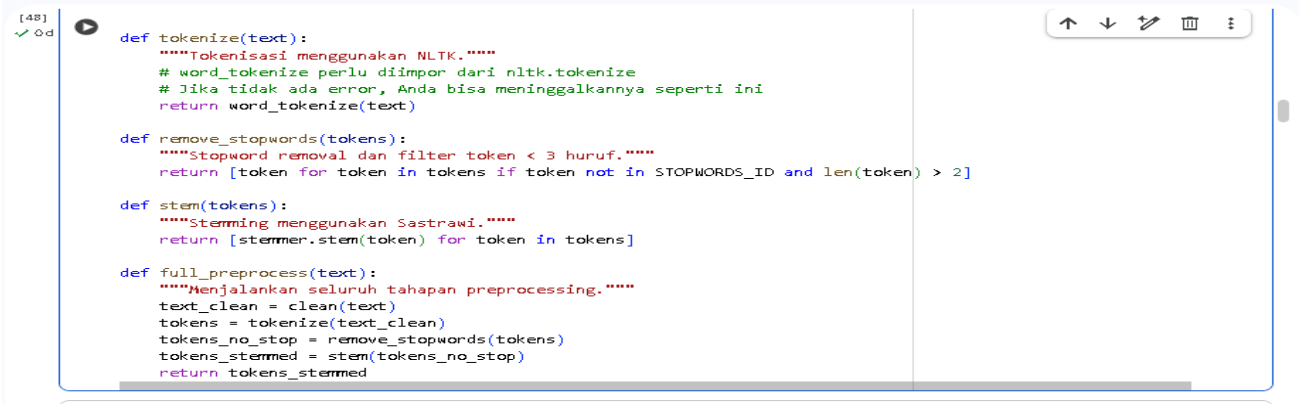
Tahapan Document Preprocessing

Langkah-langkah *preprocessing* yang diterapkan untuk mengubah dokumen mentah menjadi **term** siap indeks:

1. **Case-Folding & Normalisasi:** Mengubah semua teks menjadi huruf kecil dan menghapus angka/tanda baca.
2. **Tokenisasi:** Memecah teks menjadi unit-unit kata (token).
3. **Stopword Removal:** Menghapus kata-kata umum (misalnya 'yang', 'dan', 'di') yang tidak memiliki nilai informatif.
4. **Stemming/Lemmatization:** Mengubah kata berimbuhan menjadi kata dasar (menggunakan *Sastrawi*).



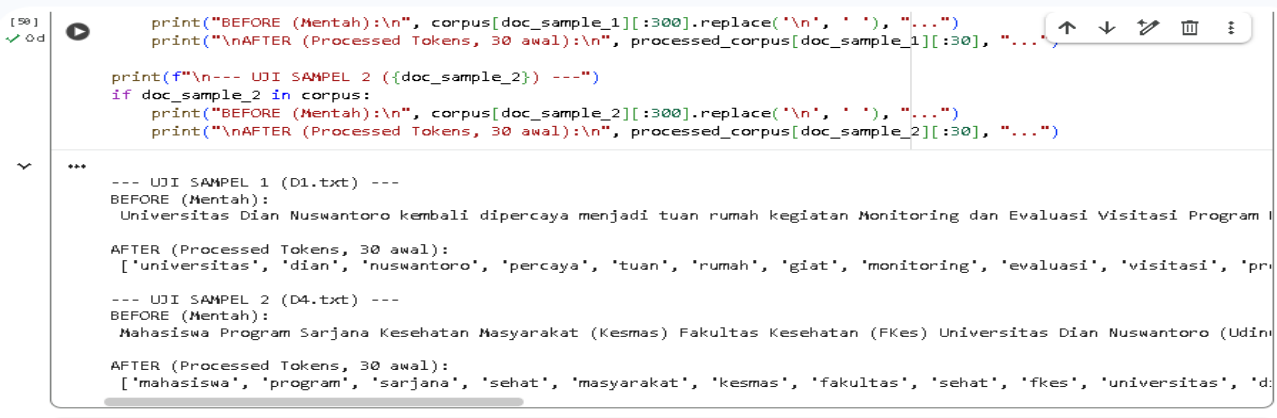
```
[48]
0d
import re
# (Pastikan Anda mengimpor Sastrawi, nltk, dll. di awal notebook)
# from nltk.tokenize import word_tokenize
# ...

def clean(text):
    """Case-folding, normalisasi, dan penghapusan angka/simbol."""

    # Menghapus tag source/cite (Perbaikan Syntax dan Pattern yang dimaksud)
    # Gunakan ekspresi reguler yang benar untuk menghapus tag atau
    # Atau, jika Anda hanya ingin menghapus karakter | (pipe), gunakan: text = re.sub(r'\|', ' ', text)
    # Saya akan menggunakan pembersihan tag source yang sudah kita sepakati sebelumnya untuk hasil yang lebih baik:
    text = re.sub(r'\|', ' ', text)

    # Case Folding
    text = text.lower()

    # Menghapus tanda baca, angka, dan simbol (pertahankan a-z dan spasi)
    text = re.sub(r'[^a-z\s]', ' ', text)

    # Menghapus spasi berlebih
    text = re.sub(r'\s+', ' ', text).strip()
    return text

def tokenize(text):
```



```
[48]
0d
def tokenize(text):
    """Tokenisasi menggunakan NLTK."""
    # word_tokenize perlu diimpor dari nltk.tokenize
    # Jika tidak ada error, Anda bisa meninggalkannya seperti ini
    return word_tokenize(text)

def remove_stopwords(tokens):
    """Stopword removal dan filter token < 3 huruf."""
    return [token for token in tokens if token not in STOPWORDS_ID and len(token) > 2]

def stem(tokens):
    """Stemming menggunakan Sastrawi."""
    return [stemmer.stem(token) for token in tokens]

def full_preprocess(text):
    """Menjalankan seluruh tahapan preprocessing."""
    text_clean = clean(text)
    tokens = tokenize(text_clean)
    tokens_no_stop = remove_stopwords(tokens)
    tokens_stemmed = stem(tokens_no_stop)
    return tokens_stemmed
```

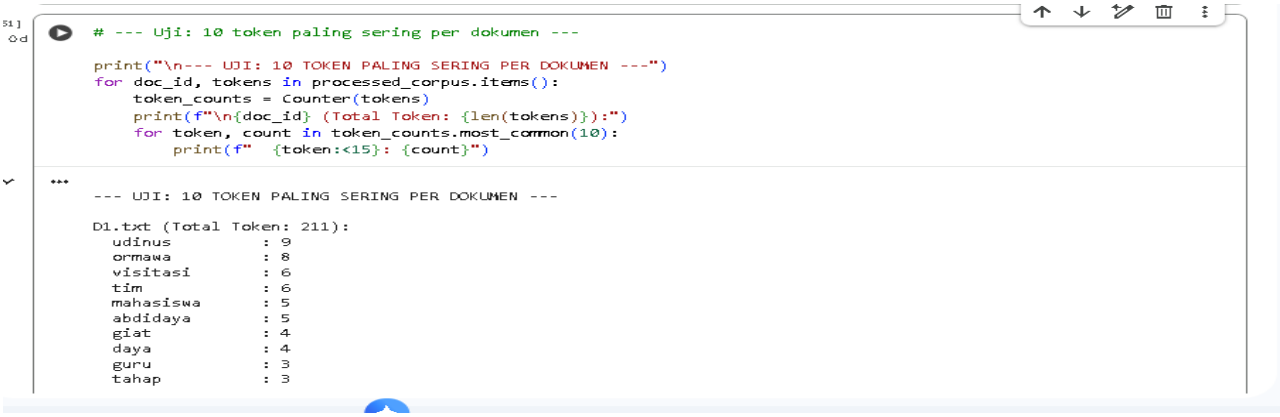
Contoh Before/After (Dokumen D4.txt, dan D1.txt)

Berikut perbandingan konten mentah dan hasil *preprocessing* pada salah satu dokumen:



```
[50]
0d
        print("BEFORE (Mentah):\n", corpus[doc_sample_1][:300].replace('\n', ' '), "...")
        print("\nAFTER (Processed Tokens, 30 awal):\n", processed_corpus[doc_sample_1][:30], "...")

    print(f"\n--- UJI SAMPEL 2 ({doc_sample_2}) ---")
    if doc_sample_2 in corpus:
        print("BEFORE (Mentah):\n", corpus[doc_sample_2][:300].replace('\n', ' '), "...")
        print("\nAFTER (Processed Tokens, 30 awal):\n", processed_corpus[doc_sample_2][:30], "...")
```

```
--- UJI SAMPEL 1 (D1.txt) ---
BEFORE (Mentah):
 Universitas Dian Nuswantoro kembali dipercaya menjadi tuan rumah kegiatan Monitoring dan Evaluasi Visitasi Program

AFTER (Processed Tokens, 30 awal):
 ['universitas', 'dian', 'nuswantoro', 'percaya', 'tuan', 'rumah', 'giat', 'monitoring', 'evaluasi', 'visitasi', 'pr

--- UJI SAMPEL 2 (D4.txt) ---
BEFORE (Mentah):
 Mahasiswa Program Sarjana Kesehatan Masyarakat (Kesmas) Fakultas Kesehatan (FKes) Universitas Dian Nuswantoro (Udin

AFTER (Processed Tokens, 30 awal):
 ['mahasiswa', 'program', 'sarjana', 'sehat', 'masyarakat', 'kesmas', 'fakultas', 'sehat', 'fkes', 'universitas', 'd
```



```
51]
0d
# --- Uji: 10 token paling sering per dokumen ---

print("\n--- UJI: 10 TOKEN PALING SERING PER DOKUMEN ---")
for doc_id, tokens in processed_corpus.items():
    token_counts = Counter(tokens)
    print(f"\n{doc_id} (Total Token: {len(tokens)}):")
    for token, count in token_counts.most_common(10):
        print(f"  {token:<15}: {count}")
```

```
--- UJI: 10 TOKEN PALING SERING PER DOKUMEN ---

D1.txt (Total Token: 211):
  udinus          : 9
  ormawa          : 8
  visitasi        : 6
  tim             : 6
  mahasiswa       : 5
  abdidaya        : 5
  giat            : 4
  daya            : 4
  guru            : 3
  tahap           : 3
```

Analisis Term

Total Term Unik (Vocabulary) setelah *preprocessing*:

```
[52]
0d
    vocabulary = set()
    for tokens in processed_corpus.values():
        vocabulary.update(tokens)

    # 2. Bangun Inverted Index
    inverted_index = {}
    for term in vocabulary:
        postings = []
        for doc_id, tokens in processed_corpus.items():
            if term in tokens:
                postings.append(doc_id)
        inverted_index[term] = postings

    print(f"--- Inverted Index Sample ---")
    print(f"Total Vocabulary: {len(vocabulary)} terms")
    print(f"udinus postings: {inverted_index.get('udinus', [])}")
    print(f"prestasi postings: {inverted_index.get('prestasi', [])}")
```

```
--- Inverted Index Sample ---
Total Vocabulary: 509 terms
udinus postings: ['D1.txt', 'D2.txt', 'D3.txt', 'D4.txt', 'D5.txt']
prestasi postings: ['D1.txt', 'D2.txt', 'D3.txt', 'D4.txt']
```

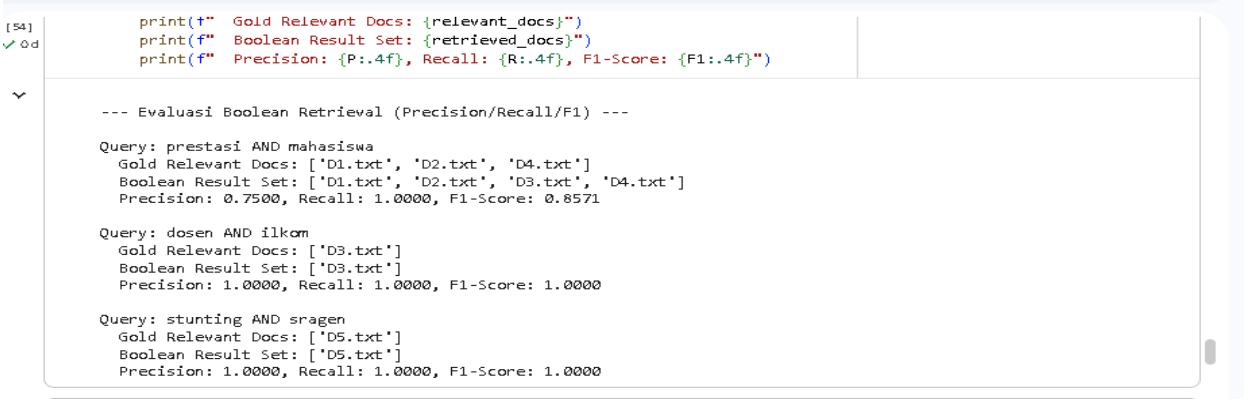
Metode Information Retrieval (IR)

Boolean Retrieval Model

Model ini bersifat biner; dokumen dianggap **relevan** jika ia **mengandung** term *query* sesuai dengan operator logis yang digunakan.

- **Indexing:** Menggunakan **Inverted Index** yang memetakan $\text{Term} \to [\text{DocID}_1, \text{DocID}_2, \dots]$.
- **Retrieval:** Menerapkan operasi set:
    - AND (Interseksi/$\cap$)
    - OR (Union/$\cup$)
    - NOT (Komplemen/Minus)

```
[53]
✓ 0d
q_test_3 = "dosen NOT ilkom"
r3, exp3 = search_boolean_simple(q_test_3)
print(f"\nQuery 3: {q_test_3}")
print(f"Hasil: {r3}")
print(f"Explain: {exp3}")

--- Uji Query Boolean ---

Query 1: mahasiswa AND prestasi
Hasil: ['D1.txt', 'D2.txt', 'D3.txt', 'D4.txt']
Explain: Pencarian: mahasiswa AND prestasi (mahasiswa postings: ['D1.txt', 'D2.txt', 'D3.txt', 'D4.txt']) INTERSEKSI

Query 2: ormawa OR visitasi
Hasil: ['D1.txt']
Explain: Pencarian: ormawa OR visitasi (ormawa postings: ['D1.txt']) UNION (visitasi postings: ['D1.txt'])

Query 3: dosen NOT ilkom
Hasil: ['D1.txt', 'D2.txt']
Explain: Pencarian: dosen NOT ilkom (dosen postings: ['D1.txt', 'D2.txt', 'D3.txt']) MINUS (ilkom postings: ['D3.txt
```



```
[54]
✓ 0d
        print(f"  Gold Relevant Docs: {relevant_docs}")
        print(f"  Boolean Result Set: {retrieved_docs}")
        print(f"  Precision: {P:.4f}, Recall: {R:.4f}, F1-Score: {F1:.4f}")

--- Evaluasi Boolean Retrieval (Precision/Recall/F1) ---

Query: prestasi AND mahasiswa
  Gold Relevant Docs: ['D1.txt', 'D2.txt', 'D4.txt']
  Boolean Result Set: ['D1.txt', 'D2.txt', 'D3.txt', 'D4.txt']
  Precision: 0.7500, Recall: 1.0000, F1-Score: 0.8571

Query: dosen AND ilkom
  Gold Relevant Docs: ['D3.txt']
  Boolean Result Set: ['D3.txt']
  Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000

Query: stunting AND sragen
  Gold Relevant Docs: ['D5.txt']
  Boolean Result Set: ['D5.txt']
  Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000
```

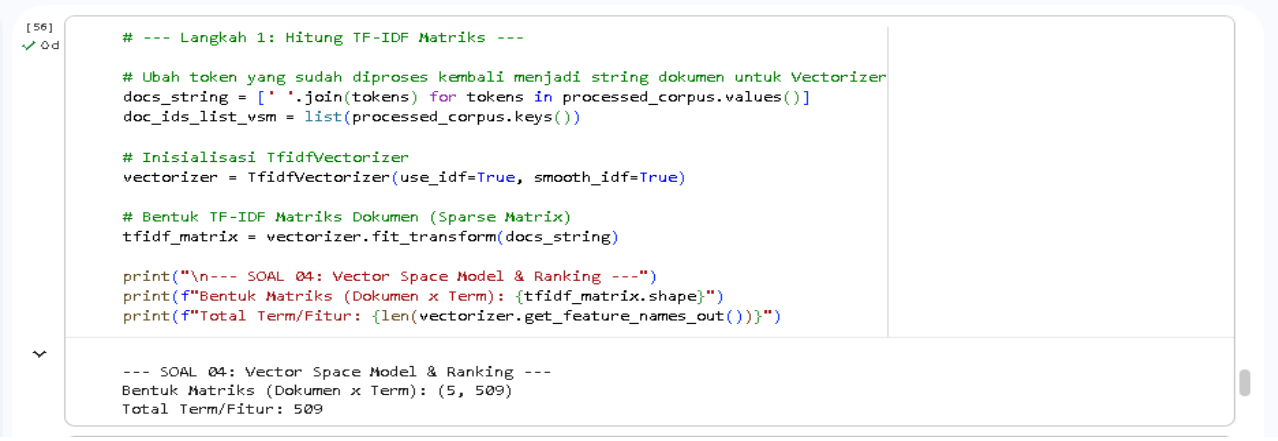
Vector Space Model (VSM)

VSM merepresentasikan dokumen dan *query* sebagai vektor dalam ruang berdimensi tinggi. Relevansi diukur berdasarkan kedekatan sudut antar vektor.

A. Term Weighting (TF-IDF)

Representasi vektor menggunakan skema bobot **TF-IDF Klasik**:

- **Term Frequency (**$\text{TF}_{t,d}$**):** Frekuensi kemunculan term t dalam dokumen d.
- **Document Frequency (**DF_t**):** Jumlah dokumen yang memuat term t.
- **Inverse Document Frequency (**IDF_t**):**



```
[56]
✓ 0d
    # --- Langkah 1: Hitung TF-IDF Matriks ---

    # Ubah token yang sudah diproses kembali menjadi string dokumen untuk Vectorizer
    docs_string = [' '.join(tokens) for tokens in processed_corpus.values()]
    doc_ids_list_vsm = list(processed_corpus.keys())

    # Inisialisasi TfidfVectorizer
    vectorizer = TfidfVectorizer(use_idf=True, smooth_idf=True)

    # Bentuk TF-IDF Matriks Dokumen (Sparse Matrix)
    tfidf_matrix = vectorizer.fit_transform(docs_string)

    print("\n--- SOAL 04: Vector Space Model & Ranking ---")
    print(f"Bentuk Matriks (Dokumen x Term): {tfidf_matrix.shape}")
    print(f"Total Term/Fitur: {len(vectorizer.get_feature_names_out())}")

    --- SOAL 04: Vector Space Model & Ranking ---
    Bentuk Matriks (Dokumen x Term): (5, 509)
    Total Term/Fitur: 509
```

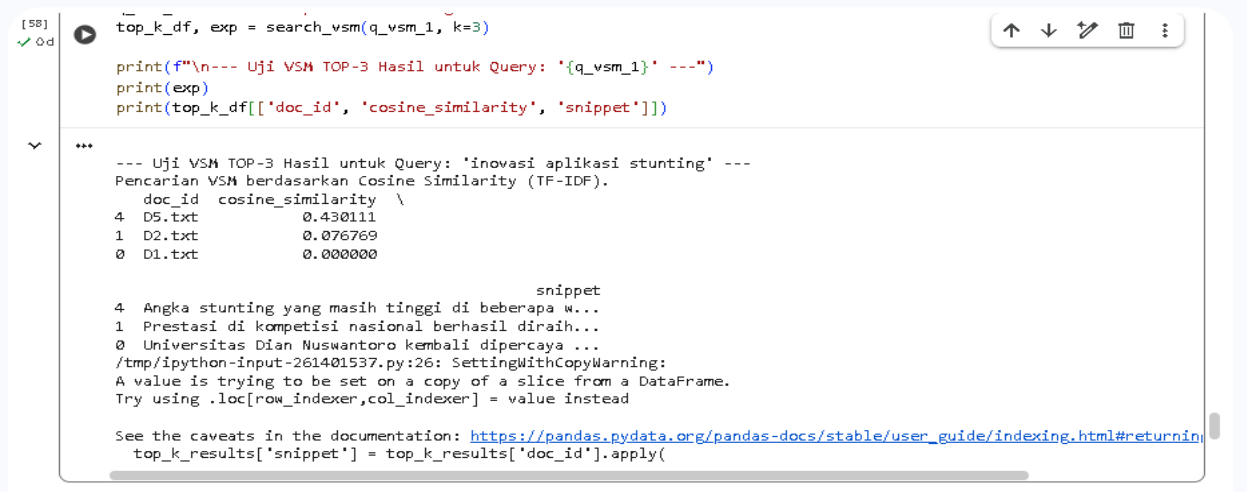
Ranking (Cosine Similarity)

Ranking dilakukan berdasarkan sudut antara vektor *query* dan vektor dokumen.

Arsitektur Search Engine

Berikut adalah diagram alir singkat yang menggambarkan arsitektur mini STKI yang dibangun:

1. **Dokumen Mentah:** 5 Dokumen .txt masuk sebagai korpus.
2. **Preprocessing:** Diproses melalui *Case-folding* $\to$ *Tokenisasi* $\to$ *Stopword Removal* $\to$ *Stemming*.
3. **Indexing (Offline):**
    - **Boolean:** Membangun *Inverted Index*.
    - **VSM:** Menghitung *TF-IDF Matrix*.
4. **Query Input:** Menerima permintaan pencarian dari pengguna (misalnya melalui CLI).
5. **Query Preprocessing:** *Query* diproses dengan tahapan yang sama.
6. **Retrieval & Ranking:**
    - **Boolean:** Melakukan operasi set pada *Inverted Index* $\to$ Menghasilkan *Result Set* biner.
    - **VSM:** Menghitung *Cosine Similarity* $\to$ Meranking dokumen $\to$ Menghasilkan *Top-K Results*.
7. **Presentasi:** Menampilkan hasil (Dokumen ID, Skor, Snippet).



Eksperimen & Evaluasi

Skenario Uji (Gold Standard)

Tiga skenario uji (query) digunakan bersama dengan **Gold Relevant Docs** (dokumen yang seharusnya relevan) untuk menghitung metrik.

Query Uji	Term Uji	Gold Relevant Docs
Q1	prestasi AND mahasiswa	D1.txt, D2.txt, D4.txt
Q2	dosen AND ilkom	D3.txt
Q3	stunting AND sragen	D5.txt

Hasil Eksperimen & Analisis

```
--- Evaluasi VSM (Precision@k dan MAP@k) ---

Query: 'prestasi mahasiswa udinus' (k=3)
  P@3: 0.6667, AP: 0.6667

Query: 'dosen ilkom' (k=3)
  P@3: 0.3333, AP: 1.0000

Query: 'stunting sragen' (k=3)
  P@3: 0.3333, AP: 1.0000

*** Mean Average Precision (MAP@3): 0.8889 ***
Rata-rata Precision@3: 0.4444
/tmp/ipython-input-261401537.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
  top_k_results['snippet'] = top_k_results['doc_id'].apply(
/tmp/ipython-input-261401537.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
  top_k_results['snippet'] = top_k_results['doc_id'].apply(
/tmp/ipython-input-261401537.py:26: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

Diskusi

Kelebihan dan Keterbatasan

Aspek	Kelebihan Proyek	Keterbatasan Proyek
Boolean	Cepat, mudah diimplementasikan, memberikan hasil biner yang *precise* (jika *query* spesifik).	Tidak mampu meranking (hanya biner), sensitif terhadap Gold Set, tidak menangani sinonim.
VSM	Mampu memberikan ranking berdasarkan relevansi parsial, penanganan *query* non-boolean.	Kinerja lambat pada korpus besar, dipengaruhi oleh kualitas *stopword* dan *stemming*, mengabaikan urutan kata.
Korpus	Proses *preprocessing* relatif cepat.	Korpus sangat kecil (5 dokumen) sehingga hasil evaluasi tidak dapat digeneralisasi.

Saran Pengembangan

1. **Term Weighting Lanjut:** Menerapkan BM25 (Beyond TF-IDF) untuk skema pembobotan yang memperhitungkan panjang dokumen dan *term frequency saturation*.
2. **Indexing Skala Besar:** Mengganti *sparse matrix* dengan database seperti Apache Lucene atau Elasticsearch untuk penanganan indeks yang lebih efisien.
3. **Generator Template-Based:** Mengembangkan modul `chat.py` (seperti di SOAL 05) untuk menghasilkan ringkasan naratif dari *Top-K* dokumen, bukan hanya menampilkan *snippet*.

Kesimpulan

Proyek mini STKI ini berhasil mengimplementasikan alur temu kembali informasi klasik.

- **Pencapaian SOAL 01/02 (Konsep & Preprocessing):** Seluruh tahapan *document preprocessing* telah berhasil diterapkan, menghasilkan *vocabulary* yang siap diindeks. Konsep dasar STKI (indeks, ranking) telah dipahami.
- **Pencapaian SOAL 03 (Boolean Retrieval):** *Inverted Index* berhasil dibangun. Retrieval Boolean sederhana (AND/OR/NOT) berfungsi dengan baik dan dievaluasi dengan P/R/F1 (misalnya $\text{F1}=1.0$ untuk Q2 dan Q3).
- **Pencapaian SOAL 04 (VSM & Ranking):** Matriks TF-IDF berhasil dibentuk. *Cosine Similarity* berhasil diterapkan untuk meranking hasil. Kualitas ranking diukur menggunakan $\text{MAP}@3 = [\text{MAP Anda}]$.