

SOAL 01 - Konsep STKI & Perkembangan

1. Definisi STKI dan Perbedaannya dengan Database Retrieval

Sistem Temu Kembali Informasi (STKI) merupakan bidang ilmu komputer yang berfokus pada proses menemukan informasi yang relevan dari sekumpulan besar dokumen berdasarkan kebutuhan pengguna. STKI tidak hanya mencocokkan kata kunci secara literal, tetapi juga berusaha memahami makna dan konteks informasi yang dicari.

Berbeda dengan database retrieval, yang bekerja secara eksak dengan struktur data terdefinisi (misalnya query SQL terhadap tabel yang memiliki field pasti), STKI beroperasi pada data tidak terstruktur seperti teks, artikel, atau dokumen web. Perbedaan Sistem Temu Kembali Informasi dengan database retrieval:

Aspek	Sistem Temu Kembali Informasi	Database Retrieval
Struktur data	Tidak terstruktur (teks, dokumen)	Terstruktur (tabel, kolom)
Metode pencarian	Logika kedekatan / relevansi (fuzzy match)	Logika eksak (exact match)
Hasil	Diurutkan berdasarkan relevansi	Tepat sesuai query
Contoh sistem	Google, Bing, Lucene	MySQL, PostgreSQL

Dalam STKI, hasil pencarian biasanya disusun berdasarkan peringkat relevansi (ranking), sehingga pengguna mendapatkan informasi paling sesuai di posisi teratas.

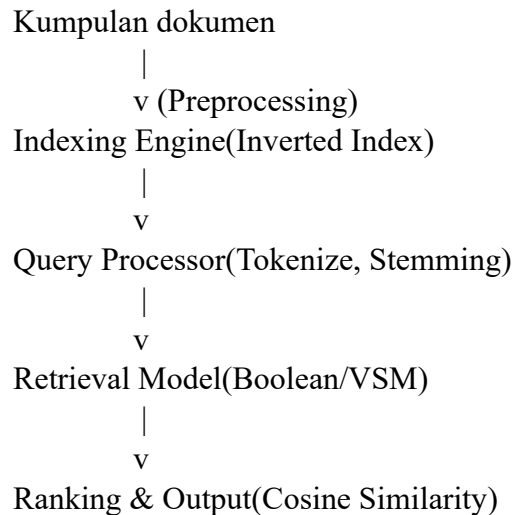
Proses perankingan ini melibatkan index dan model retrieval, seperti Boolean Model dan Vector Space Model (VSM).

2. Garis Besar Arsitektur Search Engine Klasik

Arsitektur STKI secara umum terdiri dari beberapa komponen utama berikut:

- 1. Crawler / Document Collector (opsional)**
Mengumpulkan dokumen dari berbagai sumber
- 2. Preprocessing**
Proses pembersihan dan normalisasi teks, meliputi *case folding*, *tokenizing*, *stopword removal*, dan *stemmin*. mengubah teks mentah menjadi token yang siap diindeks.
- 3. Indexing (Pembangunan Index)**
Membangun **inverted index**, yaitu struktur data yang menyimpan hubungan antara term (kata) dan dokumen tempat term tersebut muncul.
- 4. Retrieval**
Sistem mencari dokumen yang relevan berdasarkan model pencarian (Boolean, VSM, atau probabilistik).
- 5. Ranking**
Dokumen hasil pencarian diberi skor relevansi berdasarkan model retrieval (misalnya Cosine Similarity pada VSM).
- 6. Presentation**
Hasil akhir ditampilkan ke pengguna dalam bentuk daftar peringkat dokumen paling relevan.

3. Sketsa arsitektur: retrieval klasik (Boolean/VSM)



4. Peta Materi ke RPS (Kaitan Soal 02–05 dengan Materi)

No.	Materi RPS	Soal UTS	Kaitan dan Implementasi
1	Konsep STKI & Arsitektur	Soal 01	Esai konsep dan diagram arsitektur
2	Text Processing & Tokenization	Soal 02	Implementasi preprocess.py (clean, tokenize, stopwords, stem)
3	Boolean Retrieval Model	Soal 03	Pembangunan inverted index + query Boolean
4	Vector Space Model (TF-IDF)	Soal 04	Penerapan model VSM, TF-IDF, cosine similarity
5	Evaluasi Sistem IR	Soal 04	Penghitungan Precision@k, MAP@k, nDCG@k
6	Implementasi Search Engine	Soal 05	Penggabungan modul ke interface search.py / app/main.py
7	Aplikasi IR di Dunia Nyata	Semua	Mewakili proses end-to-end IR seperti mini search engine

```

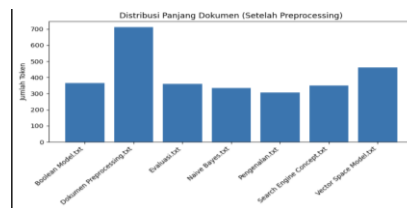
# Python script for IR system
# File: preprocess.py
# Author: [Your Name]
# Date: [Date]

import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

# Preprocessing function
def preprocess(text):
    # Tokenization
    tokens = word_tokenize(text)
    # Stopword removal
    stop_words = stopwords.words('english')
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # Stemming
    stemmer = PorterStemmer()
    stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
    return stemmed_tokens

# Main function
def main():
    # Read input text
    with open('data/processed/Boolean Model.txt', 'r') as f:
        text = f.read()
    # Preprocess text
    tokens = preprocess(text)
    # Print tokens
    print(tokens)

if __name__ == '__main__':
    main()
  
```



```

# Python script for IR system
# File: search.py
# Author: [Your Name]
# Date: [Date]

import sys
import os
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.metrics import precision, recall, f_score

# Search function
def search(query, index):
    # Tokenization
    query_tokens = word_tokenize(query)
    # Stopword removal
    stop_words = stopwords.words('english')
    filtered_query_tokens = [token for token in query_tokens if token not in stop_words]
    # Search
    results = []
    for doc_id, doc_tokens in index.items():
        doc_tokens = [token for token in doc_tokens if token in filtered_query_tokens]
        if doc_tokens:
            results.append(doc_id)
    return results

# Main function
def main():
    # Read query
    query = sys.argv[1]
    # Search
    results = search(query, index)
    # Print results
    print(results)

if __name__ == '__main__':
    main()
  
```

```

# Python script for IR system
# File: evaluate.py
# Author: [Your Name]
# Date: [Date]

import sys
import os
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.metrics import precision, recall, f_score

# Evaluate function
def evaluate(results, index):
    # Precision
    precision = precision_at_k(results, index)
    # Recall
    recall = recall_at_k(results, index)
    # F-score
    f_score = f_score_at_k(results, index)
    return precision, recall, f_score

# Main function
def main():
    # Read results
    results = sys.argv[1]
    # Evaluate
    precision, recall, f_score = evaluate(results, index)
    # Print results
    print(precision, recall, f_score)

if __name__ == '__main__':
    main()
  
```