Tugas IF4042 Sistem Temu Balik Informasi

Sistem Temu Balik Informasi dan Ekspansi *Query* dengan Word2Vec

Disusun oleh:

Kevin John Wesley Hutabarat	NIM. 13521042
Arleen Chrysantha Gunardi	NIM. 13521059
Moh. Aghna Maysan Abyan	NIM. 13521076
Austin Gabriel Pardosi	NIM. 13521084
Ryan Samuel Chandra	NIM. 13521140

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

1. Deskripsi Aplikasi

Ekspansi kueri (*query expansion*) adalah teknik yang umumnya digunakan pada proses temu balik informasi untuk meningkatkan kinerja temu balik melalui pengubahan bentuk kueri yang diberikan oleh pengguna, seperti menambahkan istilah baru atau memberikan bobot pada kueri awal (Vechtomova & Wang, 2006). Sementara itu, Word2Vec adalah teknik yang digunakan pada pemrosesan bahasa alami (NLP) untuk memperoleh vektor representasi dari sebuah kata berdasarkan konteks kata tersebut terhadap kata-kata disekitarnya (Mikolov et al., 2013).

Aplikasi ini adalah aplikasi yang dapat melakukan ekspansi kueri dengan teknik Word2Vec. Pada aplikasi ini, *tech-stack* yang digunakan adalah Next.js (didukung Typescript, Tailwind CSS, dan ShadCN UI) untuk *front-end* dan FastAPI (dengan bahasa pemrograman Python) untuk *back-end*. Aplikasi ini memungkinkan pengguna untuk:

1) Melakukan *stemming*
2) Melakukan *stop-word elimination*
3) Memilih metode pembobotan:
    - TF (*Term Frequency*): *logarithmic*, *binary*, *augmented*, atau *raw*
    - IDF (*Inverse Document Frequency*)
    - TF-IDF
    - TF-IDF-*Cosine normalization*
4) Memasukkan beberapa kata/*term* yang ingin ditambahkan atau seluruh kata ditambahkan

Selain itu, program yang telah dibangun untuk aplikasi ini dapat melakukan:

1) Menerima kueri secara interaktif maupun *batch*
2) Menampilkan kueri hasil ekspansi beserta bobot dan hasil pemeringkatannya
3) Menampilkan *inverted file* sebuah dokumen
4) Menampilkan hasil temu balik (pemeringkatan dokumen serta nilai *similarity measure*) dan MAP (*mean average precision*), baik tiap kueri maupun seluruh kueri untuk kueri yang belum melalui proses ekspansi maupun kueri yang sudah melalui proses ekspansi

2. Deskripsi Program

A. Deskripsi *Function*

1. Class QueryExpansionService

Kelas ini menangani kebutuhan *query expansion* dari sistem temu balik. Pada aplikasi ini, metode *query expansion* yang digunakan adalah *expansion* dengan Word2Vec. Koleksi dokumen yang dimiliki dilatih dengan model Word2Vec untuk dicari kemiripan antar katanya. Kelas ini memiliki fungsi-fungsi sebagai berikut.

a. **`create`**

Fungsi ini digunakan untuk menginisialisasi dan melatih model Word2Vec. Masukan dari fungsi ini adalah `document_path` yang berbentuk string. Format file yang dapat ditangani adalah dua jenis, yaitu CISI *file* dan JSON. Fungsi ini mengembalikan model Word2Vec yang akan digunakan untuk *query expansion.*

b. **`ensure_model_trained`**

Fungsi ini dibuat untuk memastikan model sudah dilatih sebelum digunakan. Jika model belum dilatih dan `document_path` diberikan, model akan dilatih terlebih dahulu. Fungsi ini menerima `document_path`, yaitu path ke dokumen untuk dilatih.

c. **`train_word2vec_model`**

Fungsi ini digunakan untuk melatih model Word2Vec dari dokumen. Fungsi ini menerima masukan documents, yang memiliki format Dict[str,str]. *Key* dari documents adalah id dokumen, sementara *value* nya adalah konten dokumen. Untuk setiap dokumen, kontennya akan dilakukan pra-pemrosesan terlebih dahulu sesuai dengan masukan pengguna. Dokumen yang sudah melalui pra-pemrosesan kemudian dilatih pada model Word2Vec dengan ukuran vektor 100, window berukuran 5 kata, frekuensi term minimum 5, 4 *thread*, dan model skip-gram.

d. **`load_pretrained_model`**

Fungsi ini digunakan untuk memuat model *pretrained* Word2Vec. Fungsi ini menerima masukan model_path berbentuk string. model_path adalah path yang mengandung model *pretrained*.

e. **`expand_query`**

Fungsi ini adalah fungsi utama dari ekspansi *query*. Fungsi ini melakukan ekspansi query dengan model Word2Vec. Masukan yang digunakan fungsi ini adalah query, threshold, dan limit. Query yang akan dicocokkan ke setiap dokumen disimpan di variabel query, sedangkan threshold digunakan sebagai ambang batas kemiripan dari pasangan kata untuk dianggap cukup mirip, dan limit adalah banyak kata yang ditambahkan ke dalam *query.* Untuk setiap term di dalam query, akan dicari term dari dokumen yang mirip dengan term tersebut. Jika argumen limit tidak kosong, term yang diambil hanyalah *term* teratas berdasarkan kemiripan sesuai dengan besarnya limit masukan. Fungsi ini menghasilkan *dictionary* berisi *query* asli dan *expanded*, serta term-term yang ditambahkan.

f. **`get_similar_terms`**

Fungsi ini digunakan untuk mendapatkan term-term yang mirip dengan term yang diberikan. Fungsi ini menerima argumen berupa term, yaitu kata yang ingin dicari kata-kata yang mirip dengannya. Selain itu, terdapat juga threshold yang berupa ambang batas nilai kemiripan untuk sebuah pasangan term dianggap mirip. Fungsi ini menghasilkan list dari term yang dianggap mirip, juga nilai kemiripannya.

g. **`read_cisi_collection`**

Fungsi ini digunakan untuk membaca koleksi CISI dan mengembalikan dokumen dalam bentuk *dictionary.* Fungsi ini menerima masukan berupa file_path, yaitu path letak dokumen berada. Fungsi ini membaca *file* dokumen dan mengubahnya ke dalam bentuk *dictionary* agar dapat digunakan oleh aplikasi.

h. **`read_json_collection`**

Fungsi ini digunakan untuk membaca dokumen berformat JSON dan mengembalikan dokumen dalam bentuk *dictionary*. Fungsi ini menerima masukan berupa file_path, yaitu path letak dokumen berada. Fungsi ini membaca *file* dokumen dan mengubahnya ke dalam bentuk *dictionary* agar dapat digunakan oleh aplikasi.

2. Class RetrievalService

Kelas ini menjadi inti dari sistem temu balik informasi, yaitu menangani kebutuhan pengembalian informasi (dokumen) berdasarkan kueri, baik yang dimasukkan secara interaktif maupun *batch*. Kelas ini akan digunakan untuk mengembalikan informasi berdasarkan kueri semula, serta kueri hasil *expansion* menggunakan *Word2Vec*. Kelas ini juga memanfaatkan fungsi *parsing* kueri berbentuk *batch* serta *relevance judgement* dari *file* "func_parser.py" pada folder "parsing". Kelas ini memiliki fungsi-fungsi sebagai berikut.

a. **`create_inverted_file`**

➢ Masukan:

- `documents: Dict[str, Any]`: merupakan hasil *parsing* koleksi dokumen dengan fungsi `parser_docs` yang dilakukan oleh front-end, berformat kamus ID dokumen terhadap isi teksnya.
- `use_stemming: bool`: merupakan keputusan apakah akan menggunakan *porter stemming* terhadap setiap *term* atau tidak, dengan format *True*/*False*.
- `use_stopword_removal: bool`, merupakan keputusan apakah akan mengeliminasi setiap *stop word* pada dokumen atau tidak, dengan format *True*/*False*.
- `document_weighting_method: Dict[str, bool]`: kamus berisi metode-metode pembobotan yang tersedia (TF: *raw*, *log*, *binary*, *augmented*; IDF, *cosine normalization*) yang akan diterapkan ke dokumen terhadap *True*/*False* terkait digunakan atau tidak.

➢ Keluaran:

- `inverted_file: Dict[str, Dict[str, float]]`: kamus berisi *term* terhadap kamus lagi yang berisi ID dokumen terhadap bobot *term* tersebut dalam dokumen tersebut.

➢ Deskripsi/Proses:

1. Setiap *value* dari kamus masukan di-*preprocess* dengan menggunakan fungsi `preprocess_text()` agar *stemming* dan *stop word removal* dapat dilakukan jika masukannya *True*.
2. Perhitungan jumlah kemunculan masing-masing token dalam masing-masing dokumen dilakukan, menghasilkan *dictionary* `freq_file` sebagai masukan untuk `calculate_tf_idf()`.
3. `freq_file` diiterasi dan setiap *value*-nya (berisi kamus kata terhadap jumlahnya) diiterasi lagi untuk menghitung bobotnya menggunakan `calculate_tf_idf()`.
4. Kamus *inverted file* diciptakan dengan membalik `freq_file` dari formasi ID dokumen-*term*-jumlah kemunculan menjadi *term*-ID dokumen-bobot.

b. **`calculate_tf_idf`**

➢ Masukan:

- `term: str`: merupakan *term*/kata yang akan dihitung bobotnya pada dokumen tertentu.
- `doc: str`: merupakan ID dokumen tempat term tersebut berada.
- `freq_file: Dict[str, Any]`: merupakan *dictionary* yang berisi frekuensi kemunculan setiap term pada tiap dokumen. Format: `{doc_id: Counter({term: freq, ...}), ...}`
- `weighting_method: Dict[str, bool]`: merupakan konfigurasi metode pembobotan yang digunakan (misalnya: `tf_raw`, `tf_log`, `tf_binary`, `tf_augmented`, `use_idf`, `use_normalization`).

➢ Keluaran:

- `Dict[str, Any]`: merupakan *dictionary* yang berisi term, dokumen, dan bobot hasil perhitungan. Format: `{"term": term, "doc": doc, "weight": weight}`.

➢ Deskripsi/Proses:

- Fungsi ini menghitung bobot sebuah *term* pada dokumen tertentu menggunakan metode pembobotan yang dipilih. Bobot-bobot tersebut digunakan dalam membentuk *inverted file*.

1. Proses dimulai dengan menghitung TF (*Term Frequency*) sesuai metode (*raw*, *log*, *binary*, *augmented*).
2. Jika `use_idf` diaktifkan, maka dihitung juga IDF (*Inverse Document Frequency*) untuk *term* tersebut.
3. Jika normalisasi diaktifkan, bobot akan dinormalisasi berdasarkan panjang dokumen.
4. Hasil akhirnya adalah bobot akhir *term* pada dokumen yang akan digunakan dalam *inverted file*.

c. **`calculate_query_weight`**

➢ Masukan:

- `query: str`: merupakan isi (teks) *query* yang akan diproses, berupa *string*.
- `weighting_method: Dict[str, bool]`: merupakan konfigurasi metode pembobotan yang digunakan untuk *query*.
- `inverted_file: Dict[str, Dict[str, float]]`: merupakan *inverted file* yang sudah dibangun dari dokumen, alias keluaran dari fungsi `create_inverted_file()`.

➢ Keluaran:

- `query_vector: Dict[str, float]`: kamus berisi setiap *term* pada *query* terhadap bobotnya.

➢ Deskripsi/Proses:

- Mirip dengan `calculate_tf_idf()`, hanya saja khusus *query*.

d. **`calculate_similarity`**

➢ Masukan:

- `query_vector: Dict[str, float]`: keluaran dari fungsi `calculate_query_weight()`
- `document_vectors: Dict[str, Dict[str, float]]`: *inverted file*, alias keluaran dari fungsi `create_inverted_file()`.

➢ Keluaran:

- `query_docs_similarities: Dict[str, Any]`: kamus berisi nilai kemiripan dari *query* dengan setiap dokumen.

➢ Deskripsi/Proses:

- Fungsi ini menghitung nilai kemiripan antara vektor query dengan setiap vektor dokumen dengan melakukan operasi *dot product.*

e. **`retrieve_document_single_query`**

➢ Masukan

- `query: str`: merupakan *query* yang dimasukkan oleh *user*.
- `inverted_file: Dict[str, Dict[str, float]]`: merupakan *inverted file* hasil pembobotan dokumen.
- `weighting_method: Dict[str, bool]`: merupakan konfigurasi metode pembobotan untuk *query*.
- `relevant_doc: List[str]`: merupakan daftar ID dokumen relevan (untuk evaluasi, bisa kosong jika kueri interaktif).

➢ Keluaran

- `Tuple[Dict[str, float], float]`: merupakan *tuple* yang berisi *dictionary* dokumen terurut berdasarkan similaritasnya dengan *query*, dan nilai *average precision* (AP) jika tersedia.

➢ Deskripsi/Proses

- Fungsi ini melakukan proses *retrieval* untuk satu *query*.

1. *Query* diubah menjadi vektor, dihitung similaritasnya dengan dokumen, lalu dokumen diurutkan berdasarkan nilai similaritas.
2. Jika tersedia daftar dokumen relevan, maka dihitung juga *average precision* (AP) sebagai metrik evaluasi.

f. **`retrieve_document_batch_query`**

➢ Masukan

- `filename: str`: merupakan nama/lokasi file *batch query* mentah yang berisi beberapa *query*.
- `inverted_file: Dict[str, Dict[str, float]]`: merupakan *inverted file* hasil pembobotan dokumen.
- `weighting_method: Dict[str, bool]`: merupakan konfigurasi metode pembobotan untuk *query*.
- `relevant_doc_filename: str`: merupakan nama/lokasi *file* daftar dokumen relevan (mentah) untuk setiap *query*.

➢ Keluaran

- `Tuple[List[Tuple[Dict[str, float], Tuple[str, str], float]], float]`: merupakan *list* dokumen hasil *retrieval* untuk setiap *query* (berisi *similarity*, informasi ID dan konten *query*, dan AP untuk hasil *retrieval* tersebut), serta nilai *mean average precision* (MAP) untuk keseluruhan *retrieval* (semua kueri).

➢ Deskripsi/Proses

- Fungsi ini melakukan proses *retrieval* untuk sekumpulan *query* (*batch*).
- ***Query* yang tidak memiliki dokumen relevan di-*skip***.

1. *File* mentah *batch query* dan *relevant document* di-*parse* menggunakan fungsi `parser_query()` dan `parser_qrels()`.
2. Kumpulan *query* hasil *parsing* diiterasi, dihitung similaritasnya dengan semua dokumen serta *average precision*-nya meng-

gunakan fungsi `retrieve_document_single_query()`, lalu hasilnya dikumpulkan.

3. Terakhir, *mean average precision* (MAP) dihitung sebagai metrik evaluasi keseluruhan *batch*.

g. **`retrieve_document_by_id`**

➢ Masukan

- `id: str`: merupakan ID dokumen yang ingin diambil.
- `documents: List[Dict[str, Any]]`: merupakan *list* dokumen yang tersedia.

➢ Keluaran

- `Dict[str, Any]`: merupakan *dictionary* berisi informasi dokumen (*author*, *title*, *content*) jika ditemukan, atau *dictionary* kosong jika tidak ditemukan.

➢ Deskripsi/Proses

- Fungsi ini mencari dokumen berdasarkan ID yang diberikan.

1. Jika ditemukan, akan mengembalikan informasi dokumen (*author*, *title*, *content*).
2. Jika tidak ditemukan, mengembalikan *dictionary* kosong.

h. **`retrieve_document_by_ids`**

➢ Masukan

- `documents: List[Dict[str, Any]]`: merupakan *list* dokumen yang tersedia.
- `ids: List[str]`: merupakan list ID dokumen yang ingin diambil.

➢ Keluaran

- `List[Dict[str, Any]]`: merupakan *list* informasi dokumen yang ditemukan berdasarkan ID yang diberikan.

➢ Deskripsi/Proses

- Fungsi ini mencari beberapa dokumen berdasarkan *list* ID yang diberikan.

1. Untuk setiap ID, fungsi akan memanggil *retrieve_document_by_id*.
2. Hasilnya adalah list dokumen yang ditemukan.

i. **`get_weight_by_document_id`**

➢ Masukan

- `document_id: str`: merupakan ID dokumen yang ingin diambil bobot *term*-nya.
- `inverted_file: Dict[str, Dict[str, float]]`: merupakan *inverted file* hasil pembobotan dokumen.

➢ Keluaran

- `Dict[str, float]`: merupakan *dictionary* yang memetakan setiap *term* pada dokumen dengan ID tersebut ke bobotnya. Format: *{term: weight, ...}*.

➢ Deskripsi/Proses

- Fungsi ini mengambil bobot setiap *term* yang terdapat pada dokumen tertentu dari *inverted file*.
- Setiap *term* yang memiliki *entry* untuk dokumen tersebut akan dimasukkan ke *dictionary* hasil.

B. Deskripsi *Endpoint*

Endpoint digunakan untuk merutekan semua permintaan yang masuk dari *front-end* ke *back-end* agar dapat ditangani dengan fungsi dan cara yang tepat. *Library* utama yang digunakan adalah FastAPI.

1. documents.py

Menampung *endpoint* dengan rute yang diawali oleh "/documents". Semua *endpoint* berkaitan dengan operasi dokumen. Terdapat 4 rute sebagai berikut.

a. **`GET "/list"`**

Mengembalikan daftar ID dokumen beserta isinya dari *file* "parsing_docs.json" yang merupakan hasil *parsing* dari *endpoint* `POST ("/parse")`. *Exception* yang

digunakan adalah *file* tidak ditemukan (404), terjadi eror pada saat *parsing* (500), dan *internal server error* (500).

b. **`POST "/upload"`**

Mengunggah dokumen dengan masukan harus bertipe "file" dengan tipe input *form-data* (*HTTP multipart upload*). Respons yang dikembalikan adalah nama *file* yang dipilih.

c. **`POST "/parse"`**

Melakukan *parsing* dokumen pada direktori yang namanya (*string*) menjadi input bertipe *application/x-www-form-urlencoded*. Kembaliannya yaitu nama direktori.

d. **`POST "/retrieve-by-ids"`**

Mengembalikan dokumen dari *file* "parsing_docs_with_field.json" berdasarkan daftar ID yang dikirim oleh klien. Dokumen akan dikonversi menjadi *list of dictionaries*, lalu dilakukan pencocokan ID berdasarkan *request*. *Exception* yang digunakan adalah *file* tidak ditemukan (404), terjadi eror pada saat *parsing* (500), dan *internal server error* (500).

2. query.py

Menampung *endpoint* dengan rute yang diawali oleh "/query". Terdapat 3 rute sebagai berikut.

a. **`POST "/query/expand"`**

Melakukan ekspansi kueri pada sebuah *string* kueri dengan Word2vec. *Exception* yang digunakan adalah *internal server error* (500).

b. **`POST "/query/expand-batch"`**

Melakukan ekspansi kueri pada beberapa kueri dalam *file* JSON. *Exception* yang digunakan adalah *file* tidak ditemukan (400) dan terjadi eror saat parsing (500).

c. **`GET "/query/model-status"`**

Melakukan pengecekan status model Word2vec. Status yang diberikan adalah *ready* (model sudah tersedia), *not_ready* (model belum tersedia), atau *error* (terdapat kesalahan saat memeriksa status model).

d. **`POST "/query/retrain-model"`**

Melatih ulang model Word2Vec secara dinamis tanpa perlu *restart* aplikasi, dengan parameter *preprocessing* (*stemming* dan *stopword removal*) yang dapat dikustomisasi. *Endpoint* ini akan mengembalikan detail konfigurasi sebelum dan sesudah *retraining*, serta status keberhasilan proses *retraining*.

e. **`GET "/query/preprocessing-config"`**

Mengembalikan konfigurasi *preprocessing* (seperti penggunaan *stemming* dan *stopword removal*) yang sedang digunakan oleh model Word2Vec saat ini. *Endpoint* ini memudahkan pengecekan konsistensi *preprocessing* antara proses pelatihan dan penggunaan model.

3. retrieval.py

Menampung *endpoint* dengan rute yang diawali oleh "/retrieval". Terdapat 10 rute sebagai berikut.

a. **`POST "/query/interactive"`**

Endpoint ini adalah *placeholder* untuk *interactive query*. Mengembalikan pesan "Interactive query placeholder".

b. **`POST "/query/batch"`**

Endpoint ini adalah *placeholder* untuk *batch query*. Mengembalikan pesan "Batch query placeholder".

c. **`POST "/retrieve"`**

Endpoint ini digunakan untuk melakukan *retrieval* dokumen menggunakan *cached inverted file*. *Endpoint* ini dapat dipanggil jika sudah memanggil GET /inverted-file terlebih dahulu. Exception yang digunakan adalah *file* tidak tersedia (400), yaitu ketika *inverted file* belum dibuat. Endpoint ini akan menghasilkan peringkat dokumen, *average precision*, *total retrieved*, dan query yang digunakan.

d. **`GET "/inverted-file"`**

Membuat dan menyimpan sebuah *inverted file* berdasarkan dokumen "parsing_docs.json" yang telah di-*parsing*. Keluarannya mencakup *inverted file* yang telah disebutkan, serta jumlah dokumen, *term*, dan status *cache*. Ketika sudah menghasilkan *inverted file*, *file* tersebut disimpan di *cache*.

e. **`GET "/cache/status"`**

Memeriksa status *cache* untuk *inverted file*. Apabila *inverted file* masih terdapat di *cache*, kembaliannya adalah status tersedia jumlah *terms*, dan parameternya. Sedangkan jika tidak ada di *cache*, kembaliannya adalah status tidak tersedia, dengan catatan agar *user* mengunjungi *endpoint* `GET /inverted-file` dahulu.

f. **`DELETE "/inverted-file/cache"`**

Menghapus semua hal yang berhubungan dengan *inverted file* dari *cache*. Kembaliannya adalah status berhasil.

g. **`GET "/model-status"`**

Memeriksa status model *Word2Vec*. Kembalian *default*-nya adalah status siap dengan tambahan *vocabulary size*. Ada dua jenis *exception* yang digunakan, yaitu `HTTPException` untuk menangkap eror dari FastAPI dan `Exception` biasa untuk menangkap eror yang lebih umum.

h. **`POST "/retrieve-batch"`**

Melakukan *batch retrieval* dengan melibatkan *cached inverted file*. Hanya bisa digunakan ketika `GET /inverted-file` sudah dipanggil dahulu. Membuat *instance* dari *class* "RetrievalService" dan menggunakan fungsi `retrieve_document_batch_query()`, lalu memformat ulang keluaran dari fungsi tersebut sehingga menjadi JSON / *dictionary* sesuai dengan yang diharapkan untuk ditampilkan di *front-end*. *Exception* yang digunakan adalah *internal server error* (500).

i. **`GET "/document-weights/{document_id}"`**

Mengembalikan bobot (*weights*) seluruh *term* dalam sebuah dokumen dari *inverted* file yang telah dibuat sebelumnya. Untuk mengeksekusi *endpoint* ini, diperlukan pengecekan ketersediaan *inverted file*. Apabila tidak ditemukan *inverted file* dari sebuah dokumen, proses eksekusi dihentikan dengan mengembalikan pesan eror. Sedangkan, apabila ditemukan *inverted file* dari sebuah dokumen, proses eksekusi tetap berjalan. Oleh karena itu, *endpoint* ini bergantung pada *endpoint* `GET "/inverted-file".`

j. **`POST "/calculate-query-weight"`**

Endpoint ini digunakan untuk menghitung bobot setiap term pada query. Endpoint ini dapat ditembak jika sudah memanggil GET /inverted-file terlebih dahulu. *Endpoint* ini memiliki body request yang berisi query dan weighting_method. *Query* adalah *query text* yang akan dihitung bobotnya, sementara weighting_method adalah metode pembobotan yang digunakan untuk *query. Endpoint* ini mengembalikan *query*, vektor *query*, banyak kata, dan metode pembobotan yang digunakan.

C. **Deskripsi *Library***

Library digunakan untuk membangun sistem *Information Retrieval* yang komprehensif dengan fitur *query expansion*, *document retrieval*, dan *text preprocessing*. Setiap library memiliki peran khusus dalam mendukung fungsionalitas yang optimal.

***Backend Libraries*:**

1. FastAPI

   Framework web *modern* untuk membangun REST API yang cepat dan efisien. FastAPI menyediakan *automatic* API *documentation*, *type hints validation*, dan *asynchronous request handling* untuk endpoint *retrieval* dan *query expansion*.

2. Uvicorn

   ASGI *server* untuk menjalankan aplikasi FastAPI secara *production-ready*. Uvicorn mendukung *high-performance async*/*await operations* dan *concurrent request processing*.

3. Pydantic

   Library untuk *data validation* dan *settings management* menggunakan Python *type annotations*. Pydantic memastikan *input*/*output* data sesuai dengan *schema* yang didefinisikan pada model *request*/*response*.

4. Python-multipart

   Library untuk *handling multipart form data*, khususnya untuk file *upload operations*. Digunakan dalam endpoint *upload* dokumen dan *batch processing*.

5. NumPy

   Library fundamental untuk *scientific computing* dan *numerical operations*. NumPy mendukung operasi matematis pada *vector* dan *matrix* untuk perhitungan *similarity score* dan *term weighting*.

6. SciPy

   Library untuk *advanced mathematical functions* dan *scientific computing*. SciPy menyediakan algoritma optimisasi dan *statistical operations* untuk *information retrieval metrics*.

7. Gensim

Specialized library untuk topic modeling dan *document similarity analysis*. Gensim digunakan untuk implementasi Word2Vec model dalam *query expansion* dan *semantic similarity calculation*.

8. Scikit-learn

   Machine learning library untuk data *preprocessing*, *feature extraction*, dan *evaluation metrics*. Scikit-learn mendukung TF-IDF *calculation*, *cosine similarity*, dan *performance evaluation metrics* seperti MAP (*Mean Average Precision*).

9. NLTK

   Natural Language Processing toolkit untuk *text preprocessing operations*. NLTK menyediakan *tokenization*, *stemming* (Porter Stemmer), *stopword removal*, dan *corpus management* untuk bahasa Indonesia dan Inggris.

10. Pandas

    Data manipulation and *analysis library* untuk *handling structured data*. Pandas digunakan untuk *document parsing*, *batch query processing*, dan *data formatting* dalam operasi *retrieval*.

Frontend Libraries:

1. Next.js

   Framework React *full-stack* untuk membangun *interface* pengguna sistem *Information Retrieval*. Next.js menyediakan *server-side rendering*, *optimized bundling*, dan *routing system* untuk performa aplikasi *web* yang optimal dalam menampilkan hasil *retrieval* dan *query interface*.

2. Axios

   HTTP *client library* untuk komunikasi antara *frontend* dan *backend* API. Axios menangani semua *request* ke *endpoint* FastAPI *including document upload*, *query submission*, *retrieval requests*, dan *real-time communication* untuk operasi *Information Retrieval*.

3. Radix UI Components

   Collection of accessible UI components (Dialog, Select, Tabs, Tooltip, dll.) untuk membangun *interface* yang *user-friendly*. Radix UI mendukung *advanced search interface*, *filter options*, *document preview modals*, dan *accessibility features* untuk sistem *retrieval* yang *inclusive*.

4. Tailwind CSS

   Utility-first CSS *framework* untuk *styling* yang *consistent* dan *responsive*. Tailwind CSS memungkinkan *rapid development* dari *search interface*, *result layouts*, dan *responsive design* yang optimal untuk berbagai *device* dalam mengakses sistem *Information Retrieval*.

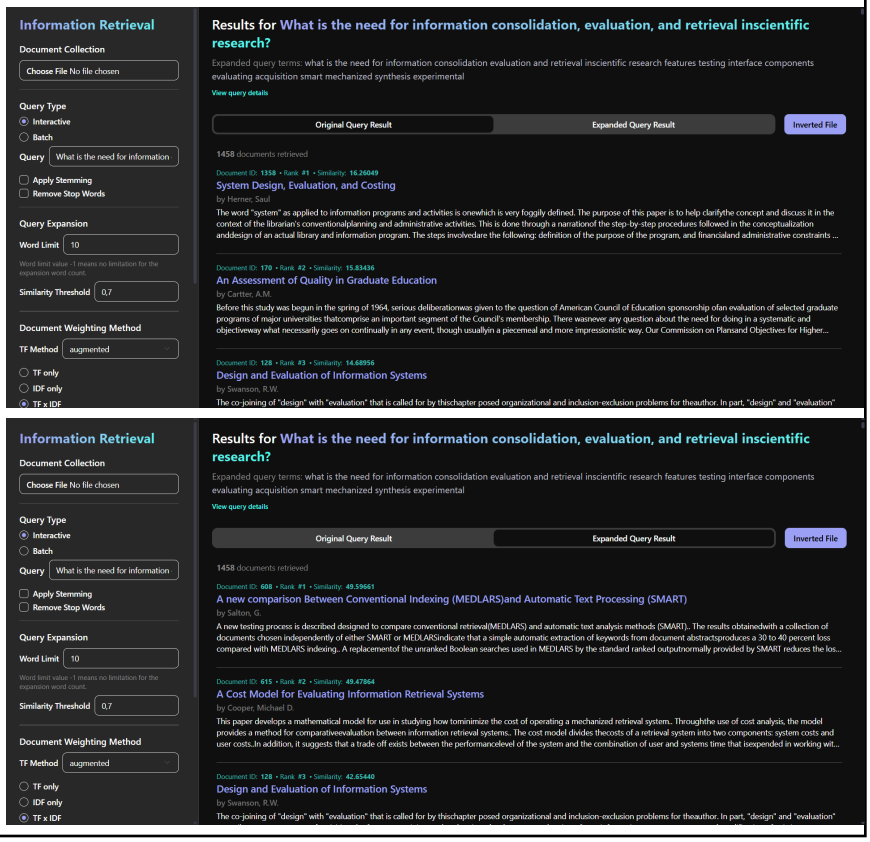
3. Hasil Eksperimen

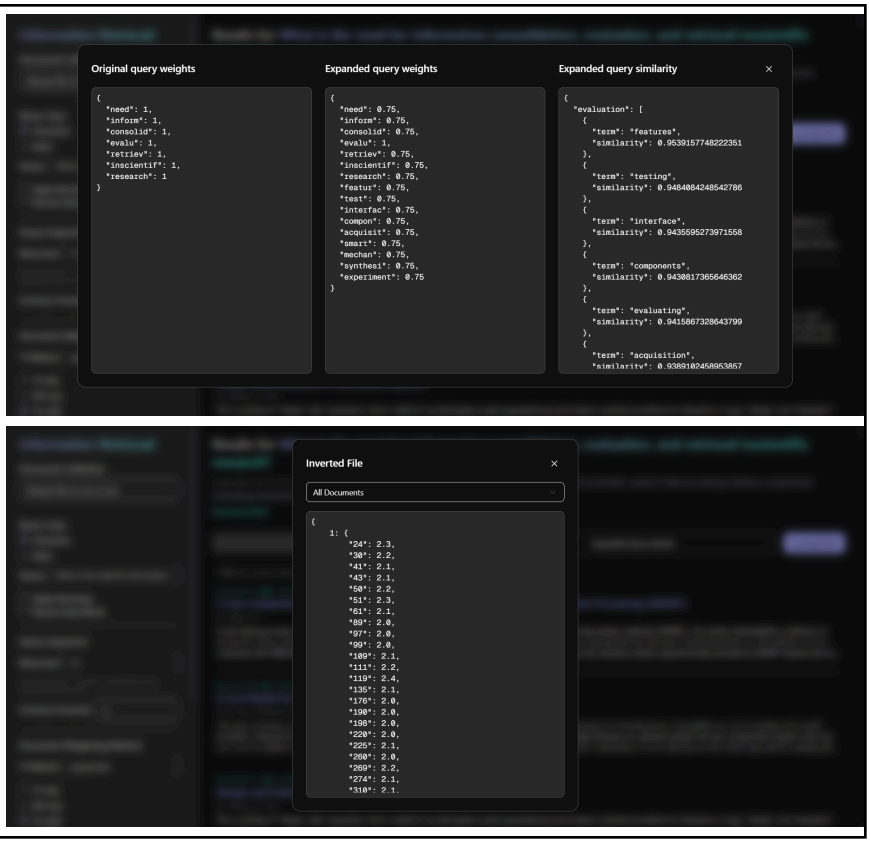
Tabel 3.1 Eksperimen 1 (interaktif)

Query asal	*What is the need for information consolidation, evaluation, and retrieval inscientific research?*
Stemming	False
Penghapusan *stop word*	False
Batas *query expansion*	10 kata
Metode pembobotan	TF (augmented) x IDF
Hasil *query expansion*	*what is the need for information consolidation evaluation and retrieval inscientific research features testing interface components evaluating acquisition smart mechanized synthesis experimental* { "evaluation": [{ "term": "features", "similarity": 0.9539157748222351 }, { "term": "testing", "similarity": 0.9484084248542786 }, { "term": "interface", "similarity": 0.9435595273971558

	},
	{
	"term": "components",
	"similarity": 0.9430817365646362
	},
	{
	"term": "evaluating",
	"similarity": 0.9415867328643799
	},
	{
	"term": "acquisition",
	"similarity": 0.9389102458953857
	},
	{
	"term": "smart",
	"similarity": 0.934873104095459
	},
	{
	"term": "mechanized",
	"similarity": 0.9342643022537231
	},
	{
	"term": "synthesis",
	"similarity": 0.9291064143180847
	},
	{
	"term": "experimental",
	"similarity": 0.9278411865234375
	}
	]
	}

Hasil *retrieval* dengan *query* asli	1458 documents retrieved Document ID: 1358 • Rank #1 • Similarity: 16.26049 System Design, Evaluation, and Costing by Herner, Saul The word "system" as applied to information programs and activities is onewhich is very foggily defined. The purpose of this paper is to help clarifythe concept and discuss it in the context of the librarian's conventionalplanning and administrative activities. This is done through a narrationof the step-by-step procedures followed in the conceptualization anddesign of an actual library

	and information program. The steps involvedare the following: definition of the purpose of the program, and financialand administrative constraints
Hasil *retrieval* dengan *expanded query*	1458 documents retrieved Document ID: 608 • Rank #1 • Similarity: 49.59661 A new comparison Between Conventional Indexing (MEDLARS)and Automatic Text Processing (SMART) by Salton, G. A new testing process is described designed to compare conventional retrieval(MEDLARS) and automatic text analysis methods (SMART).. The results obtainedwith a collection of documents chosen independently of either SMART or MEDLARSindicate that a simple automatic extraction of keywords from document abstractsproduces a 30 to 40 percent loss compared with MEDLARS indexing.. A replacementof the unranked Boolean searches used in MEDLARS by the standard ranked outputnormally provided by SMART reduces the lo
Screenshot	

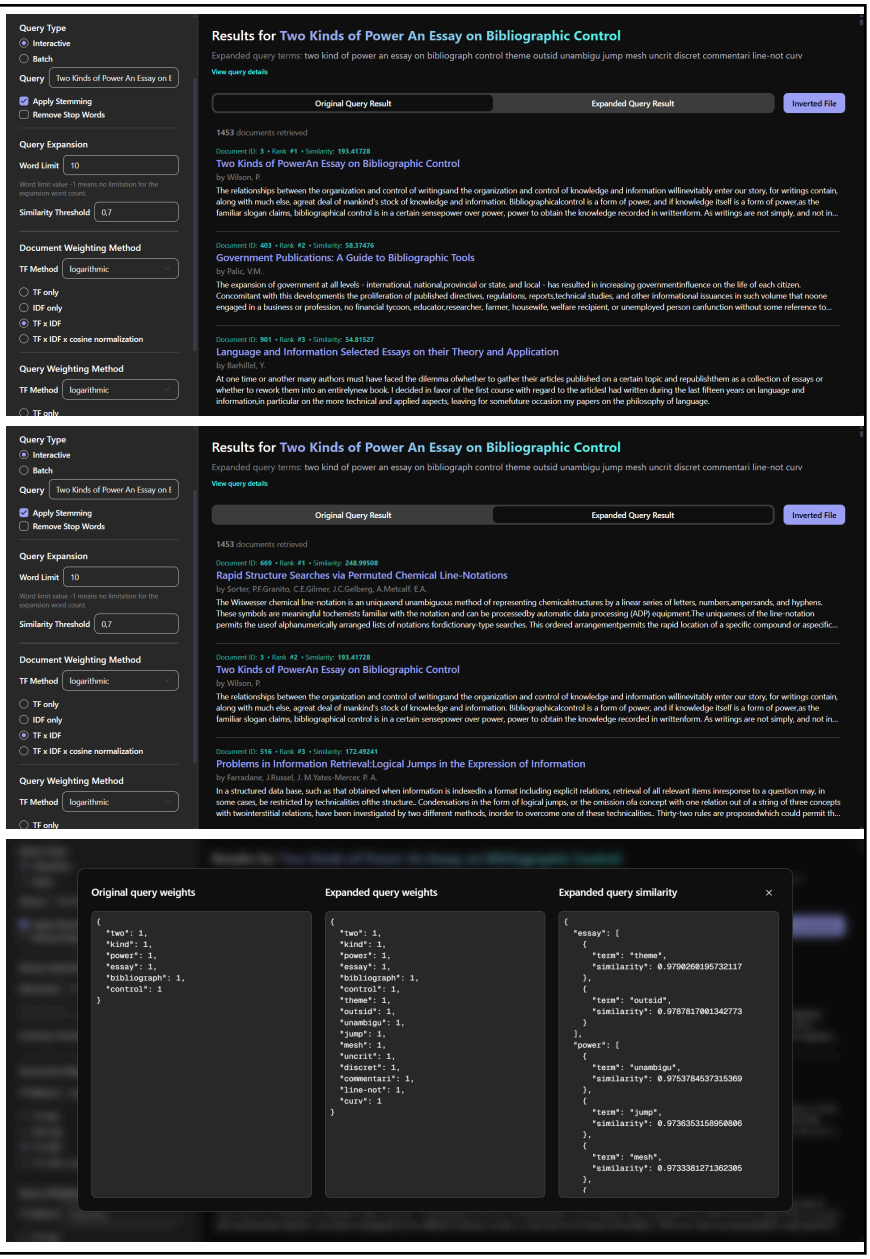
	

Tabel 3.2 Eksperimen 2 (interaktif)

Query asal	*Two Kinds of Power An Essay on Bibliographic Control*
Stemming	True
Penghapusan *stop word*	False
Batas *query expansion*	10 kata
Metode pembobotan	TF (logarithmic) x IDF

Hasil *query expansion*	*two kind of power an essay on bibliograph control theme outsid unambigu jump mesh uncrit discret commentari line-not curv*
	{
	"essay": [
	{
	"term": "theme",
	"similarity": 0.9790260195732117
	},
	{
	"term": "outsid",
	"similarity": 0.9787817001342773
	}
	],
	"power": [
	{
	"term": "unambigu",
	"similarity": 0.9753784537315369
	},
	{
	"term": "jump",
	"similarity": 0.9736353158950806
	},
	{
	"term": "mesh",
	"similarity": 0.9733381271362305
	},
	{
	"term": "uncrit",
	"similarity": 0.9729182720184326
	},
	{
	"term": "discret",
	"similarity": 0.9728000164031982
	},
	{
	"term": "commentari",
	"similarity": 0.9725228548049927
	},
	{
	"term": "line-not",
	"similarity": 0.9725039005279541
	},
	{
	"term": "curv",
	"similarity": 0.9724660515785217
	}

	] }
Hasil *retrieval* dengan *query* asli	1453 documents retrieved Document ID: 3 • Rank #1 • Similarity: 0.18350 Two Kinds of PowerAn Essay on Bibliographic Control by Wilson, P. The relationships between the organization and control of writingsand the organization and control of knowledge and information willinevitably enter our story, for writings contain, along with much else, agreat deal of mankind's stock of knowledge and information. Bibliographicalcontrol is a form of power, and if knowledge itself is a form of power,as the familiar slogan claims, bibliographical control is in a certain sensepower over power, power to obtain the knowledge recorded in writtenform. As writings are not simply, and not in
Hasil *retrieval* dengan *expanded query*	1453 documents retrieved Document ID: 669 • Rank #1 • Similarity: 248.99508 Rapid Structure Searches via Permuted Chemical Line-Notations by Sorter, P.F.Granito, C.E.Gilmer, J.C.Gelberg, A.Metcalf, E.A. The Wiswesser chemical line-notation is an uniqueand unambiguous method of representing chemicalstructures by a linear series of letters, numbers,ampersands, and hyphens. These symbols are meaningful tochemists familiar with the notation and can be processedby automatic data processing (ADP) equipment.The uniqueness of the line-notation permits the useof alphanumerically arranged lists of notations fordictionary-type searches. This ordered arrangementpermits the rapid location of a specific compound or aspecific

Screenshot	

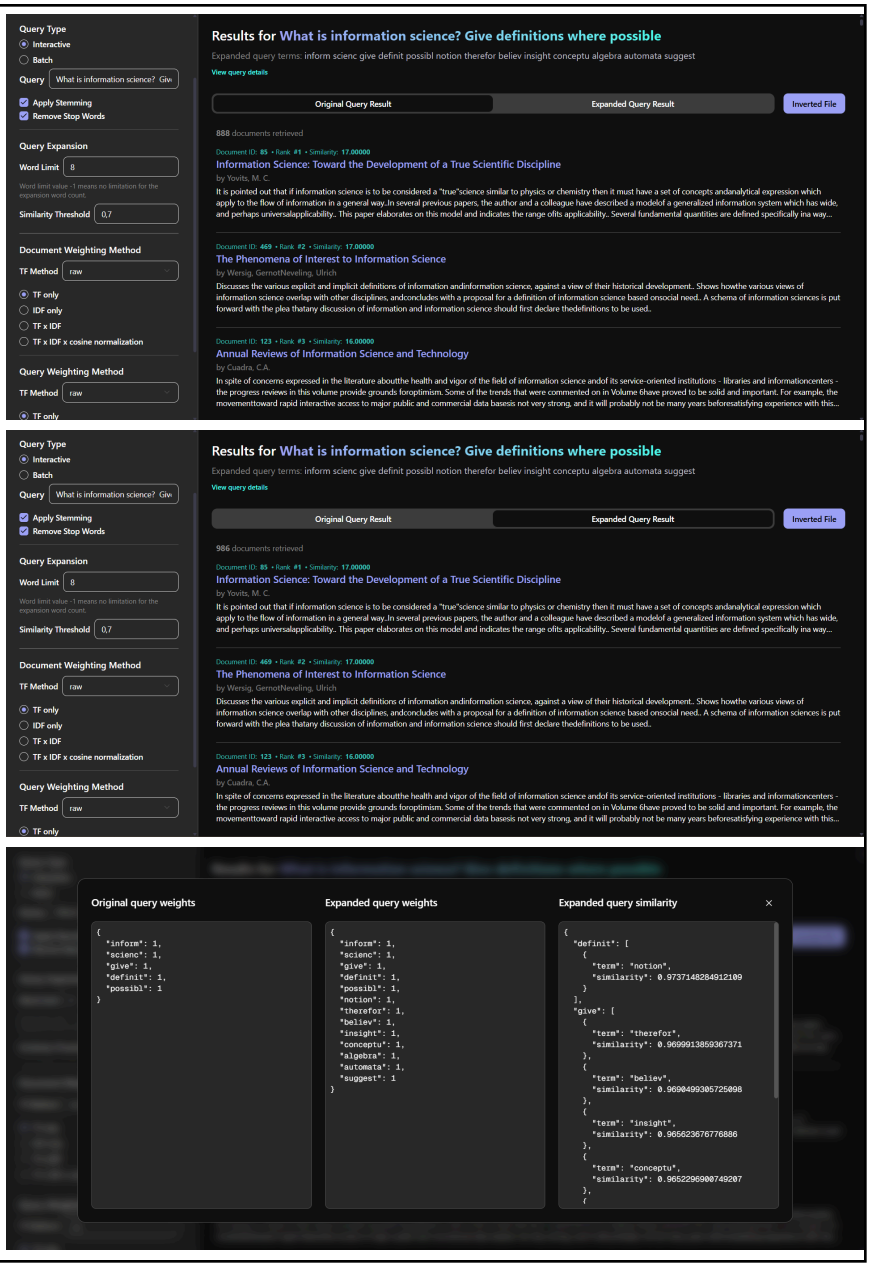
Tabel 3.3 Eksperimen 3 (interaktif)

Query asal	*What is information science? Give definitions where possible*
Stemming	True
Penghapusan *stop word*	True
Batas *query expansion*	8 kata
Metode pembobotan	TF (raw) only
Hasil *query expansion*	*inform scienc give definit possibl notion therefor believ insight conceptu algebra automata suggest*

	{
	"definit": [
	{
	"term": "notion",
	"similarity": 0.9737148284912109
	}
	],
	"give": [
	{
	"term": "therefor",
	"similarity": 0.9699913859367371
	},
	{
	"term": "believ",
	"similarity": 0.9690499305725098
	},
	{
	"term": "insight",
	"similarity": 0.965623676776886
	},
	{
	"term": "conceptu",
	"similarity": 0.9652296900749207
	},
	{
	"term": "algebra",
	"similarity": 0.9650352597236633
	},
	{
	"term": "automata",
	"similarity": 0.9638199806213379
	}
	],
	"possibl": [
	{
	"term": "suggest",
	"similarity": 0.9624667763710022
	}
	]
	}

Hasil *retrieval* dengan *query* asli	888 documents retrieved Document ID: 85 • Rank #1 • Similarity: 17.00000

	Information Science: Toward the Development of a True Scientific Discipline by Yovits, M. C. It is pointed out that if information science is to be considered a "true"science similar to physics or chemistry then it must have a set of concepts andanalytical expression which apply to the flow of information in a general way..In several previous papers, the author and a colleague have described a modelof a generalized information system which has wide, and perhaps universalapplicability.. This paper elaborates on this model and indicates the range ofits applicability.. Several fundamental quantities are defined specifically ina way which allows for quantification.. It is pointed out in this paper thatthis model can be the basis for the development of a "true" science ofinformation with all of the necessary requirements for a science.. By the useof this model and the definition of a "true" science, the goals and requirementsfor a curriculum in information science are thus established.. Within thiscontext, information is defined as data of value in decision making..Quantitative measures of information can be obtained by relating information tospecific observable actions which can be measured physically..
Hasil *retrieval* dengan *expanded query*	986 documents retrieved Document ID: 85 • Rank #1 • Similarity: 17.00000 Information Science: Toward the Development of a True Scientific Discipline by Yovits, M. C. It is pointed out that if information science is to be considered a "true"science similar to physics or chemistry then it must have a set of concepts andanalytical expression which apply to the flow of information in a general way..In several previous papers, the author and a colleague have described a modelof a generalized information system which has wide, and perhaps universalapplicability.. This paper elaborates on this model and indicates the range ofits applicability.. Several fundamental quantities are defined specifically ina way

Screenshot	

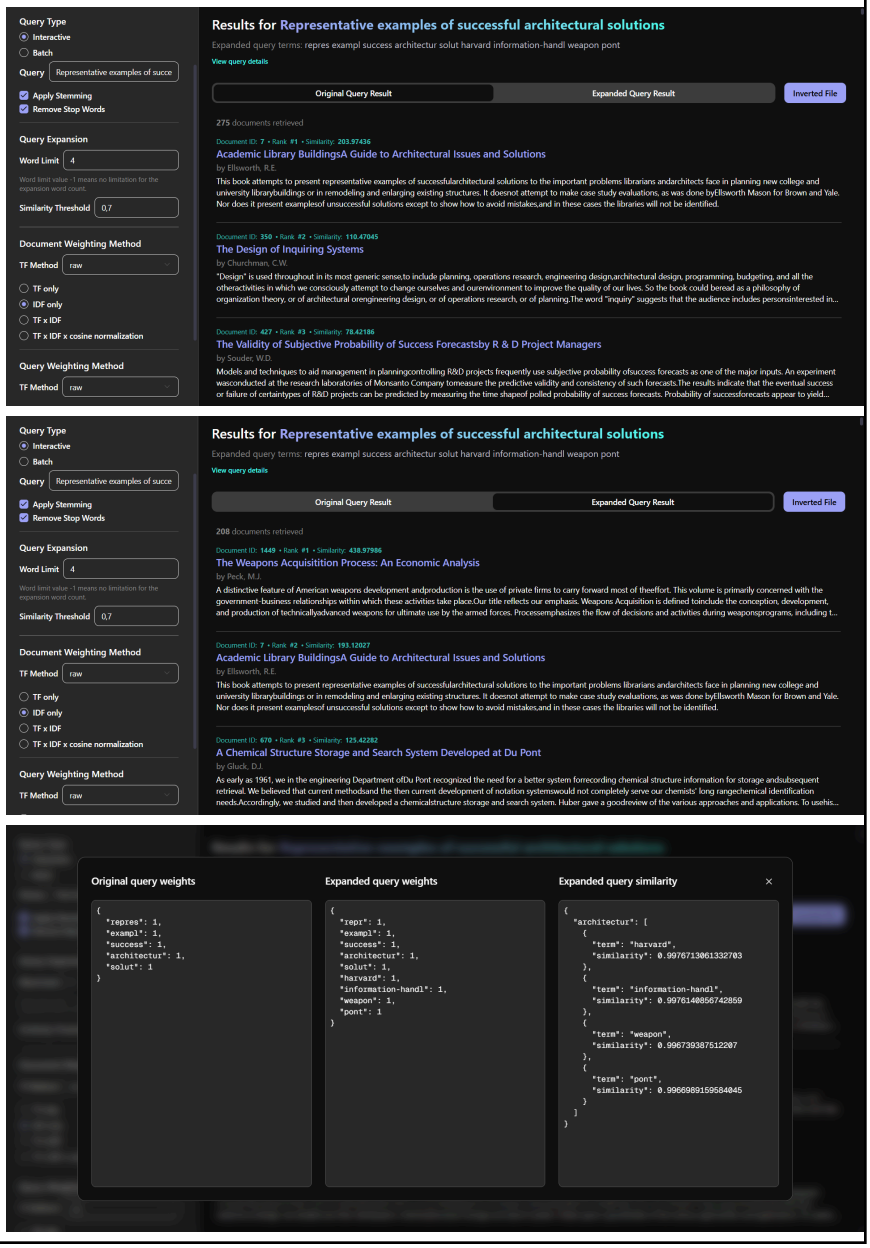
Tabel 3.4 Eksperimen 4 (interaktif)

Query asal	*Representative examples of successful architectural solutions*
Stemming	True
Penghapusan *stop word*	True
Batas *query expansion*	4 kata
Metode pembobotan	IDF only
Hasil *query expansion*	*repres exampl success architectur solut harvard information-handl weapon pont* {

	"architectur": [
	{
	"term": "harvard",
	"similarity": 0.9976713061332703
	},
	{
	"term": "information-handl",
	"similarity": 0.9976140856742859
	},
	{
	"term": "weapon",
	"similarity": 0.996739387512207
	},
	{
	"term": "pont",
	"similarity": 0.9966989159584045
	}
	]
	}

Hasil *retrieval* dengan *query* asli	275 documents retrieved Document ID: 7 • Rank #1 • Similarity: 203.97436 Academic Library BuildingsA Guide to Architectural Issues and Solutions by Ellsworth, R.E. This book attempts to present representative examples of successfularchitectural solutions to the important problems librarians andarchitects face in planning new college and university librarybuildings or in remodeling and enlarging existing structures. It doesnot attempt to make case study evaluations, as was done byEllsworth Mason for Brown and Yale. Nor does it present examplesof unsuccessful solutions except to show how to avoid mistakes,and in these cases the libraries will not be identified.
Hasil *retrieval* dengan *expanded query*	208 documents retrieved Document ID: 1449 • Rank #1 • Similarity: 438.97986 The Weapons Acquisitition Process: An Economic Analysis by Peck, M.J.

	A distinctive feature of American weapons development andproduction is the use of private firms to carry forward most of theeffort. This volume is primarily concerned with the government-business relationships within which these activities take place.Our title reflects our emphasis. Weapons Acquisition is defined toinclude the conception, development, and production of technicallyadvanced weapons for ultimate use by the armed forces. Processemphasizes the flow of decisions and activities during weaponsprograms, including the

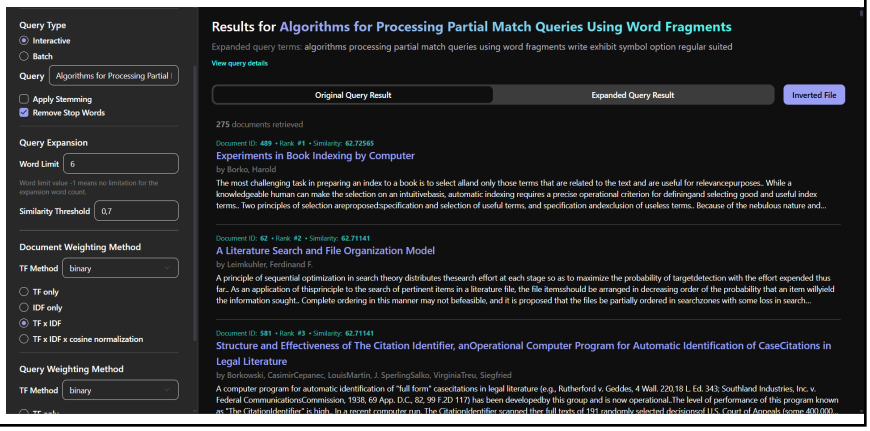
Screenshot	 Query Type
	Interactive
	Batch
	Query
	Representative examples of succe
	Apply Stemming
	Remove Stop Words
	Query Expansion
	Word Limit
	4
	Similarity Threshold
	0,7
	Document Weighting Method
	TF Method
	raw
	TF only
	IDF only
	TF x IDF
	TF x IDF x cosine normalization
	Query Weighting Method
	TF Method
	raw
	Results for Representative examples of successful architectural solutions
	Expanded query terms: repres exampl success architectur solut harvard information-handl weapon pont
	View query details
	Original Query Result
	Expanded Query Result
	Inverted File
	275 documents retrieved
	Document ID: 7 • Rank: #1 • Similarity: 203.97436
	Academic Library BuildingsA Guide to Architectural Issues and Solutions
	by Ellsworth, R.E.
	This book attempts to present representative examples of successfularchitectural solutions to the important problems librarians andarchitects face in planning new college and university librarybuildings or in remodeling and enlarging existing structures. It doesnot attempt to make case study evaluations, as was done byEllsworth Mason for Brown and Yale. Nor does it present examplesof unsuccessful solutions except to show how to avoid mistakes,and in these cases the libraries will not be identified.
	Document ID: 350 • Rank: #2 • Similarity: 110.47045
	The Design of Inquiring Systems
	by Churchman, C.W.
	"Design" is used throughout in its most generic sense,to include planning, operations research, engineering design,architectural design, programming, budgeting, and all the otheractivities in which we consciously attempt to change ourselves and ourenvironment to improve the quality of our lives. So the book could beread as a philosophy of organization theory, or of architectural orengineering design, or of operations research, or of planning.The word "inquiry" suggests that the audience includes personsinterested in...
	Document ID: 427 • Rank: #3 • Similarity: 78.42186
	The Validity of Subjective Probability of Success Forecastsby R & D Project Managers
	by Souder, W.D.
	Models and techniques to aid management in planningcontrolling R&D projects frequently use subjective probability ofsuccess forecasts as one of the major inputs. An experiment wasconducted at the research laboratories of Monsanto Company tomeasure the predictive validity and consistency of such forecasts.The results indicate that the eventual success or failure of certaintypes of R&D projects can be predicted by measuring the time shapeof polled probability of success forecasts. Probability of successforecasts appear to yield...
	Query Type
	Interactive
	Batch
	Query
	Representative examples of succe
	Apply Stemming
	Remove Stop Words
	Query Expansion
	Word Limit
	4
	Similarity Threshold
	0,7
	Document Weighting Method
	TF Method
	raw
	TF only
	IDF only
	TF x IDF
	TF x IDF x cosine normalization
	Query Weighting Method
	TF Method
	raw
	Results for Representative examples of successful architectural solutions
	Expanded query terms: repres exampl success architectur solut harvard information-handl weapon pont
	View query details
	Original Query Result
	Expanded Query Result
	Inverted File
	208 documents retrieved
	Document ID: 1449 • Rank: #1 • Similarity: 438.97986
	The Weapons Acquisition Process: An Economic Analysis
	by Peck, M.J.
	A distinctive feature of American weapons development andproduction is the use of private firms to carry forward most of theeffort. This volume is primarily concerned with the government-business relationships within which these activities take place.Our title reflects our emphasis. Weapons Acquisition is defined toinclude the conception, development, and production of technicallyadvanced weapons for ultimate use by the armed forces. Processemphasizes the flow of decisions and activities during weaponsprograms, including t...
	Document ID: 7 • Rank: #2 • Similarity: 193.12027
	Academic Library BuildingsA Guide to Architectural Issues and Solutions
	by Ellsworth, R.E.
	This book attempts to present representative examples of successfularchitectural solutions to the important problems librarians andarchitects face in planning new college and university librarybuildings or in remodeling and enlarging existing structures. It doesnot attempt to make case study evaluations, as was done byEllsworth Mason for Brown and Yale. Nor does it present examplesof unsuccessful solutions except to show how to avoid mistakes,and in these cases the libraries will not be identified.
	Document ID: 670 • Rank: #3 • Similarity: 125.42282
	A Chemical Structure Storage and Search System Developed at Du Pont
	by Gluck, D.J.
	As early as 1961, we in the engineering Department ofDu Pont recognized the need for a better system forrecording chemical structure information for storage andsubsequent retrieval. We believed that current methodsand the then current development of notation systemswould not completely serve our chemists' long rangechemical identification needs.Accordingly, we studied and then developed a chemicalstructure storage and search system. Huber gave a goodreview of the various approaches and applications. To usehis...
	Original query weights
	{
	"repres": 1,
	"exampl": 1,
	"success": 1,
	"architectur": 1,
	"solut": 1
	}
	Expanded query weights
	{
	"repr": 1,
	"exampl": 1,
	"success": 1,
	"architectur": 1,
	"solut": 1,
	"harvard": 1,
	"information-handl": 1,
	"weapon": 1,
	"pont": 1
	}
	Expanded query similarity
	{
	"architectur": [
	{
	"term": "harvard",
	"similarity": 0.9976713061332703
	},
	{
	"term": "information-handl",
	"similarity": 0.9976140856742859
	},
	{
	"term": "weapon",
	"similarity": 0.996739387512207
	},
	{
	"term": "pont",
	"similarity": 0.9966989159584045
	}
	]
	}

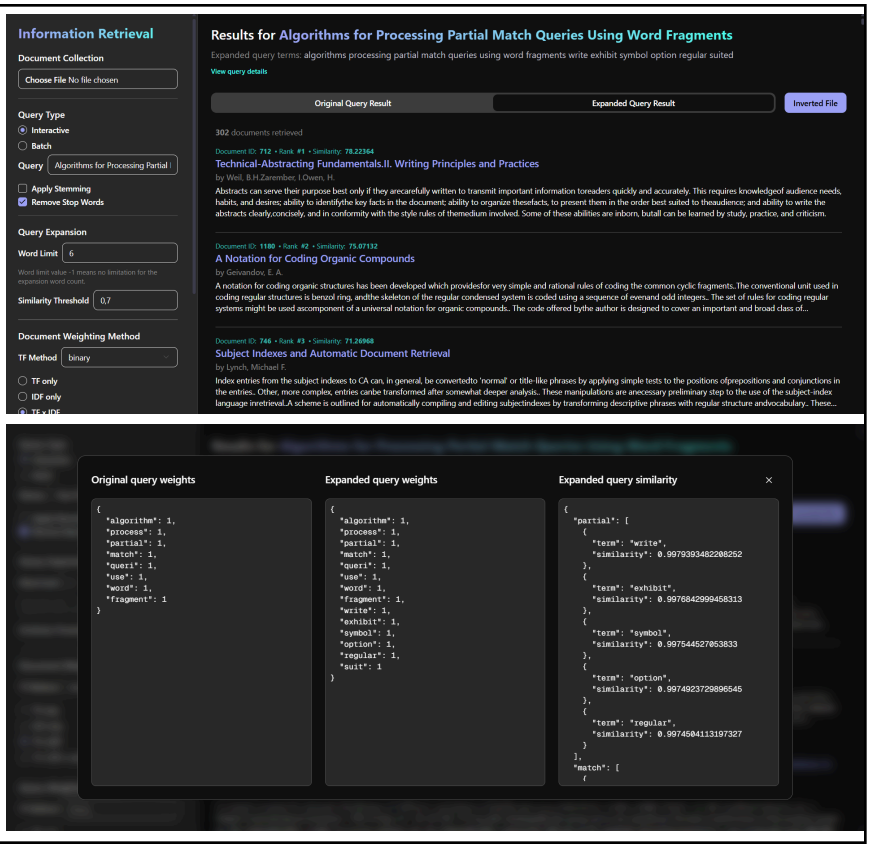
Tabel 3.5 Eksperimen 5 (interaktif)

***Query* asal**	*Algorithms for Processing Partial Match Queries Using Word Fragments*
Stemming	False
Penghapusan *stop word*	True
Batas *query expansion*	6 kata
Metode pembobotan	TF (binary) x IDF

Hasil *query expansion*	*algorithms processing partial match queries using word fragments write exhibit symbol option regular suited*
	
	"partial": [
	{
	"term": "write",
	"similarity": 0.9979393482208252
	},
	{
	"term": "exhibit",
	"similarity": 0.9976842999458313
	},
	{
	"term": "symbol",
	"similarity": 0.997544527053833
	},
	{
	"term": "option",
	"similarity": 0.9974923729896545
	},
	{
	"term": "regular",
	"similarity": 0.9974504113197327
	}
	],
	"match": [
	{
	"term": "suited",
	"similarity": 0.9973295331001282
	}
	]
	}

Hasil *retrieval* dengan *query* asli	275 documents retrieved Document ID: 489 • Rank #1 • Similarity: 62.72565

	Experiments in Book Indexing by Computer by Borko, Harold The most challenging task in preparing an index to a book is to select alland only those terms that are related to the text and are useful for relevancepurposes.. While a knowledgeable human can make the selection on an intuitivebasis, automatic indexing requires a precise operational criterion for definingand selecting good and useful index terms.. Two principles of selection areproposed:specification and selection of useful terms, and specification andexclusion of useless terms.. Because of the nebulous nature and
Hasil *retrieval* dengan *expanded query*	302 documents retrieved Document ID: 712 • Rank #1 • Similarity:78.22364 Technical-Abstracting Fundamentals.II. Writing Principles and Practices by Weil, B.H.Zarember, I.Owen, H. Abstracts can serve their purpose best only if they arecarefully written to transmit important information toreaders quickly and accurately. This requires knowledgeof audience needs, habits, and desires; ability to identifythe key facts in the document; ability to organize thesefacts, to present them in the order best suited to theaudience; and ability to write the abstracts clearly,concisely, and in conformity with the style rules of themedium involved. Some of these abilities are inborn, butall can be learned by study, practice, and criticism.
Screenshot	

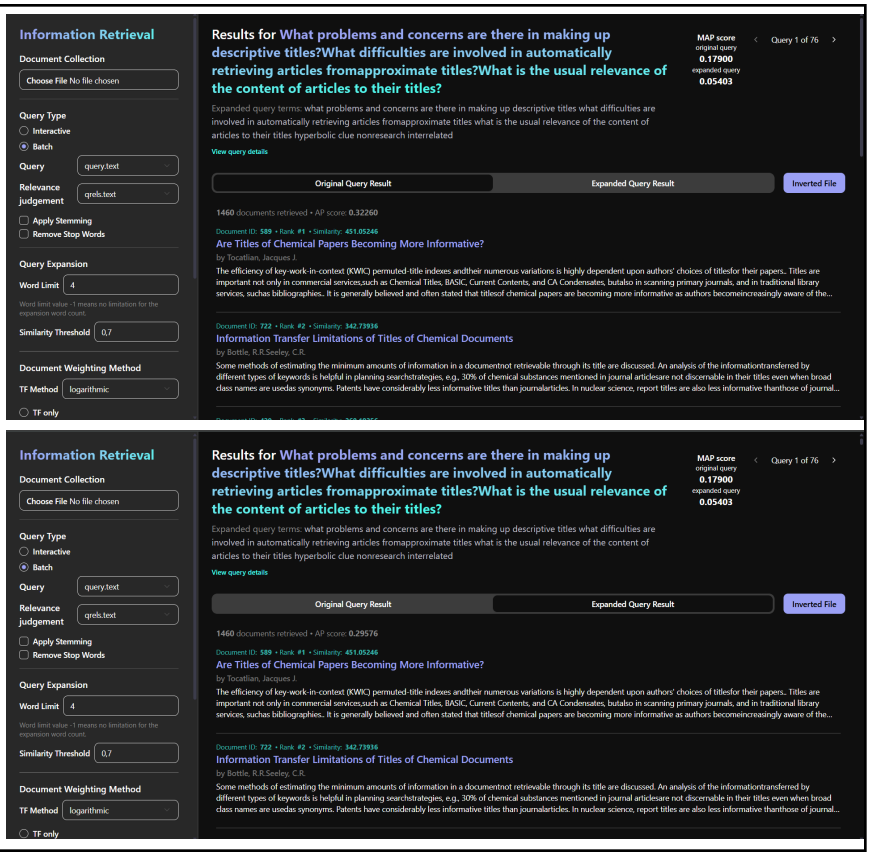
	

Tabel 3.6 Eksperimen 6 (*batch*)

File query	query.text
***Query* asal**	*What problems and concerns are there in making up descriptive titles?What difficulties are involved in automatically retrieving articles fromapproximate titles?What is the usual relevance of the content of articles to their titles?*
Stemming	False
Penghapusan *stop word*	False
Batas *query expansion*	4 kata
Metode pembobotan	IDF only
Hasil *query expansion*	*what problems and concerns are there in making up descriptive titles what difficulties are involved in automatically retrieving articles fromapproximate titles what is the usual relevance of the content of articles to their titles hyperbolic clue nonresearch interrelated* { "retrieving": [{ "term": "hyperbolic",

	"similarity": 0.9933832287788391
	},
	{
	"term": "nonresearch",
	"similarity": 0.9921275973320007
	},
	{
	"term": "interrelated",
	"similarity": 0.9919673800468445
	}
	],
	"usual": [
	{
	"term": "clue",
	"similarity": 0.9921396374702454
	}
	]
	}

Hasil *retrieval* dengan *query* asli	MAP = 0.17900 1460 documents retrieved AP score: 0.32260 Document ID: 589 • Rank #1 • Similarity: 451.05246 Are Titles of Chemical Papers Becoming More Informative? by Tocatlian, Jacques J.
Hasil *retrieval* dengan *expanded query*	MAP = 0.05403 1460 documents retrieved AP score: 0.29576 Document ID: 589 • Rank #1 • Similarity: 451.05246 Are Titles of Chemical Papers Becoming More Informative? by Tocatlian, Jacques J.

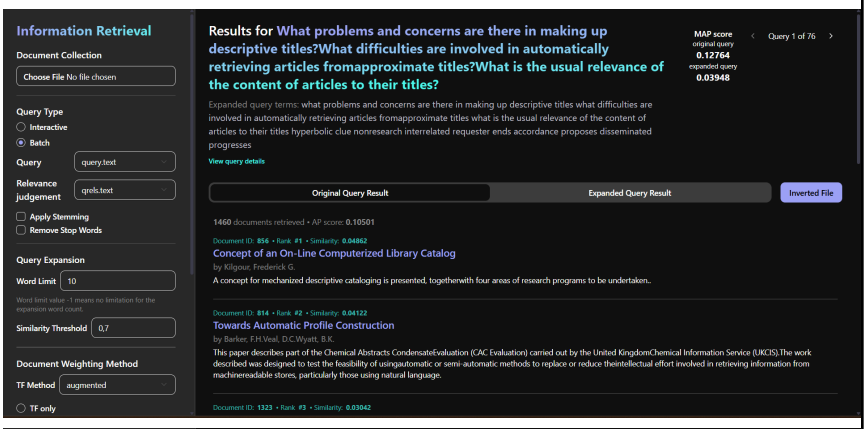
Screenshot	

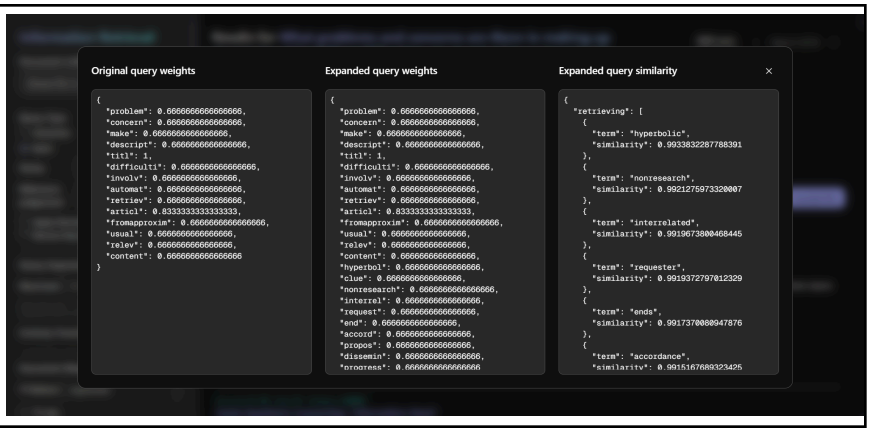
Tabel 3.7 Eksperimen 7 (*batch*)

File query	query.text
***Query* asal**	*What problems and concerns are there in making up descriptive titles?What difficulties are involved in automatically retrieving articles fromapproximate titles?What is the usual relevance of the content of articles to their titles?*
Stemming	False
Penghapusan *stop word*	False
Batas *query expansion*	10 kata
Metode pembobotan	TF (augmented) x IDF x cosine normalization
Hasil *query expansion*	*what problems and concerns are there in making up descriptive titles what difficulties are involved in automatically retrieving articles fromapproximate titles what is the usual relevance of the content of articles to their titles hyperbolic clue nonresearch interrelated requester ends accordance proposes disseminated progresses* { "retrieving": [{

	"term": "hyperbolic",
	"similarity": 0.9933832287788391
	},
	{
	"term": "nonresearch",
	"similarity": 0.9921275973320007
	},
	{
	"term": "interrelated",
	"similarity": 0.9919673800468445
	},
	{
	"term": "requester",
	"similarity": 0.9919372797012329
	},
	{
	"term": "ends",
	"similarity": 0.9917370080947876
	},
	{
	"term": "accordance",
	"similarity": 0.9915167689323425
	},
	{
	"term": "proposes",
	"similarity": 0.9914995431900024
	},
	{
	"term": "progresses",
	"similarity": 0.9913545846939087
	}
	],
	"usual": [
	{
	"term": "clue",
	"similarity": 0.9921396374702454
	},
	{
	"term": "disseminated",
	"similarity": 0.9914236664772034
	}
	]
	}

Hasil *retrieval* dengan *query* asli	MAP = 0.12764 1460 documents retrieved

	AP score: 0.10501 Document ID: 856 • Rank #1 • Similarity: 0.04862 Concept of an On-Line Computerized Library Catalog by Kilgour, Frederick G.
Hasil *retrieval* dengan *expanded query*	MAP = 0.03948 1460 documents retrieved AP score: 0.08687 Document ID: 1138 • Rank #1 • Similarity: 0.03921 Relevance and Pertinence by Polushkin, V.A.
Screenshot	

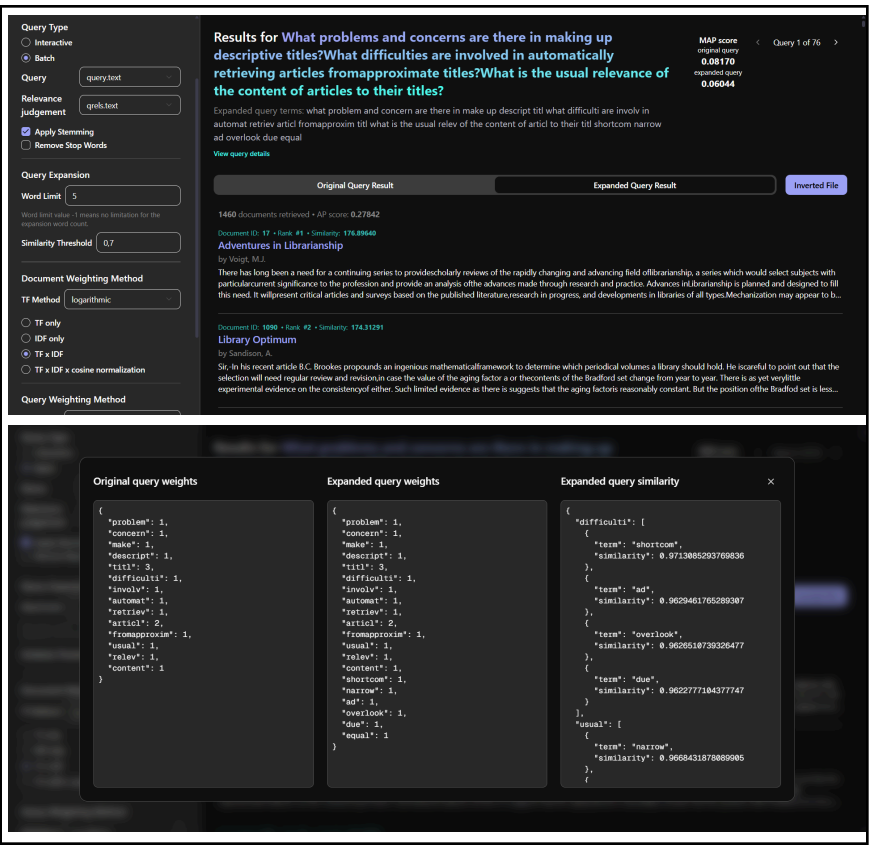
Tabel 3.8 Eksperimen 8 (*batch*)

File query	query.text
***Query* asal**	*What problems and concerns are there in making up descriptive titles?What difficulties are involved in automatically retrieving articles fromapproximate titles?What is the usual relevance of the content of articles to their titles?*
Stemming	True
Penghapusan *stop word*	False
Batas *query expansion*	5 kata
Metode pembobotan	TF (logarithmic) x IDF

Hasil *query expansion*	{
	"difficulti": [
	{
	"term": "shortcom",
	"similarity": 0.9713085293769836
	},
	{
	"term": "ad",
	"similarity": 0.9629461765289307
	},
	{
	"term": "overlook",
	"similarity": 0.9626510739326477
	},
	{
	"term": "due",
	"similarity": 0.9622777104377747
	}

	],
	"usual": [
	{
	"term": "narrow",
	"similarity": 0.9668431878089905
	},
	{
	"term": "equal",
	"similarity": 0.9618819952011108
	}
	]
	}

Hasil *retrieval* dengan *query* asli	MAP = 0.08170 1460 documents retrieved AP score: 0.02883 Document ID: 1295 • Rank #1 • Similarity: 0.49505 Communication or Chaos? by Baker, D.B.
Hasil *retrieval* dengan *expanded query*	MAP = 0.06044 1460 documents retrieved AP score: 0.27842 Document ID: 17 • Rank #1 • Similarity: 176.89640 Adventures in Librarianship by Voigt, M.J.
Screenshot	

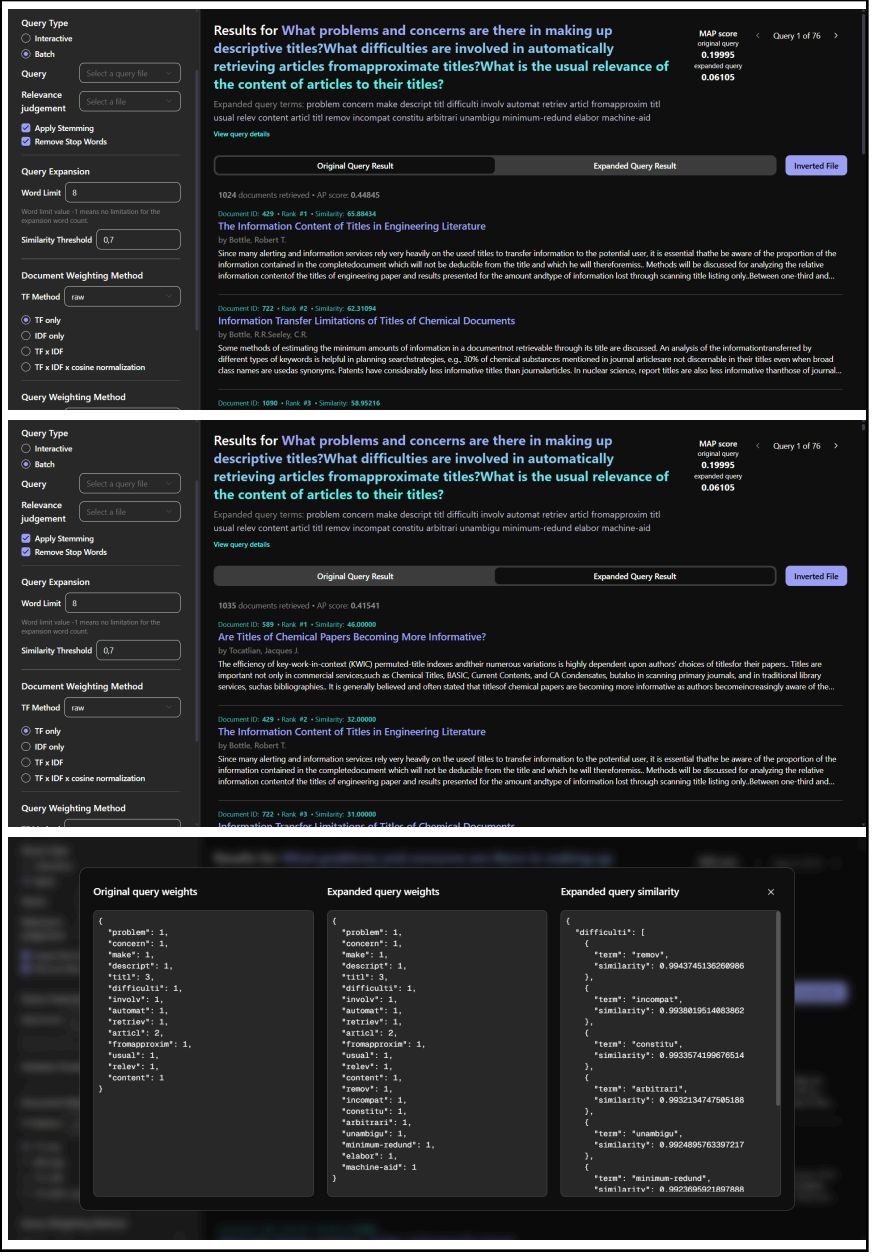
	

Tabel 3.9 Eksperimen 9 (*batch*)

File query	query.text
***Query* asal**	*What problems and concerns are there in making up descriptive titles?What difficulties are involved in automatically retrieving articles fromapproximate titles?What is the usual relevance of the content of articles to their titles?*
Stemming	True
Penghapusan *stop word*	True
Batas *query expansion*	8 kata
Metode pembobotan	TF (raw) only
Hasil *query expansion*	*problem concern make descript titl difficulti involv automat retriev articl fromapproxim titl usual relev content articl titl remov incompat constitu arbitrari unambigu minimum-redund elabor machine-aid* { "difficulti": [{ "term": "remov", "similarity": 0.9943745136260986

	},
	{
	"term": "incompat",
	"similarity": 0.9938019514083862
	},
	{
	"term": "constitu",
	"similarity": 0.9933574199676514
	},
	{
	"term": "arbitrari",
	"similarity": 0.9932134747505188
	},
	{
	"term": "unambigu",
	"similarity": 0.9924895763397217
	},
	{
	"term": "minimum-redund",
	"similarity": 0.9923695921897888
	},
	{
	"term": "elabor",
	"similarity": 0.9923436641693115
	},
	{
	"term": "machine-aid",
	"similarity": 0.9920647144317627
	}
	]
	}

Hasil *retrieval* dengan *query* asli	MAP = 0.19995 1024 documents retrieved AP score: 0.44845 Document ID: 429 • Rank #1 • Similarity: 65.88434 The Information Content of Titles in Engineering Literature by Bottle, Robert T.
Hasil *retrieval* dengan *expanded query*	MAP = 0.06105 1035 documents retrieved AP score: 0.41541

	Document ID: 589 • Rank #1 • Similarity: 46.00000 Are Titles of Chemical Papers Becoming More Informative? by Tocatlian, Jacques J.
Screenshot	

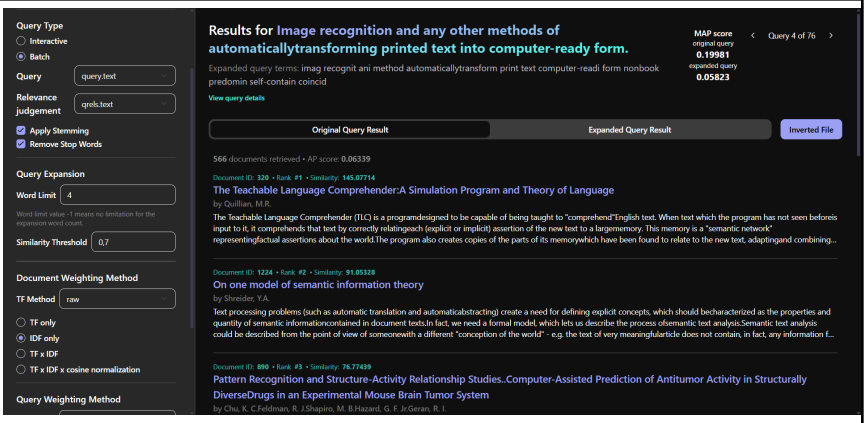
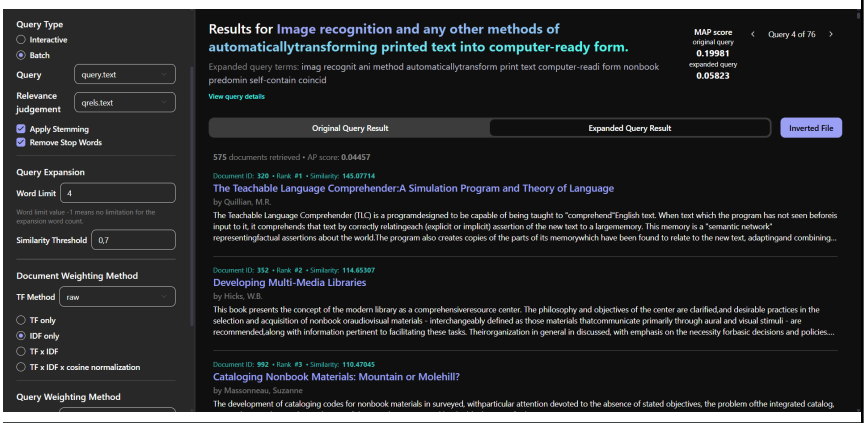
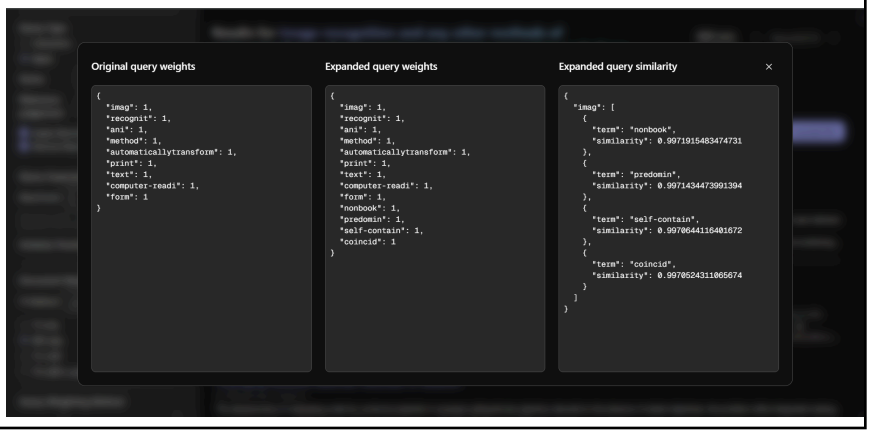
Tabel 3.10 Eksperimen 10 (*batch*)

File query	query.text

***Query* asal**	*Image recognition and any other methods of automaticallytransforming printed text into computer-ready form.*
Stemming	True
Penghapusan *stop word*	True
Batas *query expansion*	4 kata
Metode pembobotan	IDF only

Hasil *query expansion*	*imag recognit ani method automaticallytransform print text computer-readi form nonbook predomin self-contain coincid*
	{
	"imag": [
	{
	"term": "nonbook",
	"similarity": 0.9971915483474731
	},
	{
	"term": "predomin",
	"similarity": 0.9971434473991394
	},
	{
	"term": "self-contain",
	"similarity": 0.9970644116401672
	},
	{
	"term": "coincid",
	"similarity": 0.9970524311065674
	}
	]
	}

Hasil *retrieval* dengan *query* asli	MAP = 0.19981 566 documents retrieved AP score: 0.06339 Document ID: 320 • Rank #1 • Similarity:145.07714 The Teachable Language Comprehender:A Simulation Program and Theory of Language by Quillian, M.R.
Hasil *retrieval* dengan *expanded query*	MAP = 0.05823 575 documents retrieved AP score: 0.04457

	Document ID: 320 • Rank #1 • Similarity: 145.07714 The Teachable Language Comprehender:A Simulation Program and Theory of Language by Quillian, M.R.
Screenshot	  

4. Analisis Hasil Eksperimen

Berdasarkan hasil eksperimen pada bagian sebelumnya, dapat disimpulkan bahwa skor MAP untuk *expanded query* lebih rendah daripada skor MAP untuk *query* asli. Secara teoretis, *query expansion* seharusnya meningkatkan nilai skor MAP. Faktor yang menyebabkan penurunan nilai skor MAP adalah kurang akuratnya model Word2Vec yang digunakan untuk melakukan *query expansion*. Seperti yang dapat diamati pada hasil eksperimen, misalnya Eksperimen 6, kata-kata yang ditambahkan pada *query expansion* kurang memiliki kesinambungan secara makna dengan kata-kata pada *query* asli. Hal ini disebabkan oleh mekanisme Word2Vec yang melihat kemiripan kata berdasarkan kemunculan dari pasangan kata di dalam *window* yang berukuran tertentu. Konfigurasi dari ukuran *window* memengaruhi akurasi dari Word2Vec dalam menentukan kemiripan dari pasangan kata. Ada kemungkinan bahwa pemilihan ukuran *window* dari Word2Vec kurang tepat, sehingga kata-kata yang muncul bersamaan kurang menggambarkan hubungan antarkata tersebut. Kemungkinan lainnya adalah koleksi dokumen yang digunakan untuk melatih model Word2Vec kurang baik, sehingga kata-kata yang memperoleh kemiripan tinggi hanya menggambarkan kemunculan kata secara bersamaan yang sering muncul, bukan menunjukkan kemiripan kata secara arti.

LAMPIRAN

Lampiran A: Kode Program

- Repositori *Front-End*: https://github.com/arleenchr/IR-System-FE
- Repositori *Back-End*: https://github.com/AustinPardosi/IR-System-BE

Lampiran B: Panduan Penggunaan Aplikasi

Aplikasi terdiri dari dua komponen, yaitu *front-end* dan *back-end*. Keduanya harus dijalankan pada *localhost*.

Panduan untuk menjalankan *back-end*

1. Menggunakan *virtual environment*
    a. Jalankan perintah berikut untuk membuat *virtual environment*.

```
# Windows
python -m venv venv
source venv/Scripts/activate

# Linux/Mac
python3 -m venv venv
source venv/bin/activate
```

    b. Jalankan perintah berikut untuk melakukan instalasi *dependency*.

```
pip install -r requirements.txt
```

2. Menggunakan *docker*
    a. Jalankan perintah berikut untuk memulai layanan *back-end*.

```
# Menjalankan docker
docker-compose up

# Menjalankan docker di background
docker-compose up -d
```

    b. Jalankan perintah berikut untuk menghentikan layanan *back-end*.

```
docker-compose down
```

3. Setelah dijalankan, layanan *back-end* dapat diakses melalui https://localhost:8080.

Panduan untuk menjalankan *front-end*

1. Melakukan instalasi *library* dan *dependency* sebelum memulai dengan menjalankan perintah berikut.

```
npm install
```

2. Membuat konfigurasi *environment* dengan membuat *file* dengan nama `.env` pada direktori *root*. Adapun isi *file* adalah sebagai berikut.

```
API_BASE_URL=http://localhost:8080/api
```

3. Menjalankan aplikasi pada lingkungan pengembangan (*dev*) dengan perintah berikut.

```
npm run dev
```

4. Aplikasi web dapat diakses melalui https://localhost:3000.

Panduan untuk menggunakan aplikasi

Antarmuka aplikasi hanya terdiri dari satu halaman yang terdiri dari dua bagian utama, yaitu *toolbar* pada bagian kiri dan bagian untuk menampilkan hasil *retrieval* dokumen (lihat gambar C.1).

Toolbar menyediakan tempat untuk memasukkan *input*, seperti yang ditunjukkan Gambar C.2. Beberapa masukan yang dibutuhkan untuk melakukan *retrieval* adalah sebagai berikut.

1. Koleksi dokumen, berupa *file* yang memuat seluruh dokumen.
2. Jenis *query* (interaktif dan *batch*),
    a. Teks *query* (jika interaktif)
    b. *File query* beserta *file relevance judgement* (jika *batch*)
3. Pilihan untuk menerapkan *stemming* dan penghapusan *stop words*
4. Pengaturan untuk *query expansion*

    a. Batas (*limit*) banyaknya kata yang akan ditambahkan pada *query expansion*
    b. Nilai ambang kemiripan (*similarity threshold*) untuk kata yang akan ditambahkan pada *query expansion*
5. Metode pembobotan dokumen
    a. Metode TF, yaitu *logarithmic*, *raw*, *binary*, atau *augmented*
    b. Pilihan kalkulasi pembobotan: TF saja, IDF saja, TF x IDF, atau TF x IDF x *cosine normalization*
6. Metode pembobotan *query*
    a. Metode TF, yaitu *logarithmic*, *raw*, *binary*, atau *augmented*
    b. Pilihan kalkulasi pembobotan: TF saja, IDF saja, TF x IDF, atau TF x IDF x *cosine normalization*

Setelah memastikan semua *input field* pada *toolbar* terisi, klik tombol *Retrieve* untuk mulai melakukan *document retrieval*.

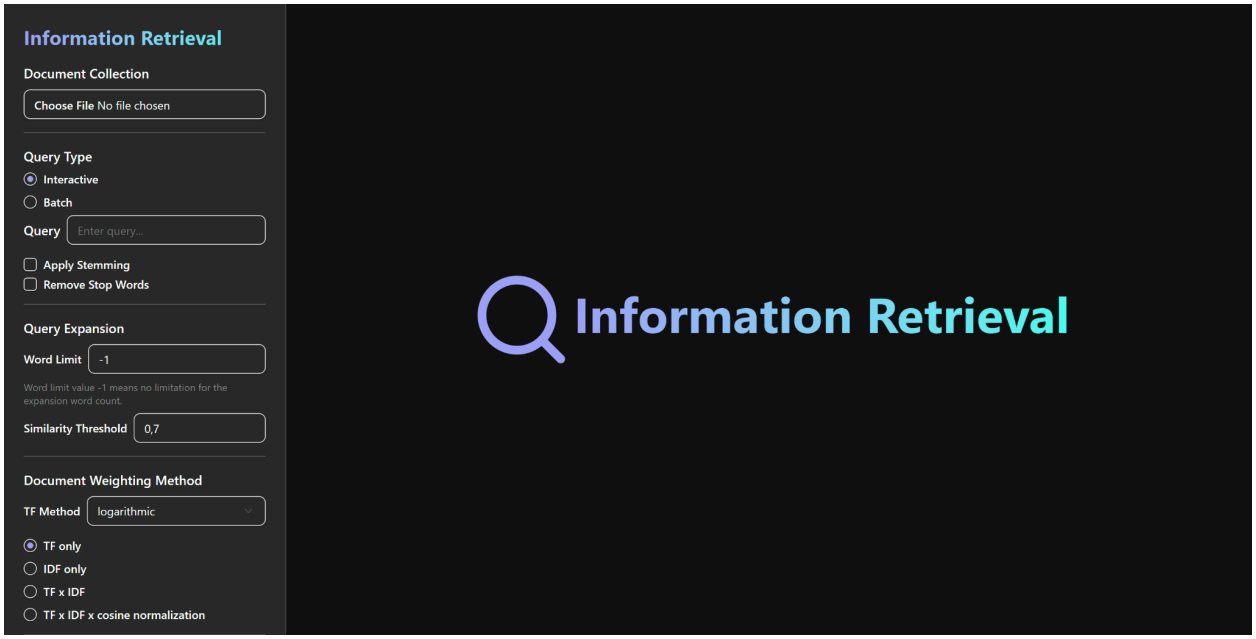
Hasil *retrieval* akan ditampilkan pada halaman yang sama. Beberapa komponen hasil yang ditampilkan adalah sebagai berikut (referensi lihat Gambar C.3).

1. *Query*
    a. *Query* asli (lihat Gambar C.3 bagian (a))
    b. *Expanded query* di bawah tulisan *query* asli (lihat Gambar C.3 bagian (b))
    c. Tombol untuk menampilkan bobot *query* asli dan *expanded query* (lihat Gambar C.3 bagian (c)). Bobot *query* akan ditampilkan pada *pop-up* seperti pada Gambar C.9.
2. Hasil *retrieval* yang ditampilkan ada dua jenis, yaitu hasil *retrieval* dari *query* asli dan hasil *retrieval* dari *expanded query*. Keduanya ditampilkan dalam bentuk *tab*.
    a. Opsi untuk memilih hasil *retrieval* yang ingin ditampilkan: *query* asli atau *expanded query* (lihat Gambar C.3 bagian (d))
    b. Skor MAP, khusus untuk *batch query* (lihat Gambar C.3 bagian (e))
    c. Navigasi untuk melihat hasil *retrieval* untuk setiap *query* untuk *batch query* (lihat Gambar C.3 bagian (f))
    d. Tombol untuk menampilkan *inverted file* (lihat Gambar C.3 bagian (g)). *Inverted file* seluruh dokumen ditampilkan pada *pop-up* seperti pada Gambar C.7. Untuk

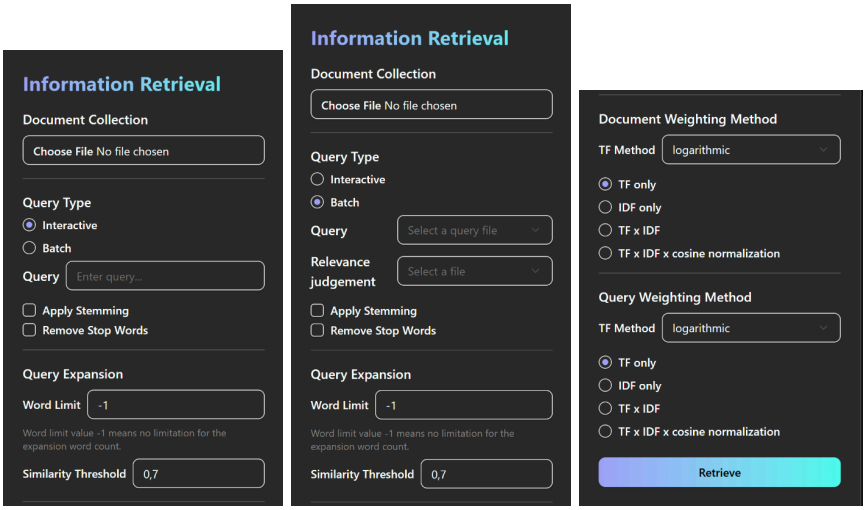
menampilkan bobot kata pada dokumen tertentu, pilih dokumen yang diinginkan pada *dropdown*, seperti pada Gambar C.8.

e. Daftar dokumen pada hasil *retrieval* (lihat Gambar C.3 bagian (h)), terdiri dari banyaknya dokumen yang diperoleh beserta skor AP (khusus *batch query*). Setiap dokumen terdiri dari ID, *rank*, *similarity*, judul, penulis, konten, serta bibliografi dokumen.

Lampiran C: Antarmuka Aplikasi

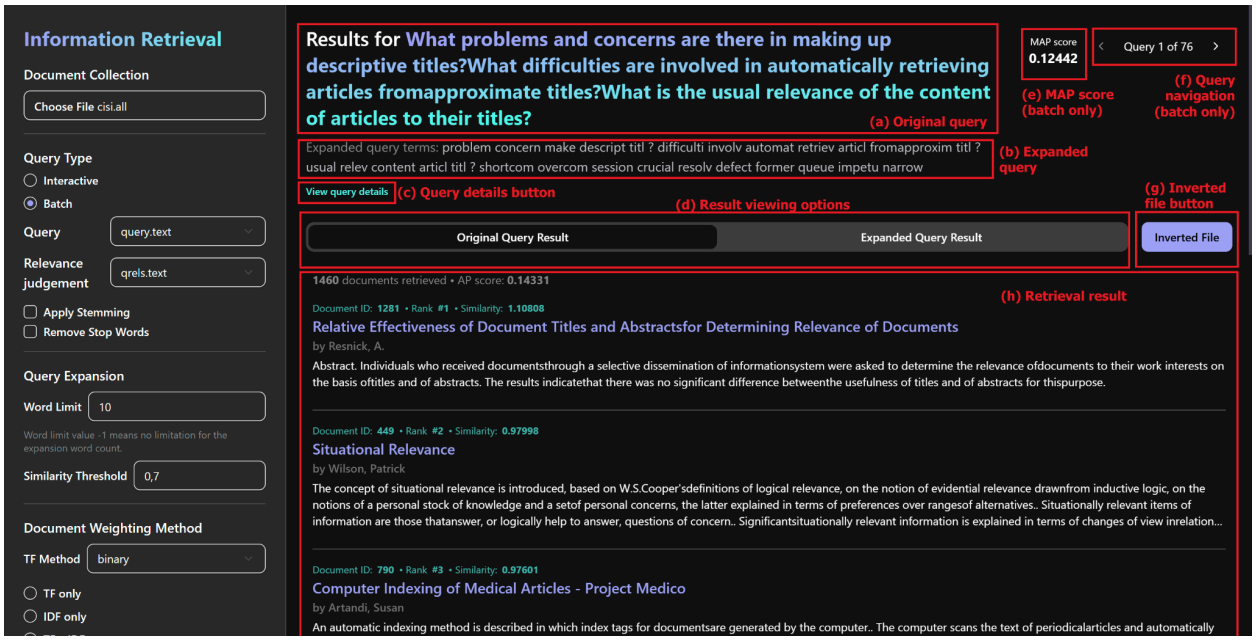


Gambar C.1 Antarmuka web dasar

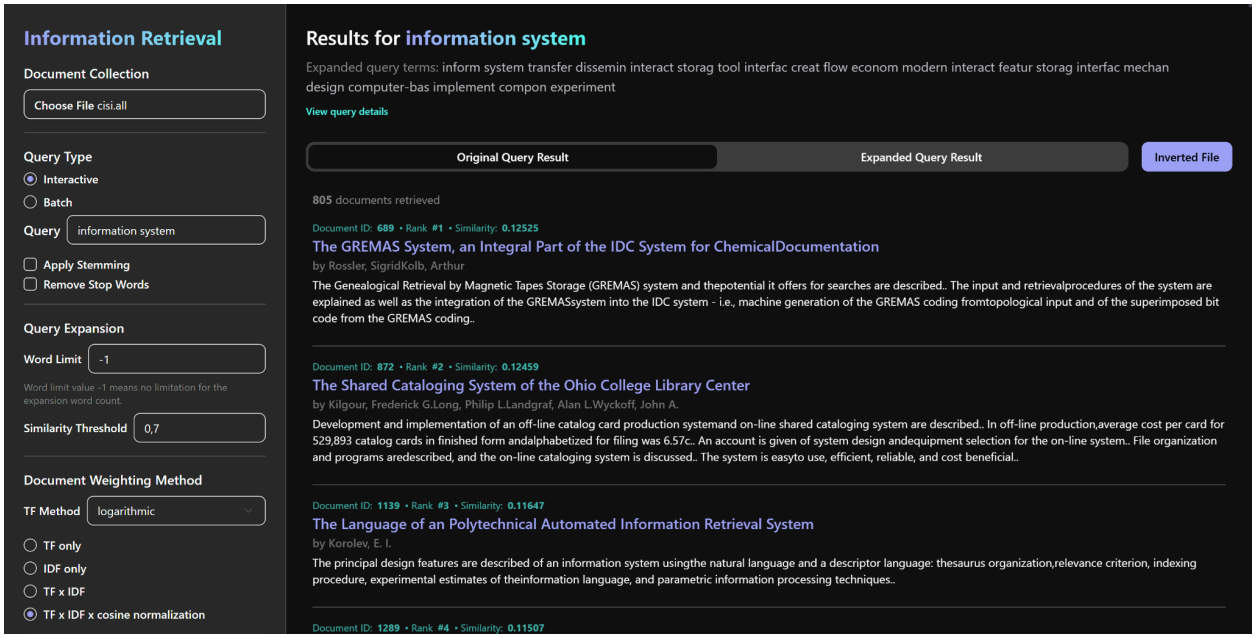


(a) (b) (c)

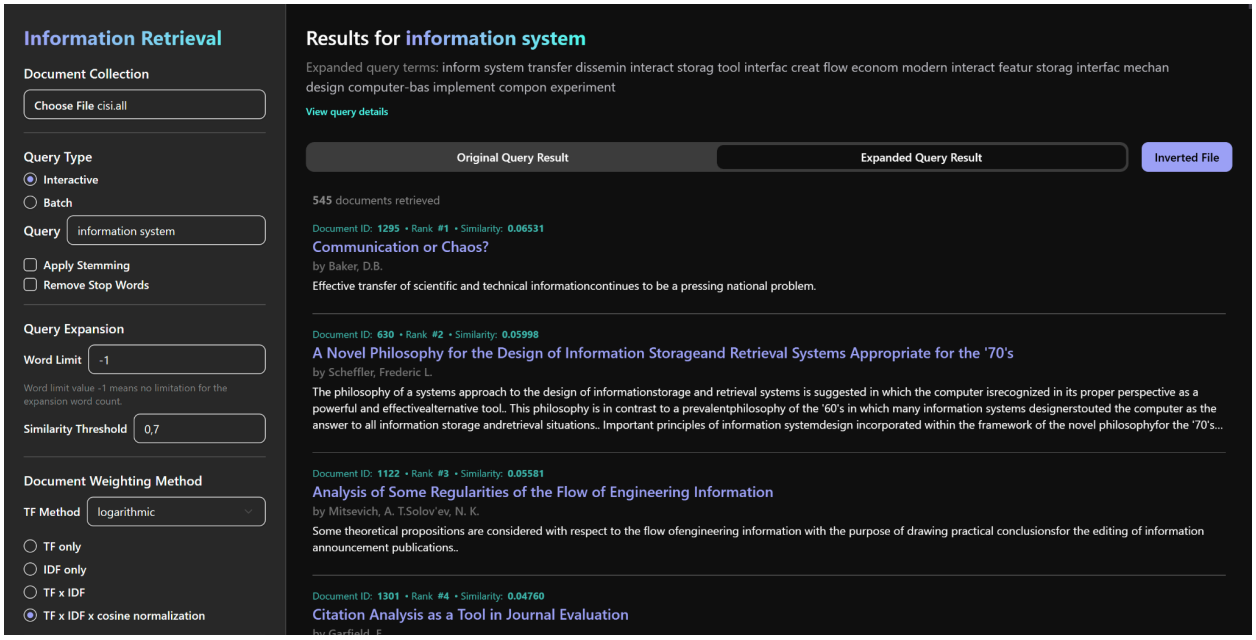
Gambar C.2 *Toolbar*: (a) tampilan *toolbar* untuk *query* interaktif / teks; (b) tampilan *toolbar* untuk *batch query*; (c) tampilan *toolbar* untuk metode pembobotan kata



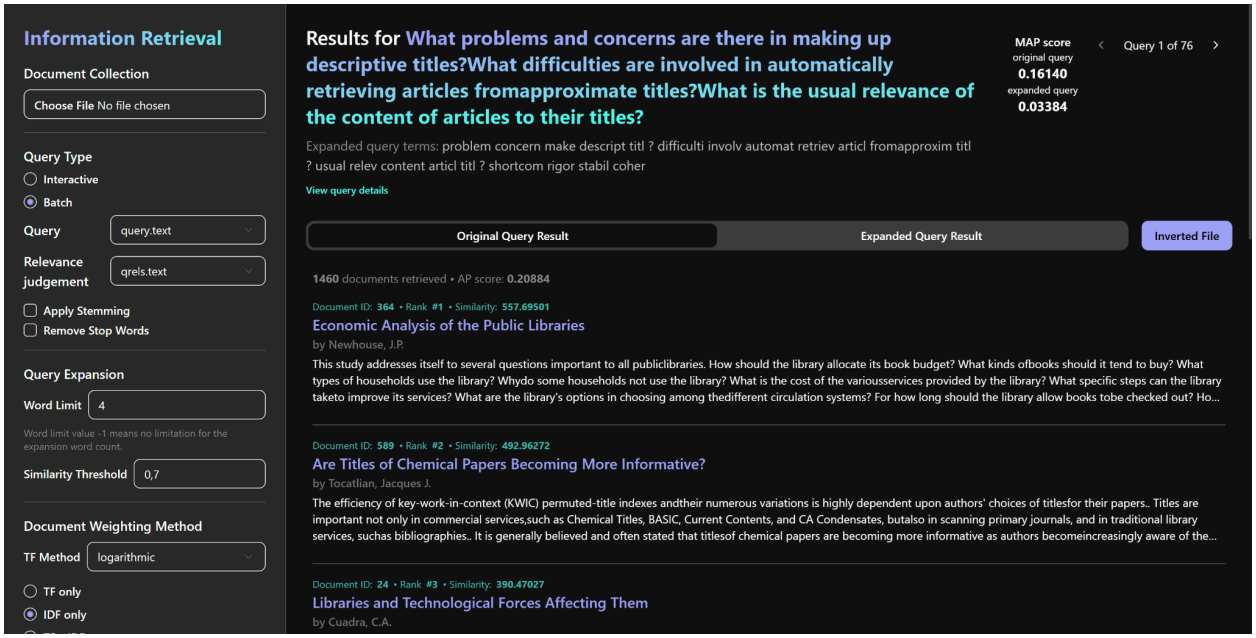
Gambar C.3 Komponen antarmuka hasil *retrieval* (*batch query*)



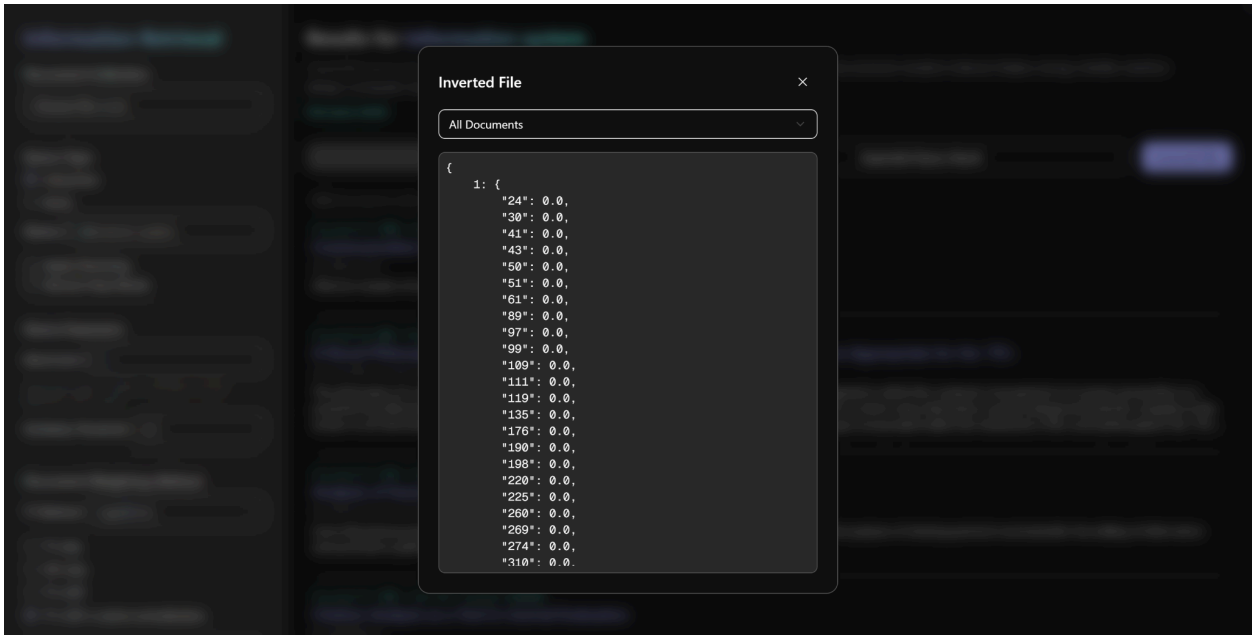
Gambar C.4 Antarmuka hasil *retrieval* untuk *interactive query* (*query* asli)



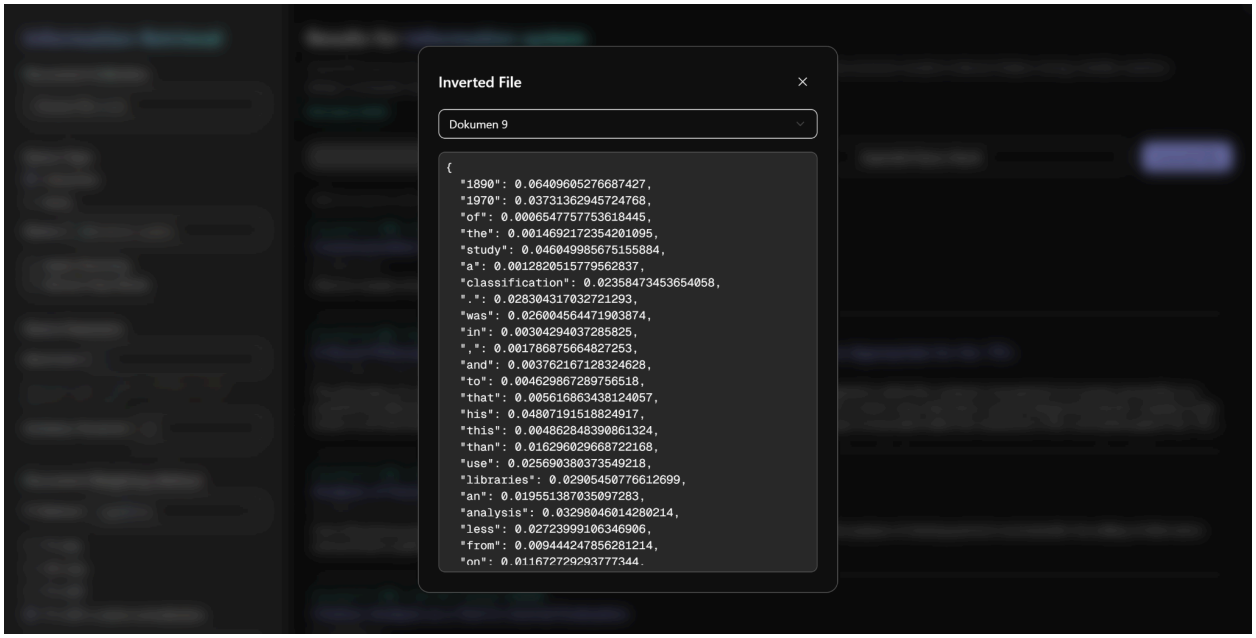
Gambar C.5 Antarmuka hasil *retrieval* untuk *interactive query* (*expanded query*)



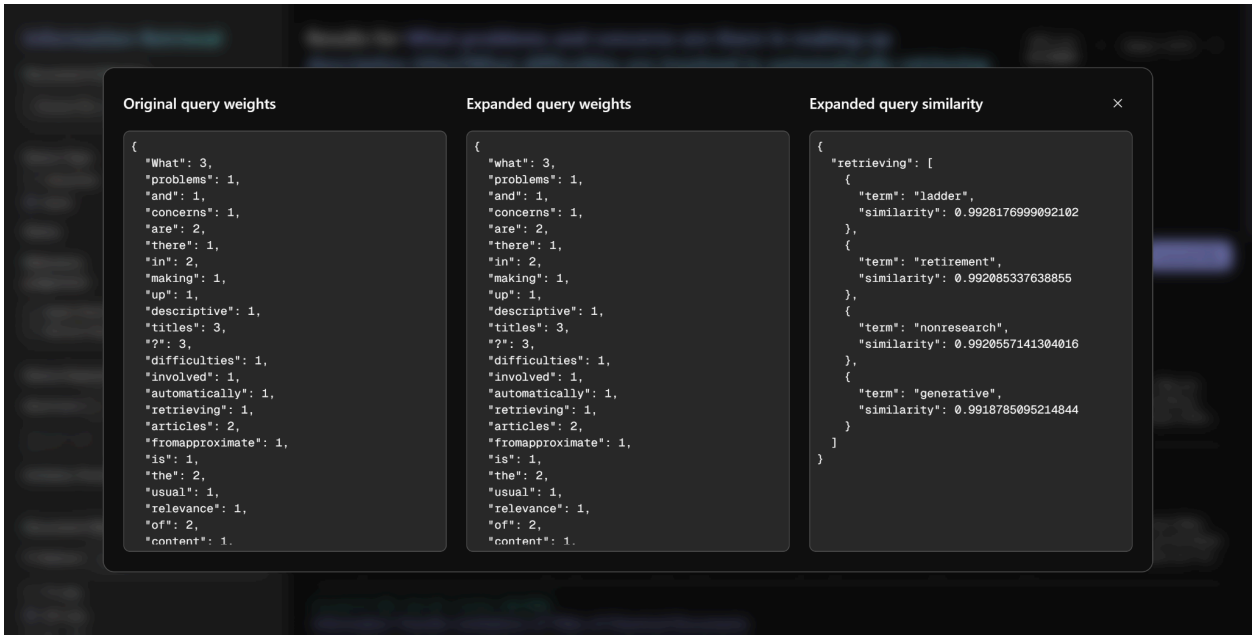
Gambar C.6 Antarmuka hasil *retrieval* untuk *batch query*



Gambar C.7 Antarmuka tampilan *inverted files* seluruh dokumen

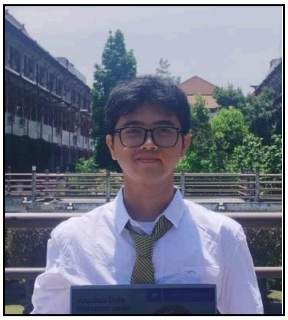
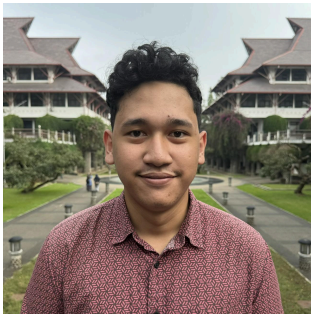


Gambar C.8 Antarmuka tampilan pembobotan kata pada dokumen tertentu



Gambar C.9 Antarmuka tampilan detail bobot *query* asli dan *expanded query*

Lampiran D: Pembagian Tugas

No	Nama	NIM	Foto	Peran
1	Kevin John Wesley Hutabarat	13521042		*Back-end*: ● Fungsi *parsing*: `parser_docs` ● Fungsi *retrieval service*: `calculate_tf_idf`, `calculate_query_weight`, `retrieve_document_single_query`, `retrieve_document_by_id`, `retrieve_document_by_ids` ● Fungsi *preprocessing:* `stem_word, stem_tokens`
2	Arleen Chrysantha Gunardi	13521059		*Front-end*: ● Desain antarmuka dan implementasinya ● Integrasi dengan API dari *back-end* ● Implementasi *flow* aplikasi untuk semua kasus dan *end-to-end* *Back-end*: ● Fungsi `expand_query`
3	Moh. Aghna Maysan Abyan	13521076		*Back-end*: ● Fungsi *parsing*: `parser_qrels`, `parser_query, parser_docs` ● *File* JSON: `parsing_qrels.json`, `parsing_query.json`, `parsing_docs.json`
4	Austin Gabriel Pardosi	13521084		*Back-end*: ● Fungsi *query expansion & query expansion batch* ● API Wrapping untuk semua service dan fungsi ● Fungsi train & retraining word2vec ● Fungsi get_document_details ● Initialize & Dockerize

				Backend
5	Ryan Samuel Chandra	13521140		*Back-end*: ● Fungsi *retrieval service*: `create_inverted_file`, `calculate_similarity`, `retrieve_document` (*single*, *batch*), `get_weight_by_document_id`; ● Fungsi *preprocessing*: tokenisasi, *remove stop word*, integrasi; ● Fungsi evaluasi: precision@K, AP, MAP

Daftar Referensi

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013, Oktober 16). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, *1*(1), 26. https://arxiv.org/abs/1310.4546

Vechtomova, O., & Wang, Y. (2006, July). A Study of the Effect of Term Proximity on Query Expansion. *Journal of Information Science*, *32*(4), 324-333. http://dx.doi.org/10.1177/0165551506065787