TBI - Tugas Pemrograman 4

Anggota:

- Alvaro Austin (2106752180)
- Lyzander Marciano Andrylie (2106750755)
- James Smith Wigglesworth (2106750225)

Links:

Frontend: Covearch
Backend: kgbunshin.org
Video: https://youtu.be/0xWsXag8xig

Deskripsi Proyek:

Pada Tugas Pemrograman 4 ini, kami mengembangkan sebuah aplikasi web *search engine* untuk pencarian informasi seputar penelitian terkait penyakit Covid-19 serta dampaknya. Aplikasi ini kami beri nama Covearch. Dalam pengerjaan ini, kami melakukan *rank retrieval* dengan menggunakan BM25 sebagai metode baseline dalam mengambil dokumen relevan, kemudian akan dilanjutkan dengan eksplorasi proses *learning-to-rank*, lebih spesifik lagi yaitu *re-ranking* dengan basis Neural Network, seperti Bi-Encoder, Cross-Encoder, serta MonoT5. Hasil dari *re-ranking* ini akan ditampilkan ke dalam UI dari aplikasi kami.

Repositori Kode:

Front-End: https://github.com/TBI-Pacil/FE-SearchEngine
Back-End: https://github.com/TBI-Pacil/BE-SearchEngine
Machine Learning: https://github.com/TBI-Pacil/ML-SearchEngine.git

Deskripsi Dataset

Dataset yang kami gunakan adalah dataset mteb/trec-covid yang kami ambil dari HuggingFace, yaitu https://huggingface.co/datasets/mteb/trec-covid. Dataset ini berisi corpus, queries serta qrels terkait beberapa *paper* dan artikel ilmiah terkait Covid-19. Dataset ini sudah memenuhi kriteria yang diberikan oleh soal. Dataset ini memiliki row sebanyak 171 ribu untuk corpus, 66.3 ribu untuk qrels, serta 50 untuk query. Dari subset corpus, kami menggunakan kolom ***_id*** sebagai corpus-id, ***title***, beserta ***text*** sebagai data dokumen. Kemudian, dari subset query, kami menggunakan kolom ***_id*** sebagai query-id beserta ***text*** sebagai data query. Terakhir, dari subset default, kami menggunakan kolom ***query-id*** dan ***corpus-id*** yang bersesuaian dengan id dari dokumen dan query, serta ***score*** yang menunjukkan relevansi dokumen relatif terhadap query.

Evaluasi Sistem:

	name	map	P@10	nDCG@10	map +	map -	map p-value	P@10 +	P@10 -	P@10 p-value	nDCG@10 +	nDCG@10 -	nDCG@10 p-value
0	BM25	0.134367	0.434	0.435145	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	BM25 >> Bi-encoder	0.021139	0.490	0.480231	1.0	49.0	2.926370e-09	21.0	12.0	6.542033e-02	22.0	22.0	1.956452e-01
2	BM25 >> Cross-encoder	0.024452	0.590	0.616905	1.0	49.0	4.159683e-09	34.0	5.0	1.650589e-07	39.0	5.0	1.093098e-07
3	Bi-encoder >> Cross-encoder	0.025877	0.668	0.693668	5.0	45.0	1.972121e-08	36.0	8.0	1.680865e-06	38.0	11.0	4.539252e-07
4	BM25 >> MonoT5	0.024869	0.620	0.647212	2.0	48.0	4.850270e-09	35.0	3.0	2.193784e-07	39.0	6.0	2.344388e-07
5	Bi-encoder >> MonoT5	0.026146	0.664	0.705318	5.0	45.0	2.257208e-08	38.0	8.0	2.108664e-06	40.0	9.0	3.241663e-07
6	Cross-encoder >> MonoT5	0.044196	0.812	0.829669	14.0	36.0	1.602329e-06	43.0	5.0	9.466084e-10	45.0	4.0	3.159294e-10
7	Bi-encoder >> Cross-encoder >> MonoT5	0.026146	0.664	0.705318	5.0	45.0	2.257208e-08	38.0	8.0	2.108664e-06	40.0	9.0	3.241663e-07

Berdasarkan hasil evaluasi yang ditampilkan, sistem retrieval diuji menggunakan berbagai pipeline dengan kombinasi algoritma berbasis BM25, Bi-encoder, Cross-encoder, dan MonoT5. Evaluasi dilakukan berdasarkan metrik-metrik utama seperti Mean Average Precision (MAP), Precision@10 (P@10), dan Normalized Discounted Cumulative Gain@10 (nDCG@10). Setiap pipeline menunjukkan perbedaan performa yang signifikan dalam meningkatkan hasil retrieval dibandingkan baseline BM25.

Pipeline "**BM25 >> Bi-encoder**" dan "**BM25 >> Cross-encoder**" menunjukkan peningkatan performa yang cukup signifikan pada metrik P@10 dan nDCG@10 dibandingkan dengan BM25 murni. Pipeline ini memanfaatkan Bi-encoder dan Cross-encoder sebagai reranker untuk meningkatkan relevansi hasil retrieval setelah BM25, meskipun performa MAP tetap sedikit di bawah MonoT5. Pipeline berbasis encoder ini efektif meningkatkan ketepatan hasil pada rank atas.

MonoT5, sebagai *generative model*, memberikan performa yang superior pada hampir semua metrik evaluasi. Pipeline "**Cross-encoder >> MonoT5**" dan "**BM25 >> MonoT5**" berhasil mencapai nDCG@10 tertinggi, menunjukkan bahwa MonoT5 sangat efektif untuk tugas reranking. Hal ini menunjukkan bahwa penggunaan MonoT5 sebagai tahap akhir dalam retrieval pipeline memberikan dampak yang signifikan pada relevansi hasil retrieval.

Dari hasil evaluasi, dapat disimpulkan bahwa kombinasi metode retrieval tradisional seperti BM25 dengan teknik pembelajaran mendalam seperti Bi-encoder, Cross-encoder, dan MonoT5 menghasilkan peningkatan yang signifikan dalam kualitas hasil retrieval. Penggunaan pipeline "**Cross-encoder >> MonoT5**" menjadi **performa terbaik** yang bisa digunakan, terutama dalam skenario yang memerlukan tingkat relevansi tinggi. Akan tetapi, **pada *production*** kami memutuskan untuk menggunakan "**Bi-encoder >> MonoT5**". Hal ini dikarenakan proses reranking yang dilakukan oleh pipeline dengan varian Cross-encoder memerlukan waktu yang cukup lama, bisa mencapai 8 menit. Hal ini tentu tidak *feasible* untuk diterapkan pada *search engine*. Oleh karena itu, kami memutuskan menggunakan varian Bi-Encoder yang hanya memerlukan waktu 4 detik untuk keseluruhan proses pipeline.

Arsitektur Aplikasi:

Metode Deployment:

1. Frontend

Frontend di-deploy dengan menggunakan cloud hosting gratis yaitu Netlify (https://www.netlify.com/). Dalam Netlify, kami diberikan banyak konfigurasi yang dapat kami kustomisasi seperti .env, dsb.

2. Backend

Backend di-deploy menggunakan Google Cloud Run, yang memanfaatkan container untuk menjalankan aplikasi secara scalable dan serverless. Proses ini menggunakan Docker Image yang dibuat sesuai kebutuhan backend. Docker Image dibuat dengan menggunakan Dockerfile configuration dari versi ringan Python (`python:<version>-slim`) untuk mengurangi ukuran image dan mempercepat waktu build.Image ini dirancang untuk menjalankan aplikasi Python menggunakan framework FastAPI.

3. Model (Pipeline Configuration)

Model ini dikembangkan menggunakan Google Colab Notebook dengan ekstensi `.ipynb`. Setelah pipeline selesai, model yang dihasilkan diserialisasi menggunakan package pickle dan diunggah ke Google Cloud Storage (GCS) untuk penyimpanan.

Agar backend dapat mengunduh model tersebut, disediakan skrip yang telah dibuat pada file `app/utils/index.py`. Skrip ini dirancang untuk mengunduh model dari bucket GCS dengan efisiensi tinggi.

Langkah-Langkah Sebelum Penggunaan:

1. **Dapatkan Kredensial Google Service Account**
   Pengguna perlu mendapatkan file kredensial (berformat JSON) dari Google Cloud melalui Google Cloud IAM (*Identity and Access Management* - IAM).
2. **Mengunggah Model ke GCS**
   Model hasil pipeline diunggah ke bucket GCS dengan konfigurasi akses yang sesuai untuk backend.
3. **Integrasi Backend**
   Letakkan file kredensial di lokasi yang dapat diakses oleh aplikasi backend,

kemudian pastikan backend menggunakan skrip pada `app/utils/index.py` untuk mengunduh model secara otomatis dari GCS.