TUGAS PEMROGRAMAN 4

Anggota:

- Airel Camilo Khairan (2106652581)
- Eugenius Mario Situmorang (2106750484)

Links:

- Dataset: https://drive.google.com/drive/folders/1xwLmQdRet3NrIv0KYStrCVn57K-2YVlZ
- GitHub: https://github.com/airelcamilo/IR-TP4

Kami menggunakan data collections dari Tugas Pemrograman 3 untuk mengerjakan tugas ini dan kami memilih untuk mengerjakan pilihan 1.

1. LambdaMART dengan Berbagai Fitur

Pada bagian ini, kami melakukan eksperimen untuk menentukan fitur yang memberikan akurasi tinggi dalam *re-ranking* Top-100 dokumen dengan pembobotan skor TF-IDF untuk eksperimen nomor 1.1., 1.2. atau BM25 untuk lainnya. Sebelum diubah menjadi fitur, teks query dan dokumen melalui tahap *preprocessing* terlebih dahulu, yaitu tokenisasi, stemming, dan penghapusan stop words. Setelah itu, query dan dokumen diubah menjadi berbagai bentuk fitur dalam sebuah vektor untuk merepresentasikan similaritas antara keduanya. Setelah diubah menjadi vektor, vektor tersebut digunakan untuk melatih model LambdaMART dan melakukan reranking Top-100 dokumen. Kami kemudian membandingkan hasilnya untuk menemukan fitur dengan akurasi yang tinggi.

Berikut adalah beberapa kombinasi fitur yang diuji:

1.1. *Baseline* (LSI query + LSI dokumen + Cosine Distance + Jaccard Similarity)
Fitur-fitur pada vektor sama saja dengan Tugas Pemrograman 3, yaitu menggunakan Latent Semantic Indexing (LSI) untuk query dan dokumen, cosine distance dari kedua LSI, serta jaccard similarity query dan dokumen.

1.2. TF-IDF + Cosine Distance LSI + Jaccard Similarity
Karena hasil evaluasi *baseline* yang kurang baik, maka LSI query dan dokumen digantikan dengan skor TF-IDF yang lebih merepresentasikan similaritas antara query dan dokumen. Fitur lainnya tetap sama dengan *baseline*.

1.3. BM25 + Cosine Distance LSI + Jaccard Similarity
Menggantikan LSI query dan dokumen dengan representasi similaritas antara query dan dokumen yang lain, yaitu skor BM25. Fitur lainnya tetap sama dengan *baseline.*

1.4. BM25 + CBOW
Dapat dilihat pada tabel bahwa menggunakan fitur skor BM25 menghasilkan hasil evaluasi yang lebih bagus daripada menggunakan fitur skor TF-IDF. Kemudian untuk meningkatkan akurasi, kami mencoba menggunakan Word2Vec. Karena ada 2 jenis arsitektur Word2Vec, pertama kami mencoba arsitektur CBOW untuk

mendapatkan fitur vektor *word embedding* query, *word embedding* dokumen, dan cosine distance kedua vektor.

1.5. BM25 + Skip-Gram
Kemudian, mencoba arsitektur Skip-Gram untuk mendapatkan fitur vektor *word embedding* query, *word embedding* dokumen, dan cosine distance kedua vektor.

1.6. BM25 + FastText CBOW
Menggunakan arsitektur FastText CBOW untuk mendapatkan fitur vektor *word embedding* query, *word embedding* dokumen, dan cosine distance kedua vektor.

1.7. BM25 + FastText Skip-Gram
Menggunakan arsitektur FastText Skip-Gram untuk mendapatkan fitur vektor *word embedding* query, *word embedding* dokumen, dan cosine distance kedua vektor.

Skor	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**1.7**
RBP	0.74	0.83	0.87	0.88	0.89	0.89	0.90
DCG	6.57	7.21	7.40	7.43	7.48	7.47	7.51
AP	0.70	0.78	0.84	0.85	0.87	0.87	0.87

Tabel 1. Hasil Evaluasi Berbagai Fitur

Analisis:

- Dari hasil evaluasi tersebut terlihat bahwa penggunaan metode skoring TF-IDF atau BM25 sebagai fitur membuat *re-ranking* lebih akurat daripada *baseline* karena dapat merepresentasikan similaritas antara query dan dokumen dengan lebih baik.
- Penggunaan BM25 sebagai fitur memberikan hasil evaluasi yang lebih baik daripada TF-IDF. Hal ini disebabkan BM25 memperhitungkan *eliteness* suatu term dan melakukan normalisasi panjang dokumen.
- Penggunaan arsitektur Skip-Gram dalam Word2Vec maupun FastText memberikan akurasi yang lebih tinggi daripada arsitektur CBOW. Hal ini disebabkan karena Skip-Gram bekerja dengan baik dengan data training kecil dan dapat lebih akurat merepresentasikan kata-kata yang jarang muncul.
- Penggunaan FastText memberikan hasil akurasi yang lebih tinggi daripada Word2Vec. Karena FastText bekerja pada level subkata atau n-gram sementara Word2Vec hanya pada level kata saja. Hal itu membuat FastText dapat menghasilkan representasi vektor *word embedding* untuk kata yang tidak ada di vocab serta dapat lebih akurat merepresentasikan kata-kata yang jarang muncul.
- Dengan demikian dari hasil eksperimen tersebut, model LambdaMART dengan fitur BM25 + FastText Skip-Gram memberikan akurasi yang terbaik dalam *re-ranking* dokumen. Kombinasi ini menggabungkan keunggulan BM25 sebagai skor fitur dan FastText Skip-Gram dalam menghasilkan representasi vektor kata yang akurat.

2. Implementasi Dense Passage Retrieval

Dalam eksperimen ini, kami menjelajahi penggunaan berbagai fitur untuk meningkatkan akurasi dalam melakukan ranking terhadap Top-100 dokumen menggunakan metode Dense Passage Retrieval (DPR). Sebelum fitur diterapkan, query dan dokumen melewati tahap *preprocessing*, termasuk tokenisasi, *stemming*, dan penghapusan *stop words*. Setelah itu, *query* dan dokumen diubah menjadi vektor fitur untuk merepresentasikan similaritas antara keduanya dalam ruang vektor.

Setelah tahap *preprocessing*, kami menggunakan metode *Dense Passage Retrieval* (DPR) untuk mendapatkan vektor representasi dari query dan dokumen. Beberapa fitur utama yang diambil dari hasil pencarian DPR, yaitu:

- *Faiss Distance* (faiss_dist): Jarak antara query dan dokumen yang diukur oleh Faiss, memberikan indikasi sejauh mana dokumen tersebut relevan dengan pertanyaan.
- *Reader Relevance* (reader_relevance): Skor relevansi yang diberikan oleh pembaca DPR, menunjukkan sejauh mana dokumen dianggap relevan berdasarkan konten pembaca.

Dari kedua nilai yang didapati dalam hasil pencarian dengan metode DPR, kami memilih *Reader Relevance* dengan Min-Max Normalization sebagai patokan perhitungan indikator RBP, DCG, dan AP. Dari eksperimen kami dapati hasil sebagai berikut:

Indikator	Skor
RBP	0.93
DCG	7.73
AP	0.89

Tabel 2. Hasil Evaluasi Implementasi DPR pada Skor *Reader Relevance*

Analisis:

- Dari hasil evaluasi tersebut terlihat bahwa penggunaan metode skor *Reader Relevance* sebagai fitur membuat *ranking* lebih akurat daripada *baseline* karena dapat merepresentasikan similaritas antara query dan dokumen dengan lebih baik.
- RBP mencapai 0.93. RBP memberikan perhatian lebih pada hasil peringkat awal, dan nilai yang tinggi menunjukkan bahwa dokumen relevan mendominasi peringkat atas. DCG mencapai 7.73. DCG mengukur keberlanjutan dan relevansi hasil peringkat, dan nilai yang tinggi menunjukkan hasil peringkat yang baik. AP mencapai 0.89. AP mengukur rata-rata presisi pada semua posisi relevan dalam hasil pencarian.
- Penggunaan skor *Reader Relevance* sebagai fitur memberikan *ranking* yang lebih akurat dibandingkan dengan *baseline*. Dengan nilai RBP, DCG, dan AP yang tinggi, DPR menunjukkan kemampuan yang baik dalam menilai relevansi dokumen

terhadap *query*. Hasil eksperimen menunjukkan bahwa model DPR dengan fitur *Reader Relevance* memberikan akurasi *ranking* yang optimal. Pemilihan fitur yang merepresentasikan relevansi pembaca (*Reader Relevance*) memberikan kontribusi signifikan terhadap peningkatan kualitas *ranking*.

Eksperimen Lainnya:

- Pada awal implementasi, kami mencoba memanfaatkan ElasticSearch untuk mempercepat proses *search* pada program kami yang dipadukan dengan DPR, tetapi terdapat kendala menjalankan ElasticSearch pada mesin localhost yang dihubungkan ke Google Collab, sehingga kami memutuskan untuk memanfaatkan json format sebagai penyimpanan *index* dan proses searchingnya dilakukan dalam *file* format tersebut.
- Terdapat pengecilan ukuran qrels-folder menjadi qrels-folder-for-dpr untuk mengatasi proses lamanya membangun model pada DPR, berikut merupakan pebandingan ukuran folder evaluasi dari yang sebelumnya (metode klasik) ke metode DPR:

File	Klasik	DPR
train_queries.txt	1050 queries	100 queries
train_qrels.txt	1050 x 100 qrels	100 x 5 qrels
train_docs.txt	36156 documents	499 documents
test_queries.txt	150 queries	50 queries
test_qrels.txt	150 x 100 qrels	50 x 5 qrels
val_queries.txt	1350 queries	100 queries
val_qrels.txt	1350 x 100 qrels	100 x 5 qrels

Tabel 3. Perbedaan ukuran *folder* pada metode klasik dan DPR

- Kami menambahkan env pada GitHub dengan menginstall melalui requirements.txt. env dapat digunakan untuk menjalankan program pada mesin lokal. Rekomendasi ke depannya adalah mencoba melakukan kombinasi *indexing* dengan ElasticSearch memanfaatkan Google Cloud Platform sebagai *virtual machine*.

Referensi:

Alfan F. Wicaksono. Slide Perkuliahan Perolehan Informasi 2023.

Cesconi, F. *What is the main difference between word2vec and FastText?*. Medium. https://cesconi.com/what-is-the-main-difference-between-word2vec-and-fasttext-57bdaf3a69ef

Riva, W. *Word embeddings: CBOW vs skip-gram*. Baeldung on Computer Science. https://www.baeldung.com/cs/word-embeddings-cbow-vs-skip-gram

Karpukhin, Oguz, Min, Lewis, Wu, Edunov, Chen, Yih. Dense Passage Retrieval for Open-Domain Question Answering. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, 2020.

Dense Passage Retriever. Facebook Research. GitHub. https://github.com/facebookresearch/DPR

Max Davish. (2020). Dense Passage Retrieval. GitHub. https://github.com/mdavish/dense-passage-retrieval