CSCE604135 • Perolehan Informasi
Semester Ganjil 2022/2023
Fakultas Ilmu Komputer, Universitas Indonesia

Tugas Pemrograman 2
Ranked Retrieval dengan TF-IDF & BM25

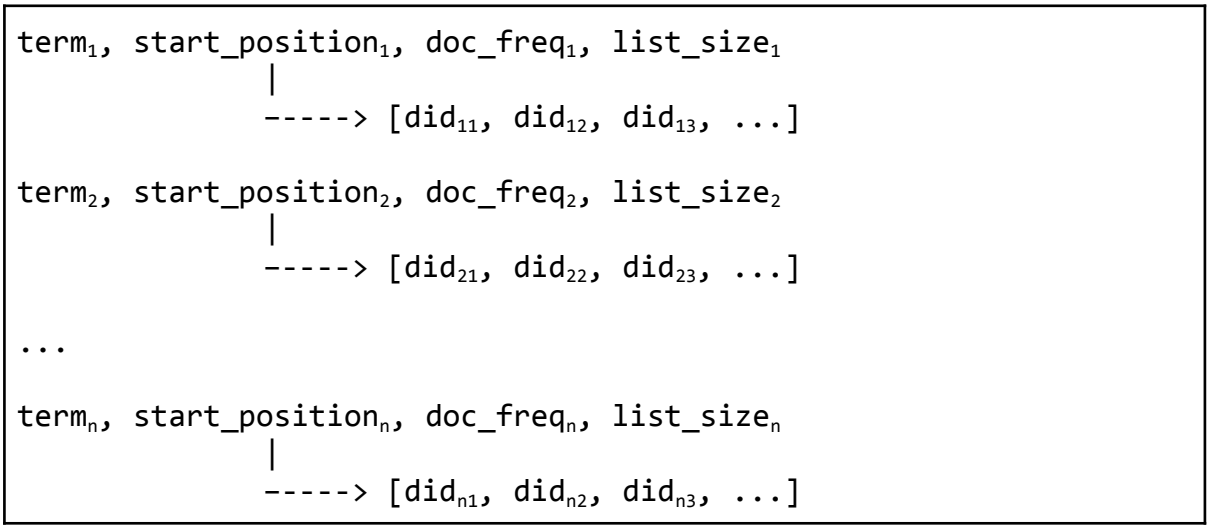
Deadline: 14 Oktober 2022, 23:55 WIB

Ketentuan:

1. Tugas Pemrograman 2 ini terdiri dari 1 buah file .zip berisi *template* program dan *dataset* dokumen forum kesehatan dalam Bahasa Inggris.
2. Lengkapi program template yang diberikan sesuai dengan petunjuk pengerjaan tugas yang disediakan.
3. Seluruh program (file .py) yang telah dibuat dikumpulkan dalam satu *folder* dan dikonversi ke dalam format .zip dengan format penamaan **TugasX_NPM.zip**
   Contoh: Tugas2_1906262623.zip
4. Kumpulkan tugas pada submisi yang telah disediakan di SCeLe sebelum tanggal **14 Oktober 2022, 23.55 WIB**. Keterlambatan pengumpulan akan dikenakan penalti sebesar 30% untuk 3 hari setelah deadline. Setelahnya submisi tidak akan diterima.
5. Tugas ini dirancang sebagai **tugas mandiri**. Plagiarisme tidak diperkenankan dalam bentuk apapun. Adapun kolaborasi berupa diskusi (tanpa menyalin maupun mengambil jawaban orang lain) dan literasi masih diperbolehkan dengan mencantumkan kolaborator dan sumber.
6. **Anda boleh konsultasi dengan asisten dosen (LINK). Asisten dosen diperbolehkan membantu Anda dengan memberikan petunjuk.**

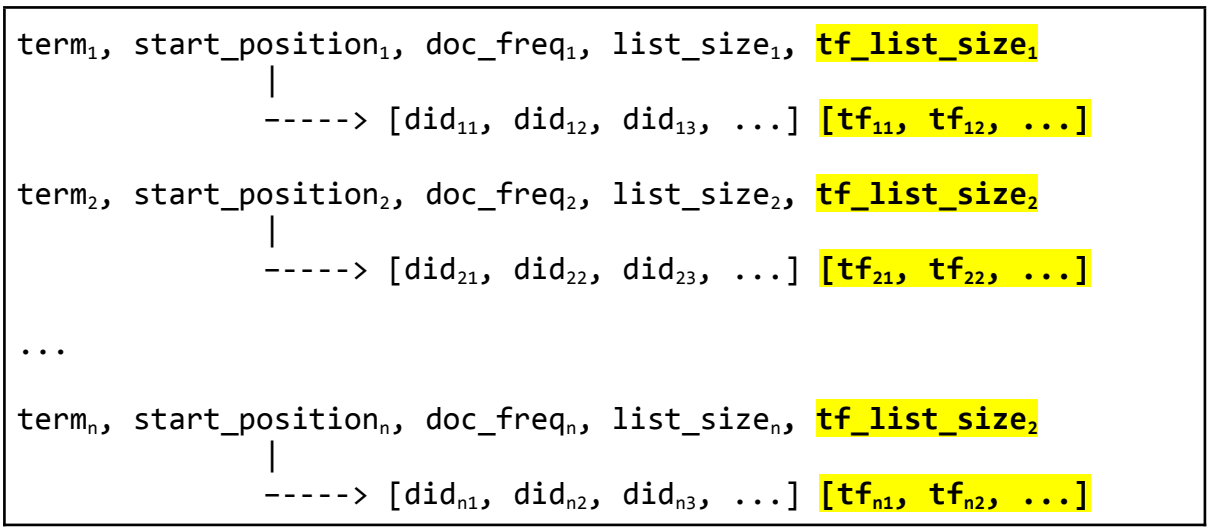
Petunjuk Pengerjaan Tugas

Sebelumnya di Tugas Pemrograman 1, Anda diminta untuk mengimplementasikan *boolean retrieval system*. Anda belajar melakukan *indexing* dan membangun *inverted index*, yang terdiri dari *dictionary* dan juga *postings lists*. Berikut adalah visualisasi *inverted index* pada Tugas Pemrograman 1:



`start_position` (dalam *bytes*) adalah posisi pada *file index* dimana *compressed/encoded postings list* tersimpan; **`doc_freq`** adalah *document frequency* yang berguna untuk perhitungan IDF; dan **`list_size`** (dalam *bytes*) adalah ukuran/panjang *postings list* yang disimpan.

Pada Tugas Pemrograman 2 ini, Anda diminta untuk **menambahkan informasi** pada *inverted index* dan mengimplementasikan *ranked retrieval model* dengan pembobotan menggunakan TF-IDF dan BM25. Berikut adalah visualisasi *inverted index* pada Tugas Pemrograman 2:



Tambahannya adalah **`tf_list_size`** merupakan ukuran/panjang (dalam *bytes*) *list* yang berisi informasi *term frequency*. Informasi posisi tidak diperlukan karena **term frequencies list disimpan persis setelah postings list**. Artinya, posisi TF list pada *file* adalah **`start_position`** + **`list_size`**; Kemudian, $\mathbf{tf_{ij}}$ berarti berapa banyak $\mathbf{term_i}$ muncul pada dokumen $\mathbf{did_{ij}}$. Yang perlu dicatat adalah **TF list jangan diubah ke Gap List** (sebelum dilakukan proses encoding) seperti *postings list*. Ingat bahwa TF list tidak dijamin terurut.

Pada tugas ini, sudah disediakan *template* program yang harus diisi oleh mahasiswa untuk bisa membuat sistem yang bisa berjalan dengan baik dan efisien. Terdapat 6 buah program yang digunakan dalam Tugas Pemrograman 2, dengan rincian sebagai berikut:

- 1 (satu) buah program dari Tugas Pemrograman 1, yaitu **search.py**.
- 4 (empat) buah program dari Tugas Pemrograman 1 yang perlu dikerjakan, yaitu **util.py**, **index.py**, **compression.py**, dan **bsbi.py**.
- 1 (satu) buah program baru yang perlu dilengkapi, yaitu **experiment.py**.

Setiap program telah dilengkapi dokumentasi dengan *template* yang diberikan. Setiap *function* dalam program juga sudah diberikan dokumentasi lengkap untuk memudahkan Anda dalam menyusun program. Bagian yang perlu anda lengkapi sudah ditandai dengan comment **#TODO.** Selain berisi program, beberapa *file* juga telah diisi *sample test case* pada bagian akhir (pada *block* if __name__ == '__main__') untuk memeriksa kebenaran dari program yang anda buat. Anda juga dibebaskan untuk mengubah bagian ini untuk testing lebih lanjut. Namun perlu diingat bahwa penilaian akan tetap dilihat pada kualitas program yang dibuat.

Untuk memudahkan Anda dalam mengerjakan, pengerjaan akan dibagi menjadi 2 (dua) bagian dengan *walkthrough* pengerjaan masing-masing bagian sebagai berikut:

Bagian 1

1. Lakukan implementasi pada *file* **util.py**. *File* ini berisi implementasi *mapping* sederhana untuk menyimpan pemetaan bagi sebuah *term* ke *integer* (*term* ID) dan juga sebuah dokumen ke sebuah *integer* (doc ID); serta sebaliknya. Selain itu, pada *file* ini juga disediakan fungsi untuk melakukan penggabungan dua dokumen yang sama dengan mengakumulasikan *term frequency*-nya.
2. Buat implementasi pada *file* **compression.py**. *File* ini berisi implementasi untuk mengubah representasi *postings* menjadi *sequence of bytes* yang akan disimpan pada memori. Selain *posting list*, disimpan juga *list* yang berisi term *frequency*. Terdapat method *encode* dan *decode* yang akan dipanggil dari *class* lain, sementara method lainnya berperan sebagai *helper method*. Ketika melakukan *encoding*, *list of postings* perlu diubah ke dalam bentuk *list of gaps*. Kemudian, *list of gaps* akan di-*compress* dengan Variable-Bytes Compression. namun *list of term frequency* tidak perlu diubah ke *list of gaps*, bisa langsung di-*compress* dengan Variable-Bytes Compression.
3. Buat implementasi pada *file* **index.py**. *File* ini berisi beberapa *class* yang merupakan abstraksi dari sebuah Inverted Index, termasuk implementasi untuk melakukan operasi

membaca dan menulis *index* yang berada pada sebuah *storage* (dalam bentuk *file* di *hard disk*).

4. Buat implementasi pada *file* **bsbi.py**. *File* ini berisi sebuah *class* yang merupakan abstraksi dari proses indexing dengan metode **Blocked Sort Based Indexing** (BSBI). Dalam melakukan proses *indexing*, Anda perlu mengimplementasikan method *parse_block* untuk mengolah dokumen menjadi bentuk *list of pairs <termID, docID>* dan method *merge* untuk menggabungkan seluruh *inverted indices* yang sudah dibuat sebelumnya. Kedua method tersebut akan dipanggil oleh method utama *index* yang melakukan proses *indexing* secara keseluruhan.
   Setelahnya terdapat method *retrieve_tfidf* dalam proses searching yang melakukan perhitungan *similarity score* antara sebuah *query* dengan dokumen-dokumen dan mengembalikan **Top-K relevant documents**. Method ini melakukan perhitungan dengan **TF-IDF** dengan framework **TaaT (Term-at-a-Time)**. Kerjakan bagian terkait proses *indexing* terlebih dahulu, lalu baru bagian mengenai proses *searching*.

   NB: Langkah 1 dan 2 bisa dilakukan secara paralel karena kedua program tersebut bersifat independen dengan program lainnya.

Jika seluruh program sudah diimplementasikan, Anda bisa mengujinya dengan langkah sebagai berikut:

1. Jalankan file **bsbi.py** untuk membangun index dari dataset yang tersedia. Jika proses indexing berhasil maka akan muncul *file index* dan *posting-dictionary* pada direktori index. Proses ini bisa memakan waktu yang cukup lama.
2. Jalankan *file* **search.py** untuk melakukan searching pada index yang dibuat. Contoh untuk melakukan searching melalui query sudah tersedia pada *file* tersebut.

Bagian 2

Sempurnakan dan tambahkan kode pada *file* **experiment.py**. **Anda bebas berkreasi pada *file* experiment.py ini**. *File* ini berisi implementasi eksperimen sederhana untuk mengevaluasi mana yang lebih baik antara TF-IDF dan BM25. Berikut kira-kira hal yang perlu Anda lakukan:

- Tambahkan method **retrieve_bm25** pada *file* **bsbi.py** untuk melakukan *retrieval* menggunakan skema *scoring* BM25 dan framework **TaaT**. Anda hanya perlu *copy-paste* dari method **retrieve_tfidf** dan melakukan modifikasi pada bagian *weighting*. Informasi panjang setiap dokumen ada di *instance variable* **`self.doc_length`** di kelas **InvertedIndex**. Informasi rata-rata panjang dokumen di seluruh koleksi bisa juga dihitung dari **`self.doc_length`**. Namun jangan dihitung setiap kali call method **retrieve_bm25**! Perhitungan bisa dilakukan sekali saja, misal ketika object **InvertedIndex** dan turunannya hidup di memori untuk pertama kali.
- Implementasikan metrik evaluasi DCG (Discounted Cumulative Gain) dan AP (Average Precision) yang merupakan metrik evaluasi yang kita gunakan untuk menilai

kualitas SERPs yang dikembalikan oleh *ranked retrieval system* yang Anda kembangkan. Implementasi fungsi RBP sudah diberikan sebagai contoh.

- Terdapat *file* **query.txt** yang berisi 30 *query* dan *file* **qrels.txt** yang merupakan **relevance judgement** yang bisa digunakan untuk evaluasi performa program.
- Lakukan eksperimen varian TF-IDF lainnya dan juga variasi nilai **k1** dan **b** pada BM25 (minimal 3 variasi). Buat *file* dokumen penjelasan hasil eksperimen yang Anda kerjakan dengan penamaan *file* **Tugas2_Experiment.pdf**. **Contoh** isi dokumen adalah seperti pada tabel berikut (berikan penjelasan tambahan, misalnya mana yang paling baik dan mana yang paling buruk):

Scoring Regimes	**DCG**	**AP**	**RBP p = 0.8**
TF-IDF	0.123	0.045	0.342
BM25 k1 = p, b = q	0.321	0.432	0.763
BM25 k1 = x, b = y	0.213	0.324	0.451

Bonus

Sebelumnya Anda implementasikan TF-IDF dan BM25 dengan framework **TaaT**. Untuk bonus, Anda perlu implementasikan TF-IDF dan BM25 dengan **WAND Top-K Retrieval** yang lebih efisien (silakan cari referensi dengan *search engine* favorit Anda). Namun, untuk bisa menerapkan WAND Top-K Retrieval, Dictionary pada inverted index perlu ditambah informasi lagi, yaitu **maximum upper bound score** atau **maximum contribution** dari sebuah term.

Poin penilaian**:**

- Bagian 1 60 poin
- Bagian 2 40 poin
- Bonus 10 poin

Referensi & Kredit:

- Soal tugas pemrograman ini merupakan hasil modifikasi dari tugas pemrograman kuliah serupa di Stanford University: https://web.stanford.edu/class/cs276/pa/pa1.zip
- Data pada tugas ini merupakan koleksi Medline dari University of Glasgow: http://ir.dcs.gla.ac.uk/resources/test_collections/

Selamat mengerjakan!