CSCE604135 • Perolehan Informasi
Semester Gasal 2023/2024
Fakultas Ilmu Komputer, Universitas Indonesia

Tugas Pemrograman 2: Ranked Retrieval dengan TF-IDF & BM25
Tenggat Waktu: Rabu, 11 Oktober 2023, 23.55 WIB

Ketentuan:

1. Untuk Tugas Pemrograman 2 ini, Anda diberikan 1 buah *file* .zip berisi ***template* program** dan ***dataset*** dokumen QnA dalam bahasa Indonesia:
    a. Template program: LINK
    b. Dataset: LINK
2. Lengkapi *template* program yang diberikan sesuai dengan petunjuk pengerjaan tugas yang disediakan.
3. Seluruh program (*file* .py), *folder* index, dan laporan (*file* .pdf) yang telah dibuat dikumpulkan dalam satu *folder* dan dikonversi ke dalam format .zip dengan format penamaan **Tugas2_NPM.zip** . Pengumpulan akan lebih detail dijelaskan selanjutnya.
   Contoh penamaan: Tugas2_2006123456.zip
4. Kumpulkan tugas pada submisi yang telah disediakan di SCeLe sebelum tanggal **Rabu, 11 Oktober 2023, 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 kedua asisten dosen berikut. Asisten dosen diperbolehkan membantu Anda dengan memberikan petunjuk.
    a. Febi Imanuela (line: febijakarta, email: febi.jakarta@gmail.com)
    b. Adrianus Saga (line:saga_ekakristi, email: adrianus.saga21@ui.ac.id)

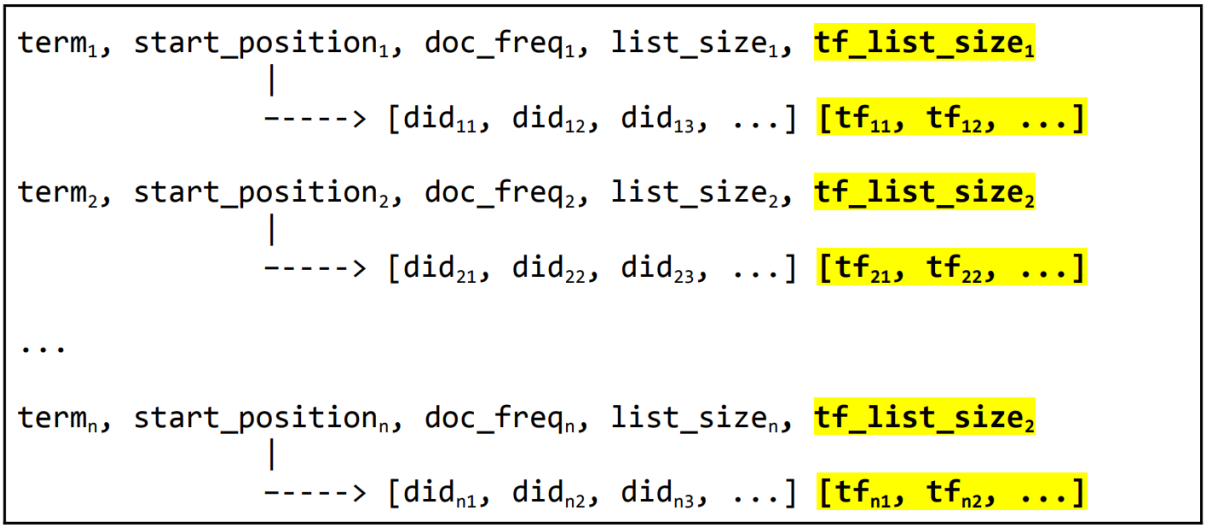
Petunjuk Pengerjaan Tugas

Pada Tugas Pemrograman 2 ini, secara garis besar, Anda diminta untuk melanjutkan Tugas Pemrograman 1 dengan mengimplementasikan *ranked retrieval model* dengan pembobotan menggunakan TF-IDF dan BM25.

Pada Tugas Pemrograman 1, Anda telah menyusun *inverted index* dengan **start_position** (dalam *bytes*), yakni posisi pada *file index* di mana *encoded postings list* tersimpan; **doc_freq**, yakni *document frequency* yang berguna untuk perhitungan IDF; **list_size** (dalam *bytes*), yakni panjang *postings list* yang disimpan.

Tambahannya pada Tugas Pemrograman 2 adalah **tf_list_size** (dalam *bytes*), yakni panjang *list* yang berisi informasi *term frequency*. Informasi posisi tidak diperlukan karena *TF list* disimpan persis setelah *postings list* sehingga posisinya adalah start_position + list_size. Jangan ubah *TF list* ke *gap lis*t. Ingat bahwa *TF list* tidak dijamin terurut.

Berikut adalah ilustrasinya.



Anda diberikan *template* program untuk memudahkan Anda dalam menyusun program. Terdapat beberapa komponen penting dalam *template* program ini:

- bsbi.py
- index.py
- compression.py
- util.py
- search.py
- experiment.py
- queries.txt
- qrels.txt

Bagian yang perlu Anda lengkapi sudah ditandai dengan comment # TODO dan Anda dibantu dengan dokumentasi yang tertulis pada setiap *function* untuk mengerjakan tugas ini. Selain berisi program, beberapa *file* juga telah diisi *sample test case* 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.

Langkah Pengerjaan

Pengerjaan dibagi menjadi 2 bagian. Bagian 1 akan berfokus pada implementasi *ranked retrieval model*. Bagian 2 akan berisi eksperimen dan evaluasi perbandingan TF-IDF dan BM25 dengan berbagai parameter. Untuk seluruh langkah, Anda dapat menggunakan implementasi yang sudah Anda kerjakan pada Tugas Pemrograman 1 untuk bagian yang beririsan. Anda juga diperbolehkan untuk membuat *function* tambahan lain bila diperlukan untuk membantu pengerjaan.

Bagian 1 (Implementasi):

1. Unduh dan ekstrak *dataset* sebagai sebuah folder bernama collections/ pada direktori template kode Anda sedemikian sehingga struktur direktori Anda adalah sebagai berikut.

```
|- collections
|   |- 0
|   |   |- 1029.txt
|   |   |- 1032.txt
|   |   |- ...
|   |- 1
|   |   |- ...
|   |- ...
|- bsbi.py
|- experiment.py
|...
```

2. Lakukan implementasi dalam program **util.py**. *Class* dan *function* dalam program ini bertujuan untuk mempermudah representasi dokumen atau *term* sebagai *integer*, yang akan digunakan di dalam program utama lain, sebagaimana pada tugas pemrograman sebelumnya. Perbedaannya adalah kini Anda akan membuat *function* untuk menggabungkan dua *lists of tuples (document ID, TF)*. Uji implementasi Anda dengan menjalankan *test* yang sudah disediakan di bagian akhir program.
3. Lakukan implementasi dalam program **compression.py**. *Class* dan *function* dalam program ini bertujuan untuk melakukan konversi *postings* (*list of document ID*) menjadi *sequence of byte*, sebagaimana pada tugas pemrograman sebelumnya. Uji implementasi Anda dengan *test* yang disediakan.
4. Lakukan implementasi dalam program **index.py**. *Class* dan *function* dalam program ini bertujuan untuk mengatur penggunaan dan penyimpanan *inverted index* ke *file* sebagai *storage*, sebagaimana pada tugas pemrograman sebelumnya. Perbedaannya adalah kini Anda akan melakukan beberapa penyesuaian *inverted index* menjadi

seperti ilustrasi di atas, khususnya pada *function* get_postings_list dan append. Uji implementasi Anda dengan *test* yang disediakan.

5. Lakukan implementasi dalam program **bsbi.py**. Dalam program ini, *class* BSBIIndex berisi *function* terkait proses *indexing*. Beberapa *function* penting yang perlu Anda perhatikan adalah sebagai berikut.
    a. merge_index: Menggabungkan *inverted indices*
    b. do_indexing: Menggunakan berbagai *function* sebelumnya untuk melakukan *indexing*
    c. retrieve_tfidf: Melakukan perhitungan *similarity score* dan mengembalikan *top-k relevant documents*. Lakukan perhitungan TF-IDF dengan *framework* **TaaT (Term-at-a-Time)**.

Setelah selesai mengimplementasi pada bagian 1, Anda dapat menguji rangkaian program dengan langkah sebagai berikut.

1. Eksekusi program **bsbi.py** untuk melakukan *indexing*. *Output* dari tahap ini adalah folder berisi file *index*, *posting-dictionary*, dan file pendukung lain.
2. Eksekusi program **search.py** untuk melakukan *searching* menggunakan *index* yang telah dibangun.

Bagian 2 (Eksperimen dan Evaluasi):

Secara garis besar, tujuan dari bagian ini adalah melakukan implementasi skema BM25 dan membandingkannya dengan TF-IDF yang sudah diimplementasikan pada bagian 1. Pada dasarnya, Anda bebas berkreasi pada *file* **experiment.py** ini dan boleh mengubah program ini selama mencapai tujuan yang sama. Beberapa langkah penting yang Anda perlu lakukan:

1. Tambahkan *function* **retrieve_bm25** pada program **bsbi.py** yang melakukan *scoring* dengan skema BM25 dan *framework* TaaT. Beberapa catatan:
    - Informasi panjang setiap dokumen ada di *instance variable* self.doc_length di *class* InvertedIndex pada index.py. Informasi rata-rata panjang dokumen di seluruh koleksi bisa juga dihitung dari *instance variable* tersebut.
    - Perhatikan bahwa akan sangat tidak efisien bila rata-rata panjang dokumen dihitung setiap kali *function* retrieve_bm25 dipanggil. Implementasi yang efisien hanya melakukan perhitungan sekali saja dan disimpan, misalkan di saat membangun InvertedIndex.
2. Lengkapi program **experiment.py**. Program ini bertujuan untuk mengevaluasi performa dari TF-IDF dan BM25 yang telah Anda buat. 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 *function* RBP sudah diberikan sebagai contoh. Perhatikan bahwa Anda diberikan **queries.txt** (berisi 150 *query*) dan **qrels.txt** (berisi *document ID* yang relevan untuk setiap *query*). Gunakan kedua *file* ini untuk mendapatkan skor metrik-metrik tersebut.
3. Jalankan program **experiment.py** dengan variasi nilai **k1** dan **b** pada BM25. Anda diharapkan mencoba minimal 3 variasi parameter tersebut.

Buat *file* dokumen penjelasan hasil eksperimen yang Anda kerjakan dengan penamaan *file* **Tugas2_Experiment.pdf**. Dokumen perlu berisi hasil eksperimen pada bagian 2. Berikut adalah contoh tabel hasil eksperimen yang diharapkan ada dalam dokumen.

Scoring Regime	**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

Sertakan juga analisis Anda terhadap hasil yang Anda dapatkan. Contohnya adalah skema mana yang menurut Anda paling baik dan buruk, beserta alasannya. Anda boleh menyertakan analisis atau *insight* lain yang anda dapatkan selama eksperimen ini dilakukan.

Catatan Tambahan:

- Soal dan jumlah data pada tugas ini dibuat sedemikian sehingga durasi eksekusi program dari awal hingga akhir dapat selesai dalam 45 menit (30 menit *indexing*, 15 menit evaluasi) atau lebih cepat. Bila eksekusi program Anda memakan waktu yang jauh lebih lama, silakan perhatikan kembali implementasi Anda.
- Soal, metode, dan jumlah data pada tugas ini dibuat untuk mampu dijalankan dengan jumlah *memory* RAM yang minim (< 200MB). Bila Anda mengalami isu *out-of-memory*, silakan perhatikan kembali implementasi Anda, terutama *variable* yang disimpan dalam *memory*.

Bonus

Pada bagian 1 dan 2, Anda telah 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 (silahkan cari referensi dengan *search engine* favorit Anda). 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*. Lakukan evaluasi kembali dan bandingkan performanya dengan TaaT. Sertakan secara detail proses dan hasil yang Anda dapatkan di dalam laporan.

Pengumpulan

Kumpulkan sebuah *file* .zip dengan format **Tugas2_NPM.zip** (contoh: **Tugas2_2006123456.zip**) yang berisi:

- util.py
- compression.py
- index.py
- bsbi.py
- search.py

- experiment.py
- *folder* index
- *file* laporan (.pdf)
- *file* terkait pengerjaan bonus bila ada

Anda tidak perlu mengumpulkan *folder* collections.

Penilaian

Komponen	Cakupan	Proporsi
Bagian 1	util.py, compression.py, index.py, bsbi.py	60 poin
Bagian 2	bsbi.py, experiment.py, laporan	40 poin
Bonus	implementasi bonus, laporan	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
- Bonifacio, L. H., Jeronymo, V., Abonizio, H. Q., Campiotti, I., Fadaee, M., Lotufo, R., & Nogueira, R. (2021). mMARCO: A Multilingual Version of MS MARCO Passage Ranking Dataset. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2108.13897

Selamat mengerjakan!