CSCE604135 • Temu-Balik Informasi
Semester Gasal 2024/2025
Fakultas Ilmu Komputer, Universitas Indonesia

Tugas Pemrograman 2: *Ranked Retrieval*

Tenggat Waktu: Kamis, 10 Oktober 2024, 23.55 WIB

Ketentuan:

1. Anda diberikan beberapa *file* kode Python yang berisi *template* kode.
2. Kumpulkan **semua program** (*.py*) dan **laporan** (*.pdf*), yang sudah dibuat dalam format *.zip* dengan penamaan **TP2_NPM.zip** melalui submisi SCeLe.
   Contoh penamaan file: TP2_2006524290.zip
3. Kumpulkan *file zip* tersebut pada submisi yang telah disediakan di SCeLe sebelum **Kamis, 10 Oktober 2024, 23.55 WIB**. Keterlambatan pengumpulan akan dikenakan penalti.
4. Tugas ini dirancang sebagai tugas mandiri. **Plagiarisme tidak diperkenankan dalam bentuk apapun**. Adapun kolaborasi berupa diskusi (tanpa menyalin maupun mengambil jawaban orang lain) dan memanfaatkan informasi dari literatur manapun masih diperbolehkan. **Pastikan** untuk mencantumkan nama kolaborator dan referensi literatur.
5. Anda boleh berkonsultasi terkait tugas ini asisten dosen berikut. Asisten dosen diperbolehkan membantu Anda dengan memberikan petunjuk.
   a. Jaycent Gunawan Ongris
      Email: jaycent.gunawan@ui.ac.id
      Discord: jaycentg
   b. Muhammad Ilham Ghozali
      Email: walangsigit@gmail.com
      Discord: myticalcat

Petunjuk Pengerjaan Tugas

Sebelumnya pada Tugas Pemrograman 1, Anda telah belajar terkait bagaimana mengimplementasikan *retrieval model* sederhana yang berbasis *boolean retrieval*. Seperti yang telah Anda ketahui di kelas, *boolean retrieval* hanya cocok dipakai oleh *expert users*, di mana *query* harus diformulasikan dengan baik agar hasil *retrieval* tidak terlalu luas dan tidak terlalu sempit. Untuk menyelesaikan masalah ini, pendekatan *ranked retrieval* digunakan, di mana terdapat *score* yang menunjukkan relevansi suatu dokumen terhadap *query* yang diberikan oleh *user*. Berdasarkan *score* ini, dokumen-dokumen tersebut diurutkan sedemikian hingga dokumen dengan *score* paling tinggi diletakkan di atas, menunjukkan dokumen tersebut yang paling relevan terhadap *query*.

Informasi Penting

Tugas Pemrograman 2 merupakan kelanjutan dari Tugas Pemrograman 1. Dataset yang digunakan sebagai *collections* sama, sehingga Anda dapat menggunakan dataset tersebut untuk di-*index* kembali. Sebagian besar *codebase* yang digunakan sama, kecuali beberapa bagian yang harus dilakukan modifikasi agar sesuai dengan model *ranked retrieval* yang digunakan, seperti *inverted index* yang tidak hanya menyimpan *list of document IDs*, tetapi juga *list of term frequencies*. Jika sudah benar, silakan gunakan kembali *codebase* dari Tugas Pemrograman 1.

Fokus dari tugas ini lebih kepada eksperimen terkait algoritma dan model yang diimplementasikan, tidak seperti Tugas Pemrograman 1 yang lebih menekankan pada pengembangan sistem *search engine*. Terdapat beberapa *task* yang harus Anda selesaikan pada tugas ini.

Retrieval Model

Pada tugas kali ini, Anda akan mengimplementasikan 2 jenis *retrieval model*, yakni berbasis TF-IDF dan BM25. Ingat bahwa Anda perlu untuk mengubah beberapa bagian saat *index construction*, seperti menambahkan *list of term frequencies* (TF) agar dapat dihitung *score*-nya. Informasi terkait modifikasi lainnya yang diperlukan telah ditulis sebagai *comment* di kode yang terkait. Anda juga dibebaskan untuk melakukan modifikasi lainnya pada kode agar model *retrieval* Anda dapat bekerja dengan baik. Pastikan untuk menambahkan *comment* yang jelas agar alur kode Anda dapat dipahami dengan baik. Berikut adalah model yang harus Anda implementasikan.

1. Model TF-IDF dengan Skema Term-at-a-Time (TaaT)
2. Model TF-IDF dengan Skema Document-at-a-Time (DaaT)

3. Model BM25 dengan Skema Term-at-a-Time (TaaT); pilih salah satu implementasi saja dari beberapa versi implementasi yang ada di *slide*
4. [Bonus] Model TF-IDF dengan Skema DaaT WAND[1]

Setelah mengimplementasikan kode untuk masing-masing skema di atas, lakukan eksperimen untuk membandingkan performa model *retrieval* yang telah Anda buat dengan menjalankan kode pada *file* `exp_taat_daat.py`. Jika Anda mengerjakan bonus, bandingkan juga performanya dengan ketiga model yang telah Anda buat sebelumnya. Laporkan hasil eksperimen ini dengan memberikan *screenshot* yang menunjukkan waktu yang diperlukan untuk melakukan *retrieval* pada masing-masing model, lalu berikan justifikasi secara singkat terkait hasil eksperimen Anda di dokumen laporan.

Perhatikan bahwa *retrieval* TF-IDF dengan skema TaaT, DaaT, dan WAND pasti memberikan hasil yang sama. Mungkin akan ada sedikit perbedaan di urutan dokumen dengan *score* yang sama, tetapi pastikan bahwa *score* dari dokumen-dokumen tersebut sama. Jika tidak sama, kode Anda salah.

Query Auto-Completion (QAC)

Di kelas, Anda telah diajarkan struktur data *prefix tree* (*trie*) yang berperan penting dalam melakukan *task* QAC. Di tugas ini, Anda akan mengimplementasikan struktur data ini. Silakan mempelajari kode pada *slide* terkait QAC untuk informasi yang lebih detail, lalu implementasikan beberapa *method* tambahan pada *class* `Trie` agar dapat menjalankan task QAC dengan baik. Dokumentasi terkait *methods* yang ingin diimplementasikan telah disertakan di bagian kode yang terkait. Anda hanya perlu melengkapi kata terakhir pada *query user* dan cukup sebanyak satu kata saja. Pembentukan *trie* dilakukan pada saat *indexing* dan pastikan kata yang Anda masukkan ke *trie* adalah *raw token* sebelum dilakukan *pre-processing* (terutama *stemming*). Penghapusan *stopwords* juga diperbolehkan. Jika ada beberapa kata yang memiliki frekuensi kemunculan yang serupa, tidak perlu ada urutan spesifik terhadap kata-kata tersebut (i.e., tidak perlu mengurutkan berdasarkan abjad sebagai *tiebreaker*, dll.); cukup gunakan urutan kata sesuai yang dikeluarkan algoritma buatan Anda saja.

[Bonus] Implementasi QAC pada kode yang ada saat ini merupakan versi yang *naïve* dan tidak efisien. Lakukan *improvement* pada kode yang ada agar *retrieval* kandidat *subwords* lebih cepat. Anda dapat menggunakan *Range Maximum Queries* (RMQ) seperti yang tertera pada *slide*, baik yang menggunakan *index* ataupun tidak (cukup satu saja). Jika Anda mengerjakan ini, tunjukkan perbedaan performa

[1] Anda bebas menambahkan informasi lain di *index* agar dapat memudahkan implementasi WAND. Akan tetapi, ingat untuk tetap menambahkan penjelasan terkait modifikasi ini di kode Anda.

implementasi *naïve* dengan implementasi yang menggunakan RMQ di dokumen laporan. Anda tidak perlu menulis apapun di dokumen laporan untuk bagian ini jika tidak mengerjakan bonus, cukup kodenya saja.

IR Evaluation

Pada bagian ini, Anda akan belajar terkait bagaimana mengevaluasi hasil *retrieval* oleh *search engine*. Untuk bisa melakukan *retrieval*, terdapat *ground truth* yang diperlukan untuk dibandingkan dengan hasil *retrieval*. Pada tugas ini, asumsikan hasil *retrieval* top-20 BM25 yang akan menjadi *ground truth* untuk kemudian Anda bandingkan dengan hasil *retrieval* top-20 TF-IDF. Daftar *queries* yang digunakan untuk membuat dataset *ground truth* terdapat pada *file* `queries.txt`. Hasil dokumen yang relevan menurut model BM25 kemudian disimpan di *file* `qrels.txt`. Setelah itu, lakukan *retrieval* untuk model TF-IDF dengan *queries* yang sama, lalu evaluasi hasilnya menggunakan metrik yang telah ditentukan di *class* `Metrics`. Kode ini dapat dilihat pada file `exp_evaluation.py`. Terakhir, dokumentasikan hasil evaluasi Anda dengan memberikan *screenshot* evaluasi di dokumen laporan. Berikan juga interpretasi Anda secara singkat terhadap hasil evaluasi ini, apakah hasil TF-IDF mirip dengan BM25 atau tidak.

TF-IDF vs. BM25

Seperti yang Anda ketahui, TF-IDF dan BM25 merupakan dua *retrieval model* yang berbeda. Akan tetapi, *hyperparameter* dari BM25 dapat disesuaikan agar *behavior*-nya mirip dengan TF-IDF. Mungkin dari segi *score* yang dihasilkan tidak akan sama, tetapi setidaknya urutan dari dokumen-dokumen yang dihasilkan diharapkan dapat mirip antara kedua model ini. Pada bagian ini, Anda akan coba melakukan *hyperparameter tuning* terhadap model BM25 agar performanya mirip dengan TF-IDF. Silakan tentukan sendiri kandidat *hyperparameters* yang diinginkan, lalu simulasikan *grid search* untuk mencari *hyperparameter* terbaik agar performanya mirip dengan TF-IDF. Anda juga dibebaskan untuk memilih fungsi *objective* yang sesuai untuk dimaksimalkan (atau diminimalkan). Silakan berikan interpretasi Anda terkait pengaruh *hyperparameters* model BM25 terhadap kemiripan urutan dokumen yang dihasilkan dengan model TF-IDF. Anda boleh menggunakan *class* `Metrics` yang telah dibuat pada *task* terkait *IR Evaluation*. Kerjakan ini di *file* `exp_hyperparameter.py`.

Simulasikan *grid search* sendiri tanpa menggunakan *library* `sklearn` ataupun *library* lain yang berkaitan.

Dataset

Sama seperti pada Tugas Pemrograman 1, Anda akan bekerja dengan sampel dataset dari kumpulan abstrak dokumen di arXiv dalam bahasa Inggris. Total dokumen yang ada sebanyak kurang lebih 250.000 dokumen yang tersebar di 25 *folder*, di mana masing-masing *folder* berisi kurang lebih 10.000 dokumen. Tiap *folder* akan merepresentasikan satu *intermediate index*.

Bonus

Anda akan diberikan bonus 10 poin jika Anda mengimplementasikan salah satu dari kedua *task* yang disebutkan berikut ini (juga sudah dijelaskan secara detail di deskripsi soal di atas).

- Implementasi TF-IDF dengan WAND, termasuk laporan perbandingannya
- Implementasi RMQ, termasuk laporan perbandingannya

Contoh Interaksi Program

```
(env) D:\ir-asdos\tp-2 master ~1 python .\search.py
Masukkan query Anda: toroidal orbi
Rekomendasi query yang sesuai:
1. toroidal orbi
2. toroidal orbits
3. toroidal orbitals
4. toroidal orbiting
5. toroidal orbifolds
6. toroidal orbitally

Masukkan nomor query yang Anda maksud: 5

Pilihan Anda adalah 'toroidal orbifolds'.
Hasil pencarian:
arxiv_collections\0\0706.1310.txt 7.941
arxiv_collections\0\0704.1823.txt 7.100
arxiv_collections\0\0705.2150.txt 5.656
arxiv_collections\2\0708.3806.txt 4.865
arxiv_collections\2\0709.3607.txt 4.787
arxiv_collections\1\0706.1974.txt 4.739
arxiv_collections\2\0709.0176.txt 4.417
arxiv_collections\0\0705.3249.txt 4.399
arxiv_collections\2\0710.2432.txt 4.345
arxiv_collections\1\0708.1875.txt 4.335
(env) D:\ir-asdos\tp-2 master ~1
```

Ketentuan urutan rekomendasi *query* dimulai dari *query* aslinya sebagaimana yang dimasukkan pengguna (pada gambar "`toroidal orbi`") sebagai rekomendasi pertama, yang kemudian diikuti oleh top-5 rekomendasi berdasarkan pencarian *trie*, diurutkan berdasarkan frekuensinya dari yang terbanyak.

Template Kode dan Dataset

- Template kode: tp2_template.zip
- Dataset: arxiv_collections.zip

Deliverables

Kompres semua *files* dan *folders* berikut ini ke dalam suatu zip *file* dengan format penamaan seperti yang disebutkan pada bagian sampul dokumen soal ini. Anda tidak perlu mengumpulkan *folder* `arxiv_collections`, cukup satu *folder index* yang dibuat saja. Jika Anda mengerjakan bonus dan ingin memisahkan *file* untuk implementasi WAND dan/atau RMQ, silakan menambahkan *file* lain sesuai keperluan Anda. Khusus untuk WAND, jika Anda memodifikasi *index*, pastikan untuk mengumpulkan *index* tersebut juga.

- `bsbi.py`
- `compression.py`
- `index.py`
- `search.py`
- `util.py`
- `trie.py`
- `exp_evaluation.py`
- `exp_hyperparameter.py`
- `exp_taat_daat.py`
- `qrels.txt`
- `queries.txt`
- `/index`
- `laporan.pdf`

Jika *folder index* yang dihasilkan terlalu besar sehingga tidak bisa di-*upload* di SCeLE, silakan meng-*upload folder* tersebut di *cloud storage* seperti Google Drive terlebih dahulu, lalu sertakan URL-nya di dalam zip *file deliverables*. Pastikan URL tersebut dapat diakses.

Catatan Revisi:

- [2 Oktober 2024 08:25] **Penggantian *test case* untuk *file* `trie.py`. Silakan akses URL berikut ini untuk *file* `trie.py` dengan *test case* yang baru.**
- [2 Oktober 2024 16:29] **Penambahan informasi tambahan terkait *interface* yang diinginkan untuk urutan *query* rekomendasi.**

Rubrik Penilaian

Komponen	Proporsi
Implementasi *retrieval* TF-IDF ● Skema TaaT: 10% ● Skema DaaT: 10%	20%
Implementasi *retrieval* BM25	10%
Implementasi *file* `util.py`, `bsbi.py` (selain *method* `retrieve`), `compression.py`, `index.py`, dan `search.py`	25%

Implementasi *trie* untuk QAC	15%
Implementasi *IR evaluation*, termasuk pembuatan metrik-metrik yang sesuai dan laporan eksperimen	15%
Laporan eksperimen berbagai *method retrieval*	5%
Implementasi dan laporan eksperimen *hyperparameter tuning* BM25	10%
Bonus: WAND atau RMQ beserta laporannya	10%

Selamat mengerjakan!