بازیابی پیشرفته اطلاعات(فاز اول)

نیم سال 4042

دانشکده‌ی مهندسی کامپیوتر – آزاد شیراز

مدرس: دکتر اسکندری

طراحان: حمید نامجو، امیرحسین همتی و علی مجاهد

زمان تحویل: 23 اردیبهشت

- پروژه انفرادی و یا نهایتا 2 نفره است.
- فقط در موارد ذکرشده مجاز به استفاده از کتابخانه‌های آماده هستید.
- هر گروه باید کد پروژه خود را در مهلت اعلام شده ارسال نماید.
- علاوه بر گزارش پروژه، در انتهای ترم یک ارائه و دفاع نیز از این پروژه گرفته خواهد شد. مهلت ارسال پاسخ تا ساعت 23:59 روز مشخص شده است.
- همکاری و هم فکری شما در انجام پروژه مانعی ندارد اما پاسخ ارسالی هر گروه حتما باید توسط خود او نوشته شده باشد.
- در صورت هم فکری و یا استفاده از هر منبع خارج درسی، نام هم فکران و آدرس منابع مورد استفاده برای حل سوال مورد نظر را ذکر کنید.

1 مقدمه

در این پروژه به صورت عملی از مفاهیم تدریس شده در کلاس درس استفاده می‌شود. در این پروژه هر دانشجو یک موتور جستجو برای بازیابی اسناد طراحی می‌کند؛ به‌گونه‌ای که کاربر پرسمان خود را وارد و سامانه اسناد مرتبط را بازیابی کند.

2 توضیح تکمیلی

این فاز از پروژه را در ۴ مرحله قرار است تکمیل کنید، ولی قبل از هرچیزی ابتدا باید پروژه را از طریق کوئرا دانلود کنید و آن را از حالت فشرده خارج کنید.

به موارد زیر توجه کنید:

- از تغییر دادن نام کلاس ها و همچنین قرار دادن فایل های پروژه خود در دایرکتوری های مختلف خودداری کنید. در انتها این فاز، پروژه شما با اجرای کد autograder نمره دهی خواهد شد و این کد از نام کلاس های مشخص شده استفاده می‌کند و همچنین فرض می‌کند که تمام فایل ها در کنار این فایل قرار گرفته‌اند.
- سه دسته تابع در کدها وجود دارد:

  - دسته‌ی اول تابع هایی هستند که با کامنت TODO مشخص شده‌اند. این قسمت ها باید توسط شما تکمیل شوند.
  - دسته‌ی دوم تابع هایی هستند که با کامنت PROVIDED (keep) مشخص شده‌اند. این قسمت ها معمولا

  helper method هایی هستند که برای اجرای بهتر و راحت تر کد مورد نیاز است. بهتر است از تغییر آن ها خودداری کنید.

  - دسته سوم تابع هایی هستند که کامنت هایی شبیه DO NOT CHANGE یا leave as-is مشخص شده‌اند. از تغییر دادن این توابع بپرهیزید چرا که اجرای کد autograder را با مشکل مواجه می‌کند.

- پس از اتمام پروژه حتما کد autograder را خودتان نیز یک بار اجرا کنید تا از درستی همه چیز مطمئن شوید.از تغییر این کد خودداری کنید چرا که برای نمره دهی ، این کد به صورت بدون تغییر در کنار پروژه شما جایگزین و اجرا می شود.

- شما می‌توانید هر تعدادی helper method که خواستید به کلاس ها اضافه کنید یا هر بخش که DO NOT CHANGE نخورده است را تغییر دهید. فقط در نهایت از درستی عملکرد autograder اطمینان حاصل کنید.
- استفاده از کتابخانه‌های آماده برای پیاده سازی متدها و توابع غیر مجاز است و باید تمام بخش ها را from scratch پیاده سازی کنید.

- توجه کنید که آن چیزی که در نهایت برای ارزیابی پروژه شما بررسی می‌شود، کارکرد آن در بازیابی درست اسناد است؛ به همین علت فارغ از انجام تمام مراحل ذکر شده در ادامه، از بازگرداندن اسناد درست با یک کوئری مشخص اطمینان حاصل کنید.

حال برای تکمیل پروژه کافی است مراحلی که در ادامه آورده شده است را به درستی و به ترتیب انجام دهید.

3 مرحله صفر – Data Collecting

در این مرحله شما قرار نیست کدی بزنید و فقط کافیست که دیتا را به یکی از روش های زیر دریافت کنید.

3.1 روش اول

فایلی در پروژه به نام crawler.py قرار دارد ، این فایل را اجرا کنید. از آنجایی که دیتا از imdb دریافت می‌شود، یک بار بدون تغییر IP فایل را اجرا کنید و در صورتی که به مشکل برقراری اتصال خوردید، ابتدا IP خود را تغییر داده و سپس فایل را اجرا کنید.

3.2 روش دوم

فایل دیتا را از طریق لینکی که در اختیار شما قرار گرفته است، دریافت کنید و آن را در کنار دیگر فایل های پروژه قرار دهید.

پس از این که دیتا را به هر یک از روش های بالا دریافت و در کنار دیگر فایل های پروژه قرار دادید، فایل validator.py را اجرا کنید تا از درستی دیتا اطمینان حاصل کنید. این کد پایتون سه آرگومان به صورت optional دریافت می‌کند:

- Data path: در صورت که دیتای دریافت شده را در آدرسی به غیر از آدرس دیفالت که root پروژه است نگه‌داری می‌کنید، آن را به عنوان اولین آرگومان هنگام اجرای کد بدهید. در صورتی که این آرگومان را ندهید آدرس دیفالت IMDB_crawled.json می‌باشد.

- min-count: تعداد رکورد موجود در دیتا را مشخص می‌کند. به صورت دیفالت ۱۰۰۰ است چرا که برای اجرای نهایی باید حداقل ۱۰۰۰ سند را بررسی کنید.

- fail-on-warning: در دیتا یک تعدادی فیلد خالی وجود دارد که این اتفاق در دیتای دنیای واقعی خیلی معمول است. در بررسی درستی دیتا در صورتی که یکی از فیلدها خالی باشد یک warning به ازای آن دریافت می‌کنید.

نیازی به استفاده از این آرگومان نیست چرا که دیتا تعدادی فیلد خالی دارد.

قبل از اینکه به سراغ دیگر مراحل بروید، بهتر است که یک نگاهی به دیتا بیاندازید. به صورت خلاصه این دیتا شامل یک مجموعه‌ای از فیلم های موجود در سایت imdb است. این فیلم ها به صورت تصادفی انتخاب نشده‌اند بلکه از طریق خویشاوندی یک فیلم با دیگر فیلم ها انتخاب شده‌اند. به عبارتی می‌توان فیلم ها را در کلاسترهای مختلف دسته بندی کرد. روش ساخت این دیتا از طریق فایل crawler قابل بررسی است.

4 مرحله اول – Data Preprocessing

در این مرحله هدف نهایی این است که یک preprocessing pipeline طراحی کنیم که در آن دیتای اولیه‌ای که در مرحله قبل بدست آورده‌ایم را به دیتای فیلدبندی شده، Tokenize و lemmatized شده تبدیل کنیم تا برای ایندکس گذاری در مرحله بعدی مناسب باشد. فیلدهایی که در این فاز از پروژه قرار است از آن ها برای بازیابی دیتا مورد نظراستفاده کنیم موارد زیر است.

- title
- Summaries
- Genres
- Stars

برای کامل کردن این مرحله باید در فایل preprocess متدهای زیر را پیاده‌سازی کنید:

4.1 clean_text

- کل متن را به حروف کوچک تبدیل کنید.
- آدرس های اینترنت را حذف کنید.
- هر رشته‌ی placeholder را حذف کنید (به self.placeholder_strings مراجعه کنید.)
- برای ترکیبات رایج عناوین فیلم، هر دو فرم چسبیده و فاصله‌دار را نگه دارید؛

  برای مثال spider man spiderman ← spider-man
- خط تیره‌های باقی مانده را با فاصله جایگزین کنید.
- کاراکتر های غیرحرف/عدد را حذف کنید، به جز فاصله و نشانه‌های ".,?!:;"
- فاصله‌های تکراری را یکی کنید و ابتدا/انتهای رشته را Trim کنید.

4.2 Process_movie

- فقط فیلدهای title, summaries, genres, stars را در نظر بگیرید.
- summaries را با "فاصله" به هم بچسبانید: سایر فهرست ها را با "," به هم وصل کنید.
- فیلدهای خالی یا Null را تغییری ندهید و رد کنید.
- اگر متن فیلد حاوی هر placeholder بود، آن فیلد را تغییری ندهید و رد کنید.
- clean_text و tokenize_and_lemetize را روی اسناد اجرا کنید.
- دیکشنری ای برگردانید که field ← token باشد.

مراحل بالا به صورت کامنت نیز در متدهای خالی موجود در فایل preprocess.py نوشته شده است که به شما در کامل کردن این فایل کمک می‌کند.

توجه: signature متدها و نام کلاس‌ها را تغییر ندهید. autograder کلاس DataProcessor را import می‌کند وهمین متدها را فراخوانی خواهد کرد.

۵ مرحله دوم - Indexing

حالا از توکن های تمیز یک Fielded Inverted Index می‌سازیم تا جست وجو سریع و هدفمند شود. برای هر term مشخص می‌کنیم در چه سندهایی و در کدام فیلدها آمده، چند بار تکرار شده (tf) و دقیقا در چه position هایی دیده می‌شود، و همچنین df آن term را نگه می‌داریم. علاوه بر خود ایندکس، متادیتاهایی مثل total_documents و توزیع طول ها ذخیره می‌شود تا بعدا پارامترهای sorting بهتر تنظیم شوند. نتیجه یک ساختار داده سریع و سبک است که پایه مرحله بعد است.

باید متدهای زیر را در indexer.py پیاده‌سازی کنید:

5.1 Build_index_structure

- روی هر doc_id و هر فیلد در ["title", "summaries", "genres", "stars"] تکرار کنید.
- فیلدهای خالی یا ناموجود را بدون تغییر بگذارید.
- آمارها را به‌روزرسانی کنید:
  - field_distribution
  - total_tokens
  - token_length_distribution
- باید unique_tokens را آپدیت نگه دارید.

5.2 Add_or_update_posting

- مقدار tf را با توجه به دوتایی (doc_id, field) آپدیت کنید.
- مقدار positions را با توجه به position آپدیت کنید.
- در صورتی که برای (doc_id, field) پستی وجود نداشت باید df را آپدیت کنید.

مراحل بالا به صورت کامنت نیز در متدهای خالی موجود در فایل indexer.py نوشته شده است که به شما در کامل کردن این فایل کمک می‌کند.

توجه: signature متدها و نام کلاس ها را تغییر ندهید. autograder کلاس FieldedInvertedIndex را import می‌کند و همین متدها را فراخوانی خواهد کرد.

6 مرحله سوم – Query Parser

در این مرحله هدف نهایی ساخت یک query parser است که ورودی یک query را از کاربر دریافت می‌کند و به عنوان خروجی باید وزن فیلدهای جست وجو را خارج کند و سپس عبارت جست‌وجو را از آن جدا کند. توجه کنید که عبارت جست وجو باید به ترتیب وارد شده در Query نگه داشته شود. در مرحله نهایی باید spelling correction را روی کلمات عبارات جست‌وجو اعمال کنید. برای مثال اگر عبارت spder-man در Query وجود داشته باشد، باید به spider man و یا spiderman تغییر پیدا کند.

6.1 Preprocess_quotes

بدون به هم خوردن ساختار و ترتیب کلمات داخل Quote ها، متن درون آن ها را خارج کنید.

6.2 Levenshtein_distance

در این متد می‌خواهیم فاصله‌ی ویرایش levenshtein را برای دو رشته بدست آوریم.

- مطابق جزوه به شکلی الگوریتم را پیاده سازی کنید که حافظه کمی بگیرد تا به مشکل حافظه بر نخورید.
    - هزینه تفاوت دو رشته را مطابق جزوه در نظر بگیرید.
- در نهایت خروجی که یک عدد صحیح نامنفی که فاصله دو رشته است باید برگردانده شود.

6.3 correct_spelling و تابع‌های کمکی آن

هدف این متد اصلاح املایی تک واژه‌های Query بر اساس vocabulary لود شده از ایندکس های مرحله قبل است.

- واژه‌های بسیار رایج را دست نخورده برگردانید. (واژه‌هایی مثل: ... ,and, for, a, an, is, on)
    - اگر واژه در vocabulary وجود دارد همان را برگردانید.

- پس از محاسبه Levenshtein، در صورتی که فاصله دو term کمتر مساوی ۲ (یا هر عددی که به نظر شما مناسب‌تر است) بود آن را به عنوان یک نامزد در نظر بگیرید و سپس در صورتی که به تساوی فاصله میان دو نامزد رسیدید، نامزدی را در نظر بگیرید که با term موجود در Query هم طول باشد و همچنین df آن بیشتر باشد.

6.4 Parse

تمام فرایند parse باید در اینجا قرار بگیرد و خروجی آن همانطور که در ابتدا گفته شد دو چیز است:

۱. یک دیکشنری از وزن فیلدها

۲. رشته‌ی process شده‌ی جست وجو

- ابتدا quotes را حذف کنید و سپس رشته داده شده را بر اساس فاصله به term های جدا از هم تقسیم کنید.

- وزن ها را از Query استخراج کنید.
- روی term های Query به صورت جداگانه correct_spelling اجرا کنید.
- حواستان باشد که وزن های استخراج شده باید normalized شوند.

7 مرحله چهارم – Search

اینجا با استفاده از ایندکس وارونه، اسناد را نسبت به کوئری ، امتیازدهی و رتبه‌بندی می‌کنیم. هسته رتبه بندی BM25 است.با length normalization و اعمال field_weights که از Query آمده. اگر phrase های متوالی (بافاصله کوچک در positions) پیدا شوند، با phrase boost نمره سند بالاتر می‌رود تا تطبیق های طبیعی‌تر جلو بیفتند. نتیجه، لیست top-k اسناد مرتب شده است و با explain() هم می‌شود دید هر term و هر field چقدر در امتیاز نهایی اثر گذاشته‌اند.

7.1 compute_doc_length_stats

طول هر سند را از مجموع tf های همه‌ی posting ها محاسبه کنید، در doc_lengths ذخیره کنید، و avg doc length را بدست آورید.

7.2 search

با Query Parser وزن فیلدها و term ها را بدست آورید؛ term ها را توکنایز کنید؛ برای هر توکن موجود در ایندکس،IDF و مؤلفه‌ی BM25 را با نرمال سازی طول سند و وزن فیلد حساب کنید و امتیاز هر سند را جمع بزنید سپس نتایج را بر اساس امتیاز مرتب کنید و top-k را همراه با (doc_id, score, detailed_token_scores) برگردانید.

7.3 identify_phrases

روی لیست توکن ها پنجره‌های ۳ -۲ کلمه‌ای بسازید و هر پنجره را با is_common_phrase تایید کنید. در نهایت باید دو دیکشنری بسازید؛ یکی از token به phrase و یکی position به phrase.

7.4 is_common_phrase

در اینجا بررس می‌کنید که همه‌ی کلمات در ایندکس باشند؛ اشتراک اسناد آن ها حداقل ۲ سند باشد؛ همچنین فاصله میانگین موقعیت بین دو term را از طریق average_token_distance بدست آورید و برای آن یک threshold قراردهید.

7.5 average_token_distance

به طور خلاصه این متد تخمینی از فاصله معمول بین توکن های متوالی یک عبارت ارائه میدهد.

7.6 apply_phrase_boosts

این متد برای افزایش امتیاز هر سند بسته به وجود term های مشابه میان عبارت مورد بررسی و سند است.
همچنین برای این سنجش از has_flexible_sequential_positions استفاده کنید.

7.7 has_flexible_sequential_positions

در این متد بررسی می‌کنید که توکن ها به صورت تقریبا متوالی (مثلا با گپ کمتر از ۳ یا هر عددی که به نظر شما بهتراست) در سند وجود دارند یا خیر.

جزئیات و راهنمایی های لازم برای پیاده سازی متدهای بالا را می توانید از داخل کامنت های هر متد مطالعه کنید.

8 ارزیابی پروژه

در این بخش از پروژه شما باید از درستی عملکرد پروژه خود اطمینان حاصل کنید. برای راحتی شما در ارزیابی درست پروژه کافی است که فایل autograder را اجرا کنید و مطمئن شوید تمام موارد passed می‌شوند و هیچ failed ای اتفاق نمی‌افتد. توجه کنید که این فایل فقط خروجی نهایی شما را نمی‌سنجد

بلکه تمام بخش های پروژه را بررسی می‌کند. به عبارتی خروجی بخش های indexer، parse_query preprocess، و search به صورت جداگانه بررسی خواهد شد.

موفق و موید باشید):