WikiSearch

Даниел Халачев

7 февруари 2025 г.

1 Задача

1.1 Лична мотивация

Да се приложат на практика част от теоретичните познания, придобити от курсовете по *Проектиране и обработка на естествен език* и *Изличане на информация*. Запознаване с езика за програмиране Python чрез изграждане на напълно функциониращa система.

1.2 Техническа задача

- Да се създаде търсачка на уебсайтове с общо предназначение, която извършва търсенето въз основа на два възможни подхода:
  - семантично търсене чрез document embeddings;
  - традиционно индексиране чрез обратен индекс и търсене и ранкиране чрез TF-IDF и BM25.
- По възможност да се имплементират допълнителни функционалности за подобряване на потребителското изживяване, като се използват изградените индекси:
  - предложения за корекции в правописа;
  - предложения за допълване на фразата на търсене.
- Да се оцени системата чрез обективни метрики.

Проектиране и обработка на естествен език	**Изличане на информация**
Изграждане на модул за премахване на стоп думи, токенизация и лематизация	Изграждане на web crawler
Векторизация на документите	Изграждане на обратен индекс.
Изграждане на векторен индекс	Имплементиране на търсене и ранкиране на резултатите с BM25
Избор на стратегия и имплементация на търсене по изградения векторен индекс	Предоставяне на допълване на заявката за търсене и корекция на правописа

Таблица 1: Конкретни задачи по дисциплините

2 Данни

За изграждане на системата беше използван корпус от всички статии в българския поддомейн на Уикипедия (`bg.wikipedia.org`). Целият корпус възлиза на 441 385 статии, от които 302 500 са конкретни, а останалите — пренасочващи. Корпусът беше изтеглен от файла `bg-wiki-20250120-pages-articles.xml.bz2`[1]

Разархивиран, той има размер от 3.0 GB и съдържа всички страници на `bg.wikipedia.org` в WikiMarkup формат и техните метаданни, организирани йерархично в XML дърво. С цел поетапна многостъпкова обработка, бяха извлечени само данните на страниците на статии, след което бяха преработени до чист текст, съхранен в ефективна база данни.

3 Избрани подходи, техники и експерименти

3.1 Лематизация

Премахването на стоп думи, токенизацията и лематизацията бяха осъществени чрез библиотеката `spacy` и модела `sakelariev/bg-news-lg`[2].

3.2 Векторизация

За изграждането на вектори от текстовете на документите беше използван моделът Alibaba[3], приложен чрез библиотеката `sentence-transformers`. Избраният модел върви с препоръчителния към него токенизатор.

3.3 Изграждане на индекса

Първоначалният избор за изграждане на индекса беше популярната библиотека за векторно търсене FAISS. Използването ѝ породи два проблема:

- Липса на поддръжка на множество стойности (вектори) за един ключ (документ). Проблемът беше решен чрез поддържане на речник на съответствията между ключа на вектора във FAISS и документа, от който е получен.
- Лоша производителност — бавно изграждане на индекса и бавно търсене в него. За отстраняването на този проблем бяха изпробвани множество техники:
  - Максимална ефективност на индексирането чрез обработка на множество вектори наведнъж;
  - Прекратяване на всички останали процеси на машината, на която се извършва индексирането;
  - Опити с различни видове индекси (HNSW, IndexL2Flat и др.);
  - Опит за извършване на индексирането на мощен сървър — въпреки това процесът остана недостатъчно бърз.

Нито една от приложените техники не доведе до решение, което може да бъде изпълнено до крайния срок на проекта.

Решението беше замяната на FAISS с USearch[4], библиотека, която за разлика от FAISS:

- Поддържа съпоставянето на множество вектори с един ключ;
- Предоставя по-добра производителност при изграждането и търсенето в индекса.

За съжаление, въпреки ускорение на процеса от 2 до 4 пъти, времето, загубено в опитите с FAISS, не позволи изграждането на индекс върху всички 441 385 статии в Уикипедия.

3.4 Търсене

Размерността на входа модела за векторизация не е достатъчна, за да обезпечи адекватно векторизиране на най-дългите документи в Wikipedia. За целта беше необходимо текстовете да се разделят на секции, които могат да бъдат векторизирани. Използването на USearch[4] позволи на един документ да бъде съпоставен неограничен брой вектори. Това обаче породи следния проблем: в резултатите от търсенето един документ може да се появи многократно (на различни позиции и с различна релеватност). Бяха разгледани и имплементирани три стратегии за справяне с този проблем:

- **Sum Pooling** — от всички двойки (документ, близост) за един документ се формира двойка (документ, сума на близкостите);
- **Min Pooling** — от всички двойки (документ, разстояние) за един документ се избира тази с най-малко разстояние;
- **Average Pooling** — за всяка двойка (документ, близост) се изчислява средната стойност на близостите.

И трите техники показаха сходни резултати.

3.5 Оценяване

Първоначалната идея за оценяване на системата върху целия домейн `bg.wikipedia.org` вече не беше приложима поради неочаквано бавното изграждане на индекса. За демонстрация, че индексът работи, бяха избрани 1000 произволни статии, които да бъдат добавени съответно във векторния и обратния индекс. За златен стандарт беше избрана библиотеката ElasticSearch, с доказана репутация и ефективност в областта. Беше създаден специален модул за генериране на заявки за търсене и оценка на резултатите. Могат да бъдат генерирани групи заявки от два вида с произволен брой и разпределение на всеки вид в групата:

- заглавие на произволен документ
- избор на най-често срещаните колокации от две думи в произволни документи, които се срещат поне 3 пъти в текста.

4 Резултати

Резултатите от WikiSearch бяха сравнени с резултатите на ElasticSearch по критериите Precision, Recall и F1. За целта бяха създадени 5 групи от по 30 заявки за търсене, които очакват до 20 резултата. За всяка заявка се изчисляват Precision, Recall и F1. Чрез тях се изчисляват средните стойности на тези показатели за всяка група. Оценителят връща окончателен резултат 95%-доверителен интервал за всяка метрика Precision, Recall, F1, изчислен върху всички 5 групи от по 30 заявки. По този начин се гарантира представителност на статистическата извадка, защото са изпълнени условията за прилагането на Централна гранична теорема.

Precision	Recall	F1
0.870 ± 0.034	0.893 ± 0.030	0.877 ± 0.030

Таблица 2: Резултати на WikiSearch за обратния индекс

Precision	Recall	F1
0.130 ± 0.021	0.420 ± 0.055	0.142 ± 0.020

Таблица 3: Резултати на WikiSearch за векторния индекс

Резултатите за обратния индекс доказват успешността на подхода за премахване на стоп думи, токенизация и лематизация. Резултатите за векторния индекс са на пръв поглед изненадващи, но по-задълбочен анализ може да посочи причини за ниските стойности:

- неподходящ модел за векторизация - избраният модел беше посочен като съвместим за български език, отличаваше се с най-голяма популярност сред тези, поддържащи български език, но е възможно посочените за модела високи показатели да не са същите конкретно за българския език
- резултатите от векторния индекс се сравняват с резултати, върнати от TF-IDF индекс. По същество, двата типа индекси имат различна философия. Векторният индекс връща желания брой резултати, подредени по релевантност (т.е.

включва и напълно нерелевантни документи, ако сме посочили голямо число за желания брой резултати). Обратният индекс от своя страна връща само "релевантните"документи - т.е. само тези документи, в които се среща лема от заявката поне веднъж. С оглед на малкия брой документи в индекса, се очаква и броят релевантни документи да е значителни по-малък от 20. За $n = 5$, резултатите са:

Precision	Recall	F1
0.308 ± 0.035	0.450 ± 0.056	0.295 ± 0.028

Таблица 4: Резултати на WikiSearch за векторния индекс за $n = 5$

Наблюдава се почти двукратно увеличение, което, въпреки че резултатът все още е много нисък, потвърждава изложената хипотеза.

5 Бъдеща работа

- Изграждане на индекс върху целия домейн `bg.wikipedia.org` и сравнение на резултатите с тези, върнати от Wikipedia, а не от ElasticSearch[1]. *Забележка: Това би било съпроводено с други проблеми в оценяването, защото Wikipedia подобрява резултатите от търсенето чрез графи на знанието.*
- Избор на по-подходящ модел за векторизация на текстовете, предназначен специално за български език.
- Внедряване на интелигентно резюмиране на резултатите.

6 Код

Кодът се съхранява в хранилище на адрес https://github.com/DanielHalachev/WikiSearch.

Литература

[1] Корпусът е достъпен на адрес https://dumps.wikimedia.org/bgwiki/20250120/bgwiki-20250120-pages-articles.xml.bz2.

[2] sakelariev/bg-news-lg, *HuggingFace*, https://huggingface.co/sakelariev/bg_news_lg

[3] Alibaba-NLP/gte-multilingual-base, *HuggingFace*, https://huggingface.co/Alibaba-NLP/gte-multilingual-base

[4] USearch, *Github*, https://github.com/unum-cloud/usearch

[1] Това би било съпроводено с други проблеми в оценяването, защото Wikipedia подобрява резултатите от търсенето чрез графи на знанието.