گزارش پروژه IRWS (بخش دوم)

ساختار پروژه

الف) فایل‌های مربوط به بخش اول (TF-IDF و مقایسه فیلدها)

- src/part1_title_text.py اجرای آزمایش TF-IDF روی سه حالت :
  1. فقط TITLE
  2. فقط TEXT
  3. TITLE+TEXT

و ذخیره خروجی‌ها در :

- metrics_title.csv
- metrics_text.csv
- metrics_title_text.csv

- src/ranker_tfidf.py پیاده‌سازی رتبه‌بند TF-IDF که در ارزیابی‌های بخش اول استفاده می‌شود.
- src/part2_norm_token.py آزمایش پیش‌پردازش فارسی (Normalizer/Tokenizer با Parsivar) و ارزیابی خروجی (خروجی مثل metrics_title_text_norm_tok.csv).
- فایل‌های src/part3_stopwords.py و metrics_stopwords_*.csv مربوط به بخش سوم/آزمایش استاپ‌ورد هستند.

ب) فایل‌های مربوط به بخش دوم (Elasticsearch + تحلیل اندازه ایندکس/پیش‌پردازش)

- src/es_config.pyساخت کلاینت ES و تنظیم اتصال.
- src/es_indexer.py ساختن/پرکردن ایندکس: ایجاد سندها، تعیین doc_id، و ایندکس‌کردن در Elasticsearch.
- src/es_search_eval.py اجرای جستجو روی ES و محاسبه متریک ارزیابی.
- src/part2_index_size.py سناریوی اصلی بخش دوم: اجرای آزمایش برای درصدهای 1,2,4,8,16,32,64,100

ساخت ایندکس برای هر درصد، ایندکس‌کردن، جستجو/ارزیابی، و رسم دو نمودار :

- response_time.png
- ndcg.png

پ) فایل‌های مشترک/زیرساختی (برای هر دو بخش)

- src/data_loader.py بارگذاری دیتاست Hamshahri2 از فایل‌های فشرده (docs/queries/judgments) و ساخت qrels برای ارزیابی.
- src/evaluation_utils.py یک لایه‌ی یکپارچه برای اجرای ارزیابی با BEIR و ذخیره خروجی‌ها در CSV تابع‌هایی مثل evaluate_and_save.
- requirements.txt وابستگی‌ها) کتابخانه‌های لازم مثلBEIR ، parsivar، elasticsearch client و غیره.
- فایل‌های src/metrics_*.csvخروجی نتایج آزمایش‌ها (متریک‌ها) که توسط کدها تولید می‌شوند.

مروری بر بخش اول پروژه و ارتباط آن با پیاده‌سازی جدید

بخش اول این پروژه پیش‌تر به‌صورت کامل در محیط Google Colab پیاده‌سازی شده است. در این بخش، یک سیستم بازیابی اطلاعات مبتنی بر TF-IDF و Cosine Similarity روی دیتاست فارسی همشهری پیاده‌سازی و ارزیابی شد.

ویژگی‌های اصلی بخش اول عبارت بودند از:

- استفاده از مدل برداری (Vector Space Model)
- بررسی تأثیر فیلدهای مختلف:
  - فقط title
  - فقط text
  - ترکیب title + text
- ارزیابی کیفیت بازیابی با متریک های متفاوت

در پیاده‌سازی جدید که به‌صورت محلی (Local) و خارج از Colab انجام شده است:

- هیچ تغییری در منطق الگوریتمی بخش اول ایجاد نشده
- همان کدها و همان تنظیمات مورد استفاده قرار گرفته‌اند
- نتایج به‌دست‌آمده از نظر مقدار کاملاً یکسان با نتایج پیاده‌سازی قبلی هستند

اجرای مجدد پروژه (به‌ویژه در ارتباط با بخش دوم) به دلایل فنی و عملی زیر انجام شده است:

محدودیت‌های Google Colab:

- اجرای پایدار سرویس‌هایی مانند Elasticsearch در Colab دشوار است
- مدیریت پورت‌ها، امنیت (Authentication) و اجرای طولانی‌مدت سرویس‌ها در Colab محدودیت دارد
- Colab برای آزمایش‌های سیستم‌محور (System-level) مناسب نیست

بخش دوم پروژه نیازمند:

- اجرای مداوم Elasticsearch به‌صورت Local
- فعال بودن Security و Authentication
- کنترل کامل روی ایندکس‌ها، تنظیمات و زمان اجرا
- اجرای آزمایش‌های طولانی (بیش از 1 ساعت برای 100٪ داده)

بنابراین، اجرای پروژه در محیط محلی ضروری بوده است.

ارتباط بخش اول و دوم:

- بخش اول نقش Baseline کیفی را ایفا می‌کند.
- بخش دوم سیستم را به یک محیط عملیاتی واقعی (Elasticsearch) منتقل می‌کند.
- مقایسه این دو بخش نشان می‌دهد که:
  - کیفیت بازیابی به‌شدت وابسته به مدل رتبه‌بندی و پردازش متن است
  - صرف استفاده از یک موتور جستجوی صنعتی بدون تنظیمات مناسب، الزاماً باعث بهبود کیفیت نمی‌شود

جمع بندی مربوط به مرور بخش اول:

- بخش اول پروژه قبلاً در Colab پیاده‌سازی، ارزیابی و مستندسازی شده است.
- کدها و نتایج آن بدون تغییر در GitHub موجود هستند.
- اجرای مجدد پروژه روی سیستم شخصی صرفاً برای:
  - یکپارچه‌سازی ساختار پروژه
  - پشتیبانی از بخش دوم
  - و اجرای Elasticsearch به‌صورت پایدار انجام شده است.
- نتایج بخش اول در اجرای جدید کاملاً با نتایج قبلی یکسان هستند.

بخش دوم پروژه (تحلیل تأثیر اندازه ایندکس Elasticsearch بر زمان پاسخ و کیفیت جستجو)

هدف بخش دوم:

هدف این بخش بررسی تأثیر افزایش اندازه ایندکس Elasticsearch بر دو معیار اصلی سیستم بازیابی اطلاعات بود:

1. زمان پاسخ (Response Time)
2. کیفیت بازیابی نتایج بر اساس معیار NDCG@10

برای این منظور، ایندکس‌ها با درصدهای مختلفی از کل داده‌ها ساخته شدند (از 1٪ تا 100٪) و عملکرد سیستم در هر مرحله اندازه‌گیری شد.

محیط اجرا و تنظیمات:

- موتور جستجو: Elasticsearch نسخه 9.3.0
- حالت امنیتی: Security فعال (Basic Authentication)
- دیتاست: Hamshahri (زبان فارسی)
- معیار ارزیابی: NDCG@10 (بر اساس BEIR Evaluator)
- درصدهای داده:

$$1\%, 2\%, 4\%, 8\%, 16\%, 32\%, 64\%, 100\%$$

در هر مرحله:

- ایندکس جدید ساخته شد
- اسناد همان درصد ایندکس شدند
- جستجو برای تمام Queryها انجام شد
- زمان اجرا و متریک کیفیت ذخیره شد

معماری و تعامل ماژول‌ها:

1. بارگذاری داده‌ها (data_loader.py) اسناد (df_docs)، کوئری‌ها (df_queries) و qrels آماده می‌شوند.
2. نمونه‌گیری درصدی (part2_index_size.py)برای هر درصد (مثلاً 8%) یک زیرمجموعه از اسناد انتخاب می‌شود.
3. ساخت ایندکس ES (es_indexer.py)
   - ایجاد index با mapping/settings
4. ایندکس‌کردن اسناد (es_indexer.py)
   - هر سند یک doc_id می‌گیرد
   - متن/فیلدها به ES ارسال می‌شود
5. جستجو و ارزیابی (es_search_eval.py)
   - برای هر query جستجو در ES انجام می‌شود
   - نتایج به فرمت مورد نیاز evaluator تبدیل می‌شود
   - متریک کیفیت استخراج می‌شود (NDCG@10)
6. ثبت زمان و متریک + رسم نمودار (part2_index_size.py)
   - زمان هر دور ذخیره می‌شود
   - NDCG هر دور ذخیره می‌شود
   - در پایان دو نمودار ساخته می‌شود (savefig)

صحت اجرا:

اجرای برنامه برای تمام درصدها بدون خطا انجام شد و خروجی کنسول نشان داد که:

- برنامه تا 100٪ داده اجرا شده است
- در هر مرحله متریک NDCG@10 با موفقیت محاسبه شده است:

```
=== 1% of data ===
100%|██████████| 3186/3186 [00:14<00:00, 224.93it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 2% of data ===
100%|██████████| 6372/6372 [00:28<00:00, 221.34it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 4% of data ===
100%|██████████| 12744/12744 [01:13<00:00, 173.32it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 8% of data ===
100%|██████████| 25489/25489 [05:21<00:00, 79.34it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 16% of data ===
100%|██████████| 50979/50979 [10:54<00:00, 77.83it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 32% of data ===
100%|██████████| 101959/101959 [21:49<00:00, 77.83it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 64% of data ===
100%|██████████| 203919/203919 [43:28<00:00, 78.18it/s]
Available metric keys: dict_keys(['NDCG@10'])

=== 100% of data ===
100%|██████████| 318624/318624 [1:08:09<00:00, 77.92it/s]
Available metric keys: dict_keys(['NDCG@10'])

Process finished with exit code 0
```

تطبیق نمودار «Response Time» با خروجی کنسول:

در خروجی برای هر درصد داده، زمان کل ایندکس + جستجو مشخص است:

زمان تقریبی اجرا	درصد داده
~14 ثانیه	**1%**
~28 ثانیه	**2%**
~1:13 دقیقه	**4%**
~5:21 دقیقه	**8%**
~10:54 دقیقه	**16%**
~21:49 دقیقه	**32%**
~43:28 دقیقه	**64%**
~1:08:09	**100%**

در کد:

- این زمان‌ها تقسیم بر تعداد کوئری‌ها (یا نرمال‌سازی شده) و به صورت میانگین زمان پاسخ ذخیره شده‌اند.
- بنابراین طبیعی است که در نمودار:
  - مقادیر در بازه‌ی حدود 0.08 تا 0.13 ثانیه باشند
  - کاملاً خطی هم نباشند

نمودار Response Time vs Index Size قرار نیست زمان کل اجرا را نشان بدهد، بلکه میانگین زمان پاسخ جستجو را نشان می‌دهد.

تحلیل نتایج:

تحلیل نمودار Response Time vs Index Size:

مشاهده از نمودار

- زمان پاسخ میانگین در بازه‌ی تقریبی 0.08 تا 0.13 ثانیه قرار دارد.
- با افزایش اندازه ایندکس، زمان پاسخ روند صعودی یکنواخت ندارد و دچار نوسان ملایم است.

تفسیر

این رفتار به دلایل زیر منطقی است:

1. هزینه ثابت سیستم (Overhead):بخشی از زمان صرف ارتباط با ES و پردازش نتایج می‌شود و مستقل از اندازه ایندکس است.
2. Cache و Warm-up در Elasticsearch:با افزایش حجم داده، برخی عملیات به کمک کش سریع‌تر انجام می‌شوند.
3. Top-k محدود (k=10):جستجو فقط تعداد کمی نتیجه برمی‌گرداند، بنابراین افزایش اسناد لزوماً زمان را به شدت افزایش نمی‌دهد.
4. نویز اندازه‌گیری و عملیات پس‌زمینه ES (refresh / segment merge)

نتیجه

افزایش اندازه ایندکس از 1٪ تا 100٪، باعث افزایش شدید و خطی زمان پاسخ نشده و سیستم از نظر زمانی پایدار باقی مانده است.

تحلیل نمودار NDCG@10 vs Index Size:

مشاهده از نمودار

- مقدار NDCG@10 برای تمام درصدها نزدیک به صفر باقی مانده است.
- افزایش اندازه ایندکس تأثیر محسوسی بر کیفیت نداشته است.

تفسیر

این نتیجه نشان می‌دهد که:

- صرفاً افزایش تعداد اسناد، بدون بهبود تنظیمات جستجو، کیفیت را افزایش نمی‌دهد.

دلایل محتمل:

1. ES با Analyzer پیش‌فرض (Standard Analyzer) کار می‌کند که این Analyzer برای زبان فارسی طراحی نشده
2. نبود نرمال‌سازی (ی/ک، نیم‌فاصله، علائم) در ES
3. استفاده از جستجوی پایه (match) بدون تنظیم وزن‌دهی یا مدل پیشرفته‌تر در ES
4. پراکندگی اسناد مرتبط و محدود بودن top-10

نتیجه

کیفیت بازیابی بیشتر از آنکه به اندازه ایندکس وابسته باشد، به نحوه پردازش متن و تنظیمات موتور جستجو وابسته است.