گزارش پروژه دوره ی RAG

محمد مهدی قنبری

مقدمه

این پروژه با هدف ساخت یک سیستم RAG برای پاسخ‌دهی به سوالات کاربران بر اساس مجموعه‌ای از اخبار فارسی از پنج خبرگزاری ایرانی (مانند ISNA، MehrNews، FarsNews و غیره) طراحی شده است. سیستم RAG ترکیبی از retrieval اسناد مرتبط و generation پاسخ با استفاده از مدل‌های زبانی بزرگ است تا پاسخ‌های دقیق و مبتنی بر زمینه ارائه دهد.

پروژه در دو نوت‌بوک Jupyter پیاده‌سازی شده است:
- rag-project-notebook-1.ipynb: تمرکز بر پیش‌پردازش داده‌ها و آماده‌سازی زیرمجموعه‌ای از اخبار.
- rag-project-notebook-2.ipynb: ساخت سیستم RAG، شامل chunking، embedding، retrieval و generation پاسخ، همراه با ارزیابی.

داده‌ها از یک فایل CSV حاوی بیش از ۳۹۰ هزار خبر استخراج شده‌اند. هدف نهایی، مقایسه روش‌های مختلف بازیابی از جمله lexical (با BM25)، امبدینگ semantic و روش ترکیبی(hybrid)، و ارزیابی عملکرد آن‌ها بر روی ۱۵ سوال نمونه است.

پیش‌پردازش داده‌ها (rag-project-notebook-1.ipynb)

در این نوت‌بوک، داده‌های خام پردازش می‌شوند تا برای استفاده در سیستم RAG آماده گردند. مراحل اصلی عبارتند از:

۱. خواندن داده‌ها
- داده‌ها از فایل CSV با مسیر `kaggle/input/persian-news-dataset/archive_v5.csv/` خوانده می‌شوند.
- dataframe با نام `df_raw` ایجاد می‌شود که شامل ستون‌هایی مانند `id`، `title`، `short_link`، `published_datetime`، `tags`، `body`، `abstract`، `subgroup`، ``service` و `agency_name` است.

۲. بررسی و حذف مقادیر نال، خالی و کوتاه
- تعداد مقادیر نال در هر ستون محاسبه می‌شود (مثلاً ۱۹۹۸۰ مورد در `body`).
- ردیف‌هایی که `body` نال دارند، حذف می‌شوند.
- خبر هایی که متن خالی(شامل تعدادی کاراکتر فاصله) دارند حذف می‌شوند.
- خبر هایی که طول کمتر از ۴۰۰ کاراکتر دارند را نیز حذف کردیم تا chunk ها خیلی کوتاه نباشند.

۳. انتخاب زیرمجموعه
- از dataframe کامل، ۲۰۰۰۰ خبر به صورت تصادفی انتخاب می‌شود تا حجم داده‌ها برای پردازش مناسب باشد.
- dataframe جدید (`df_subset`) ایجاد و ایندکس آن ریست می‌شود.

۴. نرمال‌سازی متن
- تابع `text_normalizer` برای نرمال‌سازی متن `body` استفاده می‌شود. این تابع شامل:
- تبدیل کاراکترهای عربی به معادل فارسی (مانند "ي" به "ی").
- حذف دیاکریتیک‌های عربی.
- تبدیل ارقام انگلیسی به فارسی.
- حذف کاراکترهای zero-width (مانند نیم-فاصله).
- نرمال‌سازی فاصله‌ها با regex.
- این مرحله اطمینان می‌دهد که متن‌ها برای embedding و retrieval یکنواخت باشند.

۵. ذخیره داده‌ها
- dataframe نهایی در فایل `news_subset.csv` ذخیره می‌شود.

این پیش‌پردازش، داده‌ها را تمیز و آماده برای مراحل بعدی می‌کند و از مشکلات زبانی فارسی (مانند تفاوت‌های عربی-فارسی) جلوگیری می‌نماید.

ساخت سیستم RAG و ارزیابی(rag-project-notebook-1.ipynb)

این نوت‌بوک سیستم اصلی RAG را پیاده‌سازی می‌کند. مراحل کلیدی عبارتند از:

۱. chunking اسناد
- تابع `chunking` برای تقسیم اخبار به chunkهای کوچک‌تر (حداکثر ۸۰۰ کاراکتر) استفاده می‌شود.

- اگر متن کوتاه‌تر از ۸۰۰ کاراکتر باشد، یک chunk ایجاد می‌شود؛ در غیر این صورت، به chunk های ۸۰۰ کاراکتری با overlap به اندازه ی ۱۵۰ کاراکتر تقسیم می‌گردد.
- dataframe جدید (`chunks_df`) شامل ستون‌های `chunk_id`، `doc_id`، `text`، `category` و `date` است (تعداد ۶۹۸۶۲ chunk تولید می‌شود).

۲. Embedding

-Lexical Embedding با BM25: متن‌ها توکنیزه می‌شوند (`tokenize` با regex) و مدل BM25Okapi ساخته می‌شود.
-Semantic Embedding: از مدل paraphrase-multilingual-MiniLM-L12-v2 کتابخانه ی sentence-transformers استفاده کردیم.
-یک دیتابیس FAISS با اندیس IndexFlatIP ساختیم و امبدینگ های semantic را به آن اضافه کردیم.

۳. Retrieval

- تابع `retrieve` سه روش را پشتیبانی می‌کند:
- lexical: با استفاده از BM-25 تعداد top-k (به صورت پیشفرض ۵) chunk را بر اساس امتیاز برمی‌گرداند.
- semantic: امبدینگ سوال را محاسبه و با FAISS جستجو می‌کند.
- hybrid: ابتدا ۲۰ chunk با BM25 انتخاب، سپس با semantic rerank می‌شود تا top-k نهایی انتخاب گردد.
- خروجی شامل chunkهای مرتبط و doc_idها است.

۴. Generation پاسخ با LLM

- از API Groq با مدل `llama-3.1-8b-instant` استفاده می‌شود.
- System prompt: دستیار مفید که پاسخ را بر اساس context فارسی بدهد، اگر مرتبط نبود بگوید اطلاعات کافی ندارد، و پاسخ را مختصر نگه دارد.
- تابع `answer`: بازیابی را انجام می‌دهد، context می‌سازد، و پاسخ را تولید می‌کند.

۵. مثال استفاده

- برای سوال "سرمربی تیم بارسلونا کیه؟"، روش hybrid استفاده شده و پاسخ "رونالد کومان" تولید می‌شود، همراه با doc_idهای مرتبط.

ارزیابی و نتایج

ارزیابی بر روی ۱۵ سوال نمونه از فایل `questions_template.jsonl` انجام شده است. برای هر سوال، سه روش retrieval تست می‌شود و بررسی می‌گردد آیا reference واقعی (doc_id درست) retrieve شده است یا خیر. نتایج در فایل `test_results.json` ذخیره شده‌اند.

خلاصه نتایج

- lexical: در ۱۵ مورد از ۱۵ سوال، reference واقعی retrieve شده (امتیاز ۱۵).
- semantic: در ۱۰ مورد موفق (امتیاز ۱۰).
- hybrid: در ۱۳ مورد موفق (امتیاز ۱۳).

تحلیل

- روش lexical در retrieval دقیق‌تر عمل کرده، احتمالاً به دلیل تمرکز بر کلمات کلیدی فارسی.
- روش semantic ممکن است به دلیل پیچیدگی‌های زبانی فارسی (مانند هم‌معنی‌ها) ضعیف‌تر باشد.
- hybrid تعادلی بین دو روش ایجاد کرده و عملکرد بهتری نسبت به semantic خالص دارد.
- در برخی سوالات (مانند q13 و q14)، همه روش‌ها موفق بوده‌اند، اما در مواردی مانند q15 (باشگاه فرانسوی برای سردار آزمون)، semantic و hybrid شکست خورده‌اند.

نمونه پاسخ‌ها نشان می‌دهد که LLM پاسخ‌های فارسی مختصری تولید می‌کند، اما گاهی اطلاعات اضافی یا اشتباه اضافه می‌نماید اگر context ناکافی باشد.
همچنین مشاهده شد که در مواردی که چندین خبر متفاوت درباره ی سوال کاربر وجود دارد،مدل زبانی در تولید پاسخ می‌تواند دچار اشتباه شود. به عنوان مثال،در پرسشی که درباره ی میزان کاهش شاخص سهام بود، چندین خبر متفاوت با اعداد متفاوت وجود داشت که باعث شد عددی که پاسخ داد اشتباه باشد. البته به دیگر اعداد نیز اشاره کرد.
یا در مثال دیگری، وقتی درباره ی تعداد برد های یک تیم بسکتبال سوال پرسیده شد، باز هم چندین خبر مختلف وجود داشت که عدد های متفاوتی را بیان می‌کردند و این در پاسخ LLM مشهود بود.

نتیجه‌گیری و پیشنهادات

این پروژه یک سیستم RAG کارآمد برای اخبار فارسی ساخته که از پیش‌پردازش تا ارزیابی کامل است. روش hybrid بهترین تعادل را ارائه می‌دهد، اما lexical در این داده‌ها برتر است.

نقاط قوت:
پشتیبانی از زبان فارسی، ترکیب روش‌های retrieval، استفاده از مدل‌های پیشرفته مانند SentenceTransformer و Llama.

نقاط ضعف:
وابستگی به حجم chunk ها، ممکن است در سوالات پیچیده‌تر شکست بخورد، و ارزیابی محدود به ۱۵ سوال. همچنین خبر ها به روز نیستند.

پیشنهادات آینده:
- افزایش تعداد سوالات ارزیابی و استفاده از معیارهایی مانند BLEU یا ROUGE برای کیفیت پاسخ.
- آزمایش مدل‌های embedding فارسی‌محور بیشتر (مانند ParsBERT).
- افزودن reranking پیشرفته‌تر یا fine-tuning LLM بر روی داده‌های فارسی.
- مقیاس‌پذیری با داده‌های بیشتر و بهینه‌سازی سرعت retrieval.