باسمه تعالی

درس: بازیابی اطلاعات

نام و نام خانوادگی: تینا توکلی و هادی امینی

شماره دانشجویی: 9922762220و 9912762270

شماره تمرین :02

---

1. توضیحات:

مدل Embedding:

- مدل Embedding روشی برای نمایش کلمات، جملات یا اسناد به صورت بردارهای عددی است.
- این بردارها به مدل های یادگیری ماشین کمک می کنند تا روابط معنایی بین کلمات و جملات را درک کنند.
- مدل های زیر را استفاده کرده :
- `voyage-lite-02-instruct` و `gte-large-en-v1.5` و `NV-Embed-v1`
- و در نهایت مدل دوم انتخاب شده است زیرا در کنار سبک بودن و عدم پیچیدگی دارای دقت ، سرعت و انعطاف پذیری مناسبی است که نیاز ما را برآورده میکند.

:LLM

- LLM مخفف "Large Language Model" به معنای "مدل زبانی بزرگ" است.
- LLM ها مدل های یادگیری ماشینی هستند که بر روی حجم عظیمی از داده های متنی آموزش دیده اند.
- آنها می توانند وظایف مختلفی مانند تولید متن، ترجمه زبان، و پاسخ به سوالات را انجام دهند.
- در ابتدا مدل های زیر را استفاده کردیم :
  - MaziyarPanahi/Calme-7B-Instruct-v0.9
  - BarraHome/Mistroll-7B-v2.2
  - pszemraj/Mistral-v0.3-6B
- دو مدل اول قابلیت اجرا در کولب را نداشته اند زیرا بیار سنگین بوده اند (به دلیل محدودیت سخت افزاری)و مدل سوم نیز به علت دقت کمی که داشته رد شده است
- پس از بررسی ریپازیتوری زیر:
  - https://github.com/Troyanovsky/Local-LLM-Comparison-Colab-UI

- مدل TheBloke/Mistral-7B-OpenOrca-GGUF انتخاب شد به علت اینکه امتیاز بالاتری نسبت به بقیه داشته است ،این مدل شامل gguf های مختلفی است که از بین آنها mistral-7b-openorca.Q4_K_M.gguf انتخاب شد به دلیل کیفیت مناسب و کم بودن مصرف رم و همچنین اندازه متناسب که مشخصات آن در زیر آمده است.

Name	Quant method	Bits	Size	Max RAM required	Use case
mistral-7b-openorca.Q4_K_M.gguf	Q4_K_M	4	4.37 GB	6.87 GB	medium, balanced quality - recommended

روش تقسیم سند به قطعه های کوچک:

- در RAG، اسناد به قطعه های کوچکتر به نام "سگمنت" تقسیم می شوند.
- این کار به مدل کمک می کند تا تمرکز خود را بر روی بخش های مرتبط یک سند متمرکز کند.
- موارد پایین استفاده شده:
- AI21 Semantic Text Splitter وToken و Semantic Chunker
- دلیل رد AI21 Semantic Text Splitter :زیرا در صورت پروژه ذکر شده که استفاده از سرویس های نیازمند API Key مجاز نمیباشد.
- بنابراین در ابتدا از Semantic Chunker استفاده کرده ایم به طوریکه متن ابتدا به جملات جداگانه تقسیم می‌شود .سپس، جملات کنار هم بررسی می‌شوند تا ببینند از نظر معنایی به اندازه‌ی کافی به هم مرتبط هستند یا خیر .در صورت مرتبط بودن، جملات با هم ترکیب شده و یک پاراگراف بزرگ‌تر را تشکیل می‌دهند.
- و سپس جملات را بررسی کرده وچنانچه طول آن بزرگتر از 512 باشد ،با استفاده از NLTKTextSplitter آن را به بخش های کوچکتر تقسیم میکنیم.

:Prompt

- Prompt یک جمله یا عبارت کوتاه است که به LLM می گوید چه کاری انجام دهد.
- در RAG، Prompt برای پرسیدن سوال از LLM در مورد یک سند خاص استفاده می شود.
- در این prompt متن به همراه سوال داده شده و از مدل خواسته میشود که چنانچه سوال به متن ارتباط دارد ،به آن پاسخ دهد

```
prompt = PromptTemplate.from_template(
"""
Answer based on provided text only. Errors for out-of-context questions.
```

```
      According to this document:
       {context}
      Please answer the following question: {question}

      Answer:"""
)
```

در ابتدا محتوا داده شده و سپس سوال مربوطه پرسیده میشود

Retriever:

- Retriever یک مدل یادگیری ماشینی است که برای یافتن سگمنت های مرتبط با یک Prompt خاص استفاده می شود.
- Retriever از مدل Embedding برای درک معنای Prompt و سگمنت ها استفاده می کند.
  - از Chroma و Chromadb استفاده میکنیم به طوریکه در ابتدا به Chromadb اضافه کرده و سپس از Chroma به عنوان retriever استفاده کرده ایم.
  - در گام پنجم برای بهبودretriever از MultiVectorRetriever استفاده کرده ایم به دلیل:
  - این روش باعث ایجاد چندین بردار برای هر سند می شود.
  - بسیاری از مواقع، بازیابی قطعات بزرگتر از اطلاعات مفید است، اما قطعات کوچکتر را embed می‌کنیم .این کار به embedها اجازه می‌دهد تا معنای دقیق کلمات را تا حد امکان دقیق بگیرند، در عین حال تا حد ممکن context را به مراحل بعدی منتقل کنند .

بررسی نتایج:

- از یک llm به عنوان داور استفاده کرده تا امتیاز دو پاسخ عادی و بهبود یافته را محاسبه کند .

مشارکت:

امینی	10
توکلی	10