به نام خداوند جان و خرد

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

پردازش زبان طبیعی

تمرین شماره ۶

نام و نام خانوادگی: علی خرم فر

شماره دانشجویی: ۸۱۰۱۰۲۱۲۹

تیرماه ۱۴۰۳

فهرست مطالب

۱_ دریافت و آماده سازی دادگان ۱

۱-۱_ استخراج لینک‌های فصل‌ها ۱

۱-۲_ تبدیل دادگان به فرمت مناسب ۱

۱-۳_ تقسیم اسناد به بخش‌های مناسب ۲

۱-۴_ عملکرد RecursiveCharacterTextSplitter و ضرورت آن ۲

مزایای تکه‌تکه‌کردن متن ۳

۱-۵_ اهمیت مقادیر مناسب برای chunk_size و chunk_overlap ۳

chunk_size خیلی بزرگ ۳

chunk_size خیلی کوچک ۳

chunk_overlap خیلی بزرگ ۳

chunk_overlap خیلی کوچک ۴

مشکلات ناشی از انتخاب مقادیر نامناسب ۴

۲_ تولید بازنمایی و پایگاه داده بُرداری ۴

۲-۱_ ذخیره‌سازی داده‌ها با استفاده از Embedder و FAISS ۴

۲-۲_ اهمیت استفاده از Embedder مناسب ۵

مشکلات مدل‌های نامناسب ۵

۳_ پیاده‌سازی بازیاب ترکیبی ۶

۳-۱_ تفاوت بین بازیاب‌های Lexical و Semantic ۶

بازیاب‌های Lexical ۶

بازیاب‌های Semantic ۶

۳-۲_ پیاده‌سازی بازیاب ترکیبی ۷

۳-۳_ مقادیر ضریب در بازیاب ترکیبی ۷

۳-۴_ ارزیابی عملکرد بازیاب ترکیبی ۸

پرس‌وجوها ۸

۴_ پیاده‌سازی Router chain ۹

۴-۱_ دریافت API Key ۹

۴-۲_ ایجاد زنجیر ۹

تعریف مدل با ChatTogether ۹

تعریف پرامپت ورودی: ۹
تعریف Parser: ۹
۳-۴_ دلیل استفاده از Temperature برابر با صفر ۱۰
مفهوم Temperature ۱۰
کاهش عدم قطعیت: ۱۰
افزایش دقت: ۱۱
۵_ پیاده‌سازی Search Engine Chain ۱۱
۵-۱_ دریافت API Key ۱۱
۵-۲_ تعریف ابزار جستجو ۱۱
۵-۳_ پیاده‌سازی زنجیر ۱۲
تبدیل داده‌های خروجی به Documentهای LangChain ۱۲
۶_ پیاده‌سازی Relevancy Check Chain ۱۳
۶-۱_ ایجاد زنجیر ۱۳
طراحی پرامپت ۱۳
تعریف مدل زبانی ۱۴
تعریف پس پردازشگر ۱۴
ترکیب زنجیر ۱۴
۶-۲_ نیاز به زنجیر Relevancy Check ۱۵
۷_ پیاده‌سازی Fallback Chain ۱۶
۷-۱_ ایجاد زنجیر ۱۶
طراحی پرامپت ۱۶
تعریف مدل زبانی ۱۶
تعریف پس‌پردازشگر ۱۶
ترکیب زنجیر ۱۷
اهمیت وجود این زنجیر ۱۷
۸_ پیاده سازی GENERATE WITH CONTEXT CHAIN ۱۸
طراحی پرامپت ۱۸
تعریف پس پردازشگر ۱۸
ترکیب زنجیر ۱۹
۹_ آماده‌سازی گراف با استفاده از LANGGRAPH ۱۹
۹-۱_ پیاده‌سازی گراف با LangGraph ۱۹

تعریف AgentState ۱۹
گره VectorStore ۲۰
گره FilterDocs ۲۰
گره Fallback ۲۰
گره Generate With Context ۲۰
۹-۲_ ساخت گراف ۲۱
۹-۳_ تست نهایی ۲۲
۹-۴_ اجرای چت‌بات با استفاده از Gradio ۲۲

۱_ دریافت و آماده سازی دادگان

برای انجام این تمرین از محیط Kaggle استفاده شد. در ادامه به بررسی مراحل انجام‌شده خواهیم پرداخت. ابتدا کتابخانه‌های مورد نیاز را نصب و Import کرده و همچنین کلیدهای Api مربوط به سرویس‌های Tavily و Together را تنظیم می‌کنیم. همچنین تنظیمات مربوط به عدم نمایش برخی هشدارها را غیرفعال می‌کنیم.

۱-۱_ استخراج لینک‌های فصل‌ها

ابتدا با استفاده از کتابخانه requests، محتوای صفحه اصلی مربوط به کتاب مرجع را دریافت کردیم. این صفحه شامل لینک‌های مربوط به فصل‌های مختلف کتاب است که به صورت فایل‌های PDF قرار داده شده‌اند. سپس با استفاده از BeautifulSoup، محتوای HTML صفحه بازیابی شد تا لینک‌های مربوط به فصل‌ها را استخراج کنیم. برای هر فصل، شماره و نام فصل به همراه لینک PDF آن را استخراج و در یک لیست ذخیره شد.در نهایت فقط لینک‌هایی که با فایل‌های PDF تمام می‌شدند، انتخاب و ذخیره شدند. شکل زیر کد مربوط به این مرحله است:

```python
url = "https://stanford.edu/~jurafsky/slp3/"
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
chapter_links = []

for tr in soup.find_all('tr'):
    tds = tr.find_all('td')
    if len(tds) > 1:
        chapter_number = tds[0].get_text().strip()
        if chapter_number and (chapter_number[0].isdigit() or chapter_number[0].isalpha()) and ':' in chapter_number:
            a_tag = tds[1].find('a', href=True)
            if a_tag and a_tag['href'].endswith('.pdf'):
                chapter_name = a_tag.get_text().strip()
                chapter_link = url + a_tag['href']
                chapter_links.append((chapter_name, chapter_link))

for name, link in chapter_links:
    print(f'{name}:\n {link}')
```

```
Regular Expressions, Text Normalization, Edit Distance:
 https://stanford.edu/~jurafsky/slp3/2.pdf
N-gram Language Models:
 https://stanford.edu/~jurafsky/slp3/3.pdf
Naive Bayes, Text Classification,  and Sentiment:
 https://stanford.edu/~jurafsky/slp3/4.pdf
Logistic Regression:
 https://stanford.edu/~jurafsky/slp3/5.pdf
Vector Semantics and Embeddings:
```

شکل ۱ استخراج لینک‌های فصل‌ها به صورت کد محور

۱-۲_ تبدیل دادگان به فرمت مناسب

به دلیل مشکلاتی که کتابخانه معرفی شده در تمرین داشت، این کار با استفاده از ابزار PyPDFLoader موجود در کتابخانه LangChain انجام می‌شود. با کمک این ابزار محتویات هر PDF به صورت سندهای جداگانه بارگذاری شده و در لیست documents ذخیره می‌کنیم.

ImportError: cannot import name 'PdfLoader' from 'langchain.document_loaders' (/opt/conda/lib/python3.10/site-packages/langchain/document_loaders/__init__.py)

۱-۳_ تقسیم اسناد به بخش‌های مناسب

در این بخش، هدف ما تقسیم اسناد به اندازه‌های مناسب با استفاده از ابزار RecursiveCharacterTextSplitter است. این کار به ما کمک می‌کند تا متن‌های طولانی را به بخش‌های کوچکتر و قابل پردازش تبدیل کنیم. خروجی این تابع به صورت کلاس Documents سازگار با Longchain خواهدبودکه یک لیست از بخش‌های کوچکتر یا chunks است که هر کدام حداکثر ۱۰۲۴ کاراکتر دارند و به اندازه ۶۴ کاراکتر همپوشانی دارند. در شکل زیر برخی خروجی‌ها را مشاهده می‌کنیم:

```
Chunk 1:
Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright ©2023. All
rights reserved. Draft of February 3, 2024.
CHAPTER
2Regular Expressions, Text
Normalization, Edit Distance
User: I am unhappy.
ELIZA: DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
User: I need some help, that much seems certain.
ELIZA: WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
User: Perhaps I could learn to get along with my mother.
ELIZA: TELL ME MORE ABOUT YOUR FAMILY
User: My mother takes
...
Chunk 2:
them into suitable outputs like “What would it mean to you if you got X?”. This
simple technique succeeds in this domain because ELIZA doesn’t actually need to
know anything to mimic a Rogerian psychotherapist. As Weizenbaum notes, this is
one of the few dialogue genres where listeners can act as if they know nothing of the
world. ELIZA’s mimicry of human conversation was remarkably successful: many
people who interacted with ELIZA came to believe that it really understood them
and their problem
...
Chunk 3:
and other chatbots play a crucial role in natural language processing.
We’ll begin with the most important tool for describing text patterns: the regular
expression . Regular expressions can be used to specify strings we might want to
extract from a document, from transforming “I need X” in ELIZA above, to defining
strings like $199 or$24.99 for extracting tables of prices from a document.
We’ll then turn to a set of tasks collectively called text normalization , in whichtext
normalization
regula
```

شکل ۲ تقسیم اسناد به بخش‌های مناسب

۱-۴_ عملکرد RecursiveCharacterTextSplitter و ضرورت آن

در این مرحله، از کلاس RecursiveCharacterTextSplitter برای تکه تکه کردن متون طولانی به بخش‌های کوچکتر استفاده شد. این ابزار با تنظیم اندازه‌ی هر بخش (chunk_size) به ۱۰۲۴ کاراکتر و همپوشانی بین بخش‌ها (chunk_overlap) به ۶۴ کاراکتر، متن‌ها را به صورت بازگشتی تقسیم می‌کند. ابتدا تلاش می‌کند تا متن را با استفاده از جداکننده‌های مشخص شده (مانند نقاط و پاراگراف‌ها) به بخش‌های کوچکتر تقسیم کند. اگر جداکننده‌ی مناسبی پیدا نشود، به صورت بازگشتی به تقسیم متن ادامه می‌دهد تا زمانی که بخش‌ها به اندازه‌ی مناسب برسند.

مزایای تکه‌تکه‌کردن متن

بهبود کارایی پردازش: مدل‌های پردازش زبان طبیعی معمولاً محدودیت‌هایی در طول متن ورودی دارند. تقسیم متن به بخش‌های کوچکتر این مشکل را حل می‌کند و باعث می‌شود پردازش متون سریع‌تر و کارآمدتر شود.

حفظ ارتباط بین بخش‌ها: همپوشانی بین بخش‌ها تضمین می‌کند که اطلاعات مهم و پیوستگی متن حفظ شود. این امر به ویژه در کاربردهایی مانند خلاصه‌سازی و پاسخ‌دهی به سوالات اهمیت دارد.

افزایش دقت: مدل‌ها می‌توانند با دقت بیشتری متن‌های کوتاه‌تر را پردازش کنند و نتایج بهتری ارائه دهند. بخش‌های کوچکتر و متمرکزتر به مدل کمک می‌کنند تا بهتر مفاهیم را درک کند و تحلیل‌های دقیق‌تری انجام دهد.

۵-۱_ اهمیت مقادیر مناسب برای chunk_size و chunk_overlap

chunk_size خیلی بزرگ

بخش‌های متنی طولانی‌تر از ظرفیت مدل‌های پردازش زبان طبیعی می‌شوند و باعث می‌شود مدل‌ها قادر به پردازش صحیح آن‌ها نباشند.این امر می‌تواند باعث کاهش دقت و کارایی مدل‌ها شود زیرا مدل‌ها با داده‌های بیش از حد طولانی کار می‌کنند.

chunk_size خیلی کوچک

متن به بخش‌های بسیار کوتاه تقسیم می‌شود که ممکن است اطلاعات مهم و پیوستگی متن از دست برود.مدل‌ها ممکن است نتوانند زمینه و مفهوم کلی متن را به درستی درک کنند که این می‌تواند به نتایج نامناسب منجر شود.

chunk_overlap خیلی بزرگ

همپوشانی زیاد باعث افزایش تعداد بخش‌ها می‌شود که این می‌تواند زمان و منابع پردازشی بیشتری مصرف کند.همچنین، این امر می‌تواند باعث شود که مدل‌ها اطلاعات تکراری را چندین بار پردازش کنند که باعث کاهش کارایی می‌شود.

chunk_overlap خیلی کوچک

بخش‌های متنی با هم ارتباط کمی خواهند داشت و ممکن است اطلاعات کلیدی که در مرزهای بخش‌ها قرار دارند، از دست بروند.این امر می‌تواند باعث شود مدل‌ها نتوانند پیوستگی و جریان منطقی متن را به درستی درک کنند که به کاهش دقت تحلیل‌ها منجر می‌شود.

مشکلات ناشی از انتخاب مقادیر نامناسب

کاهش دقت و کارایی: پردازش متون خیلی بزرگ یا خیلی کوچک به ترتیب می‌تواند منجر به کاهش دقت و کارایی مدل‌ها شود.

افزایش مصرف منابع: همپوشانی بیش از حد یا تعداد زیادی از بخش‌ها می‌تواند منابع پردازشی و زمانی بیشتری مصرف کند.

از دست رفتن اطلاعات: همپوشانی ناکافی می‌تواند باعث از دست رفتن اطلاعات کلیدی و کاهش کیفیت نتایج شود.

۲_ تولید بازنمایی و پایگاه داده بُرداری

۲-۱_ ذخیره‌سازی داده‌ها با استفاده از Embedder و FAISS

ابتدا یک دایرکتوری جدید برای ذخیره بازنمایی‌ها ایجاد شد تا در آینده بتوان از آن‌ها استفاده مجدد کرد. همچنین از مدل پیش‌فرض HuggingFaceEmbeddings برای ایجاد Embedder استفاده شد. این مدل پیش‌فرض sentence-transformers/all-mpnet-base-v2 است:

```
embedding_function.model_name
```

```
'sentence-transformers/all-mpnet-base-v2'
```

برای بهینه‌سازی و ذخیره‌سازی بازنمایی‌ها، از CacheBackedEmbeddings استفاده شد. این ابزار بازنمایی‌های محاسبه شده را در دایرکتوری ذخیره می‌کند تا در آینده به جای محاسبه مجدد، از این بازنمایی‌ها استفاده شود.سپس اسناد تکه‌تکه شده (chunks) با استفاده از Embedder تولید شده و در پایگاه داده برداری FAISS ذخیره شدند. همچنین تعداد اسناد را با تعداد بازنمایی‌های تولید شده مقایسه کردیم تا از صحت کار مطمئن شویم.

۲-۲_ اهمیت استفاده از Embedder مناسب

در این تمرین، هدف ما پیاده‌سازی یک چت بات پاسخگو در حوزه NLP است. استفاده از Embedder مناسب برای این هدف از اهمیت بالایی برخوردار است زیرا Embedding معنایی دقیق و معنادار از منابع کتاب، اساس عملکرد صحیح چت بات را تشکیل می‌دهد.

مشکلات مدل‌های نامناسب

Embeddingها وظیفه تبدیل جملات و متون به بردارهای عددی را دارند که مدل‌های مختلف مثلا تحلیل احساسات یا در اینجا بازیاب ترکیبی که پیاده‌سازی خواهیم کرد، از آن‌ها استفاده کند. اگر Embeddingهای تولید شده توسط مدلی باشد که به طور کلی با داده‌های فارسی آشنا نیست، مشکلات متعددی به وجود می‌آیند:

کاهش دقت و کارایی:

مدل‌هایی که داده‌های فارسی را ندیده‌اند، قادر به تولید بازنمایی‌های دقیق و معنادار از متون فارسی نخواهند بود. بازنمایی‌های نادرست می‌تواند در کاربردهای مختلف چت بات مانند جستجو در موتور جستجو، بازیابی اسناد و پاسخ‌دهی عمومی مشکلاتی ایجاد کند. به عنوان مثال، در جستجوی اسناد مرتبط، مدل ممکن است نتواند اسناد مرتبط با جستجوی کاربر را به درستی پیدا کند و پاسخ نامربوط ارائه دهد.

از دست رفتن مفاهیم زبانی:

هر زبان دارای ساختارهای نحوی و معنایی خاص خود است. مدل‌هایی که با داده‌های فارسی آموزش ندیده‌اند، نمی‌توانند این ویژگی‌ها را به درستی درک و بازنمایی کنند. این موضوع می‌تواند منجر به از دست رفتن اطلاعات مهم و عدم توانایی در تشخیص صحیح مفاهیم شود. به عنوان مثال، در جملات پیچیده یا استفاده از اصطلاحات و کلمات محاوره‌ای، مدل‌های نامناسب نمی‌توانند معنای دقیق را استخراج کنند.

استفاده از مدل‌های مناسب

مدل‌های مناسب مانند sentence-transformers/all-mpnet-base-v2 که در این تمرین استفاده کردیم، با داده‌های متنوع و گسترده آموزش دیده‌اند، می‌توانند بازنمایی‌های دقیقی از متون ایجاد کنند. این مدل‌ها با استفاده از معماری ترنسفورمرها و با کمک حجم زیادی از داده‌ها، توانایی درک و بازنمایی دقیق مفاهیم زبانی را دارند. استفاده از این مدل‌ها به بهبود دقت و کارایی چت بات کمک می‌کند و نتایج بهتری در کاربردهای مختلف ارائه می‌دهد. هرچند که در این تمرین متن ما انگلیسی است.

۳_ پیاده‌سازی بازیاب ترکیبی

۳-۱_ تفاوت بین بازیاب‌های Lexical و Semantic

تفاوت‌های اصلی بین این دو روش بازیابی در نحوه پردازش و جستجوی اطلاعات است.

بازیاب‌های Lexical

بازیاب‌های Lexical، مانند BM25Retriever، بر اساس تطابق کلمات کلیدی در اسناد عمل می‌کنند. این نوع بازیاب‌ها از الگوریتم‌های جستجوی سنتی استفاده می‌کنند که به دنبال کلمات یا ترکیبات دقیقی از کلمات در متن‌ها هستند. برای مثال، اگر کاربر کلمه‌ای خاص را جستجو کند، این بازیاب به دنبال همان کلمه یا کلمات مشابه در اسناد موجود خواهد بود.

ویژگی‌های اصلی بازیاب‌های Lexical سادگی و سرعت در جستجو است. این روش‌ها برای جستجوهای سریع و ساده بسیار مناسب هستند و به دلیل استفاده از الگوریتم‌های ساده‌تر، زمان پردازش کمتری نیاز دارند. اما مشکل اصلی این بازیاب‌ها این است که ممکن است معنای کلی متن را درک نکنند و نتایج غیرمرتبط بازگردانند، به خصوص اگر کلمات کلیدی مورد نظر به صورت دقیق در اسناد موجود نباشند.

بازیاب‌های Lexical از بازنمایی‌های Sparse استفاده می‌کنند. در این روش، هر کلمه به یک بردار با طولی برابر با تعداد کلمات موجود در فرهنگ لغت تبدیل می‌شود. بردارهای کلمات اغلب شامل مقدار زیادی صفر هستند، چرا که هر بردار فقط در موقعیت‌هایی غیر صفر است که به کلمات خاصی اشاره دارد.

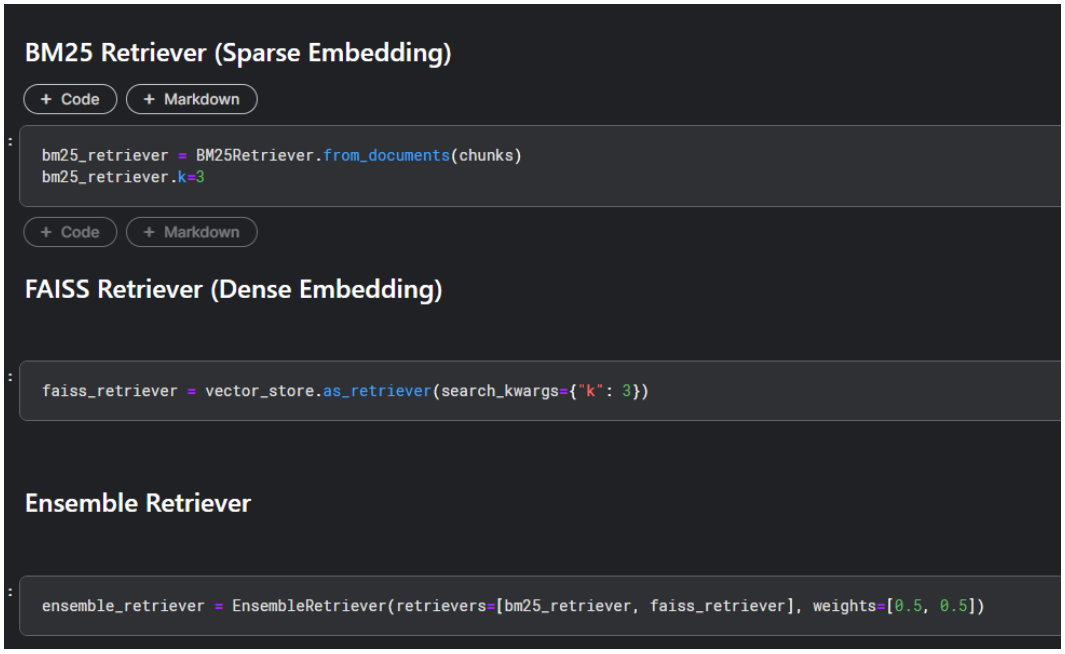
بازیاب‌های Semantic

در مقابل، بازیاب‌های Semantic، مانند FAISS، از مدل‌های پیچیده‌تری برای درک معنای کلی متن‌ها استفاده می‌کنند. این بازیاب‌ها از مدل‌های یادگیری عمیق و برداری برای تحلیل و جستجوی اسناد بهره می‌برند. به جای تطابق کلمات دقیق، بازیاب‌های Semantic تلاش می‌کنند تا معنای عمیق‌تر و روابط معنایی بین کلمات را درک کنند. این نوع بازیاب‌ها قادر به ارائه نتایج دقیق‌تر و مرتبط‌تر هستند، به خصوص در متون پیچیده و بزرگ که نیاز به درک معنای کلی دارند. اما به دلیل استفاده از مدل‌های پیچیده‌تر، این روش‌ها نیاز به منابع پردازشی بیشتری دارند و ممکن است زمان بیشتری برای پردازش و جستجو نیاز داشته باشند. بازیاب‌های Semantic از بازنمایی‌های Dense استفاده می‌کنند. در بازنمایی‌های Dense، بردارها با طول ثابت و مقادیر غیر صفر تولید می‌شوند که نمایانگر ویژگی‌ها و معنای کلمات یا جملات هستند.

۳-۲_ پیاده‌سازی بازیاب ترکیبی

ابتدا با استفاده از کلاس BM25Retriever، یک بازیاب Lexical ایجاد می‌کنیم که اسناد را براساس کلمات کلیدی جستجو می‌کند. سپس، از FAISS به عنوان بازیاب Semantic استفاده می‌کنیم که قادر به درک معنای عمیق‌تر و روابط معنایی بین کلمات است.

در نهایت، با استفاده از EnsembleRetriever، این دو بازیاب را ترکیب می‌کنیم و وزن هر کدام را به طور مساوی ۰٫۵ تنظیم می‌کنیم تا تأثیر هر دو بازیاب در نتایج نهایی یکسان باشد.



شکل ۳ پیاده‌سازی بازیاب ترکیبی

۳-۳_ مقادیر ضریب در بازیاب ترکیبی

برای انجام آزمایش، تابعی تعریف شده که وزن‌های مختلف برای هر بازیاب را امتحان کرده و نتایج را برای چندین پرسش مختلف بررسی می‌کند. پرسش‌های آزمایش ""Multihead self attention layers"، ""Maximum Spanning Tree" و "Who is the elected president of Iran?" هستند.

وقتی که FAISS با وزن بالاتری استفاده می‌شود نتایج به دست آمده بیشتر بر مبنای بازنمایی‌های معنایی هستند. برای پرسش‌های تخصصی، مانند "Multihead self attention layers"، نتایج دقیق‌تری بازگردانده شد. ولی وقتی BM25 با وزن بالاتری استفاده می‌شود و نتایج بیشتر بر مبنای کلمات کلیدی هستند. این حالت برای پرسش‌های عمومی‌تر مانند "Who is the elected president of Iran?" نتایج بهتری داشت.

آزمایش‌های مختلفی انجام شد که نتایج آن در نوت بوک است. مقادیر نهایی را به نحوی تنظیم کردیم که مقداری به وزن بازیاب معنایی اضافه شود:

ensemble_retriever.weights = [0.4, 0.6]

۴-۳_ ارزیابی عملکرد بازیاب ترکیبی

در این بخش، برای ارزیابی عملکرد بازیاب ترکیبی، سه پرس‌وجو مختلف به سیستم داده شد. این پرس‌وجوها شامل موضوعات متنوعی بودند تا میزان دقت و کارایی بازیاب ترکیبی در شرایط مختلف بررسی شود.

پرس‌وجوها

"Frame-Based Dialogue Systems": این پرسش مربوط به مباحث کتاب مرجع در زمینه سیستم‌های گفتگو مبتنی بر فریم بود.

"Dynamic Programming in Algorithm Design": این پرسش مربوط به علوم کامپیوتر و خارج از زمینه پردازش زبان‌های طبیعی بود.

"In what year did the Titanic sink?": این پرسش کاملاً خارج از حوزه عملکرد چت بات بود.

برای پرسش "Frame-Based Dialogue Systems"، نتایج بسیار دقیق و مرتبط با مباحث کتاب مرجع بازیابی شدند. این نشان می‌دهد که بازیاب ترکیبی به خوبی با محتوای کتاب مرجع هماهنگ است.

برای پرسش "Dynamic Programming in Algorithm Design"، نتایج بازگردانده شده به خوبی مباحث مرتبط با علوم کامپیوتر را پوشش دادند. این امر نشان‌دهنده توانایی سیستم در جستجوی مطالب تخصصی ولی خارج از زمینه پردازش زبان طبیعی است هرچند که بسیاری از این مطالب در کتاب مربوطه به دفعات تکرار شده‌اند.

برای پرسش "In what year did the Titanic sink?"، نتایج بازگردانده شده نامرتبط بودند.

همچنین ۳ پرسش متفاوت دیگر نیز بررسی شد که نتایج در نوت بوک است.

۴_ پیاده‌سازی Router chain

۴-۱_ دریافت API Key

این مرحله انجام شد و API در قسمت اول گزارش به نوت بوک اضافه شد.

۴-۲_ ایجاد زنجیر

تعریف مدل با ChatTogether

با استفاده از کلاس ChatTogether در LangChain، مدل meta-llama/Llama-3-70b-chat-hf را با تنظیم Temperature برابر صفر تعریف کردیم. این مدل برای پردازش و تولید پاسخ‌های دقیق و مرتبط به پرس‌وجوهای کاربران استفاده می‌شود.

Model from ChatTogether

+ Code + Markdown

```
sample_llm = ChatTogether(model_name="meta-llama/Llama-3-70b-chat-hf", temperature=0)
```

تعریف پرامپت ورودی:

پرامپت ورودی با استفاده از ChatPromptTemplate ایجاد شد. این پرامپت شامل دستورالعمل‌هایی است که به مدل زبانی کمک می‌کند تا بر اساس پرس‌وجوی کاربر، ابزار مناسب را انتخاب کند.

```
router_prompt_template = (
    "You are an expert in routing user queries to the appropriate resource for our Natural Language Processing (NLP) course.\n"
    "VectorStore contains detailed information and in-depth explanations about the NLP topics.\n"
    "Based on the user query, decide which resource to use:\n"
    "1. 'VectorStore' for specific queries related to NLP course topics.\n"
    "2. 'SearchEngine' for general information and broad queries about NLP or related fields.\n"
    "If the query is not related to NLP course content, select 'None'.\n"
    "Respond with only the name of the tool you chose and nothing more. If there is no suitable tool, respond with 'None'.\n"
    "{output_instructions}\n"
    "User Query: {query}"
)
sample_prompt = ChatPromptTemplate.from_template(template=router_prompt_template)
```

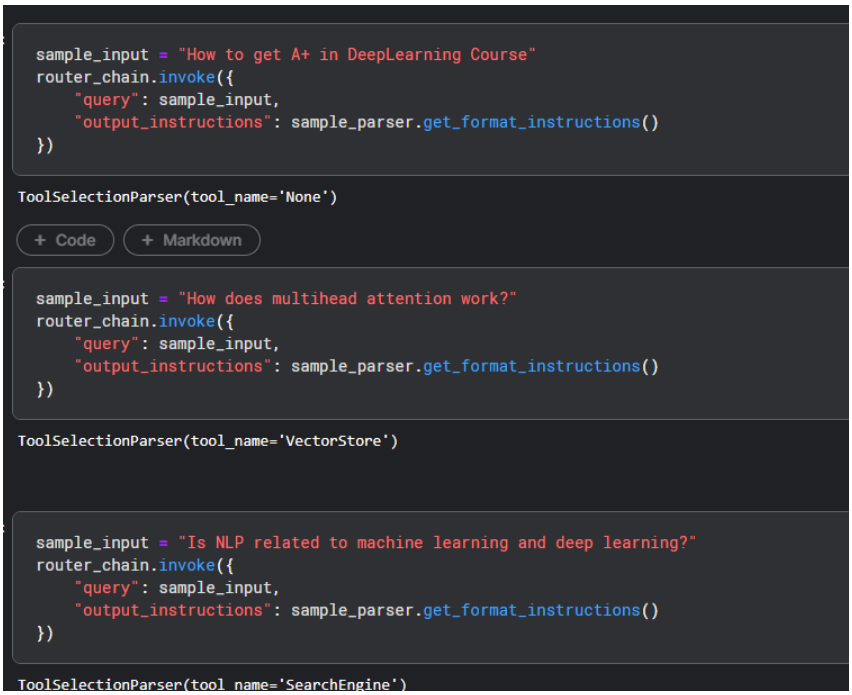
شکل ۴ پرامپت استفاده شده در Router Chain

تعریف Parser:

از PydanticOutputParser برای ارزیابی خروجی مدل زبانی استفاده شد. این Parser اطمینان حاصل می‌کند که خروجی مدل یکی از مقادیر VectorStore، SearchEngine یا None است. و در نهایت زنجیر پیاده‌سازی شد:

```
router_chain = sample_prompt | sample_llm | sample_parser
```

چندین پرس‌وجو با موضوعات مختلف به زنجیر داده شد تا عملکرد آن ارزیابی شود. نتایج نشان داد که زنجیر به خوبی توانسته پرس‌وجوهای مرتبط با پردازش زبان طبیعی را به VectorStore و پرس‌وجوهای عمومی‌تر را به SearchEngine هدایت کند. همچنین، پرس‌وجوهای نامرتبط با حوزه تخصصی چت بات به درستی به None هدایت شدند.



شکل ۵ بررسی عملکرد Router chain

۳-۴_ دلیل استفاده از Temperature برابر با صفر

مفهوم Temperature

Temperature یکی از پارامترهای مهم در مدل‌های زبانی است که بر تنوع و قطعیت خروجی‌های تولید شده توسط مدل تأثیر می‌گذارد. این پارامتر با کنترل میزان تصادفی بودن (randomness) در پیش‌بینی‌های مدل، می‌تواند نتایج مختلفی تولید کند. Temperature بالا منجر به خروجی‌های متنوع‌تر می‌شود، در حالی که Temperature پایین، خروجی‌های مشخص‌تر و قطعی‌تری را تولید می‌کند.

در این تمرین، هدف ما ایجاد یک Router Chain است که با دقت و قطعیت بالا پرس‌وجوهای کاربران را به ابزار مناسب هدایت کند. استفاده از Temperature برابر با صفر به دلایل زیر انتخاب شد:

کاهش عدم قطعیت:

با تنظیم Temperature به صفر، مدل به سمت پیش‌بینی‌های قطعی و قابل اعتماد حرکت می‌کند. این بدان معناست که مدل همواره از احتمال بالاترین پیش‌بینی استفاده می‌کند و از خروجی‌های تصادفی اجتناب می‌کند. برای یک چت بات که وظیفه آن هدایت دقیق پرس‌وجوها به منابع مناسب است، این ویژگی بسیار حیاتی است.

افزایش دقت:

در مواردی که پاسخ‌های تولید شده باید دقیق و مرتبط باشند، Temperature برابر با صفر می‌تواند بهترین گزینه باشد. این مقدار باعث می‌شود که مدل از بالاترین احتمال برای تولید پاسخ‌ها استفاده کند و از تولید پاسخ‌های غیرمرتبط جلوگیری کند. برای مثال، در تصمیم‌گیری بین VectorStore، SearchEngine و None، استفاده از Temperature پایین تضمین می‌کند که مدل به درستی و با قطعیت پاسخ مناسب را انتخاب کند. در صورتی که مدل با Temperature بالا تنظیم شود، ممکن است پاسخ‌های متنوع و گاهاً غیرمرتبطی تولید کند که می‌تواند کارایی چت بات را کاهش دهد.

۵_ پیاده‌سازی Search Engine Chain

۵-۱_ دریافت API Key

این مرحله انجام شد و API در قسمت اول گزارش به نوت بوک اضافه شد.

۵-۲_ تعریف ابزار جستجو

ابزار جستجو را به صورت زیر پیاده‌سازی کردیم:

```
tavily_tool = TavilySearchResults()
```

سپس ۲ جستجوی متفاوت را برای بررسی عملکرد آن انجام دادیم:

```
sample_input = "Who is the elected president of Iran?"
search_results = tavily_tool.invoke(sample_input)
print(search_results)
```

```
[{'url': 'https://www.cnn.com/2024/07/05/middleeast/pezeshkian-wins-irans-vote-intl-hnk/index.html', 'content': "Newly-elected Iranian President Masoud Pezeshkian is c
heered by supporters as he arrives at the shrine of the Islamic Republic's founder Ayatollah Ruhollah Khomeini in Tehran on July 6, 2024. Atta ..."}, {'url': 'https://
www.reuters.com/world/middle-east/irans-pezeshkian-brings-hopes-moderation-after-routing-hardline-rival-2024-07-06/', 'content': "Iran's president-elect, low-profile m
oderate Masoud Pezeshkian, carries the hopes of millions of Iranians seeking less restrictions on social freedoms and a more pragmatic foreign policy."}, {'url': 'http
s://www.bbc.com/news/articles/cx824y13ln4o', 'content': "Reformist Masoud Pezeshkian has been elected as Iran's new president, beating his hardline conservative rival
Saeed Jalili. The vote was declared in Dr Pezeshkian's favour after he secured 53.3% ..."}, {'url': 'https://www.politico.com/news/2024/07/06/heart-surgeon-who-rose-to
-power-in-parliament-is-now-irans-president-elect-00166692', 'content': "After Iran's June 28 presidential election saw the lowest turnout in history, Pezeshkian won 1
6.3 million votes against hard-liner Saeed Jalili 's 13.5 million votes to clinch Friday's ..."}, {'url': 'https://www.nytimes.com/2024/07/05/world/middleeast/iran-ele
ction-reformist-wins.html', 'content': 'Arash Khamooshi for The New York Times. In an election upset in Iran, the reformist candidate who advocated moderate policies a
t home and improved relations with the West won the presidential ...'}]
```

+ Code + Markdown

```
sample_input = "Is it harmful to stay up at night to do homework?"
tavily_tool.invoke(sample_input)
```

```
[{'url': 'https://www.wikihow.com/Stay-Up-All-Night-Doing-Homework',
  'content': "Staying up all night to do homework is not advised, but sometimes it's unavoidable. If your homework has piled up to the point that the only way to compl
ete it is to pull an all-nighter, then make some preparations and get your head in the game. You're in for a long night."},
 {'url': 'https://www.medicalnewstoday.com/articles/325335',
  'content': 'Sometimes it is necessary for people to stay up all night for work, studying, or other reasons. Tips that may help a person feel more awake and alert at
night can include consuming caffeine ...'},
 {'url': 'https://www.westlionsroar.com/features/2021/02/25/the-effects-homework-can-have-on-teens-sleeping-habits/',
  'content': 'Homework is pretty stressful for teens, especially if they have other things to do. Many teens have long hours at school, which limits the time for them
to do their insane amount of homework, attend extra-curricular activities, eat, do whatever they need to around the house, and sleep.'},
 {'url': 'https://www.sciencedaily.com/releases/2016/09/160919162837.htm',
  'content': 'Most people need at least seven to eight hours of sleep at night for the body and brain to function normally. So, if you stay up all night, missing out o
n the recommended amount of sleep, your ...'},
 {'url': 'https://drcraigcanapari.com/too-much-homework-too-little-sleep-structural-sleep-deprivation-in-teens/',
  'content': 'The maximum recommended homework for a high school senior is three hours per night; for younger children, it is ten minutes per grade. If the student goe
s to sleep at 10 PM and gets up at 6 AM ( a typical wake time around here for high school students), this allows 8 hours of sleep.'}]
```

خروجی به صورت یک لیست از دیکشنری است که هر دیکشنری شامل url و content سند بازیابی شده است که این مورد سازگار با چت‌بات ما که بر پایه Langchain است نیست.

۳-۵_ پیاده‌سازی زنجیر

تبدیل داده‌های خروجی به Documentهای LangChain

برای تبدیل داده‌های خروجی جستجو به Documentهای استاندارد LangChain، از کلاس Document استفاده می‌کنیم. هر سند بازیابی شده باید محتوای سند را به عنوان page_content و url سایت مربوطه را به عنوان metadata داشته باشد. برای انجام این کار و پیاده‌سازی پس‌پردازشگر ، از RunnableLambda استفاده می‌کنیم. استفاده از RunnableLambda در اینجا به ما امکان می‌دهد تا یک Parser به صورت کاستوم را به راحتی در قالب یک زنجیر LangChain اضافه کنیم. این ویژگی انعطاف‌پذیری بالایی برای تعریف و استفاده از توابع سفارشی فراهم می‌کند و به ما اجازه می‌دهد تا داده‌های خروجی جستجو را به فرمتی که نیاز داریم تبدیل کنیم.

```
def dict_to_documents(data: List[Dict[str, str]]) -> List[Document]:
    documents = []
    for item in data:
        doc = Document(page_content=item['content'], metadata={"source": item['url']})
        documents.append(doc)
    return documents

custom_parser = RunnableLambda(dict_to_documents)
```

در نهایت زنجیر به صورت زیر پیاده‌سازی شد:

```
search_engine_chain = tavily_tool | custom_parser
```

همان خروجی قبلی اینبار به صورت اسناد سازگار با چت بانت:

```
sample_input = "Is it harmful to stay up at night to do homework?"

search_result = search_engine_chain.invoke(sample_input)
search_result
```

```
[Document(metadata={'source': 'https://www.wikihow.com/Stay-Up-All-Night-Doing-Homework'}, page_content="Use this time to get up and walk around and give your brain a
break. 5. Pump yourself up with a nap. If you're tired before starting your work, take a caffeine nap. Drink a cup of coffee, then immediately take a 20-minute nap. Th
e caffeine will take effect just as you wake up and you'll feel refreshed and energized."),
 Document(metadata={'source': 'https://www.medicalnewstoday.com/articles/325335'}, page_content='Try using blackout curtains or eye masks when sleeping during the day.
Sometimes it is necessary for people to stay up all night for work, studying, or other reasons. Tips that may help a person ...'),
 Document(metadata={'source': 'https://www.sciencedaily.com/releases/2016/09/160919162837.htm'}, page_content='Most people need at least seven to eight hours of sleep
at night for the body and brain to function normally. So, if you stay up all night, missing out on the recommended amount of sleep, your ...'),
 Document(metadata={'source': 'https://www.westlionsroar.com/features/2021/02/25/the-effects-homework-can-have-on-teens-sleeping-habits/'}, page_content='According to
Oxford Learning, homework can have other negative effects on students. In their article, Oxford Learning remarks, "56 percent of students considered homework a primary
source of stress. Too much homework can result in lack of sleep, headaches, exhaustion, and weight loss". Similarly, Stanford Medicine News Center reports that ...'),
 Document(metadata={'source': 'https://www.usatoday.com/story/life/health-wellness/2021/08/16/students-mental-health-time-get-rid-homework-schools/5536050001/'}, page_
content='For older students, Kang says, homework benefits plateau at about two hours per night. "Most students, especially at these high achieving schools, they\'re do
ing a minimum of three hours, and it\'s ...')]
```

همچنین یک جستجوی دیگر هم مرتبط با محتوای کتاب‌ها انجام دادیم که در مراحل بعد یک سری سند مرتبط داشته باشیم تا زنجیرهای بعدی را آزمایش کنیم:

```
sample_input = "How does Multihead Attention Works?"

search_result = search_engine_chain.invoke(sample_input)
search_result
```

```
[Document(metadata={'source': 'https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853'}, page_content='Essential Techniques\nA Gentle Guide to Feature Engineering and Visualization with Geospatial data, in Plain English\ntowardsdatascience.com\nImage Captions with Deep Learning: State-of-the-Art Architectures\nA Gentle Guide to Image Feature Encoders, Sequence Decoders, Attention, and Multi-modal Architectures, in Plain English\ntowardsdatascience.com\nLet's keep learning!\n--\n--\n25\nWritten by Ketan Doshi\nTowards Data Science\nMachine Learning and Big Data\nHelp\nStatus\nAbout\nCareers\nBlog\nPrivacy\nTerms\nText to speech\nTeams After passing through the Layer Norm, this is fed to the Query parameter in the Encoder-Decoder Attention in the first Decoder\nEncoder-Decoder Attention\nAlong with that, the output of the final Encoder in the stack is passed to the Value and Key parameters in the Encoder-Decoder Attention.\n How Attention is used in the Transformer\nAs we discussed in Part 2, Attention is used in the Transformer in three places:\nAttention Input Parameters — Query, Key, and Valu
```

خروجی اسناد:

```
for i, doc in enumerate(search_result):
    print(f"\nDocument {i+1}:")
    print(f"URL: {doc.metadata['source']}")
    print(f"Page Content: {doc.page_content[:300]}...")
    print("="*50)
```

```
Document 1:
URL: https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853
Page Content: Essential Techniques
A Gentle Guide to Feature Engineering and Visualization with Geospatial data, in Plain English
towardsdatascience.com
Image Captions with Deep Learning: State-of-the-Art Architectures
A Gentle Guide to Image Feature Encoders, Sequence Decoders, Attention, and Multi-modal Archite...
==================================================

Document 2:
URL: https://storrs.io/attention/
Page Content: In general, the feature responsible for this uptake is the multi-head attention mechanism. Multi-head attention allows for the neural network to control the mixing of information between pieces of an input sequence, leading to the creation of richer representations, which in turn allows for increase...
==================================================

Document 3:
URL: https://theaisummer.com/self-attention/
Page Content: Insight 2: Based on the fact that the encoder-decoder attention heads are retained mostly in the last layers, it is highlighted that the first layers of the decoder account for language modeling, while the last layers for conditioning on the source sentence.
 Source: Analyzing Multi-Head Self-Attent...
==================================================
```

۶_ پیاده‌سازی Relevancy Check Chain

هدف این بخش پیاده‌سازی زنجیری برای بررسی مرتبط بودن اسناد با پرس‌وجوی کاربر است. این زنجیر با گرفتن یک سند و یک پرس‌وجو به عنوان ورودی، تشخیص می‌دهد که آیا سند با پرس‌وجو مرتبط است یا خیر. خروجی این زنجیر یکی از دو کلمه "relevant" یا "irrelevant" خواهد بود. این زنجیر هم مانند مراحل قبل شامل سه بخش اصلی است: پرامپت، مدل زبانی ، و پس پردازشگر.

۶-۱_ ایجاد زنجیر

طراحی پرامپت

برای ارزیابی ارتباط، پرامپتی طراحی کردیم که به مدل زبانی توضیح می‌دهد که وظیفه دارد ارتباط بین سند و پرس‌وجو را بررسی کند و یکی از دو کلمه "relevant" یا "irrelevant" را به عنوان خروجی ارائه دهد.

متن آن به صورت زیر است:

```
relevancy_prompt_template = (
    "You are given a user query and a document. Your task is to evaluate whether the document is relevant to the query.\n"
    "Return 'relevant' if the document is related to the query and 'irrelevant' otherwise.\n"
    "{output_instructions}\n"
    "User Query: {query}\n"
    "Document: {document}\n"
    "Answer with either relevant or irrelevant"
)

relevancy_prompt = ChatPromptTemplate.from_template(template=relevancy_prompt_template)
```

تعریف مدل زبانی

مدل زبانی را با استفاده از ChatTogether و مدل meta-llama/Llama-3-70b-chat-hf تعریف می‌کنیم و Temperature را به همان دلایل قبلی برابر صفر میگذاریم.

تعریف پس پردازشگر

پس پردازشگر نیز با استفاده از کلاس‌های Pydantic تعریف می‌شود تا خروجی مدل زبانی را ارزیابی کند و مطمئن شود که یکی از مقادیر "relevant" یا "irrelevant" است. این مورد شبیه به موارد قبلی است.

ترکیب زنجیر

زنجیر نهایی با ترکیب پرامپت، مدل زبانی و پس پردازشگر به صورت زیر تعریف شد:

```
relevancy_check_chain = relevancy_prompt | relevancy_llm | relevancy_parser
```

در نهایت مثال‌های مختلفی را بررسی کردیم. ابتدا مثال‌ها را بررسی کرده و سپس به بررسی دلیل استفاده از این زنجیر خواهیم پرداخت:

```
#sample_document = "Transformers are a type of deep learning model introduced in the paper 'Attention is All
sample_document = search_result[0]
sample_query = "How do transformers differ from RNNs and LSTMs in handling long-range dependencies?"
relevancy_check_chain.invoke({
    "query": sample_query,
    "document": sample_document,
    "output_instructions": relevancy_parser.get_format_instructions()
})
```

```
RelevancyCheckParser(relevancy='relevant')
```

در کوئری بالا اسناد همان نتایج جستجوی قسمت قبلی هستند و از آنجا که کوئری مربوط به همان اسناد است پاسخ relevant است.

در اینجا ما همه داکیومنت را داده‌ایم، مثال‌هایی هم در قسمت‌های بعدی بررسی شد که فقط متن را بدهیم. در نهایت برای ارتباط نتیجه گرفتیم متن خالی بهتر است.

```
sample_document = search_result[0]
sample_query = "What are the benefits of using renewable energy sources?"

relevancy_check_chain.invoke({
    "query": sample_query,
    "document": sample_document,
    "output_instructions": relevancy_parser.get_format_instructions()
})
```

```
RelevancyCheckParser(relevancy='irrelevant')
```

در مثال بالا داکیومنت همان قبلی است ولی کوئری در مورد انرژی تجدید پذیر است که کاملا مرتبط است.

حالت برعکس هم ممکن است رخ دهد که داکیومنت متفاوت باشد و سوال مرتبط با NLP:

```
sample_document = search_result_irrelevant[0]
sample_query = "How do transformers differ from RNNs and LSTMs in handling long-range dependencies?"
relevancy_check_chain.invoke({
    "query": sample_query,
    "document": sample_document,
    "output_instructions": relevancy_parser.get_format_instructions()
})
```

```
elevancyCheckParser(relevancy='irrelevant')
```

۲-۶_ نیاز به زنجیر Relevancy Check

ایجاد یک زنجیر بررسی ارتباط برای تعیین مرتبط بودن اسناد با پرس‌وجوی کاربر از اهمیت بالایی برخوردار است. این زنجیر به ما کمک می‌کند تا اطمینان حاصل کنیم که اطلاعاتی که به کاربر ارائه می‌شود دقیقاً مرتبط با نیازها و سوالات اوست.

کاربران به دنبال دریافت اطلاعات دقیق و مرتبط هستند. این زنجیر کمک می‌کند تا کاربران سریع‌تر به پاسخ‌های مناسب دست یابند و تجربه کاربری بهتری داشته باشند. همچنین با فیلتر کردن اسناد نامربوط، سیستم منابع خود را بهینه‌تر مدیریت می‌کند و تنها اطلاعات مرتبط و مفید را به کاربر ارائه می‌دهد.

مثال) فرض کنیم در مثال بالا یک کاربر کوئری تفاوت بین Transformer و RNN در پردازش وابستگی‌های long-range چیست؟" را وارد می‌کند. در پاسخ به این پرس‌وجو، سیستم باید اطمینان حاصل کند که اسناد بازیابی شده دقیقاً مرتبط با این پرس‌وجو هستند.

سندی که به توضیح جزئیات مربوط به مکانیزم توجه (self-attention) در Transformer و چگونگی تفاوت آن با RNN می‌پردازد، یک سند مرتبط خواهد بود. این سند به دقت به پرس‌وجوی کاربر پاسخ می‌دهد و اطلاعات مورد نیاز او را ارائه می‌کند. مثال سند نامرتبط هم در بالا مشاهده کردیم و در نوت بوک هم موجود است.

۷_ پیاده‌سازی Fallback Chain

هدف این بخش پیاده‌سازی زنجیری است که در صورتی که پرس‌وجوی کاربر خارج از حوزه تخصصی NLP باشد، عدم توانایی در پاسخگویی را اعلام کند. این زنجیر با استفاده از سابقه چت کاربر و کوئری فعلی او، به مدل زبانی داده می‌شود و با استفاده از یک پرامپت مناسب، پاسخ مناسب را تولیدشده را به کاربر نمایش می‌دهیم.

۷-۱_ ایجاد زنجیر

طراحی پرامپت

پرامپتی طراحی شد که به مدل زبانی بزرگ توضیح می‌دهد که باید پرس‌وجوهای مرتبط با NLP را پاسخ دهد و در صورتی که پرس‌وجو خارج از این حوزه باشد، عدم توانایی در پاسخگویی را اعلام کند.

```
fallback_prompt_template = (
    "You are a knowledgeable assistant for an NLP course. You can access the user's chat history and their current query. "
    "Your primary goal is to assist with queries related to Natural Language Processing (NLP), including topics such as regular expressions, language m
    "When you encounter a query that falls outside the scope NLP topics, explain that you cannot provide a direct answer.\n\n"
    "Chat History: {chat_history}\n"
    "User Query: {query}\n"
    "Answer only NLP related questions"
)
fallback_prompt = ChatPromptTemplate.from_template(template=fallback_prompt_template)
```

تعریف مدل زبانی

مدل زبانی با استفاده از کلاس ChatTogether و تنظیمات مورد نیاز تعریف می‌شود. در اینجا، از مقدار temperature برابر ۰٫۹ استفاده می‌کنیم تا مدل بتواند پاسخ‌های متنوع‌تری ارائه دهد.

```
fallback_llm = ChatTogether(model_name="meta-llama/Llama-3-70b-chat-hf", temperature=0.9)
```

تعریف پس‌پردازشگر

پس پردازشگر با استفاده از StrOutputParser تعریف می‌شود تا خروجی مدل زبانی به صورت متن ساده باشد.

```
fallback_parser = StrOutputParser()
```

ترکیب زنجیر

زنجیر نهایی با ترکیب پرامپت، مدل زبانی و پس پردازشگر تعریف می‌شود. همچنین سابقه چت کاربر به فرمت متن تبدیل می‌کنیم تا به مدل زبانی داده شود.

```
fallback_chain = (
    {
        "chat_history": lambda x: "\n".join(
            [
                (
                    f"HumanMessage: {msg.content}"
                    if isinstance(msg, HumanMessage)
                    else f"Assistant: {msg.content}"
                )
                for msg in x["chat_history"]
            ]
        ),
        "query": itemgetter("query") ,
    }
    | fallback_prompt
    | fallback_llm
    | fallback_parser
)
```

اهمیت وجود این زنجیر

وجود چنین زنجیری برای بهبود تجربه کاربری و افزایش دقت و کارایی چت بات ضروری است. با داشتن این زنجیر، سیستم می‌تواند به طور خودکار تشخیص دهد که آیا پرس‌وجوی کاربر مرتبط با حوزه تخصصی NLP است یا خیر و در صورت عدم ارتباط، به طور مناسب به کاربر اعلام کند که نمی‌تواند به پرسش او پاسخ دهد. این امر باعث افزایش رضایت کاربران و جلوگیری از ارائه پاسخ‌های نامربوط و گمراه‌کننده می‌شود.

نمونه پاسخ به Hello که آغاز گفتگو است :

```
chat_history_example = []
sample_query = "Hello."

response = fallback_chain.invoke({
    "query": sample_query,
    "chat_history": []
})
print(response)
```

```
Hello! I'm excited to help you with any Natural Language Processing (NLP) related questions you may have. What's on your mind today? Do you have a specific topic in mi
nd, such as regular expressions, language models, or text classification? Or are you looking for general guidance on where to start with NLP?
```

نمونه پاسخ به How to cook a perfect steak?:

```
I'd be happy to help with NLP-related queries! However, I noticed that your previous question was a
bout cooking a perfect steak, which is outside the scope of NLP topics. I'm not equipped to provide co
oking advice, but if you have any questions about Natural Language Processing, I'd be delighted to as
sist you.

Please feel free to ask an NLP-related question, and I'll do my best to provide a helpful response.
```

نمونه پاسخ به همان کوئری قبلی(تاثیر Temperature)

```
I apologize, but it seems like your previous query about cooking a perfect steak is not related to Natu
ral Language Processing (NLP). As a knowledgeable assistant for an NLP course, I'm only equipped t
o assist with queries related to NLP topics such as regular expressions, language models, text classifi
cation, neural networks, transformers, machine translation, and more.

If you have any questions or need help with an NLP-related topic, I'd be more than happy to assist yo
u!
```

۸_ پیاده سازی GENERATE WITH CONTEXT CHAIN

هدف این بخش پیاده‌سازی یک زنجیر برای تولید پاسخ نهایی بر اساس کوئری کاربر و اسناد مرتبط بازیابی و یا جستجوشده است. این زنجیر با استفاده از مدل زبانی ، پرامپت مناسب و پس پردازشگر مربوطه، پاسخ دقیق و جامعی را با کمک اسناد ارائه شده، به کاربر ارائه می‌دهد.

طراحی پرامپت

پرامپت طراحی شده شامل توضیحاتی در مورد اینکه سوال و یک مجموعه از جواب‌های محتمل به مدل داده شده و مدل باید پاسخ را با استفاده از اسناد داده شده تولید کند:

```
generate_with_context_prompt_template = (
    "You are an AI assistant. You have been given a user query and a set of relevant documents. "
    "Your task is to use the information in these documents to generate a comprehensive and accurate response to the query.\n\n"
    "User Query: {query}\n"
    "Relevant Documents:\n"
    "{documents}\n\n"
    "Please provide a detailed and informative response based on the given documents."
)
generate_with_context_prompt = ChatPromptTemplate.from_template(template=generate_with_context_prompt_template)
```

تعریف مدل زبانی

مدل زبانی با استفاده از کلاس ChatTogether و تنظیمات مورد نیاز تعریف می‌شود. در اینجا، از مقدار temperature برابر ۰,۵ استفاده می‌کنیم تا مدل بتواند پاسخ‌های متنوع و مناسبی ارائه دهد.

```
generate_with_context_llm = ChatTogether(model_name="meta-llama/Llama-3-70b-chat-hf", temperature=0.5)
```

تعریف پس پردازشگر

پس پردازشگر با استفاده از StrOutputParser تعریف می‌شود تا خروجی مدل زبانی به صورت متن ساده باشد.

```
generate_with_context_parser = StrOutputParser()
```

ترکیب زنجیر

زنجیر نهایی با ترکیب پرامپت، مدل زبانی و پس پردازشگر تعریف می‌شود. این زنجیر پرس‌وجوی کاربر و اسناد مرتبط را به عنوان ورودی می‌گیرد و پاسخ نهایی را تولید می‌کند.

```
generate_with_context_chain = generate_with_context_prompt | generate_with_context_llm | generate_with_context_parser
```

برای مثال از همان نتایج جستجوی قبلی استفاده می‌کنیم. در مثال اول ارائه شده در نوت بوک تمام سند را به مدل دادیم. این مورد برای زمانی که میخواهیم در پاسخ منبع هم ذکر شود مناسب است. در مثال دوم فقط متن مربوط به هر سند را به مدل زبانی می‌دهیم. هردوی مثال‌ها در نوت بوک هستند.

۹_ آماده‌سازی گراف با استفاده از LANGGRAPH

۹-۱_ پیاده‌سازی گراف با LangGraph

تعریف AgentState

ابتدا وضعیت (state) گراف را با استفاده از TypedDict تعریف می‌کنیم:

```python
class AgentState(TypedDict):
    query: str
    chat_history: List[BaseMessage]
    generation: str
    documents: List[Document]
```

گره‌های مختلف گراف و زنجیرهای مربوطه را به صورت توابع تعریف می‌کنیم:

گره Router

این گره پرس‌وجوی کاربر را می‌گیرد و یکی از سه ابزار (VectorStore، SearchEngine، Fallback) را انتخاب می‌کند:

```python
def router_node(state: dict):
    query = state["query"]
    try:
        response = router_chain.invoke({"query": query, "output_instructions": sample_parser.get_format_instructions()})
        chosen_tool = response["tool_name"].lower()
    except Exception:
        return "LLMFallback"

    if chosen_tool == 'none':
        return "LLMFallback"
    elif chosen_tool == "vectorstore":
        return "VectorStore"
    elif chosen_tool == "searchengine":
        return "SearchEngine"
```

گره VectorStore

این گره اسناد مرتبط را با استفاده از ensemble_retriever بازیابی می‌کند:

```
def vector_store_node(state: dict):
    query = state["query"]
    documents = ensemble_retriever.invoke(input=query)
    return {"documents": documents}
```

گره SearchEngine

این گره اسناد مرتبط را با استفاده از search_engine_chain بازیابی می‌کند:

```
def search_engine_node(state: dict):
    query = state["query"]
    documents = search_engine_chain.invoke(query)
    return {"documents": documents}
```

گره FilterDocs

این گره ارتباط اسناد بازیابی شده را با پرس‌وجوی کاربر بررسی می‌کند:

```
def filter_docs_node(state: dict):
    query = state["query"]
    documents = state["documents"]
    filtered_docs = [
        doc for doc in documents
        if relevancy_check_chain.invoke({"query": query, "document": doc.page_content})["result"] == "relevant"
    ]
    state["documents"] = filtered_docs
    return {"documents": filtered_docs}
```

گره Fallback

این گره پاسخ مناسب را در صورتی که پرس‌وجوی کاربر خارج از حوزه تخصصی باشد تولید می‌کند:

```
def fallback_node(state: dict):
    query = state["query"]
    chat_history = state["chat_history"]
    generation = fallback_chain.invoke({"query": query, "chat_history": chat_history})
    return {"generation": generation}
```

گره Generate With Context

این گره پاسخ نهایی را با استفاده از اسناد مرتبط تولید می‌کند:

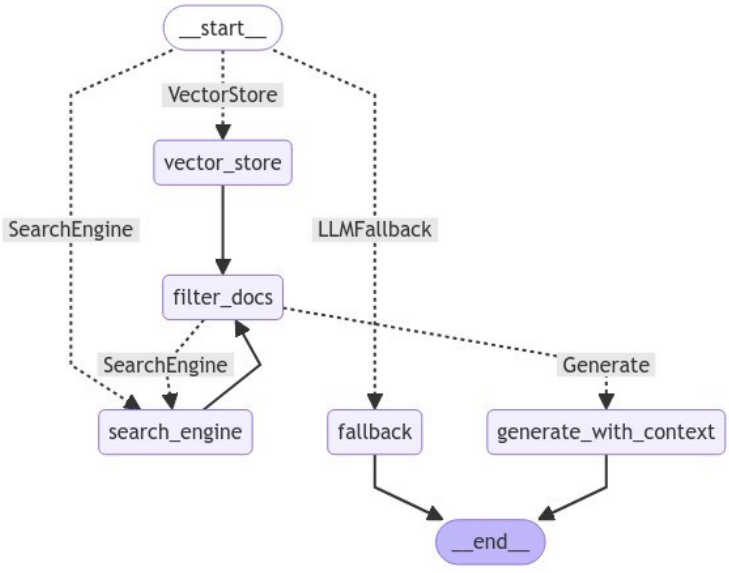
```python
def generate_with_context_node(state: dict):
    query = state["query"]
    documents = state["documents"]
    generation = generate_with_context_chain.invoke({"query": query, "documents": "\n".join([doc.page_content for doc in documents])})
    return {"generation": generation}
```

۲-۹_ ساخت گراف

گراف را با استفاده از StateGraph تعریف می‌کنیم و گره‌ها و یال‌های لازم را اضافه می‌کنیم:

```python
workflow = StateGraph(AgentState)

workflow.add_node("vector_store", vector_store_node)
workflow.add_node("search_engine", search_engine_node)
workflow.add_node("filter_docs", filter_docs_node)
workflow.add_node("fallback", fallback_node)
workflow.add_node("generate_with_context", generate_with_context_node)

workflow.set_conditional_entry_point(
    router_node,
    {
        "LLMFallback": "fallback",
        "VectorStore": "vector_store",
        "SearchEngine": "search_engine",
    },
)

workflow.add_conditional_edges(
    "filter_docs",
    router_node,
    {
        "SearchEngine": "search_engine",
        "Generate": "generate_with_context",
    },
)

workflow.add_edge("vector_store", "filter_docs")
workflow.add_edge("search_engine", "filter_docs")
workflow.add_edge("generate_with_context", END)
workflow.add_edge("fallback", END)
```

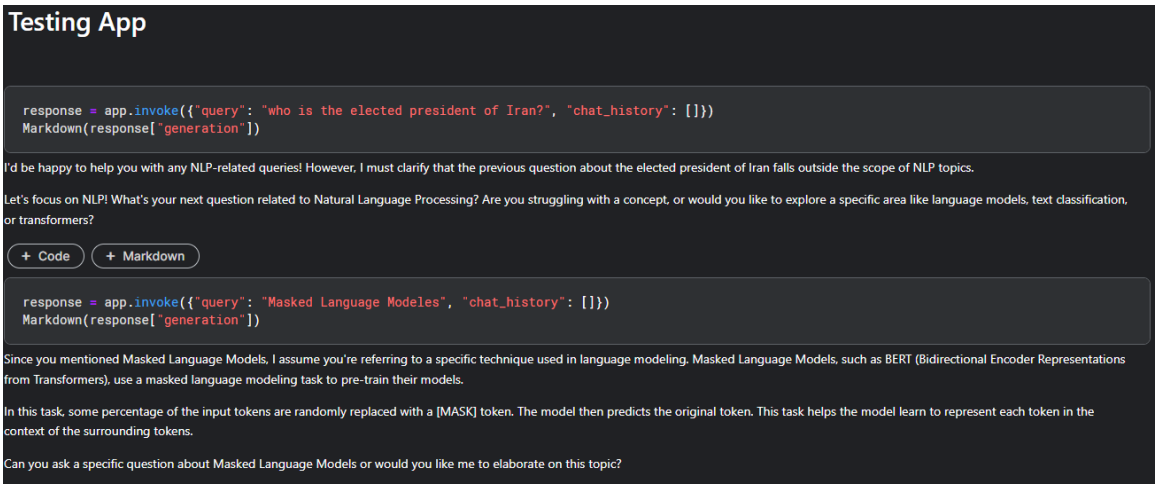
بر اساس نمودار ارائه شده و کد نوشته شده، همه بخش‌ها به درستی به هم متصل شده‌اند و روند کار به درستی پیاده‌سازی شده است.



شکل ۶ گراف تشکیل شده نهایی

۹-۳_ تست نهایی

برنامه را با پرس‌وجوهای مختلف تست می‌کنیم تا از عملکرد صحیح آن اطمینان حاصل کنیم.



با اجرای کد بالا، دو پرس‌وجو به برنامه داده می‌شود:

کوئری اول: who is the elected president of Iran?

چت بات اعلام می‌کند که این پرسش خارج از حوزه NLP است و پیشنهاد می‌دهد که کاربر سوال دیگری مرتبط با NLP بپرسد.

کوئری دوم: Masked Language Modeles

چت بات توضیحی در مورد Masked Language Models می‌دهد و کاربر را به پرسیدن سوالات دقیق‌تر راهنمایی میکند.

هردوی موارد بالا نشان می‌دهد که موارد به خوبی پیاده‌سازی شده‌اند.

۹-۴_ اجرای چت‌بات با استفاده از Gradio

باتوجه به اینکه این مورد جز صوت تمرین قرار داده‌نشد و باتوجه به زمان محدود گزارش مربوط به این قسمت تکمیل نشد.

کتابخانه Gradio: برای ایجاد رابط کاربری وب استفاده می‌شود که کاربر می‌تواند پرس‌وجوهای خود را وارد کرده و پاسخ‌های چت‌بات را دریافت کند.

تابع chat: این تابع پرس‌وجوی کاربر را پردازش می‌کند، تاریخچه چت را به‌روزرسانی می‌کند و پاسخ چت‌بات را تولید می‌کند.

رابط کاربری Gradio: یک رابط کاربری ساده با دو قسمت (یکی برای ورودی پرس‌وجوی کاربر و دیگری برای نمایش پاسخ چت‌بات و اسناد بازیابی شده) ایجاد می‌کند.