به نام خدا

یادگیری عمیق

بهار ۱۴۰۴

استاد: دکتر بجانی

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی برق

پروژه نهایی | AI Agent | مهلت ارسال: چهارشنبه ۱۰ تیر

- مهلت ارسال پاسخ تا ساعت ۲۳:۵۹ روز مشخص‌شده است.
- همکاری و همفکری شما در انجام تمرین مانعی ندارد اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد. (دقت کنید در صورت تشخیص مشابهت غیرعادی برخورد جدی صورت خواهد گرفت.)
- در صورت همفکری و یا استفاده از هر منابع خارج درسی، نام همفکران و آدرس منابع مورد استفاده برای حل سوال مورد نظر را ذکر کنید.
- لطفا تصویری واضح از پاسخ سوالات نظری بارگذاری کنید. در غیر این صورت پاسخ شما تصحیح نخواهد شد.
- نتایج و پاسخ های خود را در یک فایل با فرمت zip به نام Project-StudentNumber-Name در سایت CW قرار دهید. برای بخش عملی تمرین نیز در صورتی که کد تمرین و نتایج خود را در گیت‌هاب بارگذاری می‌کنید، لینک مخزن مربوطه (repository) را در پاسخنامه خود قرار بدهید. همچنین لازم است تا دسترسی های لازم را به دستیاران آموزشی مربوط به این تمرین بدهید.
- دقت کنید کدهای شما باید قابلیت اجرای دوباره داشته باشند، در صورت دادن خطا هنگام اجرای کدتان، حتی اگه خطا بدلیل اشتباه تایپی باشد، نمره صفر به آن بخش تعلق خواهد گرفت.

مقدمه

با رشد مدل‌های زبانی بزرگ (LLMs)، ایجاد عامل‌های هوشمند که بتوانند با انسان تعامل طبیعی، هدفمند و دانشی داشته باشند، به موضوعی داغ در حوزه هوش مصنوعی تبدیل شده‌اند. در این پروژه، هدف توسعه یک عامل هوشمند چت‌محور (AI Chat Agent) است که با بهره‌گیری از یک مدل زبانی متن‌باز با حداکثر ۷ میلیارد پارامتر، قابلیت‌های زیر را در اختیار کاربر قرار دهد:

۱. گفت‌وگو با حفظ سابقه مکالمات

۲. استفاده از تولید مبتنی بر بازیابی (RAG)

۳. ادغام اطلاعات به‌روز از وب با موتور Exa

۴. امکان اجرای توابع خارجی (Function Calling)

۵. تعامل تعاملی از طریق یک بازی (بازی ۲۰ سوالی)

۶. یک رابط کاربری ساده و قابل استفاده با Streamlit یا Gradio

توجه: تمام مراحل و خواسته‌های پروژه را باید از ابتدا پیاده کنید و حق استفاده از ابزارهای آماده را در صورت عدم ذکر در صورت سوال ندارید.

۱ مکالمه با تاریخچه

هدف این بخش طراحی یک سیستم چت‌بات مبتنی بر مدل زبان بزرگ (LLM) است که توانایی حفظ تاریخچه مکالمه را دارد و می‌تواند در پاسخ‌دهی به ورودی‌های جدید، از حافظه گفت‌وگو استفاده کند. این ویژگی تجربه‌ای طبیعی‌تر و هوشمندانه‌تر برای کاربر ایجاد می‌کند.
در سیستم‌های چت، مدل زبان باید نه‌تنها به آخرین پیام، بلکه به زمینه مکالمه (context) نیز توجه داشته باشد. بدون حافظه، مدل نمی‌تواند به پیام‌هایی که در گذشته مطرح شده‌اند ارجاع دهد. بنابراین، باید سیستمی طراحی کنیم که: کل مکالمه را به‌شکل ساختاری ذخیره کند، در هر مرحله، نسخه‌ای به‌روزشده از مکالمه را به مدل بدهد و در صورت طولانی شدن، پیام‌های قدیمی را به‌صورت هوشمندانه حذف یا خلاصه کند (context window management).

۲ پیاده‌سازی Retrieval-Augmented Generation (RAG)

هدف این بخش افزایش دقت و غنای پاسخ‌های مدل با ترکیب توانایی زبانی مدل با اطلاعات بازیابی‌شده از منابع متنی (داخلی) است. به‌جای اینکه مدل صرفاً بر حافظه آموزش‌دیده تکیه کند، از یک سیستم بیرونی برای بردارسازی، ذخیره‌سازی و بازیابی اسناد استفاده می‌کنیم. سیستم شما باید امکان پردازش فایل PDF و استفاده از اطلاعات آن در پاسخ‌گویی به سوالات کاربر را فراهم کند.
RAG یک چارچوب برای ترکیب مدل‌های زبانی با اطلاعات خارجی است. در این روش، زمانی که مدل با یک پرسش مواجه می‌شود:

۱. ابتدا از روی پرسش، یک بردار embedding تولید می‌شود.

۲. این بردار با بردارهای ذخیره‌شده از منابع دانشی (مثل اسناد یا ویکی‌پدیا) مقایسه می‌شود.

۳. نزدیک‌ترین بردارها بازیابی شده و محتوای متنی آن‌ها به مدل LLM داده می‌شود.

۴. مدل پاسخ نهایی را با در نظر گرفتن این محتوای تازه تولید می‌کند.

سیستم پیاده‌سازی شده شما باید امکان بارگذاری یک فایل PDF را داشته باشد. محتوای فایل PDF را با استفاده از کتابخانه‌هایی مانند PyMuPDF یا pdfminer استخراج کنید. پس از استخراج متن، آن را به بخش‌های کوچک (chunck) تقسیم کنید (مثلاً هر ۱۰۰ کلمه) و سپس embedding آن‌ها را محاسبه نمایید. برای embedding شما می‌توانید از یک مدل مانند all-MiniLM-L6-v2 برای تولید بردارهای متنی استفاده کنید. از کتابخانه faiss برای ساخت اندیس استفاده کنید. اندیس باید از نوع IndexFlatL2 باشد و بتواند بردارهای حاصل از embedding را ذخیره و جستجو کند. پس از ساخت اندیس، آن را ذخیره کنید تا در فراخوانی‌های بعدی نیز قابل استفاده باشد. برای هر سوال کاربر، ابتدا embedding آن را محاسبه کنید. سپس این بردار را در اندیس FAISS جستجو کرده و $k = ۳$ قطعه متنی با بیشترین شباهت را استخراج کنید. محتوای این قطعات باید به مدل زبان (در بخش اول پروژه) تزریق شود تا مدل پاسخ را با توجه به آن‌ها تولید کند.

۳ فراخوانی توابع Functions Calling

در این بخش، هدف طراحی سیستمی است که در جریان مکالمه، توانایی تشخیص هدف تعامل را داشته باشد؛ یعنی:

- اگر پرسش یا درخواست کاربر نیاز به اطلاعات به‌روز دارد (مانند اخبار، قیمت‌ها، نتایج، وضعیت هوا و...)، سیستم به‌طور خودکار تشخیص دهد که باید از وب جستجو انجام شود.
- اگر کاربر بخواهد وارد وضعیت خاصی مانند بازی ۲۰ سوالی شود، سیستم این خواست را شناسایی کند و به‌صورت خودکار وارد مود بازی شود.

۱.۳ تشخیص نیاز به جستجو در وب (Web Search Trigger):

باید مدلی طراحی یا پیکربندی شود که هر پیام ورودی را از نظر «نیاز به جستجو» ارزیابی کند. برای این منظور، از مدل LLM استفاده می‌کنید و یک prompt طراحی می‌کنید با هدف تشخیص اینکه آیا این پیام نیاز به اطلاعات خارجی دارد یا نه. اگر خروجی مدل "yes" باشد، سیستم باید به‌صورت خودکار به سراغ جستجوگر وب (مثل Exa Search API) برود و نتایج جستجو را استخراج و خلاصه‌سازی کند. اگر "no" باشد، ادامه مکالمه طبق روال عادی انجام می‌شود.
از Exa API برای جستجوی آنلاین استفاده کنید. باید متن بازیابی‌شده را خلاصه‌سازی کرده و به prompt مدل اضافه نمایید. از میان نتایج، فقط ۳ مورد مرتبط را استخراج و به مدل تزریق کنید.

۲.۳ تشخیص درخواست بازی از سوی کاربر:

باید سیستمی پیاده‌سازی شود که با ورودی کاربر تشخیص دهد آیا کاربر قصد ورود به بازی دارد یا خیر. در این پروژه، فقط بازی "۲۰ سوالی" تعریف می‌شود. با استفاده از مدل زبانی و یک prompt خاص، تشخیص دهید آیا کاربر گفته‌ای مرتبط با شروع بازی داشته یا خیر.
فرایند این بازی به صورت زیر است:

۱. پیام اولیه بازی توسط مدل مانند این جمله نمایش داده می‌شود: «لطفاً یک کلمه در ذهنتان انتخاب کنید. من سعی می‌کنم با حداکثر ۲۰ سوال بله/خیر آن را حدس بزنم.»

۲. کاربر یک کلمه در ذهن خود انتخاب میکنید

۳. در هر مرحله، مدل یک سوال بله/خیر بپرسد.

۴. پاسخ کاربر را دریافت کند (Yes یا No).

۵. بعد از هر سوال، مدل یک کلمه حدس می‌زند.

۶. کاربر اعلام می‌کند که حدس درست بوده (Yes) یا غلط بوده (No).

۷. اگر حدس درست باشد، بازی تمام شده و مدل برنده می‌شود.

۸. اگر مدل در حداکثر ۲۰ مرحله نتواند حدس صحیح را بزند، بازی پایان یافته و مدل بازنده است.

این بازی سال گذشته در Kaggle به عنوان یک مسابقه معرفی شد. شما می‌توانید اطلاعات بیشتر برای پیاده‌سازی و بهبود نتایج خود را از لینک زیر مطالعه کنید:
https://www.kaggle.com/competitions/llm-20-questions
همچنین مکانیسمی برای خروج از بازی در صورت عدم تمایل کاربر به ادامه بازی در نظر بگیرید.

خروجی این بخش شما توسط دیتاست تست تیم تی‌ای‌ها به صورت زیر ارزیابی می‌شود:
فرض کنید به جای کاربر یک مدل زبانی تایید کننده داشته باشیم که با گرفتن یک کلمه، به پرسش‌های مدل زبانی شما پاسخ بله یا خیر می‌دهد. این مدل به صورت زیر تعریف و قابل استفاده است:

```
from test import ValidatorModel  # The test.py file will be provided by the TAs

# Create an instance of the ValidatorModel, which selects a random word from a dataset
validator_model = ValidatorModel()

# Ask a yes/no question about the unknown word
question = "Is this an animal?"
res = validator_model.validate_question(question)

# Make a guess about what the word is
guess = "lion"
res = validator_model.validate_guess(guess)
```

شما باید یک script بنویسید که مدل validator را از فایل test.py لود کند و مدل شما را ارزیابی کند. این اسکریپت باید یک عدد به عنوان تعداد دفعات بازی ورودی بگیرد و نتیجه نهایی، یعنی تعداد دفعاتی که مدل شما موفق به حدس صحیح کلمه شده‌است را خروجی دهد. کد شما باید با دستور زیر قابل اجرا باشد و نتیجه را نشان دهد:

```
python evaluate_20Q.py -N 100
```

۴ طراحی رابط کاربری گرافیکی (GUI)

هدف این بخش ایجاد یک رابط کاربری گرافیکی ساده و کاربرپسند که بتواند تعامل با سیستم هوش مصنوعی را در موضوعات مطرح شده در این پروژه فراهم کند. می‌توانید از ابزارهای Streamlit یا Gradio یا هر فریم‌ورک دیگری استفاده کنید تا قابلیت‌های اصلی سیستم را از طریق یک رابط بصری در دسترس قرار دهید.