شرح مسئله

با پیشرفت مدل‌های زبانی بزرگ (LLMs)، مانند LLaMA، Claude، GPT و دیگر مدل‌ها، اکنون امکان ساخت عامل‌های هوشمند خودمختار فراهم شده است؛ عامل‌هایی که می‌توانند وظایف پیچیده را با استفاده از زبان طبیعی، تعامل با کاربر، و بهره‌گیری از ابزارهای مختلف انجام دهند.

در این چالش، شرکت کنندگان باید یک عامل هوشمند مبتنی بر LLM طراحی کنند که بتواند یک یا چند وظیفه مشخص را به صورت خودکار انجام دهد.

پیشنهاد ما: برای توسعه محصولات مختلف، معمولا از مخزنهای مختلف در github استفاده می شود و با ویرایش و تغییراتی در آنها محصولات جدید ساخته می شود. معمولا مسلط شدن بر هر مخزن قدم اصلی برای شروع فرآیند توسعه است. در این محصول ما دنبال این هستیم که یک دستیار برای این کار ایجاد کنیم. یعنی دستیاری که آدرس یک مخزن رو بگیره و بعد به سوالات ما در مورد اون جواب بده. تغییرات مد نظر ما رو ایجاد کنه یا مشکلاتی که موقع یک استفاده خاص از کتابخانه بهش برخوردیم رو برامون حل کنه.

حتی یک سری کاربردهای خاصتر می تونه تحلیل یک مخزن از لحاظ امنیتی و درآوردن مشکلات و حفره های امنیتی یک نرم افزار باشه. به این صورت که از ابزارهای مختلف تحلیل کد استفاده بشه و در کنار LLM نتایج اونها بررسی بشه و یک گزارش امنیتی درباره کتابخانه داده بشه.

یا مثلا در مورد معماری نرم افزار و الگوهای طراحی کلاس ها پیشنهادهای اصلاحی توسط agent داده بشه.

در این مساله یک عامل هوشمند می خوایم که از API های github و ابزارهایی برای کامپایل کردن و پیمایش بر روی کد استفاده کنه و مانند یک مهندس نرم افزار بر اساس سوال کاربر، فایل های مختلف رو بررسی کنه و تغییرات احتمالی لازم رو بده و با تعامل با کامپایلر به نتیجه درست برسه.

اهداف مسئله

- ساخت عامل (Agent) مبتنی بر LLM که توانایی درک وظایف، برنامه ریزی، اجرای اقدامات و تعامل چندمرحله‌ای با کاربران را برای انجام وظایف مهندسی نرم افزار داشته باشد.
- قابلیت استفاده از ابزارها، APIها، یا منابع خارجی (مثلاً جست‌وجو در وب و فایل های مختلف کد).
- ایجاد تغییرات لازم در کد در صورت درخواست کاربر
- اجرای کد و برطرف خطاهای احتمالی کامپایلری
- گزارش نتیجه در قالب مناسب (مثلا اگه کد تغییر داده شده یک فایل diff، اگه بررسی امنیتی خواسته شده یک گزارش pdf)

خروجی مورد انتظار

- نمونه اولیه قابل اجرا (**Web App، CLI، API**) یک سیستم که بتواند با دریافت آدرس مخزن پردازش های اولیه (در صورت نیاز) را آغاز کند و در ادامه در مورد کد به گفتگو با کاربر بپردازد و به سوالات پاسخ دهد.
- ویدئوی دمو: نمایش عملکرد سیستم، از ورودی گرفتن آدرس مخزن تا انجام مکالمه و نمایش نتایج.
- توضیحات **README:** شامل جزئیات معماری ایجنت، توضیحات در مورد نحوه تعامل اجزای مختلف سیستم (LLM، ابزارها، API ها، و منطق عامل). مراحل پیاده‌سازی و آزمایش های که طی مسیر طی شده به علاوه راهنمای استفاده.

ابزارها و منابع پیشنهادی

- LLM APIs: GPT-4، Claude، Mistral، Gemini و غیره
- Agent frameworks: Pydantic AI
- Vector Graph DB: Neo4J, Nebula
- Vector DBs: Milvus، Chroma
- UI tools: Streamlit، Gradio، Flask
- استفاده از ابزارهای مختلف پیمایش کد مانند tree-sitter

نکات کلیدی داوری

به دنبال راه حل‌هایی هستیم که:

- تا جای ممکن فرآیندها را به صورت خودکار و بدون دخالت انسانی انجام دهند. هر چه وظایف سخت تر و متنوع تری پشتیبانی شود، امتیاز بیشتری خواهد داشت.
- خلاقانه باشند: نوآوری در رویکرد تحلیل و فهم یک کتابخانه با استفاده از LLMها.
- تأثیر عملی یا اجتماعی واقعی داشته باشند: پتانسیل ساده‌سازی و تسریع فرآیندهای توسعه نرم افزار
- پشتیبانی از زبان پایتون ضروری است. در ادامه پشتیبانی از C و C++ امتیاز مثبت محسوب می شود.