Tóm tắt Chi Tiết Tuần 5 - Mastering RAG

I. Tóm tắt chi tiết nội dung các ngày tuần 5

Tuần 5 của khóa học "Mastering RAG: Build Advanced Solutions with Vector Embeddings & LangChain" tập trung vào việc triển khai các hệ thống Retrieval-Augmented Generation (RAG) nâng cao, giúp cải thiện hiệu suất của các mô hình ngôn ngữ lớn (LLM) bằng cách sử dụng vector embeddings và LangChain.

Ngày 1: Giới thiệu về RAG và hệ thống cơ bản

A. Nội dung chính

- Tìm hiểu về Retrieval-Augmented Generation (RAG), một kỹ thuật kết hợp truy xuất thông tin và sinh văn bản.
- Các phương pháp làm giàu thông tin cho LLM: multi-shot prompting, sử dụng công cụ trợ giúp và truy xuất tri thức từ cơ sở dữ liệu.
- Xây dựng một mô hình RAG đơn giản: lấy dữ liệu từ cơ sở dữ liệu nội bộ và bổ sung vào prompt của mô hình.
- Ứng dụng RAG vào việc phát triển chatbot hỗ trợ khách hàng trong lĩnh vực bảo hiểm.

B. Kỹ năng đạt được

- Hiểu được khái niệm và ứng dụng của RAG trong việc cải thiện phản hồi của LLM.
- Biết cách tích hợp thông tin bên ngoài vào prompt để nâng cao độ chính xác.
- Xây dựng một hệ thống RAG cơ bản với Python.

Ngày 2: Sử dụng LangChain để triển khai RAG

A. *Nội dung chính*

- Giới thiệu về LangChain, framework giúp xây dựng hệ thống LLM dễ dàng.
- Cách LangChain giúp quản lý pipeline của hệ thống RAG bằng cách chia nhỏ dữ liệu và tích hợp với cơ sở dữ liệu vector.
- Tạo text chunks từ tài liệu lớn để tăng hiệu suất tìm kiếm thông tin trong RAG.
- Sử dụng LangChain để đọc tài liệu, gán metadata, và chuẩn bị dữ liệu để lưu trữ trong vector store.

B. Kỹ năng đạt được

- Biết cách sử dụng LangChain để quản lý pipeline của RAG.
- Thành thạo cách chia nhỏ văn bản và tối ưu hóa truy xuất dữ liệu.
- Hiểu được cách lưu trữ và xử lý văn bản trong vector store.

Ngày 3: Vector Embeddings và ChromaDB

A. Nội dung chính
- Giới thiệu về vector embeddings và cách chúng giúp tìm kiếm thông tin dựa trên ngữ nghĩa.
- Tìm hiểu về OpenAI Embeddings và cách mã hóa văn bản thành vector số.
- Giới thiệu ChromaDB, một cơ sở dữ liệu vector giúp lưu trữ và truy xuất embeddings.
- Trực quan hóa dữ liệu vector bằng t-SNE và Plotly để hiểu cách chúng hoạt động.

B. Kỹ năng đạt được
- Biết cách sử dụng OpenAI Embeddings để tạo vector từ văn bản.
- Hiểu được cách vector hóa dữ liệu giúp cải thiện quá trình tìm kiếm.
- Thành thạo cách sử dụng ChromaDB để lưu trữ và tìm kiếm vector.

Ngày 4: Xây dựng Pipeline RAG hoàn chỉnh

A. Nội dung chính
- Xây dựng một pipeline hoàn chỉnh cho RAG sử dụng LangChain.
- Tìm hiểu về các thành phần chính: LLM, retriever (bộ truy xuất dữ liệu), memory (bộ nhớ hội thoại).
- Kết nối LLM với vector store và tối ưu hóa quy trình truy xuất dữ liệu.
- Tạo một chatbot có khả năng tìm kiếm thông tin và phản hồi thông minh.

B. Kỹ năng đạt được
- Xây dựng một hệ thống RAG hoàn chỉnh với LangChain.
- Biết cách tích hợp bộ nhớ hội thoại để chatbot có thể nhớ ngữ cảnh.
- Thành thạo cách sử dụng vector store để cải thiện truy vấn dữ liệu.

Ngày 5: Tối ưu hóa hệ thống RAG

A. Nội dung chính
- Xử lý các lỗi phổ biến trong hệ thống RAG và cách khắc phục.
- Giới thiệu LangChain Expression Language (LCL) để viết pipeline dễ dàng hơn.
- So sánh ChromaDB với FAISS (Facebook AI Similarity Search) trong lưu trữ vector.
- Tối ưu hóa truy vấn trong RAG để tăng hiệu suất tìm kiếm.

B. Kỹ năng đạt được
- Hiểu được cách tối ưu hóa hệ thống RAG để cải thiện tốc độ và độ chính xác.
- So sánh và lựa chọn vector store phù hợp với từng bài toán.
- Biết cách xử lý lỗi khi làm việc với LangChain và cơ sở dữ liệu vector.

II. Từ khóa quan trọng và ý nghĩa của chúng

2.1.Retrieval-Augmented Generation (RAG)

RAG là một kỹ thuật kết hợp giữa truy xuất dữ liệu và mô hình sinh văn bản (LLM). Thay vì chỉ dựa vào dữ liệu huấn luyện, hệ thống RAG có thể tìm kiếm thông tin liên quan từ một cơ sở tri thức, giúp cải thiện độ chính xác và giảm thiểu vấn đề "ảo giác" của LLM.

Ứng dụng:
- Hệ thống hỏi đáp thông minh có thể tra cứu thông tin từ tài liệu hoặc cơ sở dữ liệu.
- Chatbot hỗ trợ khách hàng có thể lấy dữ liệu từ kho thông tin của doanh nghiệp.

2.2.LangChain

LangChain là một framework giúp kết nối các thành phần quan trọng trong một ứng dụng LLM. Nó giúp đơn giản hóa quá trình xử lý dữ liệu, xây dựng pipeline và triển khai hệ thống RAG.

Ứng dụng:
- Dùng để xây dựng chatbot có khả năng ghi nhớ lịch sử hội thoại.
- Giúp quản lý pipeline trong hệ thống RAG, kết nối LLM với cơ sở dữ liệu vector.

2.3.Vector Embeddings

Là phương pháp chuyển đổi văn bản thành vector số để giúp hệ thống hiểu được mối quan hệ giữa các từ và ngữ nghĩa của chúng.

Ứng dụng:
- Tìm kiếm văn bản thông minh dựa trên ý nghĩa thay vì từ khóa.
- Phát triển hệ thống gợi ý nội dung theo ngữ cảnh.

2.4.OpenAI Embeddings

Là một mô hình chuyển đổi văn bản thành vector được phát triển bởi OpenAI, giúp cải thiện độ chính xác khi tìm kiếm thông tin.

Ứng dụng:
- Lưu trữ và tìm kiếm thông tin trong chatbot hoặc hệ thống tìm kiếm văn bản.

2.5.ChromaDB

ChromaDB là một cơ sở dữ liệu vector giúp lưu trữ và tìm kiếm embeddings nhanh chóng.

Ứng dụng:
- Dùng để lưu trữ tri thức của công ty, giúp chatbot có thể tìm kiếm thông tin

nhanh chóng.

2.6.FAISS (Facebook AI Similarity Search)

Một công cụ tìm kiếm vector hiệu suất cao, hỗ trợ tìm kiếm nhanh chóng dựa trên độ tương đồng.

Ứng dụng:

- Lọc và tìm kiếm thông tin liên quan từ dữ liệu lớn theo cách tối ưu hơn so với phương pháp tìm kiếm truyền thống.

2.7.Text Chunking

Kỹ thuật chia nhỏ văn bản thành các đoạn nhỏ hơn giúp quá trình lưu trữ và truy xuất dữ liệu hiệu quả hơn.

Ứng dụng:

- Cải thiện khả năng tìm kiếm thông tin chính xác trong hệ thống RAG.

2.8.Conversational Memory

Là công nghệ giúp chatbot lưu trữ ngữ cảnh cuộc trò chuyện, giúp phản hồi tốt hơn.

Ứng dụng:

- Chatbot thông minh có thể nhớ thông tin từ các cuộc trò chuyện trước đó để tương tác tốt hơn với người dùng.

2.9.Vector Search

Là phương pháp tìm kiếm thông tin dựa trên vector thay vì tìm kiếm theo từ khóa truyền thống.

Ứng dụng:

- Dùng trong hệ thống tìm kiếm thông minh, giúp trả về kết quả có ý nghĩa thay vì chỉ dựa vào từ khóa.

2.10.Prompt Engineering

Là nghệ thuật thiết kế prompt hiệu quả để tối ưu phản hồi của LLM.

Ứng dụng:

- Giúp cải thiện chất lượng câu trả lời của AI khi sử dụng trong chatbot hoặc hệ thống hỗ trợ khách hàng.

III. Công nghệ được sử dụng và ứng dụng trong RAG

3.1.LLM Frameworks

- OpenAI GPT: Cung cấp khả năng sinh văn bản tiên tiến, giúp tạo phản hồi thông minh trong hệ thống RAG.
- Claude: Một mô hình LLM thay thế GPT, được sử dụng để sinh văn bản có độ chính xác cao.
- BERT: Được sử dụng trong các bài toán phân loại văn bản, tìm kiếm ngữ nghĩa và tạo embeddings.

Ứng dụng:
- Tạo chatbot có thể hiểu và phản hồi tự nhiên hơn.
- Xây dựng hệ thống gợi ý thông minh dựa trên dữ liệu đầu vào của người dùng.

3.2.Vector Databases

- ChromaDB: Cơ sở dữ liệu lưu trữ vector embeddings, giúp tìm kiếm dữ liệu theo ý nghĩa thay vì từ khóa đơn thuần.
- FAISS (Facebook AI Similarity Search): Một công cụ tối ưu hóa tìm kiếm vector, giúp truy vấn thông tin nhanh hơn.

Ứng dụng:
- Lưu trữ tri thức trong hệ thống hỏi đáp thông minh.
- Cải thiện tốc độ và độ chính xác khi tìm kiếm dữ liệu trong các ứng dụng lớn.

3.3.Python Libraries

- LangChain: Giúp xây dựng hệ thống RAG bằng cách kết nối các thành phần như LLM, bộ truy xuất dữ liệu, và cơ sở dữ liệu vector.
- OpenAI API: Cung cấp các mô hình sinh văn bản và embeddings để tăng cường khả năng xử lý ngôn ngữ tự nhiên.
- t-SNE: Kỹ thuật giảm chiều dữ liệu, giúp trực quan hóa vector embeddings.
- Plotly: Thư viện vẽ biểu đồ giúp phân tích và hiển thị dữ liệu trong hệ thống RAG.

Ứng dụng:
- Tích hợp OpenAI API để xây dựng chatbot thông minh.
- Sử dụng LangChain để quản lý pipeline của hệ thống LLM.

3.4.Môi trường phát triển

- JupyterLab: Được sử dụng để thực hành và chạy thử nghiệm các mô hình AI.

Ứng dụng:
- Viết và thử nghiệm code trong quá trình phát triển hệ thống RAG.
- Dễ dàng kiểm tra dữ liệu và kết quả từ mô hình LLM.