TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

-----*-----**

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

ĐỀ TÀI

Chatbot app ứng dụng RAG và Elastic Search

Nhóm sinh viên thực hiện:

Phạm Đức Hoàng - MSV: 22022200

Hoàng Ngọc Hào - MSV: 22022668

Giảng viên hướng dẫn: TS. Trần Hồng Việt

HÀ NỘI, 12/2024

MỤC LỤC

1. MỞ ĐẦU

Trong thời đại cách mạng công nghiệp 4.0, dữ liệu đã trở thành một trong những tài nguyên quan trọng nhất của mọi lĩnh vực. Khái niệm **Big Data** không chỉ đơn thuần là sự gia tăng về khối lượng dữ liệu mà còn bao hàm khả năng xử lý và khai thác dữ liệu khổng lồ để đưa ra những thông tin có giá trị. Với sự phát triển của công nghệ, Big Data đã mở ra những cơ hội to lớn nhưng cũng đặt ra nhiều thách thức đối với các tổ chức và doanh nghiệp trong việc lưu trữ, phân tích và ứng dụng.

Big Data không chỉ thay đổi cách chúng ta tiếp cận thông tin mà còn định hình lại nhiều lĩnh vực như tài chính, y tế, giáo dục, thương mại, và chính phủ. Việc hiểu và khai thác Big Data một cách hiệu quả có thể giúp các tổ chức không chỉ tối ưu hóa hoạt động mà còn tạo ra lợi thế cạnh tranh vượt trội.

Chatbot là một hệ thống phần mềm được thiết kế để mô phỏng các cuộc hội thoại với người dùng, thường thông qua giao diện văn bản hoặc âm thanh. Chúng đóng vai trò quan trọng trong việc tự động hóa giao tiếp, giúp trả lời các câu hỏi thường gặp, cung cấp dịch vụ hỗ trợ khách hàng, hoặc thậm chí tham gia vào các cuộc trò chuyện phức tạp trong các lĩnh vực như bán hàng, chăm sóc sức khỏe và giáo dục.

Trong bối cảnh Big Data, việc tích hợp các **Large Language Models (LLMs)** vào chatbot đã tạo ra bước ngoặt lớn, mang đến khả năng xử lý dữ liệu và giao tiếp vượt trội so với các chatbot truyền thống. Những mô hình như GPT-4 hay các giải pháp LLM khác cung cấp khả năng:

- **Hiểu và xử lý ngữ nghĩa phức tạp:** LLMs cho phép chatbot không chỉ trả lời các câu hỏi đơn giản mà còn có thể giải quyết các câu hỏi phức tạp, mang tính ngữ nghĩa cao, tạo ra trải nghiệm tương tác tự nhiên hơn.

- **Tăng cường khả năng sinh ngữ cảnh:** Chatbot tích hợp LLMs có thể duy trì mạch trò chuyện mượt mà, đảm bảo người dùng luôn cảm thấy được quan tâm và hiểu rõ nhu cầu của họ.
- **Xử lý ngữ pháp và cú pháp nâng cao:** Các phản hồi của chatbot được tối ưu về ngữ pháp và cấu trúc câu, tạo sự chuyên nghiệp và chính xác trong giao tiếp.
- **Học từ dữ liệu riêng:** Khả năng học và thích ứng với dữ liệu đặc thù của tổ chức giúp chatbot cung cấp thông tin phù hợp, cá nhân hóa theo từng ngữ cảnh người dùng.

Chatbot RAG (Retrieval-Augmented Generation) là một trong những giải pháp tiên tiến trong lĩnh vực chatbot tích hợp Big Data. Thay vì chỉ dựa vào mô hình ngôn ngữ lớn để sinh phản hồi, RAG kết hợp khả năng **truy xuất thông tin (retrieval)** từ cơ sở dữ liệu lớn cùng khả năng sinh văn bản của LLMs, đảm bảo rằng các câu trả lời không chỉ thông minh mà còn chính xác và dựa trên dữ liệu thực tế.

Từ những điều trên, nhóm chúng em đã lựa chọn đề tài: “Elastic Chatbot RAG App” kết hợp Elastic Search, Langchain, LLMs.

Cụ thể, các mục tiêu bao gồm:

1. **Tích hợp với Elastic Search**: Đảm bảo khả năng truy xuất dữ liệu riêng tư một cách an toàn và hiệu quả, đáp ứng các yêu cầu về bảo mật dữ liệu.
2. **Ứng dụng Big Data**: Sử dụng lượng lớn dữ liệu để cung cấp các câu trả lời có giá trị, đồng thời đảm bảo hệ thống có khả năng mở rộng và thích ứng với các yêu cầu mới.
3. **Tăng cường trải nghiệm người dùng**: Đảm bảo rằng chatbot có khả năng duy trì các cuộc hội thoại tự nhiên, mượt mà và cá nhân hóa, mang lại trải nghiệm tối ưu cho người dùng trong các lĩnh vực khác nhau.

2. TỔNG QUAN VỀ DỮ LIỆU LỚN

2.1. Định nghĩa

Big Data là thuật ngữ dùng để chỉ một lượng dữ liệu khổng lồ, phức tạp mà các phương pháp xử lý dữ liệu truyền thống không thể xử lý hiệu quả. Big Data không chỉ bao gồm dữ liệu có cấu trúc (structured data) mà còn dữ liệu phi cấu trúc (unstructured data) và dữ liệu bán cấu trúc (semi-structured data). Những dữ liệu này thường đến từ nhiều nguồn khác nhau, với tốc độ sinh ra nhanh chóng.
Big Data thường được mô tả bởi 5 đặc điểm chính:

2.1.1. Đặc trưng của Big Data (5Vs)

- **Volume (Khối lượng):** Dữ liệu có khối lượng rất lớn, từ terabytes đến petabytes, có thể lên đến exabytes.
- **Velocity (Tốc độ):** Dữ liệu được tạo ra với tốc độ rất nhanh, ví dụ: dòng dữ liệu từ mạng xã hội, cảm biến IoT, hoặc giao dịch tài chính.
- **Variety (Đa dạng):** Dữ liệu có nhiều dạng khác nhau, bao gồm văn bản, hình ảnh, video, âm thanh, v.v.
- **Veracity (Tính chính xác):** Dữ liệu có thể chứa lỗi, thông tin sai lệch hoặc chưa được xác minh, đòi hỏi quá trình làm sạch và xác thực.
- **Value (Giá trị):** Giá trị tiềm năng mà Big Data mang lại khi được phân tích và khai thác hiệu quả.

2.1.2 Nguồn gốc của Big Data

Big Data đến từ nhiều nguồn khác nhau, bao gồm:

- **Giao dịch trực tuyến:** Dữ liệu từ các hoạt động thương mại điện tử, ngân hàng.
- **Truyền thông xã hội:** Nội dung từ Facebook, Twitter, Instagram, TikTok, v.v.
- **IoT (Internet of Things):** Dữ liệu từ các cảm biến, thiết bị thông minh, camera.
- **Dữ liệu y tế:** Hồ sơ bệnh án điện tử, hình ảnh y khoa, dữ liệu di truyền.
- **Dữ liệu khoa học:** Thông tin từ các nghiên cứu khoa học, mô hình thời tiết, khảo sát không gian.

2.2. Large Language Models (LLMs)

Mô hình ngôn ngữ lớn (Large Language Models - LLMs) là một đột phá trong lĩnh vực trí tuệ nhân tạo (AI), được thiết kế để hiểu, phân tích và tạo ra ngôn ngữ tự nhiên một cách gần gũi và chính xác như con người. Khái niệm "ngôn ngữ tự nhiên" ở đây bao gồm mọi hình thức giao tiếp bằng văn bản hoặc lời nói mà con người sử dụng hàng ngày, từ các ngôn ngữ phổ biến như tiếng Anh, tiếng Việt, đến các ngôn ngữ ít phổ biến hơn.

LLMs là kết quả của nhiều thập kỷ nghiên cứu về xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) và học sâu (Deep Learning). Các mô hình này được xây dựng trên cơ sở kiến trúc Transformer - một cấu trúc mạng nơ-ron tiên tiến đã chứng minh tính hiệu quả vượt trội trong việc xử lý các chuỗi dữ liệu phức tạp. Điểm nổi bật của các LLM không chỉ nằm ở khả năng "học" từ lượng dữ liệu khổng lồ mà còn ở khả năng "suy diễn" hoặc "sáng tạo" thông tin dựa trên ngữ cảnh đã được cung cấp.

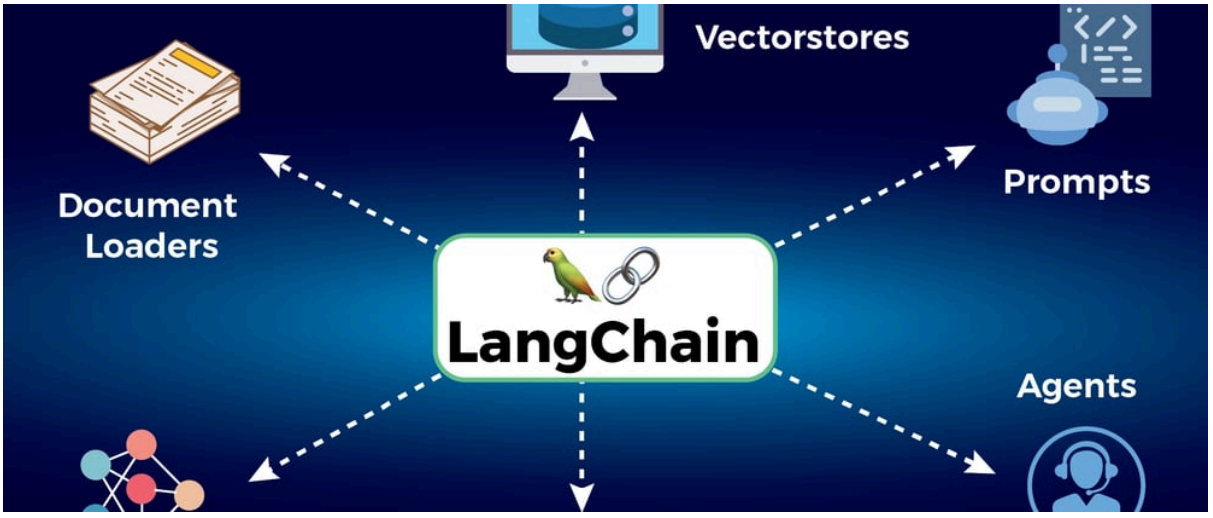
Quá trình phát triển các LLM đã trải qua nhiều giai đoạn tiến hóa. Ban đầu, các hệ thống xử lý ngôn ngữ chỉ có thể thực hiện các nhiệm vụ cơ bản như phân loại văn bản hoặc phân tích cảm xúc. Tuy nhiên, với sự ra đời của các mô hình như BERT (Bidirectional Encoder Representations from Transformers) của Google và GPT (Generative Pre-trained Transformer) của OpenAI, khả năng của các hệ thống ngôn ngữ đã được nâng lên một tầm cao mới. Các mô hình hiện đại như GPT-4 có thể tham gia vào các cuộc hội thoại phức tạp, trả lời câu hỏi, viết bài báo, thậm chí tạo ra mã nguồn phần mềm.

LLMs hiện nay không chỉ là công cụ phục vụ nghiên cứu mà còn trở thành một phần không thể thiếu trong nhiều lĩnh vực như chăm sóc khách hàng, y học, giáo dục, và sáng tạo nội dung. Chúng được sử dụng để xây dựng chatbot, trợ lý ảo, hệ thống dịch thuật tự động, và các công cụ viết lách hỗ trợ. Đặc biệt, các LLM còn có khả năng học hỏi từ các lĩnh vực chuyên môn sâu, giúp giải quyết các vấn đề phức tạp mà trước đây chỉ có con người mới có thể thực hiện.

Tuy nhiên, việc phát triển và ứng dụng LLM không phải là không có thách thức. Những mô hình này yêu cầu tài nguyên tính toán lớn để huấn luyện và vận hành, dẫn đến chi phí cao. Đồng thời, chúng cũng đối mặt với các vấn đề về đạo đức, chẳng hạn như việc tạo ra nội dung không phù hợp hoặc phản ánh các thiên kiến có trong dữ liệu huấn luyện. Ngoài ra, các mô hình ngôn ngữ lớn có thể bị khai thác để thực hiện các hành vi có hại như lan truyền thông tin sai lệch hoặc tấn công mạng.

Trong bối cảnh AI đang trở thành yếu tố cốt lõi định hình tương lai của công nghệ, LLMs không chỉ đại diện cho sức mạnh của AI trong việc xử lý và sáng tạo ngôn ngữ mà còn đặt ra các câu hỏi quan trọng về trách nhiệm, sự minh bạch và tính bền vững trong việc ứng dụng công nghệ này. Sự phát triển của các LLM không chỉ phản ánh tiềm năng to lớn của trí tuệ nhân tạo mà còn là một cơ hội để xã hội con người định hình cách tiếp cận công nghệ một cách có trách nhiệm và sáng tạo.

2.3. LangChain



2.3.1. Tổng quan

LangChain là một framework được thiết kế để hỗ trợ phát triển các ứng dụng tích hợp trí tuệ nhân tạo (AI) mạnh mẽ, tập trung vào việc sử dụng các mô hình ngôn ngữ lớn (LLMs). Mục tiêu chính của LangChain là giúp các nhà phát triển dễ dàng xây dựng các ứng dụng AI

phức tạp bằng cách kết hợp khả năng xử lý ngôn ngữ tự nhiên của LLMs với các nguồn dữ liệu bên ngoài và các tác vụ logic.

LangChain nổi bật nhờ khả năng:

- Xây dựng luồng xử lý hội thoại động.
- Kết nối với cơ sở dữ liệu, API, hoặc các nguồn dữ liệu bên ngoài.
- Tích hợp các chức năng nâng cao như lập kế hoạch, truy vấn, và quản lý bộ nhớ ngữ cảnh.

2.3.2 Các thành phần chính

LangChain bao gồm các thành phần chính giúp xây dựng ứng dụng AI mạnh mẽ và linh hoạt. **Mô hình ngôn ngữ lớn (LLMs)** là trung tâm, hỗ trợ nhiều mô hình như OpenAI GPT, Cohere hay Hugging Face, giúp tùy biến ứng dụng theo nhu cầu. **Prompt Templates** tạo ra các mẫu nhắc tối ưu, đảm bảo mô hình phản hồi chính xác. **Chains (chuỗi xử lý)** kết hợp các tác vụ thành quy trình liền mạch, trong khi **Agents (tác nhân)** có khả năng tự ra quyết định và sử dụng công cụ để hoàn thành nhiệm vụ. **Bộ nhớ (Memory)** lưu trữ ngữ cảnh hội thoại, giúp AI xử lý tốt hơn trong các tương tác dài hạn. Cuối cùng, **Data Connectors** hỗ trợ kết nối với cơ sở dữ liệu, tệp văn bản, hoặc API bên ngoài, đảm bảo AI có thể truy cập và sử dụng thông tin cần thiết theo thời gian thực. Sự phối hợp của các thành phần này làm cho LangChain trở thành công cụ hiệu quả trong việc phát triển các ứng dụng AI phức tạp.

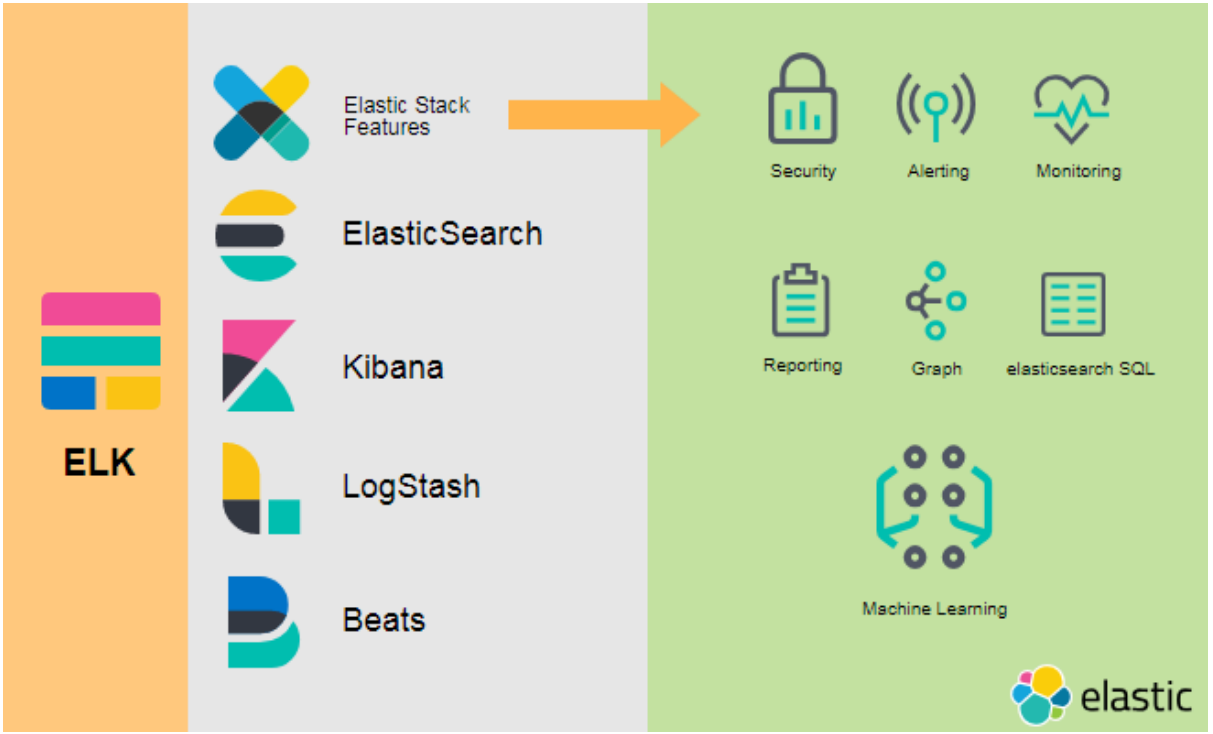
2.3.3 Ứng dụng của LangChain

LangChain có nhiều ứng dụng đa dạng trong các lĩnh vực khác nhau nhờ vào khả năng kết hợp mạnh mẽ giữa mô hình ngôn ngữ và các nguồn dữ liệu bên ngoài. Trong **chatbots thông minh**, LangChain giúp tạo ra các hệ thống hội thoại linh hoạt, có khả năng phản hồi theo ngữ cảnh và hỗ trợ khách hàng hiệu quả. Nó cũng có thể được sử dụng để **truy vấn dữ liệu và phân tích**, khi kết nối với các cơ sở dữ liệu để trả lời câu hỏi hoặc phân tích thông tin phức tạp. Một ứng dụng phổ biến khác là trong các **hệ thống khuyến nghị**, khi LangChain sử dụng dữ liệu người dùng để đưa ra gợi ý về sản phẩm, dịch vụ phù hợp. Ngoài ra, LangChain cũng hỗ trợ **học thuật và nghiên cứu**, cho phép tìm kiếm, tóm tắt và phân tích tài liệu từ các nguồn thông tin khác nhau, giúp người dùng nhanh chóng tiếp cận những kiến thức mới. Những ứng dụng này cho thấy tiềm năng lớn của LangChain trong việc giải quyết các bài toán AI thực tiễn.

2.3.4 Ưu điểm của LangChain

LangChain có nhiều **ưu điểm nổi bật** giúp nó trở thành công cụ lý tưởng cho các ứng dụng AI. Đầu tiên, nó rất **linh hoạt**, hỗ trợ tích hợp nhiều mô hình ngôn ngữ lớn khác nhau và có thể dễ dàng tùy chỉnh để phù hợp với yêu cầu của từng ứng dụng cụ thể. Thứ hai, LangChain mang lại **khả năng mở rộng** mạnh mẽ, cho phép kết nối và tích hợp với nhiều hệ thống và nguồn dữ liệu bên ngoài, giúp tạo ra các ứng dụng AI phức tạp. Bên cạnh đó, tính năng **quản lý ngữ cảnh thông minh** của LangChain giúp bộ nhớ lưu trữ và xử lý các cuộc hội thoại dài, nâng cao hiệu quả tương tác. Ngoài ra, LangChain còn được hỗ trợ bởi một **cộng đồng mã nguồn mở mạnh mẽ**, luôn cập nhật và phát triển nhanh chóng, giúp các nhà phát triển dễ dàng tiếp cận, học hỏi và đóng góp. Những ưu điểm này làm cho LangChain trở thành một công cụ mạnh mẽ và hiệu quả trong việc xây dựng các ứng dụng AI tiên tiến.

2.4. Elastic Search

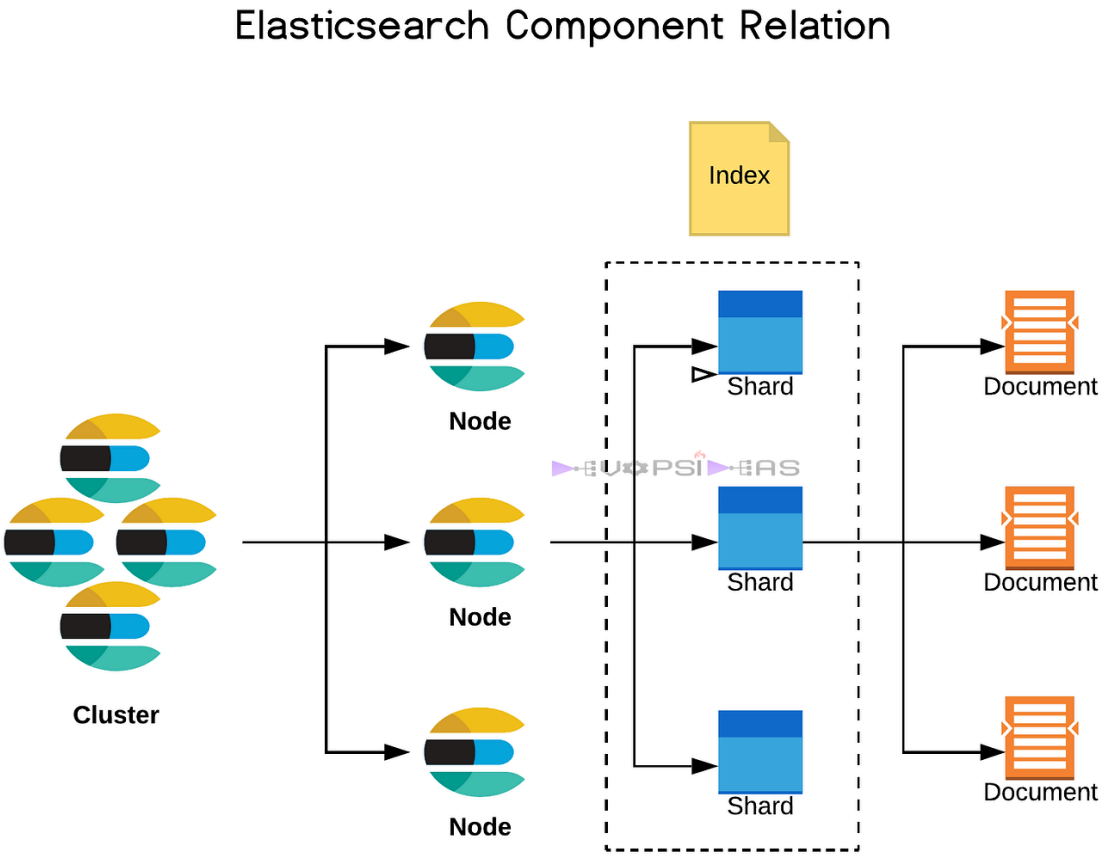


2.4.1 Tổng quan

Elasticsearch là một công cụ tìm kiếm mã nguồn mở mạnh mẽ, được xây dựng trên nền tảng Apache Lucene, sử dụng để tìm kiếm và phân tích dữ liệu với tốc độ rất cao. Nó thường được sử dụng trong các hệ thống phân tích dữ liệu lớn, log và các ứng dụng cần tìm kiếm nhanh chóng và mạnh mẽ.

Elasticsearch là phần cốt lõi của Elastic Stack (trước đây gọi là ELK Stack), bao gồm các công cụ như Logstash (xử lý dữ liệu) và Kibana (hiển thị dữ liệu). Elasticsearch có thể được sử dụng độc lập hoặc kết hợp với các công cụ khác trong Elastic Stack để cung cấp một hệ thống tìm kiếm và phân tích toàn diện.

2.4.2 Kiến trúc của Elastic Search



Node (Nút): Một node trong Elasticsearch là một máy chủ đơn lẻ. Mỗi node lưu trữ một phần dữ liệu và có thể tham gia vào việc xử lý tìm kiếm, phân tích dữ liệu. Các node này kết hợp với nhau để tạo thành một cluster.
Cluster (Cụm): Một cluster là một tập hợp các node hoạt động cùng nhau. Mỗi cluster có một tên duy nhất và chứa tất cả dữ liệu của hệ thống. Việc phân chia dữ liệu thành nhiều node giúp Elasticsearch có thể mở rộng quy mô một cách linh hoạt.
Index (Chỉ mục): Trong Elasticsearch, dữ liệu được tổ chức theo chỉ mục (index). Một chỉ mục là một tập hợp các tài liệu (documents) cùng loại. Mỗi chỉ mục có thể bao gồm nhiều shard.
Document (Tài liệu): Một document là một bản ghi dữ liệu, tương tự như một hàng trong cơ sở dữ liệu quan hệ, nhưng ở định dạng JSON. Mỗi document là một đơn vị cơ bản của dữ liệu trong Elasticsearch.
Shard (Mảnh dữ liệu): Mỗi chỉ mục có thể được chia thành nhiều shard. Các shard giúp phân phối dữ liệu và tải công việc tìm kiếm cho các node khác nhau trong cluster. Elasticsearch hỗ trợ shard động, giúp mở rộng dễ dàng.
Replica (Bản sao): Elasticsearch cho phép bạn tạo bản sao của các shard để đảm bảo tính khả dụng và chống lại sự cố. Bản sao này có thể xử lý các yêu cầu tìm kiếm, giúp tăng cường hiệu suất.

2.4.3 Cấu trúc dữ liệu trong Elastic Search

Mapping (Ánh xạ): Mapping trong Elasticsearch giống như sơ đồ của cơ sở dữ liệu quan hệ. Nó xác định cách các trường dữ liệu sẽ được lưu trữ và xử lý trong các chỉ mục. Mapping có thể bao gồm các kiểu dữ liệu (text, integer, boolean...), các chỉ mục con, và các cấu trúc dữ liệu phức tạp khác.

Full-text Search (Tìm kiếm toàn văn): Elasticsearch sử dụng các thuật toán tìm kiếm toàn văn, như TF-IDF và BM25, để tính toán độ liên quan của các tài liệu. Quá trình này giúp đảm bảo rằng kết quả tìm kiếm trả về là phù hợp với từ khóa của người dùng.

Analyzers (Bộ phân tích): Elasticsearch sử dụng các bộ phân tích để xử lý văn bản trước khi lưu trữ. Các bộ phân tích này có thể bao gồm các bộ tokenizer và filter để cắt và chuẩn hóa dữ liệu văn bản, ví dụ như loại bỏ dấu câu, chuyển thành chữ thường, hoặc tách từ.

Query DSL (Ngôn ngữ truy vấn): Elasticsearch cung cấp một ngôn ngữ truy vấn mạnh mẽ gọi là Query DSL. Ngôn ngữ này cho phép người dùng tạo các truy vấn phức tạp, kết hợp nhiều điều kiện tìm kiếm, lọc và phân tích dữ liệu.

2.4.4 Tính năng nổi bật của Elasticsearch

Tốc độ và hiệu suất: Elasticsearch có khả năng tìm kiếm và phân tích dữ liệu cực nhanh, thậm chí đối với các tập dữ liệu khổng lồ. Việc phân chia dữ liệu và sử dụng các bản sao giúp cải thiện tốc độ truy vấn và tính khả dụng của hệ thống.

Khả năng mở rộng: Elasticsearch được thiết kế để mở rộng một cách dễ dàng, với khả năng thêm nhiều node vào cluster mà không làm gián đoạn dịch vụ.

Tìm kiếm theo cấu trúc và không cấu trúc: Elasticsearch hỗ trợ cả tìm kiếm dữ liệu có cấu trúc (như các bảng cơ sở dữ liệu) và dữ liệu không cấu trúc (như văn bản, logs).

Tìm kiếm gần đúng và phân tích nâng cao: Elasticsearch hỗ trợ tìm kiếm gần đúng với khả năng phát hiện các lỗi chính tả, tìm kiếm tự động, cũng như các tính năng phân tích dữ liệu mạnh mẽ như phân tích từ khoá, tìm kiếm phức tạp, và thống kê.

Hỗ trợ phân tích dữ liệu thời gian thực: Elasticsearch cho phép phân tích và hiển thị dữ liệu thời gian thực, đặc biệt hữu ích trong việc giám sát hệ thống và phân tích logs.

2.4.5 Ứng dụng thực tế của Elasticsearch

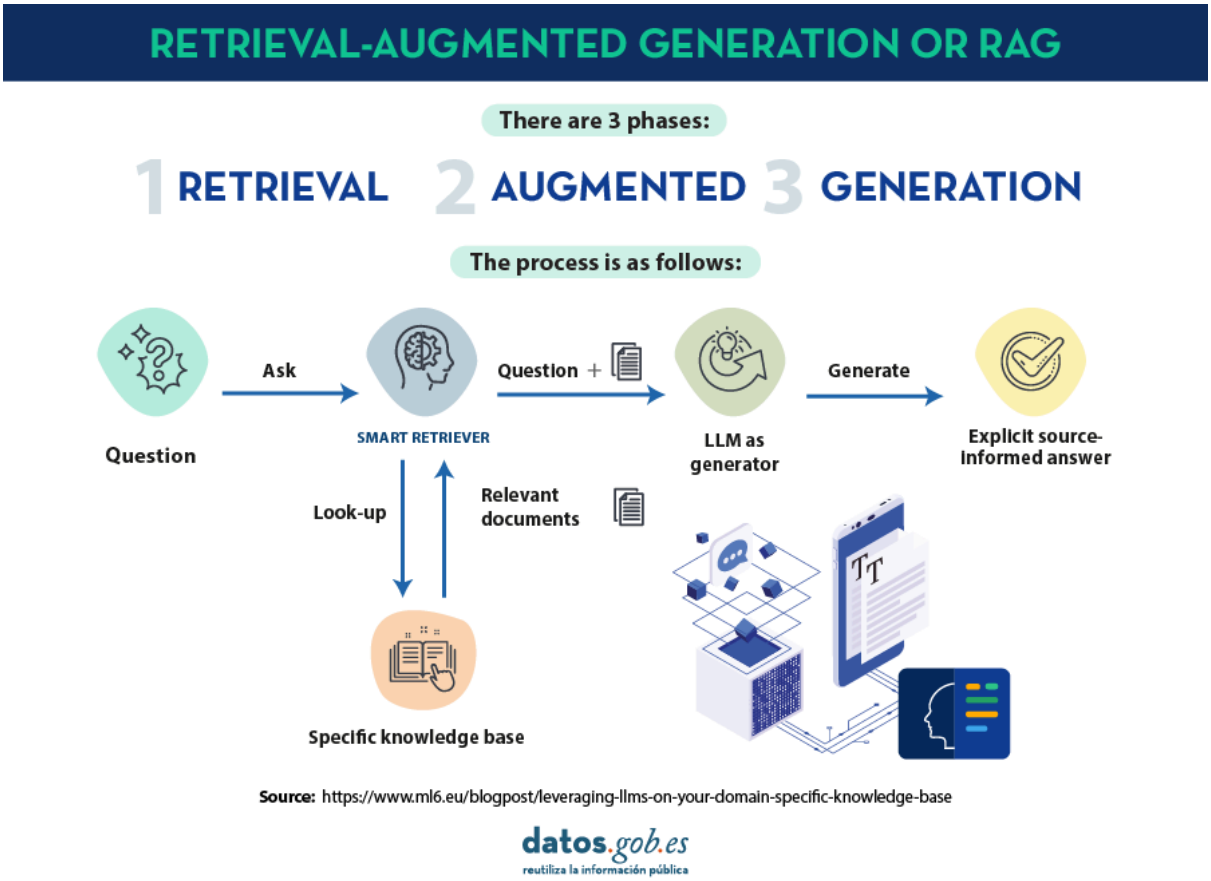
Tìm kiếm website: Elasticsearch thường được sử dụng trong các hệ thống tìm kiếm của website để cung cấp kết quả nhanh chóng và chính xác từ hàng triệu dữ liệu.

Phân tích và giám sát logs: Elasticsearch là công cụ phổ biến trong các hệ thống giám sát và phân tích logs, giúp tìm kiếm và phân tích nhanh chóng các dữ liệu log từ các hệ thống phân tán.

Tìm kiếm trong E-commerce: Các nền tảng thương mại điện tử sử dụng Elasticsearch để cải thiện trải nghiệm tìm kiếm của người dùng, giúp tìm ra sản phẩm phù hợp dựa trên các từ khoá tìm kiếm và các tiêu chí lọc khác.

Phân tích dữ liệu lớn: Với khả năng xử lý dữ liệu lớn, Elasticsearch là công cụ lý tưởng để phân tích và trực quan hóa dữ liệu từ các nguồn khác nhau, đặc biệt là khi kết hợp với Kibana.

2.5. RAG



RAG (Retrieval-Augmented Generation) là một phương pháp học máy kết hợp giữa việc truy xuất thông tin và khả năng sinh tạo văn bản, nhằm cải thiện hiệu quả của các mô hình ngôn ngữ. Trong đó, mô hình này sử dụng dữ liệu ngoài (thường là từ một kho dữ liệu lớn) để truy vấn và tìm kiếm thông tin có liên quan, sau đó sử dụng thông tin đó để tạo ra các câu trả lời hoặc nội dung phù hợp. Mô hình RAG có thể giúp cải thiện khả năng hiểu biết và tạo nội dung của mô hình ngôn ngữ, đặc biệt trong các tác vụ yêu cầu sự chính xác cao và sự hiểu biết sâu về ngữ cảnh.

3. XÂY DỰNG CHATBOT

3.1. Thiết kế hệ thống

Hệ thống chatbot được thiết kế dựa trên mô hình RAG (Retrieval-Augmented Generation), một cách tiếp cận hiện đại kết hợp khả năng truy xuất thông tin (retrieval) với mô hình sinh ngôn ngữ (generation). Điều này giúp chatbot không chỉ có khả năng trả lời dựa trên tri thức từ mô hình mà còn cung cấp thông tin chính xác từ cơ sở dữ liệu tùy chỉnh. Dưới đây là mô tả chi tiết về từng thành phần trong kiến trúc hệ thống.

3.1.1. Tiền Xử Lý Dữ Liệu

Tải và Xử Lý Dữ Liệu: Hệ thống hỗ trợ tải và xử lý dữ liệu từ nhiều nguồn, chẳng hạn như tệp văn bản hoặc cơ sở dữ liệu trực tuyến:

- Tải tài liệu: Sử dụng `TextLoader` từ thư viện LangChain để nạp các tài liệu từ các tệp `.txt`.
- Tách văn bản thành đoạn nhỏ: Các đoạn văn bản (chunk) được tạo bằng cách sử dụng các bộ chia văn bản như:
    - `CharacterTextSplitter`: Chia văn bản theo số ký tự cố định.
    - `RecursiveCharacterTextSplitter`: Một công cụ mạnh mẽ hơn, chia văn bản dựa trên các quy tắc lồng nhau để duy trì sự mạch lạc trong ngữ cảnh.

Tạo Vector Embedding: Dữ liệu sau khi được chia thành các đoạn nhỏ sẽ được biểu diễn dưới dạng vector embedding bằng các mô hình ngôn ngữ:

- Embedding cho tiếng Anh: Sử dụng mô hình từ thư viện HuggingFace, chẳng hạn `BAAI/bge-small-en`, để chuyển các đoạn văn bản tiếng Anh thành vector embedding.
- Embedding cho tiếng Việt:
    - Hệ thống sử dụng lớp `VietnameseEmbeddings`, được kế thừa từ `HuggingFaceEmbeddings`.
    - Văn bản tiếng Việt được tokenize trước bởi thư viện Pyvi, sau đó được chuyển đổi thành embedding bằng các mô hình như `dangvantuan/vietnamese-document-embedding`.

Lưu ý: Với dữ liệu song ngữ, việc chọn mô hình embedding phù hợp cho từng ngôn ngữ đảm bảo chất lượng truy xuất thông tin tốt hơn.

3.1.2. Hệ Thống Truy Xuất Thông Tin (Retriever)

Truy xuất thông tin đóng vai trò quan trọng trong mô hình RAG, giúp cung cấp ngữ cảnh phù hợp từ kho dữ liệu. Chatbot sử dụng hai hệ thống chính để thực hiện việc này:

- FAISS (Vector Search)
    - FAISS (Facebook AI Similarity Search) là một thư viện mã nguồn mở tối ưu cho việc tìm kiếm ngữ nghĩa dựa trên vector.
    - Cách hoạt động:
        - Các đoạn văn bản đã được chuyển thành vector embedding sẽ được lưu trữ trong FAISS.
        - Khi nhận được truy vấn từ người dùng, FAISS thực hiện phép so sánh vector để tìm ra các đoạn văn bản tương đồng nhất.

- Elasticsearch
    - Elasticsearch được sử dụng như một giải pháp tìm kiếm mạnh mẽ, hỗ trợ truy vấn dữ liệu lớn với tốc độ cao.
    - Lợi ích:

        - Tích hợp tốt với dữ liệu phức tạp, hỗ trợ truy vấn vector embedding.
        - Khả năng mở rộng và quản lý dữ liệu hiệu quả trong các ứng dụng thực tế.

Quy Trình Truy Xuất:

Khi nhận truy vấn từ người dùng, hệ thống sẽ chuyển truy vấn thành vector embedding.

Dựa trên độ tương đồng ngữ nghĩa (cosine similarity), các đoạn văn bản phù hợp nhất sẽ được tìm kiếm và trả về.

Các đoạn văn bản này được sử dụng làm ngữ cảnh cho mô hình sinh ngôn ngữ.

3.1.3 Mô Hình Sinh (Generator)

Lựa Chọn Mô Hình Ngôn Ngữ: Hệ thống tích hợp với các mô hình LLM mạnh mẽ, được triển khai qua OllamaLLM, một API hỗ trợ giao tiếp với nhiều mô hình:

- Các mô hình tích hợp:
    - `llama2`: Mô hình mạnh mẽ cho ngữ cảnh tổng quát.
    - `phi3:3.8b`: Mô hình tối ưu cho truy vấn phức tạp.
    - `mrjacktung/phogpt-4b-chat-gguf`: Mô hình tùy chỉnh cho ngôn ngữ cụ thể.
    - `Tuanpham/t-visstar-7b:latest`: Mô hình chuyên biệt cho tiếng Việt.

Luồng Sinh Phản Hồi

- Các đoạn văn bản truy xuất được chuyển đến mô hình sinh.
- Prompt tùy chỉnh được tạo để định hướng mô hình sinh câu trả lời dựa trên ngữ cảnh.
- Mô hình sinh phản hồi dựa trên thông tin truy xuất và câu hỏi của người dùng.

Luồng Xử Lý Tổng Quát

Nhập câu hỏi: Người dùng nhập câu hỏi thông qua giao diện Gradio.

Truy xuất ngữ cảnh: Truy vấn được chuyển thành vector embedding và so sánh với kho dữ liệu để lấy ngữ cảnh liên quan.

Sinh câu trả lời: Mô hình sinh sẽ dựa trên ngữ cảnh truy xuất và truy vấn để tạo phản hồi.

Hiển thị phản hồi: Kết quả được hiển thị trực tiếp qua giao diện trò chuyện.

Prompt Tùy Chỉnh

Dành cho lĩnh vực Y tế (Tiếng Anh)

> You are a doctor that helps to answer questions based on the context provided.
> The context is information from documents related to healthcare topics.
> Answer the following question based on the context below.
> If you do not know the answer, simply say 'I don't know'.
>
> Context:
> {context}
>
> Question: {question}
> Answer:

Dành cho lĩnh vực Âm nhạc (Tiếng Việt):

> Bạn là một chuyên gia âm nhạc giúp trả lời các câu hỏi dựa trên ngữ cảnh được cung cấp.
> Ngữ cảnh là thông tin về các ban nhạc rock, lịch sử của họ, các album và buổi biểu diễn.
> Hãy trả lời câu hỏi dưới đây dựa trên ngữ cảnh.
> Nếu bạn không biết câu trả lời, chỉ cần nói 'Tôi không biết'.
>
> Ngữ cảnh:
> {context}
>
> Câu hỏi: {question}
> Câu trả lời:

3.1.4. Giao Diện Người Dùng

Chatbot được triển khai qua Gradio, cung cấp giao diện thân thiện và dễ sử dụng:

- Chức năng:
  - Cho phép người dùng nhập câu hỏi.
  - Hiển thị câu trả lời với lịch sử hội thoại.
- Ví dụ mẫu:
  - Âm nhạc:
    - "Ai là tay guitar chính của ban nhạc Ngũ Cung?"
    - "Ban nhạc Bức Tường phát hành album nào gần đây nhất?"
  - Y tế:
    - "What are the symptoms of diabetes?"
    - "How can I prevent heart disease?"

- Vai trò của các thành phần: LangChain, ELSER, và các LLMs.

3.2. Triển khai

Việc tích hợp các mô hình ngôn ngữ lớn (LLMs) và xử lý dữ liệu thông qua ELSER (Embedding-based Language Search Representation) nhằm mục tiêu tối ưu hóa khả năng truy xuất ngữ nghĩa và sinh ngôn ngữ dựa trên dữ liệu cụ thể của người dùng. Quy trình này bao gồm nhiều bước từ chuẩn bị dữ liệu, tạo vector embedding, đến kết hợp với LLM để sinh câu trả lời. Dưới đây là chi tiết từng bước:

3.2.1. Chuẩn Bị Dữ Liệu

Thu thập và tải dữ liệu:

Nguồn dữ liệu: Các tài liệu đầu vào có thể đến từ nhiều nguồn, như tệp `.txt`, cơ sở dữ liệu nội bộ, hoặc API trực tuyến.

Định dạng dữ liệu: Tất cả dữ liệu được chuyển đổi về định dạng văn bản thô hoặc JSON để xử lý.

Phân đoạn dữ liệu

Chia nhỏ văn bản (Chunking):Sử dụng `CharacterTextSplitter` hoặc `RecursiveCharacterTextSplitter` để chia các tài liệu lớn thành các đoạn nhỏ (chunk).Kích thước mỗi chunk được cấu hình (ví dụ: 500 ký tự) để đảm bảo sự mạch lạc của ngữ cảnh.

Thêm siêu dữ liệu:Gắn kèm các thông tin như nguồn gốc tài liệu, ngày tạo, hoặc các nhãn liên quan để hỗ trợ quá trình truy xuất.

Làm sạch dữ liệu: Xử lý dữ liệu thô, loại bỏ các ký tự không cần thiết, và chuẩn hóa văn bản để đảm bảo chất lượng khi tạo embedding.

3.2.2. Tạo Vector Embedding Qua ELSER

Lợi ích: Khả năng biểu diễn ý nghĩa sâu sắc của văn bản, tối ưu cho truy vấn ngữ nghĩa trong các hệ thống tìm kiếm.

Quy trình tạo embedding

Chọn mô hình ELSER:

Sử dụng ELSER từ ElasticSearch để tạo vector embedding.

Cấu hình API để tích hợp với hệ thống.

Tiền xử lý văn bản:

Với tiếng Việt: Văn bản được tokenize bằng Pyvi trước khi chuyển đổi.

Với tiếng Anh: Sử dụng các kỹ thuật tiền xử lý tiêu chuẩn (xóa stopword, chuẩn hóa).

Sinh embedding:

Chuyển từng chunk văn bản thành vector embedding bằng API của ELSER.

Embedding được lưu trữ dưới dạng vector để sử dụng trong các bước truy vấn.

3.2.3. Lưu Trữ và Quản Lý Dữ Liệu Embedding

Lưu trữ embedding

Elasticsearch: Embedding được index và lưu trữ trong Elasticsearch để hỗ trợ truy vấn vector. Sử dụng các kỹ thuật tìm kiếm vector như k-NN (k-Nearest Neighbors) để tối ưu hóa việc tìm ngữ cảnh phù hợp.

FAISS (Tùy chọn): Một công cụ khác có thể được sử dụng để lưu trữ và tìm kiếm embedding dựa trên độ tương đồng.

Tổ chức dữ liệu: Tất cả các chunk văn bản và vector embedding đều được lưu trữ kèm siêu dữ liệu, giúp việc truy vấn không chỉ dựa trên ngữ nghĩa mà còn dựa vào thông tin cụ thể của tài liệu.

3.2.4. Kết Hợp với LLM để Sinh Câu Trả Lời

Chọn mô hình ngôn ngữ lớn (LLM): `llama2` (general-purpose), `phi3:3.8b` (phù hợp với câu hỏi phức tạp), `mrjacktung/phogpt-4b-chat-gguf` (dành cho tiếng Việt).

Quy trình tích hợp LLM

Nhận truy vấn từ người dùng: Người dùng nhập câu hỏi thông qua giao diện. Truy vấn được tiền xử lý và chuyển thành vector embedding.

Truy xuất ngữ cảnh với ELSER: Embedding của truy vấn được so sánh với các vector trong Elasticsearch hoặc FAISS để lấy ra các đoạn văn bản liên quan. Kết quả được tập hợp và sắp xếp dựa trên độ tương đồng.

Sinh câu trả lời: Các đoạn văn bản truy xuất được kết hợp làm ngữ cảnh trong prompt của LLM. Prompt được định nghĩa rõ ràng, yêu cầu mô hình trả lời dựa trên ngữ cảnh hoặc từ chối trả lời nếu không có thông tin.

3.2.5. Triển Khai Truyền Dữ Liệu Qua Pipeline

Tạo pipeline tích hợp

Pipeline gồm ba giai đoạn chính:

Retriever:Lấy thông tin phù hợp từ Elasticsearch/FAISS.

Generator: Sử dụng LLM để tạo câu trả lời từ ngữ cảnh.

Output Handler: Trả về câu trả lời cho người dùng qua giao diện Gradio.

Triển khai với Gradio

Giao diện Gradio được tích hợp để:

Nhận truy vấn từ người dùng.

Hiển thị kết quả trả lời từ LLM.

Duy trì lịch sử hội thoại để tiện theo dõi.

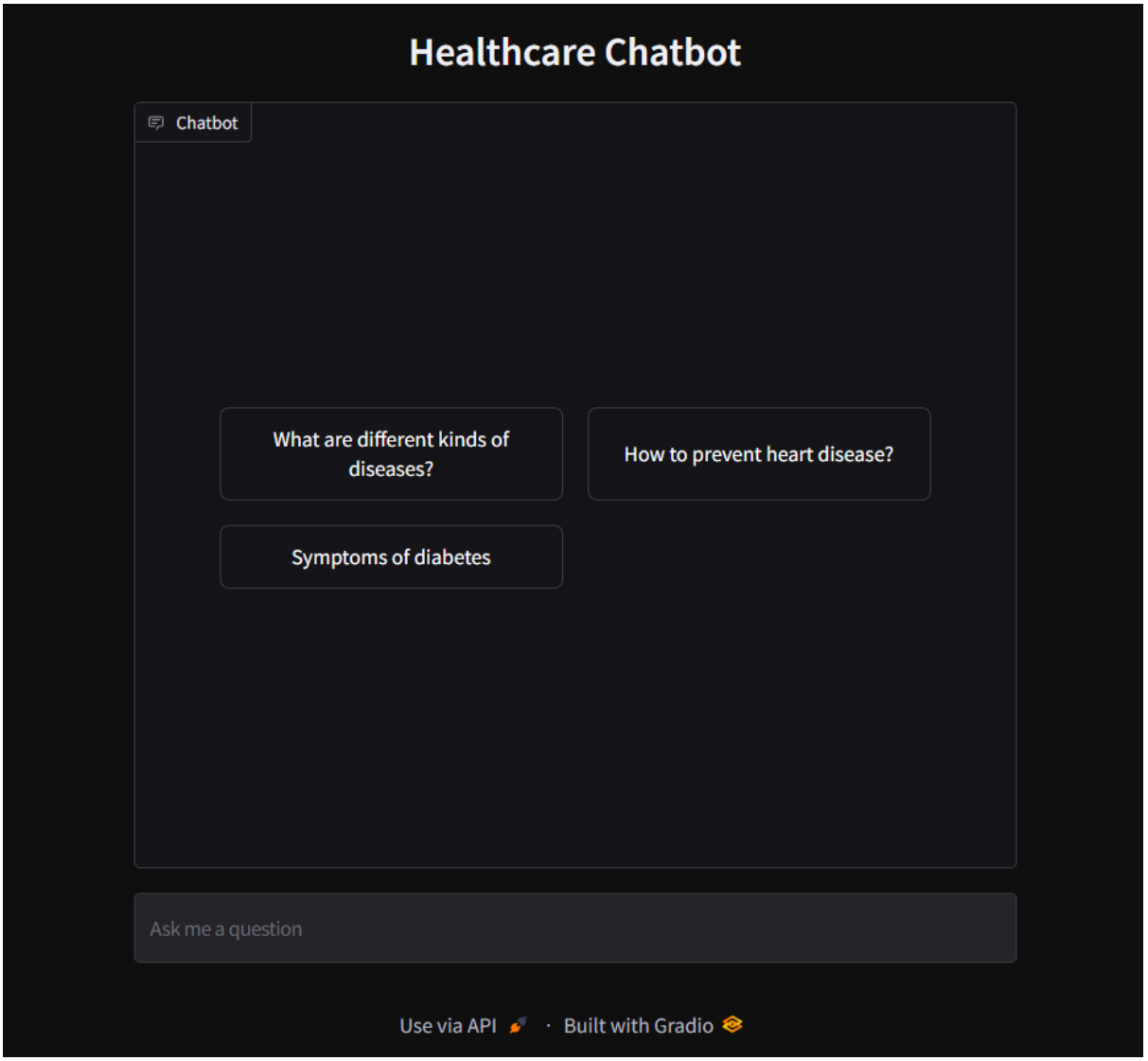
3.2.6. Tối Ưu Hóa Hệ Thống

Hiệu quả truy xuất Sử dụng chỉ số cosine similarity để đánh giá độ tương đồng giữa truy vấn và vector tài liệu. Điều chỉnh kích thước chunk và siêu dữ liệu để cải thiện chất lượng ngữ cảnh.

Tối ưu hiệu suất LLM Điều chỉnh prompt để định hướng mô hình sinh phản hồi ngắn gọn, chính xác. Giảm chi phí tính toán bằng cách sử dụng các mô hình nhỏ hơn (như `paraphrase-MiniLM-L6-v2`) cho bước kiểm tra tính liên quan.

4. DEMO VÀ KẾT QUẢ

Giao diện người dùng



5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết Luận

Với sự phát triển nhanh chóng của trí tuệ nhân tạo và dữ liệu lớn, các công cụ như LangChain, FAISS, Elasticsearch, và các mô hình ngôn ngữ lớn (LLMs) đang trở nên quan trọng trong việc xây dựng các hệ thống truy vấn và tương tác thông minh. Dự án của nhóm em đã thành công trong việc áp dụng kỹ thuật **RAG (Retrieval-Augmented Generation)** để xây dựng một chatbot thông minh, có khả năng kết hợp truy xuất thông tin từ kho dữ liệu riêng với khả năng sinh ngôn ngữ mạnh mẽ của LLMs.

Nhóm em đã đạt được một số kết quả bao gồm:

- Hiểu biết tổng quan về công nghệ LangChain và cách tích hợp LLMs.

- Hiểu rõ cách thức hoạt động của FAISS và Elasticsearch trong việc tìm kiếm ngữ nghĩa.
- Nắm vững chi tiết về mô hình RAG và cách triển khai trong các bài toán thực tế.
- Xây dựng và triển khai thành công một chatbot kết hợp truy xuất và sinh ngôn ngữ.
- Đánh giá và phân tích hiệu quả của chatbot trong việc trả lời các câu hỏi phức tạp.

Tuy nhiên, dự án vẫn còn một số hạn chế:

- Dữ liệu sử dụng còn hạn chế về số lượng và tính đa dạng, dẫn đến một số câu trả lời chưa đạt độ chính xác cao.
- Việc tối ưu hóa hiệu suất và chi phí khi sử dụng LLMs vẫn cần được cải thiện.
- Chưa tích hợp các kỹ thuật tiên tiến như xử lý ngữ cảnh đa bước hoặc các chiến lược học sâu hơn để tăng cường hiệu quả của hệ thống.

Dự án đã đạt được mục tiêu ban đầu là xây dựng một chatbot RAG cơ bản, nhưng vẫn còn nhiều cơ hội phát triển để cải thiện tính chính xác và ứng dụng trong thực tế. Nhóm em hy vọng tiếp tục cải tiến hệ thống, mở rộng dữ liệu, và thử nghiệm trên các bài toán khác nhau để đạt được kết quả tốt hơn trong tương lai.

5.2. Hướng phát triển

Để nâng cao hiệu quả của chatbot, cần tăng cường số lượng và tính đa dạng của dữ liệu bằng cách thu thập thông tin từ nhiều lĩnh vực khác nhau như giáo dục, y tế và kinh doanh, kết hợp với dữ liệu phi cấu trúc như hình ảnh, video, và âm thanh. Việc cải thiện chất lượng dữ liệu thông qua làm sạch tự động và chuẩn hóa dữ liệu đa ngôn ngữ giúp chatbot hoạt động chính xác trong các ngữ cảnh song ngữ. Sử dụng các mô hình ngôn ngữ tiên tiến như GPT-4 hoặc LLaMA 3 sẽ cải thiện khả năng xử lý ngôn ngữ, trong khi các mô hình nhỏ hơn như MiniLM giúp tối ưu tài nguyên hệ thống. Bên cạnh đó, kết hợp FAISS và HNSW sẽ cải thiện tốc độ tìm kiếm, đồng thời sử dụng Elasticsearch nâng cao hiệu quả truy vấn. Hệ thống duy trì ngữ cảnh lâu dài giúp chatbot xử lý các chuỗi câu hỏi liên tiếp một cách tự nhiên. Ngoài ra, việc phát triển các chatbot chuyên dụng cho từng ngành nghề như chăm sóc khách hàng, y tế và giáo dục, cùng với khả năng tích hợp vào hệ thống CRM, ERP, sẽ giúp tự động hóa quy trình công việc. Cuối cùng, chatbot cần hỗ trợ đa ngôn ngữ.

TÀI LIỆU THAM KHẢO

Phân chia Công Việc

Phạm Đức Hoàng	
Hoàng Ngọc Hào	