TRƯỜNG ĐẠI HỌC BÀ RỊA – VŨNG TÀU

KHOA KỸ THUẬT – CÔNG NGHỆ

BARIA VUNGTAU
UNIVERSITY
CAP SAINT JACQUES

ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG ỨNG DỤNG CHATBOT TRUY XUẤT THÔNG TIN SỬ DỤNG KỸ THUẬT RAG

Trình độ đào tạo: Đại học chính quy

Ngành: Công nghệ thông tin

Chuyên ngành: Lập trình Ứng dụng di động & Game

Giảng viên hướng dẫn: TS. Lê Thị Vĩnh Thanh

Sinh viên thực hiện: Lê Quốc Khánh

Mã số sinh viên: 19034508

Thành phố Vũng Tàu, ngày 01 tháng 04 năm 2024

LỜI NÓI ĐẦU

Trí tuệ nhân tạo và công nghệ thông tin đang ngày càng phát triển và đóng vai trò quan trọng trong nhiều lĩnh vực của cuộc sống. Trong xu thế đó, các hệ thống ChatBot đã trở thành một công cụ hữu ích, giúp tự động hóa và nâng cao hiệu quả trong giao tiếp và hỗ trợ người dùng. Đồ án tốt nghiệp của tôi với đề tài "**Xây dựng ứng dụng chatbot truy xuất thông tin sử dụng kỹ thuật rag**" được thực hiện nhằm nghiên cứu và phát triển một hệ thống ChatBot thông minh, hỗ trợ hướng dẫn người dùng có thể trò chuyện với Chatbot bằng dữ liệu riêng của mình.

Với sự hỗ trợ của công nghệ các mô hình ngôn ngữ lớn (LLM) và kĩ thuật truy vấn RAG, kết hợp cả hai cùng với LangChain, ChatBot RAG không chỉ đơn thuần là một công cụ trả lời câu hỏi mà còn có khả năng học hỏi từ các dữ liệu riêng và người dùng tải lên để cải thiện chất lượng truy vấn. Trong suốt quá trình thực hiện đồ án, tôi đã đối mặt với nhiều thử thách, từ việc phân tích và xử lý ngôn ngữ tự nhiên, thiết kế giao diện người dùng, đến việc tích hợp các thuật RAG trong học máy để nâng cao khả năng tương tác của ChatBot.

Đồ án này không chỉ là kết quả của quá trình học tập và nghiên cứu, mà còn là minh chứng cho sự nỗ lực và đam mê của tôi trong lĩnh vực công nghệ thông tin. Tôi hy vọng rằng sản phẩm này sẽ góp phần vào việc phát triển các ứng dụng ChatBot thông minh, phục vụ tốt hơn nhu cầu của người dùng và mang lại những giá trị thiết thực cho xã hội.

LỜI CẢM ƠN

Trước hết, tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Ban Giám hiệu, các thầy cô giáo và toàn thể cán bộ, nhân viên Trường Đại học Bà Rịa – Vũng Tàu đã tạo điều kiện tốt nhất cho tôi trong suốt quá trình học tập và nghiên cứu. Những kiến thức và kinh nghiệm mà tôi đã thu nhận được tại đây là nền tảng vững chắc giúp tôi hoàn thành đồ án này.

Tôi xin bày tỏ lòng biết ơn sâu sắc đến giảng viên hướng dẫn đồ án tốt nghiệp – TS. Lê Thị Vĩnh Thanh. Cô đã tận tâm chỉ bảo và hỗ trợ tôi trong suốt quá trình nghiên cứu và hoàn thiện đồ án. Sự tận tâm và kiên nhẫn của cô đã cho tôi động lực cũng như cơ hội để có thể hoàn thành đồ án này.

Tôi cũng xin cảm ơn quý thầy cô trong Khoa Công nghệ thông tin, những người đã luôn nhiệt tình giảng dạy và chia sẻ những kiến thức quý báu, giúp tôi mở rộng tầm nhìn và nâng cao kỹ năng trong suốt thời gian học tập tại trường.

Mặc dù đã cố gắng rất nhiều, nhưng đồ án vẫn không tránh khỏi những thiếu sót. Rất mong nhận được sự thông cảm, chỉ dẫn và góp ý của quý thầy cô và các bạn để đồ án được hoàn thiện hơn.

Xin chân thành cảm ơn!

Thành phố Vũng Tàu, ngày 10 tháng 06 năm 2024

Sinh viên thực hiện

Lê Quốc Khánh

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thành phố Vũng Tàu, ngày … tháng … năm 202…

Giảng viên xác nhận

MỤC LỤC

MỤC LỤC HÌNH ẢNH

CHƯƠNG 1. GIỚI THIỆU

1.1. Lý do chọn đề tài

1.1.1. Nhu cầu truy xuất thông tin hiệu quả trong thời đại số

Trong thời đại công nghệ bùng nổ như hiện nay, việc tiếp cận và xử lý thông tin trở thành nhu cầu thiết yếu cho mọi cá nhân và tổ chức. Tuy nhiên, với lượng thông tin khổng lồ và ngày càng gia tăng trên internet, việc tìm kiếm và thu thập thông tin phù hợp, chính xác một cách nhanh chóng và hiệu quả trở nên vô cùng khó khăn. Việc dành nhiều thời gian để sàng lọc qua vô số nguồn thông tin không chỉ tốn kém chi phí mà còn ảnh hưởng đến hiệu suất công việc và chất lượng cuộc sống..

1.1.2. Giải pháp Chatbot ứng dụng Large Language Model

Nhằm giải quyết vấn đề nan giải này, việc phát triển ứng dụng Chatbot có khả năng truy xuất thông tin từ nhiều nguồn dữ liệu khác nhau thông qua Large Language Model (LLM) được xem là giải pháp tối ưu. Chatbot đóng vai trò như một trợ lý thông minh, giúp người dùng tiết kiệm thời gian và công sức trong việc tìm kiếm thông tin, đồng thời nâng cao trải nghiệm truy cập thông tin một cách hiệu quả..

1.1.3. Lựa chọn Chatbot RAG

Chatbot đóng vai trò quan trọng trong việc tương tác với người dùng một cách tự nhiên và hiệu quả. Trong dự án này, tôi lựa chọn xây dựng Chatbot dựa trên mô hình Retrieval-Augmented Generation (RAG) để truy xuất thông tin và tạo câu trả lời phản hồi cho người dùng. Mô hình RAG sở hữu những ưu điểm vượt trội so với các mô hình Chatbot truyền thống, bao gồm:

- Khả năng truy xuất thông tin đa dạng: RAG có thể truy cập thông tin từ nhiều nguồn dữ liệu khác nhau, bao gồm văn bản, hình ảnh, video, v.v., giúp cung cấp cho người dùng một bức tranh toàn diện về chủ đề tìm kiếm.
- Khả năng tạo câu trả lời tự nhiên: RAG sở hữu khả năng xử lý ngôn ngữ tự nhiên tiên tiến, giúp tạo ra những câu trả lời trôi chảy, dễ hiểu và phù hợp với ngữ cảnh giao tiếp.

- Khả năng học hỏi và thích ứng: RAG có thể học hỏi từ các tương tác với người dùng và điều chỉnh hành vi của mình theo thời gian, giúp nâng cao hiệu quả hoạt động và đáp ứng tốt hơn nhu cầu của người dùng.

1.1.4. Lý do lựa chọn

Lý do tôi lựa chọn đề tài này xuất phát từ hai yếu tố chính:

- **Sự gia tăng dữ liệu trực tuyến:** Lượng thông tin trên internet ngày càng tăng theo cấp số nhân, dẫn đến nhu cầu truy cập thông tin hiệu quả hơn. Các cá nhân và tổ chức cần có công cụ hỗ trợ để xử lý và khai thác lượng thông tin khổng lồ này một cách nhanh chóng và chính xác.
- **Nhu cầu tìm kiếm thông tin của người dùng:** Nhu cầu tìm kiếm thông tin ngày càng cao, đặc biệt là thông tin chính xác, tin cậy và phù hợp với nhu cầu cá nhân. Chatbot RAG có thể đáp ứng nhu cầu này bằng cách cung cấp cho người dùng thông tin phù hợp, chính xác và nhanh chóng.

1.2. Mục tiêu và phạm vi của đề tài

Mục tiêu của dự án này là xây dựng một chatbot RAG có khả năng truy xuất thông tin từ các tài liệu PDF và tạo ra các câu trả lời phản hồi tự nhiên và chính xác. Chatbot sẽ có khả năng hiểu và đáp ứng các câu hỏi của người dùng một cách linh hoạt và thông minh.

Phạm Vi: Đề tài sẽ tập trung vào việc phát triển các chức năng cốt lõi của Chatbot, bao gồm hiểu và phản hồi các câu hỏi của người dùng, cung cấp thông tin hữu ích và tương tác linh hoạt với người dùng. Đề tài không bao gồm việc xây dựng nguồn dữ liệu mà sẽ tập trung vào việc trích xuất thông tin từ các nguồn dữ liệu đã có.

1.3. Tầm quan trọng của Chatbot trong thời đại số

Nhu cầu ngày càng tăng về truy cập thông tin:

- Trong thời đại số, lượng thông tin được tạo ra và lưu trữ ngày càng tăng theo cấp số nhân.
- Nhu cầu truy cập thông tin nhanh chóng, chính xác và hiệu quả trở nên thiết yếu hơn bao giờ hết.

- Tuy nhiên, việc tìm kiếm thông tin trong kho dữ liệu khổng lồ, đặc biệt là dữ liệu PDF, có thể tốn thời gian và gặp nhiều khó khăn.

Ưu điểm của Chatbot truy xuất thông tin từ dữ liệu PDF:

- **Tiết kiệm thời gian và công sức:** Chatbot có thể tự động hóa quy trình tìm kiếm thông tin, giúp người dùng tiết kiệm thời gian và công sức.
- **Tăng hiệu quả truy cập thông tin:** Chatbot có khả năng truy cập và xử lý thông tin từ nhiều nguồn dữ liệu PDF khác nhau một cách nhanh chóng và hiệu quả.
- **Cải thiện trải nghiệm người dùng:** Chatbot cung cấp giao diện tương tác đơn giản, dễ sử dụng, giúp người dùng dễ dàng truy cập thông tin cần thiết.
- **Nâng cao khả năng truy cập thông tin:** Chatbot giúp người dùng dễ dàng truy cập thông tin, bất kể trình độ kỹ thuật hay khả năng ngôn ngữ của họ.

Ứng dụng của Chatbot truy xuất thông tin từ dữ liệu PDF:

- **Dịch vụ khách hàng:** Chatbot có thể hỗ trợ khách hàng tìm kiếm thông tin về sản phẩm, dịch vụ, giải đáp thắc mắc và hỗ trợ giải quyết vấn đề.
- **Giáo dục:** Chatbot có thể hỗ trợ học tập cho sinh viên, cung cấp tài liệu học tập, giải đáp thắc mắc và hướng dẫn thực hành.
- **Y tế:** Chatbot có thể cung cấp thông tin y tế cho bệnh nhân, hỗ trợ chẩn đoán và điều trị bệnh.
- **Thương mại điện tử:** Chatbot có thể tư vấn sản phẩm, hỗ trợ khách hàng trong quá trình mua sắm và thanh toán.
- **Nghiên cứu khoa học:** Chatbot có thể hỗ trợ các nhà nghiên cứu tìm kiếm thông tin khoa học, phân tích dữ liệu và tổng hợp kết quả nghiên cứu.

Thách thức và giải pháp:

- **Xử lý dữ liệu PDF:** Chatbot cần có khả năng xử lý chính xác thông tin từ các tệp PDF, bao gồm cả việc trích xuất văn bản, phân loại dữ liệu và xác định mối quan hệ giữa các thông tin.
- **Hiểu ngôn ngữ tự nhiên:** Chatbot cần có khả năng hiểu ngôn ngữ tự nhiên của người dùng để có thể truy xuất thông tin chính xác và cung cấp câu trả lời phù hợp.

- **Bảo mật thông tin:** Chatbot cần đảm bảo an toàn cho dữ liệu của người dùng và tuân thủ các quy định về bảo mật thông tin.

Kết luận:

Xây dựng Chatbot truy xuất thông tin từ dữ liệu PDF là một giải pháp hiệu quả cho nhu cầu truy cập thông tin ngày càng tăng trong thời đại số. Chatbot có thể giúp người dùng tiết kiệm thời gian, công sức và nâng cao hiệu quả truy cập thông tin. Với những ứng dụng rộng rãi và tiềm năng phát triển to lớn, Chatbot truy xuất thông tin từ dữ liệu PDF hứa hẹn sẽ đóng vai trò quan trọng trong tương lai.

.

CHƯƠNG 2. KIẾN THỨC CƠ SỞ

2.1. Large Language Model(LLM) và Chatbot:

2.1.1. Large Language Model (LLM):

Là một loại mô hình máy học mạnh mẽ được huấn luyện trên các tập dữ liệu văn bản lớn để có khả năng tạo ra văn bản tự nhiên giống như con người. LLM hoạt động bằng cách kết hợp ba yếu tố chính: dữ liệu, kiến trúc mạng nơ-ron, và quá trình huấn luyện.

- Dữ liệu: LLM được huấn luyện trên các tập dữ liệu văn bản lớn, bao gồm sách, bài báo, và cuộc trò chuyện. Các mô hình này có khả năng xử lý hàng tỷ hoặc thậm chí là petabytes dữ liệu văn bản.
- Kiến trúc mạng nơ-ron: Kiến trúc chính của LLM là **Transformer**, một loại mạng nơ-ron được thiết kế để hiểu ngữ cảnh và mối quan hệ giữa các từ trong câu. Các mạng transformer giúp mô hình hiểu cấu trúc câu và ý nghĩa của các từ trong đó.
- Quá trình huấn luyện: Trong quá trình huấn luyện, mô hình học cách dự đoán từ tiếp theo trong một câu. Ví dụ, nếu câu là "The sky is...", mô hình sẽ cố gắng dự đoán từ tiếp theo, chẳng hạn như "blue". Qua mỗi vòng lặp, mô hình điều chỉnh các tham số nội tại của mình để làm giảm sai khác giữa dự đoán và kết quả thực tế. Quá trình này tiếp tục cho đến khi mô hình có thể tạo ra các câu hoàn chỉnh và có ý nghĩa.

LLMs có nhiều ứng dụng trong, bao gồm:

- Dịch vụ khách hàng: Sử dụng để tạo ra các chatbot thông minh có khả năng xử lý các câu hỏi của khách hàng, giải phóng nhân viên để xử lý các vấn đề phức tạp hơn.
- Tạo nội dung: Hỗ trợ việc tạo ra bài viết, email, bài đăng trên mạng xã hội và kịch bản video YouTube.
- Phát triển phần mềm: Có thể sử dụng để tạo và xem xét mã nguồn, giúp tăng cường hiệu suất và chất lượng trong quá trình phát triển phần mềm.

Như vậy, LLMs không chỉ là công cụ mạnh mẽ trong việc tạo ra văn bản tự nhiên mà còn có nhiều ứng dụng tiềm năng trong nhiều lĩnh vực doanh nghiệp khác nhau.

2.1.2. Chatbot

Là một chương trình máy tính mô phỏng cuộc trò chuyện với con người bằng cách sử dụng trí tuệ nhân tạo (AI) và xử lý ngôn ngữ tự nhiên (NLP). Chatbot có thể được tìm thấy trên nhiều nền tảng khác nhau như website, ứng dụng nhắn tin, mạng xã hội, v.v. Chúng có thể được sử dụng cho nhiều mục đích khác nhau, bao gồm:

- Dịch vụ khách hàng: Chatbot có thể cung cấp hỗ trợ khách hàng 24/7, trả lời các câu hỏi thường gặp, giải quyết các vấn đề đơn giản và hướng dẫn khách hàng đến các đại lý hỗ trợ con người khi cần thiết.
- Tiếp thị và bán hàng: Chatbot có thể giới thiệu sản phẩm và dịch vụ, thu thập thông tin khách hàng tiềm năng và tạo ra các ưu đãi cá nhân hóa.
- Giáo dục: Chatbot có thể cung cấp tài liệu học tập, giải đáp thắc mắc, hướng dẫn thực hành và đánh giá học tập.
- Giải trí: Chatbot có thể chơi trò chơi, kể chuyện, cung cấp thông tin giải trí và tương tác với người dùng theo cách vui vẻ.

Cách thức hoạt động của Chatbot:

Chatbot hoạt động bằng cách sử dụng các thuật toán AI và NLP để hiểu và phản hồi ngôn ngữ con người. Dưới đây là một số bước cơ bản trong cách hoạt động của chatbot:

1. **Nhận đầu vào:** Chatbot nhận đầu vào từ người dùng, thường dưới dạng tin nhắn văn bản hoặc giọng nói.
2. **Xử lý đầu vào:** Chatbot sử dụng NLP để phân tích đầu vào, xác định ý định của người dùng và trích xuất các thông tin quan trọng.
3. **Tạo phản hồi:** Chatbot sử dụng AI để tạo ra phản hồi phù hợp với ý định của người dùng, sử dụng kiến thức và thông tin được lưu trữ trong cơ sở dữ liệu của nó.
4. **Gửi phản hồi:** Chatbot gửi phản hồi cho người dùng, thường dưới dạng tin nhắn văn bản hoặc giọng nói.

Các loại Chatbot:

Có nhiều loại chatbot khác nhau, được phân loại dựa trên các tiêu chí khác nhau như:

- **Mức độ phức tạp:**
  - Chatbot đơn giản: Chỉ có thể thực hiện các nhiệm vụ đơn giản, thường dựa trên quy tắc.
  - Chatbot phức tạp: Có thể học hỏi và thích ứng với các tình huống mới, sử dụng AI và NLP tiên tiến.
- **Nền tảng:**
  - Chatbot web: Hoạt động trên website.
  - Chatbot ứng dụng nhắn tin: Hoạt động trên các ứng dụng nhắn tin như Facebook Messenger, WhatsApp, v.v.
  - Chatbot giọng nói: Hoạt động bằng giọng nói, sử dụng công nghệ nhận dạng giọng nói.
- **Mục đích sử dụng:**
  - Chatbot dịch vụ khách hàng: Cung cấp hỗ trợ khách hàng.
  - Chatbot tiếp thị: Quảng bá sản phẩm và dịch vụ.
  - Chatbot giáo dục: Hỗ trợ học tập.
  - Chatbot giải trí: Cung cấp giải trí.

Ưu điểm của Chatbot:

Chatbot mang lại nhiều ưu điểm, bao gồm:

- **Khả dụng 24/7:** Chatbot có thể hoạt động 24/7, cung cấp hỗ trợ cho người dùng bất kể thời gian nào.
- **Tương tác cá nhân hóa:** Chatbot có thể tương tác với người dùng theo cách cá nhân hóa, dựa trên thông tin và sở thích của họ.

- **Tăng hiệu quả:** Chatbot có thể tự động hóa các tác vụ đơn giản, giải phóng thời gian cho nhân viên để tập trung vào các công việc quan trọng hơn.
- **Tiết kiệm chi phí:** Chatbot có thể giúp doanh nghiệp tiết kiệm chi phí bằng cách tự động hóa các dịch vụ khách hàng và hỗ trợ bán hàng.
- **Cải thiện trải nghiệm khách hàng:** Chatbot có thể cung cấp trải nghiệm khách hàng tốt hơn bằng cách hỗ trợ nhanh chóng, hiệu quả và cá nhân hóa.

Nhược điểm của Chatbot:

Bên cạnh những ưu điểm, chatbot cũng có một số nhược điểm, bao gồm:

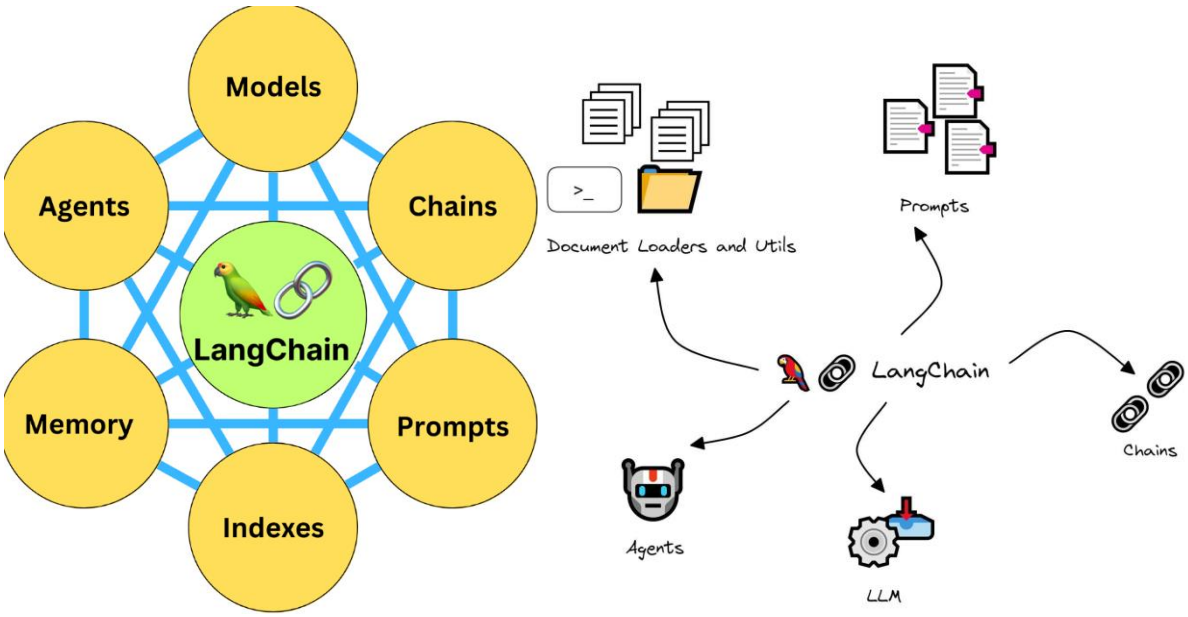
- **Chi phí phát triển:** Việc phát triển một chatbot phức tạp có thể tốn kém.
- **Bảo trì:** Chatbot cần được cập nhật và bảo trì thường xuyên để đảm bảo hoạt động hiệu quả.
- **Hạn chế về ngôn ngữ:** Chatbot có thể gặp khó khăn trong việc hiểu ngôn ngữ tự nhiên phức tạp hoặc các sắc thái ngôn ngữ.
- **Thiếu sự đồng cảm:** Chatbot

2.2. Các công nghệ và công cụ được sử dụng

2.2.1. LangChain

Langchain là gì ?

Là một framework mã nguồn mở được thiết kế để đơn giản hóa việc tạo các ứng dụng bằng các mô hình ngôn ngữ lớn (LLM). Nó cung cấp một bộ công cụ và thư viện giúp các nhà phát triển dễ dàng kết nối LLM với các ứng dụng của họ, xử lý đầu vào và đầu ra của LLM và triển khai các ứng dụng chatbot mạnh mẽ.



Hình 2. 1 Các mảng ứng dụng của LangChain

Dưới đây là một số tính năng chính của LangChain:

- **Hỗ trợ nhiều LLM:** LangChain hỗ trợ nhiều LLM khác nhau, bao gồm GPT-3, Jurassic-1 Jumbo, Megatron-Turing NLG và WuDao 2.0.
- **Giao diện đơn giản:** LangChain cung cấp một giao diện đơn giản và dễ sử dụng, giúp các nhà phát triển dễ dàng bắt đầu với LLM mà không cần kiến thức chuyên sâu về AI hoặc lập trình.
- **Mã hóa mô-đun:** LangChain sử dụng kiến trúc mã hóa mô-đun, cho phép các nhà phát triển xây dựng các ứng dụng phức tạp bằng cách kết hợp các thành phần chức năng khác nhau.
- **Khả năng mở rộng:** LangChain có thể mở rộng để đáp ứng nhu cầu của các ứng dụng lớn, với khả năng xử lý nhiều yêu cầu đồng thời.

LangChain có thể được sử dụng để xây dựng nhiều loại ứng dụng khác nhau, bao gồm:

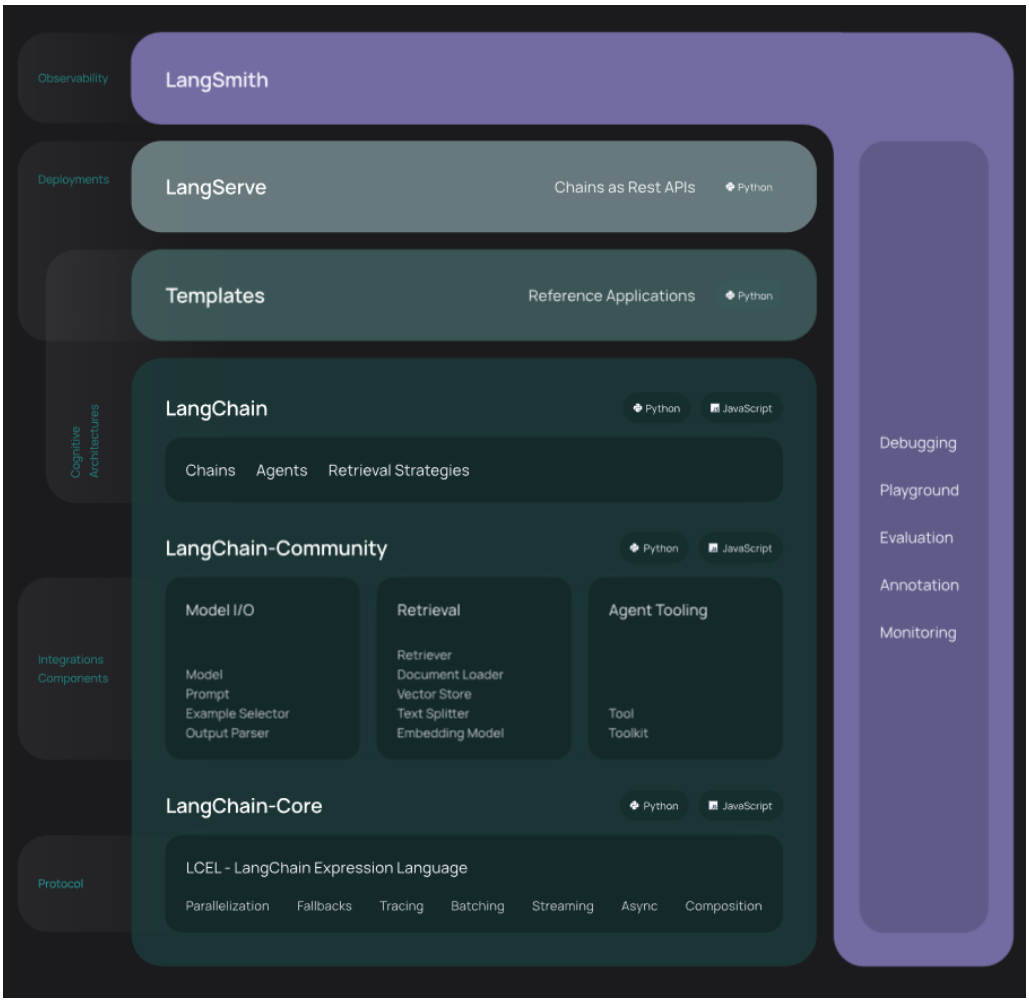
- Chatbots: LangChain có thể được sử dụng để xây dựng chatbot mạnh mẽ có thể hiểu và phản hồi ngôn ngữ tự nhiên một cách hiệu quả.
- Hệ thống trợ lý ảo: LangChain có thể được sử dụng để xây dựng hệ thống trợ lý ảo có thể giúp người dùng thực hiện các tác vụ như đặt lịch hẹn, đặt chỗ và tìm thông tin.

- Công cụ tạo nội dung: LangChain có thể được sử dụng để xây dựng công cụ tạo nội dung tự động, chẳng hạn như công cụ viết bài báo, tạo kịch bản và sáng tác nhạc.
- Hệ thống phân tích dữ liệu: LangChain có thể được sử dụng để xây dựng hệ thống phân tích dữ liệu có thể trích xuất thông tin từ văn bản và dữ liệu ngôn ngữ tự nhiên khác.

LangChain là một công cụ mạnh mẽ và linh hoạt có thể giúp các nhà phát triển xây dựng các ứng dụng LLM sáng tạo và hiệu quả. Nó đang được sử dụng bởi nhiều công ty và tổ chức khác nhau, bao gồm Google, Amazon, Microsoft và Facebook.

Ngoài những thông tin trên, bạn cũng có thể tham khảo thêm các nguồn sau để tìm hiểu thêm về LangChain:

- Trang web chính thức của LangChain: https://www.langchain.com/



Hình 2. 2 LangChain và các thư viện khác

Ứng dụng LangChain trong dự án

Vì sao chọn Langchain:

- **Giai đoạn xử lí, chuẩn bị dữ liệu với LangChain**: **PyPDFLoader** : Để có thể chat với dữ liệu riêng, cần phải định dạng dữ liệu về dạng mà chatbot cần, có thể xử lí. Với việc langchain.document_loader có thể hỗ trợ truy cập và chuyển đổi hơn 80 dạng dữ liệu trong đó có PyPDFLoader được xử dụng trong việc tải dữ liệu dạng PDF, ví dụ:

```
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf")
pages = loader.load()
```

Hình 2. 3 Code PyPDFLoader

RecursiveCharacterTextSplitter: Sau khi load được dữ liệu, chúng ta cần LangChain tách dữ liệu ra thành các vector máy có thể đọc và làm việc với nó. Cụ thể, nó chia các văn bản thành các phần nhỏ dựa trên các ký tự như dấu xuống dòng ("\n\n"), dấu xuống dòng đơn ("\n"), khoảng trắng (" "), hoặc một ký tự rỗng (""). Ví dụ:

```
from langchain.text_splitter import RecursiveCharacterTextSplitter, CharacterTextSplitter\
chunk_size =26
chunk_overlap = 4
r_splitter = RecursiveCharacterTextSplitter(
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap
)
c_splitter = CharacterTextSplitter(
    chunk_size=chunk_size,
    chunk_overlap=chunk_overlap
)
```

Hình 2. 4 Code RecursiveCharacterTextSplitter

Chroma: là một cơ sở dữ liệu vector mã nguồn mở dành cho trí tuệ nhân tạo (AI-native), tập trung vào sự hiệu quả và hạnh phúc của các nhà phát triển. Nó được cấp phép theo Apache 2.0. Sau khi các văn bản trong dữ liệu được chia thành các

vector no sẽ cần nơi để lưu chữ gọi là VectorStore, trong dự án này sử dụng VectorStore của LangChain chính là Chroma, Ví dụ:

```python
# import
from langchain_chroma import Chroma
from langchain_community.document_loaders import TextLoader
from langchain_community.embeddings.sentence_transformer import (
    SentenceTransformerEmbeddings,
)
from langchain_text_splitters import CharacterTextSplitter

# load the document and split it into chunks
loader = TextLoader("../../modules/state_of_the_union.txt")
documents = loader.load()

# split it into chunks
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
docs = text_splitter.split_documents(documents)

# create the open-source embedding function
embedding_function = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")

# load it into Chroma
db = Chroma.from_documents(docs, embedding_function)

# query it
query = "What did the president say about Ketanji Brown Jackson"
docs = db.similarity_search(query)

# print results
print(docs[0].page_content)
```

Hình 2. 5 Code Chroma

Kết luận: LangChain là một công cụ mạnh mẽ hỗ trợ người dùng trong việc tải và lưu trữ dữ liệu một cách linh hoạt, đa dạng và nhanh chóng. Với khả năng xử lý nhiều định dạng dữ liệu khác nhau, LangChain mang đến sự thuận tiện vượt trội cho các nhà phát triển và người dùng. Bạn có thể dễ dàng load và save dữ liệu từ các nguồn khác nhau mà không gặp phải những trở ngại thường thấy. Điều này giúp tiết kiệm thời gian và nâng cao hiệu quả công việc.

Hơn nữa, LangChain còn có khả năng tích hợp với HunggingFace, một nền tảng nổi tiếng về các mô hình ngôn ngữ tự nhiên. Sự liên kết này cho phép người dùng tải các mô hình tiên tiến trực tiếp từ HunggingFace, giúp việc triển khai các ứng dụng AI trở nên đơn giản hơn bao giờ hết. Tính năng này đặc biệt hữu ích trong các dự án cần đến sự xử lý ngôn ngữ tự nhiên, như chatbot, phân tích văn bản, hoặc dịch ngôn ngữ.

LangChain không chỉ dừng lại ở việc hỗ trợ load và save dữ liệu, mà còn mở rộng khả năng của mình thông qua các tích hợp khác, giúp người dùng dễ dàng quản lý và sử dụng dữ liệu một cách hiệu quả. Điều này làm cho LangChain trở thành một công cụ không thể thiếu trong các dự án công nghệ hiện đại, nơi mà dữ liệu và mô hình AI đóng vai trò then chốt.

Với những tính năng ưu việt như vậy, LangChain đóng một vai trò vô cùng quan trọng trong việc phát triển các dự án công nghệ. Nó không chỉ giúp tối ưu hóa quy trình làm việc, mà còn mở ra nhiều cơ hội mới trong việc khai thác và ứng dụng dữ liệu, đặc biệt là trong lĩnh vực trí tuệ nhân tạo và học máy. Nhờ LangChain, việc quản lý và sử dụng dữ liệu trở nên dễ dàng và hiệu quả hơn, góp phần thúc đẩy sự phát triển của các dự án một cách mạnh mẽ.

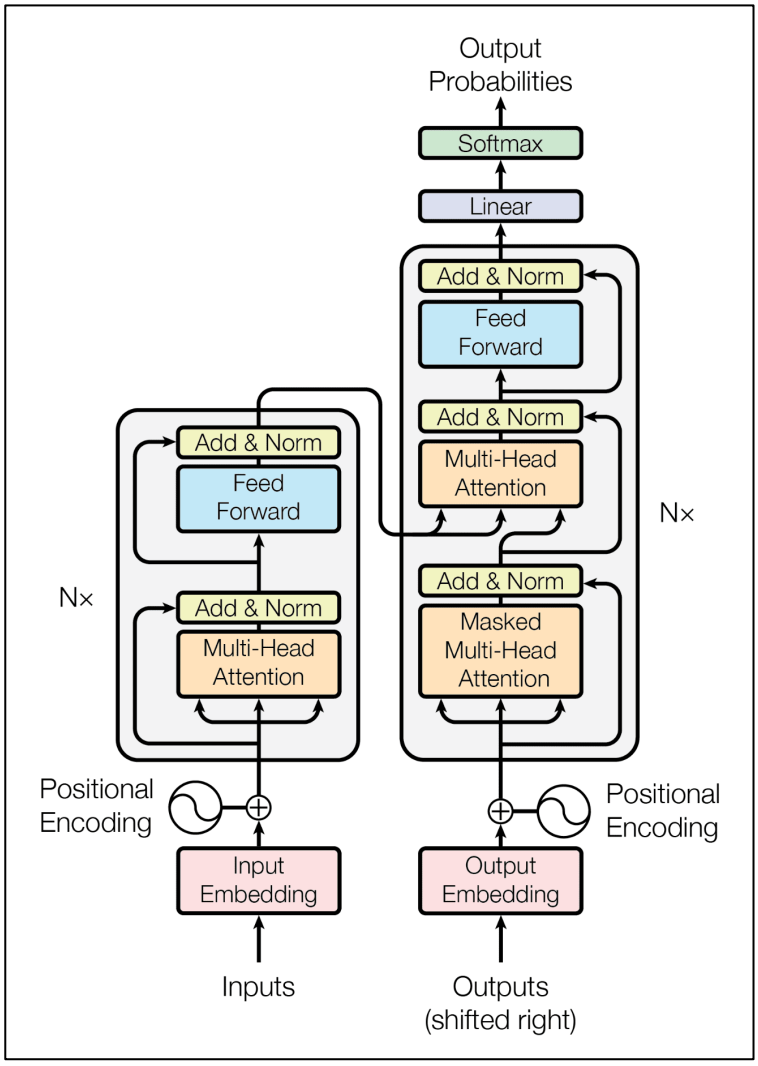
2.2.2. HuggingFace và Transformers

Hình 2. 6 HunggingFace và Transformers

Transformers là gì ?

Hình 2. 7 Transformers

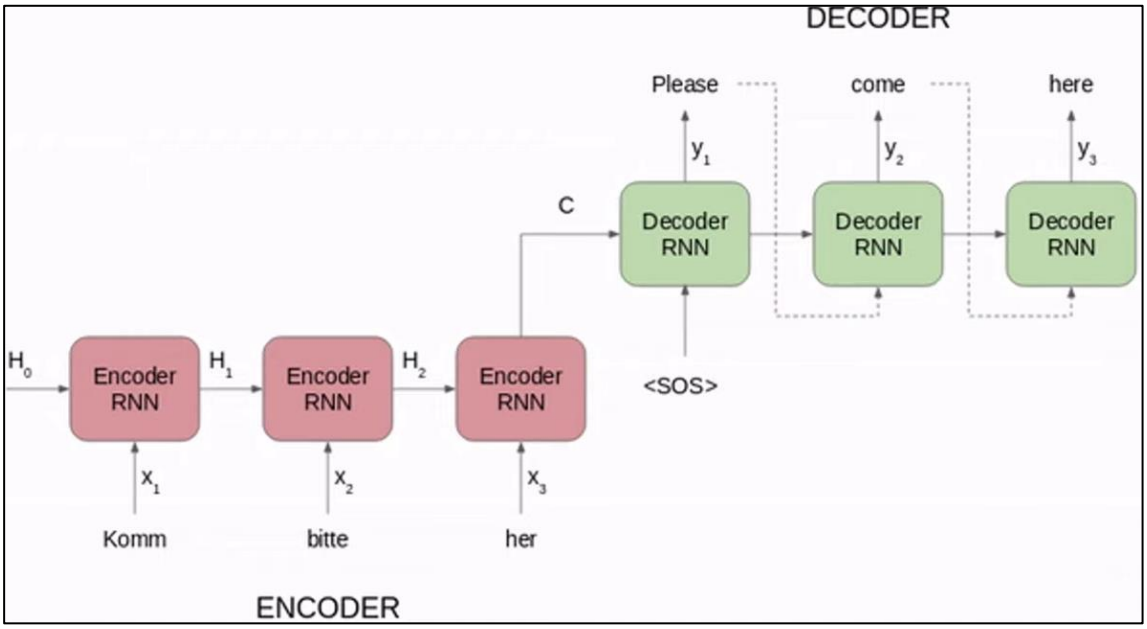
Transformers là một kiến trúc mạng nơ-ron được Google Research giới thiệu vào năm 2017 và nhanh chóng trở thành kiến trúc thống trị cho các tác vụ NLP. Transformers sử dụng cơ chế chú ý (attention mechanism) để học hỏi các mối quan hệ tầm xa giữa các từ trong câu, giúp nâng cao hiệu quả xử lý ngôn ngữ tự nhiên.



Hình 2. 8 Cơ chế hoạt động cùa Transformers

Tìm hiểu về cơ chế hoạt động của Transformers:

1. Có thể thấy trên hình ảnh Transformers vẫn được chia thành 2 khối ENCODER và DECODER như các cúc trúc mạng tiền nhiệm như: RNN,… Ví dụ về RNN:



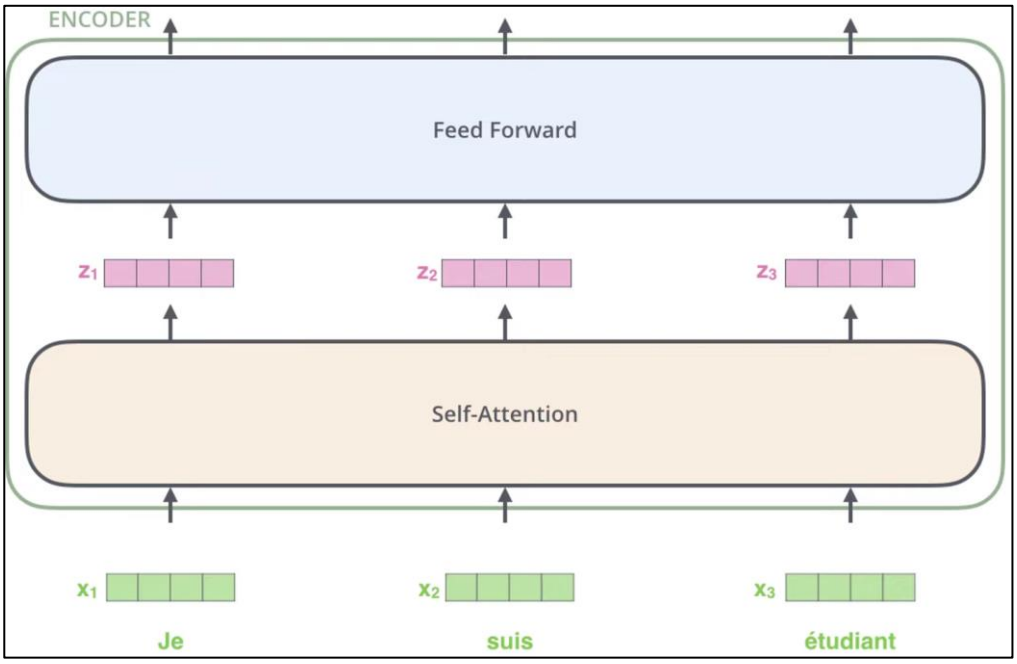
Hình 2. 9 Cấu trức RNN

2. ENCODER của Transformers:

Một khối ENCODER bao gồm hai thành phần chính: Self-Attention và Feed Forward Neural Network. Trong đó, Self-Attention là phần quan trọng nhất, đóng vai trò trụ cột của mô hình Transformers. Self-Attention cho phép mô hình xem xét mối quan hệ giữa các từ trong câu hoặc các phần tử trong chuỗi đầu vào.

Mỗi từ trong câu được biểu diễn bằng một vector, và ma trận trọng số sẽ xác định mức độ quan trọng của mỗi từ đối với các từ khác. Cách tiếp cận này giúp mô hình có thể hiểu ngữ cảnh và cấu trúc của câu một cách toàn diện. Nhờ vào Self-Attention, mô hình có thể tập trung vào các từ quan trọng trong câu, bất kể vị trí của chúng, từ đó cải thiện khả năng nắm bắt ngữ nghĩa và mối liên hệ giữa các từ.

Feed Forward Neural Network là phần còn lại trong khối ENCODER, hoạt động sau khi Self-Attention đã hoàn thành nhiệm vụ của mình. Nó giúp tiếp tục xử lý thông tin và tinh chỉnh đầu ra để tạo ra các biểu diễn ngữ nghĩa tốt hơn cho các từ.



Hình 2. 10 Sơ đồ tổng quan ENCODEE chạy dịch thuật

Self-Attention: Có thể thấy trong sơ đồ Transformers chạy xử lí dữ liệu bằng Parallelization (Song song hóa) thay vì như RNN xử lí theo dạng chuỗi, Transformers làm tất cả các phần của chuỗi đầu vào có thể được xử lý song song, tăng tốc độ huấn luyện.

Feedforward Neural Networks: Sau khi áp dụng self-attention, đầu ra sẽ được truyền qua một mạng nơ-ron đa tầng sâu để tính toán các biến đổi phi tuyến tính và tạo ra đầu ra cuối cùng.

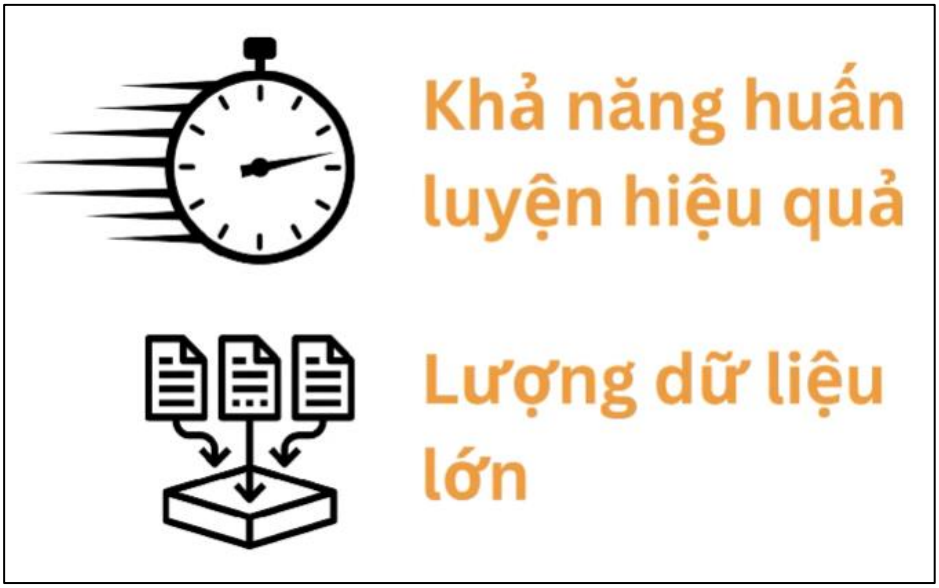
3. DECODER của Transformers:

Sau khi các vector từ ENCODER được đưa vào quá trình DECODER nó sẽ được sinh ra thành các câu trả lời, output của đầu vào từ ENCODER

Maked Multi-Head Attention: Gồm 2 phần, đầu tiên nó cũng bao gồm phần Attention ở ENCODER, và phần Masked Multi để thực hiện việc tạo ra các mask cho câu trả lời, tức là sẽ masking từng chữ trong output và input để tạo ra và dự đoán vector câu trả lời hợp lí sau khi loại bỏ được các từ được cho là không hợp lí ở ENCODER đẩy sang, sau đó đẩy kết quả cuối cùng thông qua bước Feed Forward.

Ưu điểm của Transformers:

- **Khả năng học tập tầm xa:** Transformers có khả năng học hỏi các mối quan hệ tầm xa giữa các từ trong câu, giúp hiểu rõ hơn ngữ cảnh và ý nghĩa của văn bản.
- **Hiệu quả:** Transformers có hiệu quả cao hơn các kiến trúc mạng nơ-ron truyền thống cho các tác vụ NLP.
- **Dễ dàng áp dụng:** Transformers có thể dễ dàng áp dụng cho nhiều tác vụ NLP khác nhau.



Hình 2. 11 Ưu điểm của Transformers

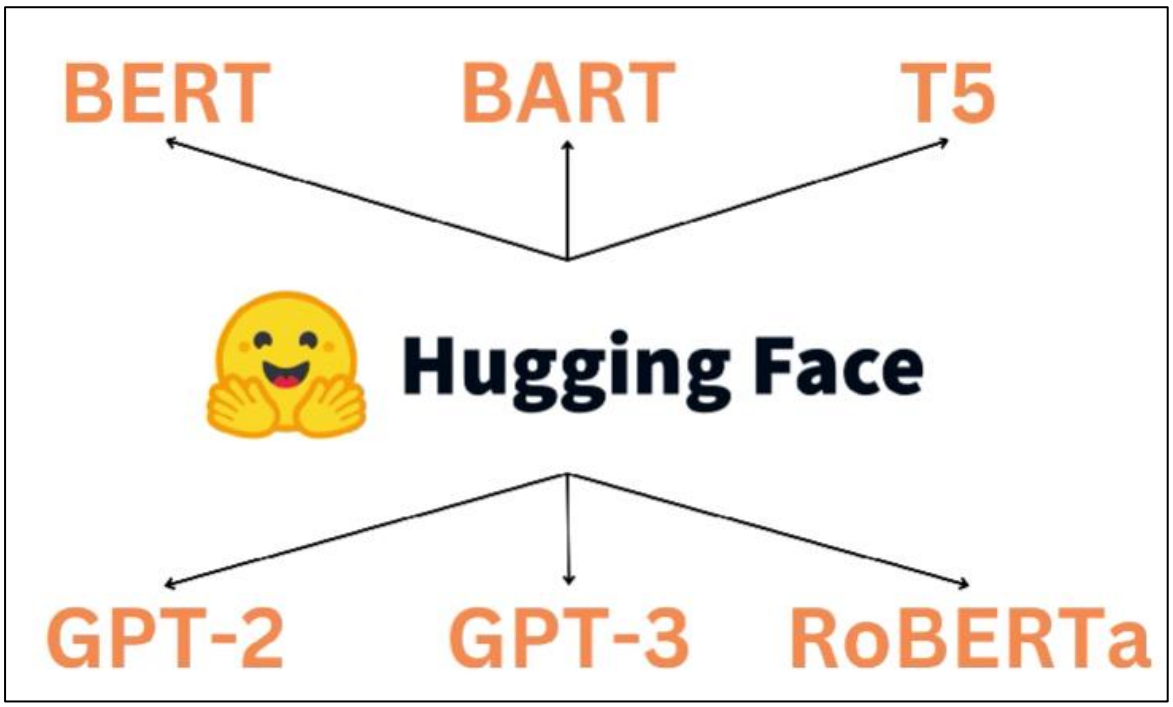
Ứng dụng của Transformers:

- **Phân loại văn bản:** Transformers được sử dụng hiệu quả cho các tác vụ phân loại văn bản như phân loại chủ đề, phân loại tình cảm, v.v.
- **Trích xuất thông tin:** Transformers được sử dụng để trích xuất thông tin từ văn bản, chẳng hạn như tên người, địa điểm, số lượng, v.v.
- **Tóm tắt văn bản:** Transformers được sử dụng để tóm tắt văn bản, tạo ra phiên bản ngắn gọn và súc tích của văn bản gốc.
- **Dịch máy:** Transformers được sử dụng để dịch máy, chuyển đổi văn bản từ ngôn ngữ này sang ngôn ngữ khác.

- **Trả lời câu hỏi:** Transformers được sử dụng để trả lời câu hỏi dựa trên một đoạn văn bản hoặc một tập dữ liệu.

Huggingface là gì ?

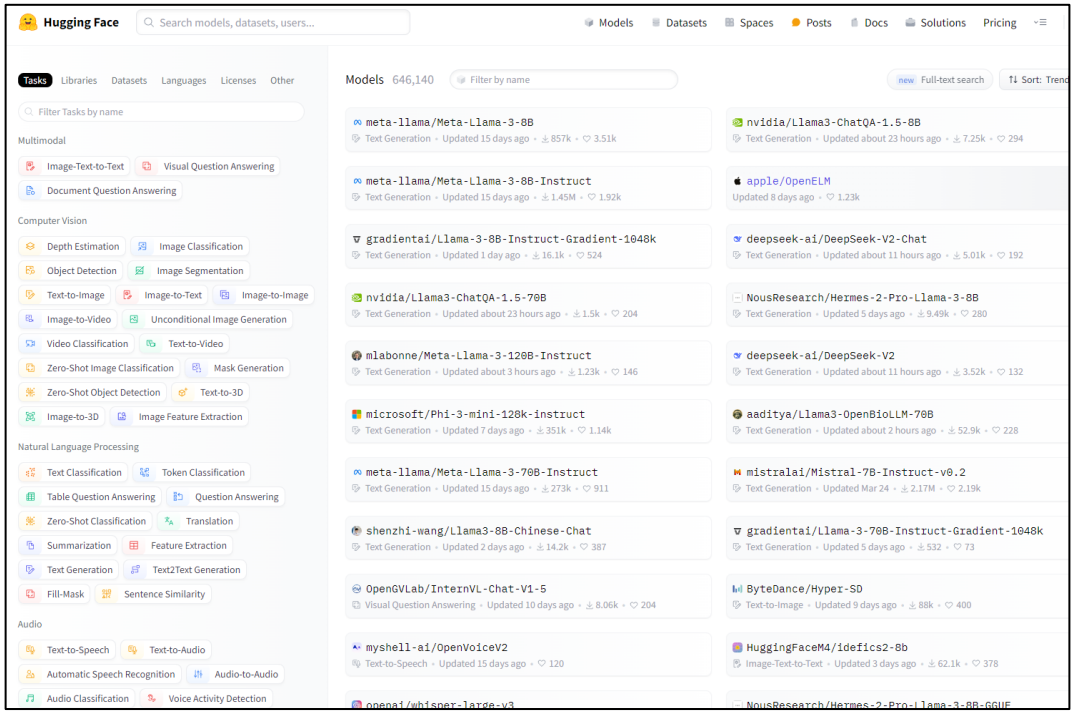
Hugging Face là một công ty khởi nghiệp tập trung vào lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và trí tuệ nhân tạo (AI). Nền tảng Hugging Face cung cấp một kho lưu trữ khổng lồ các mô hình ngôn ngữ được đào tạo sẵn (pre-trained language models) và các công cụ NLP khác, giúp các nhà nghiên cứu và nhà phát triển dễ dàng truy cập và sử dụng các công nghệ NLP tiên tiến nhất.



Hình 2. 12 HuggingFace kết nối large language models

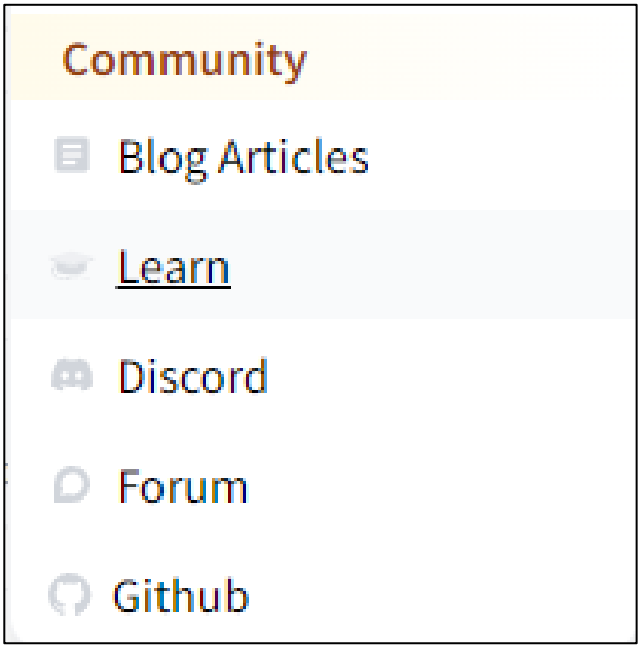
Chức năng chính của Hugging Face:

- **Kho lưu trữ mô hình ngôn ngữ:** Hugging Face cung cấp một kho lưu trữ khổng lồ với hơn 100.000 mô hình ngôn ngữ được đào tạo sẵn trên nhiều tập dữ liệu và ngôn ngữ khác nhau. Các mô hình này bao gồm các mô hình Transformer nổi tiếng như Mistra, GPT-3, RoBERTa, v.v.

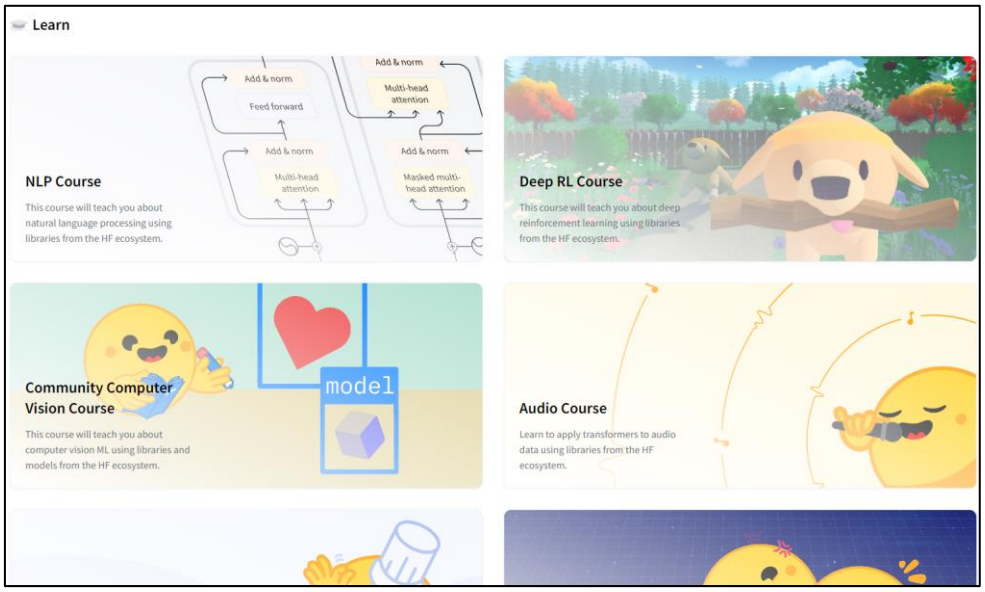


Hình 2. 13 Giáo diện Models của HuggingFace

- **Công cụ NLP:** Hugging Face cung cấp nhiều công cụ NLP giúp các nhà phát triển dễ dàng thực hiện các tác vụ NLP phổ biến như phân loại văn bản, trích xuất thông tin, tóm tắt văn bản, v.v.
- **Hỗ trợ cộng đồng:** Hugging Face có một cộng đồng người dùng và nhà phát triển tích cực, thường xuyên chia sẻ kiến thức, kinh nghiệm và hỗ trợ lẫn nhau.



Hình 2. 14 Cộng đồng HuggingFace



Hình 2. 15 Khóa học trên HuggingFace

Lợi ích của việc sử dụng Hugging Face:

- **Tiết kiệm thời gian và công sức:** Hugging Face giúp các nhà nghiên cứu và nhà phát triển tiết kiệm thời gian và công sức bằng cách cung cấp các mô hình ngôn ngữ được đào tạo sẵn và các công cụ NLP dễ sử dụng.
- **Tiếp cận các công nghệ tiên tiến:** Hugging Face giúp các nhà nghiên cứu và nhà phát triển dễ dàng truy cập và sử dụng các công nghệ NLP tiên tiến nhất.
- **Nâng cao hiệu quả:** Hugging Face giúp các nhà nghiên cứu và nhà phát triển nâng cao hiệu quả công việc bằng cách cung cấp các công cụ mạnh mẽ và dễ sử dụng.

Ứng dụng HuggingFace và Transformers trong dự án:

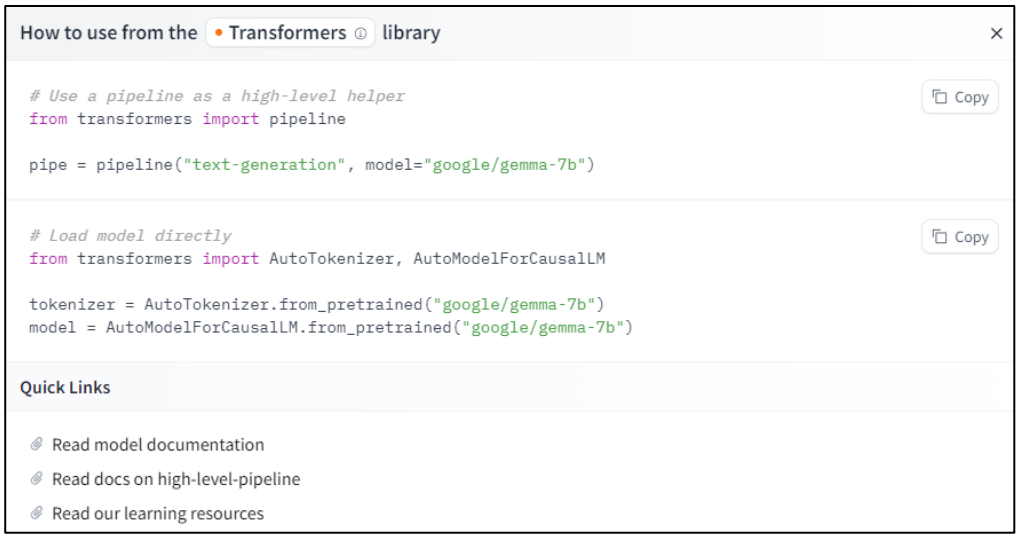
- **Transformers**:

```
tokenizer = AutoTokenizer.from_pretrained(
    APPCFG.llm_engine, token=APPCFG.gemma_token, device=APPCFG.device)
model = AutoModelForCausalLM.from_pretrained(pretrained_model_name_or_path="google/gemma-7b-it",
                                             token=APPCFG.gemma_token,
                                             torch_dtype=torch.float16,
                                             device_map=APPCFG.device
                                             )

app_pipeline = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer
)
```

Hình 2. 16 Code sử dụng Transformers

- **AutoTokenizer**: Là một lớp được sử dụng để tự động tải và tạo tokenizer từ các mô hình ngôn ngữ đã được huấn luyện
- **AutoModelForCausalLM**:AutoModelForCausalLM.from_pretrained được sử dụng để tải một mô hình ngôn ngữ dựa trên mô hình ngôn ngữ tiên tiến với kiến trúc Causal Language Model (LM) từ Hugging Face's Model Hub
- **Pipeline**: Sau khi load được large language model Gemma 7B, ta gói nó vào để sử dụng bằng Pipeline.

Hướng dẫn từ huggingface



Hình 2. 17 Hướng dẫn dùng model với Transformers

Kết luận:

Hugging Face và Transformers là hai công nghệ quan trọng trong lĩnh vực NLP. Hugging Face cung cấp một nền tảng dễ sử dụng để truy cập và sử dụng các mô hình ngôn ngữ được đào tạo sẵn và các công cụ NLP, trong khi Transformers là một kiến trúc mạng nơ-ron tiên tiến giúp nâng cao hiệu quả xử lý ngôn ngữ tự nhiên. Việc kết hợp Hugging Face và Transformers giúp các nhà nghiên cứu và nhà phát triển dễ dàng xây dựng các ứng dụng NLP mạnh mẽ và hiệu quả.

2.2.3. Flask

Flask là gì?

Flask là một **framework web vi mô** được viết bằng ngôn ngữ lập trình Python. Nó được phân loại là một microframework vì nó không yêu cầu các công cụ hoặc thư viện cụ thể. Nó không có lớp trừu tượng cơ sở dữ liệu, xác thực biểu mẫu hoặc bất kỳ thành phần nào khác mà các thư viện bên thứ ba đã có từ trước cung cấp các chức năng phổ biến.

Flask được thiết kế để **đơn giản, dễ sử dụng và linh hoạt**. Nó cung cấp một API đơn giản và trực quan giúp các nhà phát triển dễ dàng xây dựng các ứng dụng web mạnh mẽ và hiệu quả. Flask cũng rất linh hoạt, cho phép các nhà phát triển tích hợp các thư viện và công cụ bên thứ ba khác nhau để đáp ứng nhu cầu cụ thể của họ.

Hình 2. 18 Flask

Ưu điểm của Flask:

- **Dễ sử dụng:** Flask cung cấp một API đơn giản và trực quan giúp các nhà phát triển dễ dàng bắt đầu với việc phát triển web bằng Python.

- **Linh hoạt:** Flask rất linh hoạt, cho phép các nhà phát triển tích hợp các thư viện và công cụ bên thứ ba khác nhau để đáp ứng nhu cầu cụ thể của họ.
- **Nhẹ:** Flask là một framework web nhẹ, giúp nó lý tưởng cho các ứng dụng web nhỏ và đơn giản.
- **Hiệu suất cao:** Flask có thể được sử dụng để xây dựng các ứng dụng web hiệu suất cao.
- **Cộng đồng lớn:** Flask có một cộng đồng lớn và tích cực, giúp dễ dàng tìm kiếm sự trợ giúp và hỗ trợ khi cần thiết.

Nhược điểm của Flask:

- **Ít tính năng hơn:** So với các framework web khác như Django, Flask có ít tính năng hơn.
- **Cần nhiều mã hơn:** Flask yêu cầu nhiều mã hơn để xây dựng các ứng dụng web phức tạp so với các framework web khác.
- **Khó khăn trong việc quản lý dự án lớn:** Flask có thể khó quản lý cho các dự án web lớn.

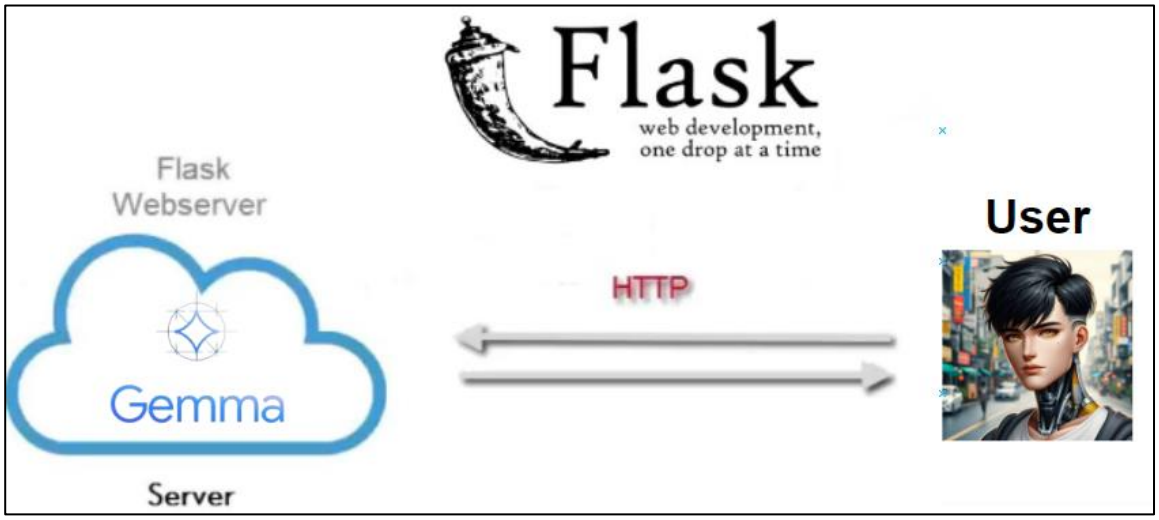
Flask phù hợp cho những ai:

- **Lập trình viên Python mới bắt đầu:** Flask là một framework web tuyệt vời cho các lập trình viên Python mới bắt đầu muốn học cách xây dựng các ứng dụng web.
- **Lập trình viên muốn xây dựng các ứng dụng web nhỏ và đơn giản:** Flask là một lựa chọn tốt cho các lập trình viên muốn xây dựng các ứng dụng web nhỏ và đơn giản một cách nhanh chóng và dễ dàng.
- **Lập trình viên muốn kiểm soát nhiều hơn đối với ứng dụng web của họ:** Flask cung cấp cho các lập trình viên nhiều quyền kiểm soát hơn đối với ứng dụng web của họ so với các framework web khác.

Ứng dụng của Flask trong dự án:

Vấn đề: Trong quá trình RAG Chatbot hoạt động, nó sẽ tiêu tốn VRAM của GPU rất nhiều, giả sử ta sử dụng model Chatgpt 3.5 có 175B parameters thì khi nhận câu hỏi từ người dùng nó sẽ được đẩy về máy chủ của OpenAI xữ lí và trả về kết quả, nhưng với việc chạy model Gemma 7B có 8.54B parameters trên máy chủ local thì việc yêu cầu lượng lớn VRAM để xử lí là rất cao, ít nhất là 24GB VRAM và việc mỗi lần load model để trả lời câu hỏi rất mất thời gian và tài nguyên.

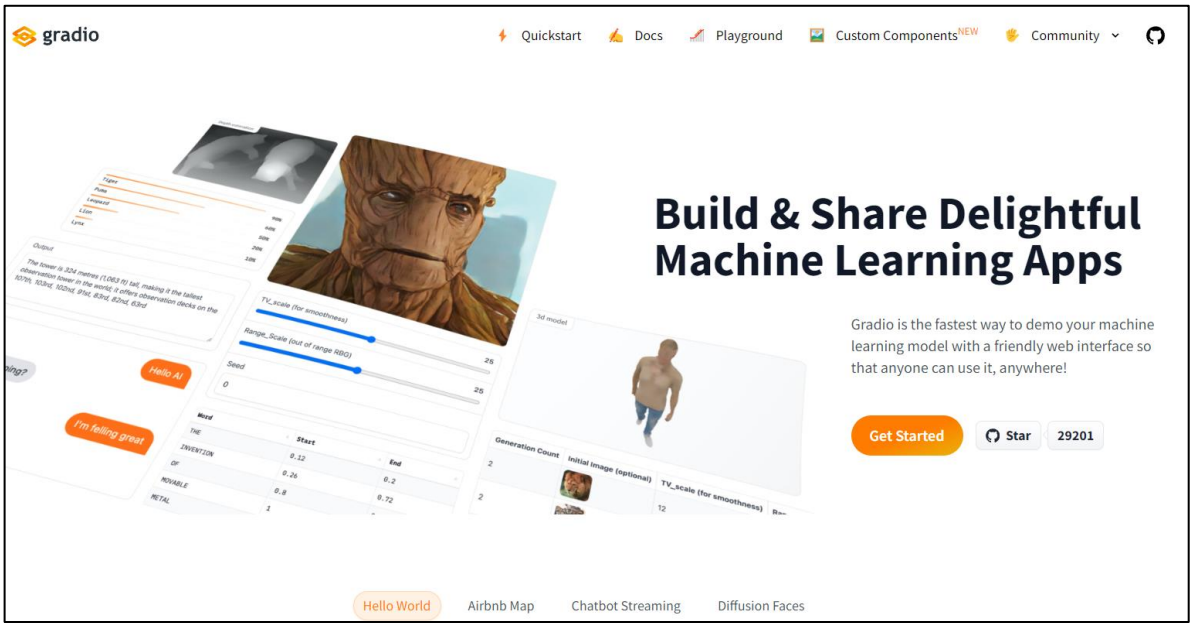
Hướng giải quyết: Ý tưởng để giải quyết vần để load model Gemma 7B với 8.54B paremeters (tham số) đó là tạo ra một web server nơi mà model đã dược chạy sẵn và chờ câu hỏi từ người dùng và trả lời. Điều này giúp model không phải load nhiều lần để trả lời câu hỏi từ người dùng và giúp cải thiện Vram GPU và thời giản phản hồi.



Hình 2. 19 Tạo web server cho dự án

2.2.4. Gradio

Gradio là một thư viện Python mã nguồn mở giúp các nhà phát triển dễ dàng xây dựng và chia sẻ các giao diện người dùng web cho các mô hình học máy. Nó cung cấp một cách đơn giản để kết nối mô hình học máy với các thành phần giao diện người dùng (UI) như thanh trượt, nút bấm và hộp văn bản, cho phép người dùng tương tác trực tiếp với mô hình và quan sát kết quả.



Hình 2. 20 Gradio

Ưu điểm chính của Gradio:

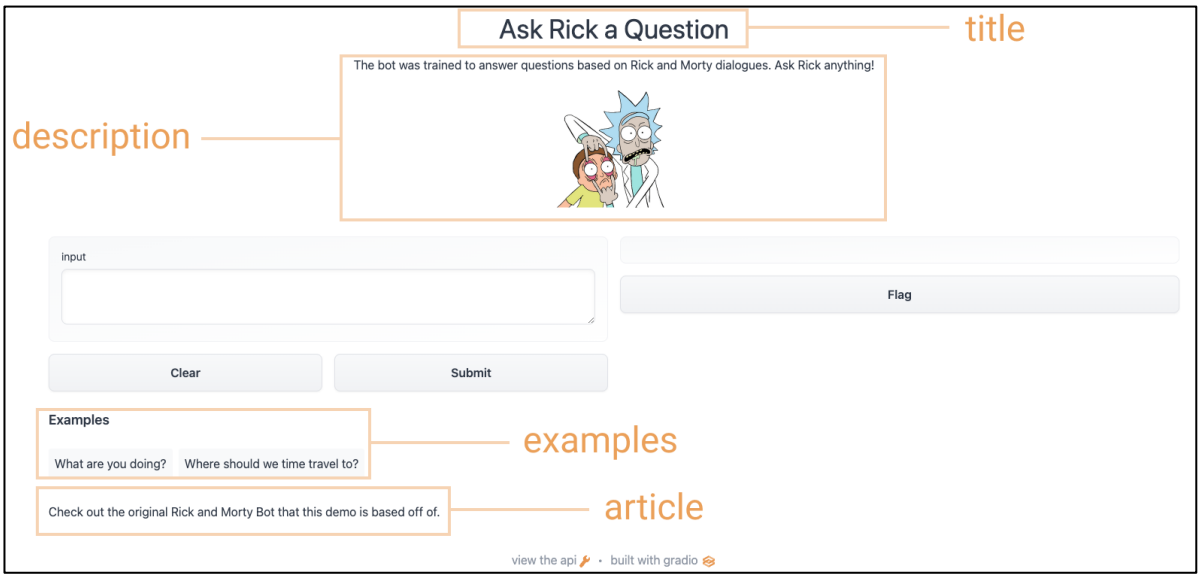
- **Dễ sử dụng:** Gradio cung cấp API đơn giản và trực quan, giúp các nhà phát triển có thể nhanh chóng xây dựng các giao diện người dùng đẹp mắt mà không cần nhiều kiến thức về lập trình web.
- **Linh hoạt:** Gradio hỗ trợ nhiều loại mô hình học máy, bao gồm mạng nơ-ron nhân tạo, mô hình ngôn ngữ và mô hình thống kê. Nó cũng hỗ trợ nhiều loại dữ liệu đầu vào, bao gồm văn bản, hình ảnh, âm thanh và video.
- **Tương tác:** Gradio cho phép người dùng tương tác trực tiếp với mô hình học máy bằng cách điều chỉnh các thanh trượt, nhấp vào nút và nhập văn bản. Điều này giúp họ dễ dàng hiểu cách mô hình hoạt động và khám phá khả năng của nó.
- **Chia sẻ:** Gradio giúp dễ dàng chia sẻ các ứng dụng học máy với người khác bằng cách tạo ra các liên kết web. Điều này cho phép người khác sử dụng ứng dụng mà không cần cài đặt bất kỳ phần mềm nào.

Ví dụ về ứng dụng Gradio:

- **Trình tạo hình ảnh:** Người dùng có thể nhập văn bản mô tả và Gradio sẽ sử dụng mô hình học máy để tạo ra hình ảnh tương ứng.

- **Phân loại văn bản:** Người dùng có thể nhập đoạn văn bản và Gradio sẽ sử dụng mô hình học máy để phân loại văn bản đó vào một trong các danh mục đã xác định trước.
- **Hệ thống hỏi đáp:** Người dùng có thể đặt câu hỏi và Gradio sẽ sử dụng mô hình học máy để trả lời câu hỏi đó dựa trên kiến thức được lưu trữ trong cơ sở dữ liệu.
- **Công cụ chỉnh sửa ảnh:** Người dùng có thể tải lên ảnh và sử dụng Gradio để áp dụng các bộ lọc và hiệu ứng do học máy tạo ra.

Gradio là một công cụ mạnh mẽ và linh hoạt giúp các nhà phát triển dễ dàng xây dựng và chia sẻ các ứng dụng học máy với người dùng không chuyên. Nó là một lựa chọn tuyệt vời cho những ai muốn khám phá tiềm năng của học máy và tạo ra các ứng dụng tương tác và thú vị.



Hình 2. 21 Giao diện demo Gradio

Ngoài những thông tin trên, bạn có thể tham khảo thêm các nguồn sau để tìm hiểu thêm về Gradio:

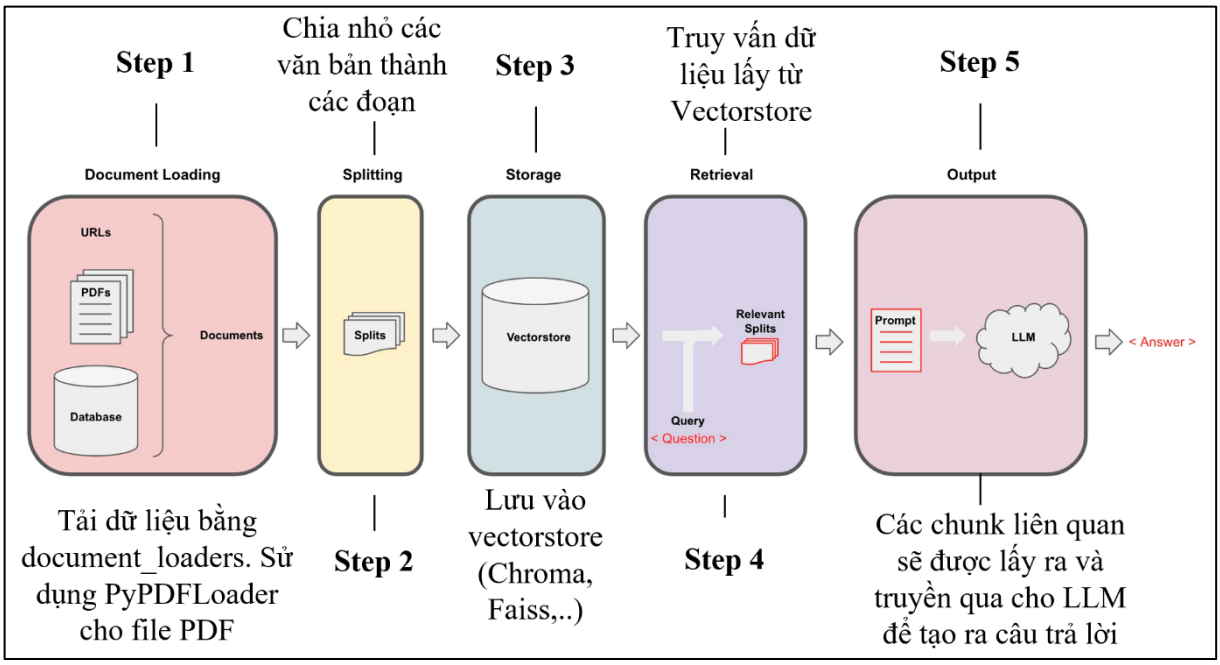
- Trang web chính thức của Gradio: https://gradio.app/
- Tài liệu Gradio: https://gradio.app/docs/

Kho lưu trữ GitHub của Gradio: https://github.com/gradio-app/gradio

2.3. Kỹ Thuật RAG (Retrieval Augmented Generation)

RAG, hay Retrieval-Augmented Generation, là một khung (framework) được thiết kế để cải thiện độ chính xác và tính hiện đại của các large language models (LLMs) trong quá trình tạo ra văn bản. Trong RAG, việc "Retrieval" (Truy xuất) đề cập đến quá trình tìm kiếm thông tin từ các nguồn dữ liệu bên ngoài trước khi tạo ra câu trả lời.

Cách thức hoạt động của RAG với LangChain:



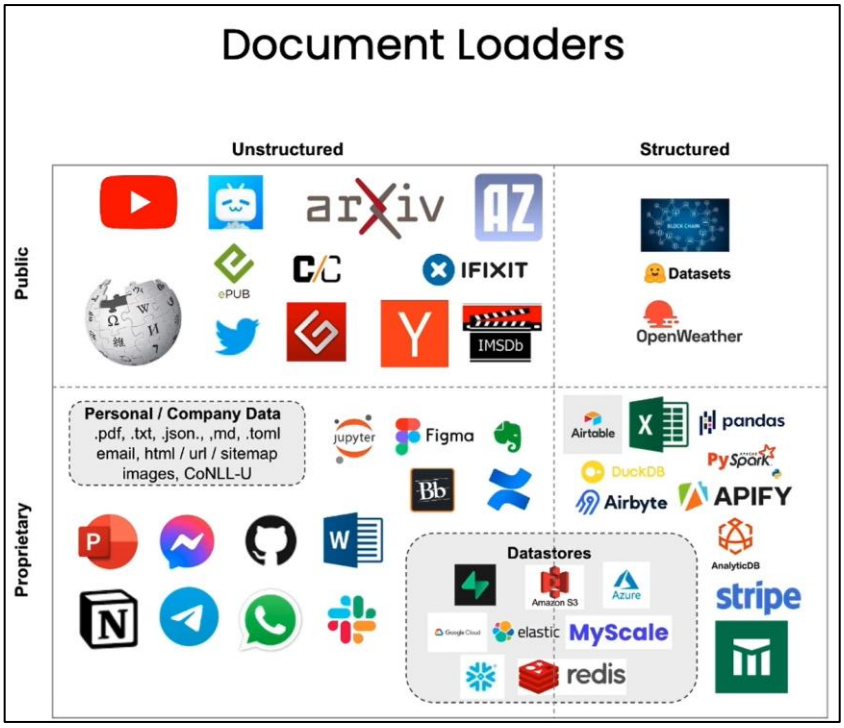
Hình 2. 22 Cách thức hoạt động của RAG với LangChain

Bước 1. Tải lên dữ liệu cần được truy vấn (Document Loading)

Tại bước này, bạn có thể sử dụng các thư viện của LangChain để tải và làm việc với một lượng lớn dữ liệu từ nhiều nguồn khác nhau như PDF, Excel, Word, Notion, YouTube, và nhiều nguồn khác. LangChain hỗ trợ hơn 80 nền tảng và nguồn dữ liệu khác nhau, mang lại sự linh hoạt và tiện lợi trong việc xử lý và quản lý dữ liệu.

Khả năng này giúp người dùng dễ dàng tiếp cận và thao tác với các loại dữ liệu cần thiết cho dự án, từ đó tối ưu hóa hiệu suất và nâng cao hiệu quả công việc. Với LangChain, việc chuyển đổi và xử lý dữ liệu trở nên đơn giản và nhanh chóng, giúp tiết kiệm thời gian và nguồn lực. Không chỉ vậy, sự hỗ trợ

rộng rãi từ nhiều nền tảng khác nhau còn cho phép người dùng tích hợp và liên kết dữ liệu từ nhiều nguồn, tạo ra một hệ sinh thái dữ liệu phong phú và mạnh mẽ.



Hình 2. 23 Các nền tảng và nguồn dữ liệu

Ví dụ về bước Document Loading với file PDF:

```
In [2]: from langchain.document_loaders import PyPDFLoader
        loader = PyPDFLoader("docs/cs229_lectures/MachineLearning-Lecture01.pdf")
        pages = loader.load()

In [3]: len(pages)

22

In [4]: page = pages[0]

In [5]: print(page.page_content[0:500])

MachineLearning-Lecture01
Instructor (Andrew Ng):  Okay. Good morning. Welcome to CS229, the machin
e
learning class. So what I wanna do today is ju st spend a little time goi
ng over the logistics
of the class, and then we'll start to  talk a bit about machine learning.
By way of introduction, my name's  Andrew Ng and I'll be instru ctor for
this class. And so
I personally work in machine learning, and I' ve worked on it for about 1
5 years now, and
I actually think that machine learning i
```

Hình 2. 24 Code mẫu load file PDF

Sau khi load thành công dữ liệu có thể kiểm tra bằng cách in ra nội dung của trang đầu tiên với giới hạn 500 ký tự.

Bước 2. Tách/Chia nhỏ dữ liệu (Splitting)

a. Lý do phải chia nhỏ văn bản:

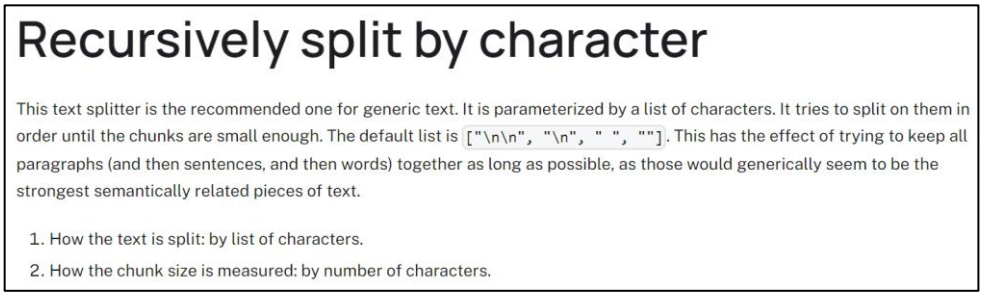
Hiệu quả tìm kiếm: Khi xử lý văn bản dài, việc chia nhỏ giúp tăng cường khả năng tìm kiếm và truy xuất thông tin chính xác hơn. Nếu văn bản không được chia nhỏ, toàn bộ tài liệu sẽ được xem xét trong mỗi lần truy vấn, gây tốn nhiều tài nguyên và thời gian.

Giới hạn mô hình: Nhiều mô hình ngôn ngữ có giới hạn về số lượng từ hoặc ký tự mà chúng có thể xử lý trong một lần (ví dụ, mô hình GPT-3 của OpenAI có giới hạn khoảng 2048 token

Cải thiện chất lượng trả lời: Khi văn bản được chia nhỏ, mô hình có thể tập trung vào một phần cụ thể của văn bản mà chứa thông tin liên quan nhất đến truy vấn.

b. Cách thức thực hiện:

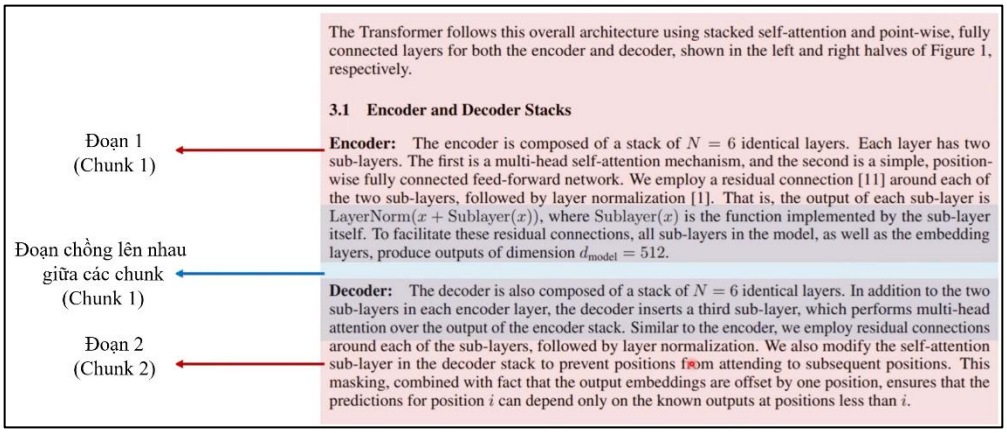
Trong thư viện text_splitter của Langchain có nhiều bộ chia văn bản nhưng được khuyến nghị trong việc sủ dụng cho generate text là bộ **Recursive CharacterTextSplitter()**



Hình 2. 25 Thông tin về RCT trên trang chủ LangChain

- Thực hiện việc chia văn bản dựa trên các ký tự. Nó sẽ chia đệ quy bằng các ký tự khác nhau để tìm ra cách chia phù hợp. Và dựa vào danh sách các ký tự mặc định hoặc tùy chỉnh từ code nó ưu tiên chọn ra đoạn có các khoảng trống, xuống dòng, sau dấu chấm xuống dòng, …. Tùy thuộc vào tùy chỉnh trong biến separator.

- Sẽ có 2 tham số chính cần lưu ý trong việc chia nhỏ văn bản: **chunk_size** và **chunk_overlap**



Hình 2. 26 Ví dụ về chunk_size và chunk_overlap

- Chunk_size: cho phép thiết lập số ký tự của 1 đoạn (ví dụ chunk_size = 150).
- Chunk_overlap: đặt cho khoảng lặp lại được chồng lên nhau giữa 2 đoạn, điều này giúp cho việc truy vấn dữ liệu không bị mất chữ giữa các đoạn với nhau và tạo ra sự vô nghĩa trong đoạn (ví dụ: câu cuối của đoạn 1: “Hôm nay tôi đ” , câu đầu của đoạn 2: “ã viết lại báo cáo tốt nghiệp”).

c. Ví dụ:

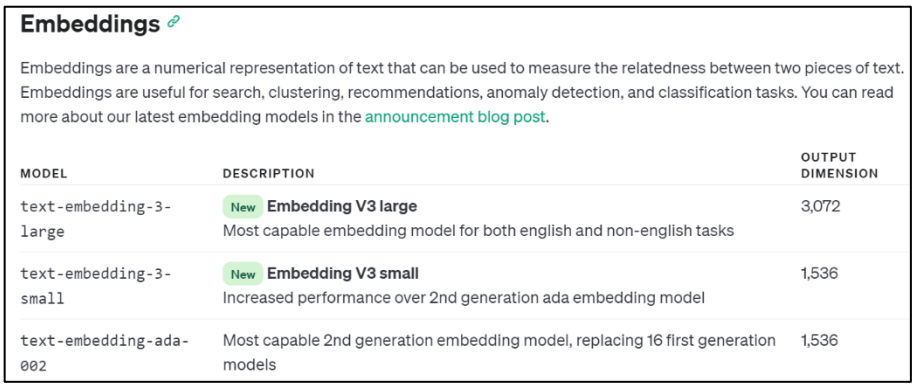
```
In [ ]: some_text = """When writing documents, writers will use document structure
        This can convey to the reader, which idea's are related. For example, clos
        are in sentances. Similar ideas are in paragraphs. Paragraphs form a docum
        Paragraphs are often delimited with a carriage return or two carriage retu
        Carriage returns are the "backslash n" you see embedded in this string. \
        Sentences have a period at the end, but also, have a space.\
        and words are separated by space."""

In [ ]: r_splitter = RecursiveCharacterTextSplitter(
            chunk_size=450,
            chunk_overlap=0,
            separators=["\n\n", "\n", " ", ""]
        )

In [ ]: r_splitter.split_text(some_text)

In [ ]: r_splitter = RecursiveCharacterTextSplitter(
            chunk_size=150,
            chunk_overlap=0,
            separators=["\n\n", "\n", "(?<=\. )", " ", ""]
        )
        r_splitter.split_text(some_text)
```

Hình 2. 27 Code mẫu tách/chia nhỏ dữ liệu

Bước 3. Lưu dữ liệu (Storage)

a. Embedding là gì ?

- Là lớp dùng để tạo ra các vector biểu diễn văn bản , cho phép người dùng nhúng và làm việc với các văn bản đấy trên máy tính. Vì máy tính không thể hiểu được ngôn ngữ con người nên việc chuyển chúng thành các vector gồm các dãy số là điều cần thiết. Các model được tạo ra với mục đích embed văn bản từ các tổ chức, công ty phổ biến như (OpenAI, BBAI, Google, Microsoft,..)



Embeddings

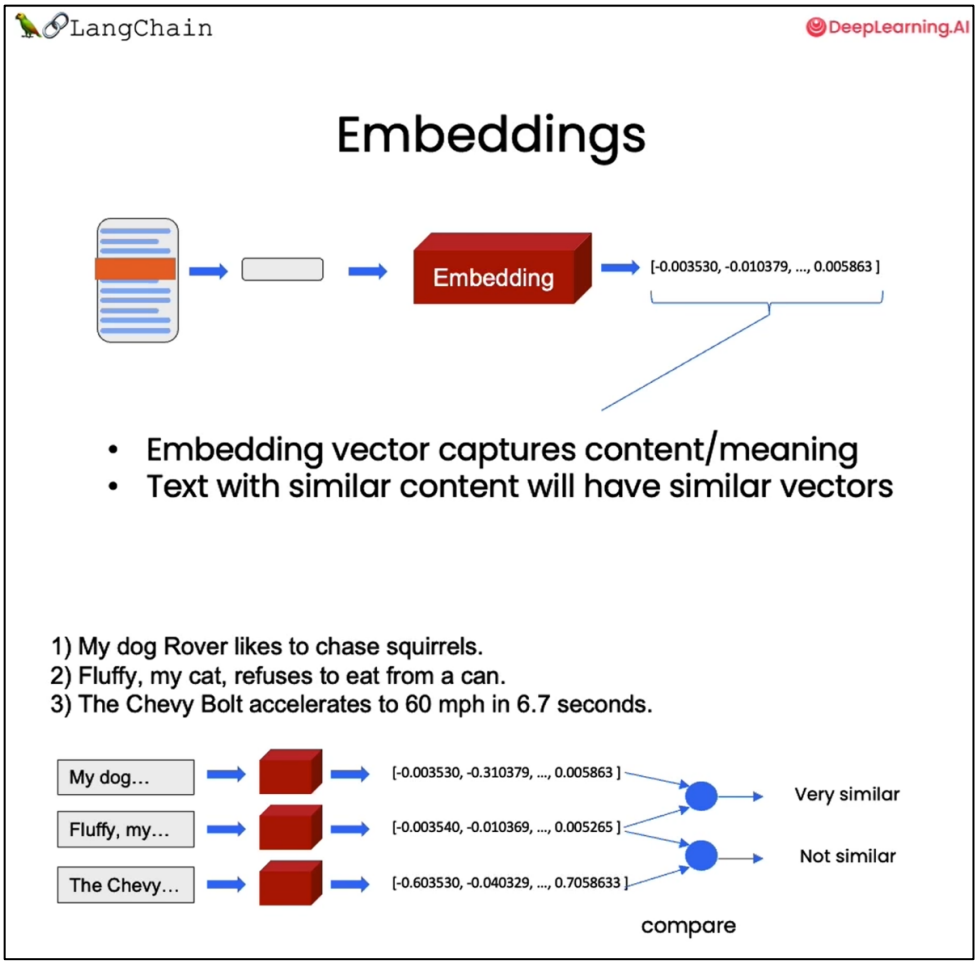
Embeddings are a numerical representation of text that can be used to measure the relatedness between two pieces of text. Embeddings are useful for search, clustering, recommendations, anomaly detection, and classification tasks. You can read more about our latest embedding models in the announcement blog post.

MODEL	DESCRIPTION	OUTPUT DIMENSION
text-embedding-3-large	New **Embedding V3 large** Most capable embedding model for both english and non-english tasks	3,072
text-embedding-3-small	New **Embedding V3 small** Increased performance over 2nd generation ada embedding model	1,536
text-embedding-ada-002	Most capable 2nd generation embedding model, replacing 16 first generation models	1,536

Hình 2. 28 Ảnh model embedding của OpenAI

- Mỗi vector là một dãy số đại diện cho nghĩa và nội dung của văn bản. Các vector có dãy số càng giống nhau thì nội dung của chúng cũng tương tự nhau. Ví dụ, nếu chúng ta có các câu như sau: câu 1 và câu 2 đều nói về chó và mèo, do đó, các vector của chúng sẽ có phần đầu dãy số giống nhau. Ngược lại, câu 3 nói về một chủ đề hoàn toàn khác, nên vector của nó sẽ bao gồm các chữ số khác biệt.
- Điều này cho phép mô hình xác định và so sánh mức độ tương đồng giữa các văn bản dựa trên cấu trúc vector của chúng. Khi các vector có nhiều phần giống nhau, điều đó cho thấy các văn bản liên quan đến nhau về mặt nội dung. Đây là một cách hiệu quả để mô hình hiểu và phân loại thông tin, giúp tối ưu hóa quá trình xử lý ngôn ngữ tự nhiên.
- Việc sử dụng vector như thế này không chỉ giúp máy học nhận diện nội dung văn bản mà còn hỗ trợ việc phân tích ngữ cảnh và mối quan hệ giữa các từ. Nhờ đó, mô hình có thể cung cấp các kết quả phân tích chính xác và có ý

nghĩa, hỗ trợ đắc lực cho các ứng dụng như tìm kiếm thông tin, phân loại văn bản, và xử lý ngôn ngữ tự nhiên.



Hình 2. 29 Giải thích Embeddings

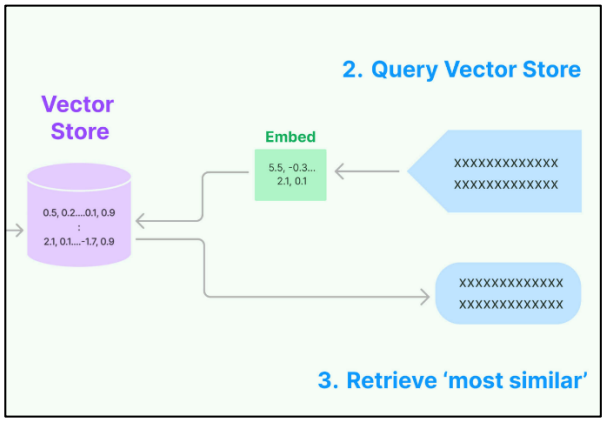
* Lưu ý: Cũng tương tự như dữ liệu văn bản, việc người dùng nhập vào câu hỏi (query) thì câu hỏi hay văn bản đấy cũng phải được embbed thành dạng vector

b. Vectorstores

Sau khi hoàn thành bước embedding, cần lưu các đoạn chunk chứa các vector dữ liệu và truy vấn của người dùng vào một nơi để có thể sử dụng cho việc truy vấn (retrieval). Việc lưu trữ này giúp dễ dàng truy xuất thông tin khi cần, đảm bảo dữ liệu được tổ chức và sẵn sàng cho các thao tác tìm kiếm.

Khi có nhu cầu truy vấn, hệ thống sẽ sử dụng các vector đã lưu để nhanh chóng tìm kiếm và so sánh, từ đó cung cấp kết quả chính xác và hiệu quả. Quy trình này không chỉ tối ưu hóa hiệu suất mà còn nâng cao trải nghiệm người

dùng bằng cách giảm thời gian phản hồi và cải thiện độ chính xác của kết quả tìm kiếm.



Hình 2. 30 Giải thích Vectorstores

Sử dụng các vectorstore mà LangChain cung cấp như: FAISS, Chroma, Lance, Pinecone,…

c. Ví dụ:

Embeddings

Let's take our splits and embed them.

```
In [5]: from langchain.embeddings.openai import OpenAIEmbeddings
        embedding = OpenAIEmbeddings()
        import numpy as np

In [6]: sentence1 = "i like dogs"
        sentence2 = "i like canines"
        sentence3 = "the weather is ugly outside"

In [7]: embedding1 = embedding.embed_query(sentence1)
        embedding2 = embedding.embed_query(sentence2)
        embedding3 = embedding.embed_query(sentence3)

In [8]: np.dot(embedding1, embedding2)
0.9630351468341317

In [9]: np.dot(embedding1, embedding3)
0.7701147942700532

In [10]: np.dot(embedding2, embedding3)
0.7591130371091993
```

Hình 2. 31 Ví dụ Embeddings

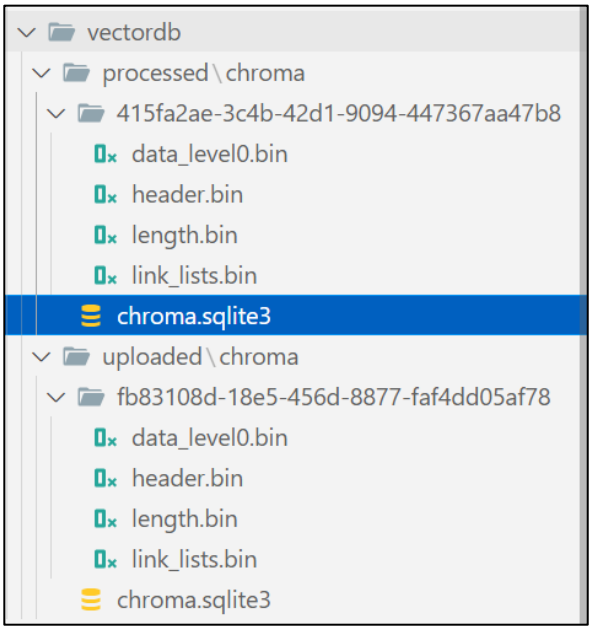
Trong ví dụ về Embeddings bằng cách sử dụng hàm dot của numpy có thể thấy được điểm tương đồng giữa vector của sentence 1 và 2 là 0.96 còn giữa 1 và 3 , 2 và 3 thì điểm số không cao chỉ khoảng 0.7. Đó là vì sentence 1 và 2 cùng nói về việc thích cái gì đó, còn câu 3 nói về thời tiết.

Vectorstores

```
In [12]: from langchain.vectorstores import Chroma

In [13]: persist_directory = 'docs/chroma/'

In [14]: !rm -rf ./docs/chroma  # remove old database files if any

In [16]: vectordb = Chroma.from_documents(
             documents=splits,
             embedding=embedding,
             persist_directory=persist_directory
         )
```

Hình 2. 32 Ví dụ Vectorstores

Trong ví dụ sử dụng vectorstore là Chroma được lưu ở đường dẫn docs/chroma. Hình ảnh sau khi model embedding hoạt động và lưu vào chroma trong dự án RAG Chatbot



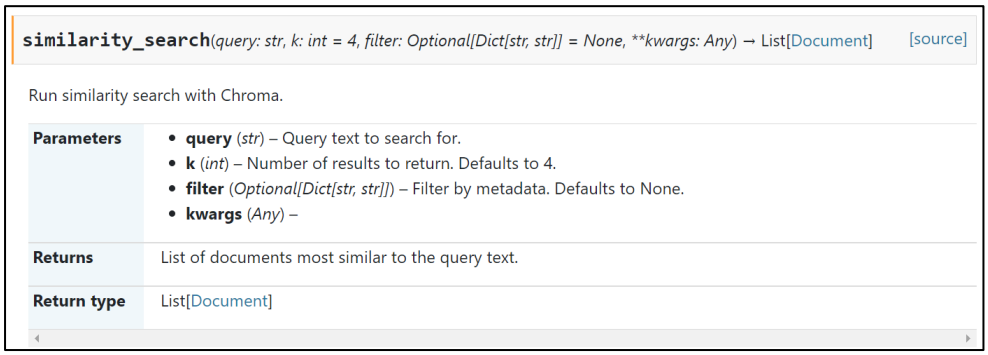
Hình 2. 33 VectorDB Chroma

Bước 4. Truy vấn dữ liệu (Retrieval)

Có nhiều thuật toán truy vấn có thể sử dụng trong việc xây xựng chatbot từ Vector data base Chroma như: Similarity Search,MMR (Maximum marginal relevance), working with metadata,…Trong dự án này sử dụng phương thức Similarity Search.

Similarity Search truy vấn bằng cách dựa vào độ giống nhau giữa **vector query** được nhập vào và các **vector lưu trong Vectorstores**. Gồm các thâm số chính là:

- Query (str): Câu hỏi, mệnh lệnh mà người dùng nhập vào cho chatbot.
- K(int): số tài liệu mà mong muốn để truy vấn (mặc định là 4).



similarity_search(*query: str, k: int = 4, filter: Optional[Dict[str, str]] = None, **kwargs: Any*) → List[Document] [source]

Run similarity search with Chroma.

Parameters	• **query** (*str*) – Query text to search for. • **k** (*int*) – Number of results to return. Defaults to 4. • **filter** (*Optional[Dict[str, str]]*) – Filter by metadata. Defaults to None. • **kwargs** (*Any*) –
Returns	List of documents most similar to the query text.
Return type	List[Document]

Hình 2. 34 Phương thức truy vấn Similarity Search

Sau đây là ví dụ về Similarity Search.

```
texts = [
    """The Amanita phalloides has a large and imposing epigeous (aboveground)
    fruiting body (basidiocarp).""",

    """A mushroom with a large fruiting body is the Amanita phalloides.
    Some varieties are all-white.""",

    """A. phalloides, a.k.a Death Cap, is one of the most
    poisonous of all known mushrooms.""",
```

Hình 2. 35 Ví dụ về Similarity Search

Dữ liệu được dùng là textx gồm 3 đoạn lần lượt nói về nấm Amanita phalloides

- "The Amanita phalloides has a large and imposing epigeous (aboveground) fruiting body (basidiocarp)."

- Amanita phalloides có một thân nấm lớn và ấn tượng mọc trên mặt đất (thể quả).

- "A mushroom with a large fruiting body is the Amanita phalloides. Some varieties are all-white."

- Một loại nấm có thân quả lớn là Amanita phalloides. Một số giống có màu trắng hoàn toàn.

- "A. phalloides, a.k.a Death Cap, is one of the most poisonous of all known mushrooms."

- phalloides, còn được gọi là Nấm mũ tử thần, là một trong những loài nấm độc nhất được biết đến.

```
In [6]: smalldb = Chroma.from_texts(texts, embedding=embedding)

In [7]: question = "Tell me about all-white mushrooms with large fruiting bodies"

In [8]: smalldb.similarity_search(question, k=2)

[Document(page_content='A mushroom with a large fruiting body is the Aman
ita phalloides. Some varieties are all-white.', metadata={}),
 Document(page_content='The Amanita phalloides has a large and imposing e
pigeous (aboveground) fruiting body (basidiocarp).', metadata={})]
```

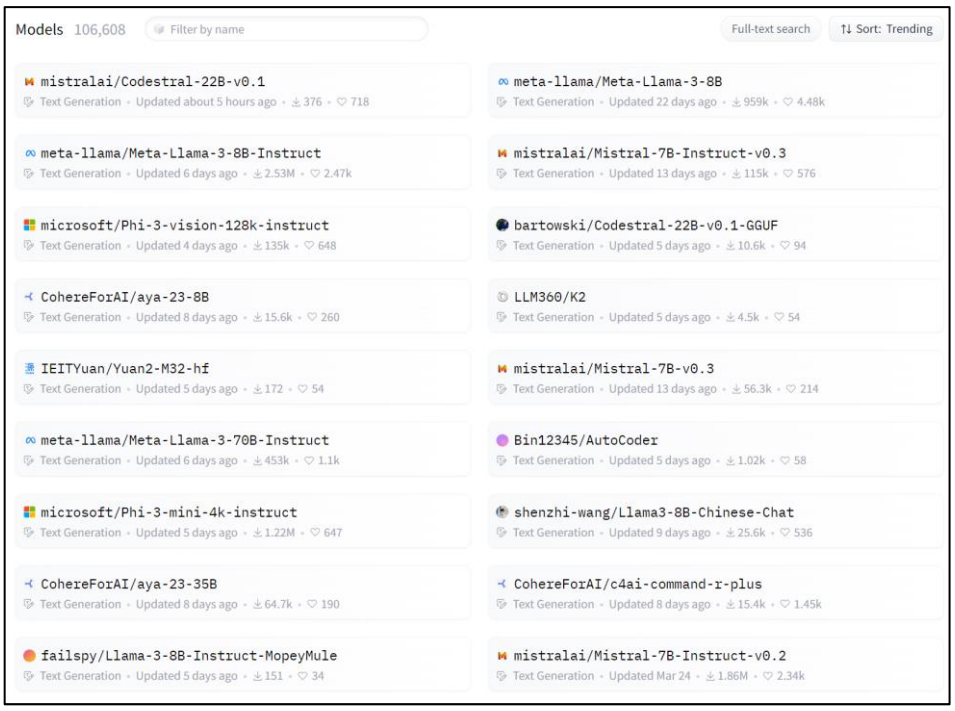
Hình 2. 36 Output của ví dụ

Sử dụng Similarity Search của Chroma output trả về 2 đoạn dữ liệu (vì k = 2) liên quan nhất tới câu hỏi, đó là đoạn 1 nói về thân nấm và đoạn 2 nói về thân màu trắng.

Bước 5. Trả ra kết quả thông qua Large Language Model (Output)

Sau khi truy vấn thành công những dữ liệu liên quan nhất tới câu hỏi việc tiếp cuối cùng chính là đưa những vector đó cho một mô hình ngôn ngữ lớn (LLM) để sử lý và trả về câu trả lời cho người dùng.

- Một số Large Language Model trả phí như: OpenAI: GPT-4 Turbo and GPT-4, GPT-3.5 Turbo,..)
- Một số model miễn phí từ các **HuggingFace** được tạo ra từ các công ty, tổ chức lớn như: Llama – 3 – 8B – in, Microsoft – Phi 3,
- Đặc biệt LLM mạnh mẽ của Việt Nam: Viet-Mistral/Vistral-7B-Chat, VietAI/gpt-j-6B-vietnamese-news, nguyenviet/PhoGPT-7B5-Instruct-GGUF,…



Hình 2. 37 LLM của Việt Nam

Tầm quan trọng của RAG trong việc truy suất dữ liệu là:

1. Hiểu ngôn ngữ tự nhiên tốt hơn: RAG có thể hiểu ngôn ngữ tự nhiên và ngữ cảnh của truy vấn tốt hơn, giúp tìm kiếm thông tin chính xác hơn trong dữ liệu riêng.
2. Tìm kiếm hiệu quả hơn: RAG có thể tìm kiếm các tài liệu có liên quan trong dữ liệu riêng nhanh hơn và hiệu quả hơn so với các phương pháp truy xuất thông tin truyền thống.

3. Cung cấp câu trả lời toàn diện hơn: RAG có thể tạo ra câu trả lời toàn diện và nhiều thông tin hơn dựa trên các tài liệu được tìm thấy trong dữ liệu riêng.
4. Bảo mật dữ liệu: RAG giúp bảo mật dữ liệu riêng vì nó không yêu cầu chia sẻ dữ liệu với các bên thứ ba.

Ví dụ ứng dụng:

- Tìm kiếm thông tin trong kho tài liệu cá nhân: RAG có thể được sử dụng để tìm kiếm thông tin trong kho tài liệu cá nhân như email, tài liệu, ghi chú, v.v.
- Hệ thống hỏi đáp cho doanh nghiệp: RAG có thể được sử dụng để xây dựng hệ thống hỏi đáp cho doanh nghiệp giúp nhân viên tìm kiếm thông tin từ tài liệu nội bộ.
- Công cụ phân tích dữ liệu: RAG có thể được sử dụng để phân tích dữ liệu văn bản và trích xuất thông tin chi tiết.

Tóm lại, RAG là một công cụ mạnh mẽ và linh hoạt có thể giúp cải thiện hiệu quả truy xuất thông tin từ dữ liệu riêng. Nó có tiềm năng to lớn cho nhiều ứng dụng khác nhau, bao gồm tìm kiếm cá nhân, hệ thống hỏi đáp doanh nghiệp và phân tích dữ liệu.

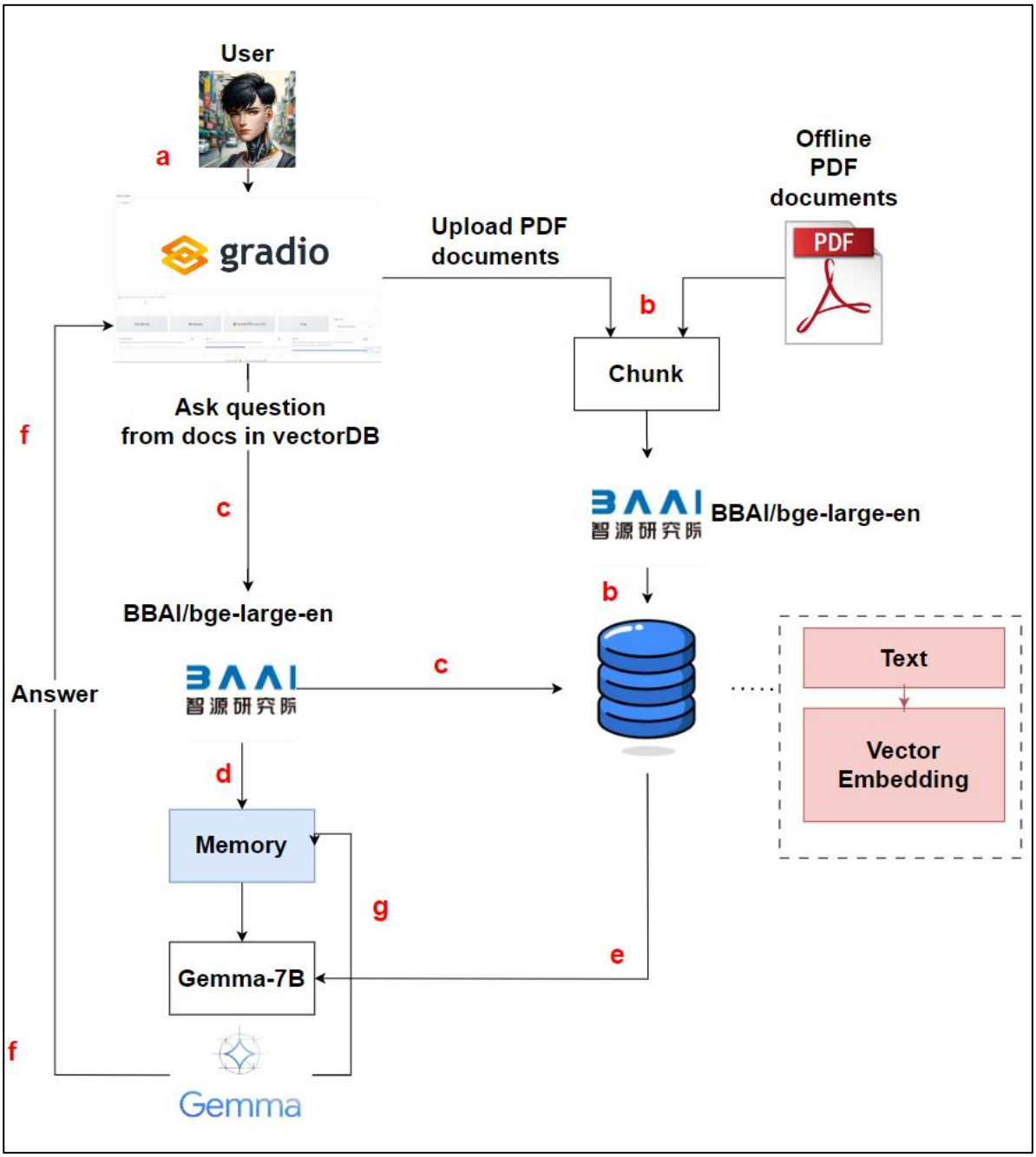
.

CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1. Phân tích hệ thống

Sơ đồ workflow tổng quát của Chatbot RAG sử dụng:

- Large language model (LLM): google/gemma-7b
- Embedding model: BAAI/bge-large-en, được fine-tuning cho tiếng Anh (en)



Hình 2. 38 Sơ đồ workflow tổng quát

Quy trình hoạt động của hệ thống được thực hiện theo các bước sau:

a. Người dùng nhập câu hỏi hoặc mệnh lệnh thông qua giao diện web Gradio.

b. Dữ liệu người dùng tải lên và dữ liệu đã qua tiền xử lý được chia thành các đoạn nhỏ (chunk) và chuyển đổi thành vector thông qua mô hình nhúng (embedding model) bge-large-en. Các vector này sau đó được lưu trữ vào cơ sở dữ liệu vector (vectorDB) Chroma.

c. Câu hỏi của người dùng cũng được chuyển đổi thành vector thông qua cùng mô hình nhúng bge-large-en và lưu vào vectorDB Chroma.

d. Đồng thời, câu hỏi này được chuyển vào bộ nhớ của Chroma, nơi lưu trữ các câu hỏi trước đó hoặc dữ liệu đã được lưu trong lịch sử trò chuyện.

e. Khi thuật toán truy vấn được thực hiện, các đoạn (chunk) chứa các vector liên quan đến câu hỏi sẽ được xác định dựa trên độ tương đồng giữa vector của truy vấn và dữ liệu đã lưu. Những đoạn liên quan này sau đó sẽ được chuyển tới mô hình ngôn ngữ lớn (LLM) Google Gemma 7B.

f. Mô hình Google Gemma 7B sẽ sử dụng dữ liệu đã được truy vấn cùng với dữ liệu từ bộ nhớ để đưa ra câu trả lời. Câu trả lời này sẽ được hiển thị trên giao diện Gradio.

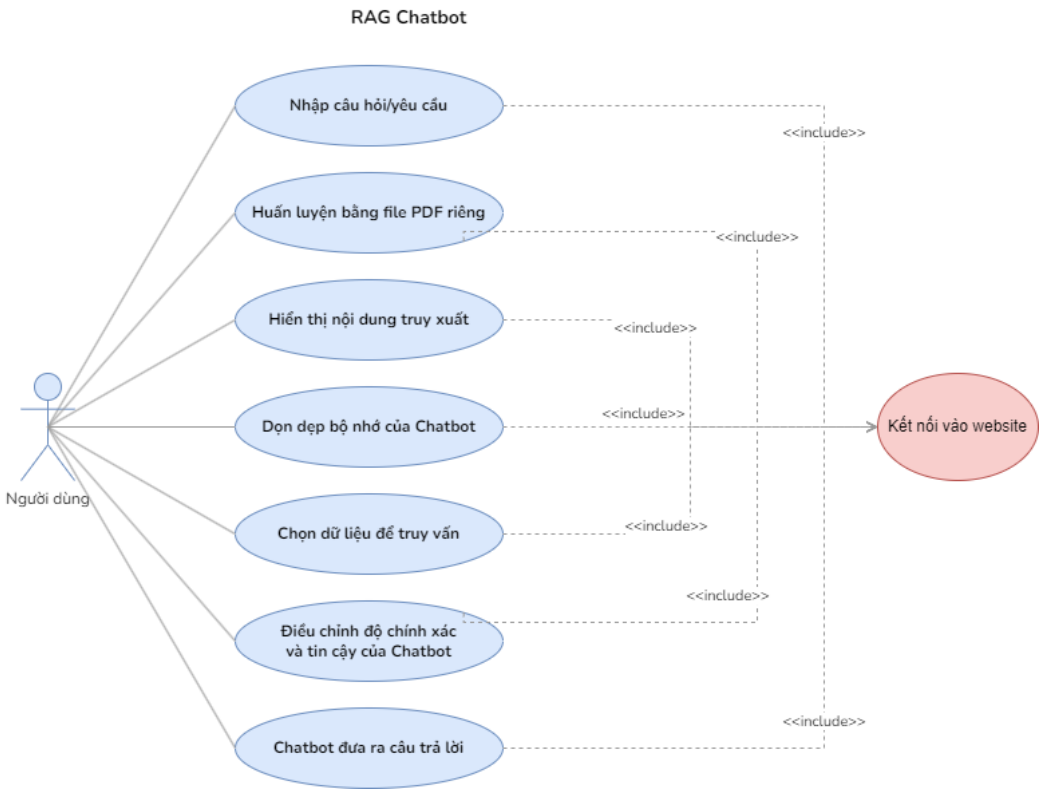
g. Cuối cùng, câu trả lời sẽ được lưu vào bộ nhớ để tăng tốc độ xử lý và truy vấn cho các câu hỏi liên quan trong tương lai.

Quy trình này đảm bảo rằng dữ liệu và truy vấn của người dùng được xử lý một cách hiệu quả và chính xác. Khi người dùng nhập câu hỏi qua Gradio, hệ thống không chỉ lưu trữ và chuyển đổi dữ liệu thành các vector một cách có tổ chức mà còn sử dụng các mô hình nhúng tiên tiến để đảm bảo tính chính xác cao. Việc lưu trữ câu hỏi và câu trả lời trong bộ nhớ không chỉ giúp tăng tốc độ xử lý các truy vấn tiếp theo mà còn giúp hệ thống học hỏi và cải thiện qua thời gian.

3.1.1. Các tác nhân

Trong dự án RAG Chatbot này, tác nhân trực tiếp sử dụng game là: Người dùng (Tương tác với tất cả các UI/UX có trong giao diện Chatbot trên Gradio).

3.1.2. Sơ đồ Usecase tổng quát



Hình 2. 39 Usecase Tổng quát

3.1.3. Các Usecase chi tiết

3.1.3.1. Đặc tả chức năng Nhập câu hỏi

a. Đặc tả chức năng

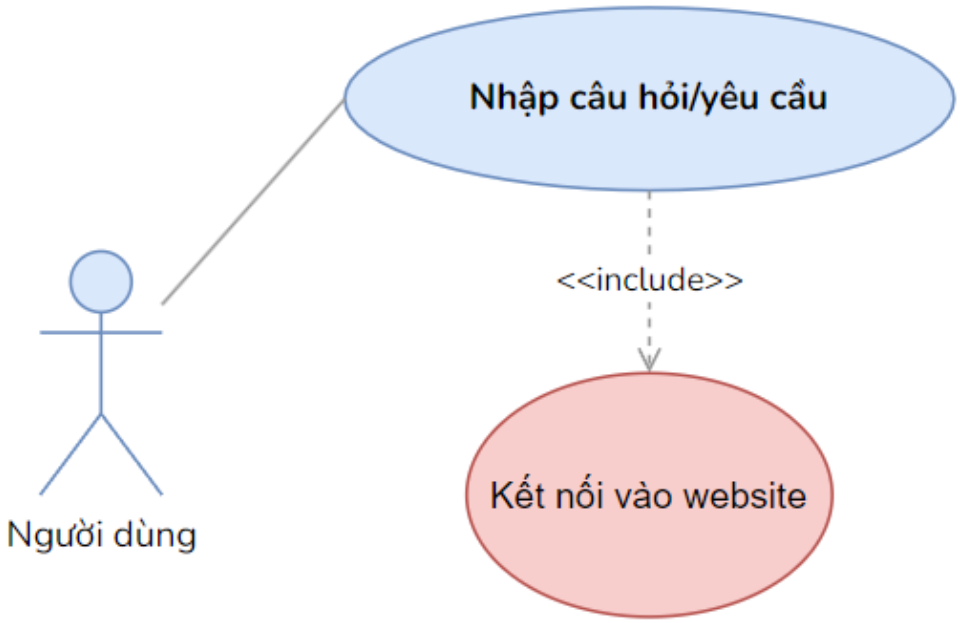
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Nhập câu hỏi cho phép người dùng đặt câu hỏi cho Chatbot để hỏi về những dữ liệu riêng đã được huấn luyện gồm dự liệu trước tiền xử lí trong file docs, và dữ liệu upload real-time từ người dùng.

Quy trình:

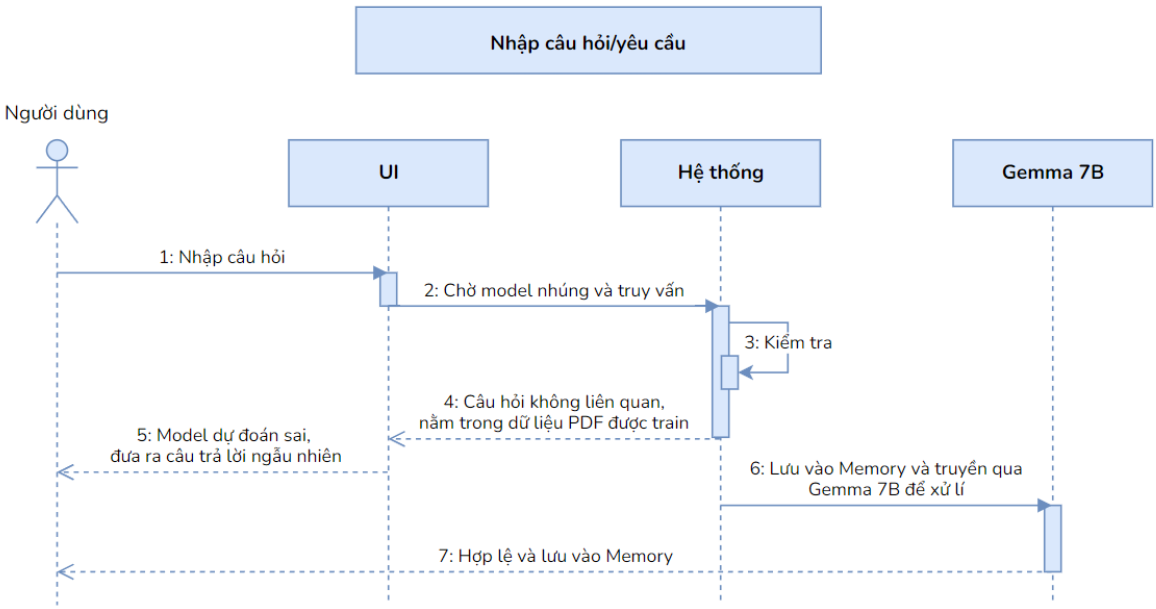
1. Người dùng nhấp vào phần text box hiện " Enter text and press enter, or upload PDF files.
2. Sau khi nhập câu hỏi người dùng bấm nút " Submit text ".

b. Biểu đồ UseCase



Hình 2. 40 Usecase nhập câu hỏi

c. Sơ đồ tuần tự



Hình 2. 41 Sơ đồ tuần tự Nhập câu hỏi

3.1.3.2. Đặc tả chức năng Huấn luyện bằng file PDF riêng

a. Đặc tả chức năng

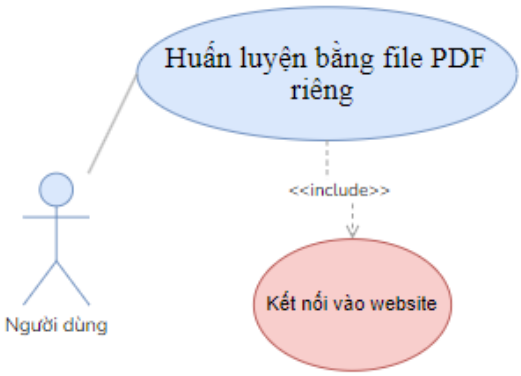
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Huấn luyện bằng file PDF riêng cho phép người dùng bấm váo nút Upload PDF or doc files, sau khi file được tải lên người đừng đợi Chatbot xử lí dữ liệu và thông báo sau khi sẵn sàng để chat với dữ liệu đó " Uploaded file are ready. Please ask your question".

Quy trình:

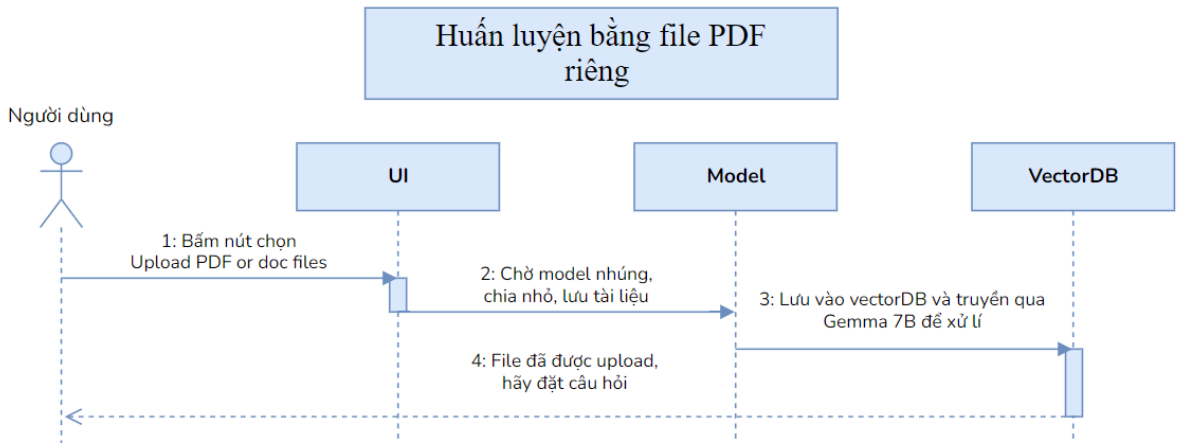
1. Người dùng bấm váo nút Upload PDF or doc files
2. Chọn file từ máy và upload

b. Biểu đồ UseCase



Hình 2. 42 Usecase Huấn luyện bằng PDF riêng

c. Sơ đồ tuần tự



Hình 2. 43 Sơ đồ tuần tự Huấn luyện bằng PDF riêng

3.1.3.3. Đặc tả chức năng Hiển thị nội dung truy xuất

a. Đặc tả chức năng

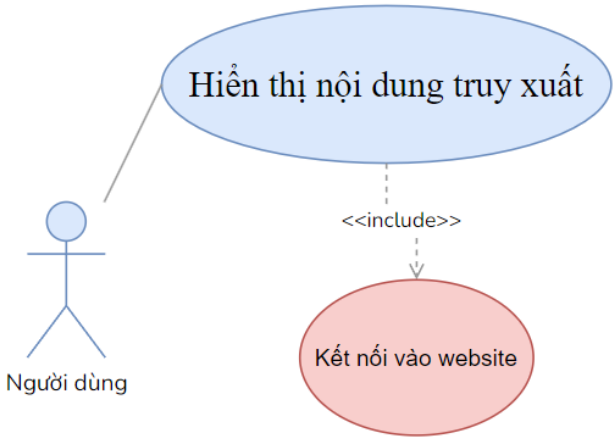
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Hiển thị nội dung truy xuất Cho phép người dùng đối chiếu từ nội dung Chatbot sử dụng để truy vấn ra câu trả lời, tại đây sẽ hiển thì 2 content liên quan tới câu trả lời.

Quy trình:

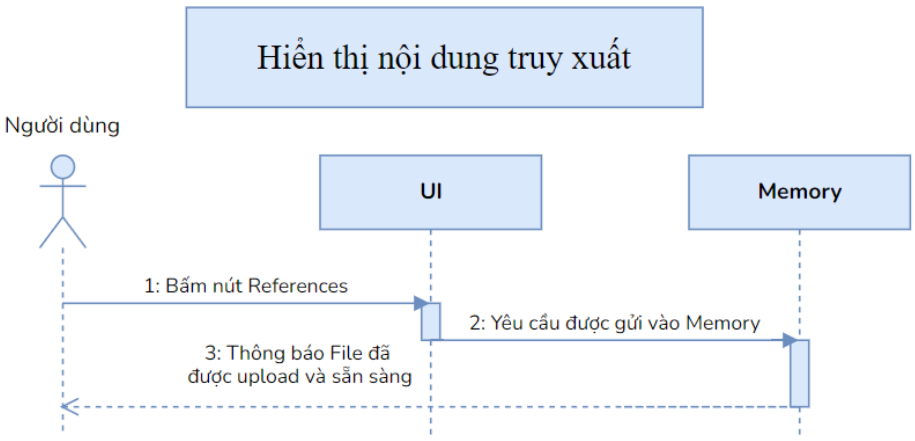
1. Người dùng bấm váo nút References

b. Biểu đồ UseCase



Hình 2. 44 Usecase Hiển thị nội dung truy xuất

c. Sơ đồ tuần tự



Hình 2. 45 Sơ đồ tuần tự Hiển thị nội dung truy xuất

3.1.3.4. Đặc tả chức năng Dọn dẹp bộ nhớ Chatbot

a. Đặc tả chức năng

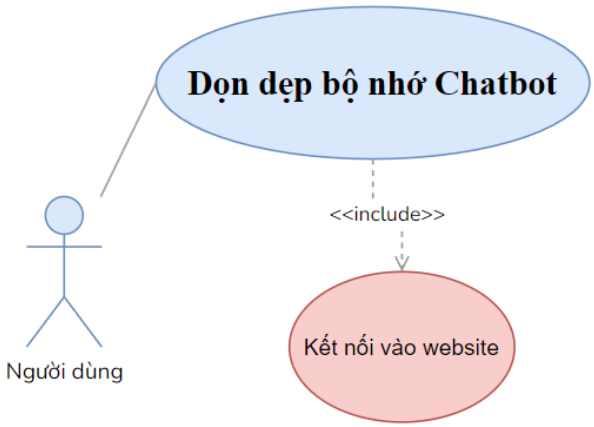
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Dọn dẹp bộ nhớ Chatbot cho phép người dùng dọn dẹp lịch sử chat.

Quy trình:

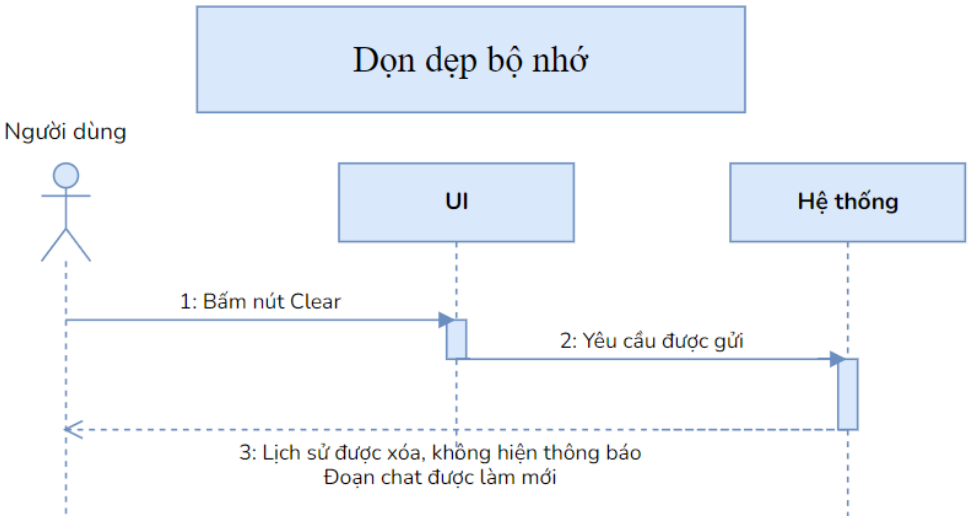
1. Người dùng bấm váo nút Clear.

b. Biểu đồ UseCase



Hình 2. 46 Usecase Dọn dẹp bộ nhớ Chatbot

c. Sơ đồ tuần tự



Hình 2. 47 Sơ đồ tuần tự Dọn dẹp bộ nhớ

3.1.3.5. Đặc tả chức năng Chọn dữ liệu để truy vấn

a. Đặc tả chức năng

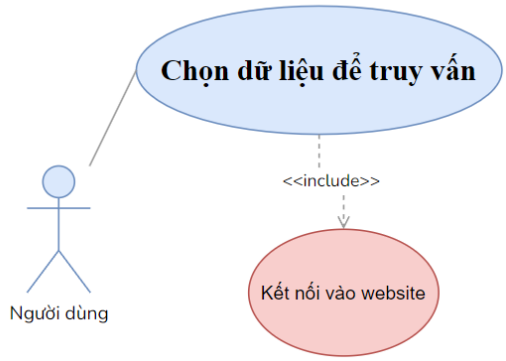
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Điều chỉnh độ chính xác và tin cậy của Chatbot cho phép người dùng chọn 1 trong 2 loại dữ liệu để chatbot sử dụng và truy vấn

Quy trình:

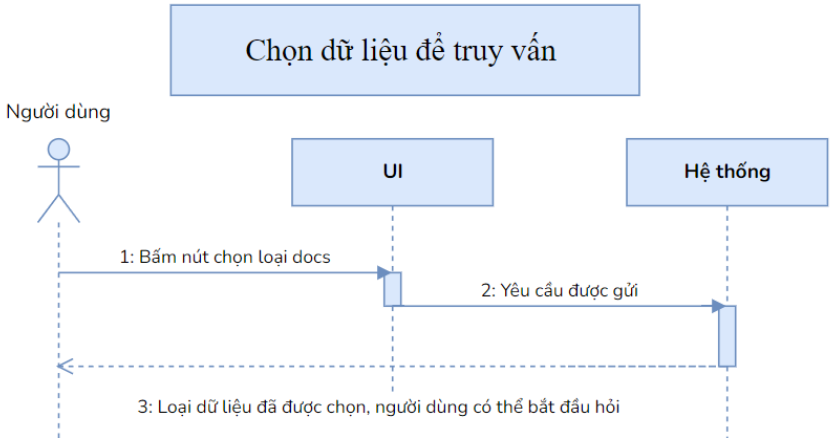
1. Người dùng bấm chọn danh sách ở RAG With.
2. Chọn Upload doc: Process for RAG hoặc Preprocessed doc

b. Biểu đồ UseCase



Hình 2. 48 Usecase Chọn dữ liệu để truy vấn

c. Sơ đồ tuần tự



Hình 2. 49 Sơ đồ tuần tự Chọn dữ liệu để truy vấn

3.1.3.6. Đặc tả chức năng Điều chỉnh độ chính xác và tin cậy của Chatbot

a. Đặc tả chức năng

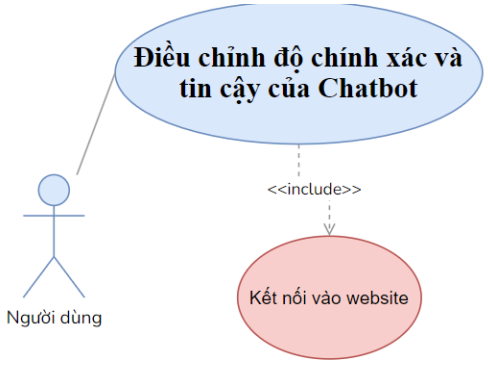
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Điều chỉnh độ chính xác và tin cậy của Chatbot cho phép người dùng điều chỉnh mức độ trên thanh slide bar của Temperature, top_k, top_p từ 0-1.0.

Quy trình:

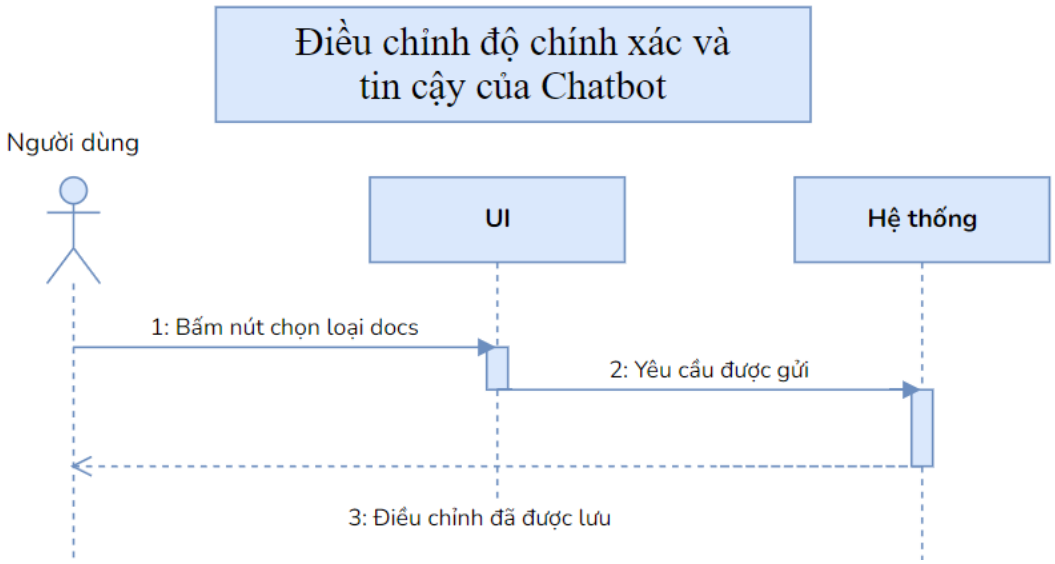
1. Người dùng kéo để điều chỉnh trên các thinh slide bar của Temperature, top_k, top_p từ 0-1.0 .

b. Biểu đồ UseCase



Hình 2. 50 Usecase Điều chỉnh độ chính xác và tin cậy

c. Sơ đồ tuần tự



Hình 2. 51 Sơ đồ tuần tự Điều chỉnh độ chính xác và tin cậy

3.1.3.7. Đặc tả chức năng Chatbot đưa ra câu trả lời

a. Đặc tả chức năng

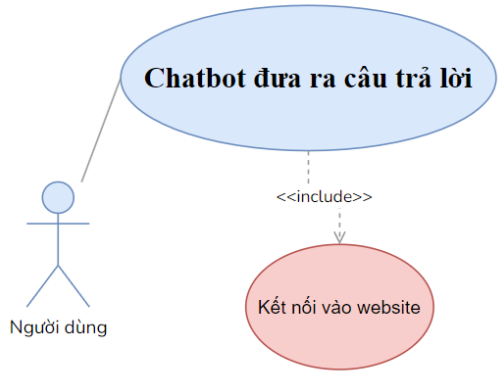
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Chatbot đưa ra câu trả lời cho phép người dùng nhận câu trả lời từ Chatbot.

Quy trình:

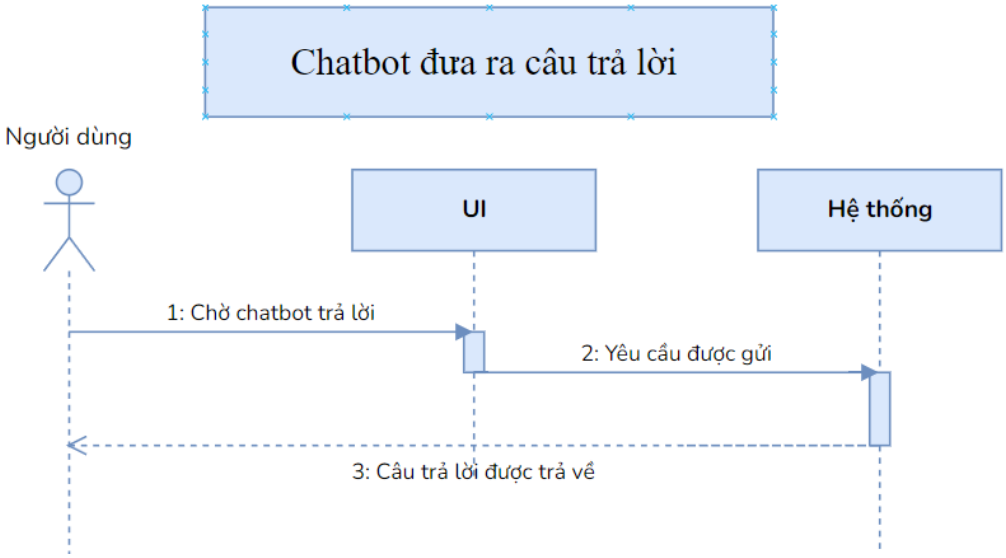
1. Chatbot hiển thị câu trả lời trên giáo diện người dùng

b. Biểu đồ UseCase



Hình 2. 52 Usecase Chatbot đưa ra câu trả lời

c. Sơ đồ tuần tự



Hình 2. 53 Sơ đồ tuần tự Chatbot đưa ra câu trả lời

3.2. Phân Tích Các Nguồn Dữ Liệu

Các nguồn dữ liệu có thể được sử dụng trong việc huấn luyện và kiểm thử Chatbot có thể bao gồm:

Tập dữ liệu PDF: Các tài liệu, bài báo, sách hoặc các nguồn dữ liệu văn bản khác có thể được sử dụng để huấn luyện mô hình Chatbot ở định dạng PDF.

Tài liệu Offlline: Tương tác với các tài liệu mà bạn đã tiền xử lý và vector hóa và có sẵn trong data. Những tài liệu này có thể được tích hợp một cách mượt mà vào các phiên trò chuyện của bạn.

Real-time Uploads: Dễ dàng tải lên các tài liệu PDF trong suốt các phiên trò chuyện của bạn thông qua giao diện tương tác với người dùng, cho phép chatbot xử lý và phản ứng với nội dung ngay lập tức.

3.3. Mô Tả Quá Trình Chuẩn Bị và Tiền Xử Lý Dữ Liệu:

Quá trình chuẩn bị và tiền xử lý dữ liệu đóng vai trò quan trọng trong việc phù hợp với mô hình Chatbot. Các bước có thể bao gồm:

3.3.1. Khởi tạo PrepareVectorDB

```python
def __init__(
        self,
        data_directory: str,
        persist_directory: str,
        chunk_size: int,
        chunk_overlap: int
) -> None:
    """
    Khởi tạo một trường hợp của PrepareVectorDB.

    """

    self.text_splitter = RecursiveCharacterTextSplitter(
        chunk_size=chunk_size,
        chunk_overlap=chunk_overlap,
        separators=["\n\n", "\n", " ", ""]
    )
    """ Có thể dùng CharacterTextSplitter, TokenTextSplitter, etc. tùy trường hợp"""
    self.data_directory = data_directory
    self.persist_directory = persist_directory
    self.embedding_function = HuggingFaceEmbeddings(
        model_name="BAAI/bge-large-en-v1.5",
        # cache_folder=os.getenv('SENTENCE_TRANSFORMERS_HOME')
    )
```

Hình 3. 1 Code hàm khởi tạo VectorDB

- Trong hàm khởi tạo, ta khởi tạo một instance của **PrepareVectorDB** với các thông số như thư mục chứa dữ liệu, thư mục để lưu trữ VectorDB, kích thước chunk và sự trùng lắp giữa các chunk.
- **RecursiveCharacterTextSplitter** được sử dụng để chia các tài liệu thành các phần nhỏ, sử dụng các separators như “\n\n”, “\n”, “ “, “”.
- **HuggingFaceEmbeddings** được sử dụng để nhúng văn bản thành vectơ, với mô hình đã được chỉ định.

3.3.2. Tải và chuẩn bị dữ liệu

```python
def __load_all_documents(self) -> List:
    """
    Tải tất cả các tài liệu từ thư mục hoặc các thư mục đã chỉ định.

    Returns:
        List: Danh sách các tài liệu đã được tải.
    """
    doc_counter = 0
    if isinstance(self.data_directory, list):
        print("Loading the uploaded documents...")
        docs = []
        for doc_dir in self.data_directory:
            docs.extend(PyPDFLoader(doc_dir).load())
            doc_counter += 1
        print("Number of loaded documents:", doc_counter)
        print("Number of pages:", len(docs), "\n\n")
    else:
        print("Loading documents manually...")
        document_list = os.listdir(self.data_directory)
        docs = []
        for doc_name in document_list:
            docs.extend(PyPDFLoader(os.path.join(
                self.data_directory, doc_name)).load())
            doc_counter += 1
        print("Number of loaded documents:", doc_counter)
        print("Number of pages:", len(docs), "\n\n")

    return docs
```

Hình 3. 2 Code hàm load documents

- Hàm **__load_all_documents** tải tất cả các tài liệu từ thư mục hoặc danh sách các thư mục đã được chỉ định và trả về một danh sách các tài liệu đã tải.

```python
def __chunk_documents(self, docs: List) -> List:
    """
    Chia nhỏ các tài liệu đã tải bằng text splitter đã chỉ định.

    Parameters:
        docs (List): The list of loaded documents.

    Returns:
        List: A list of chunked documents.

    """
    print("Chunking documents...")
    chunked_documents = self.text_splitter.split_documents(docs)
    print("Number of chunks:", len(chunked_documents), "\n\n")
    return chunked_documents
```

Hình 3. 3 Code hàm chia nhỏ dữ liệu

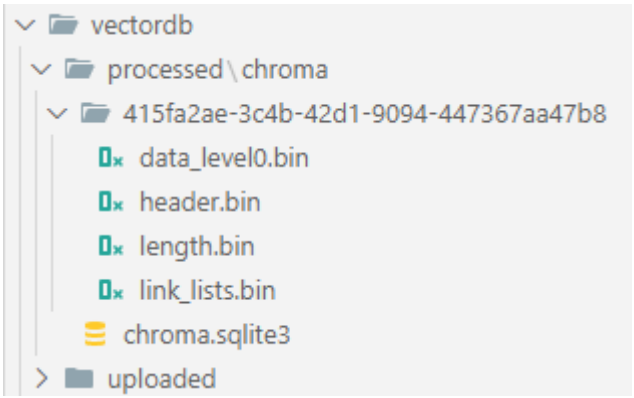
- Hàm **__chunk_documents** chia nhỏ các tài liệu đã tải thành các phần nhỏ sử dụng text splitter đã được chỉ định và trả về một danh sách các phần nhỏ của tài liệu.

3.3.3. Tạo VectorDB

```python
def prepare_and_save_vectordb(self):
    """
    Tải, chia nhỏ, và tạo một VectorDB với HuggingfaceEmbbeding, và lưu.

    Returns:
        Chroma: The created VectorDB.
    """
    docs = self.__load_all_documents()
    chunked_documents = self.__chunk_documents(docs)
    print("Preparing vectordb...")
    vectordb = Chroma.from_documents(
        documents=chunked_documents,
        embedding=self.embedding_function,
        persist_directory=self.persist_directory
    )
    print("VectorDB is created and saved.")
    print("Number of vectors in vectordb:",
          vectordb._collection.count(), "\n\n")
    return vectordb
```

Hình 3. 4 Code hàm lưu vào VectorStore

- Trong phương thức **prepare_and_save_vectordb**, các tài liệu được tải và chia nhỏ.
- Sau đó, VectorDB được tạo bằng cách sử dụng **Chroma.from_documents** với việc sử dụng embedding function đã được chỉ định.
- Cuối cùng, VectorDB được lưu trữ trong thư mục đã chỉ định.



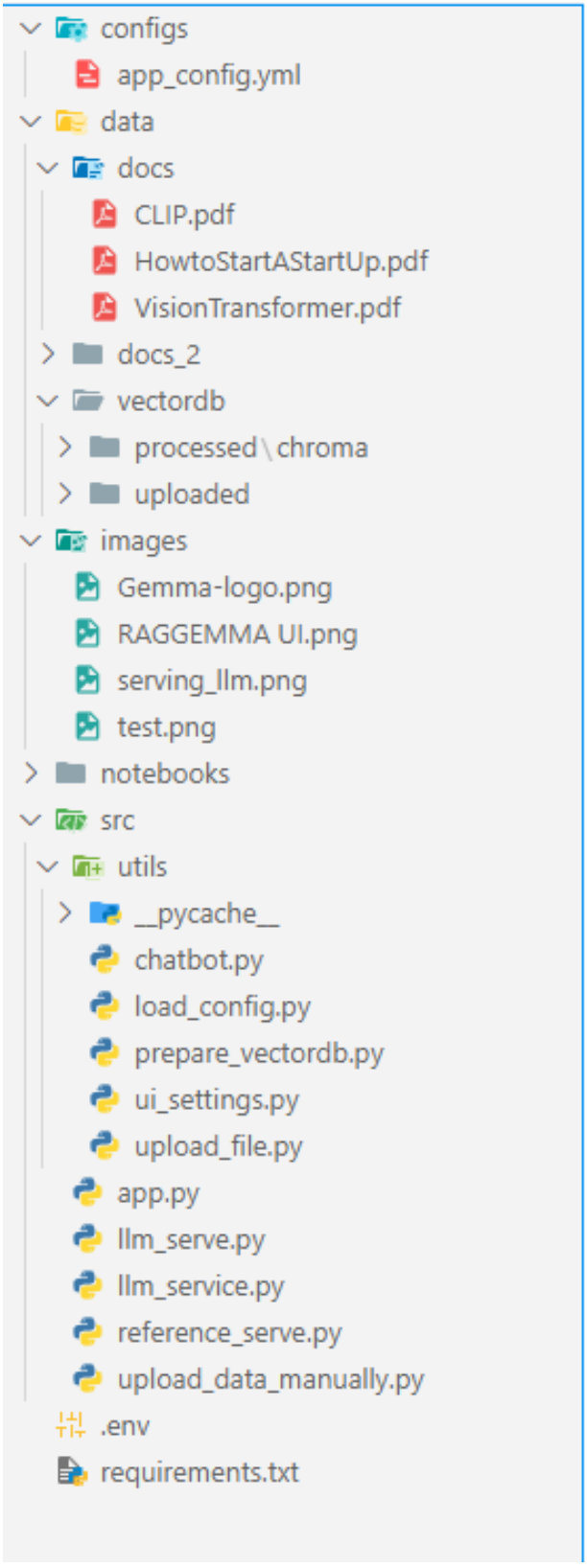
Hình 3. 5 Nơi lưu trữ dự liệu đã được chuẩn bị

3.3.4. Kết quả

VectorDB được trả về, sẵn sàng cho việc sử dụng trong việc tạo chatbot.

CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG

4.1. Cấu trúc dự án



Hình 4. 1 Cấu trúc dự án

Chú thích nội dung các thư mục:

1. **configs/app_config.yaml**: Thư mục này chứa tệp cấu hình cho chatbot, có thể bao gồm các thiết lập như cấu hình máy chủ, cấu hình cơ sở dữ liệu, cấu hình giao diện người dung, ex: directories, llm_config, splitter_config, retrieval_config, serve,…
2. **data/docs**: Thư mục này chứa các tài liệu đầu vào PDF mà chatbot sử dụng để xử lý và tạo ra dữ liệu hoặc vectơ. Các dữ liệu này được xử lí và lưu ở tiền xử lí giúp cho chatbot hoạt động tốt hơn với các dữ liệu này
3. **vectordb/processed/chroma**: Đây là thư mục lưu trữ VectorDB đã được xử lý và chuẩn bị để sử dụng trong chatbot.
4. **vectordb/uploaded**: Thư mục này có thể chứa các tệp tải lên từ người dùng hoặc dữ liệu mới để cập nhật VectorDB hoặc dữ liệu đầu vào. Các dữ liệu file PDF người dung tải lên từ giao diện sử dụng sẽ được xử lí, tách, chia nhỏ, lưu trữ vào đây
5. **images**: Thư mục này chứa các hình ảnh, biểu đồ hoặc tệp đa phương tiện khác được sử dụng trong dự án của bạn.
6. **src/utils**: Thư mục này chứa các file mã nguồn các hàm sử dụng trong chatbot RAG

Quá trình xây dựng ứng dụng Chatbot bắt đầu từ việc lập trình và triển khai các thuật toán và mô hình LLM cần thiết.

Sau đó, ứng dụng sẽ được phát triển sử dụng các công nghệ và thư viện như Flask để xây dựng một backend cho ứng dụng, và Gradio để tạo ra giao diện người dùng dễ sử dụng.

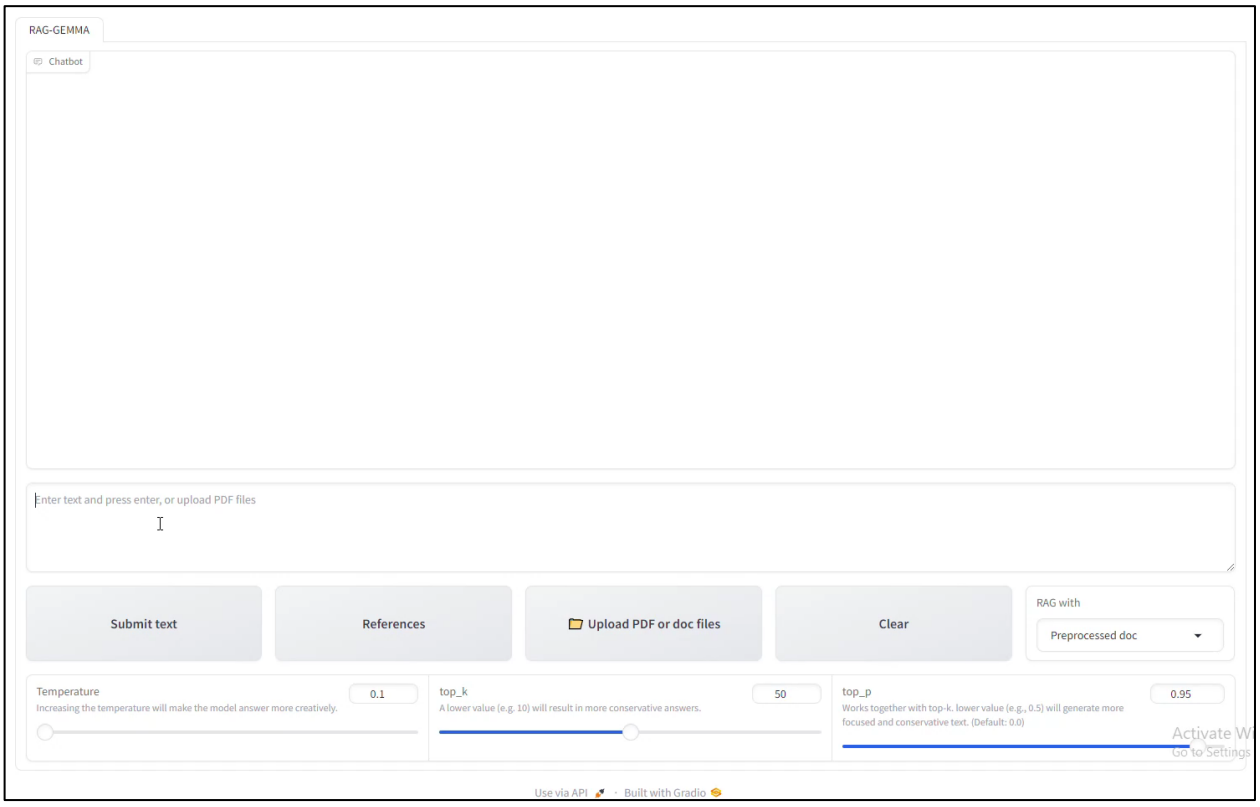
Quá trình triển khai ứng dụng sẽ bao gồm việc đưa ứng dụng lên một máy chủ hoặc một nền tảng đám mây để có thể truy cập từ mọi nơi trên internet.

4.2. Thiết kế giao diện và chức năng của Chatbot

Thiết kế giao diện người dùng của Chatbot sẽ tập trung vào việc tạo ra một trải nghiệm người dùng tốt nhất, bao gồm việc cung cấp một giao diện đơn giản và dễ sử dụng.

Giao diện sẽ cung cấp một hộp văn bản để người dùng nhập câu hỏi và một khu vực hiển thị các câu trả lời từ Chatbot.

Chức năng cơ bản của Chatbot bao gồm việc hiểu và phản hồi các câu hỏi của người dùng một cách chính xác và linh hoạt.



Hình 4. 2 Tổng quan giao diện

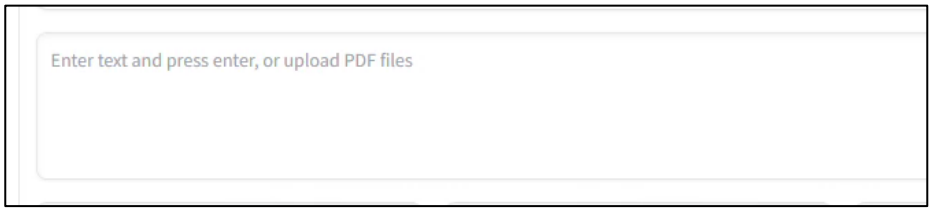
Các thành phần chính của Giao diện:

1. **Khung chứa Giao diện Tương tác**:

- Sử dụng một khung chứa để chứa toàn bộ giao diện tương tác, giúp tổ chức và hiển thị các thành phần một cách hợp lý trên giao diện người dùng.

2. **Các Phần Tương tác**:

- Sử dụng các phần tương tác của Gradio như Textbox, Button, Slider, Dropdown và UploadButton để cho phép người dùng nhập dữ liệu hoặc tương tác với ứng dụng của chúng ta.
- Textbox được sử dụng để nhập văn bản từ người dùng.

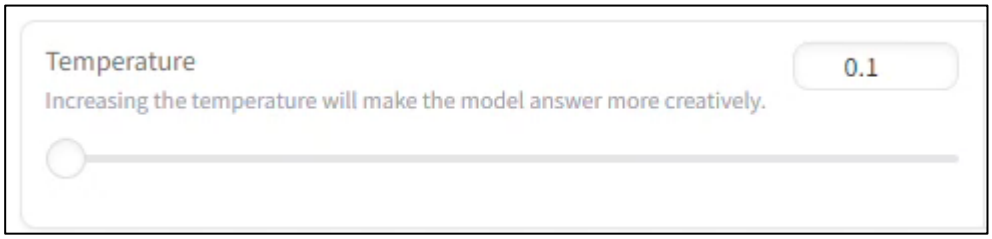


Hình 4. 3 Text box input

- UploadButton cho phép người dùng tải lên các tệp PDF hoặc doc.

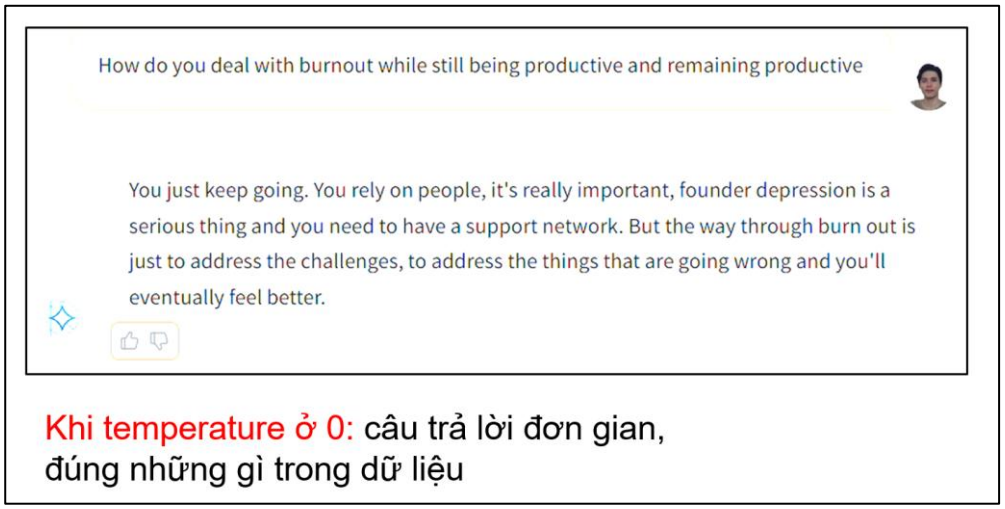
Hình 4. 4 Button upload

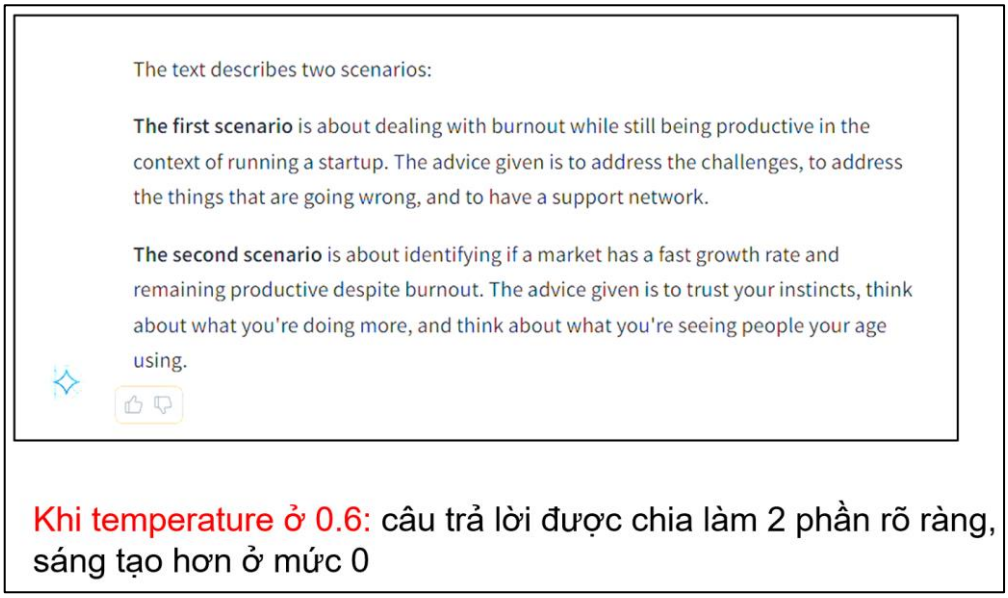
- Các Slider và Dropdown được sử dụng để chọn các tùy chọn và cài đặt cho mô hình.

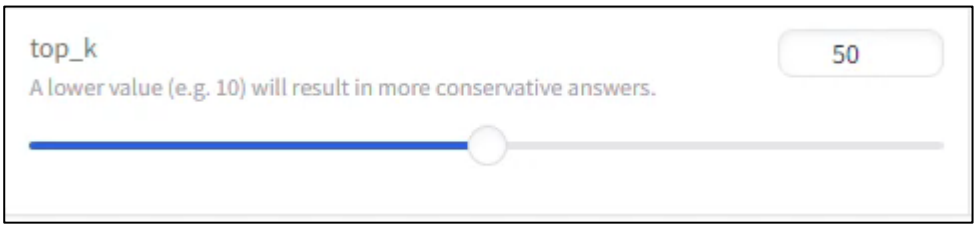


Hình 4. 5 Thanh chỉnh Temperature

Temperature: Chỉ số Temperature có trong các model LLM cho phép tùy chỉnh từ mức 0 đến 1.0. Tham số được sử dụng để kiểm soát mức độ động tính và sáng tạo của việc tạo câu trả lời hoặc văn bản từ mô hình, càng về 1 thì model sẽ đưa ra câu trả lời càng sáng tạo. Tuy nhiên điều này cũng làm cho model LLM cần nhiều token và thời gian để đưa ra câu trả lời.







Hình 4. 6 Thanh chỉnh top_k

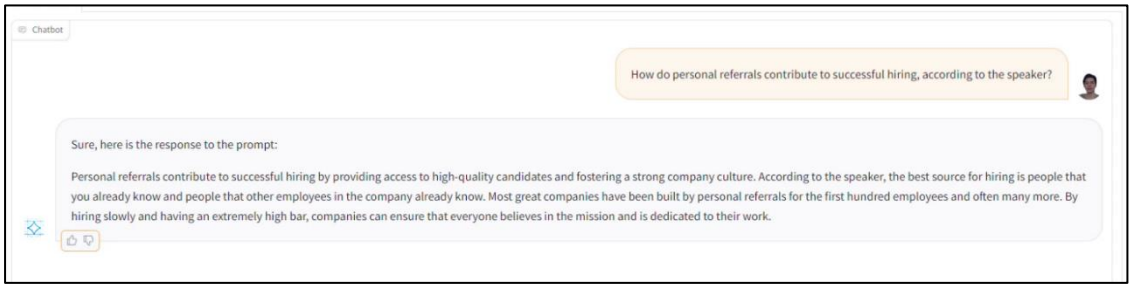
Top-k: Đây là phương pháp giới hạn số lượng từ vựng được xem xét khi mô hình quyết định từ vựng tiếp theo trong quá trình sinh văn bản. Top-k chỉ định rằng chỉ có top k từ vựng có xác suất cao nhất sẽ được xem xét. Các từ vựng còn lại nằm ngoài top-k sẽ bị loại bỏ. Việc này giúp giảm bớt sự không chắc chắn và làm cho mô hình tạo ra văn bản có tính nhất quán hơn. Có thể hiểu là chỉ số này càng thấp thì model sẽ tập trung lấy các từ có xác suất cao nhất trong khoảng token cho phép trong mỗi bước

Hình 4. 7 Thanh chỉnh top_p

Top-p (còn được gọi là nucleus sampling): Thay vì giới hạn số lượng từ vựng, top-p giới hạn tổng xác suất của các từ vựng được xem xét trong mỗi bước quyết định. Top-p chỉ định rằng chỉ có top từ vựng có tổng xác suất lớn hơn hoặc bằng p sẽ được xem xét. Các từ vựng có tổng xác suất dưới ngưỡng p sẽ bị loại bỏ, giúp giảm bớt sự không chắc chắn và làm cho văn bản sinh ra có tính nhất quán hơn. Nó được hoạt động chung với top-k giá trị càng thấp thì sẽ tạo ra câu trả lời tập trung hơn và thận trọng hơn.

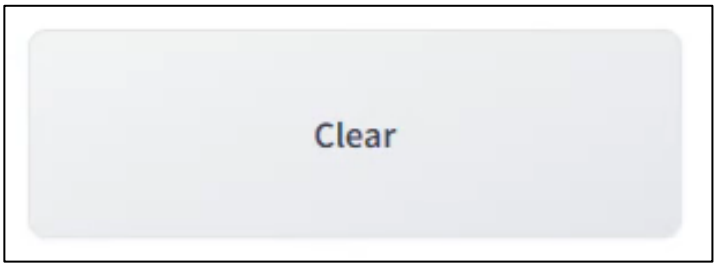
3. **Xử lý Dữ liệu và Hiển thị Kết quả**:

- Sử dụng các hàm xử lý dữ liệu từ các module và lớp khác nhau để xử lý dữ liệu nhập vào từ người dùng và trả về kết quả tương ứng.
- Kết quả được hiển thị cho người dùng thông qua các thành phần như Textbox và Chatbot.
- Cung cấp các phản hồi cho người dùng thông qua việc hiển thị thông báo và biểu tượng thích/không thích.



Hình 4. 8 Chat với Bot

- Người dùng có thể tương tác thêm bằng cách nhấp vào các nút hoặc thực hiện các hành động như là việc tải lên tệp hoặc xóa văn bản đã nhập.



Hình 4. 9 Button Clear

Việc phải nhúng (embedding) quá nhiều text, cũng như lưu trữ vào memory sẽ làm cho máy bị tràn Vram GPU, nên việc sử dụng nút Clear là cần thiết trong RAG Chatbot. Giúp làm sạch bộ nhớ cũng như giảm tránh tình trạng tràn VRAM trên GPU.

4.3. Tối ưu hóa và cải thiện hiệu suất Chatbot

Để tối ưu hóa và cải thiện hiệu suất của Chatbot, có thể áp dụng các phương pháp và kỹ thuật như:

Tối ưu hóa thuật toán và mô hình: Cải thiện hiệu suất của mô hình và thuật toán LLM thông qua việc điều chỉnh siêu tham số, tinh chỉnh kiến trúc mô hình, hoặc sử dụng các phương pháp học sâu tiên tiến.

Tối ưu hóa backend và giao diện người dùng: Tối ưu hóa mã nguồn và cấu trúc của backend để đảm bảo ứng dụng hoạt động mượt mà và nhanh chóng. Đồng thời, cải thiện giao diện người dùng để tăng tính thân thiện và trải nghiệm người dùng.

Thu thập và phản hồi từ người dùng: Liên tục thu thập phản hồi từ người dùng để cải thiện và tinh chỉnh Chatbot theo thời gian. Phản hồi từ người dùng sẽ giúp xác định các vấn đề và điểm yếu của Chatbot để có thể khắc phục và cải thiện.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết quả đạt được và những hạn chế

Trong đồ án này, một chatbot đã được phát triển dựa trên mô hình RAG-GEMMA và cung cấp giao diện người dùng tương tác thông qua Gradio. Chatbot này có khả năng trả lời các câu hỏi, cung cấp thông tin và tương tác với người dùng một cách linh hoạt và hiệu quả. Tuy nhiên, trong quá trình thực hiện dự án, một số hạn chế đã được phát hiện, bao gồm:

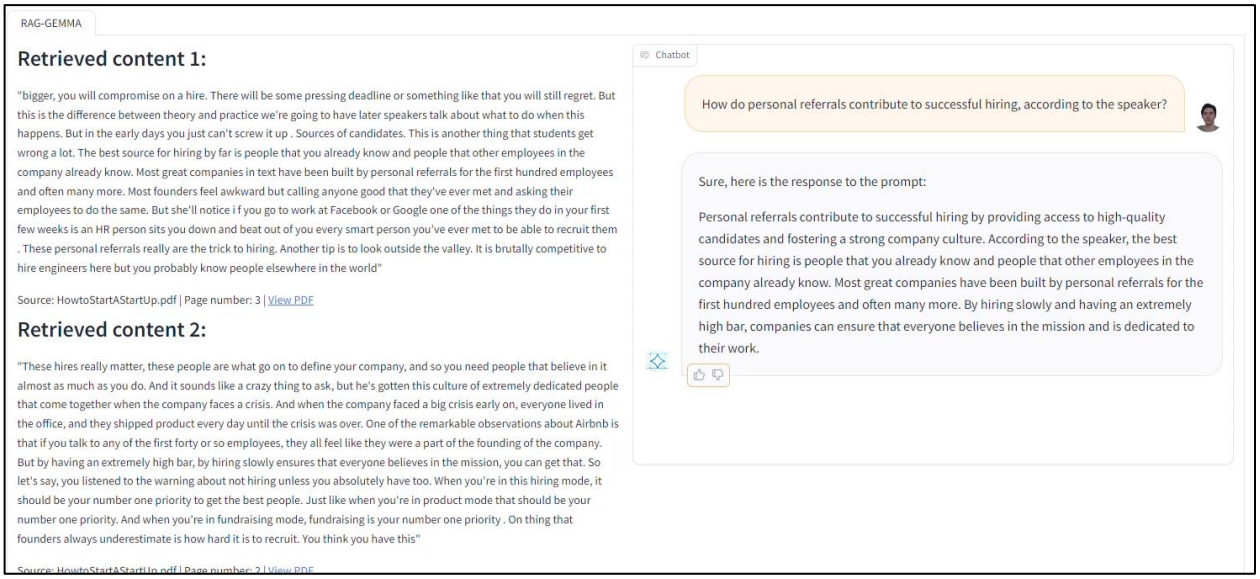
1. **Tốc độ xử lý chậm**: Một trong những thách thức chính là tốc độ xử lý của chatbot chưa đủ nhanh do hạn chế của việc chạy mô hình trên máy tính cá nhân. Để mô hình hoạt động hiệu quả, yêu cầu một lượng VRAM GPU mạnh mẽ. Dự án sử dụng mô hình Google Gemma 7B với 8.5 tỷ tham số, đây là một trong những mô hình ngôn ngữ lớn (LLM) mạnh mẽ nhất hiện nay. Mặc dù cung cấp kết quả chính xác và chi tiết, nhưng việc yêu cầu cấu hình phần cứng cao đã làm giảm tốc độ xử lý tổng thể.
2. **Sử dụng mô hình nhúng BAAI/bge-large-en**: Trong dự án, mô hình BAAI/bge-large-en được sử dụng để tạo embedding. Mô hình này, mặc dù hiệu quả trong việc chuyển đổi dữ liệu thành vector, nhưng cũng đòi hỏi tài nguyên tính toán đáng kể, góp phần làm chậm quá trình xử lý.
3. **Giới hạn trong xử lý định dạng file**: Hiện tại, chatbot RAG chỉ có khả năng xử lý các tệp tin PDF. Điều này giới hạn khả năng của hệ thống trong việc tương tác với dữ liệu từ các định dạng tệp tin khác như Word, Excel, hay các nguồn dữ liệu trực tuyến. Điều này có thể gây khó khăn khi người dùng muốn sử dụng các loại tài liệu khác ngoài PDF, giảm tính linh hoạt và tiện ích của chatbot.

Để phát triển một chatbot hiệu quả và nhanh chóng, việc nâng cấp phần cứng là cần thiết, bao gồm việc trang bị một GPU với VRAM lớn hơn để xử lý các mô hình phức tạp như Google Gemma 7B. Ngoài ra, việc tối ưu hóa mã nguồn và các thuật toán xử lý để tăng tốc độ phản hồi của hệ thống cũng là một giải pháp cần được xem xét.

Mặc dù gặp phải những hạn chế nhất định, dự án này vẫn mang lại nhiều giá trị, đặc biệt là trong việc khai thác sức mạnh của các mô hình ngôn ngữ lớn để cung cấp các câu trả lời chính xác và có giá trị cho người dùng. Khả năng tương tác thông qua giao diện Gradio tạo nên một trải nghiệm người dùng thân thiện, cho phép người dùng dễ dàng đặt câu hỏi và nhận được câu trả lời trong thời gian thực.

Để khắc phục những hạn chế hiện tại, các phiên bản tương lai của chatbot có thể mở rộng khả năng xử lý nhiều định dạng tệp tin hơn, cải thiện tốc độ xử lý thông qua tối ưu hóa phần cứng và phần mềm, và áp dụng các phương pháp tiên tiến trong xử lý ngôn ngữ tự nhiên. Bằng cách tiếp tục cải tiến và phát triển, chatbot này có thể trở thành một công cụ hữu ích và mạnh mẽ hơn, đáp ứng tốt hơn nhu cầu đa dạng của người dùng.

5.2. Đánh giá hiệu suất và kết quả:



Hình 5. 1 Đánh giá và kết quả

Mặc dù còn nhiều thiếu sót, nhưng RAG Gemma đã cho thấy những kết quả tích cực, bao gồm hoạt động ổn định, cung cấp câu trả lời đúng với câu hỏi, và truy xuất dữ liệu chính xác. Một số điểm nổi bật của RAG Gemma bao gồm:

1. **Trả lời chính xác**: Chatbot trả lời các câu hỏi có trong dữ liệu một cách chính xác, đảm bảo rằng người dùng nhận được thông tin phù hợp và đáng tin cậy.

2. **Truy xuất nội dung cụ thể**: Chatbot cho phép người dùng xem các đoạn nội dung mà nó sử dụng để truy xuất thông tin, bao gồm chỉ ra tệp dữ liệu và số trang cụ thể. Điều này giúp người dùng hiểu rõ nguồn gốc của câu trả lời và tăng tính minh bạch của hệ thống.
3. **Quản lý dữ liệu trò chuyện**: Người dùng có thể xóa dữ liệu trò chuyện thông qua chức năng Clear, giúp giảm tải VRAM và bộ nhớ, ngăn chặn tình trạng quá tải và duy trì hiệu suất hệ thống.
4. **Điều chỉnh độ sáng tạo**: Chatbot cho phép tinh chỉnh độ sáng tạo theo nhu cầu người dùng thông qua các thanh trượt điều chỉnh nhiệt độ (temp), top_k, và top_p. Điều này giúp tùy chỉnh mức độ sáng tạo trong câu trả lời, phù hợp với các tình huống khác nhau.
5. **Sử dụng miễn phí**: Chatbot sử dụng hoàn toàn miễn phí với mô hình ngôn ngữ lớn Google Gemma 7B và mô hình nhúng BAAI bge-large-en. Điều này mang lại lợi ích lớn cho người dùng, cho phép họ truy cập vào công nghệ tiên tiến mà không phải chịu thêm chi phí.

Nhìn chung, RAG Gemma đã chứng minh được hiệu quả của mình trong việc cung cấp các câu trả lời chính xác và hữu ích, cùng với khả năng truy xuất và hiển thị nội dung chi tiết. Những tính năng như quản lý dữ liệu trò chuyện và điều chỉnh độ sáng tạo giúp nâng cao trải nghiệm người dùng và tối ưu hóa hiệu suất hệ thống. Việc sử dụng miễn phí các mô hình tiên tiến cũng là một lợi thế lớn, cho phép nhiều người dùng tiếp cận và sử dụng công nghệ này mà không gặp rào cản tài chính.

Tuy vẫn còn những hạn chế cần khắc phục, nhưng RAG Gemma đã cho thấy tiềm năng lớn trong việc trở thành một công cụ hỗ trợ hiệu quả, cung cấp thông tin chính xác và nhanh chóng cho người dùng. Các cải tiến tương lai có thể tập trung vào việc nâng cao tốc độ xử lý, mở rộng khả năng xử lý nhiều định dạng tệp tin hơn và tiếp tục tối ưu hóa trải nghiệm người dùng

5.3. Hướng phát triển để cải thiện Chatbot trong tương lai

Trong dự án phát triển chatbot RAG Gemma, có một số hướng phát triển tiềm năng có thể được xem xét để nâng cao hiệu suất và tính linh hoạt của hệ thống:

1. **Mở rộng dữ liệu đầu vào**: Hiện tại, RAG Gemma chỉ xử lí được tệp PDF. Tuy nhiên, với sự linh hoạt và mạnh mẽ của LangChain, tôi dự định mở rộng khả năng xử lý để hỗ trợ các định dạng dữ liệu khác như Excel, văn bản đơn thuần (txt), và nhiều hơn nữa. Việc này sẽ mở ra khả năng truy cập và tương tác với một loạt các nguồn dữ liệu phong phú hơn, tăng tính ứng dụng và đa dạng cho chatbot.
2. **Tối ưu hóa mô hình**: Tôi sẽ tiến hành nghiên cứu và áp dụng các phương pháp tối ưu hóa mô hình để cải thiện khả năng hiểu và trả lời của chatbot. Điều này có thể bao gồm việc tinh chỉnh siêu tham số, tăng cường quá trình huấn luyện, và sử dụng các kỹ thuật mới nhất để cải thiện hiệu suất của hệ thống.
3. **Tinh chỉnh thời gian phản hồi**: Một mục tiêu quan trọng là tối ưu hóa thời gian phản hồi của chatbot để cải thiện trải nghiệm người dùng. Điều này có thể đạt được thông qua việc tối ưu hóa quá trình xử lý, tăng cường công nghệ cache, và sử dụng các kỹ thuật tự động hóa để tối ưu hóa thời gian phản hồi.
4. **Hỗ trợ đa ngôn ngữ**: Tôi dự định mở rộng chatbot để hỗ trợ nhiều ngôn ngữ khác nhau để đáp ứng nhu cầu của người dùng toàn cầu. Hiện tại, chatbot chỉ hỗ trợ tốt tiếng Anh do các mô hình tôi sử dụng được fine-tuning cho tiếng Anh. Việc mở rộng hỗ trợ ngôn ngữ sẽ mở ra cơ hội tiếp cận và tương tác với người dùng trên toàn thế giới.
5. **Thêm các cách input từ người dùng**: Tôi sẽ xem xét sử dụng whisper-large-v3 từ Huggingface để hỗ trợ việc nhập dữ liệu bằng giọng nói. Điều này sẽ tăng tính tiện lợi và đa dạng cho người dùng, cho phép họ tương tác với chatbot một cách tự nhiên hơn.
6. **Tích hợp học máy tăng cường**: Cuối cùng, tôi sẽ nghiên cứu và tích hợp các phương pháp học máy tăng cường để cải thiện khả năng tương tác và học hỏi của chatbot. Điều này có thể bao gồm việc sử dụng kỹ thuật tăng cường để tối ưu hóa quyết định và cải thiện khả năng tương tác của chatbot với người dùng.

Những hướng phát triển này có thể giúp cải thiện hiệu suất và độ chính xác của chatbot, đồng thời tạo ra một trải nghiệm tốt hơn cho người dùng trong tương lai. Bằng cách liên tục nghiên cứu và phát triển, tôi hy vọng rằng chatbot sẽ trở thành một công cụ hữu ích và đa dạng, đáp ứng được mọi nhu cầu của người dùng.

Tài liệu tham khảo

1. **LangChain**

   Automating Customer Service using LangChain:

   https://arxiv.org/pdf/2310.05421

   LangChain Docs:

   https://python.langchain.com/v0.1/docs/get_started/introduction/

   Khóa học từ Coursera:

   https://www.coursera.org/projects/langchain-chat-with-your-data-project

2. **Huggingface**

   Transformers: State-of-the-Art Natural Language Processing:

   https://arxiv.org/pdf/1910.03771

   Tài liệu từ trang chủ HuggingFace:

   https://huggingface.co/docs

   Tài liệu về Transformers của HuggingFace:

   https://huggingface.co/docs/transformers/index

3. **ChatGPT**

   https://chat.openai.com/

4. **Deeplearning.ai**

   https://learn.deeplearning.ai/

5. **Tài liệu về huggingface transformers :**

   https://huggingface.co/docs/transformers/index