TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA HÀ NỘI

XÂY DỰNG PDF CHATBOT TIẾNG VIỆT VỚI RAG

Sinh viên thực hiện: Đỗ Đức Vĩnh
Nguyễn Mạnh Tuấn

Hà Nội - 2024

LỜI NÓI ĐẦU

Với sự phát triển không ngừng của công nghệ thông tin, các ứng dụng trí tuệ nhân tạo (AI) ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực, và một trong những công cụ không thể không nhắc tới đó chính là các công cụ giao tiếp, công cụ chat, hay thường được gọi với cái tên "chatbot".

Chính vì lẽ đó, đề tài "Xây dựng PDF chatbot Tiếng Việt" được thực hiện với mục tiêu tạo một Chatbot cho phép người dùng truyền vào các văn bản PDF bất kì (tiếng anh hoặc tiếng việt) sau đó thực hiện hỏi đáp các vấn đề trong PDF đó. Công cụ này giúp người dùng có thể nhanh chóng nắm bắt, tìm kiếm vị trí, thông tin cần tìm trong PDF, cũng như chuyển đổi những bước tìm kiếm thông tin khô khan và tốn thời gian, trở thành một cuộc hội thoại thú vị và vui vẻ.

Trên phương diện cá nhân, việc tham gia vào dự án này đã giúp nhóm tiếp cận và nắm bắt những công nghệ tiên tiến nhất trong lĩnh vực trí tuệ nhân tạo, như Retrieval-Augmented Generation (RAG), Large Language Model (LLM), framework Langchain ngoài ra còn có Prompting, và Vector Searching. Quá trình nghiên cứu và ứng dụng những kỹ thuật này không chỉ mở rộng kiến thức chuyên môn mà còn trang bị cho nhóm những kỹ năng thực hành quan trọng, tạo nền tảng vững chắc cho sự nghiệp sau này.

Về mặt cộng đồng, dự án này không chỉ dừng lại ở việc xây dựng một công cụ hữu ích mà còn đóng góp những hướng đi và logic mới trong việc phát triển các hệ thống chatbot. Những công nghệ và kỹ thuật được áp dụng trong chatbot này có thể trở thành nền tảng cho các ứng dụng tương lai, góp phần nâng cao hiệu suất và tính năng của các công cụ giao tiếp tự động trong tương lai.

Mục lục

Danh sách từ viết tắt

AI Artificial Intelligence

API Application Programming Interface

FAISS Facebook AI Similarity Search

IVF Inverted File

LLM Large Language Model

PQ Product Quantization

RAG Retrieval-Augmented Generation

Danh sách hình vẽ

Chương 1

Tổng quan về đề tài

1.1 Mục tiêu và ý nghĩa

Đề tài "Xây dựng PDF chatbot Tiếng Việt" được thực hiện nhằm áp dụng những kiến thức và kỹ thuật trong lĩnh vực xử lý ngôn ngữ tự nhiên để xây dựng một hệ thống chatbot có khả năng tương tác và trích xuất thông tin từ các tài liệu PDF tiếng Việt. Mục tiêu cốt yếu của đề tài là tạo ra một công cụ giúp người dùng nhanh chóng tìm kiếm và nắm bắt nội dung chính của các tài liệu PDF mà họ quan tâm. Nhờ đó, người dùng có thể tiết kiệm thời gian và công sức trong việc tìm kiếm thông tin, đồng thời nâng cao hiệu quả học tập và làm việc.

Đề tài có ý nghĩa không chỉ về mặt nghiên cứu mà còn có thể ứng dụng rộng rãi trong các lĩnh vực như giáo dục, kinh doanh. Hệ thống chatbot sẽ hỗ trợ người dùng trong việc tra cứu tài liệu, giúp tiếp cận thông tin chính xác một cách nhanh chóng và hiệu quả. Điều này mở ra nhiều cơ hội cho việc cải thiện năng suất làm việc, tối ưu hóa thời gian học tập, nghiên cứu và nâng cao trải nghiệm người dùng trong việc truy xuất thông tin từ các tài liệu PDF tiếng Việt.

1.2 Hướng tiếp cận

Trong đề tài này, nhóm đề cập đến một cách tiếp cận kết hợp xử lý văn bản và trí tuệ nhân tạo, tập trung vào việc trích xuất và phân tích thông tin từ các tài liệu PDF tiếng Việt. Nguyên nhân cho cách tiếp cận này bao gồm:

- **Nhu cầu thực tế:** Người dùng thường cần tìm kiếm nhanh chóng và chính xác thông tin quan trọng từ các tài liệu PDF, và việc có một chatbot hỗ trợ trong việc tìm kiếm, trích xuất và tổng hợp nội dung là rất cần thiết. Điều này giúp tiết kiệm thời gian và công sức cho người dùng qua đó làm tăng hiệu quả trong học tập và làm việc.
- **Sự phát triển của công nghệ AI:** Các mô hình ngôn ngữ lớn đã đạt được những tiến bộ đáng kể trong việc hiểu và xử lý ngôn ngữ tự nhiên, cho phép phân tích sâu hơn về ngữ nghĩa và ngữ cảnh của văn bản.
- **Khả năng tích hợp:** Các framework hiện đại cho phép kết hợp liền mạch giữa các công nghệ xử lý ngôn ngữ, truy xuất thông tin và tạo phản hồi, tạo ra một hệ thống toàn diện và hiệu quả.

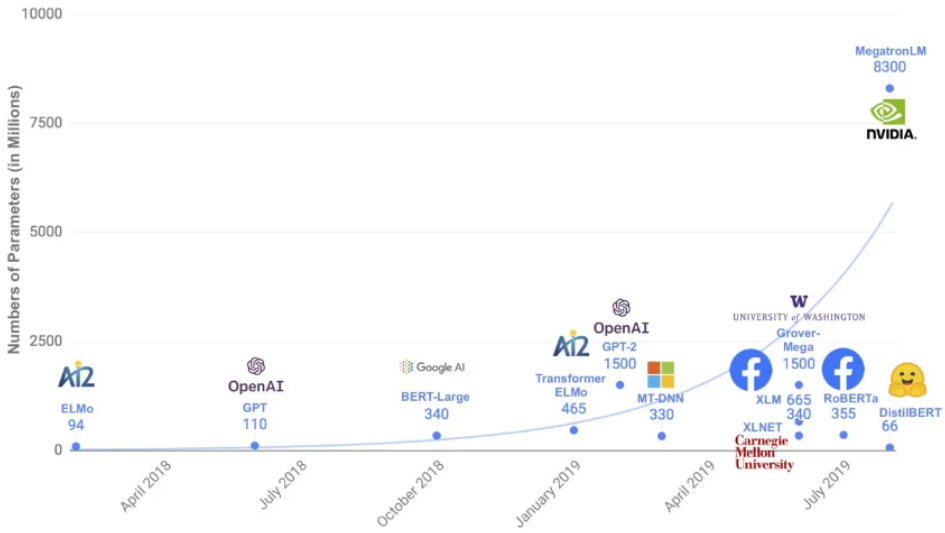
1.3 Công nghệ và kỹ thuật sử dụng

Trong báo cáo này, nhóm áp dụng một cách tiếp cận tích hợp nhằm phát triển hệ thống chatbot "Xây dựng PDF chatbot Tiếng Việt". Trong phần này, nhóm sẽ chỉ đề cập đến các công nghệ, kỹ thuật chính được sử dụng, bao gồm LLM, Langchain và RAG.

1.3.1 Mô hình ngôn ngữ lớn - LLM

Mô hình Ngôn ngữ Lớn (LLM) là một loại mô hình trí tuệ nhân tạo được thiết kế để xử lý và hiểu ngôn ngữ tự nhiên ở một cấp độ cao. Các mô hình này thường được đào tạo trên một tập dữ liệu khổng lồ được thu thập từ Internet, sách, báo và xử lý chất lượng, giúp cho chúng có khả năng hiểu biết sâu sắc về văn bản và có khả năng thực hiện nhiều tác vụ xử lý ngôn ngữ tự nhiên phức tạp như đọc, dịch, phân loại, tóm tắt văn bản, sáng tạo nội dung mới và trả lời câu hỏi.

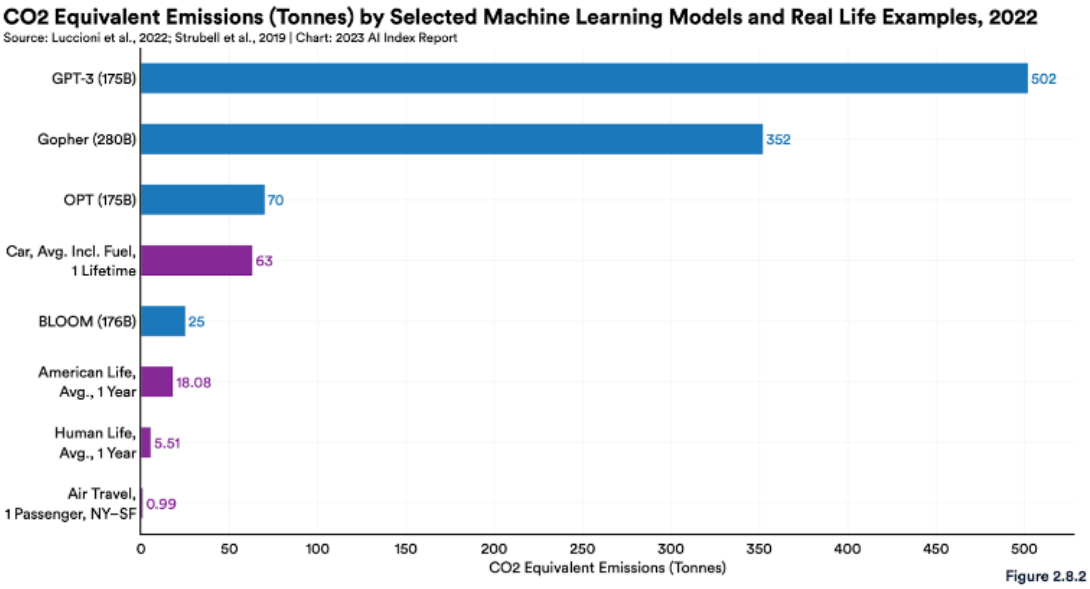
Chúng được gọi là mô hình ngôn ngữ lớn bởi vì kích thước của chúng, cả về số lượng tham số của mô hình (parameters) và dữ liệu được sử dụng để huấn luyện. Chính vì vậy, việc đào tạo, tinh chỉnh các LLM này thường rất tốn kém thời gian và tài nguyên. Dưới đây là minh họa về kích thước và tài nguyên tiêu tốn để huấn luyện các mô hình ngôn ngữ lớn:



Hình 1.1 Kích thước một số mô hình ngôn ngữ đến năm 2019

Đây là dữ liệu đã dừng tại tháng 7 năm 2019. Sau đó, rất nhiều mô hình lớn hơn đã ra đời, ta có thể kể đến OpenAI GPT-3 (170B), hay META LLama 3.1 với các phiên bản 8B, 70B và mới đây nhất vào tháng 4 năm 2024 là LLama 3.1 405B, một mô hình có khả năng xử lý ngôn ngữ tự nhiên hoàn hảo và hoàn toàn open-source.

Dưới đây là một vài minh họa về lượng khí thải CO2 tương đương (tính bằng tấn) do một số mô hình học máy lớn và các hoạt động thực tế trong đời sống phát thải vào năm 2022:



Hình 1.2 So sánh lượng CO2 phát thải khi huấn luyện LLM

Hình ảnh này cho thấy các mô hình ngôn ngữ lớn như GPT-3 và Gopher có lượng khí thải CO2 rất cao, vượt xa nhiều hoạt động trong đời sống hàng ngày. Do vậy, chúng ta nên tận dụng sức mạnh của các pretrained-model và transfer learning, sẽ giúp giảm chi phí cũng như lượng tài nguyên để huấn luyện lại mô hình từ đầu đi rất nhiều.

Tuy nhiên, trên thực tế, khi sử dụng trong các sản phẩm, một hướng đi có lẽ thích hợp hơn đó chính là sử dụng các API LLM được cung cấp sẵn. Hiện nay trên thị trường có rất nhiều API như vậy với giá cả phải chăng, có thể giúp các lập trình viên nhanh chóng tích hợp những công cụ chat AI và sản phẩm của mình. Trong đề tài này, nhóm đã sử dụng Gemini API, một công cụ tiên tiến do Google phát triển.

Đôi nét về Google Gemini

Gemini là một mô hình ngôn ngữ lớn của Google AI. Nó được đào tạo trên một tập dữ liệu văn bản và mã khổng lồ, cho phép nó giao tiếp và tạo ra văn bản giống con người để đáp lại các câu hỏi và yêu cầu của con người. Ngoài ra, Google cũng cung cấp API để các nhà phát triển có thể dễ dàng sử dụng công cụ này trong ứng dụng của mình. Một điều đặc biệt là bạn hoàn toàn có thể sử dụng Gemini API ở mức độ miễn phí (sẽ có những hạn chế), tuy nhiên điều đó cho thấy sự quan tâm của Google dành cho các nhà nghiên cứu, nhà phát triển có cơ hội tiếp cận đến các mô hình ngôn ngữ lớn hiện đại nhất. 2 phiên bản mới nhất của Gemini có thể kể đến là:

- Gemini 1.5 Pro: là một mô hình đa phương thức thực sự, có khả năng xử lý đồng thời văn bản, hình ảnh, video và âm thanh một cách liền mạch. Điều này có nghĩa là Gemini 1.5 Pro được thiết kế để hoạt động tốt trong nhiều loại dữ liệu khác nhau, từ dữ liệu văn bản đến dữ liệu hình ảnh và âm thanh.
- Gemini 1.5 Flash: vẫn là một mô hình đa phương thức nhưng nó được tối ưu hóa về mặt tốc độ, có thể "hi sinh độ thông minh" so với phiên bản pro.

Bên cạnh những ưu điểm nổi trội, LLM cũng có thể có những nhược điểm, đặc biệt là **hiện tưởng ảo giác (hallucination)**. Đây là hiện tượng các mô hình sinh ra những câu văn nghe rất trôi chảy và hợp lý nhưng lại không chính xác, có thể đánh lừa người dùng tin vào thông tin sai lệch. Vì vậy khi làm việc với LLM, cần chú ý về phần kiểm chứng thông tin. Ngoài ra, LLM còn một số nhược điểm khác:

- Có thể đòi hỏi tài nguyên phần cứng cao nếu muốn tinh chỉnh lại mô hình để tối

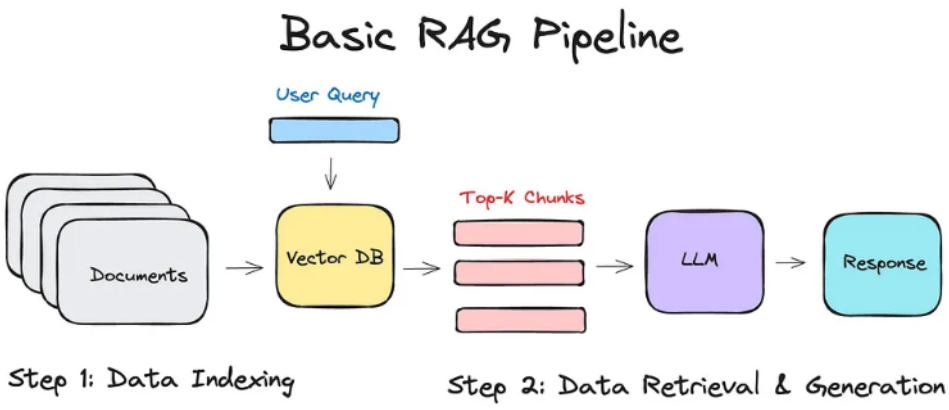
ưu hoá cho các tác vụ cụ thể.

- Hiện tại API Key mà nhóm đang sử dụng thuộc loại "Free of charge" nên có rất nhiều những giới hạn về số phản hồi trên phút, số phản hồi trong ngày. Bản trả phí sẽ mang lại chất lượng mô hình vượt trội và trải nghiệm tốt hơn rất nhiều.
- Về tính thông minh của mô hình: Gemini-1.5-flash được thiết kế để tạo phản hồi nhanh chóng cho nên nếu xét về độ thông minh, nó không thể bằng Gemini-1.5-pro. Chính vì vậy với những prompt hoặc truy vấn quá phức tạp.

Trong đề tài này, nhóm đã lựa chọn sử dụng Gemini-1.5-flash, đây là mô hình có tốc độ xử lý nhanh nhất hiện tại của Gemini, cho phép chatbot phản hồi gần như tức thì, nâng cao trải nghiệm người dùng. Dù nó không thông minh bằng phiên bản pro nhưng với sự kết hợp thông tin từ RAG, nhiệm vụ của LLM trở nên đơn giản hơn rất nhiều.

1.3.2 Retrival-Augmented-Generation - RAG

RAG là một kỹ thuật giúp nâng cao khả năng của mô hình tạo sinh kết hợp với tri thức bên ngoài. Phương pháp này thực hiện bằng cách truy xuất thông tin liên quan từ kho tài liệu (còn gọi là kho tri thức) và sử dụng chúng cho quá trình sinh câu trả lời dựa trên LLMs. Trong hệ thống này, RAG đóng vai trò quan trọng trong việc đảm bảo độ chính xác và tính phù hợp của thông tin được cung cấp. RAG cho phép chatbot không chỉ trả lời các câu hỏi dựa trên kiến thức tổng quát của mô hình, mà còn có thể trích dẫn và tham chiếu trực tiếp đến các phần cụ thể trong tài liệu PDF, nâng cao độ tin cậy của thông tin. Dưới đây là minh họa một luồng hoạt động của RAG:



Hình 1.3 Basic RAG Pipeline

Hệ thống RAG là một bước tiến lớn trong lĩnh vực tạo công cụ giao tiếp, lợi ích mà nó mang lại là rất nhiều, có thể kể đến như:

- **Giảm thiểu ảo giác (hallucination):** Vì bản thân hệ thống RAG hướng đến việc cung cấp thêm thông tin cho LLM để nó tổng hợp và trả lời chính xác hơn.
- **Giảm thiểu chi phí:** Với RAG, ta có thể tận dụng khả năng hiện có của LLM mà không cần tinh chỉnh mô hình. Nó cũng dễ dàng mở rộng dữ liệu khi cần.
- **Tính bảo mật:** Với việc tự thiết kế hệ thống RAG, chúng ta sẽ hoàn toàn kiểm soát được nơi chứa dữ liệu, việc đọc và ghi dữ liệu, qua đó đảm bảo tính bảo mật hơn.
- **Có thể tạo ra câu trả lời đúng và tự nhiên:** Với sức mạnh của LLM, câu trả lời sẽ trở nên tự nhiên và vẫn đảm bảo đúng thông tin có trong dữ liệu.
- **Xử lý tốt các đề tài khó:** Hệ thống RAG có thể vượt trội hơn các công cụ chat như ChatGPT, Gemini bởi vì nó có thể trả lời được cả những câu hỏi đặc thù chuyên ngành, miễn là được cung cấp tài liệu. Có thể kể đến như tài liệu mật của các công ty, nội quy công ty, thứ mà ta sẽ không thể truy vấn được bằng các công cụ chat thông thường (mà vẫn đảm bảo tính bảo mật).

Tuy nhiên, có một điểm đặc biệt cần lưu ý là hiệu suất của hệ thống RAG phụ thuộc **phần lớn** vào quá trình Retriever (tìm ra context phù hợp với truy vấn của người dùng). Do đó theo các chuyên gia, thời gian phát triển một hệ thống RAG chủ yếu rơi vào việc tạo ra các Embedding, cải thiện các phương pháp tìm kiếm. Prompting cũng là một vấn đề đối với hệ thống này, để câu trả lời có thể đúng và đầy đủ với mong muốn của người phát triển cũng như người dùng, các kỹ sư cần phải tinh chỉnh prompt rất kỹ lưỡng.

1.3.3 Framework Langchain

Langchain là một framework mạnh mẽ cho phép xây dựng các ứng dụng AI phức tạp bằng cách kết nối và điều phối nhiều thành phần khác nhau. Trong dự án này, Langchain được sử dụng để tạo ra một luồng làm việc liền mạch, bao gồm việc xử lý tài liệu PDF, tạo và lưu trữ vector database, trích xuất thông tin, và tích hợp với mô hình Gemini-1.5-flash để sinh ra câu trả lời tương tác với người dùng.

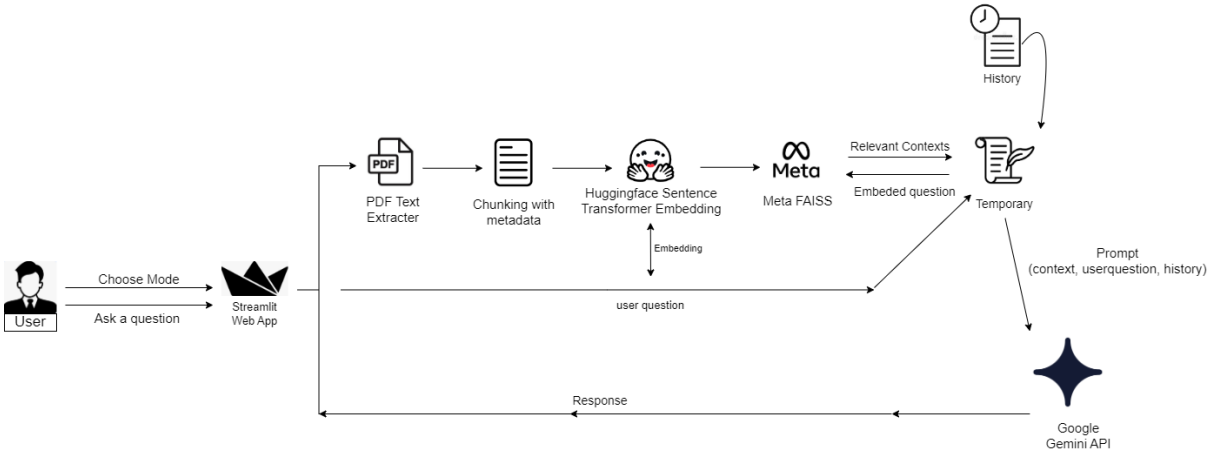
- **Ưu điểm của Langchain:**
  - Tính linh hoạt cao, dễ dàng tích hợp nhiều thành phần
  - Cung cấp các công cụ mạnh mẽ để xây dựng ứng dụng AI phức tạp
  - Hỗ trợ quản lý bộ nhớ và ngữ cảnh hội thoại
  - Dễ mở rộng, có thể tích hợp thêm các mô hình mới trong tương lai
- **Nhược điểm:**
  - Cần thời gian để làm quen với cách sử dụng và các tính năng của Langchain
  - Hiệu suất có thể bị ảnh hưởng nếu các mô hình công nghệ tích hợp hoạt động không tốt
- **Lý do lựa chọn:** Tính linh hoạt của Langchain cho phép xây dựng một hệ thống tùy chỉnh cao, dễ dàng tích hợp Gemini-1.5-flash và RAG. Khả năng quản lý ngữ cảnh hội thoại là ưu điểm nổi bật, giúp chatbot duy trì tính nhất quán trong các cuộc trò chuyện dài.

Chương 2

Triển khai đề tài

2.1 Thiết kế hệ thống RAG

Dưới đây là hình minh họa hệ thống RAG mà nhóm đã thiết kế và sử dụng trong đề tài hiện tại:



Hình 2.1 RAG System Flow

Giải thích các thành phần: vì có rất nhiều thành phần nên nhóm sẽ trình bày chi tiết về các thành phần trong từng mục tiếp theo của Chương 2. Trong mục này, nhóm chỉ mô tả một cách ngắn gọn về luồng hoạt động của hệ thống:

- **Tương tác với người dùng:** Người dùng chọn chế độ và gửi câu hỏi thông qua ứng dụng web Streamlit. Có 2 chế độ là Chat thông thường và PDF Query. Trong chế độ chat thông thường, người dùng sẽ trò chuyện trực tiếp với Gemini mà không cần

bất cứ việc truy xuất hay tìm kiếm nào.

- **Xử lý PDF:** Hệ thống trích xuất văn bản từ tài liệu PDF và chia nhỏ nó thành các đoạn kèm theo metadata. Các metadata này quan trọng trong việc cải thiện kết quả cũng như trích dẫn nguồn.
- **Embedding:** Các đoạn văn bản được chuyển đổi thành dạng vector sử dụng mô hình Huggingface Sentence Transformer. Mô hình nhóm nhóm đang sử dụng trong bài là: huggingface.co/hiieu/halong_embedding.
- **Truy xuất ngữ cảnh:** Các đoạn văn sau khi được chunking sẽ được embed và lưu trữ với FAISS. Sau đó khi câu truy vấn được đưa vào, dạng embedding của câu truy vấn đó sẽ được sử dụng để tìm kiếm tương đồng trong cơ sở dữ liệu vector, qua đó trả ra chunk liên quan nhất.
- **Tạo prompt:** Các ngữ cảnh liên quan, cùng với câu hỏi và lịch sử tương tác của người dùng, được sử dụng để tạo một prompt hướng dẫn cho LLM cách trả lời câu hỏi đúng với mong muốn.
- **Tạo phản hồi**: Prompt này được gửi đến Google Gemini API để tạo ra phản hồi và gửi lại cho người dùng thông qua giao diện streamlit.

2.2 Xử lý tài liệu PDF và văn bản

2.2.1 Đọc tài liệu với PyPDFLoader

PyPDFLoader được sử dụng như một công cụ chính để đọc và trích xuất nội dung từ các file PDF. Công cụ này được lựa chọn vì khả năng xử lý hiệu quả các tài liệu PDF phức tạp, bao gồm cả những tài liệu có nhiều trang và định dạng đa dạng. PyPDFLoader cung cấp các chức năng quan trọng sau:

- **Trích xuất văn bản:** Khả năng trích xuất nội dung văn bản từ mọi trang của tài liệu PDF, bao gồm cả những trang có cấu trúc phức tạp như bảng biểu,...
- **Xử lý metadata:** Trích xuất thông tin metadata của tài liệu, cung cấp dữ liệu bổ sung về nguồn gốc và đặc điểm của tài liệu.

- **Tích hợp linh hoạt:** Khả năng tích hợp tốt với các thư viện RecursiveCharacterTextSplitter, FAISS, tạo điều kiện thuận lợi cho các bước xử lý tiếp theo.

Đặc biệt, thư viện này cũng được tích hợp bởi Langchain, qua đó có thể dễ dàng sử dụng và đồng nhất về mặt coding trong chương trình.

2.2.2 Kiểm tra trùng lặp

Để tối ưu hóa hiệu suất và tránh xử lý lặp lại các tài liệu đã được phân tích trước đó, nhóm đã áp dụng kỹ thuật hash kết hợp với cơ chế lưu trữ và kiểm tra:

- **Tính toán Hash:** Mỗi tài liệu PDF được tính toán giá trị hash duy nhất sử dụng thuật toán SHA-256. Phương pháp này đảm bảo mỗi tài liệu có một định danh duy nhất, không phụ thuộc vào tên file hay vị trí lưu trữ.
- **Lưu trữ Hash:** Giá trị hash được lưu trữ trong một file JSON. Cơ chế lưu trữ này cho phép truy xuất và so sánh nhanh chóng.
- **Kiểm tra Trùng lặp:** Trước khi xử lý một tài liệu mới, hệ thống so sánh giá trị hash của nó với danh sách các hash đã lưu trữ. Nếu tìm thấy trùng khớp, tài liệu được coi là đã xử lý và bỏ qua để tránh các công việc sau này bị lặp lại.

Việc kiểm tra trùng lặp này là rất quan trọng, trong bối cảnh cơ sở dữ liệu vector sẽ chỉ được tạo ra mỗi khi người dùng đưa PDF vào (tức là gần như real-time), việc thực hiện kiểm tra trùng lặp sẽ giúp giảm thiểu thời gian xử lý các tài liệu đã có trong cơ sở dữ liệu.

2.2.3 Phân đoạn nội dung (chunking)

Sau khi trích xuất nội dung từ tài liệu PDF, bước tiếp theo trong quy trình xử lý là phân đoạn văn bản thành các đơn vị nhỏ hơn, có ý nghĩa. Nhóm áp dụng phương pháp RecursiveCharacterTextSplitter, một kỹ thuật tiên tiến trong lĩnh vực xử lý ngôn ngữ tự nhiên, để thực hiện quá trình này. Phương pháp này có một số đặc điểm và ưu điểm nổi bật:

- **Phân tách đa cấp:** RecursiveCharacterTextSplitter sử dụng một danh sách các ký tự phân tách được sắp xếp theo thứ tự ưu tiên: ["\n\n", "\n", " ", ".", "!", "?", ""].

Cách tiếp cận này cho phép phân đoạn văn bản một cách linh hoạt, tôn trọng cấu trúc tự nhiên của nội dung. Hệ thống sẽ ưu tiên chia tại các ranh giới đoạn văn (được đánh dấu bởi "\n\n"), sau đó là các dòng riêng lẻ, khoảng trắng, và cuối cùng là các dấu câu.

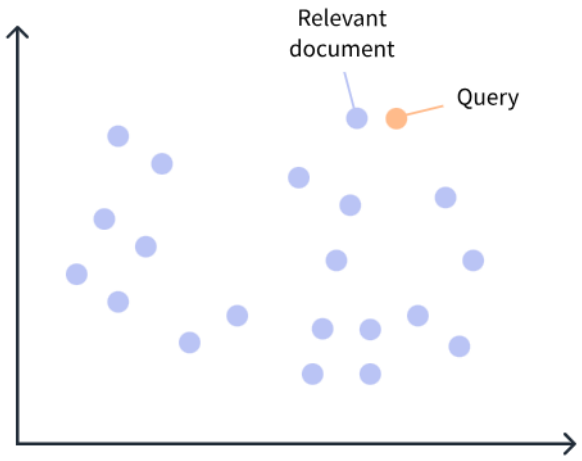
- **Kích thước chunk động:** Mỗi chunk được giới hạn ở 512 ký tự (chunk_size=512). Kích thước này được chọn để cân bằng giữa việc duy trì ngữ cảnh đủ lớn và tạo ra các đơn vị đủ nhỏ để xử lý hiệu quả. Kích thước này có thể được điều chỉnh tùy thuộc vào đặc điểm cụ thể của tập dữ liệu và yêu cầu của mô hình embedding downstream.

- **Chồng lấp chunk (overlap):** Áp dụng kỹ thuật chồng lấp với 256 ký tự giữa các chunk liền kề. Phương pháp này đảm bảo rằng thông tin quan trọng nằm ở ranh giới giữa các chunk không bị mất và duy trì tính liên tục của ngữ cảnh. Mức độ chồng lấp cao (50%) giúp giảm thiểu việc mất mát thông tin quan trọng.

- **Hàm đo độ dài:** Sử dụng hàm len() làm length_function để đo độ dài của văn bản. Điều này cho phép hệ thống xác định chính xác kích thước của mỗi chunk dựa trên số ký tự, đảm bảo tính nhất quán trong quá trình phân đoạn.

2.3 Embedding

Embedding được hiểu là một cách để biểu diễn dữ liệu (chẳng hạn như từ ngữ, câu, hoặc hình ảnh) dưới dạng các vector. Máy tính không thể hiểu được ngôn ngữ tự nhiên nhưng có thể hiểu được các con số, bằng việc này, ta có thể giúp nó xử lý và tính toán được với ngôn ngữ tự nhiên. Một cách đơn giản, máy tính không thể biết được 2 câu như thế nào là đồng nghĩa. Tuy nhiên, trên khía cạnh vector, ta có thể nói 2 vector có độ tương đồng càng cao, càng giống nhau, càng gần nhau thì tương đương với 2 câu đó cũng gần nghĩa nhau. Có rất nhiều cách tính toán độ tương đồng giữa các vector, có thể kể đến như *cosine similarity* và *Euclidean Distance*.



Hình 2.2 Minh họa về tương đồng vector

Trong đề tài này embedding chủ yếu đề cập đến việc chuyển đổi một câu, hoặc một đoạn văn (tập hợp các câu) thành vector. Với sức mạnh của học chuyển giao (transfer learning), hiện nay đã có rất nhiều mô hình embedding được tinh chỉnh riêng cho tiếng Việt, mang lại hiệu suất cao trong việc xử lý và tìm kiếm tương đồng tiếng Việt. Bảng dưới đây là so sánh về hiệu quả hoạt động của một số model embedding tốt nhất cho tiếng Việt trên cùng một bộ dữ liệu (được thực hiện bởi Google Expert Nguyễn Bá Ngọc - chuyên gia NLP của ProtonX):

	STS-B	STS12	STS13	STS14	STS15	STS16	STS-Sickr	Mean
VoVanPhuc/sup-SimCSE-VietNamese-phobert-base	**81.520000**	**85.020000**	78.220000	75.940000	81.530000	75.390000	77.750000	**79.338571**
keepitreal/vietnamese-sbert	80.540000	78.580000	**80.750000**	**76.980000**	**82.570000**	73.210000	80.160000	78.970000
nampham1106/bkcare-embedding	80.630000	80.590000	78.060000	73.970000	77.580000	74.160000	**81.230000**	78.031429
intfloat/multilingual-e5-base	77.500000	74.670000	67.040000	66.350000	79.050000	**77.070000**	76.850000	74.075714
hiieu/halong_embedding	75.670000	67.080000	71.080000	68.290000	78.550000	75.040000	77.300000	73.287143
bkai-foundation-models/vietnamese-bi-encoder	73.300000	67.840000	71.690000	69.800000	78.400000	74.290000	76.010000	73.047143

```
[32] 1 df_spearman_styled
```

	STS-B	STS12	STS13	STS14	STS15	STS16	STS-Sickr	Mean
VoVanPhuc/sup-SimCSE-VietNamese-phobert-base	**81.430000**	**76.510000**	79.190000	**74.910000**	81.720000	76.570000	76.450000	**78.111429**
keepitreal/vietnamese-sbert	80.160000	69.080000	**80.990000**	73.670000	**82.810000**	74.300000	73.400000	76.344286
nampham1106/bkcare-embedding	79.880000	72.660000	78.350000	70.740000	77.610000	75.140000	**77.380000**	75.965714
intfloat/multilingual-e5-base	76.750000	69.200000	67.260000	65.730000	79.350000	**77.530000**	72.400000	72.602857
hiieu/halong_embedding	74.540000	62.750000	71.410000	65.510000	78.660000	75.350000	70.860000	71.297143
bkai-foundation-models/vietnamese-bi-encoder	72.160000	63.860000	71.820000	66.200000	78.620000	74.240000	70.870000	71.110000

Hình 2.3 Một số mô hình embedding cho tiếng việt

Trong báo cáo này, nhóm sử dụng model "hieuu/halong_embedding" để thực hiện việc embedding, có nhiều lý do để nhóm quyết định lựa chọn sử dụng mô hình này dù nó không phải tốt nhất theo bảng:

- **Xử lý tương đồng giữa truy vấn và context dài tốt:** Sbert là mô hình thường được sử dụng trong trường hợp này, tuy nhiên theo thử nghiệm của nhóm, khi context càng dài lên so với câu truy vấn, hiệu quả của Sbert giảm đi rõ rệt, điều này không xảy ra ở SimCSE và halong embedding.
- SimCSE là tốt, ít tham số hơn nên tốc độ cũng nhanh, tuy nhiên số token đầu vào khá nhỏ, đồng thời cần phải thêm một vài bước xử lý text đặc thù mới có thể sử dụng được mô hình này.
- **Ngôn ngữ hỗ trợ:** Mô hình "hieuu/halong_embedding" là công nghệ nhúng văn bản tiếng Việt tập trung vào RAG. Vì nó là một mô hình "sentence-tranformers" được tinh chỉnh (finetuned) từ "infloat/multilingual-e5-base" nên cũng hỗ trợ đa ngôn ngữ, tuy nhiên do được tinh chỉnh với một bộ dữ liệu tốt và chất lượng, nó phù hợp hơn với tiếng việt.
- **Mô tả:** Mô hình giúp ánh xạ các câu và đoạn văn thành không gian vector 768 chiều với chiều dài chuỗi tối đa là 512 tokens có thể phục vụ cho việc so sánh ngữ nghĩa văn bản, tìm kiếm ngữ nghĩa, khai thác diễn giải, phân loại văn bản, phân cụm và nhiều mục đích khác.

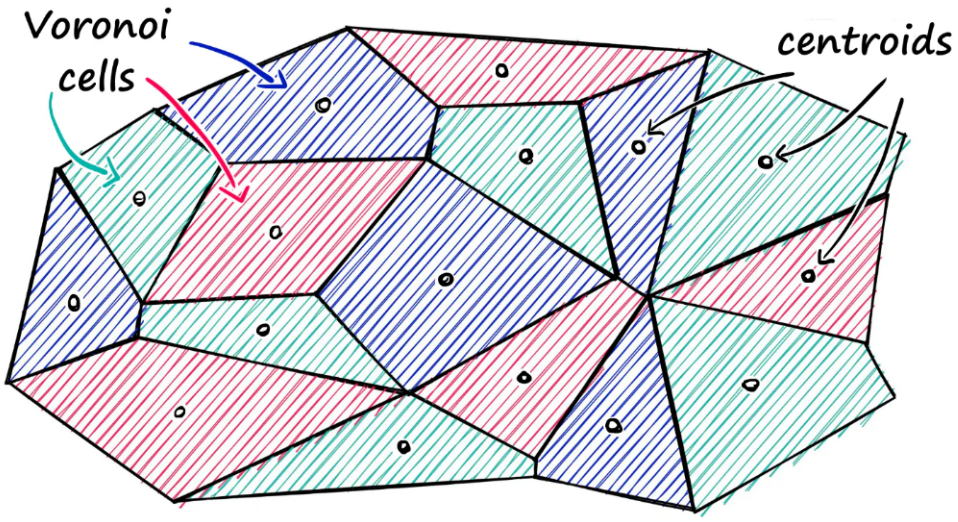
Bên cạnh đó, mô hình này cũng được phát triển bởi senior AI của Momo, Hiếu Ngô, anh cũng là người có nhiều năm kinh nghiệm trong lĩnh vực AI cũng như có nhiều đóng góp, chia sẻ cho cộng đồng, nên việc nhóm sử dụng mô hình này cũng như một bước thử nghiệm, góp phần giúp tác giả có thể cải thiện mô hình tốt hơn. Ngoài ra, kết quả đánh giá trên trang Huggingface cũng cho thấy mô hình này không chỉ cung cấp chỉ số accuracy cao trong việc truy xuất thông tin mà còn thể hiện sự cân bằng tốt giữa precision và recall. Đặc biệt, với việc đã được tinh chỉnh (finetuned) để tối ưu hoá cho tiếng Việt, mô hình sẽ giúp nâng cao đáng kể chất lượng của hệ thống RAG trong việc xử lý và truy xuất thông tin từ tài liệu tiếng Việt.

2.4 Lưu trữ và tìm kiếm ngữ cảnh

2.4.1 Ý tưởng cơ bản của FAISS

FAISS (Facebook AI Similarity Search) là một thư viện mã nguồn mở được phát triển bởi Facebook AI Research, được thiết kế để lưu trữ và tìm kiếm hiệu quả trong cơ sở dữ liệu vector lớn. FAISS cung cấp một phương pháp nhanh chóng và tối ưu để tìm kiếm các vector có độ tương đồng cao trong không gian nhiều chiều. Về mặt ý tưởng cơ bản, FAISS giống như một phương pháp chia để trị. Để giải thích tại sao thư viện này cực kỳ hiệu quả trong việc tìm kiếm tương đồng trong cơ sở dữ liệu vector lớn, trong số đó nhóm sẽ đề cập đến công nghệ nổi bật nhất, đó chính là IVF (Inverted file).

Về IVF, nó hoạt động bằng cách sử dụng các thuật toán phân cụm, như K-means, để chia không gian vector dữ liệu thành các cụm nhỏ hơn (gọi là cluster hoặc cell). Mỗi cụm có một centroid đại diện, là điểm trung bình của các vector trong cụm. FAISS lưu trữ các vector theo từng cụm trong một cấu trúc dữ liệu gọi là inverted file hoặc inverted index, nơi mỗi cụm có một codebook riêng chứa centroid và các vector của cụm đó. Khi có một vector truy vấn mới, nó sẽ so sánh vector này với các centroids để xác định cụm gần nhất. Sau đó, nó tìm kiếm trong cụm đó để xác định các vector có khoảng cách gần nhất với vector truy vấn, thường là top-k vector gần nhất.



Hình 2.4 Chia dữ liệu thành các cell và tìm kiếm theo tâm cụm

Với cách này, tốc độ tìm kiếm tương đồng của FAISS là cực kỳ nhanh và chính xác, nó có thể tìm kiếm trong cơ sở dữ liệu hàng triệu vector một cách cực kỳ nhanh

chóng. Bên cạnh đó, kết hợp với PQ (Product Quantization), FAISS thậm chí còn có thể đẩy nhanh tốc độ hơn nữa qua việc giảm thiểu lưu trữ độ lớn tính toán. Đây chính là một điểm mạnh của công nghệ này khi công cụ giao tiếp cũng cần tốc độ phản hồi nhanh.

2.4.2 Lưu trữ cơ sở dữ liệu vector

Đặc biệt, nhóm đã áp dụng chiến lược lưu trữ riêng biệt cho mỗi tài liệu PDF, tạo ra một cơ sở dữ liệu vector độc lập cho từng PDF. Quy trình này bao gồm các bước sau:

- **Kiểm tra tính duy nhất của tài liệu:**
  - Hệ thống tính toán một giá trị hash độc nhất cho mỗi file PDF đầu vào.
  - Giá trị hash này được so sánh với một danh sách các hash đã tồn tại để tránh xử lý trùng lặp các tài liệu đã được xử lý trước đó .
- **Xử lý và phân đoạn tài liệu:**
  - Nội dung của tài liệu PDF được đọc và trích xuất.
  - Sau đó, nội dung được chia thành các đoạn nhỏ (chunks) có ý nghĩa để chuẩn bị cho quá trình tạo embedding.
  - Đồng thời cũng lưu lại một bản text gốc trích ra từ tài liệu để thuận tiện cho việc làm giàu ngữ cảnh sau này.
- **Tạo embedding và xây dựng cơ sở dữ liệu FAISS:**
  - Mỗi đoạn văn bản được chuyển đổi thành vector embedding bằng cách sử dụng một mô hình embedding đã tạo trước đó.
  - Các vector này được sử dụng để xây dựng một cơ sở dữ liệu FAISS riêng cho tài liệu đó.
- **Lưu trữ cơ sở dữ liệu:**
  - Mỗi cơ sở dữ liệu FAISS được lưu trong một thư mục riêng biệt.
  - Tên của thư mục tương ứng với tên file PDF, sau khi đã loại bỏ dấu tiếng Việt để đảm bảo tính tương thích với FAISS.

Chiến lược này mang lại nhiều lợi ích:

- **Quản lý hiệu quả:** Việc lưu trữ riêng biệt cho phép cập nhật và quản lý độc lập từng tài liệu.
- **Tìm kiếm nhanh chóng:** Giới hạn phạm vi tìm kiếm trong một tài liệu cụ thể giúp tăng tốc độ truy vấn. Cho phép trích dẫn và tham chiếu trực tiếp từ tài liệu gốc giúp tăng độ tin cậy cho câu trả lời.
- **Mở rộng dễ dàng:** Hệ thống có thể thêm mới hoặc cập nhật tài liệu mà không ảnh hưởng đến toàn bộ cơ sở dữ liệu.

2.4.3 Tìm kiếm tương đồng và mở rộng ngữ cảnh

Sau khi tạo và lưu trữ cơ sở dữ liệu vector, nhóm tiến hành tìm kiếm tương đồng câu hỏi truy vấn với các vector đó. Quy trình này được tiến hành như sau:

- **Tìm kiếm tương đồng:**
  - Hệ thống bắt đầu bằng việc tìm kiếm các đoạn văn bản tương đồng nhất với câu hỏi của người dùng trong các cơ sở dữ liệu vector đã được chọn trước đó.
  - Quá trình này được thực hiện riêng biệt cho từng cơ sở dữ liệu, cho phép tìm kiếm song song để tăng tốc độ tìm kiếm.
  - Đối với mỗi cơ sở dữ liệu, hệ thống trả về hai đoạn văn bản có độ tương đồng cao nhất, cùng với điểm số tương đồng tương ứng.
- **Xếp hạng kết quả:**
  - Sau khi có kết quả từ tất cả các cơ sở dữ liệu được chọn, hệ thống xếp hạng tất cả các đoạn văn bản dựa trên điểm số tương đồng của chúng.
  - Hai đoạn văn bản có điểm số cao nhất được chọn để xử lý tiếp theo.
- **Mở rộng ngữ cảnh:**
  - Đối với mỗi đoạn văn bản được chọn, hệ thống thực hiện quá trình mở rộng context.
  - Từ đoạn văn bản đó, ta sử dụng regex để đối chiếu nó sang đúng đoạn văn bản nguyên bản ban đầu đã được lưu.
  - Sau đó ta tiến thành mở rộng đoạn context hiện tại bằng cách thêm một số lượng từ nhất định có trong văn bản gốc vào trước và sau đoạn đã tìm được.

Phương pháp mở rộng context này mang lại nhiều lợi ích:

- **Cung cấp ngữ cảnh đầy đủ:** Người dùng có thể hiểu rõ hơn về bối cảnh và ý nghĩa của thông tin được trích xuất
- **Tăng độ chính xác:** Bằng cách cung cấp thêm thông tin xung quanh, hệ thống giúp giảm thiểu khả năng hiểu sai hoặc thiếu sót thông tin quan trọng
- **Linh hoạt trong việc điều chỉnh:** Số lượng từ trước và sau có thể được điều chỉnh dễ dàng, cho phép tùy chỉnh mức độ mở rộng ngữ cảnh tùy theo nhu cầu cụ thể

Việc kết hợp tìm kiếm tương đồng với mở rộng ngữ cảnh cho phép hệ thống tạo ra các câu trả lời chính xác, có chiều sâu, và có thể truy xuất nguồn gốc, nâng cao đáng kể chất lượng tương tác giữa người dùng và hệ thống trí tuệ nhân tạo.

2.5 Prompting và LLM

- **Thiết kế prompt template:** Ở bước này nhóm đã phát triển một mẫu prompt bao gồm các hướng dẫn cụ thể cho LLM. Template này tích hợp các nguyên tắc xử lý khi thiếu thông tin, tránh bịa đặt, tuân thủ ngữ cảnh đã cung cấp, và duy trì nhất quán ngôn ngữ tiếng Việt. Cấu trúc template được thiết kế để chấp nhận ba biến đầu vào chính: lịch sử tương tác, ngữ cảnh hiện tại (context), và câu hỏi của người dùng.
- **Khởi tạo và quản lý mô hình:** Một lớp GeminiBot được triển khai để khởi tạo và quản lý mô hình Gemini. Lớp này xử lý việc cấu hình API key, thiết lập tham số sinh nội dung, và cài đặt an toàn. Quá trình khởi tạo bao gồm cơ chế đếm token để giám sát và quản lý độ dài đầu ra của mô hình.
- **Xử lý đầu vào người dùng:** Hệ thống xử lý đầu vào của người dùng thông qua phương thức response() trong lớp GeminiBot. Phương thức này thực hiện kiểm tra số lượng token và tương tác với mô hình chat Gemini để tạo ra phản hồi. Lớp này được viết riêng, không sử dụng cùng với Langchain vì nhóm muốn tùy chỉnh nhiều thứ cho LLM.
- **Tích hợp ngữ cảnh trong quá trình sinh câu trả lời:** LLM tạo ra câu trả lời dựa trên ngữ cảnh được cung cấp, thông tin lịch sử, và câu hỏi hiện tại. Hệ thống được

thiết kế để ưu tiên thông tin từ ngữ cảnh đã cho, đảm bảo tính liên quan và chính xác trong câu trả lời được tạo ra.

- **Cơ chế xử lý khi generate bị thiếu token:** Mặc dù ít xảy ra nhưng khi output token đạt đến giới hạn, câu trả lời sẽ bị cắt giữa chừng. nhóm nhóm đã bổ sung thêm một cách khắc phục phần này bằng tính năng **Generate more**, cho phép mô hình sinh tiếp câu trả lời cho đến khi nó kết thúc, bằng cách truyền vào yêu cầu, câu trả lời cũ, câu hỏi cũ, và context cũ.

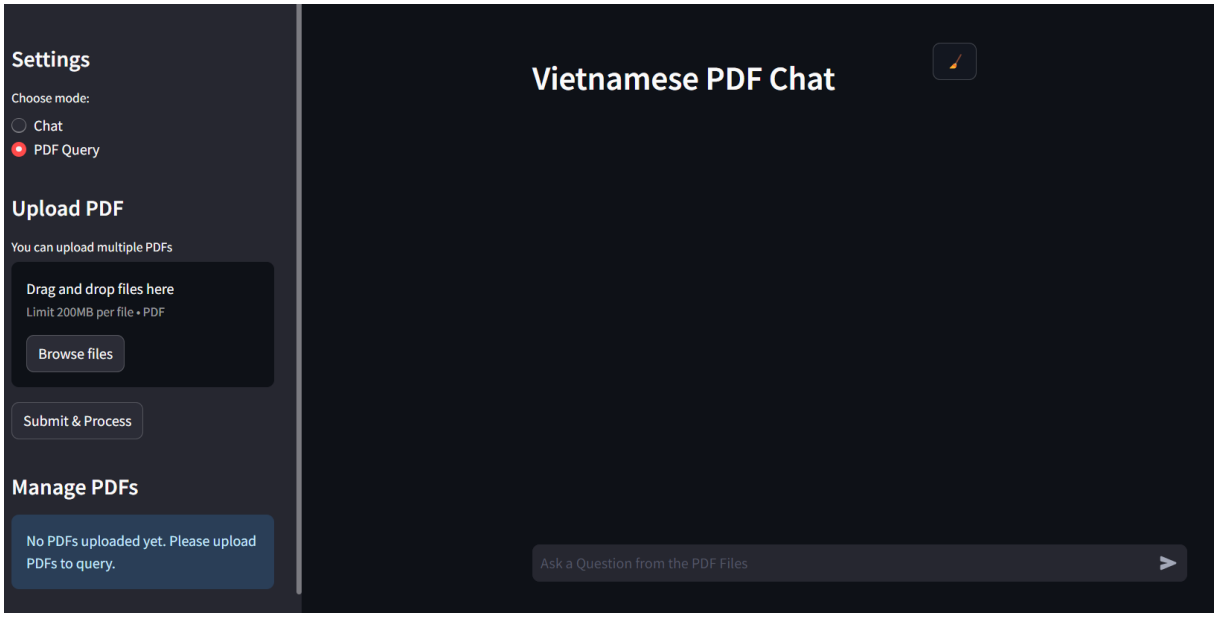
- **Hiển thị câu trả lời động:** Một cơ chế hiển thị tiến trình được phát triển để hiển thị câu trả lời theo từng phần, mô phỏng quá trình gõ chữ trong thời gian thực. Phương pháp này nâng cao sự tương tác của người dùng và tạo ra trải nghiệm tương tác hơn.

2.6 Xây dựng demo

Ứng dụng demo thành công trong việc tạo ra một giao diện trực quan và tương tác cho phép người dùng trò chuyện với nội dung PDF. Các tính năng chính bao gồm:

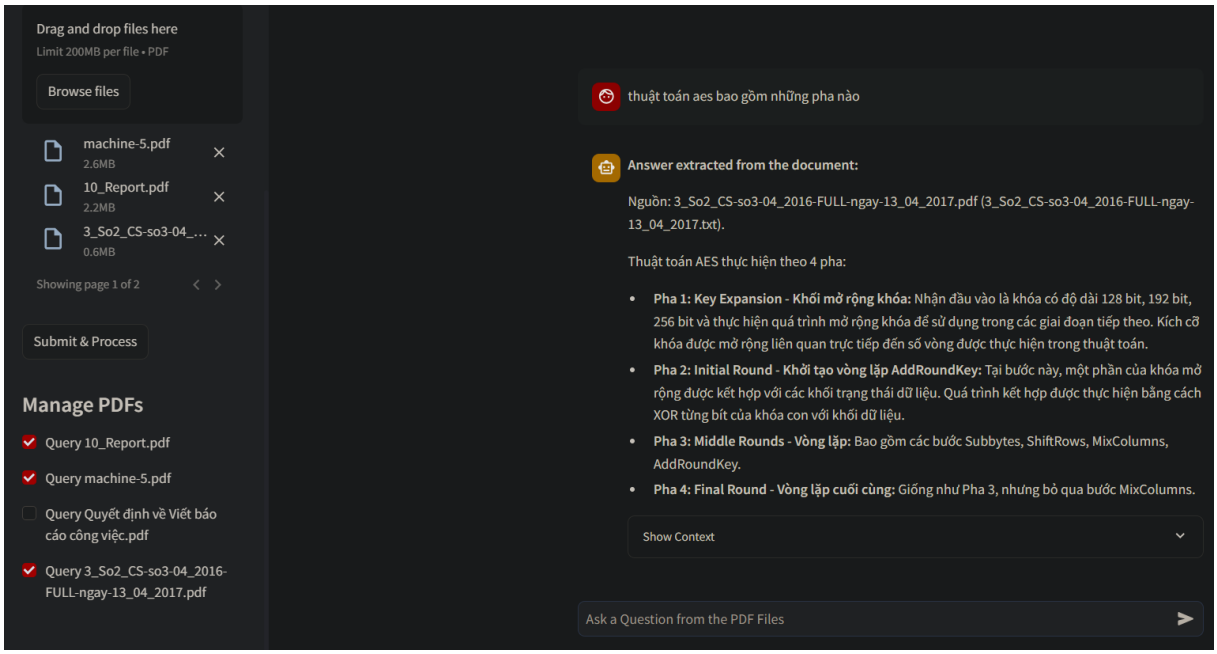
- 2 chế độ trả lời: Chat và PDF Query.
- Cho phép người dùng tải lên và xử lý nhiều tài liệu PDF.
- Người dùng còn có thể quản lý tài liệu đã tải lên, chọn lựa tài liệu cụ thể để truy vấn.
- Giao diện thân thiện với người dùng, hiển thị câu trả lời theo thời gian thực với hiệu ứng đánh máy.
- Khả năng mở rộng câu trả lời với nút "Generate More" (nếu câu trả lời bị ngắt giữa chừng).
- Nút bấm giúp người dùng có thể xoá đi toàn bộ file đã tải lên và đoạn chat trước đó.

Dưới đây là một số hình ảnh minh họa về các tính năng:



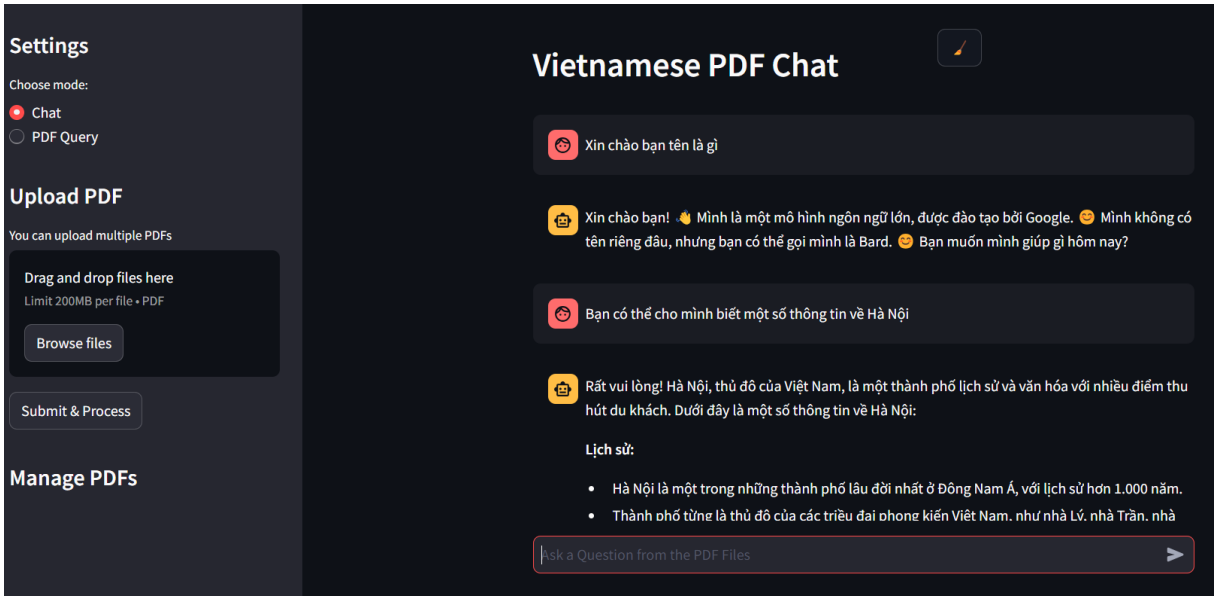
Hình 2.5 Giao diện web streamlit

Các tính năng nằm ở thanh slidebar bên trái. Ở phần trên cùng là tính năng chọn chế độ, cho phép người dùng giao tiếp với chatbot thông thường hoặc truy vấn kết hợp với tài liệu PDF. Người dùng có thể upload 1 hoặc nhiều PDF cùng lúc, khi bấm Submit, các vector database sẽ được khởi tạo. Người dùng có thể bấm chọn truy vấn trong PDF nào ở checkbox bên dưới cùng thanh bar. Khung chat sẽ xuyên suốt dù người dùng đổi mode, đặc biệt khi truy vấn kết hợp PDF, phần context sẽ được hiển thị để người dùng đối chiếu câu trả lời của chatbot với văn bản gốc trong PDF:



Hình 2.6 Chế độ chat truy vấn PDF

Đối với chế độ chat thông thường, phần context và Manage PDFs sẽ không được hiển thị, trong chế độ này, bạn có thể hỏi mô hình bất kỳ điều gì, mô hình sẽ trả lời trong phạm vi kiến thức hiện có. Như đã đề cập ở trên, các mô hình ngôn ngữ lớn thường có thể gây ra hiện tượng ảo giác, nên hãy kiểm tra thật kỹ lại thông tin trước khi quyết định tin tưởng vào chatbot.



Hình 2.7 Chế độ chat thông thường

Demo hiện tại được public trên streamlit, thầy cô và các bạn có thể thử nghiệm tại: https://chat-with-pdf-123.streamlit.app/.

Chương 3

Kết quả đạt được

3.1 Kết quả đạt được

Nghiên cứu này đã trình bày quá trình phát triển và triển khai một ứng dụng cho phép người dùng trò chuyện với nội dung của tài liệu PDF thông qua giao diện Streamlit. Bằng cách sử dụng mô hình ngôn ngữ lớn kết hợp với RAG. Kết quả của nghiên cứu đã chứng minh tính khả thi và tiềm năng của việc kết hợp các công nghệ xử lý ngôn ngữ tự nhiên với giao diện người dùng trực quan để tạo ra một công cụ hỗ trợ tương tác với tài liệu hiệu quả. Qua thử nghiệm, tỉ lệ trả lời đúng của chatbot là khá tốt, có khả năng trả lời và tổng hợp thông tin một cách tự nhiên. Tổng hợp những công việc mà chatbot của nhóm hiện tại đã làm được:

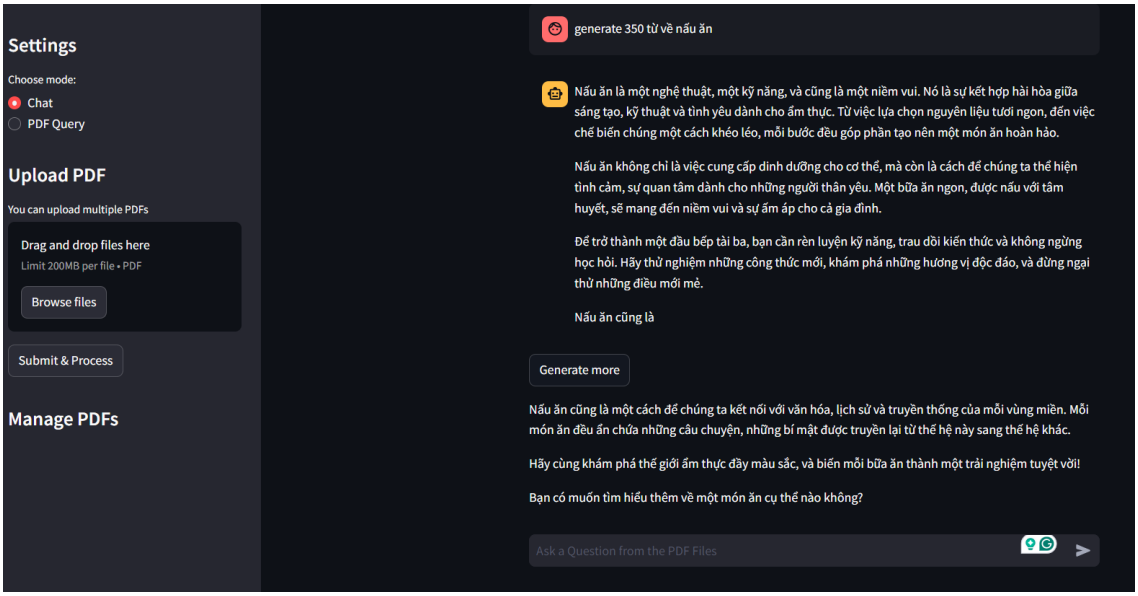
- **Kết hợp hiệu quả RAG và LLM:** Việc tích hợp RAG đã mang lại những cải thiện đáng kể:
  - **Tăng độ chính xác:** Câu trả lời dựa trên thông tin thực tế từ tài liệu, giảm thiểu việc "ảo tưởng" của LLM.
  - **Cải thiện tính liên quan:** Hệ thống có khả năng trích xuất và mở rộng ngữ cảnh, cung cấp câu trả lời toàn diện hơn.
  - **Khả năng mở rộng:** Có thể xử lý nhiều tài liệu PDF, cho phép truy vấn trên một cơ sở kiến thức rộng lớn.
- **Xây dựng giao diện người dùng tương tác:**
  - Sử dụng Streamlit để tạo giao diện web trực quan và dễ sử dụng.

  - Triển khai chức năng tải lên và quản lý nhiều tài liệu PDF.
  - Cho phép người dùng chọn chế độ chat thông thường hoặc chế độ truy vấn PDF.

- **Xử lý và quản lý tài liệu PDF:**
  - Phát triển PDFDatabaseManager để xử lý, lưu trữ và quản lý tài liệu PDF.
  - Triển khai hệ thống băm tệp để tránh xử lý trùng lặp.
  - Tạo và quản lý cơ sở dữ liệu vector sử dụng FAISS.

- **Tối ưu hóa trải nghiệm người dùng:**
  - Triển khai hiển thị câu trả lời theo thời gian thực với hiệu ứng đánh máy.
  - Phát triển chức năng "Generate more" để mở rộng câu trả lời khi cần thiết.
  - Tích hợp chức năng hiển thị ngữ cảnh và nguồn tài liệu để tăng độ tin cậy cho câu trả lời.

Giải thích thêm về tính năng generate more: Chức năng này được sử dụng trong trường hợp khi văn bản được tạo ra ở output của mô hình vượt quá số token tối đa trong config. Khi điều này xảy ra, câu trả lời sẽ bị dừng giữa chừng, không tạo thành câu hoặc ý hoàn chỉnh. Để giải quyết điều đó, nhóm bổ sung thêm một nút generate more, cho phép chatbot tiếp tục gen câu hỏi từ phần bị dừng lại. Về mặt kỹ thuật, đây chỉ đơn giản là bổ sung thêm 1 prompt mới với các thông tin cũ ở trên được truyền vào. Dưới đây là hình ảnh minh họa tính năng này:



Hình 3.1 Nối token khi bị quá giới hạn

3.2 Những vấn đề chưa giải quyết

Tuy nhiên, do thời gian thực hiện cũng như sự giới hạn về kiến thức và tài nguyên, đề tài hiện tại còn tồn đọng một số vấn đề chưa được giải quyết:

- **Hiệu suất xử lý:** Có thể bị ảnh hưởng khi đối mặt với số lượng lớn tài liệu PDF hoặc tài liệu có kích thước lớn, cần cải thiện khả năng mở rộng và tối ưu hoá hiệu suất.
- **Khả năng xử lý đa dạng định dạng tài liệu:** Chưa xử lý được với PDF dạng ảnh. Khả năng xử lý còn hạn chế đối với các định dạng tài liệu ngoài PDF.
- **Chất lượng Phản hồi:** Mặc dù hệ thống cung cấp thông tin chính xác, nhưng đôi khi phản hồi có thể thiếu chiều sâu hoặc ngữ nghĩa phong phú. Cũng có thể có trường hợp LLM sáng tạo nhiều thông tin hơn cần thiết. Giải pháp đề ra: cần sử dụng mô hình thông minh hơn, hoặc tinh chỉnh prompt tốt hơn.
- **Bảo mật và Quyền riêng tư:** Hiện tại, hệ thống chưa tích hợp các biện pháp bảo mật mạnh mẽ để bảo vệ dữ liệu người dùng và tài liệu nhạy cảm. Lưu trữ và quản lý lịch sử trò chuyện lâu dài.

KẾT LUẬN

Trong báo cáo này, nhóm đã thực hiện việc nghiên cứu, xây dựng được một hệ thống hỏi đáp thông minh, tích hợp Retrieval-Augmented Generation (RAG) với Large Language Models (LLM) và sử dụng framework Langchain. Hệ thống này cho phép truy vấn thông tin từ tài liệu PDF, đồng thời cung cấp khả năng tương tác thông minh. Kết quả cho thấy sự kết hợp này mang lại hiệu quả đáng kể trong việc cải thiện độ chính xác và tính liên quan của câu trả lời.

Đề xuất hướng phát triển: Trong thời gian tới, nhóm sẽ tập trung vào việc cải thiện chất lượng đầu ra của hệ thống hỏi đáp nhằm đưa đến cho người sử dụng kết quả tốt hơn. Cụ thể, nhóm sẽ thực hiện các hướng phát triển:

- **Tối ưu hóa hiệu suất để xử lý khối lượng lớn tài liệu và truy vấn đồng thời:** Nghiên cứu và áp dụng các kỹ thuật tối ưu hóa để cải thiện tốc độ xử lý và khả năng mở rộng của hệ thống.
- **Mở rộng hỗ trợ cho các định dạng tài liệu khác ngoài PDF:** Đọc được file PDF dạng ảnh định hướng sử dụng OCR để trích xuất. Mở rộng hỗ trợ cho nhiều định dạng tài liệu khác như docx, txt, html.
- **Cải thiện chất lượng phản hồi:** Cải thiện độ chính xác trong câu trả lời. Thực hiện các nghiên cứu sâu hơn về việc cải tiến mô hình ngôn ngữ, embedding và searching để cung cấp những phản hồi chính xác và phong phú hơn.
- **Bảo mật và Quyền riêng tư:** Tạo trang web hoặc app cho phép người dùng đăng nhập vào để có thể lưu trữ và bảo mật các tài liệu tải lên và lịch sử trò chuyện lâu dài.
- **Mở rộng Chức năng:** Xem xét việc bổ sung các tính năng mới như hỗ trợ đa ngôn ngữ, phân tích cảm xúc, và khả năng tương tác với các nguồn dữ liệu khác nhau.
- **Tinh chỉnh để phù hợp với các tác vụ cụ thể:** Trong doanh nghiệp, có thể tinh chỉnh (finetuned) để đào tạo với dữ liệu nội bộ của doanh nghiệp giúp doanh nghiệp tối ưu hoá việc quản lý và sử dụng thông tin nội bộ.

Tài liệu tham khảo

[1] PixeGami, "Python RAG Tutorial (with Local LLMs): AI For Your PDFs", Youtube Video.

[2] Hieu Ngo, "Enhancing Document Retrieval Fine-Tuning Text Embeddings for RAG," GitHub repository.

[3] Meta AI, "Nomic Embed text," Ollama library.

[4] Thallys Costa, "Chunking Techniques," Medium.

[5] Mike Callahan, "Enhance Search Engine for RAG," LinkedIn.

[6] Nguyễn Bá Ngọc, "Semantic Router for RAG," GitHub repository.

[7] Nir Diamant, "RAG Techniques," GitHub repository.

[8] Phuc Phan, "ChatGPT Series 5: Tìm hiểu về Retrieval-Augmented Generation (RAG)", Viblo.