TRẦN VĂN DU MSSV: N20DCCN011 Xây dựng Chatbot tư vấn tuyển sinh Lớp: D20CQCNPM01-N 2020-2025

TP. HCM 2024

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---

# BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC

***Đề tài*:**

**"XÂY DỰNG CHATBOT TƯ VẤN TUYỂN SINH"**

**Người hướng dẫn : TS. HUỲNH TRỌNG THƯA**
**Sinh viên thực hiện : TRẦN VĂN DU**
**Mã số sinh viên : N20DCCN011**
**Lớp : D20CQCNPM01-N**
**Khoá : 2020 – 2025**
**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 08/2024**

BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

---


# BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC

***Đề tài*:**

**"XÂY DỰNG CHATBOT TƯ VẤN TUYỂN SINH"**


| | | |
|---|---|---|
| **Người hướng dẫn** | **:** | **TS. HUỲNH TRỌNG THƯA** |
| **Sinh viên thực hiện** | **:** | **TRẦN VĂN DU** |
| **Mã số sinh viên** | **:** | **N20DCCN011** |
| **Lớp** | **:** | **D20CQCNPM01-N** |
| **Khoá** | **:** | **2020 – 2025** |
| **Hệ** | **:** | **ĐẠI HỌC CHÍNH QUY** |


**TP.HCM, tháng 08/2024**

# PHIẾU GIAO ĐỀ CƯƠNG THỰC TẬP TỐT NGHIỆP

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
**Độc lập- Tự do- Hạnh phúc**

*TP. Hồ Chí Minh, ngày tháng năm 2024*

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN THỰC TẬP TỐT NGHIỆP ĐẠI HỌC

1. **Tên đề tài: Xây dựng Chatbot tư vấn tuyển sinh**
2. **Sinh viên: Trần Văn Du Lớp: D20CQCNPM01-N**
3. **Giáo viên hướng dẫn: Huỳnh Trọng Thưa**
4. **Nơi công tác: khoa Công nghệ thông tin 2**

## NỘI DUNG NHẬN XÉT

1. Đánh giá chung:

……

……

……

2. Đánh giá chi tiết:

……

……

……

……

……

……

3. Nhận xét về tinh thần, thái độ làm việc:

……

……

4. Kết luận:

……

……

5. Điểm hướng dẫn:

**GIẢNG VIÊN HƯỚNG DẪN**
(Ký, ghi rõ họ tên)

# LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến thầy Huỳnh Trọng Thưa, người đã tận tình hướng dẫn, chỉ bảo và cung cấp những kiến thức quý báu trong suốt quá trình thực hiện đề tài. Sự hỗ trợ và định hướng của Thầy đã giúp em vượt qua những khó khăn và hoàn thiện báo cáo này.

Em cũng xin chân thành cảm ơn các thầy cô trong Khoa Công Nghệ Thông Tin 2, Học viện Công Nghệ Bưu Chính Viễn Thông đã giảng dạy, truyền đạt kiến thức và tạo điều kiện thuận lợi cho tôi trong suốt quá trình học tập.

Trong quá trình thực hiện đề tài sẽ có những thiếu sót, em xin các thầy cô trong khoa có thể đóng góp ý kiến, phản hồi cũng như cung cấp thêm những hướng dẫn chi tiết cụ thể về mặt chuyên môn để em có thể thực hiện đề tài được hoàn chỉnh nhất cho quá trình thực tập tốt nghiệp.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Trần Văn Du

# MỤC LỤC

# DANH MỤC BẢNG, HÌNH ẢNH, SƠ ĐỒ

**Bảng:**



**Hình:**



**Sơ đồ:**

# KÝ HIỆU CÁC CỤM TỪ VIẾT TẮT

| | |
|---|---|
| HTTP: HyperText Transfer Protocol | Giao thức truyền siêu văn bản |
| LLM: Large Language Model | Mô hình ngôn ngữ lớn |
| ML: Machine Learning | Học máy |
| NLP: Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| RAG: Retrieve Augmented Generation | Tạo tăng cường truy xuất |
| URL: Uniform Resource Locator | Bộ định vị tài nguyên thống nhất |

# LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển vượt bậc, việc ứng dụng Trí tuệ nhân tạo (AI) vào các lĩnh vực khác nhau đã trở thành xu hướng tất yếu. Đặc biệt, trong lĩnh vực giáo dục, các hệ thống chatbot thông minh đang dần thay thế những phương thức tư vấn truyền thống, mang lại hiệu quả cao hơn và tiết kiệm thời gian cho cả người sử dụng lẫn người quản lý.

Việc tuyển sinh vào các trường đại học luôn là một quá trình phức tạp và đòi hỏi sự hỗ trợ liên tục từ bộ phận tuyển sinh. Tuy nhiên, với sự phát triển của công nghệ, đặc biệt là các ứng dụng AI, chúng ta có thể xây dựng những hệ thống tư vấn tự động để giải quyết các thắc mắc của học sinh và phụ huynh một cách nhanh chóng và chính xác.

Đề tài "Xây Dựng Chatbot Tư Vấn Tuyển Sinh" nhằm mục đích phát triển một công cụ hỗ trợ tuyển sinh dựa trên các công nghệ hiện đại như RAG (Retrieve Augmented Generation) và PhoWhisper. Chatbot này không chỉ giúp giảm tải công việc cho bộ phận tuyển sinh mà còn cung cấp thông tin đầy đủ, chính xác và kịp thời cho người dùng.

Cấu trúc bài báo cáo:

- Chương 1: Giới thiệu đề tài
- Chương 2: Cơ sở lý thuyết
- Chương 3: Thiết kế Chatbot
- Chương 4: Triển khai Chatbot

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Trong vài năm trở lại đây, các hệ thống chatbot thông minh đã dần trở nên phổ biến và được ứng dụng trong nhiều lĩnh vực khác nhau, đặc biệt là trong giáo dục. Tính đến thời điểm hiện tại, hầu hết việc trao đổi thông tin giữa học sinh và các trường đại học đang được thực hiện một cách thủ công, quy trình này thường rất tốn thời gian và đặt gánh nặng cho bộ phận tuyển sinh. Chính vì vậy, trong nghiên cứu này, tôi sẽ xây dựng một chatbot tư vấn tuyển sinh dựa trên công nghệ Trí tuệ nhân tạo (AI), là một nền tảng giúp học sinh nhận được những cập nhật về chương trình đào tạo, thủ tục nhập học, học phí, hay các thông tin liên quan đến tuyển sinh đại học.

Chatbot tư vấn tuyển sinh được xây dựng với các công nghệ hiện đại như RAG và PhoWhisper để có thể cung cấp thông tin chính xác và đáng tin cậy cho người dùng.

## 1.2 Mục tiêu đề tài

- Mục tiêu chính: Chatbot của bạn sẽ tư vấn các thông tin liên quan đến tuyển sinh như chương trình học, yêu cầu đầu vào, thời gian nộp hồ sơ, học phí, học bổng, và các thông tin khác.
- Đối tượng sử dụng: Học sinh trung học phổ thông, sinh viên tiềm năng, phụ huynh, và những người quan tâm đến việc học đại học.

## 1.3 Công nghệ sử dụng

- Ngôn ngữ lập trình Python
- HTML, CSS, JavaScript
- Cơ sở dữ liệu: MongoDB

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Giới thiệu ngôn ngữ lập trình Python và Flask

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Flask được tạo ra bởi Armin Ronacher, một nhà phát triển phần mềm người Đức vào năm 2010. Anh ấy đã phát triển Flask với mục tiêu tạo ra một framework đơn giản nhưng mạnh mẽ để phát triển ứng dụng web bằng Python. Quá trình phát triển Flask được khai thác để đơn giản hóa các yêu cầu và nguồn mã yêu cầu, tạo ra một khung hoạt động và dễ dàng tiếp cận.

Flask được phát triển theo ý tưởng "Micro - Framework" [1], công nghệ tập trung vào các tính năng cơ bản và linh hoạt để người phát triển có thể tùy chỉnh theo nhu cầu cụ thể của họ. Điều này đã giúp Flask trở thành một trong những framework web Python phổ biến nhất và được ưa chuộng trong cộng đồng phát triển phần mềm.

Flask Framework sở hữu một số tính năng quan trọng mà nhà phát triển thường sử dụng để xây dựng hiệu ứng web. Các tính năng chính của Flask:

- Nhẹ và dễ sử dụng: Công nghệ có cấu trúc nhẹ nhàng và mã nguồn dễ đọc, giúp người phát triển dễ dàng tiếp cận và tùy chỉnh theo nhu cầu cụ thể của họ.
- Định tuyến linh hoạt: Flask cung cấp cơ chế hoạt động định tuyến [1], cho phép người phát triển xác định các mẫu URL và phân bổ chúng cho các hàm xử lý tương ứng. Điều này giúp quản lý và xử lý yêu cầu HTTP một cách hiệu quả.
- Công cụ mẫu: Flask tích hợp Jinja2 [1], đây là một loại trình biên dịch mẫu mạnh mẽ cho phép tạo ra các giao diện người dùng.
- Được mở rộng rộng rãi: Mặc dù mang đặc điểm rút gọn nhưng Flask vẫn có khả năng mở rộng mạnh mẽ thông qua việc sử dụng các tiện ích và thư viện của cộng đồng. Người dùng có thể phân tích các tính năng như xác thực, đăng nhập, điều hướng, cơ sở dữ liệu tương tác và nhiều tính năng khác.
- Máy chủ phát triển tích hợp: Flask cung cấp máy chủ phát triển hợp đồng, giúp người phát triển dễ dàng kiểm tra và phát triển ứng dụng mà không cần cấu hình bổ sung.
- Gửi yêu cầu RESTful: Flask hỗ trợ xây dựng API và các ứng dụng RESTful theo cách hoạt động và hiệu quả [1].
- Cộng đồng lớn và tích cực: Số lượng người dùng Flask rất đông và luôn nhận được hỗ trợ mạnh mẽ từ cộng đồng Python, điều này giúp người phát triển tìm kiếm thông tin và tài liệu một cách dễ dàng.

## 2.2 Giới thiệu Mongodb Atlas

MongoDB là một dạng phần mềm cơ sở dữ liệu sử dụng mã nguồn mở NoSQL. Nó có thể hỗ trợ trên nhiều nền tảng khác nhau và được thiết kế với mục đích hướng đến đối tượng. MongoDB hoạt động dựa vào các khái niệm Collection và Document [2]. Đồng thời, nó có hiệu suất cao cùng với tính khả dụng tốt và dễ dàng mở rộng.

Mongodb Atlas là một giải pháp phần mềm Database as a Service Provider có chức năng và chi phí hoàn toàn phù hợp cho mọi doanh nghiệp từ nhỏ đến vừa và đến lớn. Khi đó, phần mềm MongoDB Atlas sẽ được đánh giá bởi tất cả các người dùng lẫn với chuyên gia trong các lĩnh vực Database Software.

Những ưu điểm nổi trội của Mongodb thuyết phục lựa chọn của người dùng hiện nay là:

- Bởi vì Mongodb sử dụng các dữ liệu dưới dạng Document JSON [2] nên mỗi một collection đều có kích cỡ và document khác nhau. Nhưng chúng lại rất linh hoạt khi thực hiện lưu trữ bởi vậy nếu bạn muốn thứ gì thì chỉ cần insert thoải mái.
- Các dữ liệu có trong Mongodb thường không ràng buộc lẫn nhau, chúng không có join như trong RDBMS, nên khi bạn insert, xóa hoặc update thì sẽ không phải bỏ ra quá nhiều thời gian để kiểm tra chúng có thỏa mãn các ràng buộc như trong RDBMS hay không.
- Mongodb dễ mở rộng được, và trong Mongodb luôn có khái niệm cluster chính là cụm các node sẽ có chứa các dữ liệu giao tiếp với nhau [2]. Nên chỉ cần bạn muốn mở rộng hệ thống thì chỉ việc thêm một node mới vào cluster.
- Các trường hợp dữ liệu "_id" sẽ luôn được đánh tự động index [2], nên tốc độ truy vấn thông tin sẽ luôn đạt hiệu suất cao nhất.
- Nếu như có một truy vấn dữ liệu, thì bản ghi sẽ được cached lên bộ nhớ Ram. Khi đó, việc phục vụ lượt truy vấn sau sẽ diễn ra nhanh hơn mà bạn không cần phải đọc từ ổ cứng.
- Tốc độ truy vấn của Mongodb luôn nhanh hơn so với các hệ quản trị cơ sở dữ liệu quan hệ. Nhờ có một lượng đủ dữ liệu nên việc thử nghiệm cho thấy tốc độ insert của Mongodb sẽ nhanh gấp 100 lần so với MySQL.
- Ngoài sở vô số ưu điểm nổi bật như trên nhưng Mongodb vẫn còn tồn tại một vài điểm yếu là:
- Mongodb không sở hữu các tính chất ràng buộc như trong RDBMS [2] nên khi bạn thao tác với Mongodb cần phải cẩn thận hết sức.
- Có thể sẽ tốn bộ nhớ do dữ liệu được lưu trữ dưới dạng key-value, nên các collection sẽ chỉ khác về value do vậy mà key có thể sẽ bị lặp lại. Mongodb còn không hỗ trợ join nên rất dễ bị dữ thừa dữ liệu [2].

## 2.3 Kỹ thuật RAG (Retrieve Augmented Generation)

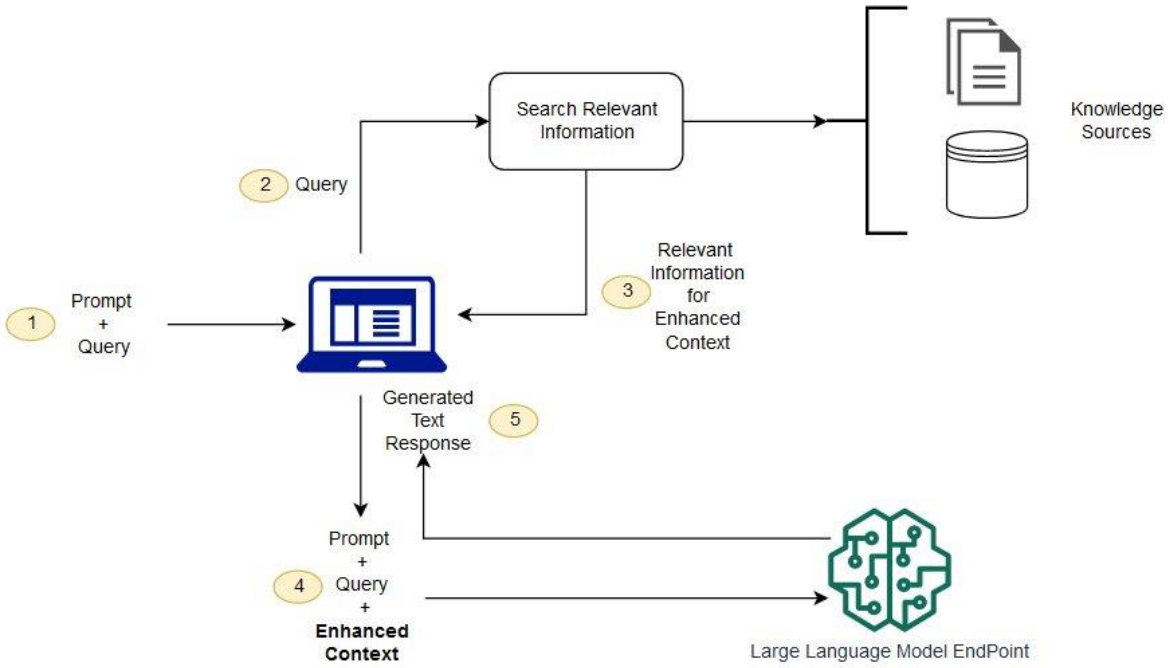
Retrieve Augmented Generation (RAG) là quá trình tối ưu hóa đầu ra của một mô hình ngôn ngữ lớn, vì vậy nó tham khảo một cơ sở kiến thức có thẩm quyền bên ngoài các nguồn dữ liệu đào tạo của nó trước khi tạo phản hồi [3]. Mô hình ngôn ngữ lớn (LLM) được đào tạo trên khối lượng dữ liệu khổng lồ và có sử dụng hàng tỷ tham số để tạo ra đầu ra ban đầu cho các nhiệm vụ như trả lời câu hỏi, dịch ngôn ngữ và hoàn thành câu. RAG mở rộng các khả năng vốn đã mạnh mẽ của LLM đến các miền cụ thể hoặc cơ sở kiến thức nội bộ của tổ chức, tất cả mà không cần đào tạo lại mô hình [3]. Đây là một cách tiếp cận hiệu quả về chi phí để cải thiện đầu ra LLM, để nó vẫn phù hợp, chính xác và hữu ích trong nhiều bối cảnh khác nhau.

LLM là một công nghệ trí tuệ nhân tạo (AI) quan trọng, hỗ trợ các chatbot thông minh và các ứng dụng xử lý ngôn ngữ tự nhiên (NLP) khác. Mục tiêu là tạo ra các bot có thể trả lời các câu hỏi của người dùng trong nhiều bối cảnh bằng cách tham chiếu chéo các nguồn kiến thức có thẩm quyền. Rất tiếc, bản chất của công nghệ LLM đưa ra sự không thể đoán trước trong các phản hồi LLM [3]. Ngoài ra, dữ liệu đào tạo LLM là tĩnh và giới thiệu ngày giới hạn về kiến thức hiện có.

Các thách thức đã biết của LLM bao gồm:

- Trình bày thông tin sai lệch khi nó không có câu trả lời [3].
- Trình bày thông tin lỗi thời hoặc chung chung khi người dùng mong chờ một phản hồi cụ thể, hiện tại [3].
- Tạo phản hồi từ những nguồn không có thẩm quyền [3].
- Tạo phản hồi không chính xác do nhầm lẫn thuật ngữ, trong đó các nguồn đào tạo khác nhau sử dụng cùng một thuật ngữ để nói về những điều khác nhau [3].

RAG là một cách giải quyết một số thách thức này. Nó chuyển hướng LLM để truy xuất thông tin liên quan từ các nguồn kiến thức có thẩm quyền, đã xác định trước [3]. Các tổ chức có quyền kiểm soát tốt hơn đối với đầu ra văn bản đã tạo và người dùng nắm được thông tin chi tiết về cách LLM tạo phản hồi.




Hình 2.3.1 Quy trình của việc sử dụng RAG với LLM.

Kỹ thuật RAG kết hợp hai thành phần chính:

- Retrieve (Truy xuất): Tìm kiếm các đoạn văn bản hoặc dữ liệu liên quan từ một cơ sở dữ liệu hoặc kho dữ liệu lớn [4].
- Generate (Sinh): Sử dụng mô hình sinh văn bản để tạo câu trả lời dựa trên các đoạn văn bản đã truy xuất [4].

### 2.3.1 Ưu điểm của RAG

RAG là giải pháp mà các công ty, doanh nghiệp hướng đến trong phát triển sản phẩm trong thời đại LLMs [5]:

- Với RAG, LLM có thể tận dụng dữ liệu bên ngoài để cung cấp tri thức cho nó.
- RAG không yêu cầu training lại mô hình, tiết kiệm thời gian và tài nguyên tính toán.
- Nó hiệu quả ngay cả với một lượng dữ liệu gán nhãn hạn chế.
- Tuy nhiên, RAG cũng có nhược điểm đó là hiệu suất của RAG phụ thuộc vào chất lượng độ chính xác của model retrieval; tính toàn diện và chính xác của kho tri thức có sẵn.
- RAG phù hợp nhất cho các tình huống có nhiều dữ liệu chưa được gán nhãn (unlabeled data) nhưng nguồn dữ liệu gán nhãn khan hiếm và lý tưởng cho các ứng dụng như trợ lý ảo cần truy cập theo thời gian thực vào thông tin cụ thể như hướng dẫn sử dụng phần mềm, tin tức, ...
    - Trường hợp mà có nhiều dữ liệu unlabeled-data nhưng lại khan hiếm dữ liệu labeled-data: RAG rất hữu ích trong những trường hợp này, tức là có sẵn nhiều dữ liệu nhưng hầu hết dữ liệu đó không được phân loại hoặc

gãn nhãn theo cách hữu ích cho mô hình có thể học. Ví dụ như internet có số lượng tin tức, văn bản text lớn nhưng hầu hết các văn bản text đó không được tổ chức theo các trả lời trực tiếp các câu hỏi cụ thể.

- RAG lý tưởng cho các ứng dụng như trợ lý ảo, chatbot: Các trợ lý ảo, chatbot, như Siri hay Alexa cần lấy thông tin từ nhiều nguồn khác nhau để trả lời các câu hỏi khác nhau trong thời gian thực. Chúng cần hiểu câu hỏi, lấy thông tin liên quan và sau đó đưa ra câu trả lời mạch lạc và chính xác.
- Cần truy cập theo thời gian thực (real-time) vào thông tin cụ thể như hướng dẫn sử dụng: Đây là 1 ví dụ về trường hợp mà RAG đặc biệt hữu ích. Tưởng tượng là bạn hỏi trợ lý ảo một câu hỏi cụ thể về một sản phẩm, chẳng hạn như "Làm thế nào để reset điều khiển ABC". RAG trước tiên sẽ truy xuất thông tin liên quan từ hướng dẫn sử dụng sản phẩm hoặc các tài liệu khác, sau đó sử dụng thông tin đó để tạo ra câu trả lời rõ ràng, ngắn gọn.

### 2.3.2 Hạn chế của kĩ thuật RAG cơ bản

RAG là một phương pháp nhằm cải thiện kết quả của các LLM bằng cách tận dụng thêm các nguồn dữ liệu tham khảo bên ngoài. Quy trình này bao gồm ba thành phần chính: hệ thống truy xuất, cơ sở dữ liệu chứa thông tin tham khảo và thành phần tạo sinh - LLM.

Trong Basic RAG, thông tin từ các nguồn dữ liệu bên ngoài được tải và phân tách thành các đoạn nhỏ (chunking), mã hóa thành vector bằng mô hình Transformer Encoder (embedding model), và lưu thành các index (indexing). Khi có truy vấn từ người dùng, hệ thống sẽ vector hóa truy vấn này và tìm kiếm trong các index để xác định các đoạn thông tin phù hợp nhất, sau đó tổng hợp và đưa vào LLM model dưới dạng ngữ cảnh để tạo nội dung phản hồi.

Dù có thể giải quyết được các nhu cầu cơ bản, tuy nhiên Basic RAG cũng có một số vấn đề như:

- Quá trình truy vấn: Thiếu độ chính xác trong việc lựa chọn thông tin liên quan đến câu hỏi của người dùng và có khả năng bỏ sót các chi tiết quan trọng.
- Quá trình tạo phản hồi: Sai lệch ngữ cảnh có thể dẫn đến việc tạo nội dung không chính xác, không liên quan, ảnh hưởng đến chất lượng và độ tin cậy của hệ thống.
- Quá trình tích hợp thông tin: Thông tin được lấy từ nhiều nguồn mà không có sự tổng hợp hoặc bổ sung sẽ có khả năng trùng lặp, xung đột, thiếu nhất quán về văn phong… ảnh hưởng đến sự mạch lạc của kết quả phản hồi, làm giảm trải nghiệm người dùng.

### 2.3.2 Các kĩ thuật RAG nâng cao

a. Indexing

Quá trình Indexing là bước thiết yếu, giúp cải thiện độ chính xác và hiệu quả của hệ thống sử dụng LLM. Indexing không chỉ đơn thuần là lưu trữ dữ liệu mà còn bao gồm việc tổ chức và tối ưu hóa dữ liệu để có thể dễ dàng hiểu và truy xuất thông tin cần thiết mà không mất đi ngữ cảnh quan trọng của dữ liệu [5].

Một số kỹ thuật trong quá trình Indexing:

- Chunk Optimization: Tối ưu hóa kích thước và cấu trúc của các đoạn văn bản (chunk) để đảm bảo rằng chúng không quá lớn hay quá nhỏ, giúp duy trì ngữ cảnh cần thiết mà không vượt quá giới hạn độ dài của LLM [5].
- Embedding Fine-tuning: Tinh chỉnh embedding model giúp cải thiện khả năng hiểu ngữ nghĩa của dữ liệu được tạo index, qua đó tăng cường khả năng khớp nội dung truy xuất với yêu cầu của người dùng [5].
- Multi-Representation [5]: Phương pháp này cho phép biến đổi tài liệu thành các đơn vị truy xuất gọn nhẹ như tóm tắt nội dung, giúp cải thiện độ chính xác và tốc độ của quá trình truy xuất khi người dùng cần thông tin cụ thể từ một tài liệu lớn.
- Hierarchical Indexing: Áp dụng mô hình phân cấp như RAPTOR để tổ chức dữ liệu thành các cấp độ tổng hợp khác nhau từ chi tiết đến tổng quát giúp cải thiện việc truy xuất thông tin dựa trên ngữ cảnh rộng hơn và chính xác hơn [5].
- Metadata Attachment: Thêm metadata vào từng chunk hoặc dữ liệu giúp tăng khả năng phân tích và phân loại thông tin, cho phép truy xuất dữ liệu một cách có hệ thống hơn, phù hợp với nhiều tình huống cụ thể [5].

b. Query Transformation & Query Routing

Quá trình Query Transformation là các kỹ thuật sử dụng mô hình LLM như một công cụ lý luận để chỉnh sửa đầu vào của người dùng nhằm cải thiện chất lượng truy xuất thông tin [5]. LLM có thể chuyển đổi câu hỏi gốc thành các câu hỏi rõ ràng, dễ hiểu hơn, từ đó tăng khả năng tìm kiếm và truy xuất thông tin một cách hiệu quả.

Một số kỹ thuật trong quá trình Query Transformation:

- HyDE: Là một kỹ thuật đảo ngược trong đó LLM được yêu cầu sinh ra dữ liệu giả định dựa trên câu hỏi, sau đó sử dụng vector của dữ liệu này cùng với vector của câu hỏi để cải thiện chất lượng truy xuất thông tin tham khảo [5]. Kỹ thuật này giúp nâng cao sự tương đồng ngữ nghĩa giữa câu hỏi và nội dung tham khảo được truy xuất, qua đó tăng cường độ chính xác và hiệu quả của quá trình truy vấn.
- Multi-Step Query: Là phương pháp phân rã câu hỏi phức tạp thành nhiều câu hỏi con đơn giản hơn, thực hiện truy xuất song song cho từng câu hỏi con này, và kết hợp các kết quả truy xuất lại với nhau [5]. Điều này cho phép LLM tổng

hợp và sinh ra câu phản hồi được chính xác hơn. Phương pháp này đặc biệt hữu ích trong các trường hợp không có thông tin trực tiếp về nội dung được hỏi.

Quá trình Query Routing là bước do LLM model thực hiện để xác định hành động tiếp theo dựa trên truy vấn của người dùng. Các lựa chọn có thể là tìm kiếm nội dung trong một tập dữ liệu cụ thể, hoặc thử nghiệm nhiều hướng tìm kiếm khác nhau và sau đó tổng hợp các kết quả lại thành một câu phản hồi thống nhất [5]. Quá trình này giúp định hướng truy vấn đến nguồn dữ liệu phù hợp nhất, từ cơ sở dữ liệu vector cổ điển đến cơ sở dữ liệu đồ thị hoặc cơ sở dữ liệu quan hệ, hoặc thậm chí là các chỉ mục phân cấp khác nhau.

Một số kỹ thuật trong quá trình Query Routing:

- Logical Routing: Là kỹ thuật sử dụng logic để định hướng truy vấn đến nguồn dữ liệu phù hợp. Bằng cách phân tích cấu trúc và mục đích của câu hỏi, router lựa chọn index hoặc nguồn dữ liệu thích hợp nhất để thực hiện truy vấn [5]. Điều này giúp tối ưu hóa quá trình truy xuất thông tin bằng cách đảm bảo rằng truy vấn được xử lý bởi nguồn dữ liệu có khả năng phản hồi chính xác nhất.
- Semantic Routing: Phương pháp này tận dụng ngữ nghĩa của câu hỏi để định hướng. Router phân tích ý nghĩa ngữ nghĩa của câu hỏi và định hướng nó đến index hoặc nguồn dữ liệu phù hợp, nhằm tăng khả năng tìm kiếm và truy xuất thông tin liên quan một cách chính xác [5]. Phương pháp này đặc biệt hữu hiệu khi xử lý các truy vấn phức tạp, cần sự hiểu biết về ngữ cảnh và ý nghĩa của từng từ và cụm từ trong truy vấn.

c. Retrieval

Quá trình Retrieval là bước cốt lõi trong hệ thống RAG, tập trung vào việc lấy dữ liệu tham khảo từ các nguồn khác nhau để cung cấp ngữ cảnh và thông tin cần thiết cho LLM model thực hiện tạo sinh câu phản hồi [5].

Một số kỹ thuật trong quá trình Retrieval:

- Recursive Retriever: Là kỹ thuật cho phép truy xuất sâu vào các dữ liệu liên quan và thực hiện truy vấn thêm dữ liệu dựa trên kết quả truy vấn trước đó [6]. Kỹ thuật này hữu ích trong các tình huống cần khám phá thông tin chi tiết hoặc chuyên sâu.
- Router Retriever [6]: Là kỹ thuật sử dụng LLM để đưa ra quyết định động về nguồn dữ liệu hoặc công cụ truy vấn dữ liệu phù hợp cho mỗi truy vấn cụ thể.
- Auto Retriever: Phương pháp tự động truy vấn cơ sở dữ liệu bằng cách sử dụng LLM để xác định metadata để thực hiện filter hoặc tạo câu truy vấn phù hợp để truy xuất...
- Fusion Retriever: Kết hợp kết quả từ nhiều truy vấn và index khác nhau [5], giúp tối ưu hóa việc truy xuất thông tin và đảm bảo kết quả thu được là toàn diện và không bị trùng lặp, mang lại cái nhìn đa chiều cho quá trình truy xuất.

- Auto Merging Retriever: Khi có nhiều phân đoạn dữ liệu con được truy vấn, kỹ thuật này sẽ chuyển chúng thành phân đoạn dữ liệu cha [6], cho phép tập hợp các ngữ cảnh nhỏ lẻ thành một ngữ cảnh lớn hơn, hỗ trợ quá trình tổng hợp thông tin. Kỹ thuật này giúp cải thiện độ liên quan và tính toàn vẹn của ngữ cảnh.

d. Post-Retrieval

Quá trình Post – Retrieval là nơi kết quả truy xuất được tinh chỉnh thông qua việc lọc (filter), sắp xếp lại (reranking), hoặc biến đổi. Mục đích của quá trình này là để chuẩn bị và cải thiện ngữ cảnh trước khi đưa vào LLM model để sinh ra câu phản hồi cuối cùng, đảm bảo thông tin được cung cấp cho LLM là chính xác và hiệu quả nhất.

Một số kỹ thuật trong quá trình Post – Retrieval:

- Rerank: Tái sắp xếp các đoạn văn bản đã truy xuất để những kết quả liên quan nhất xuất hiện đầu tiên, giúp cải thiện độ chính xác của thông tin [5]. Reranking không chỉ giúp giảm số lượng tài liệu cần đưa vào LLM model mà còn hoạt động như một bộ lọc để xử lý ngôn ngữ một cách chính xác hơn.
- Compress: Giảm bớt phần ngữ cảnh dư thừa và không cần thiết, loại bỏ các thông tin nhiễu để tăng cường nhận thức của LLM về các thông tin chính. Compress giúp tối ưu hóa độ dài của ngữ cảnh mà LLM có thể xử lý, từ đó nâng cao chất lượng của câu phản hồi bằng cách tập trung vào những thông tin quan trọng [5].
- Filter: Chọn lọc nội dung trước khi đưa vào LLM, loại bỏ những tài liệu hoặc thông tin không liên quan hoặc có độ chính xác thấp [5]. Kỹ thuật này giúp đảm bảo rằng chỉ những thông tin phù hợp và chất lượng cao mới được sử dụng, từ đó cải thiện độ chính xác và độ tin cậy của câu phản hồi.

e. Generation

Quá trình Generation là giai đoạn LLM model sinh ra câu phản hồi dựa trên thông tin đã được truy xuất và xử lý từ các bước trước. Mục tiêu của quá trình này là tạo ra câu phản hồi chính xác và có liên quan cao đối với truy vấn ban đầu của người dùng [5].

Chất lượng của quá trình này phụ thuộc nhiều vào LLM model được chọn. Tuy nhiên, hệ thống có thể áp dụng một số kỹ thuật để tăng chất lượng cho kết quả của quá trình Generation:

- FLARE: FLARE là phương pháp dựa trên kỹ thuật Prompt Engineering để kiểm soát khi nào mô hình ngôn ngữ lớn (LLM) nên thực hiện truy xuất dữ liệu [5]. Kỹ thuật này giúp đảm bảo rằng LLM chỉ tiến hành truy xuất khi thiếu thông tin thiết yếu, nhằm tránh việc thu thập dữ liệu không cần thiết hoặc không phù hợp. Quá trình này liên tục điều chỉnh câu hỏi và kiểm tra các từ khóa có xác suất xuất hiện thấp; nếu những từ này xuất hiện, hệ thống sẽ truy xuất các

tài liệu liên quan để cải thiện và tinh chỉnh câu phản hồi, qua đó nâng cao độ chính xác và độ liên quan của phản hồi cuối cùng.

- ITER-RETGEN: ITER-RETGEN là kỹ thuật lặp đi lặp lại quá trình Generation dựa trên thông tin đã truy xuất. Mỗi vòng lặp sử dụng kết quả từ lần lặp trước làm ngữ cảnh cụ thể để giúp truy xuất kiến thức liên quan hơn, từ đó liên tục cải thiện chất lượng của câu phản hồi [5].
- ToC (Tree of Clarifications): ToC là phương pháp thực hiện truy vấn một cách đệ quy để làm rõ câu hỏi ban đầu. Trong quá trình này, mỗi bước hỏi-đáp đều thực hiện đánh giá dựa trên truy vấn hiện tại để sinh ra một câu hỏi cụ thể hơn. Quá trình này giúp làm sáng tỏ các vấn đề mơ hồ trong câu hỏi ban đầu, qua đó cải thiện độ chính xác và chi tiết của câu phản hồi [5].

f. Evaluation

Quá trình Evaluation là bước thiết yếu để đảm bảo rằng cả quá trình RAG đều đạt hiệu quả cao và chính xác. Đánh giá này phản ánh khả năng của hệ thống trong việc đáp ứng các yêu cầu của người dùng và xử lý các câu hỏi phức tạp.

Các tiêu chí đánh giá chất lượng câu phản hồi:

- Context Relevance: Đánh giá độ chính xác và cụ thể của ngữ cảnh đã truy xuất, đảm bảo rằng thông tin liên quan được lựa chọn một cách chính xác và giảm thiểu chi phí xử lý cho nội dung không cần thiết [5].
- Answer Faithfulness: Đảm bảo các câu phản hồi sinh ra phải nhất quán với ngữ cảnh đã truy xuất [5], tăng cường độ tin cậy của câu phản hồi đối với người dùng.
- Answer Relevance: Yêu cầu các câu phản hồi được sinh ra phải có liên quan trực tiếp đến câu hỏi đã đặt ra [5]. Điểm này đánh giá khả năng của hệ thống trong việc tập trung vào những thông tin quan trọng nhất đối với người dùng.

Các tiêu chí đánh giá khả năng của hệ thống:

- Noise Robustness: Đánh giá khả năng của hệ thống trong việc xử lý các tài liệu có liên quan đến câu hỏi nhưng thiếu thông tin có giá trị [5]. Điều này quan trọng để đảm bảo rằng hệ thống không bị phân tâm bởi thông tin nhiễu.
- Negative Rejection: Đánh giá khả năng của hệ thống trong việc không đưa ra phản hồi khi các tài liệu truy xuất không chứa kiến thức cần thiết để phản hồi câu hỏi [5]. Điều này giúp ngăn ngừa việc cung cấp thông tin sai lệch hoặc không chính xác.
- Information Integration: Đánh giá khả năng của hệ thống trong việc tổng hợp thông tin từ nhiều tài liệu để giải quyết các câu hỏi phức tạp. Khả năng này cần thiết cho những trường hợp cần hiểu và phân tích thông tin đa chiều [5].

- Counterfactual Robustness: Đánh giá khả năng của hệ thống trong việc nhận biết và bỏ qua các thông tin sai trái đã biết trong tài liệu. Điều này giúp tăng cường độ chính xác và độ tin cậy của thông tin được sinh ra [5].

## 2.4 Mô hình PhoWhisper

PhoWhisper là một mô hình xử lý ngôn ngữ tự nhiên (NLP) đặc biệt được thiết kế cho tiếng Việt, dựa trên mô hình Whisper của OpenAI. Mô hình này được tinh chỉnh để xử lý các đặc điểm ngôn ngữ và ngữ pháp riêng của tiếng Việt. PhoWhisper sử dụng công nghệ nhận dạng giọng nói tự động (ASR), còn được gọi là chuyển giọng nói thành văn bản [7].

PhoWhisper có năm phiên bản, bao gồm PhoWhispertiny, PhoWhisperbase, PhoWhispersmall, PhoWhispermedium và PhoWhisperlarge, sử dụng cùng kiến trúc của mô hình đa ngôn ngữ Whispertiny, Whisperbase, Whispersmall, Whispermedium và Whisperlarge-v2 tương ứng [7].

Sự mạnh mẽ của PhoWhisper đạt được thông qua việc tinh chỉnh mô hình Whisper trên bộ dữ liệu bao gồm các giọng Việt đa dạng. Cụ thể, mô hình được tinh chỉnh trên bộ đào tạo ASR quy mô lớn bao gồm 844 giờ âm thanh được thu thập từ bốn nguồn khác nhau, bao gồm CMV–Vi, phần tiếng Việt của Common Voice 14 (Ardila et al., 2020), VIVOS (Luong & Vu, 2016), thử thách ASR của VLSP 2020,1 và dữ liệu riêng tư của VinAI [7]. “Dữ liệu riêng tư” của VinAI là công cụ cung cấp sự đa dạng cần thiết về giọng nói của 26 nghìn người trải dài trên 63 tỉnh và thành phố [7], mang đến một hiểu biết sâu sắc về những cách nói đa dạng của tiếng Việt. Cuối cùng, để nâng cao tính chắc chắn của các mô hình chống lại tiếng ồn tự nhiên, mô hình kết hợp âm thanh môi trường có nguồn gốc từ Piczak (2015) và tận dụng thư viện thính giác để thêm tiếng ồn vào một nửa tập huấn luyện. Mô hình này sử dụng 8 GPU A100 (mỗi bộ nhớ 40GB) với kích thước lô trên mỗi thiết bị cố định ở mức 4 và số bước tích lũy độ dốc là 2 cho tất cả các phiên bản kiểu máy, dẫn đến kích thước lô toàn cầu là 64 [7].

Tốc độ học cao nhất được đặt ở 3,75e-5, 2,5e-5, 1,25e 5, 6,25e-6 và 5e-6 tương ứng cho PhoWhispertiny, PhoWhisperbase, PhoWhispersmall, PhoWhispermedium và PhoWhisperlarge. Mô hình này thực hiện tổng cộng 48.000 bước cập nhật, tương đương với 5 kỷ nguyên [7].

| Model | #paras | CMV-Vi | VIVOS | VLSP 2020 Task-1 | VLSP 2020 Task-2 |
|---|---|---|---|---|---|
| PhoWhisper-tiny | 39M | 19.05 | 10.41 | 20.74 | 49.85 |
| PhoWhisper-base | 74M | 16.19 | 8.46 | 19.70 | 43.01 |
| PhoWhisper-small | 244M | 11.08 | 6.33 | 15.93 | 32.96 |
| PhoWhisper-medium | 769M | 8.27 | 4.97 | 14.12 | 26.85 |

| | | | | | |
|---|---|---|---|---|---|
| PhoWhisper-large | 1.55B | 8.14 | 4.67 | 13.75 | 26.68 |

Bảng 2.1: Kết quả Tỷ lệ Lỗi Từ (Word Error Rate)

- Ứng dụng trong Chatbot
    - Hiểu ngôn ngữ tự nhiên: PhoWhisper giúp chatbot hiểu và xử lý các câu hỏi và yêu cầu bằng tiếng Việt một cách chính xác.
    - Sinh văn bản: Tạo ra các câu trả lời tự nhiên và phù hợp với ngữ cảnh tiếng Việt.
- Lợi ích
    - Chính xác: Xử lý ngôn ngữ tự nhiên tiếng Việt tốt hơn so với các mô hình NLP không chuyên.
    - Đa năng: Có thể áp dụng cho nhiều tác vụ NLP như phân loại văn bản, trả lời câu hỏi, và sinh văn bản.

PhoWhisperlarge

PhoWhisperlarge là một mô hình ngôn ngữ lớn (Large Language Model - LLM) được đào tạo trên một lượng lớn dữ liệu tiếng Việt. Nó được thiết kế để hiểu và tạo ra văn bản tiếng Việt một cách tự nhiên và chính xác, tương tự như cách con người giao tiếp.

Đặc điểm nổi bật của PhoWhisperlarge:

- Hiệu suất cao: PhoWhisperlarge được tối ưu hóa để nhận diện giọng nói tiếng Việt với độ chính xác cao, phù hợp cho nhiều ứng dụng thực tế như trợ lý ảo, dịch vụ khách hàng, và phân tích dữ liệu giọng nói.
- Mã nguồn mở: Mô hình này được phát hành dưới dạng mã nguồn mở, cho phép cộng đồng phát triển và cải tiến dựa trên nền tảng có sẵn.
- Công nghệ hiện đại: Sử dụng các kỹ thuật tiên tiến trong lĩnh vực học sâu (deep learning) và xử lý ngôn ngữ tự nhiên (NLP), PhoWhisperlarge có khả năng nhận diện giọng nói với độ chính xác cao ngay cả trong môi trường nhiễu.
- Dữ liệu huấn luyện phong phú: Mô hình được huấn luyện trên một lượng lớn dữ liệu giọng nói tiếng Việt, bao gồm nhiều giọng điệu và cách phát âm khác nhau, giúp cải thiện khả năng nhận diện trong các trường hợp đa dạng.
- Ứng dụng rộng rãi: PhoWhisperlarge có thể được tích hợp vào nhiều ứng dụng khác nhau như trợ lý ảo, hệ thống tổng đài tự động, dịch vụ khách hàng, và các công cụ hỗ trợ người khuyết tật.

Tốc độ học (learning rate) là một tham số quan trọng trong quá trình huấn luyện các mô hình học sâu. Nó xác định mức độ điều chỉnh của các trọng số trong mô hình sau mỗi lần cập nhật dựa trên lỗi dự đoán. Tốc độ học quá cao có thể khiến mô hình không hội tụ, tức là không đạt được kết quả ổn định, trong khi tốc độ học quá thấp có thể làm quá trình huấn luyện trở nên rất chậm.

Trong trường hợp của PhoWhisperlarge, tốc độ học cao nhất được đặt là 5e-6, tức là 5×10−65 \times 10^{-6}5×10−6. Đây là một giá trị khá nhỏ, cho thấy sự cẩn thận trong việc điều chỉnh mô hình để đạt được hiệu suất tối ưu mà không gặp phải vấn đề về quá trình hội tụ. Tốc độ học nhỏ giúp mô hình có thể học từ từ và chắc chắn, giảm thiểu nguy cơ bỏ sót các đặc trưng quan trọng trong dữ liệu hoặc gặp phải hiện tượng quá khớp (overfitting).

# CHƯƠNG 3: THIẾT KẾ CHATBOT

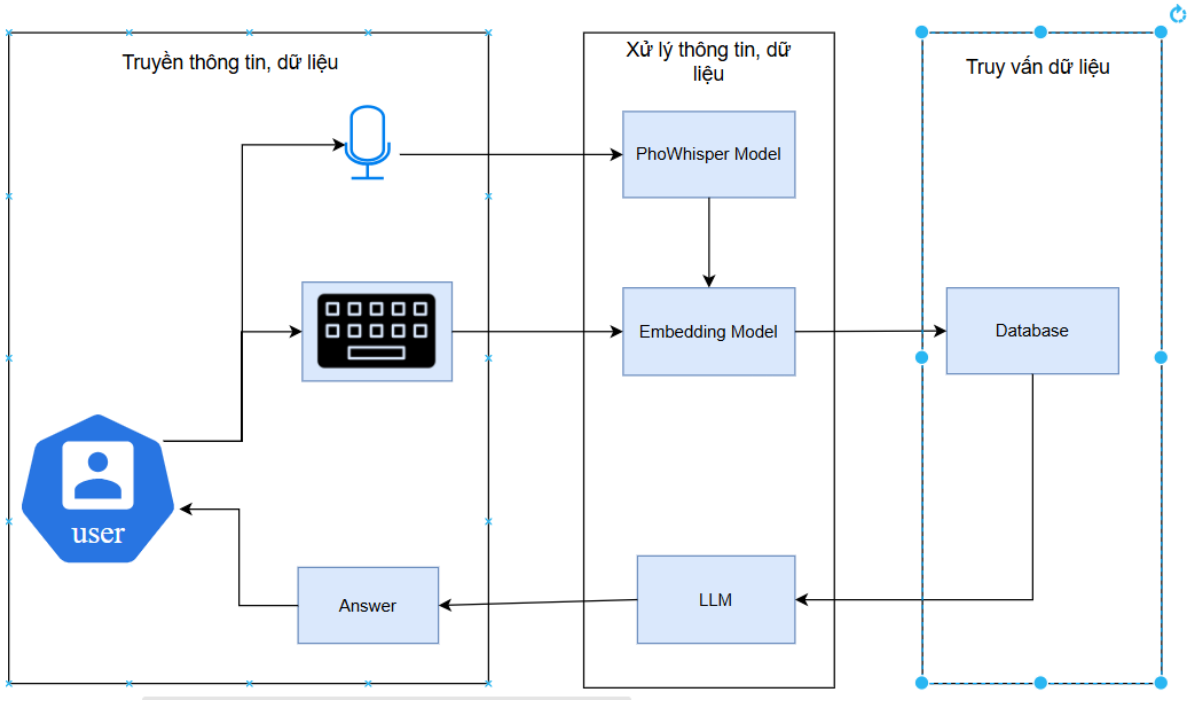
## 3.1 Mục tiêu thiết kế

Mục tiêu của chương trình là xây dựng một chatbot có nhận các câu hỏi liên quan đến tư vấn tuyển sinh đại học từ người dùng và trả lời các câu hỏi đó

## 3.2 Thiết kế hệ thống

Hệ thống được chia làm 3 phần chính: Truyền thông tin, dữ liệu; Xử lý thông tin, dữ liệu; Truy vấn dữ liệu

Mỗi phần đều có các chức năng và nhiệm vụ cụ thể, từ việc truyền tải và nhận thông tin, xử lý và biến đổi dữ liệu, đến việc tìm kiếm và truy xuất thông tin cần thiết. Việc phân chia này giúp hệ thống hoạt động một cách hiệu quả và có tổ chức, đảm bảo rằng mỗi phần có thể được tối ưu hóa riêng lẻ và hoạt động một cách phối hợp với các phần khác.

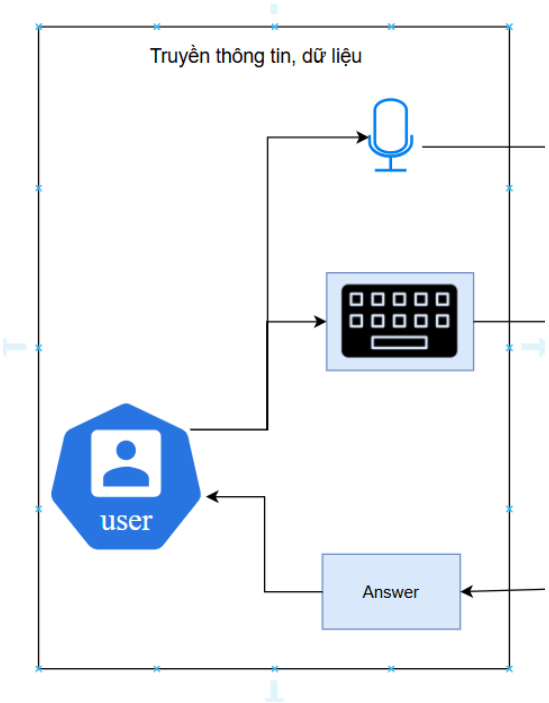



Sơ đồ 3.1 Sơ đồ hoạt động của hệ thống

- Chức năng các khối hệ thống

a. Khối truyền thông tin, dữ liệu

Khối truyền thông tin, dữ liệu nhằm gửi thông tin từ người dùng đến hệ thống và nhận kết quả trả về từ hệ thống đến người dùng, gồm có: Người dùng, dữ liệu đầu vào, Dữ liệu đầu ra.




Sơ đồ 3.2 Khối truyền thông tin, dữ liệu

Dữ Liệu Đầu Vào: Hệ thống nhận dữ liệu đầu vào từ hai nguồn chính:

- Âm thanh (Giọng nói): Người dùng có thể sử dụng microphone để đặt câu hỏi. Âm thanh này sẽ được chuyển đổi thành dạng văn bản thông qua một hệ thống nhận diện giọng nói (Speech-to-Text). Hệ thống sẽ phân tích và xử lý các dữ liệu giọng nói này để hiểu rõ nội dung câu hỏi.
- Bàn phím: Người dùng cũng có thể nhập câu hỏi trực tiếp qua bàn phím. Dữ liệu từ bàn phím được gửi trực tiếp đến hệ thống xử lý mà không cần qua bước chuyển đổi nào khác.

Khối Xử Lý Thông Tin và Dữ Liệu: Sau khi nhận được dữ liệu đầu vào, hệ thống sẽ gửi dữ liệu này đến khối xử lý thông tin. Khối xử lý này sẽ thực hiện các bước sau:

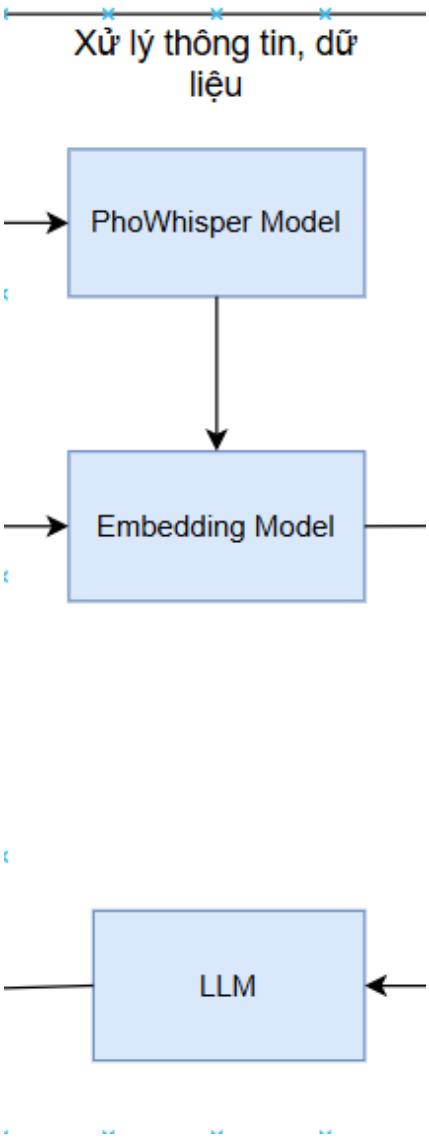
- Phân tích câu hỏi: Hệ thống sẽ phân tích cấu trúc và ngữ nghĩa của câu hỏi để hiểu rõ ý định của người dùng. Điều này bao gồm việc nhận diện từ khóa, ngữ cảnh và các yếu tố ngôn ngữ khác.
- Truy vấn thông tin: Sau khi hiểu rõ câu hỏi, hệ thống sẽ truy vấn cơ sở dữ liệu hoặc sử dụng các mô hình trí tuệ nhân tạo để tìm kiếm câu trả lời phù hợp. Hệ thống có thể sử dụng các mô hình học máy để dự đoán và đưa ra câu trả lời chính xác nhất dựa trên dữ liệu đã học.
- Xử lý dữ liệu: Dữ liệu thu thập được sẽ được xử lý để đảm bảo tính chính xác và phù hợp với câu hỏi của người dùng. Hệ thống sẽ loại bỏ các thông tin không cần thiết và tối ưu hóa dữ liệu để cung cấp câu trả lời ngắn gọn và đầy đủ.

Dữ Liệu Đầu Ra: Sau khi dữ liệu đã được xử lý, hệ thống sẽ gửi dữ liệu này đến đầu ra và phản hồi lại người dùng. Dữ liệu đầu ra sẽ được hiển thị dưới dạng văn bản để người dùng có thể dễ dàng đọc và hiểu.

- Hiển thị văn bản: Câu trả lời sẽ được hiển thị trên giao diện người dùng. Hệ thống sẽ đảm bảo rằng văn bản hiển thị rõ ràng, dễ đọc và cung cấp đầy đủ thông tin cần thiết.
- Phản hồi người dùng: Người dùng sẽ nhận được câu trả lời ngay lập tức hoặc trong thời gian ngắn nhất có thể. Hệ thống sẽ đảm bảo rằng phản hồi được đưa ra một cách nhanh chóng và chính xác, giúp người dùng cảm thấy hài lòng và tin tưởng vào hệ thống.

b. Khối xử lý thông tin, dữ liệu

Khối xử lý thông tin, dữ liệu sẽ xử lý ngôn ngữ tự nhiên của người dùng được đưa vào từ khối dữ liệu đầu vào rồi chuyển các ngôn ngữ đó thành các vector để truy vấn, bao gồm các thành phần: Phowhisper Model, Embedding Model, LLM



Sơ đồ 3.3 Khối xử lý thông tin, dữ liệu

Phowhisper Model:

- Phowhisper Model là một hệ thống chuyển đổi âm thanh, cụ thể là giọng nói, thành văn bản. Đây là một loại mô hình nhận dạng giọng nói tự động (ASR - Automatic Speech Recognition) chuyên biệt cho tiếng Việt. ASR là một trong những công nghệ chủ chốt trong lĩnh vực trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên, cho phép máy tính hiểu và chuyển đổi ngôn ngữ nói thành dạng văn bản có thể xử lý được.
- Quá trình chuyển đổi này bắt đầu bằng việc nhận đầu vào là âm thanh, cụ thể là giọng nói, thông qua một khối truyền thông tin. Khối truyền thông tin này có thể là các thiết bị ghi âm, micro hoặc các hệ thống thu âm khác. Dữ liệu âm thanh sau đó được chuyển đến mô hình để xử lý.
- Sau khi nhận được dữ liệu âm thanh, Phowhisper Model sẽ tiến hành quá trình chuyển đổi giọng nói thành văn bản. Quá trình này bao gồm việc phân tích tín hiệu âm thanh, nhận diện các từ ngữ và câu, sau đó mã hóa chúng thành dạng văn bản. Văn bản này sau đó được gửi đến mô hình embedding, một loại mô hình chuyên dụng khác trong NLP để chuyển đổi văn bản thành các vector số liệu, giúp máy tính có thể xử lý và hiểu được ngữ nghĩa của văn bản một cách hiệu quả hơn.
- Trong đề tài này, mô hình được sử dụng là PhoWhisperlarge, một phiên bản cải tiến và mở rộng của mô hình PhoWhisper. PhoWhisperlarge được thiết kế để cung cấp độ chính xác cao hơn trong việc nhận dạng giọng nói tiếng Việt, nhờ vào việc sử dụng một lượng lớn dữ liệu huấn luyện và các kỹ thuật tiên tiến trong lĩnh vực học sâu.
- PhoWhisperlarge sử dụng cùng một kiến trúc với mô hình Whisperlarge-v2. Điều này có nghĩa là cả hai mô hình này chia sẻ cùng một cấu trúc mạng nơ-ron và các phương pháp huấn luyện. Kiến trúc của Whisperlarge-v2 thường bao gồm nhiều tầng (layers) của các mạng nơ-ron sâu, bao gồm các lớp convolutional và các lớp transformer, cho phép mô hình học được các đặc trưng phức tạp từ dữ liệu âm thanh và ngôn ngữ.

Embedding Model: Sử dụng để chuyển đổi các câu hỏi từ người dùng thành các vector database để gửi đến khối truy vấn.

LLM: Nhận dữ liệu được truy vấn từ khối truy vấn để tạo và xử lý ngôn ngữ tự nhiên đưa ra câu trả lời chính xác để gửi đến khối truyền thông tin, dữ liệu.

c. Khối truy vấn dữ liệu

Khối này có nhiệm vụ nhận yêu cầu và truy vấn kết quả trong bộ dữ liệu, sau đó, gửi kết quả đến LLM để tạo ra câu trả lời trên bộ dữ liệu đã được truy vấn phản hồi cho người dùng.

# CHƯƠNG 4: TRIỂN KHAI CHATBOT

## 4.1 Chuẩn bị dữ liệu

Thu thập dữ liệu về thông tin tuyển sinh của các trường đại học năm 2024

Tối ưu hóa kích thước và cấu trúc của các đoạn văn bản (chunk) để đảm bảo rằng chúng không quá lớn hay quá nhỏ, giúp duy trì ngữ cảnh cần thiết mà không vượt quá giới hạn độ dài của LLM.

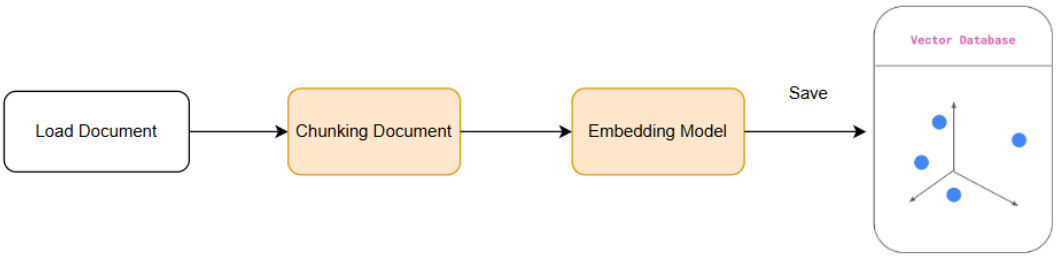
Chia nhỏ văn bản thành các đoạn văn bản có độ dài tối đa 800 ký tự, với sự trùng lặp 400 ký tự giữa các đoạn liền kề. Điều này giúp hệ thống có thể truy xuất các đoạn văn bản liên quan một cách hiệu quả và đảm bảo rằng các đoạn văn bản có đủ ngữ cảnh cho quá trình truy xuất và tạo văn bản.

```
loader_doc = TextLoader("data\hvcnbc.txt")
documents = loader_sotay.load()
text_splitter = RecursiveCharacterTextSplitter(chunk_size=800,
                                               chunk_overlap=400)

texts = text_splitter.split_documents(documents)
```

Hình 4.1.1 Mã nguồn chia nhỏ văn bản

Chuyển các dữ liệu trên thành các vector database và lưu trữ trong cơ sở dữ liệu mongodb




Sơ đồ 4.1 Sơ đồ chuyển dữ liệu thành các vector database

Dữ liệu sau khi được chia nhỏ sẽ được chuyển thành các vector database bằng cách sử dụng embedding model và lưu vào cơ sở dữ liệu.

Load Document (Tải tài liệu):

Đây là bước đầu tiên trong quy trình, nơi tài liệu được tải vào hệ thống. Tài liệu này có thể là văn bản, tệp tin, hoặc bất kỳ định dạng nào khác chứa thông tin cần xử lý.

Mục tiêu của bước này là đưa tài liệu vào hệ thống để chuẩn bị cho các bước xử lý tiếp theo.

Chunking Document (Phân đoạn tài liệu):

Sau khi tài liệu được tải vào, nó được chia thành các đoạn nhỏ hơn gọi là "chunks". Việc phân đoạn này giúp dễ dàng xử lý và phân tích tài liệu, đặc biệt khi tài liệu ban đầu có dung lượng lớn hoặc chứa nhiều thông tin phức tạp.

Quá trình phân đoạn có thể dựa trên các tiêu chí khác nhau như đoạn văn, câu, hoặc các phần thông tin logic khác.

Embedding Model (Mô hình nhúng):

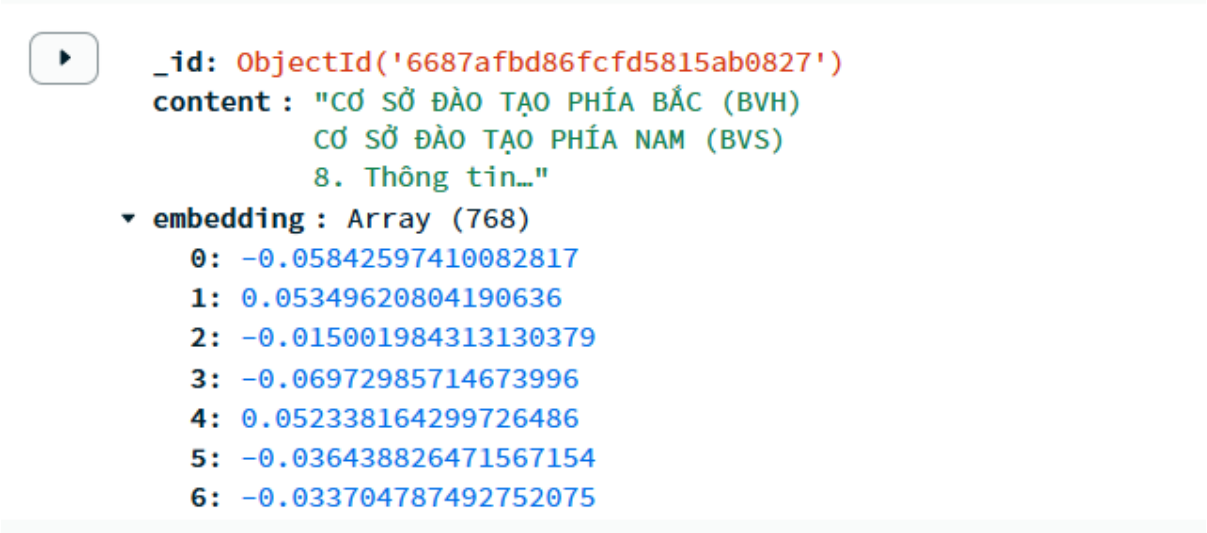
Các đoạn tài liệu sau khi được phân đoạn sẽ được đưa vào mô hình nhúng (embedding model). Mô hình này có nhiệm vụ chuyển đổi các đoạn văn bản thành các vector số liệu.

Các vector này biểu diễn nội dung của các đoạn tài liệu trong không gian đa chiều, cho phép máy tính có thể hiểu và xử lý thông tin một cách hiệu quả hơn.

Save (Lưu trữ):

Các vector sau khi được tạo ra bởi mô hình nhúng sẽ được lưu trữ trong một cơ sở dữ liệu vector (vector database).

Cơ sở dữ liệu vector cho phép lưu trữ và quản lý các vector một cách hiệu quả, hỗ trợ cho việc tìm kiếm và truy vấn thông tin sau này.




Hình 4.1.2: Dạng dữ liệu lưu trong MongoDB

Dữ liệu sẽ được lưu trữ theo dạng json:

Content: là nội dung của tài liệu được lưu trữ dưới dạng văn bản

Embedding: Phần này hiển thị một mảng các giá trị vector (embedding) được gán cho tài liệu này. Mảng này có 768 phần tử, cho thấy mô hình nhúng đã chuyển đổi nội dung văn bản thành một vector trong không gian 768 chiều. Các giá trị trong mảng này là các số thập phân, mỗi giá trị đại diện cho một thành phần trong vector. Các giá trị này được tạo ra bởi mô hình nhúng và thể hiện thông tin ngữ nghĩa của nội dung văn bản trong không gian vector.

## 4.2 Triển khai RAG

### 4.2.1 Cấu hình lớp Embedding

Định nghĩa một cấu hình và một lớp chính để tạo và sử dụng mô hình embedding từ thư viện SentenceTransformer.

```python
from pydantic.v1 import BaseModel, Field, validator
from sentence_transformers import SentenceTransformer

class EmbeddingConfig(BaseModel):
    name: str = Field(..., description="The name of the SentenceTransformer model")

    @validator('name')
    def check_model_name(cls, value):
        if not isinstance(value, str) or not value.strip():
            raise ValueError("Model name must be a non-empty string")
        return value

class MainEmbedding:
    def __init__(self, config: EmbeddingConfig):
        self.config = config
        self.embedding_model = SentenceTransformer(self.config.name)

    def encode(self, text: str):
        return self.embedding_model.encode(text)
```

Hình 4.2.1 Mã nguồn lớp embedding

EmbeddingConfig (BaseModel): Lớp này kế thừa từ BaseModel của pydantic và định nghĩa cấu hình cho mô hình embedding.

- Pydantic là một thư viện Python mạnh mẽ dùng để xác thực và cài đặt kiểu dữ liệu. Nó giúp đảm bảo rằng dữ liệu nhập vào tuân thủ theo các ràng buộc và kiểu dữ liệu đã được xác định trước.
- BaseModel là lớp cơ bản trong Pydantic mà từ đó các lớp khác có thể kế thừa. BaseModel cung cấp các tính năng cơ bản cho việc xác thực và quản lý dữ liệu.
- EmbeddingConfig là một lớp kế thừa từ BaseModel của Pydantic. Điều này có nghĩa rằng nó sẽ thừa hưởng tất cả các tính năng của BaseModel, bao gồm việc xác thực dữ liệu và cài đặt kiểu dữ liệu.
- Mục đích chính của lớp này là định nghĩa cấu hình cho mô hình nhúng (embedding model). Cấu hình này có thể bao gồm các thông số như tên mô hình, đường dẫn đến mô hình, các tham số liên quan đến quá trình huấn luyện hoặc sử dụng mô hình, và các thông tin cần thiết khác.

MainEmbedding: Lớp này chứa logic chính để sử dụng mô hình SentenceTransformer.

- SentenceTransformer là một thư viện mạnh mẽ được xây dựng trên nền tảng của thư viện transformers, được thiết kế đặc biệt để tạo ra các vector nhúng cho các câu hoặc đoạn văn bản. Nó sử dụng các mô hình học sâu tiên tiến như

BERT, RoBERTa, hoặc các biến thể của chúng để mã hóa các câu thành các vector số liệu.

- MainEmbedding là lớp chứa logic chính để sử dụng mô hình SentenceTransformer. Điều này có nghĩa là lớp này sẽ chịu trách nhiệm quản lý và sử dụng mô hình nhúng để chuyển đổi các câu hoặc đoạn văn bản thành các vector nhúng.
- Lớp này có thể bao gồm các phương thức để tải mô hình, xử lý văn bản đầu vào, và tạo ra các vector nhúng từ văn bản.

Mô hình paraphrase-multilingual-mpnet-base-v2:

- Multilingual: Mô hình này hỗ trợ nhiều ngôn ngữ, làm cho nó phù hợp cho các ứng dụng đa ngôn ngữ.
- Paraphrase: Mô hình này được huấn luyện đặc biệt để nhận biết các câu có nghĩa tương tự nhau (paraphrases), giúp cải thiện độ chính xác trong các tác vụ như tìm kiếm thông tin, phân loại văn bản, và các hệ thống khuyến nghị.
- MPNet Architecture: MPNet là một loại kiến trúc mạng nơ-ron tiên tiến, kết hợp các kỹ thuật của BERT và XLNet để cải thiện hiệu suất và độ chính xác trong việc hiểu ngữ nghĩa của văn bản

4.2.2 Cấu hình lớp RAG

a. Khởi tạo lớp RAG

```
def __init__(self,
        mongodbUri: str,
        dbName: str,
        llm,
        embeddingName: str,
    ):
    self.client = pymongo.MongoClient(mongodbUri)
    self.db = self.client[dbName]
    self.embedding_model = MainEmbedding(
        EmbeddingConfig(name=embeddingName)
    )
    self.llm = llm
```

Hình 4.2.2 Mã nguồn phương thức khởi tạo

- __init__: Hàm khởi tạo nhận vào các tham số:

- mongodbUri: URI kết nối đến MongoDB.
- dbName: Tên cơ sở dữ liệu.
- llm: Mô hình ngôn ngữ lớn (LLM) để tạo nội dung.
- embeddingName: Tên của mô hình embedding.

- Khởi tạo kết nối MongoDB và mô hình embedding:
  - self.client: Kết nối đến MongoDB.
  - self.db: Truy cập cơ sở dữ liệu cụ thể.
  - Self.embedding_model: Tạo đối tượng MainEmbedding sử dụng cấu hình EmbeddingConfig.
  - self.llm: Lưu trữ mô hình ngôn ngữ lớn.

b. Phương thức Vector Search

- Tạo embedding từ truy vấn người dùng.
- Định nghĩa pipeline cho truy vấn vector:
  - vector_search_stage: Giai đoạn tìm kiếm vector dựa trên embedding của truy vấn.
  - unset_stage: Giai đoạn loại bỏ trường embedding khỏi kết quả.
  - project_stage: Giai đoạn chọn các trường cần thiết (content, genres, score).
- Thực hiện truy vấn và trả về kết quả.

```
def vector_search(
        self,
        collection,
        user_query: str,
        limit=4):
    query_embedding = self.get_embedding(user_query)
    vector_search_stage = {
        "$vectorSearch": {
            "index": "vector_index",
            "queryVector": query_embedding,
            "path": "embedding",
            "numCandidates": 150,
            "limit": limit
                }
    }
    unset_stage = {
        "$unset": "embedding"
    }
    project_stage = {
        "$project": {
            "_id": 0,
            "content": 1,
            "genres": 1,
            "score": {
                "$meta": "vectorSearchScore"
            }
        }
    }
    pipeline = [vector_search_stage, unset_stage, project_stage]

    results = collection.aggregate(pipeline)

    return list(results)
```

Hình 4.2.3 Mã nguồn phương thức vector_search

e. Sinh văn bản từ dữ liệu đã được truy vấn

```
def generate_content(self, prompt):
    return self.llm.generate_content(prompt)
```

Hình 4.2.4 Mã nguồn tạo câu trả lời

Từ dữ liệu được truy xuất đưa vào mô hình sinh văn bản từ đó đưa ra câu trả lời cho người dùng

4.2.3 Định dạng lại câu hỏi cho phù hợp với ngữ cảnh

Tạo ra một câu hỏi độc lập từ lịch sử trò chuyện giữa người dùng và chatbot.

Lớp Reflection giúp định dạng và rút gọn lịch sử trò chuyện, sau đó sử dụng mô hình ngôn ngữ lớn để tạo ra một câu hỏi độc lập từ lịch sử đó.

```python
class Reflection():
    def __init__(self, llm):
        self.llm = llm

    def _concat_and_format_texts(self, data):
        concatenatedTexts = []
        for entry in data:
            role = entry.get('role', '')
            all_texts = ' '.join(part['text'] for part in entry['parts'])
            concatenatedTexts.append(f"{role}: {all_texts} \n")
        return ''.join(concatenatedTexts)

    def __call__(self, chatHistory, lastItemsConsidereds=100):

        if len(chatHistory) >= lastItemsConsidereds:
            chatHistory = chatHistory[len(chatHistory) - lastItemsConsidereds:]

        historyString = self._concat_and_format_texts(chatHistory)

        higherLevelSummariesPrompt = """Đưa ra lịch sử trò chuyện và câu hỏi mới nhất của
        người dùng có thể tham khảo ngữ cảnh trong lịch sử trò chuyện, hãy tạo một câu hỏi độc lập bằng
        tiếng Việt mà không cần lịch sử trò chuyện cũng có thể hiểu được.
        KHÔNG trả lời câu hỏi, chỉ sửa lại câu hỏi nếu cần và nếu không thì
        trả lại như cũ. Nếu câu hỏi không liên quan đến  thì trả về như cũ. {historyString}
        """.format(historyString=historyString)

        print(higherLevelSummariesPrompt)

        completion = self.llm.generate_content(higherLevelSummariesPrompt)

        return completion.text
```

Hình 4.2.5 Mã nguồn lớp Reflection

## 4.3 Triển khai PhoWhisper


Sơ đồ 4.2 Sơ đồ sử dụng PhoWhisper

Tạo một pipeline sử dụng mô hình "vinai/PhoWhisper-large" để thực hiện nhiệm vụ nhận dạng giọng nói tự động (ASR) và sử dụng GPU để tăng tốc quá trình xử lý.

```python
from transformers import pipeline
transcriber = pipeline("automatic-speech-recognition", model="vinai/PhoWhisper-large", device="cuda")
```

Hình 4.3.1 Tải model từ HuggingFace

```python
from flask import Flask, request, jsonify
from werkzeug.utils import secure_filename
from transformers import pipeline
from pyngrok import ngrok
import os
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = '/content/uploads'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)
@app.route('/transcribe', methods=['POST'])
def transcribe():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400
    if file:
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)
        result = transcriber(file_path)
        text = result['text']
        return jsonify({"text": text}), 200
    return jsonify({"error": "Something went wrong"}), 500
port = 5000
ngrok.set_auth_token("2i6Vs0mOw6PaGff5i2Nmf3QvRUQ_42mGwHCDECziV6g7oT3VK")
public_url = ngrok.connect(port).public_url
print(public_url)
app.run(port=port)
```

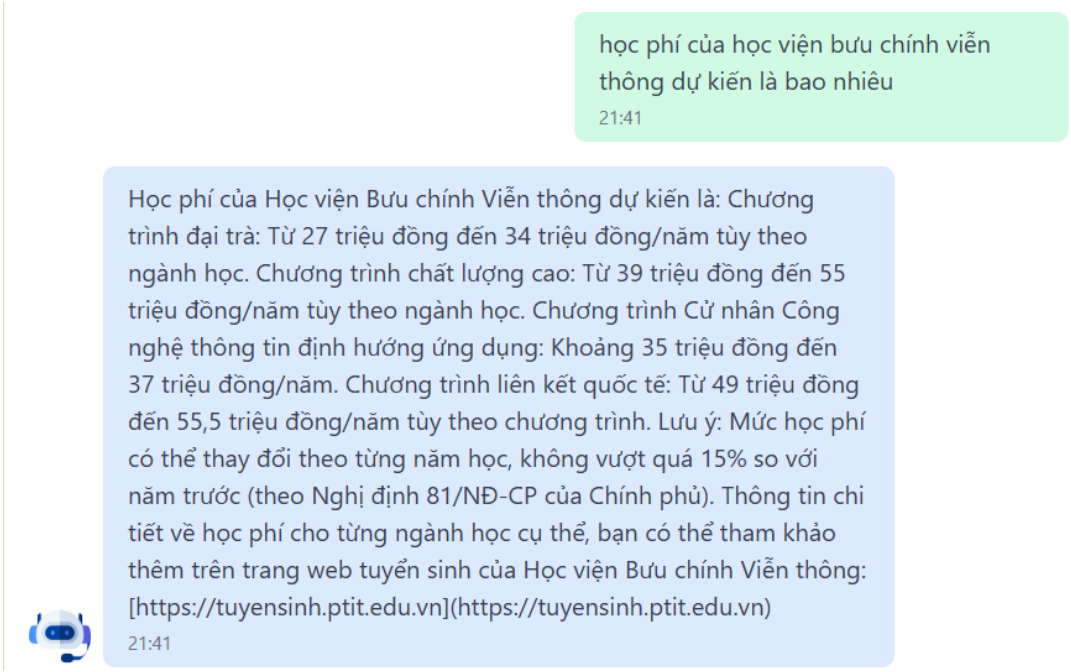
Hình 4.3.2 Mã nguồn API sử dụng model PhoWhisper

Tạo API bằng Flask để nhận tệp âm thanh, chuyển đổi âm thanh đó thành văn bản và trả về văn bản dưới dạng JSON.

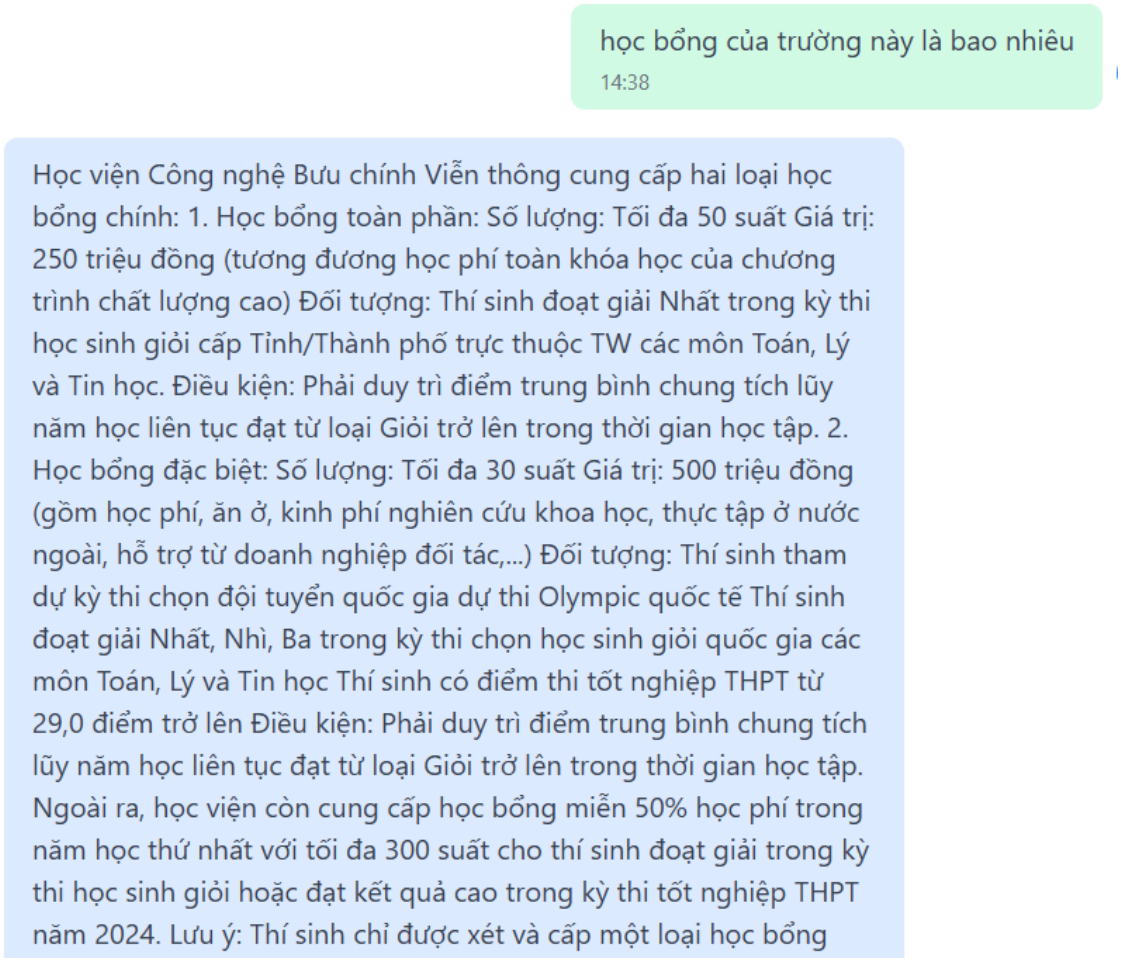
### 4.4 Kết quả thử nghiệm

Dựa vào các kết quả mà ứng dụng đã đưa ra có thể thấy là tốc độ truy xuất dữ liệu của ứng dụng khá nhanh chóng và chính xác.

Hình ảnh dưới đây cho thấy chatbot hoạt động hiệu quả trong việc cung cấp thông tin chi tiết và chính xác về học phí của Học viện Bưu chính Viễn thông. Phản hồi của chatbot không chỉ bao gồm các mức học phí cụ thể mà còn cung cấp thông tin bổ sung hữu ích như các lưu ý về sự thay đổi học phí và nguồn tham khảo chi tiết. Điều này chứng minh rằng chatbot có khả năng hiểu và đáp ứng các câu hỏi phức tạp từ người dùng một cách hiệu quả.




Hình 4.4.1: Kết quả sau của ứng dụng

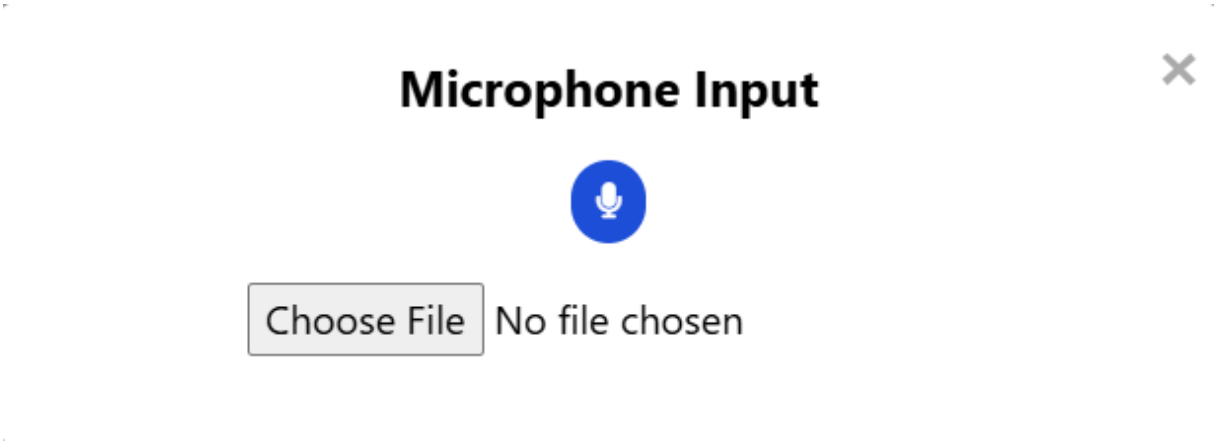



Hình 4.4.2: Ví dụ ứng dụng tạo lại câu hỏi

Khi người dùng nhập một câu hỏi mà không thể hiện rõ nội dung hoặc ngữ cảnh cụ thể, hệ thống sẽ dựa vào các câu hỏi trước đó để đưa ra các câu trả lời chính xác và mạch lạc. Điều này giúp đảm bảo rằng người dùng nhận được thông tin phù hợp và hữu ích nhất, ngay cả khi câu hỏi của họ không rõ ràng. Bằng cách này, hệ thống không chỉ đáp ứng yêu cầu hiện tại của người dùng mà còn giúp họ định hình câu hỏi một cách hiệu quả hơn, từ đó cải thiện trải nghiệm trò chuyện tổng thể.

Lớp Reflection đóng vai trò quan trọng trong việc này, với chức năng định dạng và rút gọn lịch sử trò chuyện. Sau đó, nó sử dụng một mô hình ngôn ngữ lớn để phân tích và tạo ra một câu hỏi độc lập dựa trên thông tin từ cuộc trò chuyện đã diễn ra.

Với sự trợ giúp của lớp **Reflection** và mô hình ngôn ngữ lớn, chatbot có thể cung cấp các câu trả lời chính xác và toàn diện, đồng thời giúp người dùng định hình lại câu hỏi của họ một cách rõ ràng và dễ hiểu hơn.




Hình 4.4.3 Sử dụng microphone

Ứng dụng này không chỉ hỗ trợ việc trò chuyện bằng văn bản mà còn cung cấp chức năng nhận câu hỏi bằng giọng nói. Điều này được thực hiện thông qua việc tích hợp mô hình PhoWhisper, một công cụ mạnh mẽ cho việc chuyển đổi giọng nói thành văn bản. Với chức năng này, người dùng có thể dễ dàng tương tác với hệ thống bằng cách nói thay vì gõ văn bản, tạo ra một trải nghiệm người dùng thuận tiện và tự nhiên hơn.

Khi người dùng muốn đặt câu hỏi bằng giọng nói, họ chỉ cần nhấn vào nút ghi âm trên ứng dụng và bắt đầu nói. Ứng dụng sẽ thu âm lại câu hỏi của họ và sử dụng mô hình PhoWhisper để chuyển đổi tín hiệu âm thanh này thành văn bản. PhoWhisper là một mô hình học sâu tiên tiến, được thiết kế để nhận diện và chuyển đổi giọng nói thành văn bản một cách chính xác và nhanh chóng.

## 4.5 Đánh giá Chatbot

4.5.1 Ưu điểm

- Tính tự nhiên và mạch lạc trong giao tiếp: Chatbot tư vấn tuyển sinh sử dụng RAG có khả năng tương tác rất tự nhiên và mạch lạc. RAG kết hợp giữa việc truy xuất thông tin và sinh văn bản, giúp chatbot có thể cung cấp câu trả lời chính xác và chi tiết dựa trên các tài liệu và dữ liệu đã được lưu trữ.

- Độ chính xác và nhanh chóng: Khả năng truy xuất thông tin nhanh chóng và chính xác là một điểm mạnh của chatbot này. Nhờ vào RAG, chatbot có thể tìm kiếm và tổng hợp thông tin từ nhiều trong cơ sở dữ liệu đã có sẵn trong thời gian ngắn, đảm bảo cung cấp cho người dùng các câu trả lời chính xác.

### 4.5.2 Hạn chế

- Lượng dữ liệu lưu trữ còn hạn chế: Đề tài chỉ mới thu thập được bộ dữ liệu tuyển sinh của số lượng nhỏ các trường đại học do đó khi có câu hỏi về cái thông tin khác thì chatbot sẽ đưa ra tư vấn sai.
- Giao diện người dùng: Mới chỉ làm trang kiểm thử chưa tích hợp được vào các nền tảng tuyển sinh khác.
- Khả năng xử lý âm thanh còn chậm

# KẾT LUẬN

Trong báo cáo này, đề tài đã trình bày chi tiết quá trình phát triển và triển khai chatbot tư vấn tuyển sinh sử dụng kỹ thuật Retrieval-Augmented Generation (RAG). Kết quả nghiên cứu cho thấy rằng chatbot RAG không chỉ nâng cao hiệu quả tư vấn tuyển sinh mà còn cải thiện trải nghiệm của người dùng, tiết kiệm thời gian và nguồn lực cho các trường đại học.

Qua các thử nghiệm và đánh giá, chatbot RAG đã chứng minh được khả năng cung cấp thông tin chính xác, chi tiết và nhanh chóng hơn so với phương pháp tư vấn truyền thống.

Báo cáo cũng nhấn mạnh rằng việc triển khai chatbot RAG không chỉ mang lại lợi ích về mặt hiệu quả hoạt động mà còn góp phần nâng cao uy tín và hình ảnh của các trường đại học. Sự chuyên nghiệp và tiện ích của chatbot RAG giúp xây dựng lòng tin và sự hài lòng của người dùng, từ đó thu hút nhiều thí sinh tiềm năng hơn.

Ngoài ra, công nghệ RAG còn mở ra nhiều cơ hội mới trong lĩnh vực giáo dục. Khả năng tương tác thông minh của chatbot giúp cá nhân hóa trải nghiệm tư vấn, đáp ứng nhu cầu và nguyện vọng của từng học sinh, sinh viên. Việc ứng dụng AI trong giáo dục không chỉ giới hạn ở việc tư vấn tuyển sinh mà còn có thể mở rộng sang nhiều lĩnh vực khác như hỗ trợ học tập, quản lý học sinh, và đánh giá chất lượng giáo dục.

Tóm lại, chatbot tư vấn tuyển sinh sử dụng RAG đã chứng minh được tính hiệu quả và tiềm năng to lớn trong việc cải thiện quá trình tư vấn tuyển sinh. Với những lợi ích mà nó mang lại, chatbot RAG sẽ là một công cụ hữu ích và quan trọng trong tương lai, giúp các trường đại học nâng cao hiệu quả hoạt động và đáp ứng tốt hơn nhu cầu của người học. Sự phát triển không ngừng của công nghệ AI sẽ tiếp tục mở ra nhiều cơ hội mới, và chatbot RAG sẽ là một phần quan trọng trong quá trình này.

# TÀI LIỆU THAM KHẢO

[1] M. Grinberg, Flask Web Development, United States of America.: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., 2014.

[2] M. Inc., "MongoDB," MongoDB Inc.. [Online]. Available: https://www.mongodb.com/docs/atlas/. [Accessed 27 07 2024].

[3] Amazon, "Amazon Web Services," Amazon. [Online]. Available: https://aws.amazon.com/what-is/retrieval-augmented-generation/. [Accessed 01 08 2024].

[4] G. Cloud, "What is Retrieval-Augmented Generation (RAG)?," Google, [Online]. Available: https://cloud.google.com/use-cases/retrieval-augmented-generation?hl=en. [Accessed 01 08 2024].

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pa, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv, China, 2023.

[6] LlamaIndex, "LlamaIndex.docs," LlamaIndex, [Online]. Available: https://docs.llamaindex.ai/en/stable/examples/retrievers. [Accessed 01 08 2024].

[7] Thanh-Thien Le, Linh The Nguyen, Dat Quoc Nguyen, "PHOWHISPER: AUTOMATIC SPEECH RECOGNITION FOR VIETNAMESE," ICLR 2024, Vietnam, 2024.