ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN – TIN HỌC

-----o0o----

BÁO CÁO SEMINAR

ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG CHATBOT HỖ TRỢ ĐOÀN THANH NIÊN TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

Giảng viên hướng dẫn:

PGS. TS. Nguyễn Thanh Bình

Sinh viên thực hiện:

Nguyễn Minh Hùng – 21110301

Trương Quốc Trung – 21110427

Thành phố Hồ Chí Minh, tháng 01 năm 2025

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

I. MỞ ĐẦU

1. Giới thiệu

Trong bối cảnh chuyển đổi số ngày càng diễn ra mạnh mẽ và các công nghệ ngày càng phát triển, trí tuệ nhân tạo (AI) đã và đang trở thành một trong những công nghệ mang tính cách mạng, góp phần làm thay đổi cách con người sống và làm việc. Đặc biệt trong những năm gần đây, chúng ta đã chứng kiến được sự phát triển bùng nổ AI từ khả năng nhận diện hình ảnh cho đến các mô hình học máy và học sâu. Những tiến bộ vượt bậc này không chỉ mang lại sự tiện ích mà còn mở ra sự đột phá khi ứng dụng vào trong nhiều lĩnh vực như giáo dục, y tế và kinh doanh. Nổi bật trong đó là các mô hình ngôn ngữ lớn (LLMs) đã chứng minh tiềm năng mạnh mẽ trong việc xử lý ngôn ngữ tự nhiên (NLP), hỗ trợ giao tiếp và tự động hoá các tác vụ phức tạp.

Trong môi trường giáo dục, Đoàn Thanh niên với vai trò là tổ chức chính trị xã hội của thanh niên, có nhiệm vụ theo dõi, nắm bắt tình hình học tập, sinh hoạt, tư tưởng qua đó góp phần giáo dục và rèn luyện nhân cách cho Đoàn viên, sinh viên. Điều này cho thấy sự quan trọng của tổ chức Đoàn trong môi trường Đại học. Tuy nhiên, thực tế lại cho thấy, Đoàn Thanh niên đang đối mặt với không ít thách thức trong việc tiếp cận với kết nối với sinh viên trong các hoạt động của tổ chức.

Do đó, việc đổi mới, ứng dụng các công nghệ hiện đại trong hoạt động Đoàn đang trở thành nhu cầu cấp thiết. Vì vậy, xây dựng và ứng dụng chatbot thông minh dựa trên LLMs là một giải pháp phù hợp với xu hướng phát triển hiện nay nhằm cung cấp thông tin về tổ chức đến cho Đoàn viên, thanh niên. Chatbot không chỉ có thể hỗ trợ tra cứu, trả lời các câu hỏi liên quan mà còn giúp tăng cường sự tương tác và gắn kết trong môi trường Đoàn – Hội, nắm bắt nhu cầu và nguyện vọng của sinh viên một cách nhanh chóng, từ đó nâng cao hiệu quả các phong trào của Đoàn – Hội.

2. Hướng tiếp cận

Nhu cầu tra cứu và tìm kiếm thông tin ngày càng được nâng cao, từ việc sử dụng các phương pháp tra cứu bằng cách tìm kiếm bằng từ khoá cho đến những cách phức tạp như sử dụng mô hình học máy để phân tích và trích xuất thông tin chính xác dựa trên ngữ nghĩa. Sự ra đời của các LLMs, tiêu biểu là ChatGPT – một mô hình do OpenAI phát triển đã đánh dấu sự bước tiến quan trọng trong lĩnh vực tra cứu thông tin. ChatGPT không chỉ hỗ trợ tìm kiếm

thông tin mà còn cung cấp khả năng giao tiếp dưới dạng chatbot hỏi đáp, từ đó mở đường cho sự xuất hiện của nhưng chatbot tương tự như Copilot, Gemini. Tuy nhiên, ChatGPT hay các mô hình tương tự vẫn gặp những hạn chế nhất định, đặc biệt là trong việc cập nhật thông tin mới nhất – tức bị lỗi thời về thông tin (outdated information), hay gặp vấn đề ảo giác (hallucination) khi mô hình tự sinh ra câu trả lời không phù hợp.

Hiện nay, vấn đề này đã được khắc phục nhờ các phương pháp hiện đại như mở miền dữ liệu, giúp mô hình có thể cập nhật thông tin từ nhiều nguồn khác theo theo thời gian thực. Tiêu biểu trong số đó là RAG, một kỹ thuật kết hợp với khả năng truy xuất thông tin từ cơ sở dữ liệu với LLMs, giúp tạo ra câu trả lời chính xác với ngữ cảnh.

Ở đề tài này, chúng em sẽ tìm hiểu, nghiên cứu nhằm xây dựng một hệ thống hỏi đáp về Đoàn Thanh niên. Chatbot được thiết kế dựa trên LLMs, đồng thời áp dụng kỹ thuật RAG cùng các công nghệ liên quan để tối ưu hoá việc truy xuất thông tin, khắc phục những hạn chế để có thể ứng dụng rộng rãi trong Đoàn viên, sinh viên.

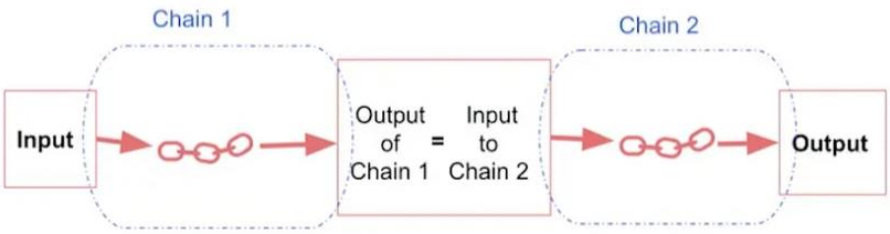
II. CƠ SỞ LÝ THUYẾT

Để xây dựng một chatbot thông minh và hiệu quả nhằm phục vụ các hoạt động của Đoàn Thanh niên, việc nghiên cứu và ứng dụng các công nghệ nền tảng hiện đại là vô cùng quan trọng. Trong phạm vi nghiên cứu này, nhóm sẽ tập trung vào ba công nghệ chính bao gồm: LangChain, RAG và Vector Database.

Ba công nghệ này không chỉ hỗ trợ lẫn nhau mà còn tạo thành một hệ thống toàn diện và mạnh mẽ, đảm bảo chatbot hoạt động hiệu quả với độ chính xác và tốc độ xử lý cao.

1. LangChain

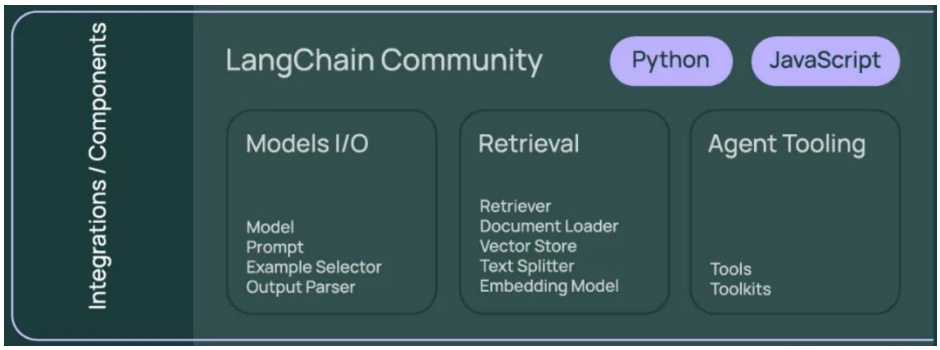
LangChain là sự kết hợp của hai từ: “Lang” – viết tắt của Language và “Chain” – tức chuỗi các mắt xích (blocks) thực hiện các tác vụ lần lượt. Mỗi mắc xích trong chuỗi đảm nhận một nhiệm vụ cụ thể: Ví dụ, mắc xích đầu tiên xử lý đầu vào (input) và tạo ra kết quả (output), sau đó kết quả này sẽ trở thành đầu vào cho mắt xích kế tiếp. Quá trình này diễn ra một cách tuần tự, tạo nên một hệ thống xử lý dữ liệu chặt chẽ và mạch lạc.



Hình 2.1: Minh hoạ cách hoạt động của chuỗi mắt xích.

Về định nghĩa, LangChain là một framework được thiết kế để hỗ trợ xây dựng và phát triển các ứng dụng dựa trên LLMs một cách đơn giản và hiệu quả, tập trung vào khả năng tương tác với dữ liệu và thực hiện các tác vụ phức tạp. LangChain cho phép tích hợp nhiều thành phần khác nhau, từ các LLMs như GPT, Llama, đến các nguồn dữ liệu bên ngoài và các công cụ xử lý có sẵn.

LangChain bao gồm các thành phần chính sau:



Hình 2.2: Các thành phần của LangChain

- Mô hình (Model): Mô hình sẽ được tải vào LangChain để thực hiện các tác vụ liên quan, giúp hiểu ý định của người dùng và sinh ra câu trả lời phù hợp.
- Lời nhắc (Prompt): Là một hoặc những câu được đưa cho LLMs giúp mô hình hiểu được ý đồ của người dùng cần thực hiện, giúp tối ưu hoá câu trả lời của mô hình. Lời nhắc thường sẽ định hình cho LLMs biết nhiệm vụ và cách đưa ra câu trả lời.

  Ví dụ:
  - "Bạn là một trợ lý AI được thiết kế để trả lời câu hỏi bằng tiếng Việt".
  - "Bạn được cung cấp các tài liệu văn bản dưới đây, từ đó hãy trả lời các câu hỏi".
  - "Nếu không có thông tin hãy trả lời rằng không có thông tin".
- Truy xuất (Retrieval): Thành phần này hỗ trợ trong việc truy vấn cơ sở dữ liệu vector (Vector Database) để lấy thông tin cần thiết trả lời cho người dùng.

- Tải văn bản (Document Loader): Công cụ giúp tự động tải và xử lý các văn bản, giúp tiết kiệm thời gian khi xử lý lượng dữ liệu lớn.
- Cơ sở dữ liệu vector (Vector Store): Thành phần lưu trữ các Vector Embedding của từng loại văn bản, giúp truy vấn các văn bản phù hợp khi người dùng đưa ra yêu cầu.
- Chia văn bản (Text Splitter): Thay vì lưu trữ một đoạn văn bản dài, Text Splitter cho phép chia nhỏ văn bản thành các đoạn nhỏ hơn và lưu chúng trong cơ sở dữ liệu, giúp quá trình tìm kiếm và truy xuất trở nên hiệu quả hơn.
- Mô hình Embedding (Embedding Model): Mô hình này dùng để chuyển đổi các văn bản thành Vector Embedding. Khi thực hiện truy vấn, hệ thống sẽ so sánh các vector này để tìm ra văn bản phù hợp nhất với yêu cầu, từ đó LLMs có thể sử dụng kết quả này để tạo ra câu trả lời chính xác.

Tóm lại, ta có thể thấy rằng LangChain được cấu thành từ nhiều thành phần và hoạt động liền mạch với nhau, tạo thành một quy trình hoàn chỉnh, đóng vai trò quan trọng trong việc xây dựng một chatbot thông minh và đáng tin cậy.

2. Retrieval-Augmented Generation (RAG)

RAG là một kỹ thuật kết hợp hai khả năng quan trọng của LLMs là khả năng truy xuất (Retrieval) và khả năng tạo sinh (Generation). Thay vì chỉ dựa vào dữ liệu học ban đầu, RAG cho phép mô hình truy xuất và sử dụng thông tin từ các nguồn tri thức bên ngoài như các Vetor Database chứa thông tin được mã hoá từ tài liệu hoặc cơ sở dữ liệu. Thông tin này được sử dụng làm ngữ cảnh để bổ sung và làm phong phú thêm nội dung sinh ra, mang lại các câu trả lời phù hợp, cải thiện độ chính xác và tính linh hoạt của mô hình sinh.

Một trong những ưu điểm lớn của RAG là khả năng khắc phục vấn đề ảo giác (hallucination), khi mà các mô hình sinh đôi khi tự sinh ra thông tin không chính xác hoặc không tồn tại trong dữ liệu huấn luyện ban đầu. Ngoài ra, một điểm mạnh khác của RAG là tính mở rộng (scalability), khi có thêm dữ liệu mới, hệ thống chỉ cần cập nhật hoặc bổ sung vào Vector Database mà không cần tái huấn luyện mô hình, giúp tiết kiệm thời gian và tài nguyên tính toán, đảm bảo mô hình luôn có các thông tin mới nhất.

Theo một nghiên cứu của Quang Nguyen và cộng sự [1], nhóm nghiên cứu đã đánh giá hiệu suất của RAG trên bốn mô hình, bao gồm GPT-4 và ba mô hình nguồn mở (Llama-3-70B, Gemma-2-27B và Mixtral-8x7B). Họ đã áp dụng kỹ thuật RAG để trả lời 260 câu hỏi về nhãn

khoa từ hai bộ dữ liệu từ Chương trình Đánh giá của Viện Hàn lâm Nhãn khoa Hoa Kỳ (BCSC Self-Assessment) và Ngân hàng câu hỏi nhãn khoa OphthoQuestions. Kết quả thử nghiệm cho thấy, việc tích hợp RAG vào các mô hình đã mang lại sự cải thiện đáng kể về độ chính xác câu việc trả lời câu hỏi.

Model	mode	BCSC	Ophtho Questions	Mean
GPT-4-turbo	ZRS	80.38	77.69	79.03
	ZRS-CoT	81.54 (1.16↑)	79.62 (1.93↑)	80.58 (1.55↑)
	RAG	91.92 (11.54↑)	85.38 (7.69↑)	88.65 (9.62↑)
Llama-3-70B-Q4	ZRS	64.62	50.38	57.50
	ZRS-CoT	70.77 (6.15↑)	65.77 (15.39↑)	68.27 (10.77↑)
	RAG	84.62 (20.0↑)	78.08 (27.7↑)	81.35 (23.85↑)
Gemma-2-27B-Q4	ZRS	64.23	60.0	62.12
	ZRS-CoT	61.54 (-2.69↓)	56.92 (-3.08↓)	59.23 (-2.89↓)
	RAG	83.46 (19.23↑)	75.0 (15.0↑)	79.23 (17.11↑)
Mixtral-8x7B-Q4	ZRS	57.69	48.08	52.89
	ZRS-CoT	53.85 (-3.84↓)	52.69 (4.61↑)	53.27 (0.385↑)
	RAG	78.46 (20.77↑)	71.54 (23.46↑)	75.00 (22.11↑)
Antanki et al. 2023 [2] (GPT-4, temparature=0.3)	ZSR	75.8	70.8	71.7

Bảng 1: Bảng so sánh độ chính xác khi không có và có RAG [1].

Những kết quả này chứng minh rằng RAG là một giải pháp mạnh mẽ trong việc tăng cường độ chính xác của các LLMs, làm giảm thiểu sai sót trong quá trình trả lời các câu hỏi phức tạp.

3. Vector Database

Vector Database là một loại cơ sở dữ liệu đặc biệt được thiết kế để lưu trữ và quản lý các vector có nhiều chiều. Các vector này thường là Embedding – tức là biểu diễn toán học của văn bản hoặc dữ liệu mà LLMs sinh ra nhằm chuyển đổi thành các dạng có thể xử lý và so sánh một cách hiệu quả trong không gian toán học.

Với mỗi điểm dữ liệu, như là một đoạn văn bản hay một câu hỏi, sẽ được chuyển đổi thành một vector có độ dài cố định, và cơ sở dữ liệu này lưu trữ tất cả các vector tương ứng. Khi thực hiện tìm kiếm, hệ thống sẽ sử dụng các thuật toán đo lường sự tương đồng giữa các vector để trả về các kết quả gần nhất. Có thể sử dụng các phép đo khoảng cách như:

- Độ tương đồng Cosine (Cosine Similarity):

  Công thức tính độ tương đồng Cosine giữa hai vector A và B như sau:

$$Cosine\ Similarity(A,B) = \frac{A.B}{\|A\|.\|B\|} = \frac{\sum_{i=1}^{n}(A_i.B_i)}{\sqrt{\sum_{i=1}^{n}A_i^2}.\sqrt{\sum_{i=1}^{n}B_i^2}}$$

Phạm vi của giá trị trong khoảng $[-1, 1]$, độ tương đồng có giá trị 1 cho thấy hai vector hoàn toàn giống nhau (đồng hướng), với 0 cho thấy hai vector không liên quan, và -1 nghĩa là hai vector hoàn toàn độc lập tức trái nghĩa hoàn toàn.

- Khoảng cách Euclidean (Euclidean Distance):

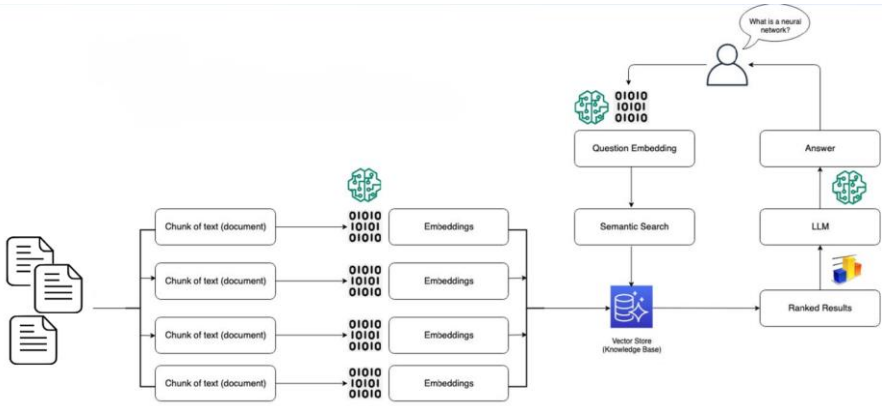
Với hai điểm bất kỳ trong toạ độ Descartes cho trước trong không gian Euclid n chiều, cho điểm p có toạ độ $(p_1, p_2, \dots, p_n)$ và điểm q có toạ độ $(q_1, q, \dots, q_n)$. Khoảng cách giữa chúng là:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 + q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}$$

Một trong những ứng dụng nổi bật của Vector Database là hệ thống tìm kiếm thông minh, nơi thông tin từ nhiều nguồn được chuyển thành các vector và lưu trữ trong cơ sở dữ liệu để nhanh chóng truy xuất và phân tích. Các vector này có thể được tạo ra bằng cách sử dụng các mô hình như Word2Vec, GloVe hay BERT.

III. QUY TRÌNH THỰC NGHIỆM

Để triển khai và đánh giá hiệu quả mô hình RAG, chúng ta cần chuẩn bị bộ dữ liệu và thiết kế một quy trình nhiều bước nhằm đảm bảo tính toàn diện của mô hình. Quy trình sẽ bao gồm ba giai đoạn chính: Biến đổi dữ liệu – Xây dựng hệ thống truy xuất – Tạo phản hồi.



Hình 3.1: Quy trình thực hiện kỹ thuật RAG.

1. Biến đổi dữ liệu (Ingestion)

Trong đề tài này, nhóm sẽ xây dựng bộ dữ liệu bao gồm các nội dung liên quan như sau:

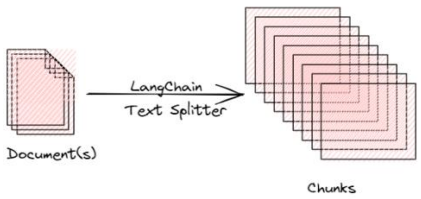
- Nghị quyết Đại hội Đảng bộ các cấp.
- Cuộc đời và sự nghiệp của Chủ tịch Hồ Chí Minh, các tác phẩm, những phẩm chất đạo đức của Bác.
- Các thông tin về Đoàn Thanh niên, Hội Sinh viên như các văn bản Nghị quyết, Điều lệ tổ chức, lịch hoạt động …
- Những thông tin trong Nhà trường như thông báo, nội quy, quy định, sổ tay sinh viên.

Trong quá trình tìm hiểu và xây dựng bộ dữ liệu, khi sử dụng những tài liệu PDF sẽ dễ mắc phải những lỗi như dính chữ, rời chữ, lỗi kiểu chữ do định dạng dữ liệu như kiểu chữ. Mặc dù đã có các công cụ có thể hỗ trợ đọc tài liệu PDF nhưng chưa thực sự tối ưu trong việc hạn chế những lỗi trên. Do đó, nhóm sẽ xây dựng và sử dụng những tài liệu Word hoặc PDF, và định dạng dữ liệu theo các tiêu chí sau:

- Chỉ bao gồm văn bản.
- Các đoạn văn nên nằm gọn trong một trang.
- Nên sử dụng tiêu đề cho mỗi đoạn, mỗi đoạn biểu diễn một thông tin cụ thể và phân biệt nhau bằng các ký tự đặc biệt (ví dụ: “#”, “###”).

a. Chia nhỏ tài liệu (Chunking)

Các file tài liệu sẽ được chia nhỏ bằng phương pháp Recursive Character Text Splitter. Phương pháp này sẽ chia tách tài liệu theo ký tự, đảm bảo mội đoạn có độ dài nằm trong giới hạn nhất định. Phương pháp này rất hữu ích đối với các tài liệu có ngắt đoạn hoặc ngắt câu tự nhiên, giúp tối ưu hoá khả năng tìm kiếm.



Hình 3.2: Minh hoạt việc chia văn bản ban đầu thành chunks.

b. Vector Embedding

Sau khi chia nhỏ tài liệu, các chunk sẽ được mã hoá (encode) thành các Vector Embedding bằng cách sử dụng các mô hình hiện đại như:

- Sentence Transformers
- OpenAI Embedding
- Multilingual SBERT

Các Vector Embedding biểu diễn dạng toán học của văn bản sẽ giúp văn bản dễ dàng tìm kiếm dựa trên ngữ nghĩa thay vì chỉ dựa vào từ khoá. Các vector này sau đó được lưu trữ vào trong Vector Database để phục vụ quá trình truy vấn.

c. Chọn Vector Database

Trong nghiên cứu của nhóm Xiaohua Wang và cộng sự [2] về việc tìm kiếm Vector Database tốt nhất cho RAG, Milvus được đánh giá là một giải pháp toàn diện trong việc tìm kiếm nhờ vào hỗ trợ đa dạng loại chỉ mục (Multiple Index Type), khả năng xử lý dữ liệu quy mô lớn (Billion-Scale), tìm kiếm kết hợp (Hybrid Search) và tích hợp lên các hệ thống đám mây (Cloud-Native). Ngoài ra, Qdrant cũng được nhóm nghiên cứu đánh giá cao nhờ vào khả năng linh hoạt và dễ sử dụng hơn khi cung cấp API tích hợp vào các ứng dụng, có khả năng xử lý tốt trên môi trường dữ liệu lớn, dễ tuỳ chỉnh chỉ mục hay tham số tìm kiếm và tốt cho các dự án vừa và nhỏ khi không yêu cầu lớn tài nguyên hệ thống.

Database	Multiple Index Type	Billion-Scale	Hybrid Search	Cloud-Native
Weaviate	✗	✗	✓	✓
Faiss	✓	✗	✗	✗
Chroma	✗	✗	✓	✓
Qdrant	✗	✓	✓	✓
Milvus	✓	✓	✓	✓

Bảng 2: Bảng so sánh các Vector Database khác nhau [2].

Như vậy, Vector Database đóng một vai trò nền tảng trong hệ thống RAG, đảm bảo khả năng lưu trữ và truy xuất thông tin. Và trong đề tài này, nhóm sẽ sử dụng **Qdrant** vào trong kỹ thuật RAG, đây là lựa chọn tốt với nhiều điểm mạnh, đồng thời tiết kiệm chi phí và tài nguyên nhưng vẫn có thể duy trì hiệu suất tìm kiếm cao.

2. Xây dựng hệ thống truy xuất (Retrieval)

Sau quá trình biến đổi dữ liệu hoàn tất, truy xuất (retrieval) là thành phần cốt lõi trong hệ thống RAG, đóng vai trò như bộ nhớ ngoài của chatbot và chịu trách nhiệm truy xuất thông tin liên quan từ cơ sở dữ liệu.

Vai trò của Retrieval:

- Nâng cao hiệu suất trả lời câu hỏi: Giảm không gian tìm kiếm, giúp hệ thống hoạt động hiệu quả hơn.
- Cải thiện độ chính xác: Tăng chất lượng câu trả lời bằng cách truy xuất các tài liệu liên quan một cách ngữ nghĩa.

Trong hệ thống này, ta sẽ chọn phương pháp kết hợp giữa Vector–Based Retrieval và Deep Learning–Based Reranking để đảm bảo tính hiệu quả và độ chính xác cao.

a. Vector-Based Retrieval

Vector–Based Retrieval là một bước tiến lớn trong truy xuất thông tin, tận dụng khả năng hiểu ngữ nghĩa để tìm kiếm chính xác hơn.

Quy trình thực hiện:

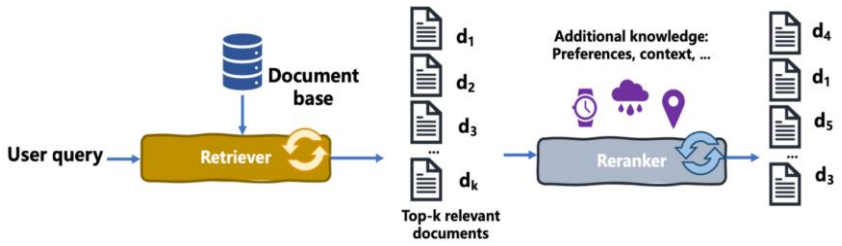
- Chuyển đổi truy vấn: Khi người dùng nhập truy vấn, hệ thống sẽ chuyển truy vấn đó thành vector thông qua các mô hình học máy (ví dụ: Word2Vec, TF-IDF hay BERT). Tương tự, mỗi tài liệu trong cơ sở dữ liệu cũng được chuyển thành vector.
- Đánh giá độ tương đồng: Vector của truy vấn được so sánh với các vector tài liệu trong cơ sở dữ liệu bằng các phương pháp tính độ tương đồng như Cosine Similarity.
- Lọc tài liệu: Dựa trên kết quả sự tương đồng, hệ thống có nhiệm vụ truy xuất ra k tài liệu (top-k documents) có độ tương đồng cao nhất, tức là các tài liệu có thông tin liên quan nhất đến truy vấn của người dùng.

b. Deep Learning-Based Reranking

Deep Learning–Based Reranking là phương pháp nâng cao sử dụng mô hình học sâu để đánh giá lại danh sách tài liệu ban đầu để cải thiện thứ hạng, dựa trên độ liên quan ngữ nghĩa của tài liệu với truy vấn.

Quy trình thực hiện:

- Đầu vào: Nhận danh sách tài liệu từ Vector–Based Retrieval.
- Xếp hạng lại: Sử dụng mô hình học sâu (Ví dụ: Mạng nơ-ron nhân tạo) để đánh giá mức độ liên quan giữa từng tài liệu và truy vấn. Mô hình có thể học và xác định thứ hạng dựa trên mức độ liên quan mà mô hình dự đoán cho tài liệu.
- Kết quả: Danh sách tài liệu đã được sắp xếp lại theo độ chính xác cao hơn.

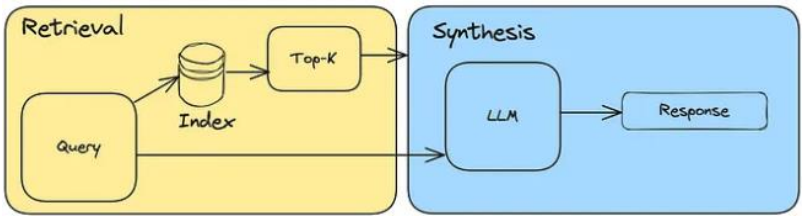


Hình 3.3: Minh hoạ quy trình trong bước xây dựng hệ thống truy xuất.

Sự kết hợp giữa Vector-Based Retrieval và Deep Learning-Based Reranking không chỉ đảm bảo hệ thống tìm kiếm ngữ nghĩa hiệu quả mà còn tăng chất lượng kết quả trả lời, giúp quy trình có thể mang lại hiệu suất cao cho chatbot.

3. Tạo phản hồi (Response Generation)

Bước cuối cùng trong hệ thống RAG là quá trình tạo phản hồi (Response Generation). Đây là giai đoạn mà hệ thống tổng hợp các thông tin đã được truy xuất từ cơ sở dữ liệu với tri thức có sẵn trong LLMs để đưa ra phản hồi.



Hình 3.4: Minh hoạ quy trình trong bước tạo ra phản hồi.

Quá trình này đảm bảo rằng các phản hồi đưa ra phù hợp với ngữ cảnh truy vấn của người dùng, tích hợp đa nguồn tri thức, phản hồi chính xác và mang tính trò chuyện, tạo trải nghiệm thân thiện cho người dùng. Hơn nữa, các phản hồi này không mang tính tiêu cực hay chứa những nội dung nhạy cảm để phù hợp trong môi trường Đoàn Thanh niên.

Với những nguồn tri thức và truy vấn được tìm thấy, chúng sẽ được lọc qua mô hình phân loại cảm xúc, nếu mô hình đưa ra đánh giá là tiêu cực (Negative), chatbot sẽ từ chối câu hỏi. Ngược lại, những nguồn tri thức và truy vấn trên sẽ được đưa vào mô hình Text Generation và đưa ra câu trả lời.

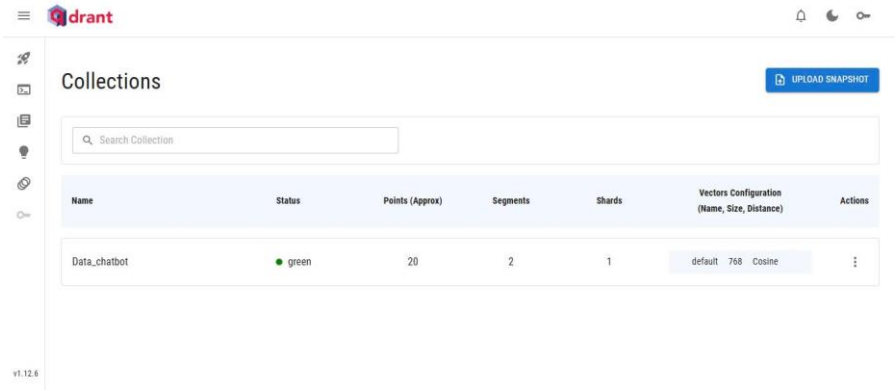
Khi tạo phản hồi, một trong những yếu tố quan trọng là xây dựng prompt đầu vào cho LLMs. Prompt này thường bao gồm top-k chunks được truy xuất trong bước trước đó. Việc bố trí chiến lược các thông tin trong prompt, đặc biệt là đặt các thông tin quan trọng ở đầu hoặc cuối chuỗi đầu vào có thể cải thiện hiệu quả hoạt động của hệ thống. Theo nghiên cứu của Nelson F. Liu và cộng sự [3], việc tổ chức thông tin theo cách này giúp LLMs xử lý ngữ cảnh dài (long contexts) tốt hơn. Nghiên cứu chỉ ra rằng các mô hình có xu hướng ưu tiên thông tin ở đầu và cuối chuỗi, trong khi thông tin ở giữa dễ bị bỏ qua.

Đây là giai đoạn giúp RAG trở nên hoàn thiện hơn trong việc chuyển đổi thông tin thành giá trị thực tế cho người dùng. Đây cũng là bước quyết định hiệu quả tổng thể của hệ thống, phản ánh sự kết hợp chặt chẽ giữa khả năng truy xuất thông tin và tạo ngôn ngữ của LLMs.

IV. KẾT QUẢ NGHIÊN CỨU

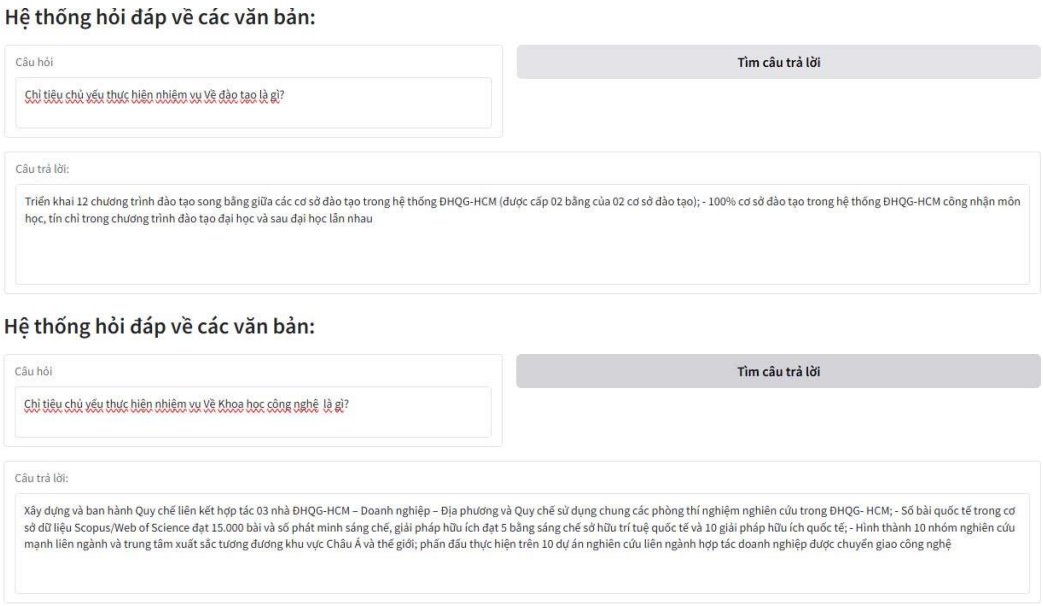
Trong quá trình tìm hiểu bài toán, nhóm đã sử dụng dữ liệu với thông tin về *Nghị quyết Đại hội Đảng bộ Đại học Quốc gia Thành phố Hồ Chí Minh lần thứ VI, nhiệm kỳ 2020 – 2025*, nhóm đã thực hiện tiền xử lý dữ liệu để có được dữ liệu đúng định dạng.

Sau đó, nhóm đã thực hiện việc mã hoá dữ liệu bằng mô hình Sentence Transformers đa ngôn ngữ (MPNet-base-v2) và lưu trữ vào Vector Database Qdrant thành công:



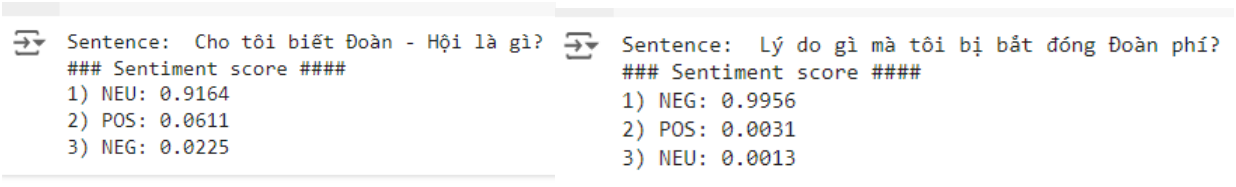
Hình 4.1: Giao diện Qdrant sau khi tạo thành công.

Trong nghiên cứu, nhóm đã tìm hiểu, xây dựng và chạy thử chương trình với mô hình BARTpho được tinh chỉnh (fine-tuning) theo kiểu Question Answering, kết quả mô hình trả về đúng câu trả lời với dữ liệu tìm kiếm được từ Qdrant. Tuy nhiên, mô hình vẫn chưa trả lời được các đoạn văn dài.



Hình 4.2: Các kết quả khi thử nghiệm trên mô hình BARTpho.

Ngoài ra, nhóm đã thử nghiệm thành công với mô hình phân loại cảm xúc cho tiếng Việt (5CD-ViSoBERT for Vietnamese Sentiment Analysis) nhằm bước đầu hiểu được cách mô hình hoạt động và khả năng lọc câu hỏi là tích cực (Positive – POS), tiêu cực (Negative – NEU) hay trung lập (Neutral – NEU). Từ đó, tạo tiền đề để áp dụng mô hình phân loại cảm xúc cho chatbot.



Hình 4.3: Các kết quả phân loại khi thử nghiệm trên mô hình 5CD-ViSoBERT).

Những kết quả trên đã giúp nhóm định hình cách mà một chatbot LLMs hoạt động, từ đó nhóm sẽ tiếp tục xem xét, khắc phục những hạn chế ban đầu và đưa ra hướng phát triển phù hợp hơn.

V. THẢO LUẬN

1. Thách thức

Việc triển khai hệ thống cùng RAG mang lại tiềm năng trong việc nâng cao hiệu suất của chatbot. Tuy nhiên, trong quá trình nghiên cứu, nhóm đã gặp phải một số thách thức chính:

- Xử lý dữ liệu thủ công: Quy trình xử lý dữ liệu hiện tại còn phụ thuộc nhiều vào thao tác thủ công, dẫn đến mất thời gian và dễ xảy ra lỗi.
- Kích thước chunk: Việc xác định kích thước chunk phù hợp là bài toán phức tạp, vì chunk lớn cung cấp bối cảnh đầy đủ hơn nhưng làm tăng thời gian xử lý, trong khi chunk nhỏ giúp tối ưu hoá thời gian nhưng có thể làm mất thông tin quan trọng.
- Kiểm soát nội dung phản hồi: Cần đảm bảo các phản hồi của chatbot luôn lịch sự, chính xác và tránh các thông tin nhạy cảm.
- Tối ưu hoá hệ thống: Mặc dù hệ thống RAG không quá phức tạp, nhưng việc tinh chỉnh và đạt được hiệu suất tối ưu trong thực tế đòi hỏi nhiều công sức và thời gian.

2. Định hướng phát triển

Để khắc phục các thách thức và nâng cao chất lượng hệ thống, nhóm sẽ tập trung vào các mục tiêu sau:

- Tự động hoá quy trình xử lý dữ liệu: Nghiên cứu và áp dụng các công nghệ phù hợp để tự động hoá việc xử lý dữ liệu. Đồng thời, cần tối ưu hoá phương pháp chunking nhằm đạt được hiệu suất và độ chính xác cao.
- Cải thiện khả năng phân loại và đánh giá nội dung: Tìm kiếm và thử nghiệm các LLMs phù hợp cho ngôn ngữ tiếng Việt, đặc biệt là dùng những mô hình có khả năng phân loại, đánh giá câu hỏi và câu trả lời, từ đó hệ thống có quyền trả lời hoặc từ chối câu hỏi để đảm bảo tính tích cực và không chứa thông tin nhạy cảm.
- Tích hợp và mở rộng hệ thống: Xây dựng giao diện thân thiện với người dùng, tích hợp chatbot để Đoàn viên, sinh viên có thể dễ dàng tương tác với hệ thống. Ngoài ra, phát triển các tính năng bổ sung có thể hỗ trợ trong học tập và tư vấn sức khoẻ tinh thần cho sinh viên.
- Tối ưu hoá mô hình: Tiếp tục tối ưu hoá quy trình RAG để cải thiện hiệu suất, giảm thời gian xử lý và tăng độ chính xác, đáp ứng các nhu cầu trong tương lai.

3. Kết luận

Nhóm chúng em đã tìm hiểu và nghiên cứu về quá trình tạo ra một chatbot thông minh, trong đó có ứng dụng kỹ thuật RAG trong việc tối ưu hoá hệ thống hỏi đáp dựa trên các LLMs, đặc biệt là khi tìm hiểu bài toán trong chủ đề cụ thể về Đoàn Thanh niên. Sự kết hợp RAG và LLMs cho thấy tiềm năng lớn trong việc xây dựng chatbot đáp ứng với các nhu cầu hiện tại khi mang lại độ chính xác cao. Kết quả nghiên cứu này không chỉ khẳng định hiệu quả của việc áp dụng RAG hay các công nghệ liên quan mà từ đó có thể tìm hiểu và khai thác sâu hơn những ứng dụng tiềm năng của chatbot trong nhiều lĩnh vực, đặc biệt trong môi trường Đoàn – Hội khi chatbot không chỉ hỗ trợ tra cứu mà còn có thể nâng cao sự tương tác giữa Đoàn viên và tổ chức.

Nhóm hy vọng rằng kết quả nghiên cứu trên sẽ là bước khởi đầu để tiếp tục xây dựng và phát triển một chatbot hoàn chỉnh trong tương lai để ứng dụng rộng rãi nhằm phục vụ Đoàn viên và sinh viên.

TÀI LIỆU THAM KHẢO

[1] Nguyen, Q., Nguyen, D.-A., Dang, K., Liu, S., Nguyen, K., Wang, S. Y., Woof, W., Thomas, P., Patel, P. J., Balaskas, K., Thygesen, J. H., Wu, H., & Pontikos, N. (2024). Advancing Question-Answering in Ophthalmology with Retrieval Augmented Generations (RAG): Benchmarking Open-source and Proprietary Large Language Models. *medRxiv*, 2024-11. https://doi.org/10.1101/2024.11.18.24317510.

[2] Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., Lv, C., Zheng, X., & Huang, X. J. (2024). Searching for best practices in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, (pp. 17716–17736), Miami, Florida, USA: Association for Computational Linguistics.

[3] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, 157-173.