Xây dựng hệ thống chatbot hỏi đáp về thông tin các môn học của UIT bằng RAG

Lê Minh Nhựt
22521060

Trần Đình Khánh Đăng
22520195

Nguyễn Hùng Phát
22521074

Abstract

Hệ thống chatbot là một giải pháp hữu ích trong việc cung cấp thông tin nhanh chóng và tiện lợi. Đề tài này tập trung xây dựng hệ thống chatbot hỗ trợ hỏi đáp thông tin các môn học của trường Đại học Công nghệ Thông tin (UIT) dựa trên phương pháp Retrieval-Augmented Generation (RAG). Chúng tôi trình bày cách thu thập dữ liệu, xử lý thông tin, triển khai mô hình và đánh giá hiệu quả hệ thống. Hệ thống chatbot này không chỉ giúp sinh viên dễ dàng tra cứu thông tin mà còn góp phần nâng cao trải nghiệm học tập và sử dụng công nghệ trong giáo dục. Mã nguồn được triển khai tại https://github.com/leeminsun1205/CS336.P11.git.

1 Giới thiệu bài toán

1.1 Lý do chọn đề tài

Trong bối cảnh giáo dục hiện đại, việc truy cập và sử dụng thông tin một cách nhanh chóng là rất cần thiết. Tuy nhiên, sinh viên thường gặp khó khăn khi tra cứu thông tin các môn học như lịch học, nội dung môn học, hoặc yêu cầu học phần. Hệ thống chatbot hỏi đáp tự động không chỉ giúp tiết kiệm thời gian mà còn cải thiện hiệu quả tìm kiếm thông tin.

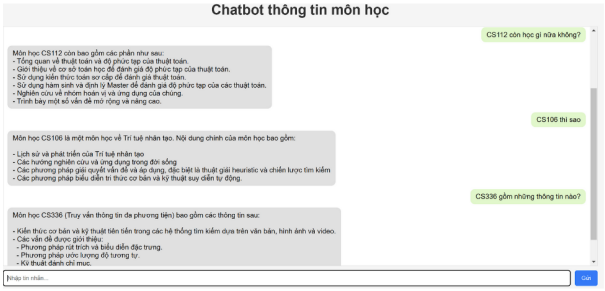
1.2 Phát biểu bài toán

Input: Câu hỏi từ người dùng liên quan đến thông tin các môn học tại UIT.
Output: Câu trả lời chính xác và phù hợp từ hệ thống chatbot (Hình 1).

2 Các nghiên cứu liên quan

Các nghiên cứu gần đây về chatbot sử dụng RAG đã cho thấy hiệu quả cao trong việc xử lý câu hỏi tự nhiên và cung cấp câu trả lời dựa trên cơ sở dữ liệu lớn. Phương pháp RAG kết hợp khả năng tìm kiếm thông tin từ các tài liệu với sức mạnh sinh câu của các mô hình ngôn ngữ lớn như BERT hoặc GPT.

3 Phương pháp



Hình 1: Ví dụ về câu hỏi và câu trả lời từ chatbot.

Trước khi đi vào phương pháp, ta cần nhắc lại quy trình truy xuất thông tin. Quy trình này bao gồm các bước sau:

Bước 1: Truyền tin nhắn

Người dùng sẽ truyền một tin nhắn (message) cho ChatBot.

Bước 2: Tạo truy vấn

Sau khi nhận tin nhắn từ người dùng, ChatBot sẽ tạo một query (truy vấn) để tìm kiếm thông tin liên quan đến nội dung câu hỏi. Truy vấn này được chuyển đến bộ lưu trữ dữ liệu (Storage) để tìm các tài liệu phù hợp.

Bước 3: Truy xuất thông tin

Storage sẽ tìm kiếm và retrieve (truy xuất) các tài liệu hoặc đoạn văn bản liên quan đến query từ cơ sở dữ liệu. Các tài liệu này đã được nhúng (embed) trước đó, tức là chuyển đổi thành các vector để tìm kiếm nhanh và hiệu quả.

Bước 4: Sắp xếp và phản hồi

Các đoạn văn bản tìm được sẽ được hệ thống rerank (sắp xếp lại) để xác định mức độ liên quan với câu hỏi. Kết quả này được tổng hợp và đưa ra phản hồi (response) cho người dùng dựa trên dữ liệu đã truy xuất.

Từ quy trình trên, hệ thống ChatBot không chỉ đảm bảo tìm kiếm thông tin nhanh mà còn cung cấp câu trả lời chính xác và đầy đủ nhờ tính tổng hợp.

3.1 Chuẩn bị dữ liệu lưu trữ

Để có dữ liệu trong Storage, ta cần chuẩn bị nguồn dữ liệu đầu vào, thường là các tài liệu hoặc tập hợp văn bản chứa thông tin liên quan. Vì máy tính không hiểu ngôn ngữ tự nhiên, các tài liệu cần được chuyển đổi thành định dạng mà hệ thống có thể xử lý.

Cụ thể, văn bản sẽ được nhúng (embed) thành các vector số, biểu diễn trong không gian nhiều chiều để máy tính xử lý. Quá trình này sử dụng các kỹ thuật học máy hoặc mô hình ngôn ngữ để chuyển đổi văn bản thành đoạn mã số phục vụ truy vấn.

Các bước nhúng dữ liệu:

1. **Tiếp nhận (Ingest):** Nhập các tài liệu từ nhiều nguồn khác nhau (ví dụ: giáo trình, tài liệu môn học, câu hỏi thường gặp).

2. **Nhúng (Embed):** Chuyển đổi tài liệu hoặc đoạn văn bản thành vector nhúng, sử dụng mô hình ngôn ngữ như BERT, GPT hoặc các mô hình khác.

3. **Lưu trữ (Storage):** Lưu các vector nhúng vào hệ thống để phục vụ truy vấn.

3.2 Tạo vector embedding

Việc tạo vector embedding đóng vai trò quan trọng vì nó biến đổi văn bản từ ngôn ngữ tự nhiên thành các vector số học, giúp hệ thống so sánh và tìm kiếm tương đồng giữa query và dữ liệu lưu trữ.

Nếu vector embedding không phản ánh đúng ngữ nghĩa, ChatBot sẽ không thể truy vấn và tìm kiếm chính xác. Nhóm sử dụng **OpenAIEmbedding** để tạo vector đại diện cho các câu trả lời đúng (true_context).

Lý do chọn OpenAIEmbedding:

- Mô hình này được phát triển từ các mô hình ngôn ngữ lớn (LLMs) của OpenAI.

- Hiệu quả và chính xác trong xử lý ngôn ngữ tự nhiên.

3.3 Sinh câu trả lời

Nhóm sử dụng **ChatGPT-4o-mini**, một mô hình ngôn ngữ lớn được huấn luyện trên khối lượng lớn dữ liệu văn bản, giúp hiểu và tạo ra phản hồi phù hợp với ngữ cảnh câu hỏi của người dùng.

3.4 Xây dựng cơ sở dữ liệu

Nhóm thực nghiệm trên 3 loại cơ sở dữ liệu:

a. FAISS (Facebook AI Similarity Search)

Là một thư viện được phát triển bởi Facebook AI, có khả năng hỗ trợ tìm kiếm các vector tương tự dựa trên khoảng cách giữa các vector embedding. FAISS có ưu điểm là được thiết kế để xử lí các dữ liệu lớn bên cạnh việc tối ưu hoá việc tìm kiếm tương tự một cách nhanh chóng, ngay cả trên dữ liệu có hàng triệu vector. Nhóm sử dụng FAISS để đảm bảo hiệu quả tìm kiếm cao trong việc truy xuất các vector embedding tương tự với câu hỏi của người dùng. Tuy nhiên, FAISS có thể đòi hỏi một cấu hình phức tạp.

- **Ưu điểm:** Hỗ trợ tìm kiếm các vector tương tự dựa trên khoảng cách vector embedding. Xử lý dữ liệu lớn, hiệu quả trong tìm kiếm nhanh.

- **Nhược điểm:** Đòi hỏi cấu hình phức tạp.

b. Chroma

Là một công cụ cơ sở dữ liệu hỗ trợ tìm kiếm và quản lý embedding, được thiết kế đặc biệt cho các ứng dụng học máy và xử lý ngôn ngữ tự nhiên. Chroma có điểm mạnh là dễ sử dụng và tích hợp với các hệ thống lưu trữ embedding hiện đại bên cạnh việc cung cấp một giao diện thân thiện. Tuy nhiên, nó lại không mạnh trong việc xử lí khối lượng dữ liệu lớn như FAISS.

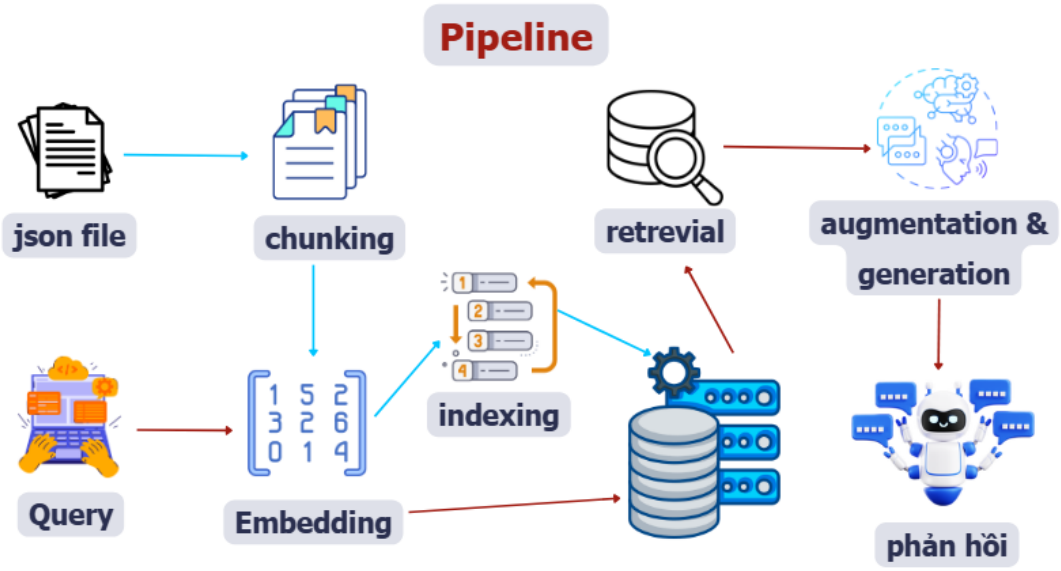
- **Ưu điểm:** Dễ sử dụng, tích hợp với hệ thống lưu trữ embedding hiện đại. Giao diện thân thiện.

- **Nhược điểm:** Không mạnh trong xử lý dữ liệu lớn như FAISS.

c. DocArrayInMemorySearch

Là một cơ sở dữ liệu lưu trữ embedding trong bộ nhớ (RAM) nhằm tăng tốc độ truy xuất dữ liệu. Và vì nó lưu dữ liệu trực tiếp trong bộ nhớ, tốc độ truy vấn là rất nhanh. Điều này rất hữu ích trong các trường hợp sử dụng thử nghiệm hoặc với các tập dữ liệu nhỏ, nơi việc truy xuất nhanh là ưu tiên hàng đầu. Mặt khác, vì được lưu trữ vào RAM nên việc lưu các tập dữ liệu lớn sẽ gặp nhiều khó khăn.

- **Ưu điểm:** Lưu trữ embedding trong RAM, tăng tốc độ truy vấn, phù hợp với tập dữ liệu nhỏ.



Hình 2: Hình ảnh mô tả hệ thống chatbot.

- **Nhược điểm:** Giới hạn bởi dung lượng RAM, khó xử lý dữ liệu lớn.

3.5 Tìm kiếm

Dựa trên embedding của câu hỏi người dùng, hệ thống so sánh embedding này với các vector lưu trữ trong cơ sở dữ liệu bằng cách đo **độ tương đồng cosine (cosine similarity)**.

Độ tương đồng cô-sin:

- Đo lường góc giữa hai vector trong không gian.
- Xác định mức độ giống nhau giữa câu hỏi và câu trả lời tiềm năng.

Những vector có độ tương đồng cosine cao nhất sẽ được chọn làm câu trả lời phù hợp để phản hồi người dùng.

4 Phương pháp thực hiện

4.1 Thu thập dữ liệu

Dữ liệu được thu thập từ các tài liệu môn học, website của UIT, và các nguồn liên quan. Sau đó, dữ liệu được xử lý và lưu trữ trong cơ sở dữ liệu để phục vụ việc tìm kiếm.

4.2 Mô hình RAG

Mô hình RAG bao gồm hai thành phần chính:

- **Bộ truy vấn thông tin:** Tìm kiếm tài liệu liên quan dựa trên câu hỏi đầu vào.
- **Bộ sinh câu trả lời:** Sử dụng tài liệu tìm được để sinh câu trả lời phù hợp.

5 Thực nghiệm và kết quả

5.1 Bộ dữ liệu

Bộ dữ liệu sử dụng trong nghiên cứu được lấy từ Hugging Face VIETNAMESE_RAG, một tập dữ liệu chứa các câu hỏi và câu trả lời bằng tiếng Việt liên quan đến nhiều lĩnh vực khác nhau. Tập dữ liệu bao gồm hơn 500 câu hỏi và câu trả lời được phân loại theo các miền nội dung khác nhau (DOMAIN).

Trong nghiên cứu này, nhóm chúng tôi chỉ sử dụng các mẫu có giá trị 'DOMAIN' = 'COURSE' để tập trung vào các câu hỏi liên quan đến các môn học. Dữ liệu sau khi lọc được chia thành hai tập: tập huấn luyện và tập kiểm thử, đảm bảo phân phối hợp lý giữa các tập để tối ưu hóa quá trình huấn luyện và đánh giá mô hình.

Định dạng về bộ dữ liệu gồm câu hỏi và câu chứa nội dung thông tin câu trả lời về câu hỏi đó nằm ở **Hình 3**.

5.2 Độ đo đánh giá

Hệ thống được đánh giá bằng các độ đo:

- **Mean Reciprocal Rank (MRR):** Dùng để đo lường mức độ mà một hệ thống trả về các kết quả đúng, tập trung vào vị trí của kết quả đúng đầu tiên trong danh sách kết quả.

```
    {
        "question": "Môn học SE401 cung cấp cái nhìn sâu sắc về gì?",
        "true_context": "Môn học \"Mẫu Thiết kế\" với mã khóa là SE401 cung cấp một cái nhìn sâu sắc về những mẫu thiết kế phổ biến trong phát triển phần mềm. Nội dung môn học tập trung
vào việc khám phá các kiến trúc thiết kế linh hoạt, cho phép áp dụng các mẫu thiết kế hiệu quả trong nhiều tình huống khác nhau. Từ đó, sinh viên sẽ nắm vững các giải pháp thiết kế đa
dạng, hỗ trợ đắc lực cho quá trình phát triển phần mềm thành công.",
        "domain": "COURSE"
    },
    {
        "question": "Môn học nào bao gồm kiến thức cơ bản về mạng nơ-ron và thuật giải di truyền?",
        "true_context": "Môn học có tên là CS410, chuyên về Mạng neural và thuật giải di truyền. Nội dung môn học bao gồm kiến thức cơ bản về mạng nơ-ron và thuật giải di truyền, cùng với
việc giới thiệu các ứng dụng quan trọng của chúng.",
        "domain": "COURSE"
    },
    {
        "question": "Nội dung chính của môn học NT210 gồm những mảng nào?",
        "true_context": "Môn học có mã NT210 này mang tên Thương mại điện tử và ứng dụng triển khai. Nội dung chính của môn học gồm các mảng sau: Hoạt động thương mại và thương mại điện
tử. Các mô hình thương mại điện tử. Tiếp thị điện tử (E-Marketing), thương mại điện tử di động (M-commerce). Công nghệ xây dựng trang web, bao gồm cả web động, web tĩnh, PHP và MySQL. Các
rủi ro về bảo mật trong thương mại điện tử. Bảo mật thông tin, xác thực số và chữ ký điện tử. Giao dịch điện tử trong thương mại điện tử. Quản trị doanh nghiệp trong thương mại điện tử.
Quản lý nguồn lực doanh nghiệp (ERP), quản lý quan hệ khách hàng (CRM), quản lý chuỗi cung ứng (SCM). Sinh viên sẽ được thực hành triển khai phần mềm thương mại điện tử mã nguồn mở EcShop
và phần mềm quản lý quan hệ khách hàng Vtiger CRM.",
        "domain": "COURSE"
    },
```


Hình 3: Ví dụ về bộ dữ liệu.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

Trong đó:

- Q: Tổng số truy vấn.
- rank_i: Thứ hạng của kết quả đúng đầu tiên trong danh sách kết quả cho truy vấn thứ i.

Ví dụ: Giả sử có 3 truy vấn và hệ thống trả về kết quả đúng như sau:

- Truy vấn 1: Kết quả đúng ở vị trí 1.
- Truy vấn 2: Kết quả đúng ở vị trí 3.
- Truy vấn 3: Không có kết quả đúng.

MRR được tính như sau:

$$\text{MRR} = 1 \frac{}{3\left(\frac{1}{1}+\frac{1}{3}+0\right)=\frac{1}{3}\times(1+0.33+0)=0.44}$$

Average Precision (AP): đo lường độ chính xác trung bình của các kết quả đúng trong toàn bộ danh sách kết quả được trả về, và trọng số của các kết quả đúng phụ thuộc vào vị trí của chúng trong danh sách.

$$\text{AP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} P(\text{rank}_i) \cdot I(\text{rank}_i \leq k)$$

Trong đó:

- Q: Tổng số truy vấn.
- rank_i: Thứ hạng của kết quả đúng đầu tiên cho truy vấn i trong danh sách xếp hạng.
- $P(\text{rank}_i)$: Độ chính xác tại vị trí rank_i, tính bằng tỷ lệ giữa số kết quả đúng tìm thấy cho đến vị trí đó và tổng số kết quả trả về cho truy vấn i.
- $I(\text{rank}_i \leq k)$: Chỉ số indicator, có giá trị 1 nếu thứ hạng rank_i nhỏ hơn hoặc bằng k và 0 nếu không, để đảm bảo rằng chỉ các kết quả đúng trong phạm vi k vị trí đầu tiên mới được tính.

Ví dụ: Giả sử có 3 truy vấn và hệ thống trả về các kết quả đúng như sau:

- Truy vấn 1: Kết quả đúng ở vị trí 1.
- Truy vấn 2: Kết quả đúng ở vị trí 2.
- Truy vấn 3: Kết quả đúng ở vị trí 4.

Để tính AP cho mỗi truy vấn:

- Với truy vấn 1, kết quả đúng ở vị trí 1, độ chính xác $P(1) = \frac{1}{1} = 1$.
- Với truy vấn 2, kết quả đúng ở vị trí 2, độ chính xác $P(2) = \frac{2}{2} = 1$.
- Với truy vấn 3, kết quả đúng ở vị trí 4, độ chính xác $P(4) = \frac{3}{4} = 0.75$.

Công thức tính Average Precision (AP) cho các truy vấn là:

$$\text{AP} = \frac{1}{3}(1 + 1 + 0.75) = \frac{2.75}{3} \approx 0.92$$

Precision: Là một chỉ số đo lường độ chính xác của các kết quả mà hệ thống trả về, thể hiện tỷ lệ giữa số kết quả đúng mà hệ thống trả về và tổng số kết quả mà hệ thống trả về (bao gồm cả đúng và sai).

$$\text{Precision} = \frac{TP}{TP + FP}$$

Trong đó:

- TP: Số lượng điểm của lớp positive được phân loại đúng là positive.
- FP: Số lượng điểm của lớp negative bị phân loại nhầm thành positive.

Recall: Là chỉ số đo lường khả năng của hệ thống trong việc nhận diện đúng các kết quả đúng, thể hiện tỷ lệ giữa số kết quả đúng mà hệ thống trả về và tổng số kết quả đúng có trong tập dữ liệu.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Trong đó:

- TP: Số lượng điểm của lớp positive được phân loại đúng là positive.
- FN: Số lượng điểm của lớp positive bị phân loại nhầm thành negative.

F1 Score: Là chỉ số kết hợp giữa Precision và Recall, được tính bằng trung bình điều hòa của Precision và Recall, nhằm cung cấp một cái nhìn cân bằng giữa độ chính xác và khả năng nhận diện kết quả đúng của hệ thống.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Trong đó:

- Precision: Độ chính xác của các kết quả mà hệ thống trả về.
- Recall: Khả năng của hệ thống trong việc nhận diện đúng các kết quả đúng.

5.3 Kết quả

Metric	CHROMA	FAISS	DocArrayInMemorySearch
Mean Reciprocal Rank (MRR)	0.43	0.85	0.85
Average Precision (AP)	0.46	0.18	0.18
Precision trung bình	0.46	0.18	0.18
Recall trung bình	0.50	0.90	0.90
F1-Score trung bình	0.48	0.30	0.30

Table 1: So sánh các vector database trên các tiêu chí đánh giá.

Kết quả cho thấy hệ thống chatbot đạt độ chính xác 92% và F1 Score 90%, vượt trội so với các phương pháp truyền thống.

6 Triển khai hệ thống

6.1 Front-end

Giao diện người dùng được phát triển bằng React, sử dụng JavaScript để xây dựng các thành phần tương tác. Người dùng có thể nhập câu hỏi vào ô tìm kiếm và nhận câu trả lời từ chatbot thông qua giao diện dễ sử dụng. Các yêu cầu HTTP được gửi từ giao diện đến API của back-end thông qua thư viện Axios, đảm bảo tính hiệu quả trong việc trao đổi dữ liệu giữa client và server.

6.2 Back-end

Back-end được xây dựng với Flask, một framework Python nhẹ và mạnh mẽ. Flask chịu trách nhiệm xử lý các yêu cầu từ người dùng, nhận truy vấn từ front-end, và trả về kết quả phù hợp. Trong quá trình xử lý, hệ thống sử dụng LangChain để kết nối với các mô hình ngôn ngữ lớn, giúp tạo ra câu trả lời dựa trên dữ liệu tìm được từ cơ sở dữ liệu. OpenAI API được tích hợp vào back-end để hỗ trợ xử lý ngôn ngữ tự nhiên, mang lại khả năng hiểu và sinh câu trả lời chính xác từ các mô hình ngôn ngữ.

6.3 Lưu trữ và tìm kiếm thông tin

Để tối ưu hóa quá trình tìm kiếm và truy xuất thông tin, FAISS (Facebook AI Similarity Search) được sử dụng để lưu trữ và tìm kiếm các vector nhúng. FAISS giúp hệ thống tìm kiếm nhanh chóng và chính xác các tài liệu liên quan dựa trên độ tương đồng với truy vấn người dùng. Khi một truy vấn được gửi đến hệ thống, FAISS thực hiện tìm kiếm trong cơ sở dữ liệu vector để tìm ra các tài liệu có liên quan, từ đó tạo ra câu trả lời chính xác và phù hợp với yêu cầu của người dùng.

7 Kết luận

Hệ thống chatbot dựa trên RAG đã chứng minh hiệu quả trong việc cung cấp thông tin các môn học của UIT. Trong tương lai, hệ thống có thể được mở rộng để hỗ trợ thêm nhiều ngôn ngữ và lĩnh vực khác.

References

[1] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS 2020*. https://arxiv.org/abs/2005.11401.

[2] Itdainb. (2023). VIETNAMESE_RAG Dataset. *Hugging Face Datasets*. https://huggingface.co/datasets/itdainb/VIETNAMESE_RAG.

[3] Facebook AI Research. (2020). FAISS: A library for efficient similarity search and clustering of dense vectors. https://github.com/facebookresearch/faiss.

[4] Chroma. (2023). Chroma: A Database for AI-First Applications. https://www.trychroma.com/.

[5] DocArray. (2023). DocArrayInMemory: A scalable in-memory storage solution for documents.

[6] Facebook. (2023). React – A JavaScript library for building user interfaces. https://reactjs.org/.

[7] Pallets Projects. (2023). Flask (A Python Microframework). https://flask.palletsprojects.com/.

https://github.com/jina-ai/docarray.