SỬ DỤNG OPENAI API VÀ FILE SEARCH ĐỂ XÂY DỰNG HỆ THỐNG HỎI – ĐÁP THÔNG MINH THEO NỘI DUNG TÀI LIỆU: NGHIÊN CỨU VÀ THỰC NGHIỆM TRONG BÀI TOÁN TUYỂN SINH ĐẠI HỌC

Lê Xuân Hùng
Trường Đại học Đồng Nai
Email: hunglx.it@gmail.com

(Ngày nhận bài: 8/1/2026, ngày nhận bài chỉnh sửa: 7/3/2026, ngày duyệt đăng: 4/6/2026)

TÓM TẮT

Sự gia tăng của các nguồn tài liệu số trong giáo dục đại học đặt ra yêu cầu về các giải pháp hỗ trợ khai thác và cung cấp thông tin một cách chính xác, nhất quán và dễ tiếp cận. Đối với các bài toán hỏi – đáp trên dữ liệu chuyên biệt như tuyển sinh đại học, việc sử dụng trực tiếp mô hình ngôn ngữ lớn có thể dẫn đến nguy cơ sinh thông tin không phù hợp với dữ liệu nguồn. Nghiên cứu này đề xuất phương pháp xây dựng hệ thống hỏi – đáp dựa trên tài liệu bằng cách kết hợp OpenAI API với công cụ File Search nhằm hỗ trợ truy xuất nội dung theo ngữ nghĩa. Trên cơ sở kiến trúc Retrieval-Augmented Generation (RAG), nghiên cứu xây dựng quy trình gồm chuẩn hóa dữ liệu đầu vào, tổ chức tài liệu trong Vector Store, thiết kế cơ chế kiểm soát quá trình sinh nội dung và triển khai hệ thống trên nền tảng ASP.NET Core. Thực nghiệm được tiến hành trên bộ tài liệu tuyển sinh đại học năm 2025 cùng tập câu hỏi mô phỏng các tình huống thực tế. Kết quả cho thấy hệ thống cung cấp câu trả lời bám sát tài liệu nguồn, hạn chế hiện tượng suy diễn ngoài dữ liệu và duy trì hiệu quả truy xuất ổn định trong quá trình vận hành.

Từ khóa*: File Search; hệ thống hỏi – đáp dựa trên tài liệu; OpenAI API; Retrieval-Augmented Generation (RAG); Truy xuất ngữ nghĩa (Semantic Retrieval); Vector Store*

1. Đặt vấn đề

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ, các cơ sở giáo dục đại học ngày càng tích lũy khối lượng lớn tài liệu số phục vụ các hoạt động đào tạo, tuyển sinh, quản lí và nghiên cứu khoa học. Riêng đối với công tác tuyển sinh đại học, hệ thống văn bản như đề án tuyển sinh, thông báo, quy chế, hướng dẫn và các văn bản điều chỉnh hằng năm thường có dung lượng lớn, nội dung phức tạp và được cập nhật liên tục. Điều này đặt ra yêu cầu cấp thiết về các công cụ hỗ trợ khai thác thông tin hiệu quả, chính xác và nhất quán cho thí sinh, phụ huynh và cán bộ tư vấn.

Trong thực tiễn triển khai, việc tra cứu thông tin tuyển sinh tại nhiều cơ sở giáo dục vẫn chủ yếu dựa trên tìm kiếm từ khóa hoặc đọc thủ công tài liệu. Các phương pháp này bộc lộ hạn chế rõ rệt khi người dùng không nắm rõ cấu trúc văn bản hoặc sử dụng cách diễn đạt khác với nội dung trong tài liệu. Kết quả truy xuất thường rời rạc, thiếu bao quát và đòi hỏi nhiều công sức tổng hợp, đặc biệt trong bối cảnh nhu cầu hỏi – đáp trực tuyến ngày càng gia tăng.

Sự phát triển của các mô hình ngôn ngữ lớn đã mở ra khả năng tương tác với tài liệu thông qua ngôn ngữ tự nhiên [1]. Tuy nhiên, việc sử dụng trực tiếp mô hình ngôn ngữ cho bài toán tra cứu thông tin nội bộ tiềm ẩn nguy cơ sinh ra câu trả lời không chính xác hoặc vượt ra ngoài phạm vi dữ liệu được cung cấp. Đối với các lĩnh vực đòi hỏi độ tin cậy cao như tuyển sinh đại học, những hạn chế này

làm giảm khả năng ứng dụng thực tế của các mô hình ngôn ngữ nếu không có cơ chế kiểm soát phù hợp.

Để khắc phục các hạn chế nêu trên, mô hình kết hợp truy xuất và sinh nội dung (Retrieval-Augmented Generation – RAG) đã được đề xuất, trong đó câu trả lời của mô hình ngôn ngữ được tăng cường bởi các đoạn thông tin được truy xuất từ kho dữ liệu bên ngoài [2]. Cách tiếp cận này cho phép tận dụng khả năng hiểu ngữ nghĩa của mô hình ngôn ngữ lớn, đồng thời đảm bảo nội dung trả lời bám sát tài liệu nguồn và giảm thiểu hiện tượng suy diễn ngoài dữ liệu.

Trong hệ sinh thái OpenAI API, công cụ File Search cung cấp cơ chế quản lí và truy xuất tài liệu dựa trên biểu diễn ngữ nghĩa thông qua Vector Store, cho phép tìm kiếm nội dung theo ngữ cảnh thay vì chỉ dựa trên từ khóa [3], [4]. Việc tích hợp File Search với mô hình ngôn ngữ tạo điều kiện thuận lợi cho việc xây dựng các hệ thống hỏi – đáp dựa trên tài liệu, đồng thời giảm đáng kể khối lượng công việc triển khai ở tầng kĩ thuật. Tuy nhiên, các nghiên cứu và báo cáo ứng dụng File Search trong bối cảnh dữ liệu tiếng Việt, đặc biệt trong lĩnh vực giáo dục và tuyển sinh đại học, hiện vẫn còn hạn chế và thiếu các phân tích mang tính hệ thống.

Xuất phát từ thực tiễn và khoảng trống nghiên cứu đó, bài báo này đề xuất và trình bày một cách tiếp cận xây dựng hệ thống hỏi – đáp dựa trên OpenAI API với công cụ File Search, được triển khai trên nền tảng ASP.NET Core và thực nghiệm trong bài toán tuyển sinh đại học. Thay vì tập trung vào việc phát triển một chatbot tư vấn theo nghĩa chung, nghiên cứu hướng tới làm rõ các vấn đề cốt lõi gồm: (i) cơ sở lí thuyết của truy xuất ngữ nghĩa trong mô hình RAG; (ii) quy trình chuẩn hóa và tổ chức tài liệu đầu vào phù hợp với cơ chế File Search; (iii) thiết kế hệ thống hỏi – đáp có kiểm soát hành vi sinh nội dung; và (iv) đánh giá hiệu quả hệ thống thông qua thực nghiệm trên dữ liệu thực tế.

Đóng góp của bài báo không chỉ nằm ở việc minh họa cách sử dụng OpenAI API mà còn ở việc đề xuất một quy trình triển khai có tính hệ thống, khả năng tái sử dụng và phù hợp với điều kiện ứng dụng tại các cơ sở giáo dục đại học. Cách tiếp cận này góp phần nâng cao hiệu quả cung cấp thông tin tuyển sinh, đồng thời tạo nền tảng cho các nghiên cứu tiếp theo về tối ưu hóa thành phần truy xuất trong các hệ thống hỏi – đáp dựa trên tài liệu.

2. Tổng quan nghiên cứu

Hệ thống hỏi – đáp dựa trên tài liệu (Document-based Question Answering) là một hướng nghiên cứu quan trọng trong lĩnh vực truy xuất thông tin và trí tuệ nhân tạo, đặc biệt trong các bối cảnh yêu cầu khai thác tri thức từ nguồn dữ liệu có cấu trúc hoặc bán cấu trúc. Các nghiên cứu ban đầu chủ yếu dựa trên các phương pháp tìm kiếm từ khóa truyền thống như TF-IDF hoặc BM25 [5], trong đó câu hỏi của người dùng được ánh xạ thành tập từ khóa và so khớp với văn bản nguồn. Mặc dù có ưu điểm về tính đơn giản và chi phí tính toán thấp, các phương pháp này thường cho kết quả hạn chế khi câu hỏi mang tính ngữ nghĩa, diễn đạt tự nhiên hoặc không trùng khớp trực tiếp với từ ngữ trong tài liệu.

Sự phát triển của các kĩ thuật biểu diễn ngữ nghĩa dựa trên vector (embedding) đã thúc đẩy sự chuyển dịch từ tìm kiếm dựa trên từ khóa sang tìm kiếm ngữ nghĩa (semantic search) [3], [6]. Trong hướng tiếp cận này, cả câu hỏi và tài liệu được ánh xạ vào cùng một không gian vector, cho phép đo lường mức độ tương đồng về mặt ngữ nghĩa. Nhiều nghiên cứu đã chỉ ra rằng truy xuất ngữ nghĩa giúp cải thiện đáng kể độ chính xác của các hệ thống hỏi – đáp, đặc biệt trong các trường hợp câu hỏi phức

tạp, yêu cầu tổng hợp thông tin hoặc sử dụng cách diễn đạt linh hoạt.

Trên nền tảng truy xuất ngữ nghĩa, mô hình Retrieval-Augmented Generation (RAG) đã được đề xuất nhằm kết hợp khả năng truy xuất thông tin với sức mạnh sinh ngôn ngữ của các mô hình ngôn ngữ lớn [2]. Trong mô hình này, các đoạn văn bản liên quan được truy xuất từ kho dữ liệu bên ngoài và cung cấp làm ngữ cảnh cho mô hình ngôn ngữ trong quá trình sinh câu trả lời. Cách tiếp cận RAG cho phép hệ thống duy trì tính linh hoạt trong diễn đạt, đồng thời giảm thiểu hiện tượng sinh thông tin không có căn cứ và tăng khả năng kiểm soát nội dung trả lời. Do đó, RAG được xem là hướng tiếp cận phù hợp cho các bài toán hỏi – đáp trên dữ liệu nội bộ, nơi yêu cầu cao về độ chính xác và tính nhất quán.

Song song với các nghiên cứu ở mức mô hình, nhiều nền tảng đã phát triển các công cụ truy xuất tích hợp nhằm đơn giản hóa việc triển khai RAG trong thực tế. Trong hệ sinh thái OpenAI API, công cụ File Search cung cấp cơ chế quản lí tài liệu và truy xuất ngữ nghĩa thông qua Vector Store, cho phép người phát triển khai thác trực tiếp các khả năng truy xuất theo ngữ cảnh mà không cần tự xây dựng toàn bộ chu trình embedding, lập chỉ mục và tìm kiếm [4]. File Search có thể được xem như một hình thức hiện thực hóa mô hình RAG ở mức hệ thống, trong đó một số chiến lược truy xuất và phân đoạn dữ liệu được đóng gói sẵn.

Mặc dù RAG đã được nghiên cứu và ứng dụng trong nhiều lĩnh vực khác nhau, phần lớn các công trình hiện nay tập trung vào tối ưu thành phần retriever hoặc generator, hoặc so sánh hiệu năng giữa các thuật toán truy xuất khác nhau. Các nghiên cứu khai thác các công cụ truy xuất tích hợp sẵn như File Search, đặc biệt trong bối cảnh dữ liệu tiếng Việt và các ứng dụng giáo dục, vẫn còn tương đối hạn chế. Bên cạnh đó, các báo cáo ứng dụng thường thiếu mô tả chi tiết về quy trình chuẩn hóa dữ liệu và kiểm soát hành vi sinh câu trả lời, vốn là những yếu tố có ảnh hưởng trực tiếp đến độ tin cậy của hệ thống hỏi – đáp.

Để làm rõ hơn sự khác biệt giữa các hướng tiếp cận trong các hệ thống hỏi – đáp dựa trên RAG, bảng 1 trình bày so sánh ngắn giữa giải pháp sử dụng File Search và một số hướng triển khai phổ biến.

Bảng 1: *So sánh cách tiếp cận File Search với các hệ thống RAG phổ biến*

Hướng tiếp cận	**Công cụ truy xuất**	**Độ phức tạp triển khai**	**Kiểm soát pipeline**	**Đặc điểm chính**
RAG + FAISS	FAISS vector database	Cao	Cao	Cho phép kiểm soát hoàn toàn embedding, chunking và retrieval; phù hợp nghiên cứu thuật toán
RAG + Elasticsearch	Elasticsearch + hybrid search	Trung bình	Trung bình	Hỗ trợ tìm kiếm kết hợp keyword và vector, phù hợp hệ thống dữ liệu lớn
RAG + Chroma / LlamaIndex	Vector database mã nguồn mở	Trung bình	Cao	Dễ tích hợp với các framework RAG, phù hợp phát triển thử nghiệm
RAG + File Search (đề xuất)	Vector Store tích hợp trong OpenAI API	Thấp	Hạn chế	Tích hợp sẵn embedding, indexing và semantic retrieval, giúp giảm đáng kể độ phức tạp triển khai

Từ sự so sánh này có thể thấy rằng các hệ thống RAG tự xây dựng cho phép kiểm soát chi tiết pipeline truy xuất nhưng yêu cầu triển khai phức tạp hơn. Ngược lại, File Search cung cấp một cơ chế truy xuất tích hợp sẵn giúp đơn giản hóa việc xây dựng hệ thống hỏi – đáp.

Mặc dù việc sử dụng các mô hình ngôn ngữ lớn và kiến trúc Retrieval-Augmented Generation đã được nghiên cứu rộng rãi, nhiều báo cáo ứng dụng hiện nay chủ yếu tập trung vào mô tả công cụ hoặc xây dựng chatbot ở mức triển khai hệ thống. Nghiên cứu này không hướng tới việc đề xuất một thuật toán truy xuất mới mà tập trung làm rõ quy trình và kiến trúc triển khai hệ thống hỏi – đáp dựa trên tài liệu trong bối cảnh dữ liệu tiếng Việt và môi trường ứng dụng giáo dục.

Cụ thể, bài báo có ba đóng góp chính. Thứ nhất, nghiên cứu đề xuất một quy trình chuẩn hóa và tổ chức dữ liệu tài liệu phù hợp với cơ chế truy xuất ngữ nghĩa của File Search, bao gồm việc tái cấu trúc tài liệu theo chủ đề, nhúng metadata trực tiếp vào nội dung và đảm bảo khả năng truy vết nguồn thông tin trong quá trình hỏi – đáp. Thứ hai, bài báo trình bày chiến lược kiểm soát hành vi sinh nội dung của mô hình ngôn ngữ thông qua system prompt, nhằm hạn chế hiện tượng suy diễn ngoài dữ liệu và đảm bảo câu trả lời bám sát tài liệu gốc. Thứ ba, nghiên cứu xây dựng và thực nghiệm một kiến trúc hệ thống hỏi – đáp hoàn chỉnh kết hợp OpenAI API, File Search và nền tảng ASP.NET Core trong bài toán tư vấn tuyển sinh đại học.

Những đóng góp này giúp làm rõ cách thức triển khai hiệu quả các hệ thống hỏi – đáp dựa trên tài liệu trong các môi trường dữ liệu thực tế, đồng thời cung cấp một khung tham chiếu có thể áp dụng cho các hệ thống thông tin giáo dục tương tự.

3. Cơ sở lí thuyết và phương pháp nghiên cứu

3.1. Mô hình Retrieval-Augmented Generation trong hệ thống hỏi – đáp

Mô hình Retrieval-Augmented Generation (RAG) là cách tiếp cận kết hợp giữa truy xuất thông tin và sinh ngôn ngữ tự nhiên nhằm giải quyết bài toán hỏi – đáp trên tập dữ liệu ngoài phạm vi huấn luyện của mô hình ngôn ngữ [2]. Thay vì để mô hình ngôn ngữ sinh câu trả lời hoàn toàn dựa trên tri thức nội tại, RAG bổ sung một bước truy xuất các đoạn tài liệu liên quan từ kho dữ liệu bên ngoài, sau đó sử dụng các đoạn này làm ngữ cảnh đầu vào cho quá trình sinh câu trả lời. Cách tiếp cận này giúp cải thiện tính chính xác của hệ thống, giảm hiện tượng tạo thông tin không có căn cứ, đồng thời tăng khả năng kiểm soát nội dung sinh ra.

Về mặt lí thuyết, RAG bao gồm hai thành phần chính: (i) module truy xuất (retriever), chịu trách nhiệm xác định các đoạn văn bản có độ tương đồng ngữ nghĩa cao với câu hỏi; và (ii) module sinh (generator) [2], [7], là mô hình ngôn ngữ lớn thực hiện nhiệm vụ tổng hợp và sinh câu trả lời dựa trên ngữ cảnh được cung cấp. Hiệu quả của mô hình phụ thuộc đáng kể vào chất lượng biểu diễn ngữ nghĩa của dữ liệu, chiến lược truy xuất và cách tích hợp ngữ cảnh vào mô hình sinh.

3.2. File Search và Vector Store trong OpenAI API

File Search trong OpenAI API được thiết kế như một lớp truy xuất ngữ nghĩa tích hợp, hỗ trợ quản lí, tổ chức và tìm kiếm nội dung tài liệu dựa trên biểu diễn vector [4]. Về cấu trúc, File Search bao gồm ba thành phần chính: (i) tập tin dữ liệu gốc (files), (ii) cơ chế biểu diễn ngữ nghĩa thông qua embedding và (iii) Vector Store dùng để lưu trữ, lập chỉ mục và truy vấn các vector tương ứng với nội dung tài liệu.

Khi tài liệu được tải lên hệ thống, nội dung được phân tách thành các đơn vị nhỏ (chunk) theo chiến lược nội bộ nhằm đảm bảo mỗi đoạn mang ý nghĩa ngữ nghĩa tương đối độc lập. Mỗi chunk sau đó được ánh xạ thành vector embedding trong không gian đa chiều. Các vector này được lưu trữ trong Vector Store, cho phép thực hiện các phép đo độ tương đồng ngữ nghĩa giữa câu hỏi và nội dung tài liệu.

Nguyên lí tìm kiếm của File Search dựa trên việc ánh xạ câu hỏi của người dùng thành một vector truy vấn, sau đó so sánh với các vector trong Vector Store để xác định tập các đoạn văn bản liên quan nhất (top-k) [4]. Các đoạn truy xuất này được lựa chọn làm ngữ cảnh đầu vào cho mô hình ngôn ngữ lớn trong quá trình sinh câu trả lời. Cách tiếp cận này cho phép hệ thống truy xuất thông tin theo ngữ cảnh, thay vì phụ thuộc vào sự trùng khớp từ khóa, qua đó nâng cao độ chính xác và khả năng bao phủ nội dung.

3.3. Phương pháp nghiên cứu và xây dựng hệ thống hỏi – đáp

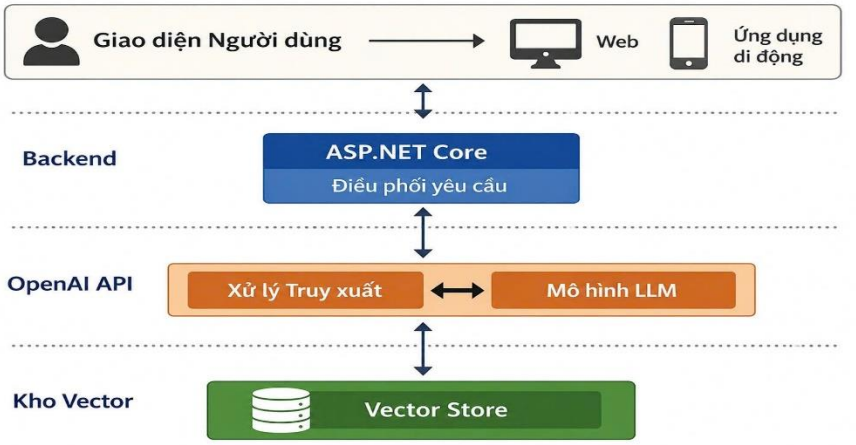
Nghiên cứu được thực hiện theo phương pháp nghiên cứu ứng dụng kết hợp với thực nghiệm hệ thống. Trên cơ sở mô hình RAG và cơ chế File Search, bài báo đề xuất quy trình xây dựng hệ thống hỏi – đáp theo tài liệu gồm các bước chính: (i) thu thập và chuẩn hóa dữ liệu tài liệu; (ii) tổ chức và quản lí dữ liệu trong Vector Store; (iii) xây dựng cơ chế truy vấn hỏi – đáp; và (iv) đánh giá hiệu quả hệ thống thông qua thực nghiệm.

Trong bước xây dựng hệ thống, tập tài liệu đầu vào được chuẩn hóa và tổ chức trước khi tải lên Vector Store. Quy trình chuẩn hóa dữ liệu được trình bày chi tiết ở mục 4.

3.4. Vai trò của nền tảng ASP.NET Core

ASP.NET Core được sử dụng làm nền tảng triển khai hệ thống hỏi – đáp trong nghiên cứu này. Tầng backend chịu trách nhiệm tiếp nhận và xử lí yêu cầu từ giao diện người dùng, quản lí luồng nghiệp vụ, giao tiếp với OpenAI API và trả kết quả cho người dùng. Việc sử dụng ASP.NET Core cho phép xây dựng hệ thống theo kiến trúc dịch vụ, hỗ trợ mở rộng, bảo trì và tích hợp với các hệ thống thông tin hiện có của cơ sở giáo dục.

3.5. Kiến trúc tổng thể hệ thống hỏi – đáp



Hình 1: *Kiến trúc tổng thể hệ thống hỏi – đáp*

Hệ thống hỏi – đáp được thiết kế theo kiến trúc nhiều lớp nhằm đảm bảo tính tách biệt chức năng, khả năng mở rộng và tính linh hoạt trong triển khai. Về tổng thể, kiến trúc hệ thống bao gồm bốn thành phần chính: (i) giao diện người dùng (frontend); (ii) tầng xử lí ứng dụng (backend) triển khai trên ASP.NET

Core; (iii) dịch vụ OpenAI API với công cụ File Search; và (iv) kho dữ liệu tài liệu dưới dạng Vector Store.

Giao diện người dùng đóng vai trò tiếp nhận câu hỏi và hiển thị câu trả lời. Tầng backend thực hiện điều phối luồng xử lí, bao gồm gửi yêu cầu truy vấn đến OpenAI API, tiếp nhận kết quả truy xuất và sinh nội dung, đồng thời kiểm soát logic ứng dụng. OpenAI API với File Search đảm nhiệm vai trò truy xuất ngữ nghĩa và cung cấp ngữ cảnh cho mô hình ngôn ngữ, trong khi Vector Store lưu trữ và quản lí các biểu diễn vector của tài liệu. Kiến trúc này phản ánh rõ mô hình RAG, trong đó quá trình truy xuất và sinh nội dung được tích hợp chặt chẽ nhưng vẫn giữ được tính module.

Tóm lại, phần cơ sở lí thuyết và phương pháp nghiên cứu đã trình bày nền tảng RAG, cơ chế File Search và Vector Store, cùng với quy trình xây dựng và kiến trúc tổng thể của hệ thống hỏi – đáp. Cách tiếp cận này tạo cơ sở cho việc đánh giá thực nghiệm hệ thống trong bối cảnh hỏi – đáp về tuyển sinh đại học, được trình bày trong các phần tiếp theo của bài báo.

4. Chuẩn hóa nội dung trước khi upload

4.1. Vai trò của chuẩn hóa dữ liệu trong hệ thống hỏi – đáp

Trong các hệ thống hỏi – đáp dựa trên Retrieval-Augmented Generation (RAG), chất lượng dữ liệu đầu vào đóng vai trò quyết định đến độ chính xác và tính nhất quán của câu trả lời [2]. Khác với các hệ thống tìm kiếm truyền thống, mô hình truy xuất ngữ nghĩa dựa trên embedding đặc biệt nhạy cảm với lỗi định dạng, nhiễu nội dung và sự không đồng nhất trong cách trình bày văn bản. Do đó, chuẩn hóa nội dung tài liệu trước khi tải lên Vector Store là bước tiền xử lí bắt buộc nhằm đảm bảo hệ thống chỉ truy xuất và sinh câu trả lời dựa trên thông tin có căn cứ.

4.2. Thu thập và phân loại nguồn tài liệu

Trong nghiên cứu này, tập tài liệu đầu vào được thu thập từ các nguồn chính thức liên quan đến tuyển sinh đại học, bao gồm đề án tuyển sinh, thông báo tuyển sinh, quy chế đào tạo và các văn bản hướng dẫn của cơ sở đào tạo. Các tài liệu gốc chủ yếu tồn tại ở dạng PDF scan có đóng dấu, chữ kí hoặc các tệp Word/PDF hành chính.

Để phục vụ mục tiêu hỏi – đáp, các tài liệu được phân loại theo nhóm chủ đề như thông tin chung, đối tượng và điều kiện tuyển sinh, phương thức xét tuyển, ngành đào tạo, học phí và quy đổi chứng chỉ. Việc phân loại này giúp tổ chức dữ liệu rõ ràng, đồng thời hỗ trợ quá trình truy xuất ngữ nghĩa trong các bước tiếp theo.

4.3. Chuẩn hóa định dạng và nội dung văn bản

Các tài liệu gốc ở dạng scan được xử lí thông qua công nghệ nhận dạng kí tự quang học (OCR) nhằm chuyển đổi sang dạng văn bản có thể xử lí tự động. Kết quả OCR được rà soát và chỉnh sửa để loại bỏ các lỗi phổ biến như kí tự sai, dấu tiếng Việt không chính xác, khoảng trắng dư thừa hoặc các thành phần trình bày không mang giá trị ngữ nghĩa.

Đối với các tài liệu ở dạng Word hoặc PDF văn bản, nội dung được chuẩn hóa về mã Unicode, loại bỏ header, footer, số trang và các yếu tố định dạng thuần túy. Văn bản sau chuẩn hóa được trình bày theo cấu trúc đơn giản, thống nhất thuật ngữ và cách diễn đạt, nhằm giảm thiểu sự sai lệch ngữ nghĩa trong quá trình tạo embedding.

4.4. Nhúng metadata vào nội dung tài liệu

Do OpenAI API hiện chưa hỗ trợ metadata ở mức truy vấn và lọc độc lập trong Vector Store, nghiên cứu này sử dụng cách nhúng metadata trực tiếp vào nội dung của mỗi tài liệu đã được chuẩn

hóa. Metadata được đặt ở phần đầu tài liệu dưới dạng một khối cấu trúc chuẩn, bao gồm các trường như tên văn bản, đơn vị ban hành, năm tuyển sinh, chủ đề và liên kết đến tài liệu gốc.

Khối metadata này được xem như một phần của nội dung văn bản và được đưa vào quá trình embedding cùng với phần nội dung chính. Cách tiếp cận này đảm bảo rằng thông tin nguồn luôn đi kèm với các đoạn văn bản được truy xuất, tạo tiền đề cho việc trích dẫn trong quá trình sinh câu trả lời.

4.5. Chia nhỏ nội dung theo chủ đề và ngữ nghĩa

Sau khi chuẩn hóa định dạng và nhúng metadata, nội dung tài liệu được chia nhỏ thành các đoạn văn bản theo chủ đề và ngữ nghĩa. Thay vì chia đoạn một cách cơ học theo số lượng kí tự, nghiên cứu ưu tiên phân tách nội dung dựa trên cấu trúc logic của tài liệu như mục, tiểu mục hoặc các đoạn thông tin hoàn chỉnh.

Mỗi đoạn văn bản được thiết kế để mang một ý nghĩa tương đối độc lập, phù hợp với cơ chế truy xuất ngữ nghĩa của Vector Store. Việc chia nhỏ hợp lí giúp giảm nguy cơ truy xuất dư thừa ngữ cảnh và nâng cao độ chính xác của câu trả lời.

4.6. Cơ chế trích dẫn nguồn trong quá trình hỏi – đáp

Trong hệ thống được đề xuất, việc trích dẫn nguồn không được xử lí ở tầng ứng dụng mà được thực hiện trực tiếp trong quá trình sinh câu trả lời của mô hình ngôn ngữ, dưới sự kiểm soát chặt chẽ của system prompt. Mô hình được yêu cầu chỉ sử dụng thông tin có trong tài liệu truy xuất, không suy đoán và không sử dụng tri thức bên ngoài. Đồng thời, nếu trong nội dung truy xuất có trường liên kết đến tài liệu gốc (link_file_goc), mô hình phải chèn liên kết này vào cuối câu trả lời theo cú pháp Markdown chuẩn.

Cách tiếp cận này cho phép đơn giản hóa kiến trúc hệ thống, giảm xử lí hậu kì ở tầng backend, đồng thời vẫn đảm bảo tính minh bạch và khả năng kiểm chứng của thông tin cung cấp cho người dùng. Việc kiểm soát hành vi sinh câu trả lời thông qua system prompt giúp hạn chế nguy cơ mô hình tự sinh nguồn không có căn cứ, phù hợp với yêu cầu ứng dụng trong bối cảnh cung cấp thông tin tuyển sinh đại học.

4.7. Tóm tắt bước chuẩn hóa nội dung

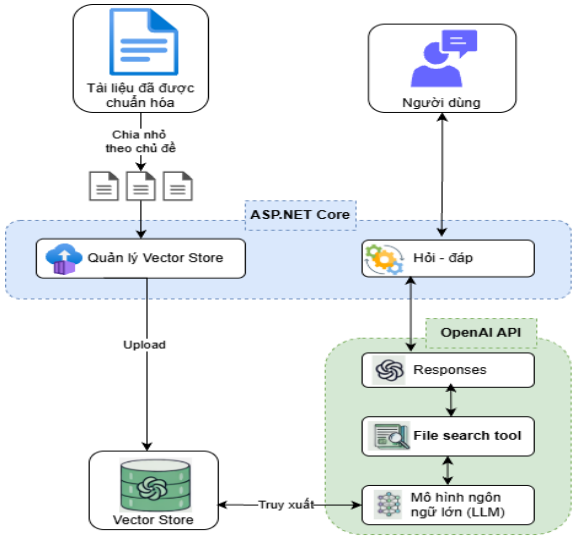
Tóm lại, quy trình chuẩn hóa nội dung trong nghiên cứu này bao gồm thu thập và phân loại tài liệu, chuẩn hóa định dạng và nội dung văn bản, nhúng metadata trực tiếp vào tài liệu và chia nhỏ nội dung theo chủ đề và ngữ nghĩa. Cách tiếp cận này được thiết kế đồng bộ với cơ chế hỏi – đáp của hệ thống, tạo nền tảng dữ liệu nhất quán và đáng tin cậy cho các bước triển khai và thực nghiệm ở các phần tiếp theo của bài báo.

5. Quy trình xây dựng hệ thống hỏi – đáp

5.1. Tổng quan quy trình

Quy trình xây dựng hệ thống hỏi – đáp trong nghiên cứu này được thiết kế theo hướng đơn giản hóa kiến trúc nhưng vẫn đảm bảo tính kiểm soát và độ tin cậy của kết quả. Hệ thống được xây dựng trên nền tảng ASP.NET Core, kết hợp OpenAI API với công cụ File Search và Vector Store [3], [4], [8]. Toàn bộ quy trình có thể được chia thành bốn giai đoạn chính: (i) chuẩn bị và tải dữ liệu; (ii) tổ chức dữ liệu trong Vector Store; (iii) xử lí yêu cầu hỏi – đáp; và (iv) sinh câu trả lời kèm trích dẫn nguồn.

Hệ thống hỏi – đáp được triển khai dưới dạng một ứng dụng web chatbot và đã được đưa vào vận hành thử nghiệm tại địa chỉ ts.dnpu.edu.vn. Ứng dụng cho phép người dùng tương tác trực tiếp thông qua trình duyệt để đặt câu hỏi và nhận câu trả lời dựa trên các tài liệu tuyển sinh đã được chuẩn hóa và tải lên Vector Store.



Hình 2: *Sơ đồ luồng quy trình hệ thống hỏi – đáp sử dụng OpenAI File Search và Vector Store*

5.2. Chuẩn bị và tải dữ liệu vào Vector Store

Sau khi nội dung tài liệu được chuẩn hóa theo quy trình đã trình bày ở mục 4, các tệp văn bản được tải lên hệ thống thông qua chức năng upload của ứng dụng ASP.NET Core. Mỗi tài liệu đã được nhúng metadata ở phần đầu nội dung, bao gồm thông tin mô tả và liên kết đến tài liệu gốc. Trong bước này, hệ thống chỉ thực hiện việc tải tệp lên OpenAI API và gắn tệp vào Vector Store tương ứng, không xử lí thêm về mặt ngữ nghĩa.

Việc tách riêng bước chuẩn hóa nội dung và bước tải dữ liệu giúp đảm bảo rằng Vector Store chỉ chứa các tài liệu đã được kiểm soát chất lượng, đồng thời tạo điều kiện thuận lợi cho việc cập nhật hoặc thay thế tài liệu trong các kì tuyển sinh tiếp theo.

Đoạn mã dưới đây minh họa thao tác cốt lõi trong quá trình upload và liên kết tài liệu với Vector Store [8].

```
var fileClient = _client.GetOpenAIFileClient();
using var stream = file.OpenReadStream();
var uploadResult = await fileClient.UploadFileAsync(
    stream,file.FileName,
FileUploadPurpose.Assistants
    );
    await
_client.GetVectorStoreClient().AddFile
ToVectorStoreAsync(
    VectorStoreId,
uploadResult.Value.Id);
```

5.3. Tiếp nhận câu hỏi từ người dùng

Người dùng tương tác với hệ thống thông qua giao diện web, nơi câu hỏi được nhập bằng ngôn ngữ tự nhiên. Câu hỏi sau đó được gửi đến controller của ứng dụng ASP.NET Core và chuyển tiếp đến tầng xử lí nghiệp vụ. Tại đây, hệ thống không thực hiện tiền xử lí phức tạp đối với câu hỏi, nhằm giữ nguyên ý nghĩa ngữ cảnh ban đầu của người dùng.

5.4. Xây dựng yêu cầu hỏi – đáp với OpenAI API

Tầng xử lí nghiệp vụ chịu trách nhiệm xây dựng yêu cầu hỏi – đáp gửi đến OpenAI API. Yêu cầu này bao gồm ba thành phần chính: (i) system prompt dùng để kiểm soát hành vi sinh câu trả lời; (ii) câu hỏi của người dùng; và (iii) công cụ File Search gắn với Vector Store chứa dữ liệu tuyển sinh.

Trong quá trình xây dựng yêu cầu hỏi – đáp, hệ thống sử dụng system prompt để kiểm soát hành vi sinh câu trả lời của mô hình ngôn ngữ. Prompt được thiết kế nhằm ràng buộc mô hình chỉ sử dụng thông tin có trong tài liệu đã được truy xuất, không suy đoán và không sử dụng tri thức bên ngoài. Đồng thời, prompt quy định rõ định dạng trả lời và cơ chế chèn liên kết đến tài liệu gốc nếu trường link_file_goc tồn tại trong nội dung truy xuất. Cách thiết kế này cho phép kiểm soát hành vi của mô hình ngay tại thời điểm sinh câu trả lời.

Đoạn mã dưới đây minh họa thao tác hỏi đáp với OpenAI API [8].

```
options.InputItems.Add(ResponseItem.CreateUserMessageItem(question));
options.Tools.Add(ResponseTool.CreateFileSearchTool(new[] { vectorStoreId }));
var response = await responsesClient.CreateResponseAsync(options);
```

5.5. Truy xuất ngữ nghĩa và sinh câu trả lời

Sau khi yêu cầu được gửi đi, OpenAI API sử dụng công cụ File Search để truy xuất các đoạn nội dung liên quan nhất từ Vector Store dựa trên độ tương đồng ngữ nghĩa giữa câu hỏi và nội dung tài liệu. Các đoạn truy xuất này, bao gồm cả phần metadata đã được nhúng, được cung cấp làm ngữ cảnh cho mô hình ngôn ngữ trong quá trình sinh câu trả lời.

Mô hình ngôn ngữ tổng hợp thông tin từ các đoạn truy xuất để sinh câu trả lời phù hợp với câu hỏi. Trong trường hợp nội dung truy xuất có chứa liên kết đến tài liệu gốc, mô hình được yêu cầu chèn liên kết này vào cuối câu trả lời theo định dạng Markdown. Nếu không tìm thấy thông tin phù hợp trong tài liệu, hệ thống trả về thông báo mặc định theo đúng quy định đã đặt ra trong system prompt.

```
var answer = response.Value.GetOutputText();
return string.IsNullOrWhiteSpace(answer)
? "Không tìm thấy thông tin phù hợp trong tài liệu.": answer;
```

5.6. Trả kết quả và hiển thị cho người dùng

Kết quả trả lời từ OpenAI API được ứng dụng ASP.NET Core tiếp nhận và hiển thị trực tiếp cho người dùng. Câu trả lời được trình bày ở dạng văn bản Markdown, cho phép hiển thị rõ ràng nội dung và liên kết trích dẫn. Hệ thống đồng thời lưu lại lịch sử hỏi – đáp trong phiên làm việc của người dùng để phục vụ việc theo dõi và tham khảo lại thông tin.

Cách tiếp cận này giúp phân tách rõ ràng vai trò của các thành phần trong hệ thống: dữ liệu được kiểm soát thông qua quá trình chuẩn hóa, hành vi sinh câu trả lời được kiểm soát thông qua system prompt, và ứng dụng web chỉ đảm nhiệm vai trò điều phối và hiển thị kết quả.

5.7. Tóm tắt quy trình xây dựng hệ thống

Tóm lại, quy trình xây dựng hệ thống hỏi – đáp trong nghiên cứu này tập trung vào việc kết hợp chuẩn hóa dữ liệu, truy xuất ngữ nghĩa và kiểm soát hành vi sinh câu trả lời thông qua prompt. Cách tiếp cận này cho phép triển khai một hệ thống hỏi – đáp gọn nhẹ về mặt kiến trúc, dễ triển khai trong môi trường thực tế, đồng thời vẫn đảm bảo tính chính xác và khả năng truy vết nguồn thông tin.

6. Thực nghiệm và đánh giá

6.1. Mục tiêu thực nghiệm

Thực nghiệm được tiến hành nhằm đánh giá hiệu quả của hệ thống chatbot tư vấn tuyển sinh được xây dựng dựa trên OpenAI API kết hợp File Search và Vector Store. Trọng tâm đánh giá tập trung vào khả năng truy xuất chính xác thông tin từ tài liệu tuyển sinh đã được chuẩn hóa, mức độ bao phủ các tình huống hỏi – đáp thực tế, cũng như khả năng kiểm soát suy diễn ngoài phạm vi dữ liệu được cung cấp.

6.2. Dữ liệu và thiết lập thực nghiệm

Nguồn dữ liệu sử dụng trong nghiên cứu là tài liệu chính thức "Thông tin tuyển sinh đại học năm 2025" do Trường Đại học Đồng Nai ban hành. Tài liệu gốc được cung cấp ở định dạng Microsoft Word (.docx) với dung lượng khoảng 4.000 từ, bao gồm các nội dung như đối tượng tuyển sinh, phương thức xét tuyển, ngành đào tạo, chính sách ưu tiên và các quy định liên quan.

Trước khi đưa vào hệ thống hỏi – đáp, tài liệu được xử lí theo quy trình chuẩn hóa dữ liệu gồm các bước sau:

Bước 1. Loại bỏ các thành phần trình bày không mang giá trị ngữ nghĩa, bao gồm header, footer, số trang và các kí hiệu định dạng hành chính.

Bước 2. Tái cấu trúc nội dung theo chủ đề, chia tài liệu gốc thành 8 tệp văn bản độc lập, tương ứng với các nhóm thông tin chính của tài liệu tuyển sinh.

Bước 3. Nhúng metadata vào phần đầu mỗi tệp, bao gồm các thông tin như tên văn bản, đơn vị ban hành, năm tuyển sinh và liên kết đến tài liệu gốc. Việc nhúng metadata trực tiếp vào nội dung văn bản nhằm đảm bảo rằng thông tin nguồn luôn đi kèm với các đoạn nội dung được truy xuất trong quá trình hỏi – đáp.

Bước 4. Sau khi chuẩn hóa, 8 tệp văn bản được tải lên Vector Store thông qua công cụ của OpenAI API.

Trong hệ thống File Search, nội dung tài liệu được tự động phân đoạn (chunking) và biểu diễn dưới dạng vector embedding để phục vụ truy xuất ngữ nghĩa. Các tham số kĩ thuật như kích thước đoạn văn bản, mô hình embedding và chiến lược truy xuất được OpenAI quản lí nội bộ, do đó nghiên cứu không thực hiện điều chỉnh trực tiếp các tham số này.

Trong quá trình xử lí câu hỏi, hệ thống gửi truy vấn đến OpenAI API, nơi File Search thực hiện truy xuất các đoạn văn bản liên quan từ Vector Store và cung cấp chúng làm ngữ cảnh cho mô hình ngôn ngữ [3]. Mô hình được sử dụng trong nghiên cứu là GPT-4o-mini, được lựa chọn nhằm cân bằng giữa chi phí vận hành, tốc độ phản hồi và chất lượng sinh ngôn ngữ.

Cách tiếp cận này cho phép hệ thống khai thác hiệu quả cơ chế truy xuất ngữ nghĩa tích hợp của File Search, đồng thời đơn giản hóa quá trình triển khai so với các hệ thống Retrieval-Augmented Generation (RAG) tự xây dựng.

Với quy mô dữ liệu tương đối nhỏ và có cấu trúc rõ ràng, cách tiếp cận sử dụng File Search cho phép triển khai nhanh hệ thống hỏi – đáp mà không cần xây dựng riêng các thành phần embedding, indexing và retrieval như trong các kiến trúc RAG truyền thống.

Hệ thống được triển khai dưới dạng ứng dụng web và được công bố tại địa chỉ ts.dnpu.edu.vn nhằm phục vụ quá trình thực nghiệm và đánh giá trong điều kiện sử dụng thực tế.

6.3. Bộ câu hỏi thử nghiệm

Bộ câu hỏi thử nghiệm được xây dựng dựa trực tiếp trên nội dung của tài liệu "Thông tin tuyển sinh đại học năm 2025", nhằm mô phỏng các tình huống hỏi – đáp thực tế của thí sinh và phụ huynh trong quá trình tìm hiểu thông tin tuyển sinh. Các câu hỏi được phân thành bốn nhóm chính, tương ứng với các mục tiêu đánh giá khác nhau của hệ thống.

Nhóm 1: Câu hỏi truy vấn trực tiếp thông tin

Nhóm này nhằm đánh giá khả năng truy xuất chính xác các thông tin được trình bày rõ ràng trong tài liệu tuyển sinh. Ví dụ một vài câu hỏi tiêu biểu:

– Trường tuyển sinh những đối tượng nào trong năm 2025?

– Trường có những phương thức xét tuyển nào?

– Ngành Kế toán được xét tuyển theo những phương thức nào?

– Học phí của ngành Sư phạm Toán học là bao nhiêu?

Nhóm 2: Câu hỏi tổng hợp thông tin theo chủ đề

Nhóm câu hỏi này yêu cầu hệ thống tổng hợp thông tin từ nhiều phần hoặc nhiều tệp khác nhau trong Vector Store. Ví dụ một vài câu hỏi tiêu biểu:

– Các ngành đào tạo thuộc khối kinh tế của trường gồm những ngành nào?

– Với phương thức xét tuyển học bạ, thí sinh cần đáp ứng những điều kiện gì?

– Những chính sách ưu tiên nào được áp dụng trong tuyển sinh năm 2025?

Nhóm 3: Câu hỏi kiểm tra thông tin không tồn tại trong tài liệu

Nhóm này nhằm đánh giá khả năng kiểm soát suy diễn và tránh tạo ra thông tin không có trong nguồn dữ liệu. Ví dụ một vài câu hỏi tiêu biểu:

– Trường có tuyển sinh ngành Công nghệ thông tin không?

– Có quy định về học bổng toàn phần cho sinh viên năm nhất không?

– Trường có xét tuyển bằng chứng chỉ IELTS không?

Nhóm 4: Câu hỏi diễn đạt tự nhiên, không trùng khớp từ khóa

Nhóm câu hỏi này phản ánh cách hỏi tự nhiên của người dùng, không bám sát tiêu đề hoặc từ khóa trong tài liệu. Ví dụ một vài câu hỏi tiêu biểu:

– Muốn học ngành Sư phạm thì cần điều kiện gì?

– Nếu tốt nghiệp trung học phổ thông thì có đủ điều kiện nộp hồ sơ không?

– Bao giờ thì trường bắt đầu nhận hồ sơ xét tuyển?

6.4. Tiêu chí đánh giá

Hệ thống được đánh giá dựa trên các tiêu chí sau: (1) độ chính xác nội dung so với tài liệu gốc; (2) mức độ bao phủ thông tin trong câu trả lời; (3) khả năng từ chối trả lời khi tài liệu không đề cập nội dung được hỏi; (4) tính tự nhiên và dễ hiểu của ngôn ngữ trả lời; và (5) tính nhất quán của kết quả đối với các câu hỏi tương đương về ngữ nghĩa.

6.5. Kết quả thực nghiệm và thảo luận

Kết quả thực nghiệm được tổng hợp trong bảng 1 cho thấy hệ thống chatbot đạt độ chính xác cao ở hầu hết các nhóm câu hỏi thử nghiệm. Đối với các câu hỏi truy vấn trực tiếp và câu hỏi tổng hợp theo chủ đề, chatbot trả lời đúng hoàn toàn dựa trên nội dung tài liệu, thể hiện rõ ưu thế của phương pháp truy vấn ngữ nghĩa khi dữ liệu được phân mảnh và lưu trữ trong Vector Store.

Đối với nhóm câu hỏi kiểm tra thông tin không tồn tại, hệ thống không tạo ra thông tin giả mà phản hồi phù hợp rằng tài liệu không đề cập nội dung được hỏi. Đây là một điểm mạnh quan trọng trong bối cảnh các hệ thống tư vấn tuyển sinh đòi hỏi độ tin cậy cao. Với các câu hỏi diễn đạt tự nhiên, không trùng khớp từ khóa, chatbot vẫn duy trì tỉ lệ trả lời chính xác cao, cho thấy khả năng hiểu ngữ nghĩa tốt, dù một số trường hợp cần thêm ngữ cảnh để đạt độ bao phủ thông tin tối ưu.

Bảng 2: *Tổng hợp kết quả đánh giá thực nghiệm hệ thống chatbot tư vấn tuyển sinh*

Nhóm câu hỏi	Mục tiêu đánh giá	Số câu hỏi	Tỉ lệ trả lời chính xác	Nhận xét học thuật
Truy vấn trực tiếp thông tin	Khả năng truy xuất thông tin có cấu trúc rõ ràng	10	100%	Câu trả lời chính xác, bám sát nội dung tài liệu, không xuất hiện thông tin dư thừa

Nhóm câu hỏi	Mục tiêu đánh giá	Số câu hỏi	Tỉ lệ trả lời chính xác	Nhận xét học thuật
Tổng hợp theo chủ đề	Khả năng kết hợp thông tin từ nhiều tệp trong Vector Store	8	100%	Thể hiện rõ ưu thế của truy vấn ngữ nghĩa so với tìm kiếm từ khóa
Kiểm tra thông tin không tồn tại	Khả năng kiểm soát suy diễn	6	100%	Không tạo thông tin giả, phản hồi phù hợp với phạm vi dữ liệu
Diễn đạt tự nhiên	Khả năng hiểu câu hỏi không trùng từ khóa	8	100%	Phần lớn câu hỏi được trả lời chính xác, một số trường hợp cần bổ sung ngữ cảnh

Nhìn chung, kết quả thực nghiệm cho thấy hệ thống đạt độ chính xác cao ở tất cả các nhóm câu hỏi thử nghiệm. Kết quả thực nghiệm khẳng định tính hiệu quả và độ tin cậy của chatbot trong bài toán hỏi – đáp dựa trên tài liệu tuyển sinh chính thức, đồng thời cho thấy tiềm năng ứng dụng thực tế của mô hình trong công tác tư vấn tuyển sinh tại các cơ sở giáo dục đại học.

6.6. Phân tích lỗi của hệ thống

Thứ nhất, đối với các câu hỏi nằm ngoài phạm vi nội dung của tài liệu, nếu không có cơ chế kiểm soát phù hợp trong system prompt, mô hình ngôn ngữ có thể sinh ra câu trả lời mang tính suy đoán. Trong nghiên cứu này, system prompt được thiết kế để yêu cầu mô hình chỉ sử dụng nội dung truy xuất từ tài liệu, giúp hạn chế hiện tượng này.

Thứ hai, hiệu quả truy xuất phụ thuộc vào cách phân đoạn nội dung tài liệu. Nếu các thông tin liên quan bị tách thành nhiều đoạn khác nhau trong quá trình chunking, hệ thống có thể cần nhiều ngữ cảnh hơn để tổng hợp đầy đủ thông tin.

Thứ ba, trong các ứng dụng thực tế sử dụng tài liệu ở dạng scan, sai lệch trong quá trình OCR có thể ảnh hưởng đến chất lượng embedding và kết quả truy xuất. Trong nghiên cứu này, tài liệu được cung cấp trực tiếp ở dạng văn bản nên không phát sinh vấn đề này.

Những phân tích này cho thấy việc chuẩn hóa dữ liệu và thiết kế prompt kiểm soát hành vi sinh nội dung đóng vai trò quan trọng trong việc đảm bảo độ tin cậy của hệ thống hỏi – đáp dựa trên tài liệu.

6.7. Phân tích chi phí và khả năng triển khai

Ngoài độ chính xác của câu trả lời, các yếu tố về độ trễ và chi phí vận hành cũng được xem xét nhằm đánh giá khả năng triển khai thực tế của hệ thống. Trong quá trình thử nghiệm, thời gian phản hồi trung bình của hệ thống dao động khoảng 1–3 giây cho mỗi truy vấn, bao gồm cả bước truy xuất ngữ nghĩa từ Vector Store và sinh câu trả lời bởi mô hình ngôn ngữ. Mức độ trễ này được đánh giá là phù hợp cho các ứng dụng hỏi – đáp trực tuyến.

Hệ thống sử dụng mô hình GPT-4o-mini, cho phép duy trì chi phí API ở mức thấp trong khi vẫn đảm bảo chất lượng sinh ngôn ngữ. Về khả năng mở rộng, kiến trúc dựa trên File Search và Vector Store cho phép bổ sung tài liệu mới mà không cần thay đổi cấu trúc hệ thống, cho thấy tiềm năng triển khai trong các môi trường dữ liệu lớn hơn.

7. Kết luận và hướng nghiên cứu tiếp theo

Bài báo đã trình bày một cách có hệ thống việc sử dụng OpenAI API với công cụ File Search để xây dựng hệ thống hỏi – đáp thông minh dựa trên nội dung tài liệu, thông qua nghiên cứu và thực nghiệm trong bài toán tuyển sinh đại học. Trọng tâm của nghiên cứu không nhằm phát triển một chatbot tư vấn theo nghĩa chung mà tập trung làm rõ quy trình, kiến trúc và phương pháp khai thác tài liệu để phục vụ bài toán hỏi – đáp chính xác, có kiểm soát và bám sát nguồn dữ liệu gốc.

Trên cơ sở mô hình Retrieval-Augmented Generation, bài báo đã phân tích vai trò của File Search và Vector Store như một lớp truy xuất ngữ nghĩa tích hợp, cho phép đơn giản hóa quá trình triển khai hệ thống hỏi – đáp mà vẫn đảm bảo các đặc trưng cốt lõi của RAG [2], [4]. Thông qua việc chuẩn hóa tài liệu, tổ chức dữ liệu trong Vector Store và kiểm soát chặt chẽ hành vi sinh câu trả lời của mô hình ngôn ngữ, hệ thống có khả năng trả lời các câu hỏi đa dạng dựa hoàn toàn trên nội dung tài liệu được cung cấp, đồng thời hạn chế hiệu quả hiện tượng suy diễn ngoài dữ liệu.

Kết quả thực nghiệm với bộ tài liệu tuyển sinh đại học cho thấy phương pháp đề xuất đạt độ chính xác cao đối với các nhóm câu hỏi khác nhau, từ truy vấn trực tiếp, tổng hợp thông tin theo chủ đề đến các câu hỏi diễn đạt tự nhiên hoặc kiểm tra thông tin không tồn tại trong tài liệu. Những kết quả này khẳng định tính phù hợp của cách tiếp cận File Search trong các bài toán hỏi – đáp dựa trên tài liệu chính thức, đặc biệt trong bối cảnh dữ liệu có cấu trúc phức tạp, thường xuyên cập nhật và yêu cầu cao về tính nhất quán.

Từ góc độ ứng dụng, hệ thống được triển khai thử nghiệm tại địa chỉ https://ts.dnpu.edu.vn đóng vai trò như một minh chứng thực tế cho khả năng áp dụng mô hình nghiên cứu. Việc triển khai này không nhằm giới thiệu một sản phẩm hoàn chỉnh mà nhằm kiểm chứng tính khả thi của quy trình và kiến trúc đề xuất trong môi trường sử dụng thực tế, qua đó làm rõ tiềm năng mở rộng của phương pháp đối với các bài toán hỏi – đáp tài liệu trong giáo dục và quản lí.

Bên cạnh các kết quả đạt được, nghiên cứu vẫn tồn tại những giới hạn nhất định. Phương pháp hiện tại phụ thuộc lớn vào chất lượng chuẩn hóa và phân chia tài liệu đầu vào; các tài liệu chưa được xử lí tốt có thể ảnh hưởng trực tiếp đến hiệu quả truy xuất ngữ nghĩa. Ngoài ra, hệ thống trong nghiên cứu này sử dụng File Search như một cơ chế truy xuất tích hợp sẵn, chưa đi sâu vào việc tối ưu hoặc so sánh các chiến lược retrieval khác nhau.

Trong các nghiên cứu tiếp theo, hướng mở rộng quan trọng là khai thác sâu hơn thành phần retrieval trong mô hình RAG, bao gồm việc tách riêng và đánh giá độc lập các module truy xuất, thử nghiệm các chiến lược embedding, phân đoạn và xếp hạng ngữ cảnh khác nhau. Việc so sánh giữa File Search và các cơ chế retrieval tự xây dựng sẽ giúp làm rõ hơn vai trò của từng thành phần trong hệ thống hỏi – đáp, đồng thời mở ra hướng nghiên cứu chuyên sâu hơn cho các bài báo kế tiếp. Những hướng phát triển này không chỉ có ý nghĩa học thuật mà còn góp phần nâng cao khả năng ứng dụng thực tiễn của các hệ thống hỏi – đáp dựa trên tài liệu trong bối cảnh chuyển đổi số giáo dục. Nghiên cứu này được xem như bước nền tảng cho các nghiên cứu tiếp theo tập trung chuyên sâu vào thành phần retrieval trong mô hình RAG, bao gồm so sánh File Search với các chiến lược truy xuất tự xây dựng.

TÀI LIỆU THAM KHẢO

[1] T. B. Brown *et al*., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.
[2] P. Lewis *et al*., "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
[3] OpenAI, "OpenAI API documentation," 2024. [Online]. Available: https://platform.openai.com/docs.
[4] OpenAI, "File Search and Vector Stores," 2024. [Online]. Available: https://platform.openai.com/docs/guides/tools-file-search.
[5] N. Thakur et al., "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models," NeurIPS Datasets and Benchmarks, 2021.
[6] S. Ruder et al., "XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization," ICML, 2021.
[7] V. Karpukhin *et al*., "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020, pp. 6769–6781.
[8] OpenAI, "OpenAI .NET API library," 2025. [Online]. Available: https://github.com/openai/openai-dotnet.

USING THE OPENAI API AND FILE SEARCH TO BUILD AN INTELLIGENT DOCUMENT-BASED QUESTION–ANSWERING SYSTEM: A STUDY AND EXPERIMENTAL EVALUATION IN UNIVERSITY ADMISSIONS

Le Xuan Hung
Dong Nai University
Email: hunglx.it@gmail.com
(Received: 8/1/2026, Revised: 7/3/2026, Accepted for publication: 4/6/2026)

ABSTRACT

The increasing volume of digital documents in higher education creates a demand for solutions that support accurate, consistent, and accessible information delivery. For document-based question–answering tasks in specialized domains such as university admissions, directly using large language models may introduce the risk of generating information that is not grounded in the source data. This study proposes a document-based question–answering approach by integrating the OpenAI API with the File Search tool to support semantic information retrieval. Based on the Retrieval-Augmented Generation (RAG) architecture, the proposed approach includes data preprocessing and normalization, document organization within a Vector Store, the design of a controlled answer-generation mechanism, and system implementation on the ASP.NET Core platform. Experimental evaluation was conducted using official university admissions documents for the 2025 academic year along with a set of questions designed to simulate real-world inquiry scenarios. The results indicate that the system generates responses closely aligned with source documents, effectively reduces unsupported content generation, and maintains stable retrieval performance during operation.

Keywords: *File Search; document-based question-answering system; OpenAI API; Retrieval-Augmented Generation (RAG); Semantic Retrieval; Vector Store*