



# Báo cáo dự án

Môn học: Khai phá dữ liệu

Học kỳ I, năm học 2025-2026

---

## Bài thuyết trình kèm ghi chú của người nói được tạo bởi AI (tự động thuyết minh)

---

Khoa Toán – Cơ – Tin học

Trường Đại học Khoa học Tự nhiên, ĐHQGHN

*Sinh viên thực hiện:*

Hoàng Lê Tuấn – 22001653

*Giảng viên hướng dẫn:*

PGS.TS.Lê Hoàng Sơn

Viện công nghệ thông tin, ĐHQGHN

# Lời cảm ơn

# Danh sách hình vẽ

# Danh sách bảng

# Mục lục

# Lời mở đầu

Trong bối cảnh kỷ nguyên chuyển đổi số và cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ trên toàn cầu, việc ứng dụng trí tuệ nhân tạo (AI) và kỹ thuật khai phá dữ liệu vào các hoạt động nghiệp vụ hàng ngày đã trở thành xu hướng tất yếu. Đặc biệt trong lĩnh vực giáo dục và truyền thông, nhu cầu tạo ra các bài thuyết trình chất lượng cao với thời gian ngắn và chi phí tối ưu đang ngày càng gia tăng. Tuy nhiên, quá trình tạo lập một bài thuyết trình hoàn chỉnh, bao gồm cả nội dung slide và ghi chú thuyết minh chi tiết cho người trình bày thường đòi hỏi nhiều thời gian, công sức và kỹ năng chuyên môn cao. Nhận thức được thực trạng này, đề tài nghiên cứu "Bài thuyết trình kèm ghi chú của người nói được tạo bởi AI (tự động thuyết minh)" được triển khai nhằm xây dựng một hệ thống tự động hóa toàn diện, ứng dụng các kỹ thuật tiên tiến trong lĩnh vực khai phá dữ liệu và học máy để tạo sinh các bài thuyết trình PowerPoint cùng với âm thanh thuyết minh bằng tiếng Việt.

## Bối cảnh và tính cấp thiết của đề tài

Theo nghiên cứu về khai phá dữ liệu (data mining), đây là quá trình phân loại, sắp xếp các tập hợp dữ liệu lớn để xác định các mẫu và thiết lập các mối liên hệ nhằm giải quyết các vấn đề thông qua hoạt động phân tích dữ liệu. Trong kỷ nguyên số hiện nay, dữ liệu được tạo ra với tốc độ chưa từng có, và việc khai thác hiệu quả các nguồn dữ liệu này để phục vụ cho các ứng dụng thực tiễn là một thách thức lớn. Đặc biệt

đối với ngôn ngữ tiếng Việt, một ngôn ngữ thuộc nhóm tài nguyên thấp (low-resource language) trong lĩnh vực xử lý ngôn ngữ tự nhiên, việc phát triển các ứng dụng AI chất lượng cao còn gặp nhiều hạn chế về nguồn dữ liệu và mô hình được huấn luyện sẵn.

Retrieval-Augmented Generation (RAG) là một kỹ thuật tiên tiến cho phép các mô hình ngôn ngữ lớn (LLMs) truy xuất và kết hợp thông tin mới từ các nguồn dữ liệu bên ngoài. Theo nghiên cứu được công bố trên Wikipedia, RAG giúp giảm thiểu hiện tượng "ảo giác" (hallucination) trong các chatbot AI, hiện tượng mà hệ thống tạo ra các thông tin không chính xác nhưng có vẻ hợp lý. Bằng cách kết hợp cơ chế truy xuất thông tin với khả năng sinh văn bản, RAG cho phép LLMs truy cập kiến thức cập nhật và cụ thể theo từng lĩnh vực mà không cần phải huấn luyện lại toàn bộ mô hình. Điều này đặc biệt quan trọng trong bối cảnh chi phí tính toán và tài chính của việc huấn luyện lại các mô hình lớn là cực kỳ tốn kém.

Prompt engineering là một kỹ thuật thiết kế các câu lệnh đầu vào để hướng dẫn mô hình AI, đã phát triển từ các phương pháp cơ bản đến các kỹ thuật nâng cao như phương pháp prompt theo chuỗi lập luận, tính tự nhất quán và tự cải thiện theo vòng lặp đệ quy. Các nghiên cứu gần đây cho thấy rằng việc áp dụng các kỹ thuật prompt engineering tiên tiến có thể cải thiện đáng kể tỷ lệ giải quyết vấn đề trong các mô hình ngôn ngữ lớn, đặc biệt là đối với các tác vụ phức tạp đòi hỏi suy luận logic và phân tích đa chiều.

Về mặt ứng dụng, việc tự động hóa PowerPoint bằng Python thông qua thư viện python-pptx đã được chứng minh là một giải pháp hiệu quả cho việc tạo lập và chỉnh sửa các bài thuyết trình một cách lập trình. Điều này cho phép xây dựng các hệ thống có khả năng tạo ra hàng loạt bài thuyết trình với nội dung động dựa trên dữ liệu đầu vào, tiết kiệm thời gian và công sức đáng kể so với việc tạo thủ công.

Đối với công nghệ chuyển văn bản thành giọng nói (Text-to-Speech - TTS) tiếng Việt,

các nghiên cứu cho thấy có nhiều giải pháp khả dụng từ các nền tảng như Google Text-to-Speech (gTTS), ElevenLabs, và MiniMax Audio. Các công nghệ này có khả năng tạo ra giọng nói tự nhiên với các đặc điểm ngữ điệu phù hợp với ngôn ngữ tiếng Việt, hỗ trợ hiệu quả cho việc tạo nội dung âm thanh cho các bài thuyết trình.

Trong lĩnh vực DataOps, phương pháp luận nhằm tối ưu hóa quản lý và vận hành các luồng dữ liệu, việc xây dựng các pipeline dữ liệu tự động và có khả năng mở rộng là yếu tố then chốt để đảm bảo chất lượng và hiệu suất của các hệ thống dựa trên dữ liệu. Các thực hành tốt nhất trong DataOps bao gồm tự động hóa các pipeline, đảm bảo chất lượng dữ liệu thông qua các cơ chế validation, và xây dựng các hệ thống có khả năng giám sát và phục hồi tự động khi có lỗi xảy ra.

Về mặt nền tảng triển khai, Drupal là một hệ quản trị nội dung (CMS) mã nguồn mở mạnh mẽ, được sử dụng bởi hàng triệu tổ chức trên toàn cầu. Với kiến trúc API-first, Drupal cho phép tách biệt nội dung khỏi lớp trình bày, tạo điều kiện thuận lợi cho việc phát triển các ứng dụng đa nền tảng. Apache Superset, một nền tảng business intelligence (BI) mã nguồn mở, cung cấp khả năng trực quan hóa dữ liệu mạnh mẽ với giao diện kéo-thả trực quan và khả năng tích hợp với nhiều nguồn dữ liệu khác nhau. Sự kết hợp giữa Drupal và Apache Superset tạo nên một giải pháp toàn diện cho việc quản lý nội dung và phân tích dữ liệu trong doanh nghiệp.

# Tình hình nghiên cứu trong và ngoài nước

Trên thế giới, lĩnh vực RAG đã có những bước phát triển vượt bậc kể từ khi được giới thiệu lần đầu trong nghiên cứu năm 2020. Các nghiên cứu gần đây đã mở rộng ứng dụng của RAG từ các tác vụ hỏi đáp mở (open-domain Q&A) sang nhiều lĩnh vực khác như tóm tắt văn bản, các tác nhân hội thoại, và hệ thống gợi ý cá nhân hóa. Một nghiên cứu tổng quan có hệ thống về RAG đã chỉ ra rằng việc tích hợp hệ thống bản thể học và biểu đồ tri thức với RAG có thể cải thiện đáng kể độ chính xác trong việc

truy xuất thông tin và sinh câu trả lời.

Về prompt engineering, các nghiên cứu đã phát triển nhiều kỹ thuật tiên tiến như phương pháp prompt với vài mẫu, lập luận theo chuỗi tư duy, tự tinh chỉnh, và phân tách bài toán để tạo ra các giải pháp AI hiệu quả hơn. Năm 2025 đánh dấu sự chuyển mình từ việc xem AI như một hệ thống thực thi lệnh đơn giản sang việc tận dụng khả năng siêu nhận thức, nhìn nhận từ nhiều góc độ và cải tiến lặp đi lặp lại của các mô hình.

Tại Việt Nam, nghiên cứu về khai phá dữ liệu đã được quan tâm và ứng dụng trong nhiều lĩnh vực như ngân hàng, thương mại điện tử, và dự báo thời tiết. Các công cụ khai phá dữ liệu phổ biến như RapidMiner, Orange, và Oracle DataMining đã được nhiều tổ chức trong nước áp dụng. Tuy nhiên, việc nghiên cứu và phát triển các ứng dụng AI cho ngôn ngữ tiếng Việt vẫn còn nhiều thách thức do hạn chế về nguồn dữ liệu chất lượng cao và các mô hình được huấn luyện đặc thù cho tiếng Việt.

Trong lĩnh vực NLP tiếng Việt, các nghiên cứu gần đây đã phát triển các mô hình ngôn ngữ như PhoBERT, một biến thể của BERT được huấn luyện trên corpus tiếng Việt quy mô lớn. Nghiên cứu cho thấy các mô hình chuyên biệt cho một ngôn ngữ cụ thể thường cho kết quả tốt hơn so với các mô hình đa ngôn ngữ tổng quát. Đối với embedding models, các nghiên cứu đã phát triển Vietnamese Document Embedding, một mô hình chuyên biệt cho việc tạo embedding cho văn bản tiếng Việt dài với độ chính xác cao.

Về vấn đề đạo văn trong viết học thuật tại Việt Nam, các nghiên cứu cho thấy tỷ lệ đạo văn trong các báo cáo tốt nghiệp của sinh viên Việt Nam còn khá cao, với nhiều nghiên cứu ghi nhận tỷ lệ tương đồng (Similarity Index) trung bình lên đến 42.6%. Điều này phần lớn do sinh viên thiếu hiểu biết về các hình thức đạo văn và kỹ năng trích dẫn nguồn còn hạn chế. Do đó, việc xây dựng các hệ thống có khả năng tổng hợp và trích dẫn nguồn tài liệu một cách chính xác là một yêu cầu quan trọng trong

bối cảnh giáo dục Việt Nam hiện nay.

# Lý do chọn đề tài

Xuất phát từ nhu cầu thực tiễn trong việc tự động hóa quy trình tạo bài thuyết trình và nhận thức được tiềm năng to lớn của các kỹ thuật AI tiên tiến, đề tài "Bài thuyết trình kèm ghi chú của người nói được tạo bởi AI (tự động thuyết minh)" được lựa chọn với những lý do chính sau đây:

**Thứ nhất,** đề tài giải quyết một nhu cầu thực tế đang ngày càng tăng trong xã hội hiện đại. Trong môi trường giáo dục, giảng viên và sinh viên thường xuyên phải chuẩn bị các bài thuyết trình cho các môn học, hội thảo, và báo cáo nghiên cứu. Quá trình này tốn rất nhiều thời gian và công sức, đặc biệt là việc soạn thảo ghi chú thuyết minh chi tiết. Một hệ thống tự động có thể giảm thiểu đáng kể gánh nặng này và cho phép người dùng tập trung vào nội dung chuyên môn thay vì các công việc định dạng và tổ chức.

**Thứ hai,** đề tài cho phép nghiên cứu và ứng dụng các kỹ thuật khai phá dữ liệu tiên tiến, cụ thể là RAG, một trong những kỹ thuật được yêu cầu trong đề cương môn học. RAG đại diện cho sự kết hợp sáng tạo giữa truy xuất thông tin và sinh văn bản, mở ra hướng tiếp cận mới trong việc xây dựng các hệ thống AI có khả năng truy cập và tổng hợp thông tin từ các nguồn dữ liệu bên ngoài. Việc triển khai RAG trên dữ liệu tiếng Việt còn có ý nghĩa nghiên cứu quan trọng, góp phần vào việc phát triển các ứng dụng AI cho ngôn ngữ tiếng Việt, một lĩnh vực còn nhiều dư địa để khai phá.

**Thứ ba,** đề tài tích hợp nhiều kỹ thuật và công nghệ tiên tiến trong một hệ thống hoàn chỉnh, bao gồm: xử lý dữ liệu quy mô lớn, embedding models đa ngôn ngữ, vector database (FAISS), prompt engineering nâng cao, tự động hóa PowerPoint, text-to-speech, và tích hợp với các nền tảng mã nguồn mở như Drupal và Apache

Superset. Sự kết hợp này không chỉ đáp ứng yêu cầu kỹ thuật của môn học mà còn tạo ra một giải pháp có tính ứng dụng cao trong thực tế.

**Thứ tư,** đề tài xây dựng một DataOps pipeline hoàn chỉnh với khả năng tự động hóa cao, từ việc thu thập và xử lý dữ liệu đến việc tạo sinh sản phẩm cuối cùng. Pipeline này tuân theo các phương pháp thực tiễn tốt nhất trong ngành về tính mô-đun hóa, khả năng mở rộng, và khả năng giám sát. Việc áp dụng các nguyên tắc DataOps không chỉ đảm bảo chất lượng của hệ thống mà còn tạo điều kiện thuận lợi cho việc bảo trì và phát triển trong tương lai.

**Thứ năm,** đề tài có khả năng mở rộng và phát triển trong tương lai. Kiến trúc mô-đun hóa của hệ thống cho phép dễ dàng thay thế hoặc nâng cấp các thành phần riêng lẻ mà không ảnh hưởng đến toàn bộ hệ thống. Ví dụ, mô hình embedding có thể được nâng cấp lên các phiên bản mới hơn, nguồn dữ liệu có thể được mở rộng, và các tính năng mới có thể được bổ sung một cách linh hoạt.

**Thứ sáu,** từ góc độ học thuật, đề tài đóng góp vào việc giải quyết vấn đề đạo văn, một vấn đề nghiêm trọng trong giáo dục Việt Nam hiện nay. Bằng cách sử dụng RAG với cơ chế trích dẫn nguồn rõ ràng, hệ thống không chỉ tạo ra nội dung mà còn cung cấp thông tin về nguồn gốc của các tri thức được sử dụng, giúp người dùng hiểu và tuân thủ các nguyên tắc trích dẫn học thuật.

# Chương 1

# Giới thiệu về đề tài dự án

Chương này trình bày các khái niệm nền tảng và công nghệ liên quan đến đề tài "Bài thuyết trình kèm ghi chú của người nói được tạo bởi AI". Nội dung chương bao gồm các kiến thức về khai phá dữ liệu, Retrieval-Augmented Generation (RAG), kỹ thuật thiết kế câu lệnh, các công nghệ embedding và cơ sở dữ liệu vector, cùng với các công nghệ hỗ trợ như python-pptx, chuyển văn bản thành giọng nói, Drupal CMS và Apache Superset. Việc nắm vững các khái niệm này là cần thiết để hiểu rõ thiết kế và triển khai hệ thống trong các chương tiếp theo.

## 1.1 Tổng quan về khai phá dữ liệu

### 1.1.1 Định nghĩa và vai trò

**Khai phá dữ liệu (data mining)** là một quá trình tính toán nhằm khám phá và phân tích một lượng lớn dữ liệu để xác định các mẫu (patterns), thiết lập các mối quan hệ (relationships), và trích xuất thông tin hữu ích phục vụ cho việc giải quyết vấn đề thông qua hoạt động phân tích dữ liệu. Theo định nghĩa từ các tài liệu học thuật, khai phá dữ liệu là quá trình sử dụng phân tích thống kê và học máy để phát hiện các mẫu

ẩn, mối tương quan và bất thường trong các tập dữ liệu lớn. Đây không chỉ đơn thuần là việc trích xuất dữ liệu, mà còn là quá trình chuyển đổi dữ liệu thô thành kiến thức có giá trị cho việc ra quyết định.

Trong kỷ nguyên số hiện nay, vai trò của khai phá dữ liệu trở nên quan trọng hơn bao giờ hết. Với sự bùng nổ của dữ liệu lớn (big data), các tổ chức thu thập và lưu trữ khối lượng dữ liệu khổng lồ từ nhiều nguồn khác nhau như giao dịch khách hàng, hoạt động trên mạng xã hội, dữ liệu cảm biến IoT, và hồ sơ y tế. Khai phá dữ liệu cho phép các tổ chức khai thác các nguồn dữ liệu này để phát hiện các thông tin giá trị, dự đoán xu hướng tương lai, và tối ưu hóa các quy trình kinh doanh. Ví dụ, trong lĩnh vực bán lẻ, khai phá dữ liệu giúp phân tích hành vi mua sắm của khách hàng để tạo ra các chiến dịch marketing cá nhân hóa. Trong y tế, nó hỗ trợ việc chẩn đoán bệnh và phát hiện các tác dụng phụ của thuốc.

Khai phá dữ liệu khác biệt với việc truy vấn dữ liệu truyền thống ở chỗ nó không chỉ tìm kiếm thông tin đã biết mà còn phát hiện ra các kiến thức mới mà trước đó chưa được nhận biết. Quá trình này thường bao gồm nhiều giai đoạn như tiền xử lý dữ liệu (data preprocessing), lựa chọn đặc trưng (feature selection), áp dụng các thuật toán khai phá, và đánh giá kết quả. Mục tiêu cuối cùng là chuyển đổi dữ liệu thành hành động có thể thực hiện được để hỗ trợ việc ra quyết định kinh doanh hoặc nghiên cứu khoa học.

### 1.1.2 Các kỹ thuật chính

Khai phá dữ liệu bao gồm nhiều kỹ thuật khác nhau, mỗi kỹ thuật phù hợp với các loại vấn đề và dữ liệu cụ thể. Các kỹ thuật này có thể được chia thành hai nhóm chính: học có giám sát (supervised learning) và học không giám sát (unsupervised learning).

- **Phân loại (classification)** là một kỹ thuật học có giám sát được sử dụng rộng rãi nhất trong khai phá dữ liệu. Trong phân loại, mục tiêu là xây dựng một mô

hình có khả năng gán một nhãn lớp (class label) cho các thực thể mới dựa trên các thuộc tính của chúng. Dữ liệu huấn luyện trong phân loại có một thuộc tính đặc biệt được gọi là thuộc tính lớp (class attribute), và nhiệm vụ là tìm ra một hàm ánh xạ từ các thuộc tính đầu vào đến thuộc tính lớp này. Các phương pháp phân loại phổ biến bao gồm khớp láng giềng gần nhất (nearest neighbor matching), cây quyết định (decision trees), phân loại dựa trên luật (rule-based classification), và mạng nơ-ron (neural networks). Ví dụ, một công ty bảo hiểm có thể sử dụng phân loại để phân loại khách hàng thành các nhóm rủi ro cao, trung bình, hoặc thấp dựa trên các đặc điểm như tuổi, nghề nghiệp, và lịch sử bồi thường.

- **Phân cụm (Clustering)** là một kỹ thuật học không giám sát nhằm chia một tập các đối tượng thành các nhóm đồng nhất (clusters) sao cho các đối tượng trong cùng một cụm có độ tương đồng cao với nhau và khác biệt với các đối tượng trong các cụm khác. Khác với phân loại, phân cụm không có thuộc tính lớp được định nghĩa trước, và tất cả các thuộc tính được xem xét ngang nhau. Các thuật toán phân cụm phổ biến bao gồm k-means, phân cụm phân cấp (hierarchical clustering), và DBSCAN. Phân cụm được ứng dụng rộng rãi trong phân khúc khách hàng (customer segmentation), phát hiện bất thường (anomaly detection), và tổ chức dữ liệu. Ví dụ, một nền tảng thương mại điện tử có thể sử dụng phân cụm để nhóm khách hàng có hành vi mua sắm tương tự nhau, từ đó tạo ra các chiến dịch marketing nhắm mục tiêu cho từng nhóm.

- **Khai phá luật kết hợp (Association rule mining)** là kỹ thuật tìm kiếm các mối quan hệ thú vị và mối tương quan tồn tại giữa các giá trị của các biến trong tập dữ liệu. Kỹ thuật này đặc biệt hữu ích trong phân tích giỏ hàng (market basket analysis), nơi mục tiêu là tìm ra các sản phẩm thường được mua cùng nhau. Các luật kết hợp thường được biểu diễn dưới dạng "nếu-thì" (if-then), ví dụ: "nếu khách hàng mua bánh mì và bơ, thì họ có khả năng cao mua sữa". Để

đánh giá chất lượng của các luật kết hợp, người ta sử dụng các chỉ số như độ hỗ trợ (support - tần suất xuất hiện), độ tin cậy (confidence), và độ nâng (lift - mức độ nâng cao). Thuật toán Apriori là một trong những thuật toán phổ biến nhất cho khai phá luật kết hợp.

- **Phân tích hồi quy (regression analysis)** là một kỹ thuật học có giám sát được sử dụng khi thuộc tính đích là một giá trị số liên tục thay vì một nhãn lớp rời rạc. Mục tiêu của hồi quy là xây dựng một mô hình toán học mô tả mối quan hệ giữa các biến độc lập (independent variables) và biến phụ thuộc (dependent variable). Các kỹ thuật hồi quy phổ biến bao gồm hồi quy tuyến tính (linear regression), hồi quy đa thức (polynomial regression), và hồi quy vector hỗ trợ (support vector regression). Hồi quy được ứng dụng rộng rãi trong dự báo (forecasting), ví dụ như dự đoán giá bất động sản dựa trên diện tích, vị trí, và các đặc điểm khác, hoặc dự đoán doanh số bán hàng dựa trên chi tiêu quảng cáo.

Ngoài các kỹ thuật chính trên, khai phá dữ liệu còn bao gồm nhiều kỹ thuật khác như phát hiện bất thường (anomaly detection), khai phá mẫu tuần tự (sequential pattern mining), và khai phá văn bản (text mining). Mỗi kỹ thuật có những ưu điểm và hạn chế riêng, và việc lựa chọn kỹ thuật phù hợp phụ thuộc vào bản chất của vấn đề cần giải quyết, loại dữ liệu có sẵn, và mục tiêu cụ thể của phân tích.

### 1.1.3 Quy trình

Quy trình khai phá dữ liệu là một chuỗi các bước có hệ thống nhằm chuyển đổi dữ liệu thô thành kiến thức có giá trị. Quy trình này thường tuân theo mô hình CRISP-DM (Cross-Industry Standard Process for Data Mining) hoặc KDD (Knowledge Discovery in Databases), cả hai đều bao gồm các giai đoạn tương tự nhau.

- **Giai đoạn hiểu nghiệp vụ (business understanding)** là bước đầu tiên và quan

trọng nhất trong quy trình khai phá dữ liệu. Trong giai đoạn này, cần xác định rõ mục tiêu kinh doanh, các yêu cầu cụ thể, và chuyển đổi chúng thành các mục tiêu khai phá dữ liệu cụ thể. Ví dụ, mục tiêu kinh doanh "tăng doanh số bán hàng" có thể được chuyển đổi thành mục tiêu khai phá dữ liệu "xây dựng mô hình dự đoán khách hàng có khả năng cao mua sản phẩm X".

- **Giai đoạn hiểu dữ liệu (data understanding)** bao gồm việc thu thập dữ liệu ban đầu, làm quen với dữ liệu thông qua phân tích khám phá (exploratory data analysis), và đánh giá chất lượng dữ liệu. Trong giai đoạn này, các nhà phân tích cần hiểu về cấu trúc dữ liệu, phân bố các giá trị, mối quan hệ giữa các biến, và các vấn đề tiềm ẩn như giá trị thiếu (missing values), giá trị ngoại lai (outliers), hay tính không nhất quán (inconsistencies). Việc trực quan hóa dữ liệu thông qua các biểu đồ và đồ thị cũng rất hữu ích trong giai đoạn này.

- **Giai đoạn chuẩn bị dữ liệu (data preparation)** thường chiếm phần lớn thời gian trong toàn bộ quy trình, đôi khi lên đến 70-80% tổng thời gian dự án. Giai đoạn này bao gồm nhiều hoạt động như làm sạch dữ liệu (data cleaning - xử lý giá trị thiếu, loại bỏ bản sao, sửa lỗi), chuyển đổi dữ liệu (data transformation - chuẩn hóa, mã hóa biến phân loại), tích hợp dữ liệu từ nhiều nguồn (data integration), và kỹ thuật tạo đặc trưng (feature engineering - tạo các đặc trưng mới từ các biến hiện có). Chất lượng của dữ liệu sau giai đoạn này có ảnh hưởng quyết định đến hiệu quả của các mô hình được xây dựng.

- **Giai đoạn mô hình hóa (modeling)** là nơi các thuật toán khai phá dữ liệu được áp dụng vào dữ liệu đã được chuẩn bị. Trong giai đoạn này, có thể thử nghiệm nhiều thuật toán khác nhau và điều chỉnh các siêu tham số (hyperparameters) để tìm ra mô hình tốt nhất. Việc chia dữ liệu thành tập huấn luyện (training set), tập xác thực (validation set), và tập kiểm thử (test set) là một thực hành quan trọng để đảm bảo mô hình có khả năng tổng quát hóa tốt.

- **Giai đoạn đánh giá (evaluation)** nhằm đánh giá chất lượng và tính hiệu quả của các mô hình đã được xây dựng. Các chỉ số đánh giá phụ thuộc vào loại bài toán, ví dụ độ chính xác (accuracy), độ chính xác dương (precision), độ thu hồi (recall), điểm F1 (F1-score) cho phân loại; RMSE, MAE, R-squared cho hồi quy. Ngoài các chỉ số định lượng, cần đánh giá xem mô hình có thực sự giải quyết được vấn đề kinh doanh ban đầu hay không.
- **Giai đoạn triển khai (deployment)** là bước cuối cùng, trong đó mô hình được đưa vào sử dụng thực tế trong môi trường sản xuất. Điều này bao gồm việc tích hợp mô hình vào các hệ thống hiện có, thiết lập giám sát để theo dõi hiệu suất mô hình theo thời gian, và xây dựng quy trình để cập nhật mô hình khi cần thiết. Việc triển khai cũng cần đảm bảo các yếu tố về bảo mật, khả năng mở rộng, và độ tin cậy.

## 1.2 Retrieval-Augmented Generation (RAG)

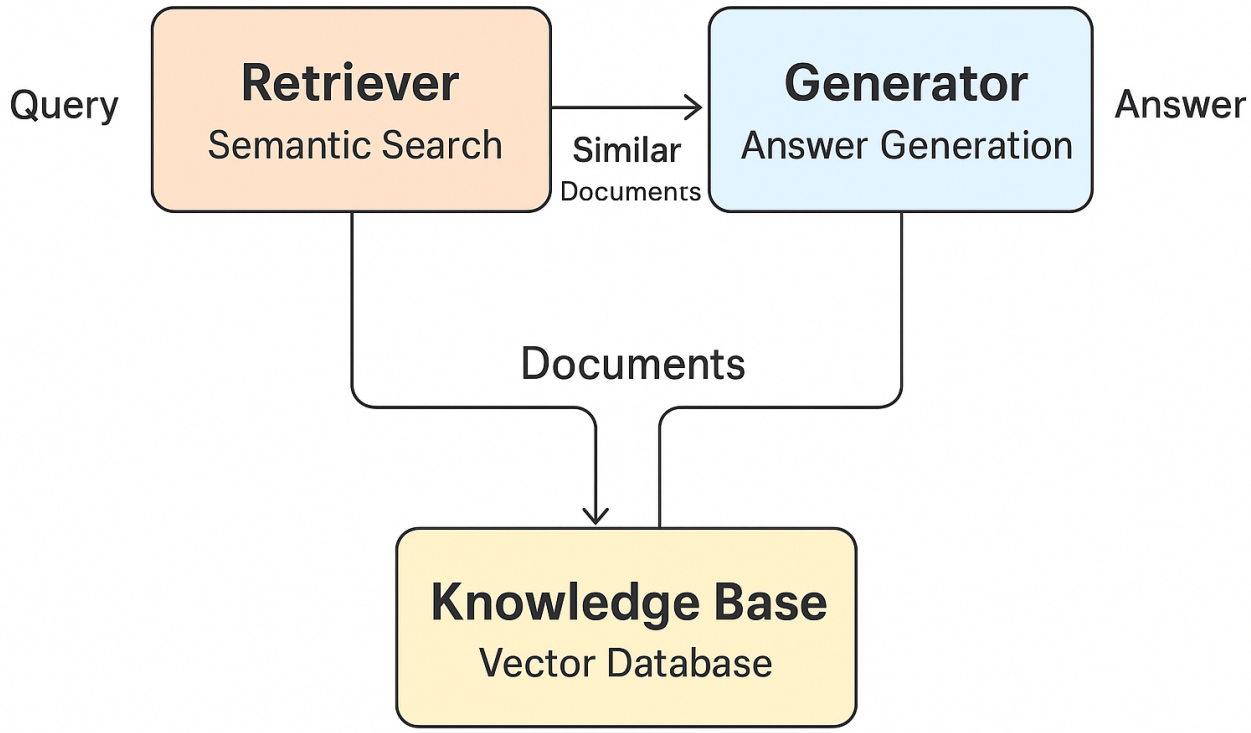
### 1.2.1 Khái niệm và kiến trúc

**Retrieval-Augmented Generation (RAG)** là một kỹ thuật tiên tiến trong lĩnh vực trí tuệ nhân tạo nhằm nâng cao khả năng của các mô hình ngôn ngữ lớn (Large Language Models - LLMs) bằng cách kết hợp cơ chế truy xuất thông tin (information retrieval) với khả năng sinh văn bản (text generation). Được giới thiệu lần đầu tiên vào năm 2020, RAG đại diện cho một bước tiến quan trọng trong việc giải quyết vấn đề "ảo giác" (hallucination) - hiện tượng mà các mô hình AI tạo ra thông tin không chính xác nhưng có vẻ hợp lý.

Ý tưởng cốt lõi của RAG là thay vì chỉ dựa vào kiến thức được mã hóa trong các tham số của mô hình trong quá trình huấn luyện, hệ thống sẽ truy xuất thông tin liên quan từ một cơ sở tri thức bên ngoài và sử dụng thông tin này như ngữ cảnh để sinh câu trả

lời. Điều này giúp mô hình có thể truy cập kiến thức cập nhật và cụ thể mà không cần phải huấn luyện lại toàn bộ mô hình - một quá trình rất tốn kém về mặt tính toán và tài chính.

Kiến trúc RAG bao gồm ba thành phần chính hoạt động phối hợp với nhau được mô tả dựa trên hình 1.1 dưới đây.




Hình 1.1: Sơ đồ kiến trúc RAG

- **Thành phần bộ truy xuất (retriever)** là thành phần đầu tiên trong luồng xử lý của RAG. Bộ truy xuất có nhiệm vụ tìm kiếm và truy xuất các tài liệu hoặc đoạn văn liên quan nhất từ cơ sở tri thức dựa trên câu truy vấn của người dùng. Quá trình này thường sử dụng các kỹ thuật tìm kiếm ngữ nghĩa (semantic search) thay vì tìm kiếm từ khóa truyền thống (keyword search). Bộ truy xuất nhận đầu vào là câu hỏi hoặc câu truy vấn từ người dùng, chuyển đổi nó thành biểu diễn

vector, sau đó thực hiện tìm kiếm độ tương đồng trong cơ sở dữ liệu vector để tìm ra top-k tài liệu có độ tương đồng cao nhất. Các phương pháp truy xuất phổ biến bao gồm truy xuất dày đặc (dense retrieval - sử dụng neural embeddings), truy xuất thưa thớt (sparse retrieval - sử dụng TF-IDF hoặc BM25), và truy xuất kết hợp (hybrid retrieval - kết hợp cả hai).

- **Thành phần bộ sinh (generator)** là mô hình ngôn ngữ lớn có nhiệm vụ tổng hợp thông tin từ các tài liệu được truy xuất và tạo ra câu trả lời cuối cùng. Bộ sinh thường dựa trên kiến trúc Transformer như BART, T5, hoặc GPT. Thành phần này nhận đầu vào là câu hỏi của người dùng được tăng cường với các tài liệu được truy xuất, sau đó sử dụng cơ chế chú ý (attention mechanism) để kết hợp thông tin từ nhiều nguồn và sinh ra câu trả lời có tính mạch lạc và ngữ cảnh phù hợp. Có hai chiến lược chính để kết hợp thông tin: Fusion-in-Decoder (FiD) - nơi mỗi đoạn văn được truy xuất được xử lý riêng rẽ trong bộ mã hóa (encoder), và Fusion-in-Encoder (FiE) - nơi các đoạn văn được nối liền trước khi đưa vào bộ mã hóa.

- **Cơ sở tri thức (knowledge base)** là nơi lưu trữ các tài liệu, đoạn văn, hoặc tri thức có cấu trúc được sử dụng để tăng cường quá trình sinh nội dung. Cơ sở tri thức có thể là các văn bản từ Wikipedia, tài liệu nội bộ của công ty, cơ sở dữ liệu khách hàng, hoặc bất kỳ nguồn thông tin nào khác. Để hỗ trợ việc truy xuất hiệu quả, cơ sở tri thức thường được lập chỉ mục dưới dạng vector embeddings và lưu trữ trong các cơ sở dữ liệu vector như FAISS, Pinecone, hoặc Milvus. Việc chuẩn bị cơ sở tri thức bao gồm các bước như tải tài liệu, phân đoạn (chunking - chia tài liệu thành các đoạn nhỏ hơn), sinh embedding, và xây dựng chỉ mục.

Luồng hoạt động của một hệ thống RAG điển hình diễn ra theo các bước sau:

B1 Câu truy vấn của người dùng được nhận và tiền xử lý.

B2 Câu truy vấn được chuyển đổi thành vector embedding sử dụng cùng mô hình embedding đã được dùng để lập chỉ mục cơ sở tri thức.

B3 Cơ sở dữ liệu vector thực hiện tìm kiếm độ tương đồng để tìm top-k tài liệu liên quan.

B4 Các tài liệu được truy xuất được kết hợp với câu truy vấn gốc để tạo thành câu lệnh tăng cường.

B5 Câu lệnh tăng cường được đưa vào mô hình bộ sinh.

B6 Bộ sinh tạo ra phản hồi dựa trên cả câu truy vấn và ngữ cảnh được truy xuất.

B7 Phản hồi được hậu xử lý và trả về cho người dùng.

### 1.2.2 Ưu điểm và ứng dụng

RAG mang lại nhiều ưu điểm vượt trội so với việc chỉ sử dụng các mô hình ngôn ngữ lớn thuần túy:

- **Giảm thiểu ảo giác và tăng độ chính xác:** Đây là lợi ích quan trọng nhất của RAG. Bằng cách dựa câu trả lời trên các tài liệu thực tế được truy xuất từ cơ sở tri thức, RAG giúp giảm đáng kể khả năng mô hình tạo ra thông tin sai lệch hoặc không có cơ sở. Các nghiên cứu cho thấy RAG có thể cải thiện độ chính xác của câu trả lời lên đến 20-30% so với các mô hình ngôn ngữ lớn không có truy xuất. Điều này đặc biệt quan trọng trong các ứng dụng yêu cầu độ chính xác cao như y tế, pháp lý, hoặc tài chính.
- **Khả năng cập nhật kiến thức mà không cần tinh chỉnh:** Một trong những hạn chế lớn của các mô hình ngôn ngữ lớn truyền thống là kiến thức của chúng bị "đóng băng" tại thời điểm huấn luyện. Để cập nhật kiến thức, cần phải tinh chỉnh hoặc thậm chí huấn luyện trước lại mô hình - một quá trình cực kỳ tốn

kém. Với RAG, việc cập nhật kiến thức chỉ đơn giản là thêm tài liệu mới vào cơ sở tri thức và lập chỉ mục lại chúng. Điều này cho phép hệ thống luôn có thông tin mới nhất mà không cần động đến mô hình bộ sinh.

- **Tính minh bạch và khả năng giải thích:** RAG cung cấp khả năng truy vết nguồn gốc của thông tin trong câu trả lời. Vì các tài liệu được truy xuất được sử dụng để tạo câu trả lời, hệ thống có thể cung cấp trích dẫn hoặc tham chiếu cho người dùng, giúp họ kiểm chứng thông tin và hiểu nguồn gốc của kiến thức. Đây là một yếu tố quan trọng trong việc xây dựng lòng tin với người dùng, đặc biệt trong các ứng dụng doanh nghiệp hoặc học thuật.

- **Ứng dụng cho từng lĩnh vực cụ thể và doanh nghiệp:** RAG cho phép tùy chỉnh hệ thống cho các lĩnh vực cụ thể bằng cách sử dụng cơ sở tri thức đặc thù cho lĩnh vực đó. Ví dụ, một công ty có thể xây dựng hệ thống RAG sử dụng tài liệu nội bộ, chính sách, và thực hành tốt nhất của họ để tạo ra chatbot hỗ trợ nhân viên. Điều này không thể thực hiện hiệu quả với các mô hình ngôn ngữ lớn thuần túy vì chúng không có kiến thức về tài liệu riêng của công ty.

- **Hiệu quả về chi phí:** So với việc tinh chỉnh hoặc huấn luyện các mô hình ngôn ngữ lớn từ đầu, RAG là một giải pháp tiết kiệm chi phí hơn nhiều. Nó cho phép sử dụng các mô hình được huấn luyện trước sẵn có kết hợp với cơ chế truy xuất, giảm đáng kể chi phí tính toán và thời gian phát triển.

RAG đã được ứng dụng thành công trong nhiều lĩnh vực:

- **Hỏi đáp:** Đây là ứng dụng phổ biến nhất của RAG, nơi hệ thống cần trả lời các câu hỏi dựa trên một kho văn bản lớn. Ví dụ, các chatbot hỗ trợ khách hàng sử dụng RAG để trả lời câu hỏi của khách hàng dựa trên cơ sở tri thức về sản phẩm và dịch vụ.

- **Tóm tắt và tổng hợp tài liệu:** RAG có thể được sử dụng để tóm tắt thông tin từ nhiều tài liệu hoặc tổng hợp thông tin về một chủ đề cụ thể từ nhiều nguồn khác nhau. Điều này hữu ích trong nghiên cứu khoa học, báo chí, và trí tuệ doanh nghiệp.

- **AI hội thoại và trợ lý ảo:** RAG nâng cao khả năng của các chatbot và trợ lý ảo bằng cách cho phép chúng truy cập thông tin động từ cơ sở dữ liệu, API, hoặc tài liệu. Điều này giúp tạo ra các cuộc hội thoại có thông tin phong phú và chính xác hơn.

- **Tạo nội dung và hỗ trợ viết:** RAG có thể hỗ trợ việc viết bài báo, báo cáo, hoặc bài thuyết trình bằng cách truy xuất và tổng hợp thông tin liên quan từ nhiều nguồn. Đây chính là ứng dụng cốt lõi của đề tài này, tạo bài thuyết trình có ghi chú dựa trên RAG.

- **Sinh mã nguồn và tài liệu kỹ thuật:** RAG có thể được áp dụng để sinh mã nguồn hoặc tài liệu bằng cách truy xuất các ví dụ, tài liệu tham khảo API, hoặc thực hành tốt nhất từ cơ sở mã nguồn hoặc tài liệu kỹ thuật.

## 1.3 Kỹ thuật thiết kế câu lệnh và các kỹ thuật nâng cao

### 1.3.1 Giới thiệu về kỹ thuật thiết kế câu lệnh

**Kỹ thuật thiết kế câu lệnh (Prompt engineering)** là nghệ thuật và khoa học của việc thiết kế các câu lệnh đầu vào (prompts) để hướng dẫn các mô hình ngôn ngữ lớn tạo ra kết quả mong muốn. Trong bối cảnh các mô hình ngôn ngữ lớn ngày càng mạnh mẽ như GPT-4, Claude, hoặc Gemini, việc biết cách "nói chuyện" với các mô hình này trở thành một kỹ năng quan trọng. Kỹ thuật thiết kế câu lệnh không chỉ đơn giản là viết câu hỏi, mà còn bao gồm việc cấu trúc thông tin, cung cấp ngữ cảnh, thiết kế

ví dụ, và điều chỉnh giọng điệu/phong cách để đạt được kết quả tối ưu.

Tầm quan trọng của kỹ thuật thiết kế câu lệnh nằm ở chỗ nó cho phép khai thác tối đa khả năng của các mô hình ngôn ngữ lớn mà không cần tinh chỉnh hay huấn luyện lại, những quá trình rất tốn kém. Một câu lệnh được thiết kế tốt có thể cải thiện đáng kể chất lượng đầu ra, giảm hiện tượng ảo giác, và làm cho mô hình tuân thủ các ràng buộc cụ thể. Ngược lại, một câu lệnh kém có thể dẫn đến kết quả không liên quan, thiếu chính xác, hoặc không đáp ứng được yêu cầu.

Kỹ thuật thiết kế câu lệnh đã phát triển từ các kỹ thuật đơn giản đến các phương pháp phức tạp. Ban đầu, các câu lệnh thường là các câu hỏi trực tiếp hoặc hướng dẫn đơn giản. Tuy nhiên, với sự tiến bộ của các mô hình và hiểu biết sâu hơn về cách chúng hoạt động, các kỹ thuật nâng cao như học với ít ví dụ (few-shot learning), suy luận chuỗi tư duy (chain-of-thought reasoning), và tự nhất quán (self-consistency) đã được phát triển. Năm 2025 đánh dấu một bước chuyển mình quan trọng, khi kỹ thuật thiết kế câu lệnh không còn chỉ là việc cung cấp hướng dẫn mà trở thành việc tận dụng khả năng siêu nhận thức (metacognitive), suy nghĩ đa góc độ (multi-perspective thinking), và cải tiến đệ quy (recursive self-improvement) của các mô hình.

### 1.3.2 Zero-shot prompting

**Zero-shot prompting** là kỹ thuật cơ bản nhất, trong đó mô hình được yêu cầu thực hiện một nhiệm vụ mà không được cung cấp bất kỳ ví dụ nào. Mô hình chỉ dựa vào kiến thức đã học trong quá trình huấn luyện trước và khả năng hiểu hướng dẫn bằng ngôn ngữ tự nhiên. Ví dụ, một câu lệnh zero-shot có thể là: "Phân loại câu sau đây là tích cực hay tiêu cực: ’Sản phẩm này thật tuyệt vời!’" Mô hình sẽ cố gắng trả lời dựa trên hiểu biết chung về phân tích cảm xúc mà không cần ví dụ cụ thể.

Ưu điểm của zero-shot prompting là tính đơn giản và không cần chuẩn bị ví dụ. Nó đặc biệt hữu ích cho các nhiệm vụ đơn giản hoặc khi không có ví dụ có sẵn. Tuy

nhiên, hiệu quả của zero-shot prompting phụ thuộc rất nhiều vào cách hình thành câu lệnh và khả năng của mô hình trong việc tổng quát hóa từ hướng dẫn.

### 1.3.3 Few-shot prompting

**Few-shot prompting** cải thiện hiệu quả bằng cách cung cấp một số ít ví dụ (thường từ 1 đến 10) trong câu lệnh để mô hình có thể học mẫu và áp dụng cho đầu vào mới. Các ví dụ này được gọi là "minh họa" và chúng giúp mô hình hiểu rõ hơn về định dạng đầu ra mong muốn, phong cách, và logic cần áp dụng. Ví dụ của few-shot prompt:

"Phân loại cảm xúc của các câu sau:

Câu: ’Dịch vụ khách hàng rất tốt!’
Cảm xúc: Tích cực

Câu: ’Sản phẩm không đáng tiền.’
Cảm xúc: Tiêu cực

Câu: ’Giao hàng nhanh chóng và đúng hẹn.’
Cảm xúc: ?"

Trong ví dụ trên, mô hình được cung cấp hai ví dụ trước khi được yêu cầu phân loại câu thứ ba.

Các lợi ích chính của few-shot prompting bao gồm:

- **Quen thuộc với mẫu:** Các ví dụ phục vụ như mẫu mà mô hình có thể bắt chước.
- **Giảm sự mơ hồ:** Các ví dụ làm rõ kỳ vọng và giảm khả năng hiểu sai.
- **Tính linh hoạt:** Có thể dễ dàng thích ứng cho nhiều loại nhiệm vụ khác nhau.

- **Cải thiện độ chính xác:** Thường cho kết quả tốt hơn đáng kể so với zero-shot, đặc biệt với các nhiệm vụ phức tạp.

Các bước thực hành tốt nhất cho few-shot prompting bao gồm:

B1 Chọn các ví dụ đa dạng và đại diện cho nhiệm vụ.

B2 Đảm bảo các ví dụ có chất lượng cao và chính xác.

B3 Duy trì định dạng nhất quán giữa các ví dụ.

B4 Sử dụng các dấu phân cách rõ ràng để tách biệt các ví dụ khỏi đầu vào thực tế.

B5 Cung cấp hướng dẫn rõ ràng về định dạng đầu ra mong muốn.

### 1.3.4 Các kỹ thuật thiết kế câu lệnh nâng cao (tự tìm hiểu thêm)

## 1.4 Mô hình Embedding và cơ sở dữ liệu Vector

### 1.4.1 Sentence Transformers và mô hình đa ngôn ngữ

**Sentence Transformers** là một framework mạnh mẽ cho việc tạo các embedding câu, biểu diễn vector dày đặc của các câu có khả năng nắm bắt ý nghĩa ngữ nghĩa. Khác với các embedding từ truyền thống như Word2Vec hay GloVe chỉ đại diện cho từ đơn lẻ, các embedding câu đại diện cho toàn bộ câu hoặc đoạn văn trong một không gian vector mà các ý nghĩa tương tự tương ứng với các vector tương tự.

Sentence Transformers được xây dựng trên nền tảng của BERT và các mô hình transformer khác, nhưng được tinh chỉnh đặc biệt cho nhiệm vụ về độ tương đồng văn bản ngữ nghĩa. Framework sử dụng cấu trúc mạng siamese hoặc triplet để học các biểu diễn câu có ý nghĩa. Quá trình bao gồm:

- Các câu đầu vào được đưa qua mô hình transformer (như BERT) để sinh các embedding từ.
- Các embedding từ được tổng hợp thông qua chiến lược gộp (mean pooling, max pooling, hoặc token CLS) để tạo embedding câu có kích thước cố định.
- Các embedding kết quả có thể được so sánh trực tiếp sử dụng độ tương đồng cosine hoặc tích vô hướng.

Một trong những ưu điểm lớn của Sentence Transformers là sự sẵn có của nhiều mô hình được huấn luyện trước được tối ưu hóa cho các nhiệm vụ và ngôn ngữ khác nhau. Các mô hình này đã được huấn luyện trên các bộ dữ liệu quy mô lớn và có thể được sử dụng ngay mà không cần huấn luyện bổ sung cho nhiều ứng dụng.

**Mô hình đa ngôn ngữ (multilingual models)** đặc biệt quan trọng cho các ứng dụng cần hỗ trợ nhiều ngôn ngữ, như trong đề tài này với tiếng Việt. Các mô hình embedding câu đa ngôn ngữ được huấn luyện trên dữ liệu từ nhiều ngôn ngữ đồng thời, cho phép chúng ánh xạ các câu từ các ngôn ngữ khác nhau vào một không gian embedding chung. Điều này có nghĩa là các câu có ý nghĩa ngữ nghĩa tương tự trong các ngôn ngữ khác nhau sẽ có các embedding gần nhau trong không gian vector.

Một số mô hình đa ngôn ngữ phổ biến bao gồm:

- **distiluse-base-multilingual-cased** hỗ trợ hơn 12 ngôn ngữ bao gồm tiếng Ả Rập, Trung Quốc, Anh, Pháp, Đức, Ý, Hàn, Tây Ban Nha, và các ngôn ngữ khác. Mô hình này được chưng cất từ mô hình lớn hơn để đạt được sự cân bằng giữa hiệu suất và hiệu quả.
- **paraphrase-multilingual-MiniLM-L12-v2** là mô hình được sử dụng trong đề tài này, hỗ trợ hơn 50 ngôn ngữ và được tối ưu hóa cho các nhiệm vụ phát hiện câu diễn đạt lại và độ tương đồng ngữ nghĩa. Mô hình có 12 lớp và chiều

embedding là 384, cung cấp sự cân bằng tốt giữa chất lượng và hiệu quả tính toán.

- **paraphrase-multilingual-mpnet-base-v2** là mô hình lớn hơn với hiệu suất tốt hơn nhưng yêu cầu nhiều tài nguyên tính toán hơn. Nó đặc biệt tốt cho các nhiệm vụ yêu cầu độ chính xác cao trong khớp ngữ nghĩa.

Để sử dụng các mô hình đa ngôn ngữ cho độ tương đồng xuyên ngôn ngữ, quá trình rất đơn giản. Điểm tương đồng cosine gần bằng 1 cho thấy độ tương đồng ngữ nghĩa cao bất kể sự khác biệt về ngôn ngữ.

Các lợi ích chính của sentence transformers đa ngôn ngữ bao gồm:

- **Dễ sử dụng:** Sẵn sàng sử dụng ngay mà không cần huấn luyện.
- **Phạm vi ngôn ngữ rộng:** Hỗ trợ hàng chục ngôn ngữ.
- **Tính đa dụng:** Áp dụng được cho nhiều nhiệm vụ NLP như tìm kiếm ngữ nghĩa, phân cụm, phân loại.
- **Khả năng xuyên ngôn ngữ:** Cho phép các ứng dụng trải rộng nhiều ngôn ngữ.
- **Khả năng tinh chỉnh:** Có thể được tinh chỉnh thêm trên dữ liệu đặc thù lĩnh vực để cải thiện hiệu suất.

Đối với tiếng Việt cụ thể, nghiên cứu cho thấy các mô hình đa ngôn ngữ như paraphrase-multilingual-MiniLM-L12-v2 hoạt động khá tốt. Tuy nhiên, các mô hình chuyên biệt cho tiếng Việt như Vietnamese Document Embedding có thể đạt được kết quả thậm chí tốt hơn cho các ứng dụng thuần Việt. Đánh đổi là các mô hình chuyên biệt có thể không hỗ trợ các nhiệm vụ xuyên ngôn ngữ.

### 1.4.2 Cơ sở dữ liệu vector FAISS

**FAISS (Facebook AI Similarity Search)** là một thư viện mạnh mẽ được phát triển bởi Facebook AI Research cho việc tìm kiếm độ tương đồng và phân cụm hiệu quả các vector dày đặc. FAISS đặc biệt được thiết kế để xử lý các bộ dữ liệu vector quy mô lớn, có thể mở rộng đến hàng tỷ vector, với tốc độ tìm kiếm và hiệu quả bộ nhớ tối ưu.

Ý tưởng cốt lõi của FAISS là lập chỉ mục các vector theo cách cho phép tìm kiếm láng giềng gần nhất gần đúng hoặc chính xác nhanh chóng. Thay vì so sánh vector truy vấn với mọi vector trong cơ sở dữ liệu (tìm kiếm vét cạn - rất chậm với các bộ dữ liệu lớn), FAISS sử dụng các cấu trúc dữ liệu và thuật toán phức tạp để giảm đáng kể không gian tìm kiếm.

FAISS hỗ trợ nhiều thước đo độ tương đồng khác nhau:

- **Khoảng cách L2 (Euclidean):** Đo khoảng cách đường thẳng giữa các vector.
- **Tích vô hướng (dot product):** Hữu ích cho các vector được chuẩn hóa.
- **Độ tương đồng cosine:** Đo góc giữa các vector, thường được sử dụng cho độ tương đồng ngữ nghĩa. Đối với các ứng dụng tìm kiếm ngữ nghĩa như trong đề tài này, tích vô hướng hoặc độ tương đồng cosine thường được ưu tiên vì chúng nắm bắt tốt hơn các mối quan hệ ngữ nghĩa.

FAISS cung cấp nhiều loại chỉ mục với các đánh đổi khác nhau giữa tốc độ, độ chính xác, và sử dụng bộ nhớ:

- **IndexFlatL2 và IndexFlatIP** là các loại chỉ mục đơn giản nhất thực hiện tìm kiếm toàn diện. IndexFlatL2 sử dụng khoảng cách L2 trong khi IndexFlatIP sử dụng tích vô hướng. Đây là các phương pháp tìm kiếm chính xác, đảm bảo tìm thấy các láng giềng gần nhất thực sự, nhưng chậm với các bộ dữ liệu lớn vì phải so sánh với mọi vector. Tuy nhiên, chúng hữu ích như đường cơ sở và cho các bộ dữ liệu nhỏ hơn (dưới 1 triệu vector).

- **IndexIVFFlat (Inverted File Index)** phân vùng không gian vector thành các ô sử dụng phân cụm (thường là k-means), sau đó chỉ tìm kiếm trong các ô liên quan thay vì toàn bộ cơ sở dữ liệu. Điều này tăng tốc đáng kể việc tìm kiếm nhưng cho kết quả gần đúng. Các tham số chính bao gồm nlist (số lượng ô) và nprobe (số ô để tìm kiếm). Tăng nprobe cải thiện độ chính xác nhưng làm chậm tìm kiếm.

- **IndexHNSW (Hierarchical Navigable Small World)** sử dụng cách tiếp cận dựa trên đồ thị cho tìm kiếm gần đúng rất nhanh. HNSW xây dựng cấu trúc đồ thị nhiều lớp cho phép điều hướng hiệu quả đến các láng giềng gần nhất. Nó cung cấp sự đánh đổi tốc độ-độ chính xác xuất sắc và được sử dụng rộng rãi trong các hệ thống sản xuất.

**Các chỉ mục Product Quantization (PQ)** sử dụng các kỹ thuật nén để giảm đáng kể việc sử dụng bộ nhớ trong khi duy trì chất lượng tìm kiếm hợp lý. PQ mã hóa các vector sử dụng các sổ mã đã học, cho phép hàng tỷ vector vừa với bộ nhớ.

**Triển khai FAISS cho RAG** trong đề tài này bao gồm nhiều bước. Đối với các bộ dữ liệu lớn hơn, có thể sử dụng IndexIVFFlat để tăng tốc độ tìm kiếm trong khi vẫn duy trì độ chính xác hợp lý. Các bước thực hành tốt nhất cho việc sử dụng FAISS:

B1 **Chuẩn hóa embedding** khi sử dụng tích vô hướng để đạt được độ tương đồng cosine.

B2 **Sử dụng kiểu dữ liệu float32** vì FAISS được tối ưu hóa cho float32, không phải float64.

B3 **Huấn luyện chỉ mục** với tập con đại diện của dữ liệu cho các chỉ mục IVF và PQ.

B4 **Điều chỉnh nprobe** để cân bằng tốc độ và độ chính xác cho các chỉ mục IVF.

B5 **Xem xét tăng tốc GPU** cho các triển khai quy mô lớn - FAISS hỗ trợ các thao tác GPU.

B6 **Giám sát kích thước chỉ mục** và sử dụng bộ nhớ, xem xét các kỹ thuật nén nếu cần.

Tích hợp FAISS với quy trình công việc RAG cung cấp cơ chế truy xuất hiệu quả cần thiết cho các ứng dụng thời gian thực. Trong đề tài này, chỉ mục FAISS lưu trữ các embedding của tất cả các đoạn văn bản từ bộ dữ liệu tiếng Việt, cho phép tìm kiếm ngữ nghĩa nhanh khi sinh nội dung slide và ghi chú cho người nói.

## 1.5 Công nghệ hỗ trợ triển khai hệ thống

### 1.5.1 Python-pptx cho tự động hóa PowerPoint

**Python-pptx** là một thư viện Python cho phép tạo, đọc, và chỉnh sửa các file PowerPoint (.pptx) một cách lập trình. Thư viện này triển khai đặc tả định dạng file trình bày Open XML, cho phép các nhà phát triển thao tác các bài thuyết trình PowerPoint mà không cần cài đặt Microsoft Office.

Trong bối cảnh của đề tài, python-pptx được sử dụng để:

- Tạo cấu trúc bài thuyết trình với slide tiêu đề và các slide nội dung.
- Điền các tiêu đề slide và điểm đánh dấu được sinh bởi hệ thống RAG.
- Thêm ghi chú cho người nói chứa bài thuyết trình chi tiết cho người trình bày.
- Áp dụng định dạng nhất quán trên tất cả các slide.
- Xuất bài thuyết trình cuối cùng dưới dạng file .pptx sẵn sàng sử dụng.

### 1.5.2 Chuyển văn bản thành giọng nói cho tiếng Việt

**Công nghệ chuyển văn bản thành giọng nói (Text-to-Speech - TTS)** chuyển đổi văn bản viết thành âm thanh nói, cho phép máy móc "đọc to" nội dung. Đối với tiếng Việt, các hệ thống TTS phải xử lý các đặc điểm ngữ âm độc đáo của tiếng Việt bao gồm hệ thống thanh điệu, mũi hóa, và kho âm vị cụ thể.

Trong đề tài, TTS được áp dụng để tạo các file âm thanh MP3 cho ghi chú cho người nói của mỗi slide, cung cấp bài thuyết trình tự động hỗ trợ người trình bày. Điều này cho phép người trình bày nghe bài thuyết trình trước khi thực sự trình bày, hoặc sử dụng âm thanh như một hướng dẫn trong quá trình trình bày. Âm thanh cũng có thể được sử dụng cho các bài thuyết trình tự động hoàn toàn, nơi không có người trình bày trực tiếp.

### 1.5.3 Drupal CMS và Apache Superset

**Drupal** là một trong những hệ quản trị nội dung (CMS) mã nguồn mở phổ biến và mạnh mẽ nhất thế giới. Được phát triển từ năm 2001, Drupal hiện được sử dụng bởi hàng triệu trang web trên toàn cầu, từ các blog cá nhân nhỏ đến các nền tảng doanh nghiệp lớn và các trang web chính phủ. Drupal nổi tiếng với tính linh hoạt, khả năng mở rộng, và các tính năng bảo mật mạnh mẽ.

**Apache Superset** là nền tảng business intelligence và trực quan hóa dữ liệu mã nguồn mở hàng đầu. Ban đầu được phát triển tại Airbnb, Superset là dự án cấp cao nhất của Apache cung cấp các khả năng sẵn sàng doanh nghiệp mà không có chi phí cấp phép. Superset cung cấp một nền tảng hiện đại, có thể mở rộng cho việc khám phá dữ liệu và trực quan hóa.

Trong bối cảnh của đề tài, Drupal CMS và Apache Superset được dùng để tích hợp với nhật ký pipeline:

- Nhật ký CSV được tạo bởi module `bi_logging.py`.
- Cấu hình Superset để kết nối với cơ sở dữ liệu chứa dữ liệu nhật ký.
- Tạo bộ dữ liệu từ các bảng nhật ký.
- Xây dựng trực quan hóa theo dõi:
  - Thời gian thực thi pipeline theo giai đoạn.
  - Các chỉ số sinh bài thuyết trình.
  - Tỷ lệ lỗi và loại.
  - Sử dụng tài nguyên.
- Lắp ráp dashboard để giám sát sức khỏe hệ thống.

Lợi ích cho dự án:

- Khả năng hiển thị thời gian thực về hiệu suất hệ thống.
- Xác định các nút thắt cổ chai trong pipeline.
- Theo dõi các chỉ số chất lượng theo thời gian.
- Hỗ trợ các quyết định tối ưu hóa dựa trên dữ liệu.
- Chi phí cấp phép bằng không.
- Kiểm soát hoàn toàn việc triển khai.

Cùng nhau, Drupal và Superset cung cấp nền tảng mã nguồn mở mạnh mẽ cho việc triển khai và giám sát hệ thống sinh bài thuyết trình được hỗ trợ bởi AI. Drupal phục vụ như backend nội dung quản lý bài thuyết trình, người dùng, và metadata, trong khi Superset cung cấp khả năng giám sát và phân tích để đảm bảo hệ thống hoạt động tối ưu. Sự kết hợp này tạo ra một giải pháp end-to-end cho cả quản lý nội dung và trí tuệ doanh nghiệp.

# Chương 2

# Thiết kế xây dựng

Chương này trình bày chi tiết về kiến trúc tổng thể, thiết kế các thành phần chính, và quy trình triển khai hệ thống "Bài thuyết trình kèm ghi chú của người nói được tạo bởi AI (tự động thuyết minh)". Nội dung chương bao gồm kiến trúc hệ thống, thiết kế pipeline xử lý dữ liệu, cài đặt hệ thống RAG, thiết kế mẫu câu lệnh, và các module tự động hóa PowerPoint cùng chuyển văn bản thành giọng nói. Việc hiểu rõ thiết kế hệ thống là nền tảng để đánh giá tính khả thi và hiệu quả của giải pháp trong các phần tiếp theo.

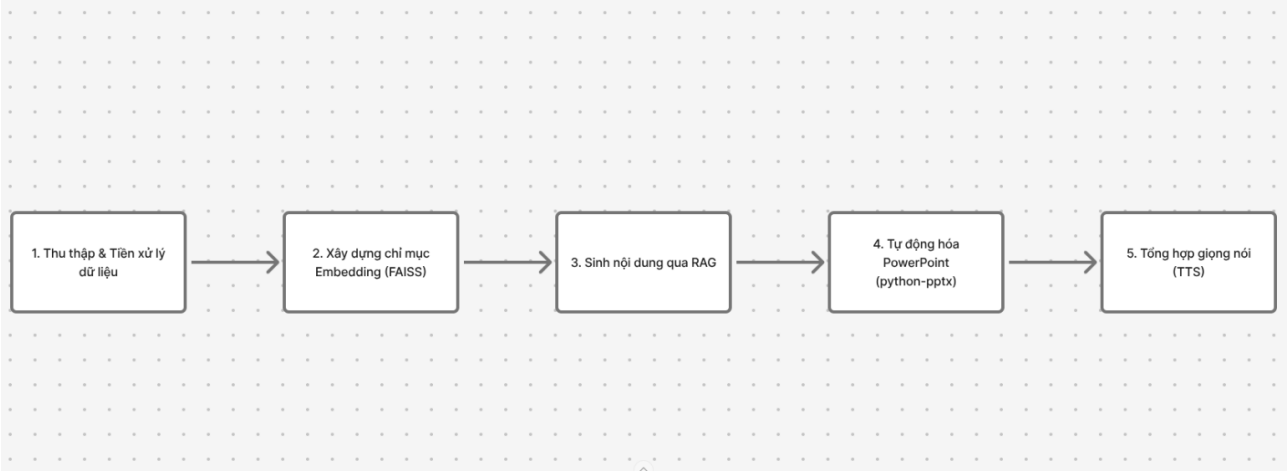
## 2.1 Kiến trúc tổng thể hệ thống

### 2.1.1 Tổng quan kiến trúc

Hệ thống được thiết kế theo kiến trúc pipeline đa giai đoạn (multi-stage pipeline), trong đó mỗi giai đoạn thực hiện một chức năng cụ thể và độc lập, đồng thời đầu ra của giai đoạn trước trở thành đầu vào cho giai đoạn tiếp theo. Kiến trúc này tuân theo các nguyên tắc thiết kế phần mềm hiện đại như tính modular (modularity), tách biệt mối quan tâm (separation of concerns), và khả năng mở rộng (scalability), đảm bảo

hệ thống dễ bảo trì, kiểm thử, và phát triển trong tương lai.

Pipeline tổng thể như hình 2.1 dưới đây bao gồm năm giai đoạn chính hoạt động tuần tự.




Hình 2.1: Sơ đồ kiến trúc pipeline mô tả năm giai đoạn thiết kế hệ thống

- **Giai đoạn thu thập và tiền xử lý dữ liệu** là bước đầu tiên, nơi hệ thống thu thập dữ liệu văn bản tiếng Việt từ nhiều nguồn khác nhau, sau đó thực hiện các thao tác làm sạch, chuẩn hóa, và phân đoạn để chuẩn bị dữ liệu cho các giai đoạn tiếp theo. Giai đoạn này đóng vai trò nền tảng cho chất lượng của toàn bộ hệ thống, vì dữ liệu đầu vào kém chất lượng sẽ dẫn đến kết quả đầu ra không đạt yêu cầu.

- **Giai đoạn xây dựng chỉ mục embedding** chuyển đổi các đoạn văn bản đã được xử lý thành các biểu diễn vector số học sử dụng mô hình embedding đa ngôn ngữ. Các vector này sau đó được lập chỉ mục trong cơ sở dữ liệu vector FAISS để cho phép tìm kiếm độ tương đồng ngữ nghĩa nhanh chóng. Giai đoạn này là trái tim của khả năng truy xuất thông tin của hệ thống RAG.

- **Giai đoạn sinh nội dung qua RAG** là giai đoạn cốt lõi, nơi hệ thống nhận câu truy vấn về chủ đề bài thuyết trình, thực hiện truy xuất các tài liệu liên quan từ

chỉ mục FAISS, và sử dụng các mẫu câu lệnh được thiết kế cẩn thận để sinh ra cấu trúc slide và ghi chú cho người nói. Cơ chế RAG đảm bảo nội dung được tạo ra dựa trên thông tin thực tế từ cơ sở tri thức, giảm thiểu hiện tượng ảo giác.

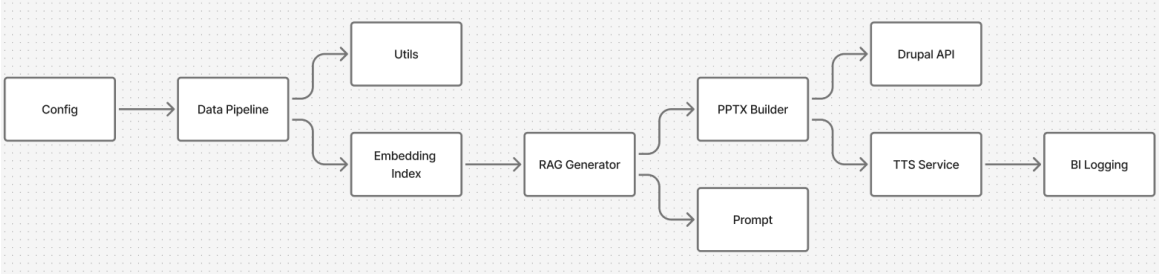
- **Giai đoạn tự động hóa PowerPoint** nhận cấu trúc nội dung từ giai đoạn trước và tạo ra file bài thuyết trình PowerPoint hoàn chỉnh với định dạng chuyên nghiệp. Module này sử dụng thư viện python-pptx để lập trình tạo các slide với tiêu đề, điểm đánh dấu, và ghi chú cho người nói một cách tự động.

- **Giai đoạn tổng hợp giọng nói** là bước cuối cùng, chuyển đổi ghi chú cho người nói từ văn bản thành các file âm thanh MP3 sử dụng công nghệ chuyển văn bản thành giọng nói. Các file âm thanh này có thể được sử dụng để hỗ trợ người trình bày hoặc tạo ra các bài thuyết trình tự động hoàn toàn.

Toàn bộ pipeline được điều phối bởi một module chính có nhiệm vụ khởi tạo cấu hình, gọi các giai đoạn theo đúng thứ tự, xử lý lỗi, và ghi nhận các chỉ số hiệu suất. Module điều phối này cũng tích hợp với hệ thống ghi nhật ký để theo dõi hoạt động của pipeline và hỗ trợ phân tích sau này.

### 2.1.2 Các thành phần chính và vai trò

Hệ thống như hình 2.2 dưới đây bao gồm các thành phần được thiết kế để thực hiện các chức năng cụ thể, có thể hoạt động độc lập và dễ dàng tái sử dụng.




Hình 2.2: Sơ đồ kiến trúc pipeline mô tả các thành phần của hệ thống

Các thành phần bao gồm:

- **Quản lý cấu hình (Config)** chịu trách nhiệm quản lý tất cả các tham số cấu hình của hệ thống như đường dẫn thư mục lưu trữ dữ liệu, đường dẫn thư mục đầu ra, số lượng mẫu tối thiểu trong bộ dữ liệu, mã ngôn ngữ xử lý, và tiêu đề mặc định cho bài thuyết trình. Thành phần này sử dụng cấu trúc dataclass của Python để định nghĩa các thuộc tính cấu hình một cách rõ ràng và có kiểu dữ liệu. Ngoài ra, thành phần này còn cung cấp phương thức để tự động tạo các thư mục cần thiết nếu chúng chưa tồn tại, đảm bảo hệ thống có thể khởi chạy thành công ngay cả khi chạy lần đầu tiên. Việc tập trung hóa quản lý cấu hình giúp dễ dàng điều chỉnh các tham số hệ thống mà không cần sửa đổi mã nguồn ở nhiều nơi.

- **Pipeline dữ liệu (Data Pipeline)** đảm nhiệm hai chức năng chính: thu thập dữ liệu và tiền xử lý dữ liệu. Về mặt thu thập dữ liệu, thành phần này cung cấp khả năng tải dữ liệu từ nguồn bên ngoài như bộ dữ liệu UIT ViNews thông qua API của Hugging Face, và cũng có khả năng sinh dữ liệu tổng hợp có kiểm soát khi dữ liệu thực không đủ hoặc không khả dụng. Cơ chế dự phòng (fallback) này đảm bảo hệ thống luôn có đủ dữ liệu để hoạt động. Về mặt tiền xử lý, thành phần này thực hiện chuẩn hóa văn bản tiếng Việt để loại bỏ các ký tự đặc biệt không cần thiết và thống nhất định dạng, sau đó phân đoạn văn bản thành các chunk có kích thước tối ưu cho việc truy xuất. Chiến lược phân đoạn được thiết kế để cân bằng giữa việc giữ nguyên ngữ cảnh và tạo ra các đơn vị truy xuất có kích thước phù hợp. Thành phần này cũng bao gồm lớp DatasetRecord để đại diện cho mỗi mẫu dữ liệu với các thuộc tính như chủ đề, mục đích, đối tượng khán giả, văn bản gốc, và các đoạn văn đã được phân đoạn.

- **Mẫu câu lệnh (Prompt Templates)** chứa logic tạo cấu trúc slide và sinh ghi chú cho người nói. Thành phần này cung cấp hàm tạo bản phác thảo slide với

cấu trúc chuẩn bao gồm các phần giới thiệu, khái niệm và nền tảng, chiến lược và triển khai, thách thức và giải pháp, kết luận và hướng phát triển. Mỗi slide trong phác thảo có tiêu đề và danh sách các điểm đánh dấu chính. Thành phần này cũng cung cấp hàm sinh ghi chú cho người nói dựa trên thông tin slide, các đoạn văn được truy xuất từ RAG, và đối tượng khán giả mục tiêu. Ghi chú được tạo ra có độ dài được kiểm soát từ 120 đến 180 từ, đảm bảo đủ chi tiết nhưng không quá dài. Logic sinh ghi chú kết hợp thông tin từ cấu trúc slide với nội dung được truy xuất để tạo ra các đoạn văn mạch lạc và có tính liên kết cao.

- **Xây dựng PowerPoint (PPTX Builder)** chịu trách nhiệm chuyển đổi cấu trúc nội dung thành file PowerPoint thực tế. Thành phần này sử dụng thư viện python-pptx để tạo đối tượng bài thuyết trình mới, thêm slide tiêu đề với thông tin chính, và tạo các slide nội dung với tiêu đề, điểm đánh dấu được định dạng, và ghi chú cho người nói. Thành phần này áp dụng các nguyên tắc định dạng nhất quán để đảm bảo bài thuyết trình có giao diện chuyên nghiệp. Việc tách biệt logic xây dựng PowerPoint thành module riêng cho phép dễ dàng mở rộng để hỗ trợ các định dạng khác hoặc tùy chỉnh mẫu thiết kế.

- **Dịch vụ chuyển văn bản thành giọng nói (TTS Service)** cung cấp chức năng chuyển đổi ghi chú văn bản thành âm thanh nói. Thành phần này lặp qua tất cả các slide, trích xuất ghi chú cho người nói, sử dụng công nghệ gTTS để tạo âm thanh, và lưu các file MP3 với tên tương ứng với số thứ tự slide. Thành phần này xử lý các vấn đề như chuẩn hóa văn bản trước khi chuyển đổi, quản lý file đầu ra, và xử lý lỗi khi dịch vụ TTS không khả dụng. Thiết kế thành phần này cho phép dễ dàng thay thế gTTS bằng các dịch vụ TTS khác nếu cần.

- **Ghi nhật ký trí tuệ doanh nghiệp (BI Logging)** thu thập và lưu trữ các chỉ số về hiệu suất và hoạt động của pipeline. Thành phần này cung cấp hàm ghi các metric vào file CSV với khả năng tự động tạo file và thêm tiêu đề cột nếu

file chưa tồn tại. Các metric được ghi nhận bao gồm thời gian thực thi từng giai đoạn, số lượng slide được tạo, độ dài trung bình của ghi chú, và các thông tin khác hữu ích cho việc phân tích và tối ưu hóa. Dữ liệu nhật ký được cấu trúc theo định dạng CSV để dễ dàng nhập vào Apache Superset cho việc trực quan hóa và phân tích.

- **Tiện ích (Utils)** cung cấp các hàm hỗ trợ chung được sử dụng xuyên suốt hệ thống. Thành phần này bao gồm hàm chuẩn hóa văn bản tiếng Việt để loại bỏ khoảng trắng thừa, dấu câu lặp, và các ký tự không cần thiết, hàm thiết lập seed ngẫu nhiên để đảm bảo tính tái lập (reproducibility) của các thí nghiệm, và các hàm tiện ích khác để xử lý chuỗi, file, và dữ liệu. Việc tập trung các hàm tiện ích vào một module giúp tránh lặp mã và dễ dàng bảo trì.

- **Tích hợp Drupal API** cung cấp các hàm để tương tác với Drupal CMS thông qua REST API hoặc JSON:API. Thành phần này bao gồm các hàm xác thực người dùng, tải lên bài thuyết trình đã tạo, truy xuất bài thuyết trình đã lưu, và cập nhật metadata. Thiết kế thành phần này cho phép hệ thống sinh bài thuyết trình tích hợp liền mạch với hệ thống quản lý nội dung Drupal, mở ra khả năng xây dựng ứng dụng web hoàn chỉnh.

### 2.1.3 Luồng dữ liệu và tương tác giữa các thành phần

Luồng dữ liệu trong hệ thống tuân theo mô hình pipeline tuyến tính với các điểm kiểm soát chất lượng và xử lý lỗi ở mỗi giai đoạn.

Khi hệ thống khởi động, module chính đầu tiên khởi tạo đối tượng cấu hình và đảm bảo tất cả các thư mục cần thiết tồn tại. Sau đó, pipeline dữ liệu được kích hoạt để xây dựng hoặc tải bộ dữ liệu. Pipeline dữ liệu đầu tiên cố gắng tải dữ liệu từ nguồn bên ngoài UIT ViNews thông qua API Hugging Face. Nếu việc tải thành công và số lượng mẫu đạt yêu cầu tối thiểu, dữ liệu sẽ được sử dụng. Nếu việc tải thất bại hoặc

số lượng mẫu không đủ, hệ thống tự động chuyển sang tạo dữ liệu tổng hợp. Dữ liệu sau khi thu thập được đi qua quá trình chuẩn hóa văn bản, loại bỏ các ký tự đặc biệt, thống nhất khoảng trắng, và chuẩn hóa dấu câu. Tiếp theo, văn bản được phân đoạn thành các chunk với kích thước tối ưu khoảng 350-400 ký tự, đảm bảo mỗi chunk chứa một lượng thông tin vừa đủ và không bị cắt giữa câu. Cuối cùng, toàn bộ bộ dữ liệu được lưu vào file JSONL với định dạng chuẩn, mỗi dòng là một đối tượng JSON chứa thông tin về chủ đề, mục đích, đối tượng, văn bản gốc, và các đoạn văn đã phân đoạn.

Sau khi bộ dữ liệu sẵn sàng, module chỉ mục embedding được khởi tạo và thực hiện quá trình xây dựng chỉ mục. Module đọc file JSONL, trích xuất tất cả các chunk cùng với metadata của chúng, tải mô hình Sentence Transformer paraphrase-multilingual-MiniLM-L12-v2 từ thư viện Hugging Face, và sử dụng mô hình để chuyển đổi mỗi chunk thành vector embedding 384 chiều. Các vector được chuẩn hóa để có độ dài đơn vị, sau đó được thêm vào chỉ mục FAISS sử dụng IndexFlatIP cho tìm kiếm tích vô hướng. Metadata của mỗi chunk được lưu trữ trong danh sách song song với chỉ mục vector để có thể truy xuất thông tin đầy đủ khi cần. Sau khi hoàn thành, chỉ mục sẵn sàng để phục vụ các truy vấn.

Sau khi bộ dữ liệu sẵn sàng, module chỉ mục embedding được khởi tạo và thực hiện quá trình xây dựng chỉ mục. Module đọc file JSONL, trích xuất tất cả các chunk cùng với metadata của chúng, tải mô hình Sentence Transformer paraphrase-multilingual-MiniLM-L12-v2 từ thư viện Hugging Face, và sử dụng mô hình để chuyển đổi mỗi chunk thành vector embedding 384 chiều. Các vector được chuẩn hóa để có độ dài đơn vị, sau đó được thêm vào chỉ mục FAISS sử dụng IndexFlatIP cho tìm kiếm tích vô hướng. Metadata của mỗi chunk được lưu trữ trong danh sách song song với chỉ mục vector để có thể truy xuất thông tin đầy đủ khi cần. Sau khi hoàn thành, chỉ mục sẵn sàng để phục vụ các truy vấn.

Khi người dùng yêu cầu tạo bài thuyết trình với chủ đề, mục đích, và đối tượng cụ thể, module sinh nội dung RAG được gọi. Module này đầu tiên tạo câu truy vấn bằng cách kết hợp các tham số đầu vào thành một chuỗi văn bản đại diện cho yêu cầu. Câu truy vấn được chuyển thành vector embedding sử dụng cùng mô hình Sentence Transformer đã dùng để tạo chỉ mục. Vector truy vấn được so sánh với tất cả các vector trong chỉ mục FAISS để tìm top-k đoạn văn có độ tương đồng cao nhất, thường k được đặt là 10. Các đoạn văn được truy xuất cùng với điểm tương đồng và metadata được trả về. Tiếp theo, module sử dụng hàm tạo phác thảo slide để sinh cấu trúc cơ bản với số lượng slide mong muốn. Đối với mỗi slide trong phác thảo, module gọi hàm sinh ghi chú cho người nói, truyền vào tiêu đề slide, các điểm đánh dấu, các đoạn văn RAG đã truy xuất, và đối tượng khán giả. Hàm sinh ghi chú kết hợp thông tin từ slide với nội dung được truy xuất để tạo ra đoạn văn mạch lạc với giọng điệu phù hợp với đối tượng. Kết quả là một danh sách các slide hoàn chỉnh, mỗi slide có tiêu đề, điểm đánh dấu, và ghi chú chi tiết.

Danh sách slide sau đó được chuyển đến module xây dựng PowerPoint. Module tạo đối tượng Presentation mới, thiết lập kích thước slide tiêu chuẩn, thêm slide tiêu đề với tên bài thuyết trình, và lặp qua danh sách slide để tạo các slide nội dung. Đối với mỗi slide, module chọn layout phù hợp có vùng tiêu đề và nội dung, điền tiêu đề vào vùng tiêu đề, tạo khung văn bản cho các điểm đánh dấu và định dạng chúng dưới dạng danh sách, và thêm ghi chú cho người nói vào vùng ghi chú của slide. Sau khi tất cả slide được tạo, file PowerPoint được lưu vào đường dẫn đầu ra đã chỉ định.

Đồng thời hoặc sau khi tạo PowerPoint, module TTS được kích hoạt để tạo âm thanh. Module lặp qua danh sách slide, trích xuất ghi chú cho người nói từ mỗi slide, thực hiện chuẩn hóa văn bản nhẹ để loại bỏ các ký tự có thể gây lỗi TTS, sử dụng gTTS để chuyển văn bản thành âm thanh với mã ngôn ngữ tiếng Việt, và lưu file MP3 với tên chứa số thứ tự slide. Tất cả các file âm thanh được lưu trong một thư mục audio chuyên dụng để dễ quản lý.

Xuyên suốt quá trình thực thi, module ghi nhật ký liên tục thu thập các chỉ số hiệu suất. Thời điểm bắt đầu và kết thúc của mỗi giai đoạn được ghi nhận để tính toán thời gian thực thi. Các metric được tổ chức thành hai loại: metric về thời gian được lưu vào file `pipeline_times.csv` chứa thời gian cho từng giai đoạn và tổng thời gian, metric về thống kê bài thuyết trình được lưu vào file `presentation_stats.csv` chứa thông tin về chủ đề, số slide, độ dài trung bình của ghi chú. Dữ liệu nhật ký này có thể được nhập vào Apache Superset để tạo dashboard theo dõi hiệu suất hệ thống theo thời gian.

Tất cả các tương tác giữa các thành phần được thiết kế để tối thiểu hóa sự phụ thuộc chặt chẽ. Mỗi module chỉ phụ thuộc vào interface công khai của module khác, không truy cập trực tiếp vào trạng thái nội bộ. Điều này cho phép thay thế hoặc nâng cấp một module mà không ảnh hưởng đến các module khác, miễn là interface được duy trì. Ví dụ, có thể thay thế gTTS bằng một dịch vụ TTS khác chỉ bằng cách sửa đổi module TTS Service mà không cần thay đổi bất kỳ module nào khác trong hệ thống.

## 2.2 Thu thập và xử lý dữ liệu

### 2.2.1 Nguồn dữ liệu và phương pháp thu thập

Dữ liệu là yếu tố nền tảng quyết định chất lượng của hệ thống RAG. Để đảm bảo hệ thống có thể tạo ra các bài thuyết trình chất lượng cao với nội dung phong phú và đa dạng, chiến lược thu thập dữ liệu của đề tài kết hợp cả dữ liệu thực từ nguồn công khai và dữ liệu tổng hợp có kiểm soát.

**Nguồn dữ liệu chính là bộ dữ liệu UIT ViNews,** một bộ dữ liệu tin tức tiếng Việt quy mô lớn được phát triển bởi Đại học Công nghệ Thông tin - Đại học Quốc gia Thành phố Hồ Chí Minh. Bộ dữ liệu này chứa hàng ngàn bài báo tin tức tiếng Việt từ nhiều lĩnh vực khác nhau như kinh tế, chính trị, xã hội, công nghệ, giải trí, và thể

thao. Đây là một nguồn dữ liệu có chất lượng cao với văn phong chuẩn mực, cấu trúc tốt, và nội dung phong phú, rất phù hợp cho việc huấn luyện các hệ thống xử lý ngôn ngữ tự nhiên tiếng Việt. Bộ dữ liệu được công khai trên nền tảng Hugging Face, cho phép truy cập thông qua API một cách thuận tiện.

Quy trình thu thập dữ liệu từ UIT ViNews được triển khai thông qua hàm `download_uits_vienews` trong module `data_pipeline`. Hàm này gửi yêu cầu HTTP GET đến API Datasets Server của Hugging Face với các tham số chỉ định bộ dữ liệu cần tải, cấu hình mặc định, phân vùng huấn luyện (train split), và số lượng mẫu cần lấy. API trả về dữ liệu dưới định dạng JSON chứa danh sách các bản ghi, mỗi bản ghi chứa văn bản bài báo và các metadata liên quan. Hàm phân tích cú pháp phản hồi JSON, trích xuất trường văn bản từ mỗi bản ghi, thực hiện chuẩn hóa văn bản, suy luận heuristic để gán các thuộc tính như chủ đề, mục đích, và đối tượng khán giả, sau đó phân đoạn văn bản thành các chunk phù hợp cho truy xuất. Các bản ghi hoàn chỉnh được tạo dưới dạng đối tượng DatasetRecord và thêm vào danh sách kết quả.

Quy trình bao gồm xử lý lỗi toàn diện để đối phó với các tình huống bất thường. Nếu kết nối mạng thất bại, API không phản hồi trong thời gian chờ 30 giây, hoặc API trả về mã lỗi HTTP, hàm bắt exception và trả về danh sách rỗng để báo hiệu cần sử dụng cơ chế dự phòng. Thiết kế này đảm bảo hệ thống có thể hoạt động ngay cả khi không có kết nối internet hoặc dịch vụ Hugging Face tạm thời không khả dụng.

**Cơ chế dự phòng sử dụng dữ liệu tổng hợp** được kích hoạt khi dữ liệu thực không đủ hoặc không khả dụng. Hàm `synthesize_samples` tạo ra các mẫu dữ liệu giả lập một cách có kiểm soát bằng cách kết hợp các yếu tố có cấu trúc. Hàm định nghĩa các danh sách chủ đề phổ biến trong lĩnh vực công nghệ và kinh doanh bao gồm Chuyển đổi số, Khởi nghiệp, AI trong giáo dục, Marketing số, Thương mại điện tử, và Phân tích dữ liệu. Các mục đích khác nhau được định nghĩa như giảng dạy, thuyết minh, và bán hàng. Các đối tượng khán giả mục tiêu bao gồm sinh viên, quản lý, và

công chúng. Hàm sử dụng vòng lặp để tạo số lượng mẫu mong muốn, với mỗi mẫu được gán chủ đề, mục đích, và đối tượng theo phương pháp xoay vòng để đảm bảo sự phân bố đều. Văn bản cho mỗi mẫu được tạo bằng cách sử dụng mẫu câu với các vị trí giữ chỗ được điền bởi các giá trị cụ thể, tạo ra các đoạn văn có cấu trúc tương tự nhưng nội dung khác nhau. Mặc dù dữ liệu tổng hợp không có độ phong phú bằng dữ liệu thực, nhưng nó đảm bảo hệ thống có đủ dữ liệu để hoạt động và phục vụ mục đích minh họa và kiểm thử.

Hàm `build_dataset` là điểm vào chính cho quá trình xây dựng bộ dữ liệu. Hàm này điều phối logic thu thập dữ liệu với cơ chế dự phòng. Đầu tiên, hàm cố gắng tải dữ liệu thực từ UIT ViNews với giới hạn số lượng cao hơn mức tối thiểu yêu cầu để có buffer. Nếu số lượng bản ghi nhận được ít hơn mức tối thiểu, hàm chuyển sang tạo dữ liệu tổng hợp. Sau khi có đủ dữ liệu, toàn bộ bộ dữ liệu được lưu vào file JSONL với mã hóa UTF-8 để đảm bảo xử lý đúng các ký tự tiếng Việt. Mỗi bản ghi được chuyển thành đối tượng JSON chứa tất cả các thuộc tính quan trọng và được ghi trên một dòng riêng biệt. Định dạng JSONL được chọn vì tính đơn giản, khả năng xử lý streaming, và khả năng tương thích rộng rãi với các công cụ xử lý dữ liệu.

### 2.2.2 Đặc điểm và cấu trúc của bộ dữ liệu

Bộ dữ liệu được xây dựng có các đặc điểm quan trọng đảm bảo tính phù hợp cho hệ thống RAG và khả năng tạo các bài thuyết trình chất lượng.

**Về quy mô,** bộ dữ liệu được thiết kế để chứa tối thiểu 100 mẫu, với mục tiêu lý tưởng là 120-150 mẫu. Con số này được xác định dựa trên cân nhắc giữa yêu cầu kỹ thuật của đề tài môn học và yêu cầu thực tế để hệ thống RAG hoạt động hiệu quả. Với 100+ mẫu, sau khi phân đoạn sẽ tạo ra hàng trăm chunk văn bản, cung cấp đủ đa dạng để hệ thống có thể truy xuất thông tin liên quan cho nhiều loại câu truy vấn khác nhau. Quy mô này cũng phù hợp với khả năng xử lý của mô hình embedding và FAISS trên

phần cứng tiêu chuẩn, đảm bảo thời gian xây dựng chỉ mục và truy vấn vẫn ở mức chấp nhận được.

**Về độ đa dạng nội dung,** bộ dữ liệu bao phủ nhiều lĩnh vực khác nhau trong công nghệ và kinh doanh. Nếu sử dụng dữ liệu từ UIT ViNews, sự đa dạng đến từ việc bộ dữ liệu này chứa tin tức từ nhiều chuyên mục và nguồn khác nhau. Nếu sử dụng dữ liệu tổng hợp, sự đa dạng được tạo ra một cách có chủ đích thông qua việc biến đổi các chủ đề, mục đích, và đối tượng khán giả. Sự đa dạng này rất quan trọng vì nó cho phép hệ thống xử lý các yêu cầu tạo bài thuyết trình về nhiều chủ đề khác nhau và phục vụ nhiều đối tượng khác nhau. **Về cấu trúc dữ liệu,** mỗi mẫu trong bộ dữ liệu được đại diện bằng lớp DatasetRecord với các thuộc tính cụ thể. Thuộc tính topic chứa chủ đề chính của nội dung, ví dụ như "Chuyển đổi số" hoặc "Khởi nghiệp". Thuộc tính intent mô tả mục đích sử dụng nội dung, có thể là "giảng dạy" khi nội dung dùng cho môi trường học thuật, "thuyết minh" khi dùng để trình bày thông tin, hoặc "bán hàng" khi dùng cho mục đích marketing. Thuộc tính audience xác định đối tượng mục tiêu như "sinh viên", "quản lý", hoặc "công chúng", cho phép hệ thống điều chỉnh giọng điệu và mức độ kỹ thuật phù hợp. Thuộc tính text chứa văn bản gốc đầy đủ của tài liệu. Thuộc tính chunks là danh sách các đoạn văn đã được phân đoạn, mỗi chunk có kích thước tối ưu cho việc truy xuất.

**Về chất lượng văn bản,** tất cả dữ liệu đều trải qua quá trình chuẩn hóa nghiêm ngặt. Hàm `normalize_vi_text` trong module utils thực hiện nhiều thao tác để đảm bảo văn bản có định dạng nhất quán. Các khoảng trắng thừa được loại bỏ, bao gồm khoảng trắng ở đầu và cuối chuỗi cũng như nhiều khoảng trắng liên tiếp trong văn bản. Dấu câu lặp lại bất thường như nhiều dấu chấm hoặc dấu phẩy liên tiếp được chuẩn hóa thành một ký tự duy nhất. Các ký tự điều khiển đặc biệt và ký tự Unicode không in được được loại bỏ. Kết quả là văn bản sạch, dễ xử lý, và không chứa các yếu tố có thể gây lỗi trong các giai đoạn xử lý sau.

**Về phương pháp phân đoạn,** chiến lược được thiết kế để cân bằng giữa việc duy trì ngữ cảnh và tạo ra các đơn vị truy xuất có kích thước phù hợp. Hàm `chunk_text` sử dụng phương pháp phân đoạn dựa trên câu với giới hạn kích thước linh hoạt. Văn bản đầu tiên được tách thành các câu dựa trên dấu chấm. Sau đó, các câu được nhóm lại thành các chunk sao cho mỗi chunk không vượt quá độ dài tối đa khoảng 350-400 ký tự. Nếu thêm một câu mới vào chunk hiện tại sẽ vượt quá giới hạn, chunk hiện tại được hoàn thiện và câu mới bắt đầu một chunk mới. Phương pháp này đảm bảo rằng ranh giới chunk luôn là ranh giới câu, tránh việc cắt giữa câu có thể làm mất ngữ cảnh. Kích thước chunk 350-400 ký tự được chọn dựa trên nghiên cứu về độ dài tối ưu cho truy xuất - đủ dài để chứa một ý tưởng hoàn chỉnh nhưng đủ ngắn để có độ chính xác cao trong matching.

**Về metadata,** mỗi chunk được lưu trữ cùng với metadata phong phú cho phép truy xuất có ngữ cảnh. Ngoài chính văn bản của chunk, metadata bao gồm chủ đề, mục đích, và đối tượng của tài liệu mẹ. Thông tin này cho phép hệ thống không chỉ tìm các chunk có nội dung tương tự mà còn lọc dựa trên các tiêu chí khác như đối tượng phù hợp. Ví dụ, khi tạo bài thuyết trình cho sinh viên, hệ thống có thể ưu tiên các chunk từ tài liệu được đánh dấu cho đối tượng sinh viên.

**Về định dạng lưu trữ,** bộ dữ liệu được lưu dưới dạng file JSONL (JSON Lines) với tên `viet_presentation_dataset.jsonl`. Mỗi dòng trong file là một đối tượng JSON hợp lệ độc lập, chứa tất cả thông tin của một mẫu. Định dạng này có nhiều ưu điểm: đơn giản để phân tích cú pháp, hỗ trợ xử lý streaming cho phép đọc và xử lý từng dòng mà không cần tải toàn bộ file vào bộ nhớ, dễ dàng thêm hoặc xóa các mẫu bằng cách chỉnh sửa các dòng tương ứng, và tương thích rộng rãi với nhiều công cụ và ngôn ngữ lập trình. File được mã hóa UTF-8 để hỗ trợ đầy đủ các ký tự tiếng Việt bao gồm dấu thanh và các ký tự đặc biệt.

### 2.2.3 Quy trình tiền xử lý dữ liệu và đảm bảo chất lượng

Quá trình tiền xử lý dữ liệu là giai đoạn quan trọng quyết định chất lượng của toàn bộ hệ thống. Các bước tiền xử lý được thiết kế để chuyển đổi dữ liệu thô thành định dạng tối ưu cho embedding và truy xuất.

**Giai đoạn làm sạch ban đầu** xử lý dữ liệu ngay sau khi được tải về hoặc tạo ra. Văn bản đầu tiên được kiểm tra để loại bỏ các mẫu rỗng hoặc quá ngắn không có giá trị thông tin. Các ký tự điều khiển đặc biệt như xuống dòng, tab, và các ký tự Unicode điều khiển được thay thế bằng khoảng trắng hoặc loại bỏ hoàn toàn. Dấu câu lặp lại bất thường được chuẩn hóa - ví dụ, ba dấu chấm liên tiếp được thay thế bằng dấu ba chấm tiêu chuẩn, nhiều dấu phẩy được giảm xuống một dấu. Khoảng trắng được chuẩn hóa bằng cách thay thế nhiều khoảng trắng liên tiếp bằng một khoảng trắng duy nhất và loại bỏ khoảng trắng ở đầu và cuối văn bản.

**Giai đoạn chuẩn hóa văn bản tiếng Việt** áp dụng các quy tắc cụ thể cho tiếng Việt. Các dấu thanh được kiểm tra để đảm bảo chúng được mã hóa đúng theo Unicode Normalization Form C (NFC), chuẩn được khuyến nghị cho tiếng Việt. Các từ viết tắt phổ biến được mở rộng để tránh sự mơ hồ - ví dụ, "TP" có thể được mở rộng thành "Thành phố". Số được giữ nguyên nhưng được đảm bảo có khoảng trắng phù hợp xung quanh. Các từ tiếng Anh hoặc thuật ngữ kỹ thuật được giữ nguyên và không chuẩn hóa để bảo toàn ý nghĩa gốc.

**Giai đoạn phân đoạn văn bản** là bước quan trọng nhất trong tiền xử lý cho hệ thống RAG. Phân đoạn tốt có tác động trực tiếp đến chất lượng truy xuất. Chiến lược phân đoạn được sử dụng là phân đoạn dựa trên câu với cửa sổ trượt mềm. Văn bản được chia thành câu dựa trên dấu chấm, dấu chấm hỏi, và dấu chấm than. Các câu được nhóm lại để tạo thành các chunk với kích thước mục tiêu 350-400 ký tự. Nếu một câu đơn lẻ dài hơn giới hạn, nó vẫn được giữ nguyên như một chunk riêng để không phá vỡ cấu trúc câu. Ngược lại, các câu ngắn được kết hợp để đạt được kích thước chunk

lý tưởng. Mỗi chunk sau khi được tạo ra đều trải qua một lần chuẩn hóa cuối cùng để đảm bảo định dạng nhất quán.

**Kiểm tra chất lượng dữ liệu** được thực hiện ở nhiều điểm trong quy trình. Sau khi thu thập dữ liệu, hệ thống kiểm tra xem số lượng mẫu có đạt mức tối thiểu yêu cầu hay không. Nếu không đủ, cơ chế dự phòng được kích hoạt. Sau khi phân đoạn, hệ thống kiểm tra để đảm bảo mỗi mẫu có ít nhất một chunk hợp lệ. Các mẫu không có chunk được loại bỏ khỏi bộ dữ liệu. Trước khi xây dựng chỉ mục embedding, hệ thống xác minh rằng tất cả các chunk đều là chuỗi văn bản không rỗng và không chứa chỉ khoảng trắng. Các kiểm tra này đảm bảo chỉ dữ liệu chất lượng cao được đưa vào các giai đoạn xử lý tiếp theo.

**Xử lý lỗi và phục hồi** được tích hợp vào mọi giai đoạn của pipeline dữ liệu. Nếu việc tải dữ liệu từ nguồn bên ngoài thất bại do lỗi mạng, hết thời gian chờ, hoặc lỗi API, hệ thống không crash mà tự động chuyển sang sử dụng dữ liệu tổng hợp. Nếu một mẫu cụ thể gặp lỗi trong quá trình xử lý do định dạng không hợp lệ hoặc nội dung bất thường, mẫu đó được bỏ qua và quá trình tiếp tục với các mẫu khác. Nếu quá nhiều mẫu bị lỗi khiến tổng số mẫu hợp lệ giảm xuống dưới ngưỡng tối thiểu, hệ thống tạo thêm dữ liệu tổng hợp để bổ sung. Thiết kế chịu lỗi này đảm bảo pipeline dữ liệu có độ tin cậy cao và có thể hoạt động trong nhiều điều kiện khác nhau.

**Tối ưu hóa hiệu suất** cũng được xem xét trong thiết kế pipeline dữ liệu. Việc đọc và ghi file sử dụng xử lý streaming để tránh tải toàn bộ dữ liệu vào bộ nhớ cùng một lúc. Các thao tác xử lý chuỗi được tối ưu hóa để tránh tạo quá nhiều đối tượng chuỗi trung gian. Dữ liệu được lưu ở định dạng JSONL cho phép xử lý tăng dần thay vì phải đợi xử lý xong toàn bộ bộ dữ liệu mới bắt đầu giai đoạn tiếp theo.

## 2.3 Triển khai hệ thống RAG

### 2.3.1 Lựa chọn và cấu hình mô hình embedding

Mô hình embedding là thành phần cốt lõi quyết định khả năng nắm bắt ý nghĩa ngữ nghĩa và chất lượng truy xuất của hệ thống RAG. Sau khi nghiên cứu và so sánh nhiều mô hình khác nhau, paraphrase-multilingual-MiniLM-L12-v2 được lựa chọn là mô hình chính cho đề tài này.

**Lý do lựa chọn mô hình** dựa trên nhiều tiêu chí quan trọng. Về mặt hỗ trợ ngôn ngữ, mô hình này hỗ trợ hơn 50 ngôn ngữ bao gồm tiếng Việt, cho phép tạo các embedding chất lượng cao cho văn bản tiếng Việt mà không cần huấn luyện lại. Về mặt kiến trúc, mô hình có 12 lớp transformer và tạo ra các embedding 384 chiều, cung cấp sự cân bằng tốt giữa khả năng biểu diễn và hiệu quả tính toán. Về mặt hiệu suất, mô hình đã được huấn luyện đặc biệt cho nhiệm vụ phát hiện câu diễn đạt lại và độ tương đồng ngữ nghĩa, phù hợp hoàn hảo với yêu cầu của hệ thống RAG. Về mặt tài nguyên, mô hình có kích thước tương đối nhỏ khoảng 420MB, có thể chạy hiệu quả trên CPU mà không cần GPU chuyên dụng, làm cho nó phù hợp với môi trường triển khai học thuật.

**Cấu hình mô hình** được thực hiện thông qua thư viện Sentence Transformers. Mô hình được khởi tạo với tên đầy đủ sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 từ Hub của Hugging Face. Khi mô hình được tải lần đầu tiên, nó được tải xuống từ Hugging Face và lưu cache cục bộ cho các lần sử dụng sau. Quá trình encoding được cấu hình với các tham số quan trọng: `convert_to_numpy = True` để chuyển đổi kết quả sang mảng NumPy phù hợp với FAISS, `normalize_embeddings = True` để chuẩn hóa các vector về độ dài đơn vị cho phép sử dụng tích vô hướng như độ tương đồng cosine, và `batch_size` được để mặc định để mô hình tự động tối ưu dựa trên tài nguyên có sẵn.

**Quá trình tạo embedding** diễn ra trong hai giai đoạn chính. Giai đoạn đầu là embedding toàn bộ bộ dữ liệu để xây dựng chỉ mục. Tất cả các chunk văn bản được trích xuất từ file JSONL và tập hợp thành một danh sách. Danh sách này được đưa vào mô hình để tạo embedding trong một hoặc nhiều batch tùy thuộc vào kích thước. Mô hình xử lý từng chunk bằng cách tokenize văn bản thành các sub-token, đưa các token qua 12 lớp transformer để tạo embedding ngữ cảnh cho mỗi token, áp dụng mean pooling để tổng hợp các embedding token thành một embedding câu duy nhất, và chuẩn hóa vector kết quả về độ dài đơn vị. Kết quả là một ma trận NumPy có kích thước (`n_chunks`, 384) trong đó mỗi hàng là một vector embedding 384 chiều đại diện cho một chunk. Giai đoạn thứ hai là embedding câu truy vấn trong thời gian truy xuất. Khi người dùng yêu cầu tạo bài thuyết trình với các tham số cụ thể, câu truy vấn được tạo bằng cách kết hợp chủ đề, mục đích, và đối tượng thành một chuỗi văn bản. Chuỗi này được đưa qua cùng mô hình embedding để tạo vector truy vấn 384 chiều. Việc sử dụng cùng mô hình và cấu hình cho cả indexing và querying đảm bảo các vector nằm trong cùng không gian vector và có thể so sánh trực tiếp.

**Tối ưu hóa hiệu suất** được áp dụng ở nhiều mức độ. Mô hình được tải một lần duy nhất và tái sử dụng cho tất cả các thao tác encoding thay vì tải lại cho mỗi lần sử dụng. Encoding được thực hiện theo batch thay vì từng chunk riêng lẻ để tận dụng song song hóa và giảm overhead. Cache của Hugging Face được tận dụng để tránh tải mô hình từ internet mỗi lần chạy chương trình. Đối với các triển khai sản xuất lớn hơn, có thể cân nhắc sử dụng quantization để giảm kích thước mô hình hoặc sử dụng GPU để tăng tốc độ encoding.

### 2.3.2 Xây dựng chỉ mục FAISS

FAISS (Facebook AI Similarity Search) được lựa chọn là công cụ quản lý và truy vấn các vector embedding do hiệu quả cao và tính dễ sử dụng. Quá trình xây dựng chỉ mục FAISS được triển khai trong phương thức build của lớp EmbeddingIndex.

**Kiến trúc chỉ mục** được thiết kế phù hợp với quy mô và yêu cầu của hệ thống. IndexFlatIP (Inner Product) được chọn làm loại chỉ mục chính vì nhiều lý do. Đầu tiên, IndexFlatIP thực hiện tìm kiếm chính xác (exact search) bằng cách so sánh vector truy vấn với mọi vector trong chỉ mục, đảm bảo kết quả truy xuất chính xác nhất có thể. Thứ hai, inner product (tích vô hướng) kết hợp với vector đã chuẩn hóa tương đương với độ tương đồng cosine, một độ đo phổ biến và hiệu quả cho so sánh ngữ nghĩa. Thứ ba, với quy mô bộ dữ liệu khoảng 100-150 mẫu tương ứng với vài trăm đến vài nghìn chunk, tìm kiếm chính xác vẫn rất nhanh (vài milliseconds) trên phần cứng hiện đại. Thứ tư, IndexFlatIP đơn giản, không yêu cầu huấn luyện hoặc điều chỉnh tham số, giảm độ phức tạp trong triển khai và bảo trì.

**Quy trình xây dựng chỉ mục** bao gồm các bước tuần tự được thực hiện một cách có hệ thống. Bước đầu tiên là tải mô hình embedding và khởi tạo các cấu trúc dữ liệu. Đối tượng SentenceTransformer được tạo với tên mô hình đã cấu hình. Hai danh sách rỗng được khởi tạo: một để chứa tất cả các chunk văn bản và một để chứa metadata tương ứng. Bước thứ hai là đọc và phân tích file JSONL. File được mở với encoding UTF-8 và đọc từng dòng. Mỗi dòng được parse thành đối tượng Python dictionary. Từ mỗi đối tượng, danh sách chunks được trích xuất và lặp qua. Mỗi chunk được thêm vào danh sách văn bản, và một đối tượng metadata chứa chủ đề, mục đích, đối tượng, và chính chunk đó được thêm vào danh sách metadata. Bước thứ ba là tạo embedding cho tất cả các chunk. Danh sách văn bản được đưa vào phương thức encode của mô hình với các cờ chuẩn hóa được bật. Mô hình trả về ma trận NumPy có kích thước (số_chunk, số_chiều). Kích thước embedding (384) được trích xuất từ ma trận. Bước thứ tư là khởi tạo và điền chỉ mục FAISS. Đối tượng IndexFlatIP được tạo với số chiều embedding. Phương thức add được gọi để thêm tất cả các embedding vector vào chỉ mục. FAISS nội bộ lưu trữ các vector trong cấu trúc dữ liệu tối ưu cho tìm kiếm nhanh. Bước cuối cùng là lưu trữ các thành phần trong đối tượng. Chỉ mục FAISS, ma trận embedding, và danh sách metadata được gán vào các thuộc tính của lớp để sử

dụng sau này.

**Xác thực và kiểm tra lỗi** được tích hợp vào quy trình xây dựng. Nếu không có chunk nào được trích xuất từ file dữ liệu (danh sách văn bản rỗng), một ValueError được raise với thông báo rõ ràng. Điều này ngăn việc tạo chỉ mục rỗng có thể gây lỗi khi truy vấn. Nếu file JSONL không tồn tại hoặc không thể đọc, exception được raise và truyền lên cho caller xử lý. Nếu một dòng JSON không hợp lệ, exception được bắt và dòng đó được bỏ qua với ghi log cảnh báo. Các kiểm tra này đảm bảo quá trình xây dựng chỉ mục chỉ thành công khi dữ liệu hợp lệ và đầy đủ.

**Tối ưu hóa bộ nhớ và hiệu suất** được xem xét trong thiết kế. Ma trận embedding được lưu trữ dưới dạng NumPy array với dtype float32, cân bằng giữa độ chính xác và tiêu thụ bộ nhớ. Với 1000 chunk và 384 chiều, bộ nhớ cần thiết khoảng 1.5MB, hoàn toàn chấp nhận được. Chỉ mục FAISS được xây dựng trong bộ nhớ để truy cập nhanh. Đối với các bộ dữ liệu lớn hơn, FAISS hỗ trợ lưu chỉ mục ra đĩa và memory mapping. Metadata được lưu dưới dạng list of dictionaries trong Python, cung cấp truy cập linh hoạt và nhanh. Đối với quy mô lớn hơn, có thể cân nhắc sử dụng cơ sở dữ liệu NoSQL như MongoDB để lưu metadata.

### 2.3.3 Cơ chế truy xuất và xếp hạng

Quá trình truy xuất là trái tim của hệ thống RAG, xác định những tài liệu nào được sử dụng để sinh nội dung bài thuyết trình. Cơ chế truy xuất được triển khai trong phương thức search của lớp EmbeddingIndex.

**Quy trình truy xuất cơ bản** bắt đầu khi người dùng cung cấp thông tin về bài thuyết trình cần tạo. Từ các tham số như chủ đề, mục đích, và đối tượng, một câu truy vấn văn bản được tạo ra. Ví dụ, với chủ đề "Khai phá dữ liệu", mục đích "giảng dạy", và đối tượng "sinh viên", câu truy vấn có thể là "Khai phá dữ liệu giảng dạy sinh viên". Câu truy vấn này sau đó được chuyển thành vector embedding sử dụng cùng mô hình

đã được dùng để tạo chỉ mục. Vector truy vấn có cùng số chiều (384) với các vector trong chỉ mục. Phương thức search của FAISS được gọi với vector truy vấn và tham số `top_k` chỉ định số lượng kết quả mong muốn (thường là 8-10). FAISS thực hiện tìm kiếm bằng cách tính tích vô hướng giữa vector truy vấn và mọi vector trong chỉ mục, sắp xếp các vector theo điểm tích vô hướng giảm dần, và trả về chỉ số (indices) và điểm số (scores) của `top_k` vector có điểm cao nhất.

**Xử lý kết quả truy xuất** chuyển đổi đầu ra thô của FAISS thành định dạng dễ sử dụng. Phương thức search trả về hai mảng NumPy: scores chứa điểm tương đồng và idxs chứa chỉ số của các chunk được truy xuất. Các mảng này có hình dạng (1, `top_k`) vì FAISS hỗ trợ batch query. Hàm lặp qua các chỉ số, kiểm tra xem chỉ số có hợp lệ không (FAISS trả về -1 cho các vị trí không có kết quả), lấy metadata tương ứng từ danh sách metadata đã lưu, tạo một dictionary mới chứa toàn bộ metadata cộng thêm điểm tương đồng, và thêm dictionary vào danh sách kết quả. Danh sách kết quả cuối cùng chứa các dictionary, mỗi dictionary đại diện cho một chunk được truy xuất với đầy đủ thông tin: văn bản của chunk, chủ đề, mục đích, đối tượng, và điểm tương đồng. Các chunk được sắp xếp theo điểm tương đồng giảm dần, với chunk liên quan nhất ở đầu danh sách.

**Điều chỉnh độ nhạy truy xuất** được thực hiện thông qua tham số `top_k`. Giá trị `top_k` quyết định số lượng tài liệu được truy xuất và do đó ảnh hưởng đến độ đa dạng và độ chính xác của nội dung được sinh. Giá trị `top_k` nhỏ (3-5) tạo ra tập kết quả tập trung cao, chỉ bao gồm những chunk cực kỳ liên quan, phù hợp khi cần độ chính xác cao và tránh thông tin nhiễu. Giá trị `top_k` trung bình (8-10) cung cấp sự cân bằng tốt, đủ đa dạng để phủ các khía cạnh khác nhau của chủ đề nhưng vẫn duy trì độ liên quan cao, đây là cài đặt mặc định được sử dụng trong hệ thống. Giá trị `top_k` lớn (15-20) tăng độ đa dạng nhưng có thể bao gồm các chunk ít liên quan hơn, có thể dẫn đến nhiễu trong nội dung được sinh. Trong triển khai hiện tại, `top_k` mặc định là 10 cho quá trình truy xuất ban đầu, nhưng chỉ có 2 chunk đầu tiên được

sử dụng trực tiếp trong sinh ghi chú để đảm bảo tính mạch lạc.

**Xử lý các trường hợp đặc biệt** đảm bảo hệ thống hoạt động ổn định trong mọi tình huống. Nếu chỉ mục chưa được xây dựng (`index is None`), RuntimeError được raise yêu cầu xây dựng chỉ mục trước khi truy vấn. Nếu câu truy vấn rỗng hoặc chỉ chứa khoảng trắng, một vector embedding mặc định hoặc random có thể được sử dụng, hoặc exception có thể được raise tùy thuộc vào chính sách. Nếu số lượng chunk trong chỉ mục ít hơn `top_k` yêu cầu, FAISS tự động trả về tất cả các chunk có sẵn, và code xử lý kết quả điều chỉnh phù hợp. Nếu tất cả các chunk có điểm tương đồng rất thấp (ví dụ dưới 0.1), điều này có thể chỉ ra rằng không có tài liệu thực sự liên quan, và cảnh báo có thể được ghi log.

**Khả năng mở rộng trong tương lai** được xem xét trong thiết kế. Hiện tại hệ thống sử dụng truy xuất dựa trên vector thuần túy, nhưng kiến trúc modular cho phép dễ dàng tích hợp các kỹ thuật nâng cao hơn. Truy xuất hybrid kết hợp cả tìm kiếm ngữ nghĩa (dense retrieval) và tìm kiếm từ khóa (sparse retrieval như BM25) có thể được thêm vào. Xếp hạng lại (reranking) sử dụng mô hình cross-encoder có thể được áp dụng sau truy xuất ban đầu để tinh chỉnh thứ tự. Lọc theo metadata như chủ đề hoặc đối tượng có thể được tích hợp vào logic truy xuất. Mở rộng câu truy vấn (query expansion) sử dụng synonyms hoặc related terms có thể cải thiện recall. Tất cả những mở rộng này có thể được thực hiện mà không cần thay đổi cấu trúc tổng thể của hệ thống.

## 2.4 Thiết kế mẫu câu lệnh và sinh nội dung

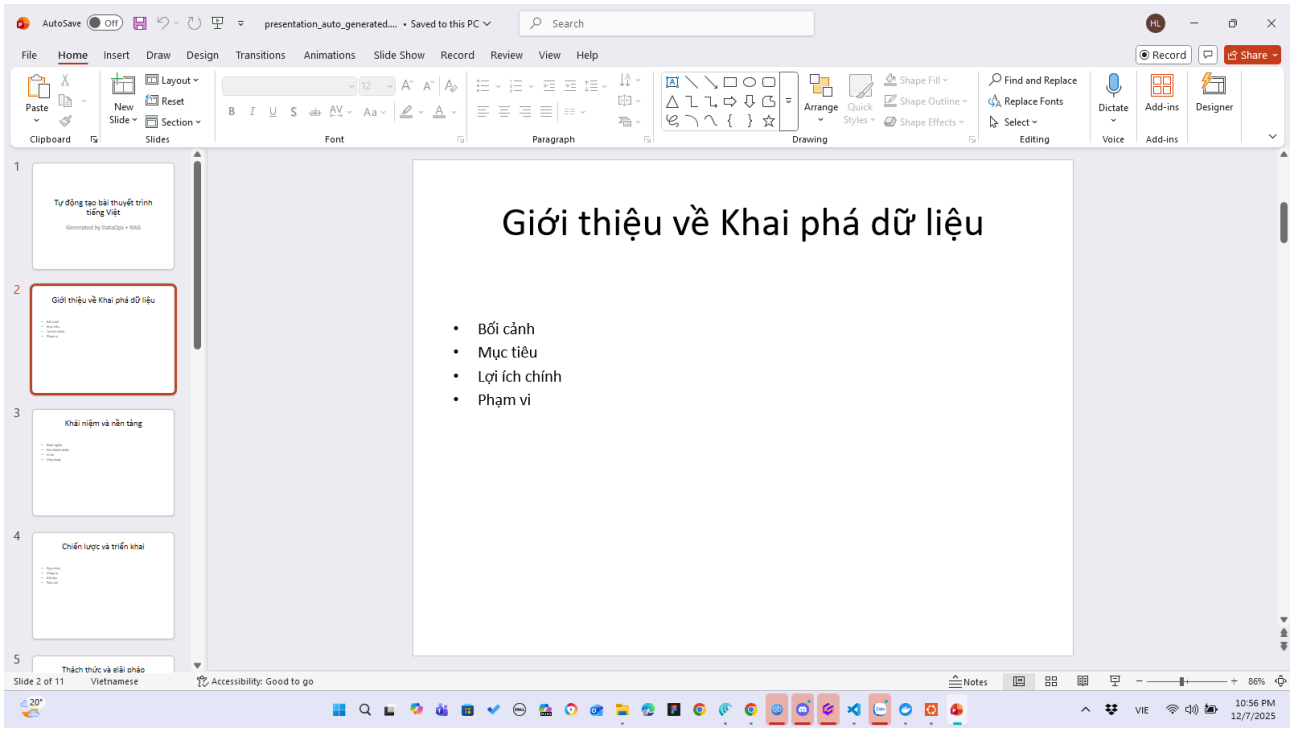
### 2.4.1 Chiến lược thiết kế mẫu câu lệnh

Mẫu câu lệnh (prompt templates) đóng vai trò quan trọng trong việc hướng dẫn quá trình sinh nội dung, đảm bảo các slide và ghi chú cho người nói có cấu trúc tốt, mạch lạc, và phù hợp với mục đích sử dụng. Chiến lược thiết kế mẫu câu lệnh trong đề

tài này tuân theo các nguyên tắc của kỹ thuật thiết kế câu lệnh nâng cao, kết hợp cả phương pháp có cấu trúc và tích hợp thông tin từ RAG.

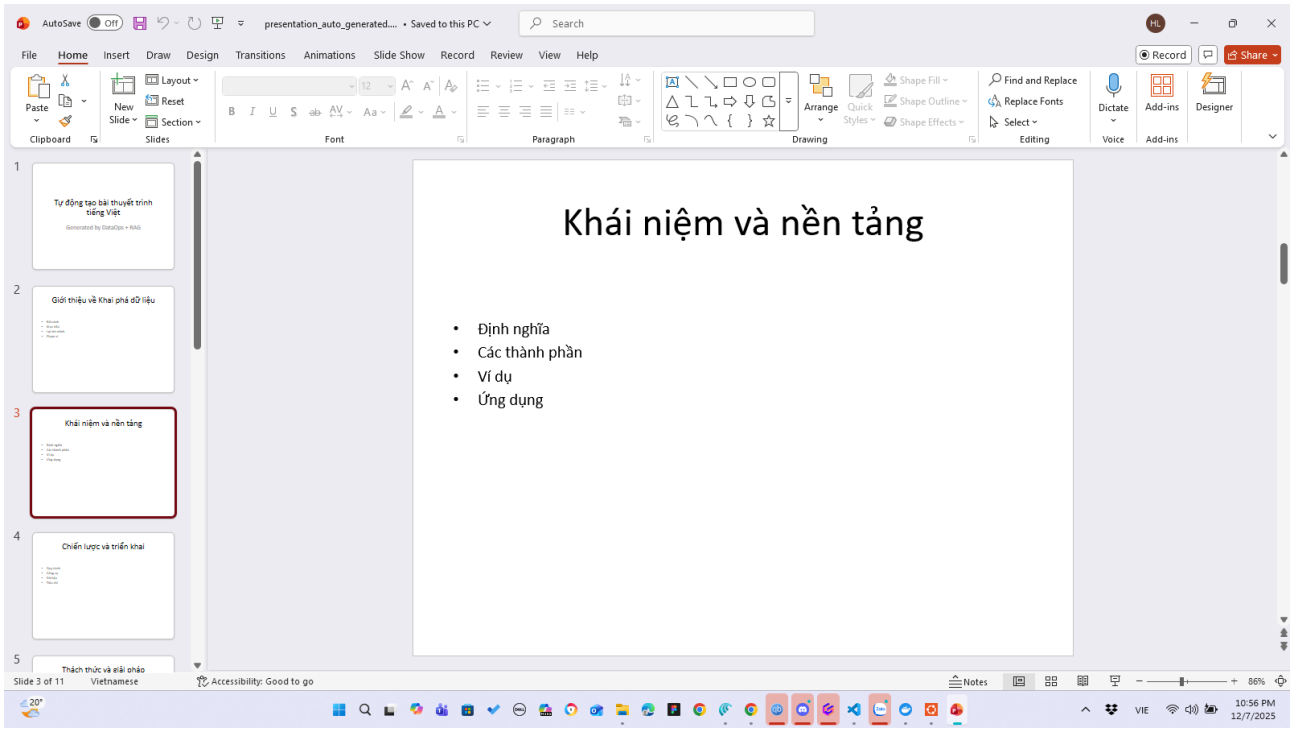
**Nguyên tắc thiết kế tổng quát** được áp dụng nhất quán trong toàn bộ hệ thống mẫu. Nguyên tắc đầu tiên là tính rõ ràng và cụ thể, mỗi mẫu câu lệnh phải chỉ định rõ ràng đầu ra mong muốn bao gồm định dạng, độ dài, và nội dung cần có. Điều này giúp hạn chế sự mơ hồ và đảm bảo kết quả nhất quán. Nguyên tắc thứ hai là tính có cấu trúc, mẫu câu lệnh được tổ chức thành các thành phần logic như phần hướng dẫn, phần ngữ cảnh, phần ví dụ, và phần yêu cầu cụ thể. Cấu trúc rõ ràng giúp dễ hiểu và dễ bảo trì. Nguyên tắc thứ ba là tính dựa trên ngữ cảnh, mẫu câu lệnh tích hợp thông tin ngữ cảnh từ các tham số đầu vào như chủ đề, mục đích, đối tượng khán giả và các chunk được truy xuất từ RAG. Ngữ cảnh phong phú giúp tạo ra nội dung có độ liên quan và độ chính xác cao. Nguyên tắc thứ tư là tính điều chỉnh theo đối tượng, giọng điệu, mức độ kỹ thuật, và phong cách trình bày được điều chỉnh dựa trên đối tượng khán giả mục tiêu. Ví dụ, nội dung cho sinh viên sẽ ngắn gọn và dễ hiểu hơn so với nội dung cho quản lý có thể chuyên nghiệp và súc tích hơn. **Mẫu tạo phác thảo slide** được thiết kế để tạo ra cấu trúc bài thuyết trình có logic và toàn diện. Hàm `build_slide_plan` nhận các tham số đầu vào là chủ đề (topic), mục đích (intent), và đối tượng khán giả (audience), sau đó tạo ra một danh sách các slide với tiêu đề và các điểm đánh dấu chính. Cấu trúc phác thảo tuân theo mô hình năm phần chuẩn cho bài thuyết trình học thuật và kinh doanh.

- Slide thứ hai như hình 2.3 dưới đây là phần giới thiệu với tiêu đề "Giới thiệu về [chủ đề]" và các điểm đánh dấu bao gồm Bối cảnh (cung cấp ngữ cảnh và động lực), Mục tiêu (nêu rõ những gì bài thuyết trình muốn đạt được), Lợi ích chính (tại sao chủ đề này quan trọng), và Phạm vi (giới hạn những gì sẽ được đề cập).



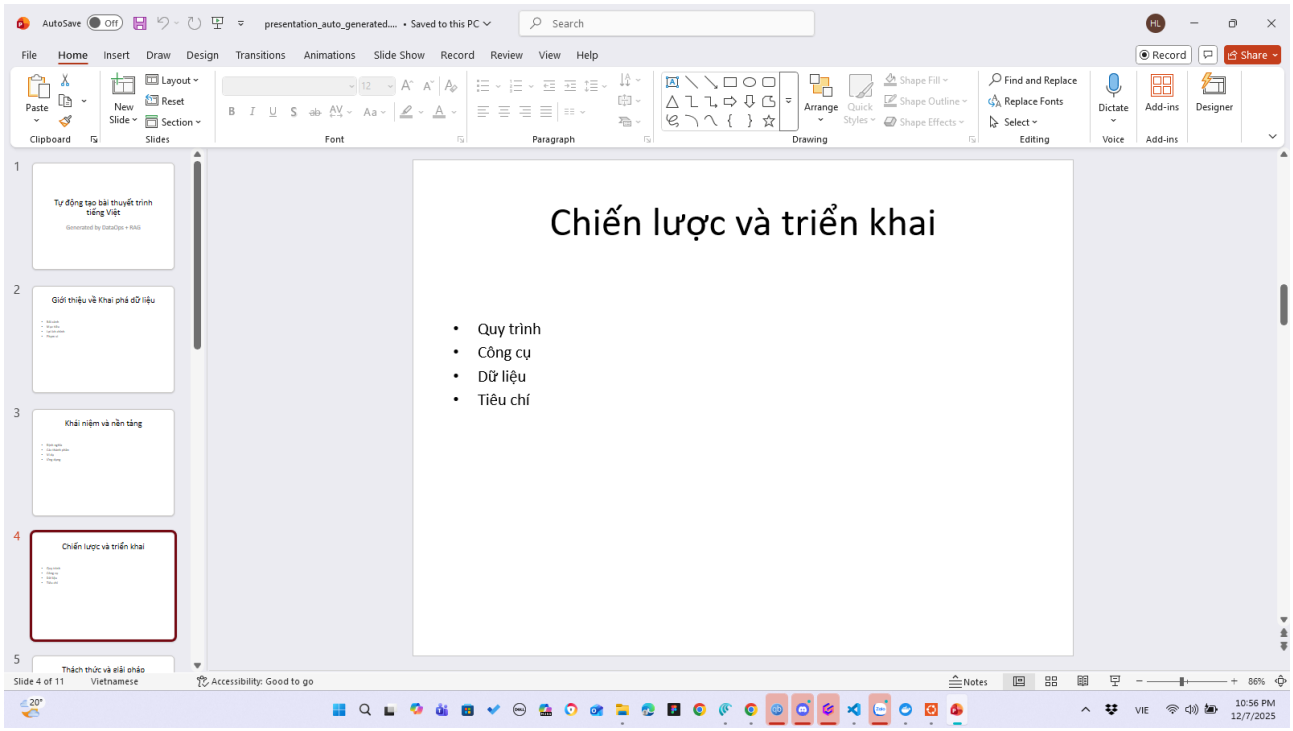

Hình 2.3: Slide thứ hai sau khi tạo sinh

- Slide thứ ba như hình 2.4 dưới đây là về khái niệm và nền tảng bao gồm Định nghĩa (giải thích thuật ngữ cơ bản), Các thành phần (phân tích các yếu tố cấu thành), Ví dụ (minh họa cụ thể), và Ứng dụng (các trường hợp sử dụng thực tế).



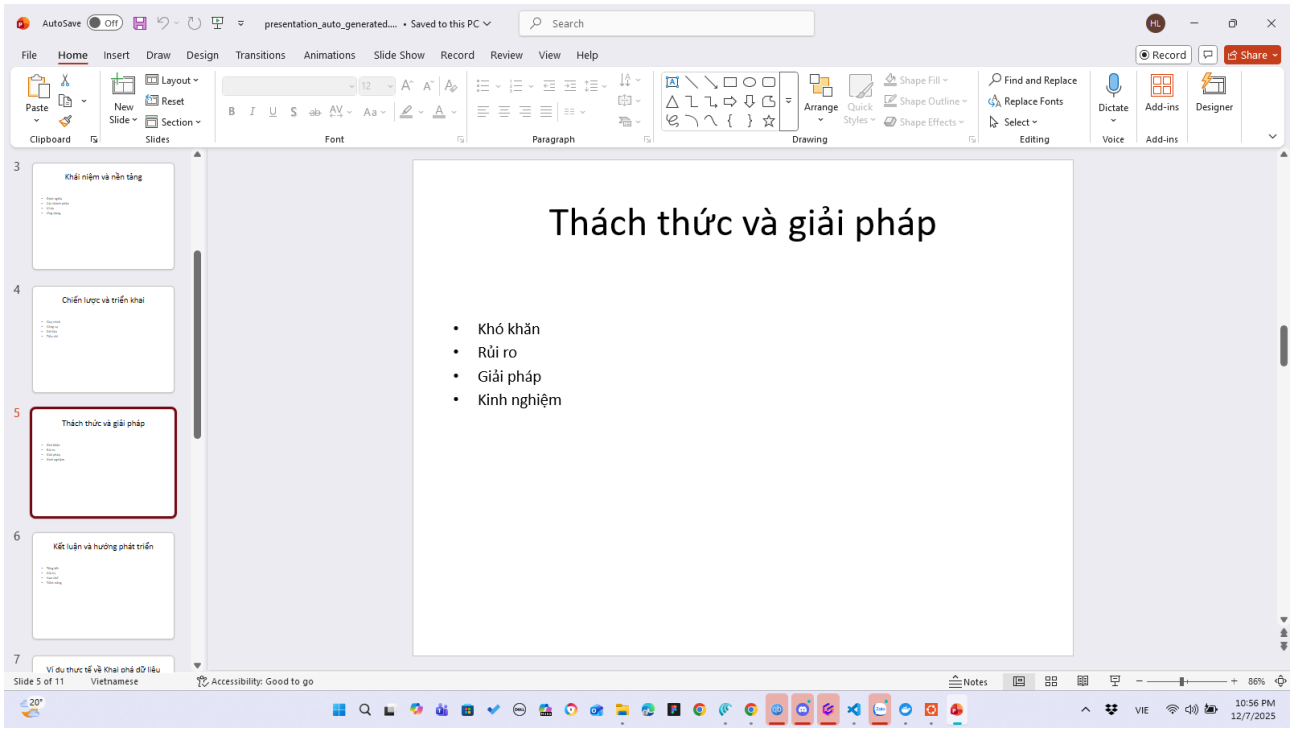

Hình 2.4: Slide thứ ba sau khi tạo sinh

- Slide thứ tư như hình 2.5 dưới đây là về chiến lược và triển khai bao gồm Quy trình (các bước thực hiện), Công cụ (các phương tiện hỗ trợ), Dữ liệu (thông tin cần thiết), và Tiêu chí (các chỉ số đánh giá).



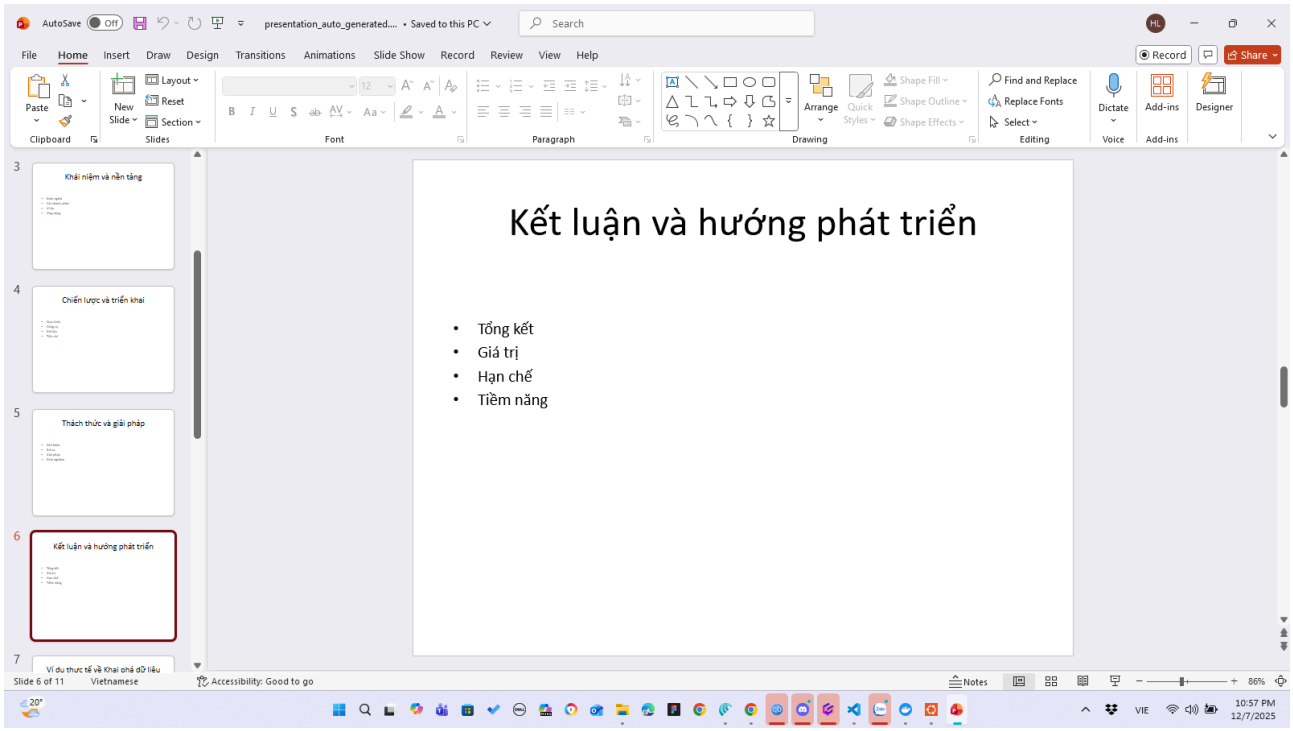

Hình 2.5: Slide thứ tư sau khi tạo sinh

- Slide thứ năm như hình 2.6 dưới đây là về thách thức và giải pháp bao gồm Khó khăn (các vấn đề thường gặp), Rủi ro (các nguy cơ tiềm ẩn), Giải pháp (cách khắc phục), và Kinh nghiệm (bài học từ thực tế).




Hình 2.6: Slide thứ năm sau khi tạo sinh

- Slide thứ sáu như hình 2.7 dưới đây là kết luận với các điểm Tổng kết (tóm tắt nội dung chính), Giá trị (nhấn mạnh ý nghĩa), Hạn chế (thừa nhận những gì chưa đề cập), và Tiềm năng (hướng phát triển tương lai).




Hình 2.7: Slide thứ sáu sau khi tạo sinh

Cấu trúc này đảm bảo bài thuyết trình có sự phát triển logic từ giới thiệu, qua nội dung chính, đến kết luận, giúp người nghe dễ theo dõi và tiếp thu thông tin.

**Mở rộng phác thảo linh hoạt** được tích hợp để đáp ứng các yêu cầu về số lượng slide khác nhau. Nếu số slide yêu cầu lớn hơn năm slide cơ bản, hệ thống tự động mở rộng bằng cách thêm các slide về ví dụ thực tế với cấu trúc "Ví dụ thực tế về [chủ đề]" và các điểm đánh dấu bao gồm Bài học, Kết quả, Tác động, và Khuyến nghị. Cơ chế này cho phép tạo bài thuyết trình với độ dài từ năm đến mười hoặc nhiều slide hơn tùy theo nhu cầu, đồng thời vẫn duy trì tính nhất quán trong cấu trúc.

### 2.4.2 Quy trình sinh nội dung tích hợp RAG

Module `rag_generator.py` tích hợp tất cả các thành phần để tạo ra nội dung bài thuyết trình hoàn chỉnh. Hàm `generate_slides_with_notes` là điểm vào chính, điều phối toàn bộ quy trình từ truy xuất thông tin đến sinh slide và ghi chú.

**Quy trình tích hợp** bắt đầu bằng việc tạo câu truy vấn từ các tham số đầu vào. Câu truy vấn được hình thành bằng cách nối các tham số topic, intent, và audience thành một chuỗi văn bản duy nhất với các khoảng trắng ngăn cách. Ví dụ, với topic = "Khai phá dữ liệu", intent = "giảng dạy", và audience = "sinh viên", câu truy vấn sẽ là "Khai phá dữ liệu giảng dạy sinh viên". Chuỗi này đơn giản nhưng hiệu quả trong việc nắm bắt yêu cầu của người dùng.

Sau khi có câu truy vấn, hệ thống thực hiện truy xuất thông tin từ chỉ mục embedding. Phương thức search của đối tượng index được gọi với câu truy vấn và tham số `top_k` = 10 để lấy mười chunk có độ tương đồng cao nhất. Kết quả trả về là danh sách các dictionary, mỗi dictionary chứa chunk văn bản, metadata, và điểm tương đồng. Từ danh sách này, chỉ có các chunk văn bản được trích xuất và lưu vào danh sách `rag_chunks` để sử dụng sau này. Các chunk này chứa kiến thức thực tế từ cơ sở dữ liệu sẽ được tích hợp vào ghi chú cho người nói.

Tiếp theo, hệ thống tạo phác thảo cấu trúc slide bằng cách gọi hàm `build_slide_plan` với các tham số topic, intent, và audience. Hàm này trả về một danh sách các dictionary outline, mỗi dictionary đại diện cho một slide với thuộc tính title (tiêu đề) và bullets (danh sách điểm đánh dấu). Nếu số lượng slide trong outline nhỏ hơn `n_slides` yêu cầu, một vòng lặp while liên tục thêm các slide về ví dụ thực tế cho đến khi đạt đủ số lượng mong muốn. Cơ chế này cho phép tạo bài thuyết trình với độ dài linh hoạt từ năm đến mười slide hoặc nhiều hơn tùy theo yêu cầu của người dùng.

Sau khi có phác thảo đầy đủ, hệ thống lặp qua từng item trong outline để tạo nội dung chi tiết cho mỗi slide. Đối với mỗi slide, hàm `build_speaker_notes` được gọi với các tham số bao gồm tiêu đề slide, danh sách điểm đánh dấu, các chunk RAG đã truy xuất, và đối tượng khán giả. Hàm này trả về một đoạn ghi chú văn bản được tạo theo các nguyên tắc đã mô tả ở phần trước. Một dictionary mới được tạo chứa ba thuộc tính: title (tiêu đề slide), bullets (danh sách điểm đánh dấu), và notes (ghi chú

cho người nói). Dictionary này được thêm vào danh sách slides tích lũy. Quá trình này được lặp lại cho tất cả các slide, tạo ra một danh sách hoàn chỉnh các đối tượng slide sẵn sàng để xây dựng bài thuyết trình PowerPoint.

**Thiết kế modular của quy trình** mang lại nhiều lợi ích. Mỗi giai đoạn (truy xuất, tạo phác thảo, sinh ghi chú) được đóng gói trong hàm riêng biệt với trách nhiệm rõ ràng, giúp dễ hiểu, kiểm thử, và bảo trì. Các hàm có thể được sử dụng độc lập hoặc kết hợp, cung cấp tính linh hoạt cao. Ví dụ, có thể dễ dàng thay đổi logic tạo phác thảo mà không ảnh hưởng đến logic sinh ghi chú. Dữ liệu được truyền giữa các giai đoạn thông qua các cấu trúc đơn giản như danh sách và dictionary, giảm sự phụ thuộc chặt chẽ. Thiết kế này cũng dễ dàng mở rộng - ví dụ, có thể thêm giai đoạn tiền xử lý câu truy vấn, giai đoạn lọc và xếp hạng lại kết quả RAG, hoặc giai đoạn hậu xử lý ghi chú mà không cần cấu trúc lại toàn bộ module.

**Tính nhất quán trong ngữ cảnh** được duy trì xuyên suốt quá trình sinh nội dung. Các chunk RAG được truy xuất một lần duy nhất ở đầu quy trình và được sử dụng cho tất cả các slide, đảm bảo toàn bộ bài thuyết trình dựa trên cùng một tập thông tin nền tảng. Điều này tạo ra tính nhất quán và mạch lạc trong toàn bộ bài thuyết trình. Thông tin về chủ đề, mục đích, và đối tượng được truyền xuyên suốt tất cả các giai đoạn, đảm bảo mỗi quyết định sinh nội dung đều phù hợp với ngữ cảnh tổng thể.

## 2.5 Tự động hóa PowerPoint và tổng hợp giọng nói

### 2.5.1 Xây dựng bài thuyết trình PowerPoint

Module `pptx_builder.py` cung cấp chức năng chuyển đổi cấu trúc nội dung dạng dữ liệu thành file bài thuyết trình PowerPoint thực tế có thể mở và trình bày. Hàm `build_presentation` là hàm chính thực hiện toàn bộ quá trình xây dựng.

**Khởi tạo bài thuyết trình** là bước đầu tiên trong quy trình. Một đối tượng Presentation mới được tạo từ thư viện python-pptx, đại diện cho một file PowerPoint trống. Đối tượng này cung cấp các phương thức để thêm slide, định dạng nội dung, và lưu file. Kích thước slide được để mặc định là kích thước tiêu chuẩn (tỷ lệ 4:3 hoặc 16:9 tùy theo cấu hình mặc định của python-pptx), phù hợp cho hầu hết các tình huống trình bày.

**Tạo slide tiêu đề** giới thiệu bài thuyết trình. Một slide mới được thêm vào bài thuyết trình sử dụng layout slide tiêu đề (title slide layout), thường là layout đầu tiên trong danh sách các layout có sẵn. Layout tiêu đề thường có hai vùng giữ chỗ: một cho tiêu đề chính và một cho phụ đề. Tiêu đề được điền vào vùng giữ chỗ tiêu đề với văn bản được truyền qua tham số title, thường là một chuỗi mô tả tổng quát về bài thuyết trình như "Tự động tạo bài thuyết trình tiếng Việt". Vùng phụ đề có thể được để trống hoặc điền thông tin bổ sung như tên tác giả, ngày tháng, hoặc tổ chức. Slide tiêu đề thiết lập bối cảnh cho toàn bộ bài thuyết trình và tạo ấn tượng đầu tiên với khán giả.

**Tạo các slide nội dung** là phần chính của quy trình. Hệ thống lặp qua danh sách slides, mỗi phần tử là một dictionary chứa thông tin về một slide. Đối với mỗi slide, một slide PowerPoint mới được thêm vào bài thuyết trình sử dụng layout nội dung tiêu chuẩn, thường là layout thứ hai trong danh sách các layout, có cấu trúc với vùng tiêu đề ở trên và vùng nội dung ở dưới. Tiêu đề của slide được lấy từ thuộc tính title của dictionary và điền vào vùng giữ chỗ tiêu đề. Tiêu đề thường là một chuỗi ngắn gọn mô tả nội dung chính của slide, ví dụ "Giới thiệu về Khai phá dữ liệu" hoặc "Khái niệm và nền tảng".

**Định dạng điểm đánh dấu** yêu cầu xử lý cẩn thận để tạo ra bố cục đẹp mắt và dễ đọc. Vùng nội dung của slide được truy cập thông qua thuộc tính placeholders. Khung văn bản (text frame) của vùng nội dung được xóa sạch để loại bỏ bất kỳ văn bản mặc định nào. Sau đó, hệ thống lặp qua danh sách bullets từ dictionary slide. Đối với mỗi

bullet, một đoạn văn (paragraph) mới được thêm vào khung văn bản. Văn bản của bullet được gán vào thuộc tính text của đoạn văn. Mức độ thụt lề được đặt thông qua thuộc tính level, với giá trị 0 cho các điểm đánh dấu cấp cao nhất. Điều này tạo ra bố cục danh sách có cấu trúc với các dấu đầu dòng tiêu chuẩn. Font, kích thước, và màu sắc có thể được tùy chỉnh thông qua thuộc tính font của các đoạn chạy (runs) nếu cần, nhưng trong triển khai hiện tại, định dạng mặc định được sử dụng để duy trì tính đơn giản và nhất quán.

**Thêm ghi chú cho người nói** là bước quan trọng để cung cấp hướng dẫn cho người trình bày. Mỗi slide trong PowerPoint có một phần ghi chú riêng biệt không hiển thị cho khán giả nhưng có thể xem được bởi người trình bày. Phần ghi chú được truy cập thông qua thuộc tính `notes_slide` của đối tượng slide. Khung văn bản ghi chú (notes text frame) chứa nội dung ghi chú. Ghi chú được lấy từ thuộc tính notes của dictionary slide và gán vào thuộc tính text của khung văn bản. Điều này tạo liên kết giữa slide trực quan và hướng dẫn thuyết trình chi tiết, giúp người trình bày có thể đọc hoặc tham khảo ghi chú trong khi trình bày mà không ảnh hưởng đến trải nghiệm của khán giả.

**Lưu file bài thuyết trình** là bước cuối cùng. Sau khi tất cả các slide đã được tạo và định dạng, đối tượng Presentation được lưu vào đường dẫn file được chỉ định thông qua tham số `out_path`. Phương thức save của đối tượng Presentation xử lý việc tuần tự hóa tất cả dữ liệu thành định dạng Open XML (file .pptx) và ghi vào đĩa. File kết quả là một file PowerPoint hợp lệ có thể được mở bằng Microsoft PowerPoint, LibreOffice Impress, Google Slides, hoặc bất kỳ phần mềm tương thích nào khác. File chứa đầy đủ nội dung bao gồm slide tiêu đề, các slide nội dung với tiêu đề và điểm đánh dấu, và ghi chú cho người nói cho mỗi slide.

**Xử lý lỗi và xác thực** được tích hợp để đảm bảo độ tin cậy. Nếu danh sách slides rỗng, một cảnh báo có thể được ghi log và chỉ slide tiêu đề được tạo. Nếu một slide

trong danh sách thiếu thuộc tính bắt buộc như title hoặc bullets, giá trị mặc định hoặc chuỗi rỗng được sử dụng để tránh lỗi. Nếu đường dẫn đầu ra không hợp lệ hoặc không có quyền ghi, exception sẽ được raise với thông báo rõ ràng. Các kiểm tra này đảm bảo quá trình xây dựng PowerPoint chỉ thất bại khi có lỗi thực sự không thể phục hồi.

### 2.5.2 Chuyển văn bản thành giọng nói

Module `tts_service.py` cung cấp chức năng chuyển đổi ghi chú văn bản thành các file âm thanh có thể phát, tạo ra bài thuyết trình tự động hoàn chỉnh. Hàm `synthesize_slide_audio` là hàm chính điều phối quá trình tổng hợp giọng nói.

**Quy trình tổng hợp giọng nói** bắt đầu bằng việc lặp qua danh sách slides. Mỗi slide được xử lý tuần tự để tạo file âm thanh tương ứng. Đối với mỗi slide, ghi chú cho người nói được trích xuất từ thuộc tính notes của dictionary slide. Ghi chú này là văn bản sẽ được chuyển thành giọng nói. Trước khi chuyển đổi, văn bản có thể trải qua một số tiền xử lý nhẹ như loại bỏ các ký tự đặc biệt có thể gây lỗi TTS, chuẩn hóa khoảng trắng, và đảm bảo văn bản không rỗng. Nếu ghi chú rỗng, một thông báo mặc định có thể được sử dụng hoặc file âm thanh có thể được bỏ qua tùy theo chính sách.

**Tạo đối tượng gTTS** để thực hiện chuyển đổi. Một đối tượng gTTS mới được tạo với hai tham số chính: text (văn bản cần chuyển đổi) và lang (mã ngôn ngữ). Tham số lang được đặt là "vi" để chỉ định tiếng Việt, đảm bảo mô hình TTS sử dụng phát âm và ngữ điệu phù hợp với tiếng Việt bao gồm các thanh điệu đặc trưng. Đối tượng gTTS nội bộ kết nối với API của Google Translate để thực hiện tổng hợp giọng nói, sử dụng công nghệ WaveNet hoặc các công nghệ TTS tiên tiến khác của Google để tạo ra giọng nói tự nhiên.

**Lưu file âm thanh** được thực hiện thông qua phương thức save của đối tượng gTTS. Đường dẫn file đầu ra được xây dựng bằng cách kết hợp thư mục đầu ra (`out_dir`),

một prefix cố định, số thứ tự slide được đệm bằng số không để duy trì thứ tự sắp xếp đúng (ví dụ slide_1, slide_2,. . . ), và phần mở rộng .mp3. Phương thức save tải xuống dữ liệu âm thanh từ API Google và ghi vào file MP3. File MP3 kết quả chứa âm thanh giọng nói đọc ghi chú với chất lượng và tốc độ phù hợp cho việc nghe.

**Xử lý nhiều slide** được tự động hóa thông qua vòng lặp. Hệ thống duy trì một biến đếm hoặc sử dụng enumerate để theo dõi số thứ tự slide hiện tại. Điều này đảm bảo mỗi file âm thanh có tên duy nhất tương ứng với vị trí slide trong bài thuyết trình. Người dùng có thể dễ dàng xác định file âm thanh nào thuộc về slide nào dựa trên quy ước đặt tên. Tất cả các file âm thanh được lưu trong cùng một thư mục audio chuyên dụng để dễ quản lý và phân phối.

**Xử lý lỗi mạng và dịch vụ** rất quan trọng vì gTTS phụ thuộc vào kết nối internet và tính sẵn có của dịch vụ Google. Mỗi lần gọi save được bao bọc trong khối try-except để bắt các exception như lỗi mạng, hết thời gian chờ, hoặc lỗi dịch vụ. Nếu một slide cụ thể không thể chuyển đổi, lỗi được ghi log với thông tin chi tiết về slide và lỗi, sau đó quá trình tiếp tục với slide tiếp theo thay vì crash toàn bộ chương trình. Nếu quá nhiều slide gặp lỗi, một cảnh báo tổng thể có thể được phát ra nhưng các file đã tạo thành công vẫn được giữ lại. Thiết kế chịu lỗi này đảm bảo hệ thống có thể tạo ra một phần kết quả ngay cả khi không thể hoàn thành toàn bộ quá trình.

**Tối ưu hóa hiệu suất** có thể được áp dụng cho các bộ dữ liệu lớn. Hiện tại, các file âm thanh được tạo tuần tự, nhưng vì mỗi thao tác TTS độc lập, chúng có thể được song song hóa sử dụng threading hoặc multiprocessing để giảm thời gian tổng thể. Cache có thể được triển khai để tránh tạo lại âm thanh cho các ghi chú giống hệt nhau. Nén hoặc điều chỉnh chất lượng âm thanh có thể được áp dụng nếu kích thước file là mối quan tâm. Tuy nhiên, đối với quy mô hiện tại của hệ thống (dưới mười slide), phương pháp tuần tự đơn giản đã đủ nhanh và đáng tin cậy.

### 2.5.3 Tích hợp ghi nhật ký và giám sát

Module `bi_logging.py` cung cấp chức năng ghi nhật ký các chỉ số hiệu suất và thống kê của pipeline để hỗ trợ phân tích và tối ưu hóa. Hàm `append_metric` là hàm tiện ích để ghi các metric vào file CSV.

**Cấu trúc dữ liệu nhật ký** được thiết kế để dễ dàng phân tích. Mỗi metric được biểu diễn dưới dạng dictionary với các cặp key-value, trong đó key là tên metric và value là giá trị đo được. Ví dụ, một metric về thời gian thực thi có thể bao gồm các key như `dataset_seconds`, `index_seconds`, `generation_seconds`, `pptx_seconds`, `tts_seconds`, và `total_seconds` với các giá trị là số giây được làm tròn đến ba chữ số thập phân. Metric về thống kê bài thuyết trình có thể bao gồm `topic`, `intent`, `audience`, `slides_count`, và `avg_notes_len_words`. Cấu trúc dictionary linh hoạt cho phép thêm hoặc xóa các metric mà không cần thay đổi schema cố định.

**Quy trình ghi metric** được thực hiện một cách có hệ thống. Hàm `append_metric` nhận ba tham số: thư mục logs (`logs_dir`), tên file (filename), và dictionary metric (metric). Đầu tiên, đường dẫn đầy đủ đến file CSV được xây dựng bằng cách kết hợp `logs_dir` và filename. Sau đó, hàm kiểm tra xem file có tồn tại hay không. Nếu file chưa tồn tại, điều này cho biết đây là lần ghi đầu tiên và một file mới cần được tạo với dòng tiêu đề. Dòng tiêu đề chứa tất cả các key từ metric dictionary, được nối bằng dấu phẩy để tạo thành header CSV chuẩn. Nếu file đã tồn tại, không cần thêm tiêu đề và dữ liệu mới sẽ được append vào cuối file. Tiếp theo, các giá trị từ metric dictionary được trích xuất theo thứ tự tương ứng với các key trong tiêu đề và nối lại thành một dòng CSV. Dòng này được ghi vào file bằng cách mở file ở chế độ append. Cuối cùng, file được đóng để đảm bảo dữ liệu được flush vào đĩa.

**Các loại metric được thu thập** bao gồm nhiều khía cạnh của hệ thống. Metric về thời gian được thu thập ở nhiều điểm trong pipeline sử dụng hàm `perf_counter`

từ module time để đo thời gian với độ chính xác cao. Thời gian để xây dựng bộ dữ liệu (`dataset_seconds`), xây dựng chỉ mục embedding (`index_seconds`), sinh nội dung qua RAG (`generation_seconds`), tạo file PowerPoint (`pptx_seconds`), tổng hợp giọng nói (`tts_seconds`), và tổng thời gian từ đầu đến cuối (`total_seconds`) đều được ghi nhận. Các metric này giúp xác định các giai đoạn nào là nút thắt cổ chai và cần tối ưu hóa. Metric về thống kê bài thuyết trình bao gồm thông tin về chủ đề (`topic`), mục đích (`intent`), đối tượng (`audience`), số lượng slide (`slides_count`), và độ dài trung bình của ghi chú tính bằng số từ (`avg_notes_len_words`). Các metric này giúp hiểu đặc điểm của các bài thuyết trình được tạo ra.

**Tích hợp với Apache Superset** được hỗ trợ thông qua định dạng CSV chuẩn. Các file CSV được tạo bởi module logging có thể được nhập trực tiếp vào cơ sở dữ liệu mà Superset kết nối, hoặc Superset có thể được cấu hình để đọc trực tiếp từ file CSV. Sau khi dữ liệu có trong Superset, người dùng có thể tạo các trực quan hóa như biểu đồ đường thể hiện xu hướng thời gian thực thi theo thời gian, biểu đồ cột so sánh thời gian của các giai đoạn khác nhau, biểu đồ phân tán khám phá mối quan hệ giữa số slide và thời gian tạo, và dashboard tổng hợp cung cấp cái nhìn tổng quan về sức khỏe hệ thống. Khả năng trực quan hóa này rất có giá trị cho việc giám sát sản xuất và ra quyết định tối ưu hóa dựa trên dữ liệu.

**Mở rộng và tùy chỉnh** hệ thống logging rất dễ dàng nhờ thiết kế linh hoạt. Các metric mới có thể được thêm vào bằng cách đơn giản bổ sung thêm các cặp key-value vào dictionary metric khi gọi `append_metric`. Các file nhật ký mới có thể được tạo cho các loại metric khác nhau bằng cách sử dụng tên file khác nhau. Định dạng dữ liệu có thể được mở rộng để bao gồm `timestamp`, `user_id`, hoặc các thông tin ngữ cảnh khác. Các cảnh báo có thể được thiết lập dựa trên ngưỡng của các metric nhất định, ví dụ cảnh báo khi thời gian tổng vượt quá 5 phút hoặc khi tỷ lệ lỗi cao hơn 10%.

# Chương 3

# Kết quả thực nghiệm

# Kết luận và hướng phát triển

# Tài liệu tham khảo