XÂY DỰNG HỆ THỐNG HỎI ĐÁP TỰ ĐỘNG VỀ LỊCH SỬ – DU LỊCH VIỆT NAM

Thành viên thực hiện:

Nguyễn Vĩnh Hưng (52200097@student.tdtu.edu.vn)

Nguyễn Hòa An (52200182@student.tdtu.edu.vn)

Giảng viên hướng dẫn:

TS. Trần Thanh Phước (tranthanhphuoc@tdtu.edu.vn)

Trường Đại học Tôn Đức Thắng, Quận 7, TP.HCM

Tháng 6, 2025

Tóm tắt nội dung

Báo cáo trình bày đề tài xây dựng hệ thống chatbot hỏi đáp tự động về lịch sử – du lịch Việt Nam. Đề tài tập trung vào việc chuẩn hóa tri thức từ văn bản, xây dựng mô hình trả lời tự động và triển khai ứng dụng minh họa dưới dạng chatbot tiếng Việt.

1 Giới thiệu

1.1 Bối cảnh thực hiện

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, việc tìm kiếm và truy xuất thông tin một cách nhanh chóng, chính xác đã trở thành nhu cầu thiết yếu của nhiều lĩnh vực, đặc biệt là giáo dục, du lịch và nghiên cứu khoa học. Các hệ thống hỏi đáp tự động (Question Answering Systems) đang ngày càng phổ biến nhờ khả năng hỗ trợ người dùng tiếp cận tri thức một cách trực quan, dễ dàng. Trong lĩnh vực giáo dục, đặc biệt là môn Lịch sử, nhu cầu tra cứu thông tin về sự kiện, nhân vật, giai đoạn lịch sử... để phục vụ học tập, giảng dạy và nghiên cứu ngày càng cao. Bên cạnh đó, việc liên kết kiến thức lịch sử với các thông tin về du lịch, di tích, danh lam thắng cảnh không chỉ giúp nâng cao trải nghiệm học tập mà còn góp phần quảng bá và bảo tồn di sản văn hóa, lịch sử của dân tộc.

Hiện nay, Việt Nam sở hữu một kho tàng dữ liệu phong phú về lịch sử và du lịch, bao gồm sách, tài liệu nghiên cứu, cơ sở dữ liệu trực tuyến và nhiều nguồn thông tin khác. Tuy nhiên, phần lớn các nguồn tư liệu này tồn tại dưới dạng văn bản dài, rời rạc và chưa được chuẩn hóa, khiến việc tìm kiếm thông tin theo yêu cầu cụ thể còn gặp nhiều khó khăn.

Mối liên kết giữa thông tin lịch sử và thông tin du lịch hầu như chưa được khai thác đầy đủ. Người dùng muốn tìm hiểu về một nhân vật hoặc sự kiện lịch sử thường phải tự tìm kiếm ở nhiều nguồn khác nhau, trong khi đó việc kết hợp những thông tin này để gợi ý các địa điểm du lịch liên quan vẫn còn hạn chế.
Trên thế giới, đã xuất hiện nhiều hệ thống chatbot hỏi đáp nổi tiếng như ChatGPT, Google Gemini, Claude, hay Bing AI, có khả năng trả lời linh hoạt và bao quát nhiều lĩnh vực. Tuy nhiên, các hệ thống này vẫn chưa có một phiên bản được thiết kế chuyên biệt cho lịch sử và du lịch Việt Nam. Ngay cả khi có thể đưa ra câu trả lời, độ chính xác đối với các thông tin chuyên sâu về lịch sử Việt Nam vẫn chưa được đảm bảo tuyệt đối. Điều này đặc biệt quan trọng trong bối cảnh chatbot phục vụ giáo dục và nghiên cứu lịch sử – nơi tính chính xác là yếu tố then chốt. Một số nguyên nhân dẫn đến hạn chế này có thể kể đến như:

- Thiếu nguồn dữ liệu được xác thực và chuẩn hóa, dẫn đến việc mô hình có thể đưa ra thông tin sai lệch hoặc "ảo giác" (hallucination).
- Một số tài liệu quan trọng bị giới hạn quyền truy cập hoặc khó tích hợp vào hệ thống.
- Chưa có cơ sở dữ liệu liên kết đầy đủ giữa thông tin lịch sử và thông tin du lịch, khiến việc truy xuất theo ngữ cảnh liên ngành trở nên khó khăn.

Trong bối cảnh đó, việc xây dựng một hệ thống hỏi đáp tự động được thiết kế chuyên biệt cho lịch sử – du lịch Việt Nam là cần thiết và có ý nghĩa thực tiễn cao. Để giải quyết vấn đề, đề tài này đề xuất áp dụng kiến trúc **RAG (Retrieval-Augmented Generation)** kết hợp với **đồ thị tri thức (Knowledge Graph)**. RAG cho phép hệ thống truy xuất thông tin từ cơ sở dữ liệu bên ngoài và kết hợp với sức mạnh sinh ngôn ngữ của các mô hình LLM (Large Language Models) để tạo ra câu trả lời vừa chính xác vừa có ngữ cảnh đầy đủ. Trong khi đó, Knowledge Graph giúp tổ chức và liên kết tri thức lịch sử – du lịch theo cấu trúc có ngữ nghĩa, tạo nền tảng cho việc tìm kiếm thông minh và mở rộng thông tin theo mối quan hệ.
Với cách tiếp cận này, hệ thống chatbot được kỳ vọng sẽ:

- Tăng độ chính xác trong việc truy xuất và trả lời các câu hỏi về lịch sử Việt Nam.
- Tạo ra một công cụ hỗ trợ học tập, nghiên cứu và giảng dạy môn Lịch sử một cách trực quan, hiệu quả.
- Mở rộng khả năng khám phá du lịch dựa trên dữ liệu lịch sử, góp phần quảng bá di sản văn hóa và lịch sử Việt Nam đến với cộng đồng trong và ngoài nước.

Nhờ sự kết hợp giữa RAG và Knowledge Graph, hệ thống không chỉ giải quyết được vấn đề độ chính xác và tính cập nhật, mà còn tạo ra cầu nối giữa kiến thức lịch sử và du lịch

– một hướng tiếp cận mới, góp phần đổi mới phương pháp tiếp cận tri thức trong thời đại số.

1.2 Mục tiêu đồ án

Mục tiêu chính:

- Xây dựng một hệ thống chatbot hỏi đáp tiếng Việt về lịch sử – du lịch Việt Nam dựa trên kết hợp RAG + Knowledge Graph.
- Xây dựng kho ngữ liệu hỏi đáp tự động về lịch sử - du lịch tiếng Việt và công bố một phần dữ liệu cho cộng đồng nghiên cứu.

Mục tiêu cụ thể:

1. Xây dựng được hệ thống chatbot có khả năng hỏi đáp về lịch sử và du lịch Việt Nam.
2. Xử lý, chuẩn hóa dữ liệu từ sách lịch sử – du lịch để xây dựng tri thức.
3. Trích xuất thực thể - quan hệ từ dữ liệu thô để tạo đồ thị tri thức.
4. Xây dựng mô hình hỏi đáp dựa trên kiến trúc RAG kết hợp Knowledge Graph.
5. Triển khai giao diện demo trên web.

Giới hạn phạm vi đồ án:

- Dữ liệu giới hạn lịch sử từ thời Khởi thuỷ đến năm 2000.
- Dữ liệu du lịch bao gồm các địa điểm trong Cơ sở dữ liệu du lịch Việt Nam.
- Ngôn ngữ sử dụng: Tiếng Việt.
- Chatbot hỏi đáp tập trung vào việc xử lý ngôn ngữ tiếng Việt, chưa có tích hợp dùng giọng nói, xử lý đa ngôn ngữ.

1.3 Cấu trúc đồ án

[Cấu trúc các chương...]

2 Kiến thức nền tảng

2.1 Công trình liên quan

2.1.1 Bối cảnh

Trong những năm gần đây, với sự phát triển của các mô hình ngôn ngữ lớn (LLM), các kỹ thuật hiện đại mới dần ra đời nhằm khắc phục những hạn chế của các mô hình trước đó. Đặc biệt là kỹ thuật Retrieval-Augmented Generation (RAG) trong lĩnh vực Hệ thống hỏi đáp (Question Answering - QA) được đánh giá là một giải pháp để nâng cao độ chính xác, khả năng truy vết và giảm lệ thuộc vào dữ liệu huấn luyện.

2.1.2 Các công trình nghiên cứu nổi bật

Các công trình quốc tế tiêu biểu:

- Yunfan Gao et al. (2023) đã hệ thống hóa toàn bộ tiến trình phát triển của các mô hình Retrieval-Augmented Generation (RAG), từ kiến trúc cơ bản gồm retriever–generator đến các biến thể nâng cao như Multi-hop RAG, Fact-augmented Generation, hay KG-RAG. Bài báo khảo sát cách tích hợp dữ liệu không cấu trúc và có cấu trúc (như Knowledge Graphs) vào pipeline sinh văn bản, đồng thời so sánh ưu nhược điểm giữa RAG và fine-tuning theo nhiều tiêu chí như hiệu quả, khả năng mở rộng, và độ tin cậy. [1]

- Sanmartin et al. (2024) giới thiệu mô hình KG-RAG: Bridging the Gap Between Knowledge and Creativity, trong đó KG được sử dụng để định hướng quá trình truy hồi thông tin thông qua cơ chế Chain of Explorations. Mô hình cho phép dẫn dắt việc lựa chọn các tài liệu liên quan từ KG theo chuỗi quan hệ ngữ nghĩa, sau đó kết hợp vào đầu vào của LLM để sinh phản hồi chính xác hơn. Kết quả cho thấy mô hình này vượt trội trong các tác vụ yêu cầu lập luận nhiều bước. [2]

- Zhu et al. (2025) đề xuất mô hình KG^{2}RAG kết hợp truy hồi ngữ nghĩa và graph-guided chunk expansion (lấy thêm các đoạn văn bản (chunk) có liên quan đến câu hỏi thông qua các quan hệ trong KG), giúp tăng đa dạng và mạch lạc cho nội dung trả lời. Kết quả thực nghiệm cho thấy KG-RAG mô hình trả lời tốt hơn các câu hỏi phức tạp hoặc nhiều ngữ cảnh. [3]

- Hogan et al. (2021) đã thực hiện một khảo sát toàn diện về *Knowledge Graphs*, bao quát từ các mô hình dữ liệu (RDF, property graph) và ngôn ngữ truy vấn (SPARQL, Cypher, Gremlin) cho đến các kỹ thuật suy luận (deductive) và học suy luận (inductive). Bài báo cũng phân loại các phương pháp hoàn thiện KG

(Knowledge Graph Completion) thành truyền thống và dựa trên học biểu diễn, đồng thời thảo luận thách thức và hướng nghiên cứu tương lai. [4]

- Wang et al. (2017) đã khảo sát các phương pháp *knowledge graph embedding*, từ mô hình dịch (translation-based) như TransE, TransH, TransR cho đến mô hình so khớp ngữ nghĩa, phân tích ưu – nhược điểm của từng nhóm và các ứng dụng như dự đoán liên kết (link prediction) hay phân loại thực thể (entity classification). [5]

- Schlichtkrull et al. (2018) giới thiệu mô hình *Relational Graph Convolutional Networks (R-GCNs)* cho dữ liệu quan hệ, cho phép kết hợp thông tin cấu trúc đồ thị và đặc trưng thuộc tính để dự đoán liên kết và phân loại thực thể. Thực nghiệm cho thấy R-GCN cải thiện đáng kể hiệu quả trên các tập dữ liệu đồ thị tri thức so với các mô hình factorization thuần túy. [6]

- Angles et al. (2017) cung cấp nền tảng lý thuyết cho các ngôn ngữ truy vấn đồ thị, so sánh SPARQL, Cypher, và Gremlin dựa trên mô hình hình thức, khả năng biểu đạt và độ phức tạp. Công trình này đóng vai trò cơ sở cho việc lựa chọn và tối ưu ngôn ngữ truy vấn trong các hệ thống KG. [7]

- Hur et al. (2021) khảo sát các kỹ thuật xây dựng *Knowledge Graph* tự động từ dữ liệu thô, phân loại theo nguồn dữ liệu (văn bản, bảng, web) và phương pháp (trích xuất thông tin, liên kết thực thể, hợp nhất tri thức). Bài báo cũng nêu các thách thức như xử lý dữ liệu nhiễu, cập nhật tri thức và đánh giá chất lượng KG. [8]

- Mukherjee et al. (2025) đề xuất hệ thống KG-RAG dùng KG để cải thiện độ chính xác trong chatbot truy vấn tài liệu doanh nghiệp, với kỹ thuật xây dựng KG chất lượng cao cho RAG giúp giảm 50% các phản hồi không liên quan và tăng 88% phản hồi hoàn toàn phù hợp. [9]

- Chen et al. (2024) phát triển KG-Retriever, một framework RAG kết hợp chỉ mục phân cấp dựa trên KG và tài liệu, tăng cường kết nối nội dung để cải thiện hiệu suất truy hồi đa văn bản. [10]

- Xu et al. (2024) giới thiệu phương pháp kết hợp RAG với KG trong trợ lý trả lời câu hỏi dịch vụ khách hàng, xây dựng KG từ dữ liệu hỗ trợ và cải thiện MRR lên 77.6% và BLEU thêm 0.32. [11]

- Böckling et al. (2025) đề xuất Walk&Retrieve — phương pháp RAG kết hợp walk trên KG và verbalization nâng cao truy hồi zero-shot, cải thiện độ chính xác và giảm ảo tưởng (hallucination). [12]

- Mavromatis Karypis (2024) cho ra GNN-RAG, kết hợp GNN để truy hồi từ KG và sử dụng đường đi KG verbalized làm ngữ cảnh cho LLM, vượt hiệu năng GPT-4 trên các benchmark KGQA. [13]

- "Knowledge Graph-Guided Retrieval Augmented Generation (KG^{2}RAG)" (2025) sử dụng KG để mở rộng và tổ chức các đoạn nội dung retrieved, cải thiện tính mạch lạc và đa dạng trên HotpotQA. [14]
- "KG Retrieval-Augmented Generation for LLM-based Recommendation" (2025) ứng dụng KG-RAG để nâng cao hiệu quả hệ thống đề xuất, giảm hallucination và cập nhật kiến thức. [15]
- Swacha Gracel (2025) khảo sát các ứng dụng chatbot RAG trong giáo dục, liệt kê nhiều hệ thống như ChatPapers, ChatDoctor, ChatLaw, và PaperQA kết hợp LLM với RAG trong các lĩnh vực chuyên ngành. [16]
- "Systematic Analysis of RAG-Based LLMs for Medical Chatbot Applications" (2025) phân tích hệ thống RAG trong chatbot y tế, so sánh các kiểu RAG và xem xét hiệu quả trong giảm ảo tưởng và đảm bảo độ tin cậy cao. [17]
- "Document GraphRAG: Knowledge Graph Enhanced RAG for Document QA in Manufacturing Domain" (2024) khảo sát các cải tiến RAG tiêu chuẩn bằng cách tích hợp cấu trúc văn bản và KG để xử lý hiệu quả thông tin dài, đa nguồn trong môi trường sản xuất. [18]

Các công trình trong nước tiêu biểu:

- Đại học Bách Khoa TP.HCM (2024) xây dựng mô hình KG từ nhiều nguồn dữ liệu cho một lĩnh vực cụ thể (giáo dục) và tích hợp nó vào hệ thống QA sử dụng LLM và RAG để tăng hiệu quả hỏi đáp. Đây là một trong những nghiên cứu hiếm tại Việt Nam áp dụng RAG kết hợp KG trong thực tế. [19]
- Dự án Graph-RAG (Huy-L-D) xây dựng hệ thống thử nghiệm QA lịch sử Việt. Dự án sử dụng KG lịch sử kết hợp với retrieval-guided chunk expansion từ đồ thị để truy hồi nội dung ngữ cảnh phù hợp, sau đó sinh phản hồi bằng LLM. Mặc dù chưa có đánh giá chính thức, hệ thống này có định hướng gần tương tự với đề tài hiện tại. [20]

2.1.3 Điểm mới và khoảng trống trong đề tài

Các nghiên cứu hiện nay chỉ mới tiếp cận ở hai hướng đó là áp dụng RAG với dữ liệu không cấu trúc để tăng hiệu quả truy hồi và sinh phản hồi và kết hợp Knowledge Graph (KG) như một nguồn dữ liệu phụ trợ trong pipeline hỏi đáp. Vẫn còn tồn tại một số khoảng trống trong nghiên cứu như:

- Chưa có nghiên cứu nào tích hợp sâu giữa RAG và KG trong lĩnh vực lịch sử và du lịch.

- Chưa có nghiên cứu đánh giá so sánh giữa việc kết hợp RAG với KG và các hướng fine-tuing kết hợp KG trong hệ thống hỏi đáp.

Từ những khoảng trống này, đề tài này đề xuất một hướng tiếp cận mới bằng việc xây dựng ra đồ thị tri thức chuyên biệt cho lĩnh vực lịch sử - du lịch Việt Nam. Đánh giá 2 hướng tiếp cận RAG + KG và fine-tuning + KG, từ đó đưa ra các đánh giá định lượng cụ thể nhằm chọn ra mô hình tối ưu cho hệ thống hỏi đáp tiếng Việt theo miền.

2.2 Tổng quan lý thuyết

2.2.1 Kiến thức về Hệ thống trả lời tự động

Hệ thống trả lời tự động (Question Answering: QA) được xây dựng để cung cấp thông tin cho người dùng thông qua việc phân tích câu hỏi và tìm kiếm để đưa ra câu trả lời. Là một vấn đề được nghiên cứu rất nhiều trong lĩnh vực Xử lý ngôn ngữ tự nhiên và có tính ứng dụng cao trong hiện nay.
Mô hình Question Answering (QA) có thể được sử dụng để tự động phản hồi câu hỏi từ người dùng bằng cách sử dụng cơ sở kiến thức đã được cung cấp.

Một số loại mô hình Question Answering (QA):

1. Phân loại theo cách thức tạo câu trả lời:

    - Extractive QA: Mô hình tìm kiếm và trích xuất ra câu trả lời từ dữ liệu được cung cấp. Do đó câu trả lời có thể nằm trong đoạn dữ liệu được cung cấp.
    - Open Generative QA: Mô hình tự tạo văn bản dựa trên dữ liệu được cung cấp.
    - Closed Generative QA: Đây là loại mô hình dữ liệu không được cung cấp sẵn. Câu trả lời sẽ được mô hình đưa ra dựa trên kiến thức đã được học trong quá trình huấn luyện.

2. Phân loại theo phạm vi kiến thức:

    - Closed-domain: Mô hình QA đưa ra câu trả lời trong một lĩnh vực cụ thể hoặc dùng để khai thác các kiến thức chuyên ngành.
    - Open-domain: Mô hình QA có thể đưa ra câu trả lời cho bất kỳ lĩnh vực nào với lượng dữ liệu và kiến thức lớn.

Một số ứng dụng của hệ thống QA:

- Xác thực thông tin bằng cách đặt ra câu hỏi Có/Không (Yes/No), Đúng/Sai (True/False) được ứng dụng trong việc kiểm chứng dữ liệu.

- Hỗ trợ kỹ thuật: Đưa ra câu trả lời nhanh chóng từ câu hỏi của người dùng, giúp tiết kiệm được thời gian, nhanh chóng và đem lại trải nghiệm tốt hơn cho khách hàng.
- Chăm sóc khách hàng: Với việc ứng dụng Chatbot tự động trả lời các câu hỏi thường gặp, hướng dẫn quy trình sử dụng dịch vụ hoặc hỗ trợ đặt hàng, từ đó nâng cao trải nghiệm khách hàng và giảm tải cho đội ngũ hỗ trợ trực tiếp.

2.2.2 Large Language Models (LLMs)

Khái niệm chung

Large Language Models (LLMs) là các mô hình ngôn ngữ có kích thước rất lớn, thường được huấn luyện trên hàng chục đến hàng trăm tỷ từ, với mục tiêu xử lý và sinh ngôn ngữ tự nhiên. Nhờ kích thước lớn và khả năng học ngữ cảnh sâu rộng, LLM có thể thực hiện nhiều tác vụ phức tạp như sinh văn bản, dịch máy, tóm tắt, trả lời câu hỏi, thậm chí là viết mã lập trình.

Các LLM tiêu biểu hiện nay bao gồm GPT-3/4 (OpenAI), BERT (Google), Claude (Anthropic), Gemini (Google DeepMind), LLaMA (Meta), Phi (Microsoft) và nhiều mô hình mã nguồn mở khác như Mistral, Falcon, BLOOM. Những mô hình này đã đạt được những bước tiến vượt bậc trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

Trong khuôn khổ đề tài này, LLM được sử dụng như một thành phần trung tâm trong hệ thống hỏi đáp, cụ thể là dùng để sinh câu trả lời tự nhiên dựa trên ngữ cảnh và dữ kiện liên quan.

Nguyên lý hoạt động

Kiến trúc Transformer

Cốt lõi của hầu hết các LLM hiện đại là kiến trúc **Transformer**, được giới thiệu bởi Vaswani et al. (2017) với khẩu hiệu nổi tiếng "Attention is All You Need". Transformer là mô hình xử lý chuỗi dữ liệu đầu tiên không dựa vào tuần tự (như RNN hay LSTM) mà sử dụng cơ chế **self-attention** để tính toán quan hệ giữa các phần tử trong chuỗi một cách song song và hiệu quả.

Kiến trúc Transformer bao gồm hai thành phần chính:

- **Encoder:** Mã hóa đầu vào thành các vector ngữ nghĩa
- **Decoder:** Giải mã để tạo ra đầu ra tuần tự, dựa trên đầu vào đã mã hóa

Tùy theo mục tiêu mô hình mà chỉ sử dụng encoder (như BERT), decoder (như GPT), hoặc cả hai (như T5, BART).

Cơ chế Self-Attention

Self-Attention là cơ chế cho phép mô hình tập trung vào các phần khác nhau của chuỗi để hiểu ngữ cảnh tốt hơn. Tại mỗi bước, mô hình tính toán trọng số thể hiện mức độ liên quan giữa một từ với tất cả các từ còn lại trong chuỗi, từ đó xác định thông tin nào cần chú ý khi sinh ra từ tiếp theo.

Self-Attention không chỉ giúp mô hình hiểu được mối quan hệ giữa các từ gần nhau, mà còn cả các từ cách xa nhau trong chuỗi, làm tăng đáng kể khả năng nắm bắt ngữ nghĩa dài hạn.

Biểu diễn ngôn ngữ

Trước khi đưa vào mô hình, văn bản được biến đổi thành các token (mã hóa từng từ, cụm từ hoặc ký tự), sau đó chuyển thành vector qua các lớp embedding. Quá trình này gọi là **tokenization** và **embedding**, giúp chuyển ngôn ngữ tự nhiên sang dạng số để mô hình có thể xử lý được.

Quá trình huấn luyện LLMs

Việc huấn luyện một LLM thường bao gồm hai giai đoạn chính:

- **Tiền huấn luyện (Pretraining):** Mô hình học từ một lượng lớn dữ liệu văn bản không gán nhãn, bằng cách thực hiện các nhiệm vụ như dự đoán từ tiếp theo (causal language modeling) hoặc điền vào chỗ trống (masked language modeling).
- **Tinh chỉnh (Fine-tuning):** Sau giai đoạn tiền huấn luyện, mô hình được tinh chỉnh trên các tập dữ liệu nhỏ hơn, có gán nhãn, để phù hợp với các tác vụ cụ thể như hỏi đáp, phân loại văn bản, tóm tắt,...

Ngoài ra, các mô hình như GPT-4 còn sử dụng kỹ thuật **RLHF (Reinforcement Learning from Human Feedback)**, tức là mô hình được tinh chỉnh thêm dựa trên phản hồi từ con người, giúp tăng độ tự nhiên, an toàn và đúng đắn của phản hồi do AI sinh ra.

Các mô hình LLM tiêu biểu

Tên mô hình	Tổ chức phát triển	Đặc điểm nổi bật
GPT-3/4	OpenAI	Mô hình sinh văn bản, hỗ trợ chat, viết mã
BERT	Google	Mô hình encoder, mạnh về hiểu ngôn ngữ
T5/BART	Google/Facebook	Mô hình encoder-decoder, mạnh về tóm tắt, QA
Claude	Anthropic	Nhấn mạnh sự an toàn và tuân thủ đạo đức
LLaMA	Meta	Mã nguồn mở, hiệu quả cao, nhẹ hơn GPT
Phi-3	Microsoft	Nhẹ, hiệu quả cao trên thiết bị tài nguyên thấp
Mistral	Mistral AI	Mô hình mã nguồn mở, tốc độ cao, hiệu quả tốt

Bảng 1: Một số mô hình LLM tiêu biểu hiện nay (trích từ [21])

Ứng dụng thực tế của LLMs

LLMs được ứng dụng trong nhiều lĩnh vực của đời sống và công nghệ:

- **Chatbot và trợ lý ảo:** Trả lời câu hỏi, hỗ trợ người dùng, điều khiển bằng giọng nói
- **Sinh nội dung:** Viết bài báo, nội dung marketing, email, tài liệu kỹ thuật
- **Tóm tắt và dịch văn bản:** Hiểu và rút gọn nội dung, hỗ trợ dịch đa ngôn ngữ
- **Trợ lý lập trình:** Sinh code, gợi ý sửa lỗi, giải thích thuật toán (như GitHub Copilot)
- **Hỏi đáp dựa trên tài liệu:** Truy xuất thông tin và sinh câu trả lời tự nhiên
- **Ứng dụng chuyên ngành:** Y tế, pháp lý, giáo dục, du lịch thông minh,...

Hạn chế và thách thức

Mặc dù LLMs có tiềm năng lớn, nhưng vẫn tồn tại nhiều vấn đề cần được khắc phục:

- **Ảo giác (Hallucination):** Mô hình có thể tạo ra thông tin sai lệch, không đúng thực tế.
- **Thiên kiến (Bias):** Mô hình học các định kiến trong dữ liệu và có thể phản ánh chúng trong đầu ra.
- **Chi phí tính toán cao:** Việc huấn luyện và vận hành LLM yêu cầu tài nguyên tính toán lớn (GPU/TPU).

- **Khả năng kiểm soát hạn chế:** Khó kiểm soát đầu ra hoàn toàn, nhất là trong môi trường quan trọng.
- **Bảo mật và riêng tư:** Mô hình có thể ghi nhớ dữ liệu huấn luyện và rò rỉ thông tin nhạy cảm.

Do đó, việc áp dụng LLM vào hệ thống thực tế cần có sự đánh giá, kiểm định và kết hợp các kỹ thuật giám sát đầu ra một cách chặt chẽ.

Xu hướng phát triển của LLMs

Trong thời gian tới, LLMs tiếp tục phát triển theo các hướng:

- **Mã nguồn mở và tối ưu hóa mô hình nhỏ:** Các mô hình nhỏ nhưng hiệu quả cao như LLaMA, Phi, Mistral giúp LLM dễ tiếp cận hơn.
- **Triển khai trên thiết bị biên (edge):** Chạy LLM trực tiếp trên điện thoại, trình duyệt, thiết bị IoT.
- **Kết hợp đa mô thức (multimodal):** Kết hợp ngôn ngữ với hình ảnh, âm thanh, video (như GPT-4 Vision).
- **Cá nhân hóa và kiểm soát đạo đức:** Hướng tới mô hình hiểu người dùng và phản hồi an toàn, đáng tin cậy.
- **Kết hợp với cơ sở tri thức (Retrieval-Augmented Generation - RAG):** Sử dụng kiến thức ngoài để tăng tính chính xác.

2.2.3 Knowledge Graph (Đồ thị tri thức)

Đồ thị tri thức là cơ sở tri thức được biểu diễn dưới dạng cấu trúc đồ thị. Cấu trúc này mô tả mối quan hệ giữa các thực thể - đối tượng, sự vật, khái niệm trừu tượng.
Mỗi đơn vị tri thức trong đồ thị được biểu diễn bằng bộ ba (subject, relation, object).

Lý thuyết đồ thị

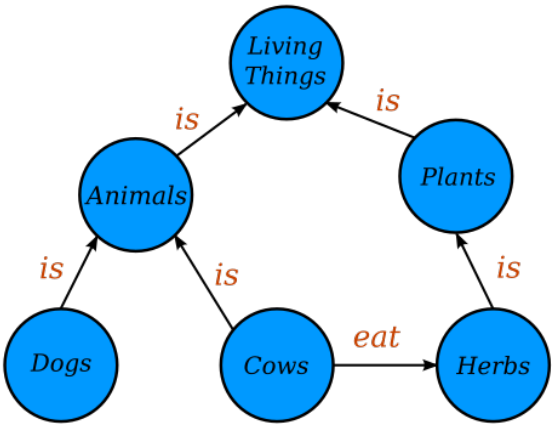
Đồ thị được định nghĩa là một cặp (V, E) trong đó:

- V (Vertices): tập hợp các đỉnh hoặc các nút (nodes) là các thực thể của đồ thị.
- E (Edges): tập hợp các cạnh hoặc các liên kết nối giữa 2 đỉnh, biểu thị mối quan hệ giữa các thực thể.

Cạnh trong đồ thị có thể là cạnh có hướng và không hướng (Directed vs. Undirected Edges):

- Cạnh không hướng: là cạnh nối giữa 2 đỉnh A và B mà từ đỉnh A có thể đi đến đỉnh B và từ đỉnh B có thể đi đến đỉnh A.
- Cạnh có hướng: thể hiện quan hệ một chiều, có thể đi từ đỉnh A đến đỉnh B, nhưng không thể đi ngược lại.

Đường đi của đồ thị là một chuỗi liên kết các đỉnh thông qua các cạnh.



Hình 1. Sơ đồ khái niệm về Đồ thị tri thức (trích từ [22])

Ví dụ minh họa về một đồ thị tri thức đơn giản được thể hiện trong Hình 1. Trong hình, các thực thể như *Dogs*, *Cows*, *Herbs*, *Animals*, *Plants* được kết nối thông qua các cạnh có hướng với nhãn quan hệ như *is*, *eat*. Chẳng hạn, cạnh có hướng từ *Cows* đến *Herbs* với quan hệ *eat* thể hiện rằng "Bò ăn cỏ", trong khi cạnh từ *Animals* đến *Living Things* biểu diễn rằng "Động vật là một loại sinh vật".

Biểu diễn Tri thức (Knowledge Representation)

Biểu diễn tri thức được hiểu là quá trình chuyển tri thức của con người về dạng dữ liệu mà máy tính có thể hiểu, xử lý, suy luận được.
Đồ thị tri thức là một dạng nâng cao của mạng ngữ nghĩa (Semantic Network) dưới dạng bộ ba (triple): chủ thể, quan hệ, đối tượng (subject, relation, object).

- Ngữ nghĩa (Semantic): Đồ thị tập trung biểu diễn ý nghĩa mối quan hệ giữa các thực thể, hiểu được chúng.
- Thể hiện (Instantiation) và Khái niệm (Concept):
  - Khái niệm/Lớp (Class/Concept): Là một tập hợp các thực thể có chung đặc điểm hoặc thuộc tính. Ví dụ: "Người", "Thành phố", "Sách".

  - Thể hiện/Cá thể (Instance/Individual): Là một đối tượng cụ thể thuộc về một khái niệm/lớp. Ví dụ: "Thỏ"là một thể hiện của lớp "Động vật"; "Sài Gòn"là một thể hiện của lớp "Thành phố". Đồ thị tri thức kết nối các thể hiện với các khái niệm của chúng.

- Ontology (Bản thể học): Mô hình phân cấp phức tạp của tri thức, định nghĩa lớp, thuộc tính và quan hệ.. Trong ngữ cảnh Đồ thị tri thức, ontology cung cấp cấu trúc xương sống, định nghĩa các loại thực thể (classes), các loại quan hệ (properties) và các ràng buộc ngữ nghĩa (constraints) mà dữ liệu trong đồ thị phải tuân theo. Nó giống như một "ngữ pháp"cho tri thức.

Đồ thị tri thức chính là sự kết hợp chặt chẽ giữa Lý thuyết Đồ thị và Biểu diễn Tri thức :

- Lý thuyết đồ thị cung cấp cấu trúc nền tảng (đỉnh và cạnh) để tổ chức và liên kết thông tin.

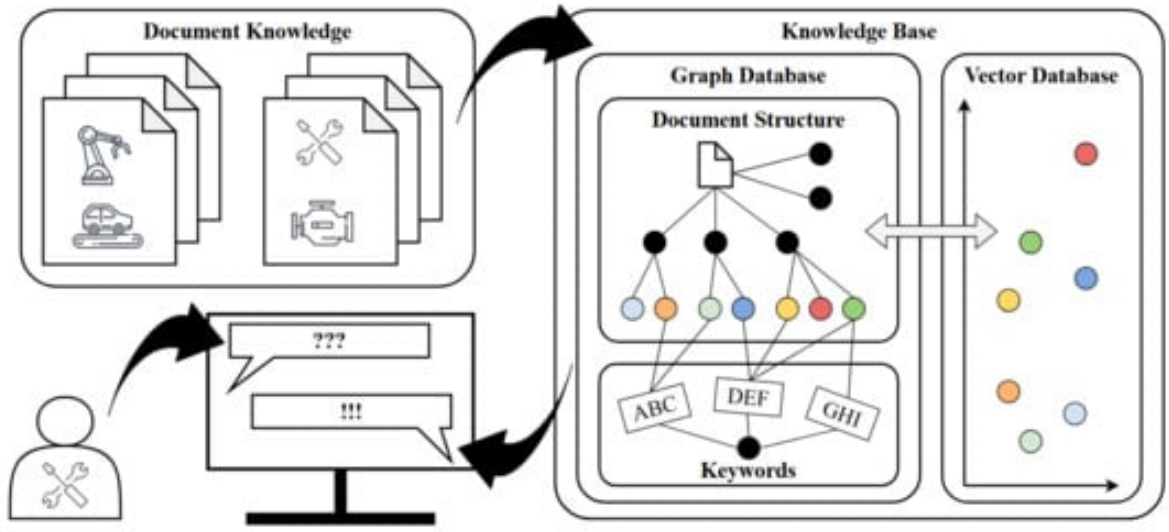
- Biểu diễn tri thức bổ sung ngữ nghĩa và ý nghĩa cho cấu trúc đó thông qua các bộ ba, khái niệm, quan hệ và ontology.

Quy trình xây dựng được minh họa trong Hình 2. Ban đầu, các tài liệu kỹ thuật hoặc chuyên ngành (ví dụ: hướng dẫn sử dụng, bản vẽ máy móc, quy trình sửa chữa, hay tài liệu lịch sử – du lịch) được xử lý để trích xuất ra cấu trúc nội dung, từ khóa, và các thực thể quan trọng. Thông tin này được lưu trữ đồng thời trong hai loại cơ sở dữ liệu:

- Graph Database: tổ chức tri thức dưới dạng cây phân cấp hoặc mạng lưới các node và quan hệ, phản ánh cấu trúc văn bản và các liên kết ngữ nghĩa.

- Vector Database: mã hóa các đoạn nội dung thành vector embedding, phục vụ cho việc truy vấn ngữ nghĩa theo độ tương đồng.

Khi người dùng nhập câu hỏi, hệ thống sẽ truy vấn đồng thời cả Graph Database và Vector Database để thu thập thông tin phù hợp. Các kết quả sau đó được tổng hợp, đưa vào mô hình sinh (LLM), từ đó trả về câu trả lời vừa chính xác theo ngữ cảnh, vừa bảo đảm khai thác được tri thức sâu từ dữ liệu gốc.



Hình 2. Sơ đồ kiến trúc hệ thống hỏi-đáp kết hợp giữa tri thức tài liệu và cơ sở tri thức (trích từ [23])

2.2.4 RAG architecture

Giới thiệu về Retrieval-augmented generation (RAG)

Retrieval-augmented generation (RAG) là một kiến trúc cho phép các mô hình ngôn ngữ lớn (LLM) kết hợp hai thành phần truy hồi (retriever) và sinh văn bản (generator). RAG sẽ tìm kiếm các văn bản có liên quan từ cơ sở dữ liệu, hoặc các nguồn tài liệu khác. Các văn bản này sau đó sẽ được đưa vào mô hình sinh để sinh ra câu trả lời hoàn chỉnh. Khác với các LLM truyền thống dựa trên dữ liệu huấn luyện tĩnh, RAG giúp khắc phục được hạn chế trong việc bám sát dữ liệu thực tế, giảm nhu cầu huấn luyện lại LLMs với dữ liệu mới, từ đó giảm được chi phí tính toán và tài chính.

Kiến trúc của RAG

1. **Mô hình truy hồi (Retriever)**

   - Mục đích: Truy xuất tài liệu (documents) để chọn ra các tài liệu liên quan nhất đến truy vấn của người dùng. Thông thường trong RAG, thường sử dụng mô hình Neural Retriever sử dụng mạng neural để thực hiện tìm kiếm.

     $$w_z^* = \arg\max_{w_z} P(w_z \mid q, G)$$

     Trong đó:

     - q: truy vấn đầu vào
     - w_z: một tài liệu trong cơ sở tri thức

- G: kho dữ liệu (knowledge base)
- $P(w_z \mid q, G)$: xác suất tài liệu w_z là phù hợp nhất với truy vấn q

- Cách thức hoạt động của Retriever:

(a) Tìm kiếm Vector (Vector Search):

- Đây là phương pháp tìm kiếm dựa trên vector bằng cách nhúng (embedding) tài liệu và truy vấn dưới dạng vector số học. Vector embedding được tạo bởi các mô hình học sâu như: BERT, RoBERTa, Word2Vec,....
- Tất cả tài liệu sau khi được chuyển đổi thành vector embedding sẽ được lưu trữ trong cơ sở dữ liệu vector (vector database).
- Sau đó tìm độ tương đồng giữa các vector để tìm độ tương đồng giữa tài liệu và truy vấn. Các phép đo độ tương đồng thường được sử dụng là: Consine, Euclid.
- Độ tương đồng Cosine:

$$\cos(\theta) = \frac{\vec{q} \cdot \vec{w_z}}{\|\vec{q}\| \cdot \|\vec{w_z}\|}$$

  Trong đó:

  * $\vec{q}$: vector biểu diễn truy vấn
  * $\vec{w_z}$: vector biểu diễn tài liệu w_z,
  * $\|\vec{q}\|, \|\vec{w_z}\|$: độ dài (chuẩn) của các vector.
- Top-k tài liệu có độ tương đồng cao nhất sẽ được trả về làm kết quả truy vấn.
- Việc sử dụng Vector Embedding có thể nắm bắt được ngữ cảnh của văn bản, hiểu được mối quan hệ giữa các từ trong câu để cho ra kết quả truy vấn chính xác. Ngoài ra còn có thể hiểu được ngữ nghĩa thực sự của từ (Semantic gap).

(b) Tìm kiếm dựa trên từ khoá (keyword-based search):

- Lập chỉ mục (Indexing): Là quá trình tạo danh sách chỉ mục các từ khoá quan trọng của tài liệu trong kho dữ liệu và tạo danh sách chỉ mục ngược (inverted index). Danh sách chỉ mục này ánh xạ từ khoá đến tài liệu chứa từ khoá đó.
- Truy vấn (Query): Trích xuất các từ khoá trong câu hỏi, sử dụng chỉ mục ngược để tìm ra các tài liệu chứa một hoặc nhiều từ khoá đó bằng thuật toán xếp hạng:
  * TF-IDF (Term Frequency-Inverse Document Frequency): Kỹ thuật đánh giá mức độ quan trọng của một từ trong tài liệu.

* BM25 (Okapi BM25): Cải tiến của TF-IDF, giúp tính toán độ phù hợp giữa câu truy vấn và các tài liệu.

$$\text{score}(d,q) = \sum_{i=1}^{|q|} \text{IDF}(q_i) \cdot \frac{f(q_i,d) \cdot (k_1+1)}{f(q_i,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}$$

Trong đó:

· $f(q_i, d)$: tần suất xuất hiện của từ q_i trong tài liệu d

· $|d|$: độ dài tài liệu

· avgdl: độ dài trung bình của các tài liệu trong tập

· k_1, b: các tham số điều chỉnh.

(c) Mô hình Bi-encoder và Cross-encoder:

i. Bi-encoder:

– Mô hình gồm 2 bộ mã hoá (encoder), một bộ mã hoá câu truy vấn của người dùng sang vector embedding, một bộ mã hoá mỗi tài liệu trong kho dữ liệu sang vector embedding.

– Các phép khoảng cách giữa 2 vector được sử dụng để tìm ra các tài liệu liên quan nhất với câu hỏi của người dùng.

– Do dữ liệu được chuyển đổi sang vector embedding nên việc tìm kiếm diễn ra rất nhanh. Nhưng thay vào đó là yêu cầu bộ nhớ lớn để lưu trữ tất cả embedding của tài liệu.

– Ngoài ra, mô hình Bi-encoder mã hoá độc lập truy vấn và tài liệu do đó, thiếu hiểu biết và mối quan hệ ngữ nghĩa giữa chúng.

ii. Cross-encoder:

– Mô hình kết hợp câu truy vấn và tài liệu để đưa vào bộ mã hoá (encoder) duy nhất.

– Cross-encoder sẽ tính toán điểm số tương quan cho từng cặp (câu hỏi, tài liệu) bằng việc xem xét sự tương tác giữa từng token của câu truy vấn và từng token của tài liệu.

– Cross-encoder cho kết quả truy vấn chính xác hơn Bi-encoder nhờ vào việc hiểu sâu hơn về mối quan hệ ngữ nghĩa giữa câu truy vấn và tài liệu.

– Tuy nhiên, mô hình tốn kém thời gian để tính toán dẫn đến hiệu suất thấp với kho dữ liệu lớn.

- Kho dữ liệu (Knowledge Base): Đây là nơi lưu trữ tri thức mà RAG truy cập để trả lời câu hỏi của người dùng. Tuỳ thuộc vào ứng dụng và loại thông tin mà kho dữ liệu được tổ chức với nhiều hình thức khác nhau, như:

- Knowledge Graph (Đồ thị tri thức): Chuyển đổi tài liệu văn bản, sách,.. sang dạng đồ thị và được lưu trữ trong vector database. (GraphRAG).
- Tài liệu văn bản: PDF, DOC, TXT files, HTML files,...
- Dữ liệu dạng bảng: CSV, Excel
- Cơ sở dữ liệu: SQL, MongoDB,...

2. Mô hình Tạo sinh (Generator/Large Language Model - LLM)

- Mục đích: Mô hình tạo sinh trong RAG thường là mô hình LLM được sử dụng để đưa ra phản hồi cho người dùng dựa trên câu hỏi của người dùng và các tài liệu liên quan được trích xuất từ Retriver.
- Cách thức hoạt động của LLM trong RAG:
  (a) Kết hợp câu truy vấn của người dùng q với các đoạn văn bản (chunks) từ Retriever w_z thành dữ liệu đầu vào $[q; w_z]$
  (b) Đưa dữ liệu đầu vào sau khi kết hợp vào Transformer với cơ chế Attention giúp mô hình hiểu được các mối quan hệ giữa các từ trong câu hỏi, giữa câu hỏi và ngữ cảnh, giữa các phần khác nhau của ngữ cảnh.
  (c) LLM tạo phản hồi cho người dùng bằng cách tạo ra tuần tự từng token với mỗi token được dự đoán dựa trên dữ liệu đầu vào và các token đã tạo ra trước đó. Ở mỗi bước j, mô hình tính xác suất sinh token a_j dựa trên các token đã sinh trước đó $a_<j$, truy vấn q, và tri thức w_z:

$$P(a_j \mid a_{<j}, q, w_z)$$

Quá trình sinh toàn bộ câu trả lời $a = (a_1, a_2, \ldots, a_m)$ được mô hình hóa như sau:

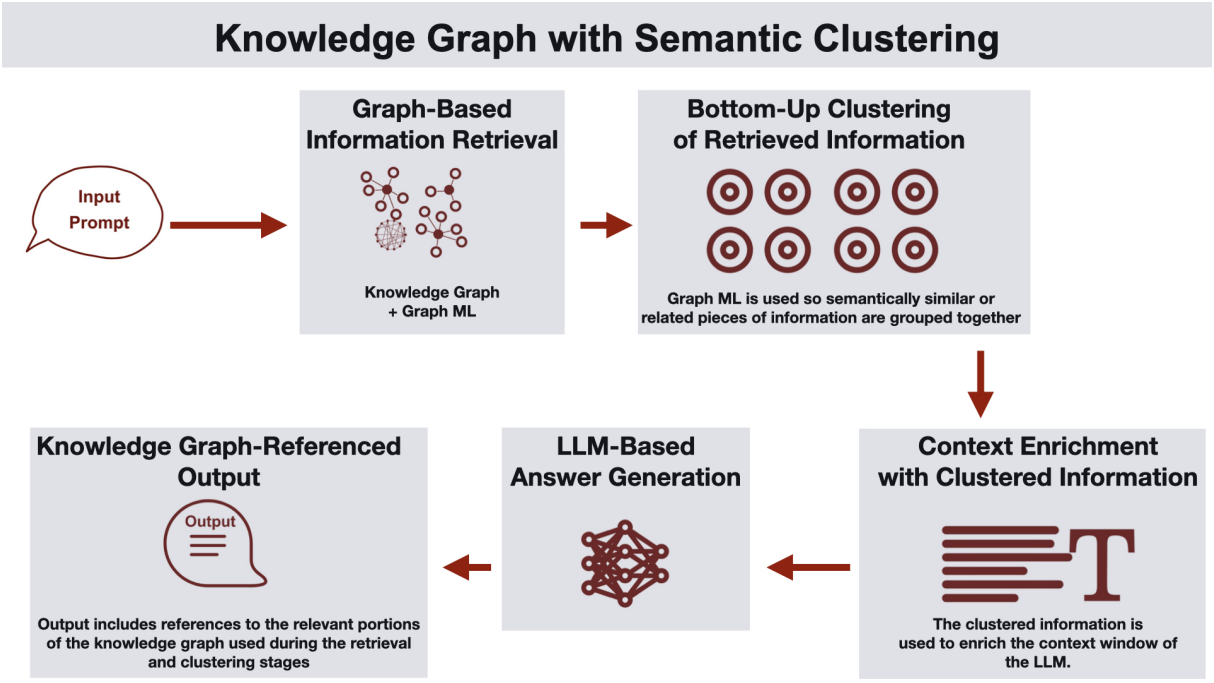
$$a^* = \arg\max_{a \in \mathcal{A}} P(a \mid q, w_z) = \arg\max_{a \in \mathcal{A}} \prod_{j=1}^{m} P(a_j \mid a_{<j}, q, w_z)$$

Trong đó:

- a^*: câu trả lời tối ưu
- $\mathcal{A}$: không gian các câu trả lời có thể sinh ra
- m: số lượng token trong câu trả lời
- a_j: token tại vị trí thứ j
- $a_{<j}$: chuỗi các token đã sinh trước bước j.

Quy trình hoạt động của RAG:

RAG kết hợp truy vấn của người dùng với các tài liệu liên quan được truy xuất từ kho tri thức. Sau đó, các thông tin này được đưa vào mô hình ngôn ngữ để sinh ra câu trả lời phù hợp, chính xác và có ngữ cảnh.



Hình 3. Kiến trúc RAG (Retrieval-Augmented Generation) kết hợp với Knowledge Graph (KG) (trích từ [24])

1. Input Prompt: Người dùng đưa vào câu hỏi hoặc truy vấn.
2. Graph-Based Information Retrieval: Sử dụng Knowledge Graph kết hợp với Graph Machine Learning (Graph ML) để truy xuất thông tin liên quan đến truy vấn.
3. Bottom-Up Clustering of Retrieved Information: Dùng Graph ML để phân cụm thông tin theo ngữ nghĩa.
4. Context Enrichment with Clustered Information: Các cụm thông tin được sử dụng để mở rộng ngữ cảnh đầu vào cho LLM (giúp tăng độ liên quan khi sinh văn bản).
5. LLM generation: Mô hình sinh trả lời dựa trên truy vấn đã được tăng cường.
6. Knowledge Graph-Referenced Output: Kết quả đầu ra có thể bao gồm cả thông tin tham chiếu đến các phần của KG đã sử dụng (giải thích truy vết nguồn gốc).

So sánh RAG với LLM truyền thống:

Các mô hình ngôn ngữ lớn (LLM) truyền thống thường được huấn luyện trên một tập dữ liệu cố định và không thể cập nhật tri thức mới sau khi huấn luyện. Trong khi đó,

kiến trúc RAG (Retrieval-Augmented Generation) cho phép mô hình kết hợp với nguồn dữ liệu bên ngoài (như cơ sở tri thức hoặc đồ thị tri thức), từ đó nâng cao độ chính xác và tính linh hoạt.

Bảng 2 trình bày sự khác biệt giữa LLM truyền thống và kiến trúc RAG dựa trên một số tiêu chí chính.

Tiêu chí	**LLM truyền thống**	**RAG**
Dữ liệu sử dụng	Tĩnh, cố định trong quá trình huấn luyện	Cập nhật từ kho tri thức bên ngoài
Độ chính xác	Hạn chế với kiến thức mới	Cao hơn nhờ truy hồi thông tin
Tính linh hoạt	Kém	Cao
Chi phí fine-tune	Cao	Giảm đáng kể

Bảng 2: So sánh giữa LLM truyền thống và kiến trúc RAG

2.2.5 So sánh giữa RAG và Fine-tuning

Để lựa chọn phương pháp xây dựng hệ thống hỏi đáp cần phải đánh giá và so sánh các kỹ thuật được dùng để triển khai hệ thống. Dưới đây là bảng so sánh giữa hai kỹ thuật: huấn luyện lại mô hình ngôn ngữ (Fine-tuning) và sử dụng kiến trúc RAG (Retrieval-Augmented Generation):

Tiêu chí	**Fine-tuning**	**RAG**
Độ chính xác	Tùy thuộc vào dữ liệu huấn luyện	Tùy thuộc vào dữ liệu truy hồi
Khả năng cập nhật tri thức	Yêu cầu retrain lại mô hình	Cập nhật kho dữ liệu
Chi phí	Tốn tài nguyên (GPU)	Thấp hơn nhiều vì không cần huấn luyện lại
Nguy cơ "ảo giác"	Cao nếu dữ liệu huấn luyện thiếu hoặc lệch sẽ tạo ra thông tin không chính xác.	Thấp do cần sử dụng tri thức từ nguồn dữ liệu đã truy hồi.

Bảng 3: So sánh giữa RAG và Fine-tuning (phỏng theo [25])

Từ bảng so sánh trên, nhận thấy kiến trúc RAG ưu điểm vượt trội hơn so với Fine-tuning: RAG không yêu cầu huấn luyện lại mô hình ngôn ngữ, có khả năng cập nhật tri thức tức thời thông qua đồ thị tri thức và phù hợp hơn với tài nguyên hiện có. Đồng thời, RAG dễ dàng tích hợp với hệ thống Knowledge Graph đã được xây dựng, giúp hệ thống trả lời chính xác hơn, có căn cứ rõ ràng.

Giả sử trong trường hợp hệ thống hỏi đáp về lịch sử - du lịch Việt Nam hiện tại tập trung vào lịch sử lớp 12, nội dung cố định theo sách giáo khoa. Thì khi đó, mô hình Fine-tune sẽ thích hợp hơn vì mô hình sẽ học sâu và trả lời nhanh, chính xác với dữ liệu trong sách giáo khoa.

3 Xây dựng hệ thống trả lời tự động về Lịch sử - Du lịch Việt Nam

3.1 Tổng quan hệ thống

...

3.2 Chi tiết xây dựng

3.2.1 Tiền xử lý dữ liệu lịch sử

Một trong những nguồn dữ liệu lịch sử chính được sử dụng trong nghiên cứu là bộ sách *Lịch sử Việt Nam (15 tập)*, do Viện Hàn lâm Khoa học Xã hội Việt Nam chủ trì biên soạn. Đây là một công trình nghiên cứu đồ sộ, phản ánh toàn diện tiến trình phát triển của dân tộc Việt Nam từ thời tiền sử đến đầu thế kỷ XXI.

Giới thiệu bộ sách

Bộ sách gồm 15 tập, được biên soạn bởi các nhà sử học hàng đầu, với mục tiêu cung cấp một hệ thống tri thức lịch sử chính thống, khoa học và có độ tin cậy cao. Nội dung được chia theo tiến trình lịch sử, phản ánh đầy đủ các giai đoạn phát triển của quốc gia, dân tộc, từ cấu trúc chính trị, kinh tế, văn hóa đến các phong trào cách mạng và hội nhập quốc tế.

Cấu trúc bộ sách

Tập	Giai đoạn lịch sử	Nội dung chính
1	Thời tiền sử – TK I	Văn hóa Đông Sơn, Phùng Nguyên, thời Hùng Vương
2	Bắc thuộc (179 TCN – 938)	Kháng chiến chống đô hộ phương Bắc, Ngô Quyền
3	TK X – XIII	Nhà Ngô, Đinh, Tiền Lê, Lý
4	TK XIII – XV	Nhà Trần, ba lần kháng Nguyên, Hồ Quý Ly
5	TK XV – XVI	Lê sơ, Lê Thánh Tông, mở rộng lãnh thổ
6	TK XVI – XVIII	Trịnh – Nguyễn phân tranh, biến động xã hội
7	Cuối TK XVIII – XIX	Khởi nghĩa Tây Sơn, nhà Nguyễn thống nhất
8	Giữa TK XIX – 1884	Xâm lược của Pháp, mất nước
9	1885 – 1925	Cần Vương, phong trào yêu nước đầu thế kỷ XX
10	1925 – 1945	Sự ra đời ĐCSVN, Cách mạng Tháng Tám
11	1945 – 1954	Kháng chiến chống Pháp, Điện Biên Phủ
12	1954 – 1975	Chia cắt đất nước, kháng chiến chống Mỹ
13	1975 – 1986	Giai đoạn hậu chiến, tái thiết đất nước
14	1986 – 2000	Đổi mới, cải cách kinh tế, hội nhập
15	2000 – đầu TK XXI	Thách thức toàn cầu hóa và phát triển bền vững

Ý nghĩa đối với xử lý dữ liệu lịch sử

Bộ sách cung cấp hệ thống thông tin lịch sử phong phú, có thể trích xuất và mã hóa dưới dạng dữ liệu có cấu trúc để phục vụ các bài toán xử lý ngôn ngữ tự nhiên (NLP), trích xuất thực thể lịch sử (NER), xây dựng mốc sự kiện và phát triển đồ thị tri thức trong lĩnh vực lịch sử. Do có nội dung chuẩn hóa, rõ ràng theo dòng thời gian và sự kiện, đây là một nguồn dữ liệu lý tưởng để huấn luyện mô hình học máy hoặc làm dữ liệu gốc trong các hệ thống hỏi–đáp (QA) về lịch sử Việt Nam.

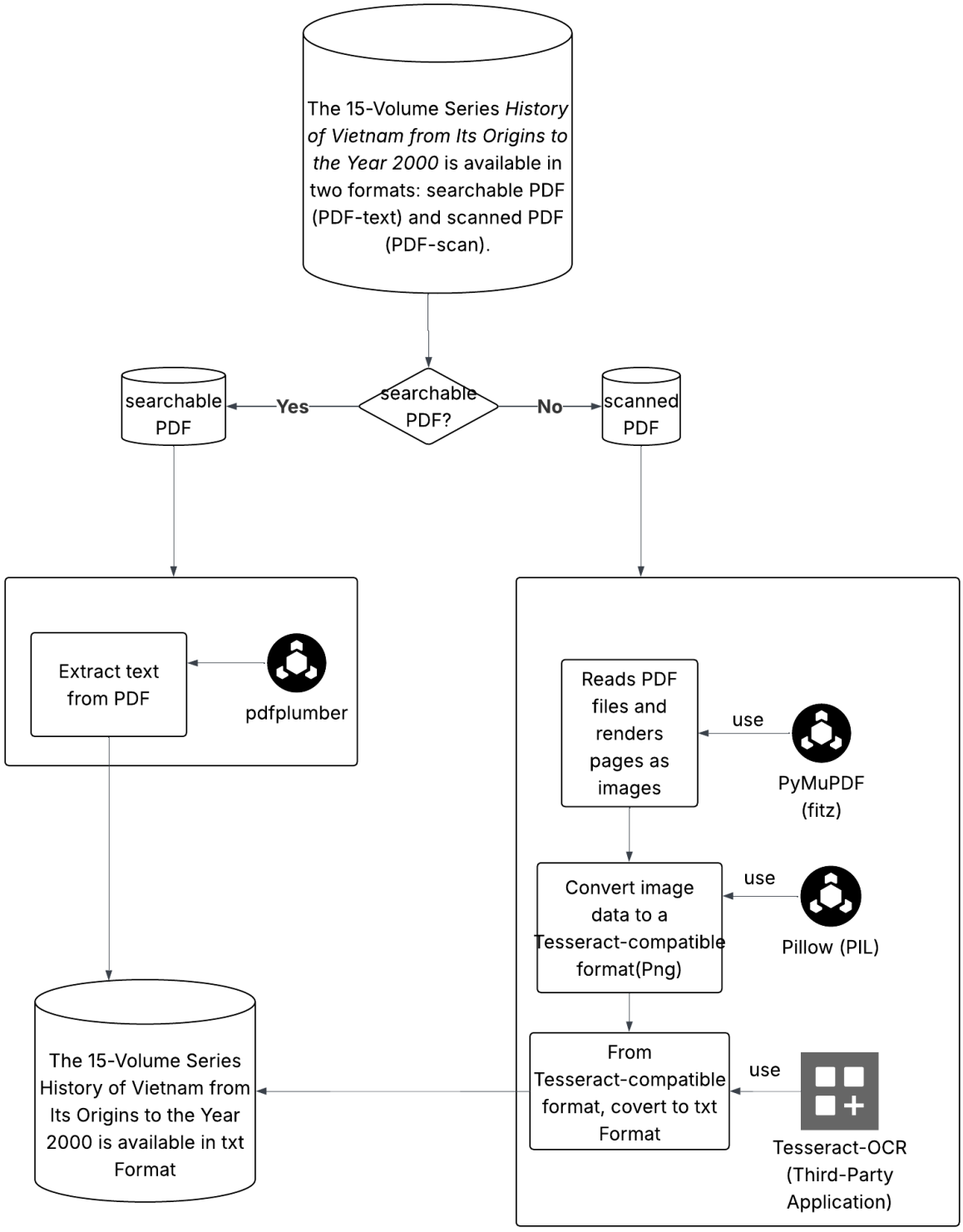
Quy trình chuẩn hóa dữ liệu lịch sử

Để đưa dữ liệu lịch sử từ nguồn tài liệu dạng PDF (cụ thể là bộ sách *Lịch sử Việt Nam (15 tập)*) vào hệ thống xử lý, cần tiến hành một chuỗi các bước chuẩn hóa nhằm chuyển đổi dữ liệu từ định dạng văn bản không cấu trúc thành dạng có cấu trúc, sẵn sàng cho các mô hình học máy và xử lý ngôn ngữ tự nhiên. Quy trình này gồm ba bước chính như sau:

- **Bước 1: Chuyển đổi từ PDF sang văn bản thuần (TXT)**

  Ở bước đầu tiên, các tệp PDF chứa nội dung lịch sử được chuyển đổi sang định dạng văn bản thuần (`.txt`) để thuận tiện cho việc xử lý tự động. Quá trình này cần

đảm bảo giữ lại cấu trúc văn bản hợp lý (tiêu đề, đoạn, mục lục...) và hạn chế lỗi ký tự. Lưu đồ minh họa được trình bày tại Hình 4.



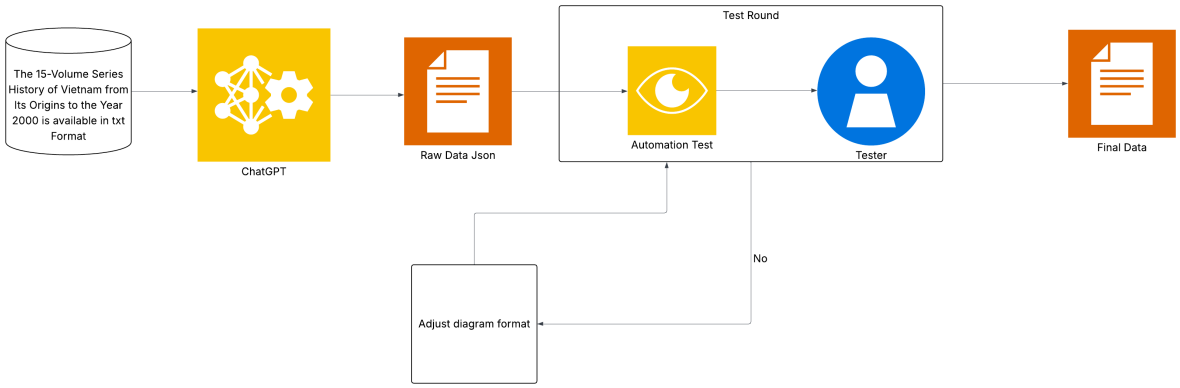
Hình 4. Lưu đồ chuyển đổi dữ liệu từ định dạng PDF sang TXT

- **Bước 2: Trích xuất dữ liệu có cấu trúc (JSON) bằng mô hình ngôn ngữ**

  Sau khi có được nội dung văn bản thuần, dữ liệu sẽ được xử lý bằng mô hình ngôn

ngữ ChatGPT nhằm trích xuất các thực thể lịch sử và sự kiện theo định dạng có cấu trúc (ví dụ: JSON). Việc sử dụng mô hình ngôn ngữ giúp rút ngắn thời gian phân tích và đảm bảo mức độ khái quát cao trong nhận diện thông tin. Lưu đồ xử lý được minh họa trong Hình 5.

Tuy nhiên, do quá trình sinh dữ liệu tự động có thể phát sinh các lỗi như sai cú pháp JSON, thiếu trường dữ liệu, hoặc gán nhầm loại thực thể, nên sau khi ChatGPT tạo đầu ra, người kiểm thử (Tester) sẽ đánh giá kết quả và xác nhận dữ liệu nào đạt yêu cầu, dữ liệu nào cần điều chỉnh. Với các trường hợp sai hoặc chưa đúng định dạng, dữ liệu sẽ được chuyển trở lại bước điều chỉnh định dạng (Adjust diagram format) để hiệu chỉnh thủ công hoặc bán tự động, sau đó mới quay lại vòng kiểm thử. Chỉ những dữ liệu đã vượt qua cả hai bước kiểm tra tự động và thủ công mới được chấp nhận làm Final Data.



Hình 5. Lưu đồ chuyển đổi từ TXT sang JSON sử dụng mô hình ngôn ngữ

- **Bước 3: Đưa dữ liệu vào mô hình học máy**

  Sau khi đã chuẩn hóa và trích xuất, dữ liệu JSON sẽ được sử dụng làm đầu vào cho các mô hình học máy (machine learning), phục vụ các bài toán như phân loại, trích xuất thực thể lịch sử, xây dựng đồ thị tri thức hoặc hệ thống hỏi–đáp. Bước này yêu cầu định dạng dữ liệu đầu vào phải thống nhất và chất lượng cao nhằm đảm bảo hiệu quả huấn luyện mô hình.

```
1 [
2  {
3     "question": "Người chết được chôn theo tư thế nào trong các ngôi mộ
      ↪  Quỳnh Văn?",
```

```
    "answer": "Người chết được chôn theo tư thế ngồi xổm."
  },
  {
    "question": "Thành phần chủng tộc của các mộ táng ở Quỳnh Văn được
    ↪ xác định dựa trên những di cốt nào?",
    "answer": "Thành phần chủng tộc các mộ táng ở Quỳnh Văn dựa trên 2
    ↪ di cốt sọ thuộc chủng Australoid và có một vài nét của chủng
    ↪ Mongoloid."
  }
   {
    "question": "Điều gì đặc biệt về một ngôi mộ duy nhất được tìm thấy
    ↪ ở Quỳnh Văn?",
    "answer": "Có một ngôi mộ duy nhất chôn 2 người, kèm theo công cụ
    ↪ lao động và đồ trang sức."
  },
  {
    "question": "Hình thức mộ táng 'ngồi xổm' của người Quỳnh Văn cho
    ↪ thấy mối quan hệ gần gũi với những nền văn hóa nào?",
    "answer": "Hình thức mộ táng chôn người 'ngồi xổm' cho thấy mối quan
    ↪ hệ truyền thống gần gũi của người Quỳnh Văn với người Hòa Bình
    ↪ và Văn hóa Đa Bút."
  }
]
```

3.2.2 Tiền xử lý dữ liệu du lịch

Để xây dựng một hệ thống hỏi đáp về các địa điểm du lịch tại Việt Nam có liên hệ với dữ liệu lịch sử, chúng tôi đã tiến hành thu thập và chuẩn hóa một tập dữ liệu du lịch từ trang thông tin chính thức `https://csdl.vietnamtourism.gov.vn/dest`. Đây là một nguồn dữ liệu đáng tin cậy được vận hành bởi Tổng cục Du lịch Việt Nam, cung cấp thông tin về các điểm đến văn hoá, lịch sử, và danh lam thắng cảnh trên cả nước.

Mục tiêu lựa chọn dữ liệu:

Dữ liệu này được lựa chọn vì nó cung cấp mô tả chi tiết về các điểm đến gắn liền với lịch sử và văn hoá, có thể tích hợp với tập dữ liệu lịch sử Việt Nam để xây dựng một đồ thị tri thức và hệ thống hỏi đáp thông minh.

Cấu trúc dữ liệu:

Mỗi mục dữ liệu (item) tương ứng với một điểm đến, chứa các thông tin như:

- **ID**: mã định danh duy nhất cho mỗi điểm.
- **Tiêu đề (title)**: tên địa danh.
- **Mô tả (description)**: đoạn văn bản mô tả địa điểm, thường bao gồm yếu tố lịch sử, văn hoá, và kiến trúc.
- **Thông tin phụ trợ (metadata)**: các trường như địa chỉ, khu vực, loại hình di tích,...

Quy trình thu thập và chuẩn hóa dữ liệu:

Quá trình tiền xử lý được thực hiện qua các bước sau:

- **Bước 1: Cào dữ liệu từ Website**
  Sử dụng thư viện `requests` và `BeautifulSoup` trong Python, chúng tôi truy cập tuần tự vào các địa chỉ có dạng:

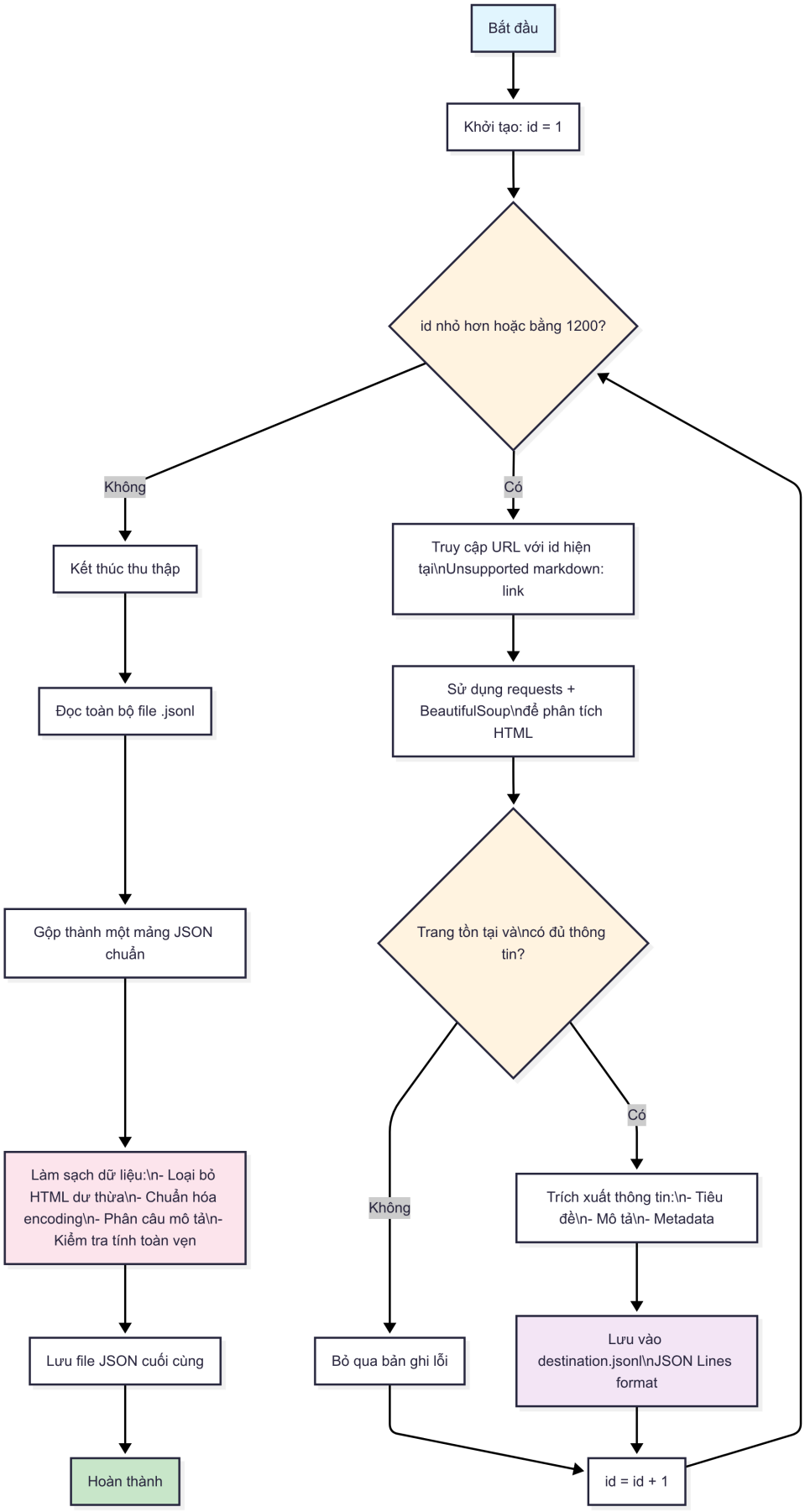
  `https://csdl.vietnamtourism.gov.vn/dest/?item={id}`

  Với `id` chạy từ 1 đến 1200. Nhóm phân tích cấu trúc HTML để trích xuất các trường thông tin cần thiết như tiêu đề, mô tả, và metadata.

- **Bước 2: Lưu dữ liệu dạng JSON Lines**
  Mỗi điểm đến được lưu ở dạng một dòng JSON trong file `destination.jsonl`, giúp dễ dàng ghi nối tiếp mà không mất toàn bộ file khi lỗi.

- **Bước 3: Gộp thành một file JSON chuẩn hoá**
  Sau khi thu thập đủ, toàn bộ dữ liệu được đọc lại từ `.jsonl` và gộp thành một mảng JSON để dễ dàng xử lý bởi các mô hình học máy hoặc hệ thống tìm kiếm.

- **Bước 4: Làm sạch và kiểm tra**
  Trong quá trình thu thập, các bản ghi lỗi (ví dụ không tồn tại hoặc thiếu trường thông tin) được loại bỏ. Các bước làm sạch có thể được mở rộng ở bước sau như loại bỏ HTML dư thừa, chuẩn hoá encoding, phân câu mô tả, v.v.



Hình 6. Flowchart mô tả quy trình tiền xử lý dữ liệu

Ví dụ một mẫu dữ liệu du lịch được cào về và lưu trong file JSON:

```
[
  {
    "id": 1,
    "title": "Văn Miếu - Quốc Tử Giám",
    "description": "Văn Miếu được xây dựng từ năm 1070 (tức năm Thần
    ↪  Vũ)...",
    "metadata": {}
  }
]
```

Toàn bộ quá trình giúp xây dựng một tập dữ liệu du lịch có cấu trúc, phục vụ cho việc tích hợp vào hệ thống đồ thị tri thức và trả lời câu hỏi kết hợp giữa dữ kiện lịch sử và địa danh văn hoá.

3.2.3 Xây dựng đồ thị tri thức

Xác định nguồn tri thức đầu vào:

- Tài liệu lịch sử lấy từ Bộ sách "Lịch sử Việt Nam" đã được chuyển sang .txt (xem mục 3.2.1).
- Tài liệu du lịch lấy từ website Cơ sở dữ liệu du lịch Việt Nam đã được chuyển sang file json. (xem mục 3.2.2).

Các văn bản này đã được xử lý loại bỏ các heading, ký tự, mục lục, phụ lục không cần thiết.

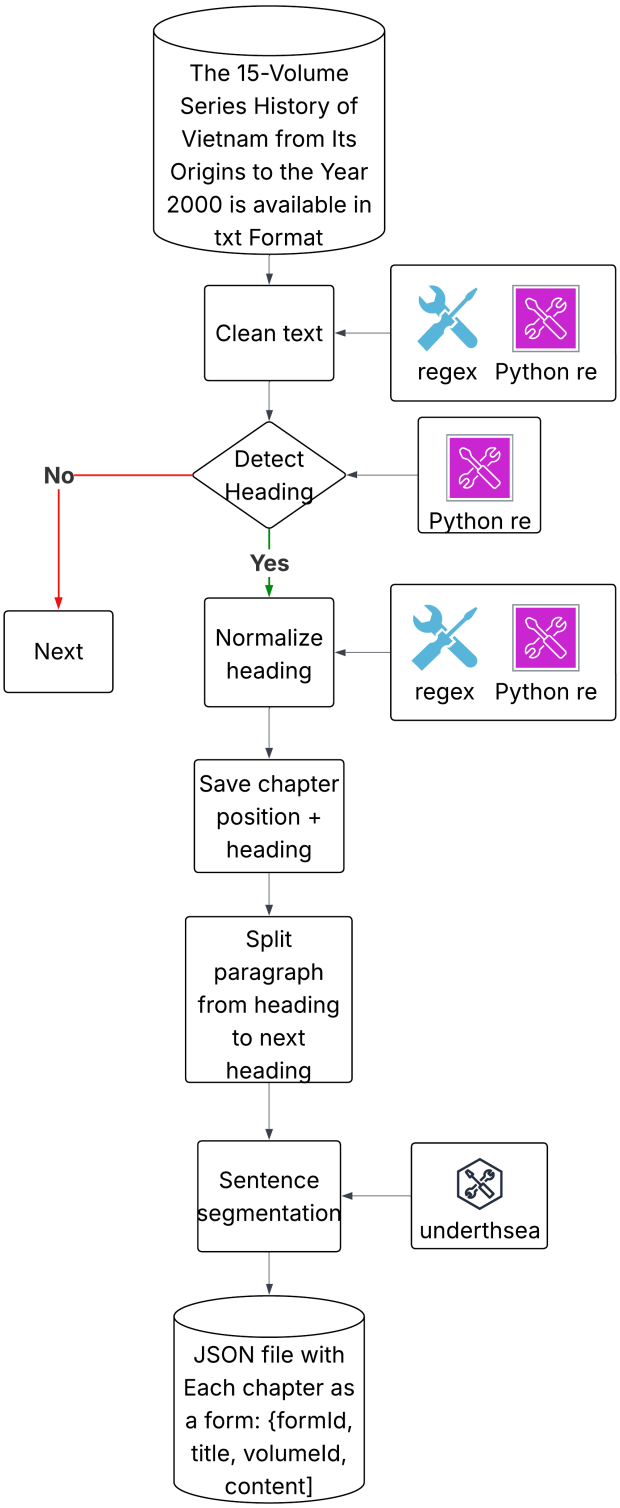
Bước 1: Phân đoạn nội dung thành các "form"và các "chunk"

Đối với dữ liệu du lịch, thông tin đầu vào đã được chuẩn hóa sẵn dưới dạng cấu trúc JSON, trong đó mỗi phần tử tương ứng với một địa điểm du lịch riêng biệt. Do đó, mỗi địa điểm này được xem như một form, chứa đầy đủ thông tin mô tả về vị trí, đặc điểm, lịch sử và giá trị văn hóa của địa điểm đó.

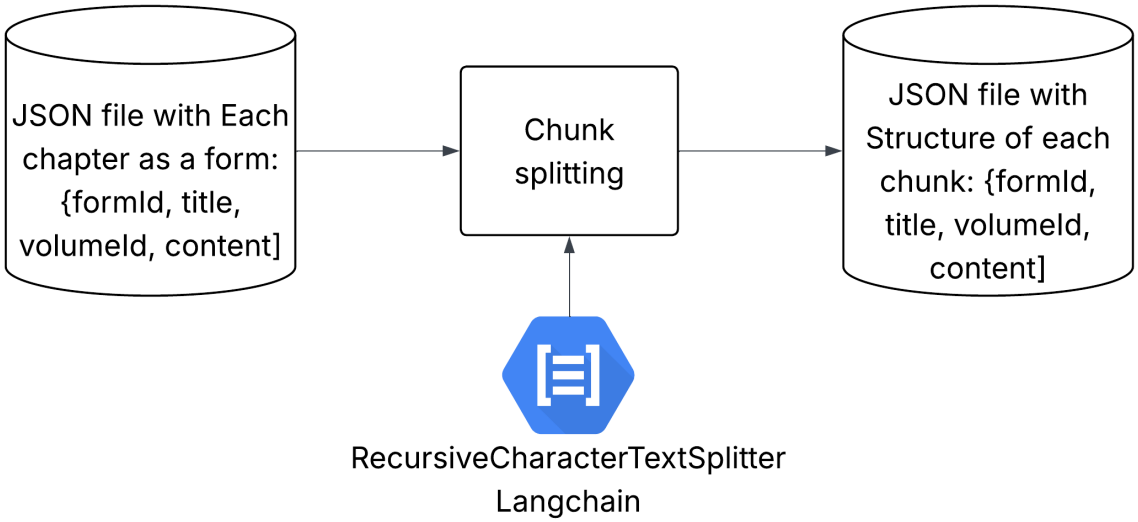
Ngược lại, đối với dữ liệu lịch sử đầu vào là văn bản dạng txt, do đó để tối ưu hóa trong việc truy hồi tri thức thì dữ liệu sẽ được phân thành hai cấp:

- **Form:** đại diện cho một chương trong dữ liệu, gồm: `formId`, `title`, `volumeId`, `source`, `content`.
- **Chunk:** đoạn văn nhỏ hơn từ form, độ dài 1000 từ, gồm: `chunkId`, `chunkSeqId`, `formId`, `text`.

Quy trình phân đoạn nội dung lịch sử thành các form được mô tả trong Hình 7. Sau khi xác định heading, hệ thống chuẩn hóa, lưu trữ vị trí chương, và tiến hành tách văn bản thành các form. Tiếp đó, quá trình tách nhỏ form thành các chunk bằng `RecursiveCharacterTextSplitter` trong thư viện LangChain được minh họa ở Hình 8.



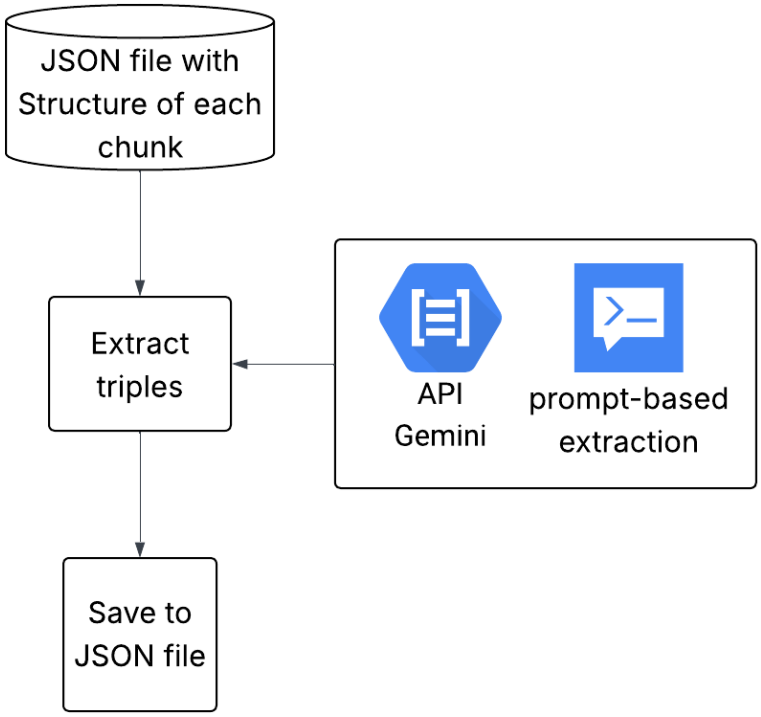
Hình 7. Phân đoạn nội dung thành các "form"



Hình 8. Phân đoạn nội dung form thành các "chunk"

Bước 2: Trích xuất thực thể

Sau khi dữ liệu được phân đoạn thành các chunk, hệ thống tiến hành bước trích xuất tri thức để xây dựng đồ thị. Quy trình này được minh họa trong Hình 9.



Hình 9. Trích xuất thực thể

- Đối với dữ liệu lịch sử, sử dụng nội dung trong trường "content"; đối với dữ liệu du lịch, sử dụng trường "description".

- Từ mỗi đoạn văn bản, trích ra tối đa 1000 từ làm ngữ liệu đầu vào.
- Đưa ngữ liệu này vào prompt và gửi đến mô hình Gemini 2.0 Flash để thực hiện yêu cầu trích xuất thực thể: subject, relation, object bao gồm: subject, relation, object, đồng thời xác định thêm subject type và object type.
- Các thực thể sau khi trích xuất sẽ được lưu vào tệp định dạng JSON để phục vụ cho các bước xử lý tiếp theo.

Mẫu dữ liệu thực thể đã trích xuất:

```
// Dữ liệu du lịch
{"id": 1, "chunk_index": 0, "triples": [{"subject": "Văn Miếu",
↪  "subject_type": "Place", "relation": "được sáng lập bởi", "object":
↪  "Lý Thánh Tông", "object_type": "HistoricalFigure"}, {"subject":
↪  "Văn Miếu - Quốc Tử Giám", "subject_type": "Place", "relation": "nằm
↪  ở", "object": "Hà Nội", "object_type": "Location"}], "text": "Văn
↪  Miếu được xây dựng từ năm 1070 (tức năm Thần Vũ thứ hai đời Vua Lý
↪  Thánh Tông) để thờ ..."}

// Dữ liệu lịch sử
{"formId": "vol15_ch01", "chunk_index": 4, "triples": [{"subject": "Đại
↪  hội Đảng lần thứ VIII", "subject_type": "Event", "relation": "có
↪  liên quan đến", "object": "CNH, HĐH", "object_type": "Event"},
↪  {"subject": "Việt Nam", "subject_type": "Location", "relation": "gắn
↪  với thời kỳ lịch sử", "object": "giai đoạn 1996-2000",
↪  "object_type": "Time"}], "text": "xuất công nghiệp bình quân hằng
↪  năm 14-15%. - Tăng nhanh khả năng và tiềm lực tài chính của đất
↪  nước, lành mạnh hóa nền tài chính quốc gia:..."}
```

Bước 3: Thiết kế cấu trúc đồ thị

Hệ thống sử dụng sơ đồ lớp sau để định nghĩa tri thức:

Loại thực thể (Entity Type)	**Mô tả**
1. Địa lý – Thời gian	
Location	Địa danh, tỉnh thành, vùng miền
Time	Mốc thời gian, giai đoạn lịch sử, thế kỷ
Route	Tuyến du lịch, hành trình di chuyển
2. Con người và tổ chức	
HistoricalFigure	Nhân vật lịch sử có thật, có vai trò cụ thể
Person	Người hiện đại: du khách, hướng dẫn viên, người kể chuyện...
Ethnic, Religion	Các dân tộc, nhóm văn hóa, tôn giáo ở Việt Nam
Organization	Cơ quan, tổ chức liên quan đến lịch sử hoặc du lịch
3. Di sản – Văn hóa – Công trình	
Place	Di tích, công trình, địa điểm văn hóa – lịch sử
Festival	Lễ hội truyền thống gắn với yếu tố lịch sử hoặc văn hóa
Heritage / Artifact / HistoricalStructure	Di sản văn hóa, cổ vật, công trình lịch sử
Cuisine	Món ăn truyền thống, đặc sản vùng miền
4. Sự kiện – Tác phẩm	
Event	Các sự kiện lịch sử, sự kiện văn hóa – xã hội quan trọng
Dynasty	Triều đại, chế độ cai trị trong lịch sử Việt Nam
Work	Tác phẩm văn học, sách, tài liệu có giá trị lịch sử – du lịch

Bảng 4: Các loại thực thể trong đồ thị tri thức Lịch sử – Du lịch Việt Nam

Các loại quan hệ:

Loại quan hệ (Relation Type)	Ý nghĩa / Mô tả
1. Vị trí – Không gian	
`located_in`	Thực thể nằm trong một địa danh lớn hơn (tỉnh, vùng)
`is_located_in`	Tương tự `located_in`, thể hiện vị trí vật lý
`next_to`	Gần với địa danh khác
`suggests_route_to`	Gợi ý tuyến du lịch từ một địa điểm đến địa điểm khác
2. Thời gian – Giai đoạn	
`happened_on` / `occurred_on`	Sự kiện xảy ra vào thời điểm cụ thể
`belongs_to_period`	Gắn với thời kỳ lịch sử
`founded_on` / `established_in`	Thời điểm thành lập địa điểm / tổ chức
`ruled`	Triều đại / nhân vật cai trị một giai đoạn cụ thể
3. Sự kiện – Hoạt động	
`happened_in`	Sự kiện xảy ra ở một địa điểm
`takes_place_in`	Sự kiện, lễ hội diễn ra tại địa điểm
`participated_in`	Nhân vật / tổ chức tham gia vào sự kiện
`related_to_event`	Gợi ý sự kiện lịch sử liên quan tới một điểm đến
4. Nhân vật – Tổ chức – Công trình	
`founded`	Nhân vật sáng lập một địa danh, tổ chức, triều đại
`built_by`	Công trình, địa điểm được xây bởi ai
`under_feudal_rule_of`	Địa điểm thuộc sự cai trị của một triều đại
`ruled`	Triều đại / nhân vật cai trị thời kỳ / địa phương cụ thể
5. Văn hóa – Lễ hội – Tín ngưỡng	
`worships` / `dedicated_to`	Địa điểm thờ ai, dành để tưởng nhớ ai
`has_festival`	Gắn với lễ hội truyền thống
`features_cuisine`	Gắn với món ăn đặc sản vùng miền
`recognized_as`	Được công nhận là di tích / di sản / danh hiệu cụ thể
6. Các quan hệ khái quát khác	
`belongs_to`	Thuộc về địa phương, tổ chức, thời kỳ nào đó
`is_capital_of`	Là thủ đô của triều đại / chính quyền
`character_in` / `mentioned_in`	Nhân vật xuất hiện hoặc được đề cập trong sự kiện / phẩm
`related_to`	Có liên quan đến thực thể khác (địa danh, nhân vật, sự kiện

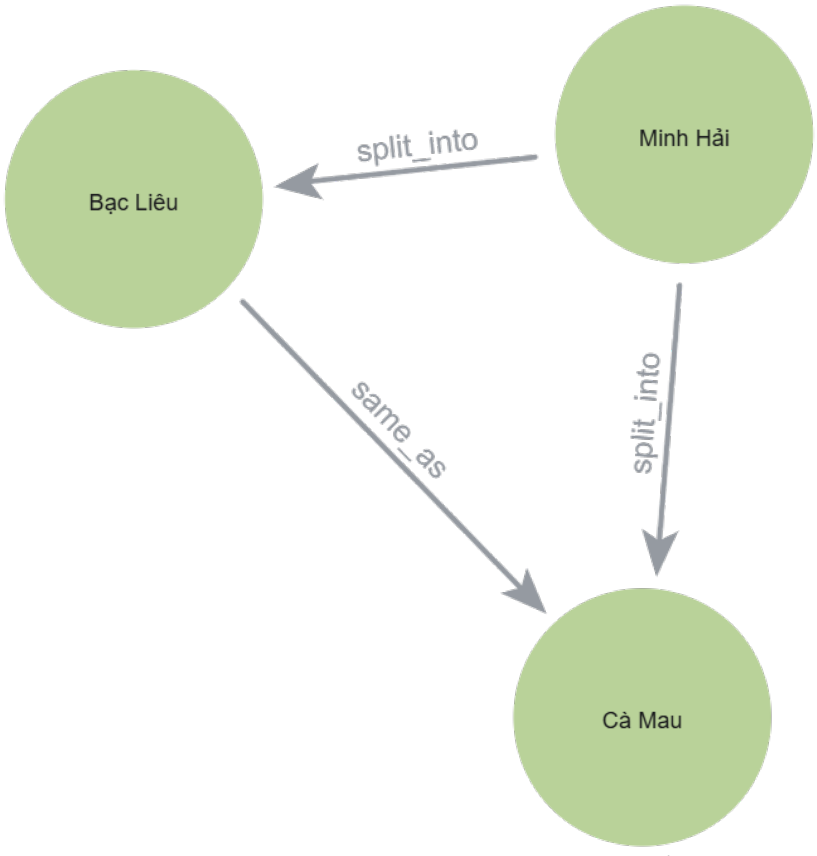
Bảng 5: Các loại quan hệ trong đồ thị tri thức Lịch sử – Du lịch Việt Nam

Bảng ánh xạ: Có những địa danh, tên tỉnh ở dữ liệu lịch sử không còn khớp với địa điểm hiện tại, do đó cần phải xây dựng bảng ánh xạ để kết nối các tỉnh.
Ví dụ cấu trúc bảng ánh xạ:

```
[
  {"old_name": "Xích Quỷ", "new_name": "Việt Nam", "period": "Cổ đại"},
  {"old_name": "Ái Châu", "new_name": "", "period": "Thời Bắc thuộc"},
  {"old_name": "Thăng Long", "new_name": "Hà Nội", "period":
  ↪ "1010-1831"},
  {"old_name": "Bình Trị Thiên", "new_name": "Quảng Bình + Quảng Trị +
  ↪ Thừa Thiên-Huế", "period": "1976-1992"}
]
```

Bảng ánh xạ khi áp dụng vào đồ thị tri thức, được biểu diễn bởi các mối quan hệ:

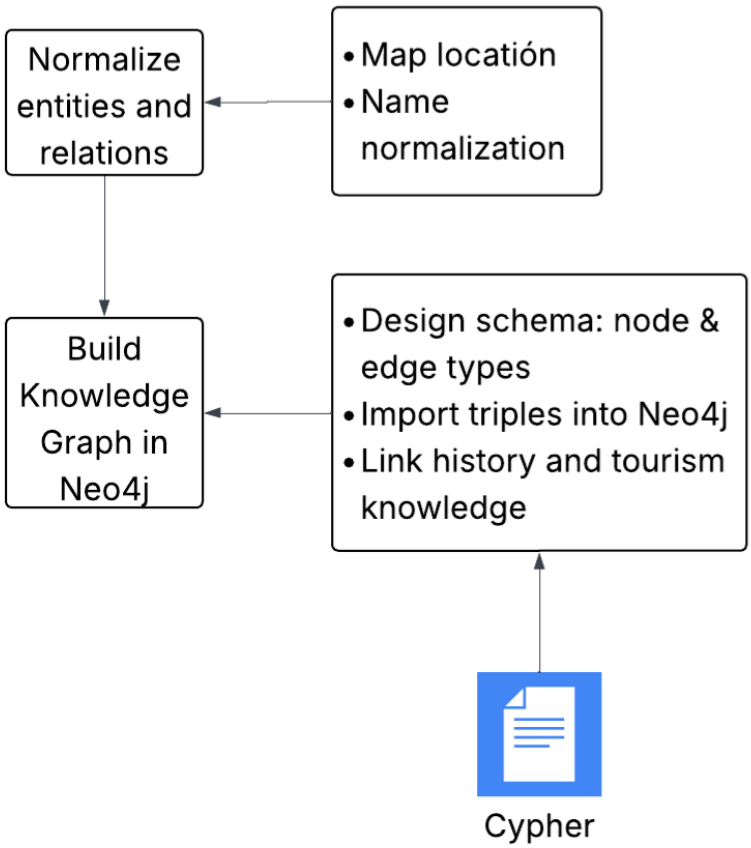
- same_as: chỉ những thực thể có tên gọi đã được thay đổi.
- split_into: chỉ những thực thể bị tách ra, ví dụ như Hình 10 "Minh Hải"tách thành "Bạc Liêu", "Cà Mau".



Hình 10. Ví dụ ánh xạ tên tỉnh

Bước 4: Xây dựng các nút và cạnh của đồ thị

Sau khi trích xuất và chuẩn hóa thực thể, hệ thống tiến hành xây dựng đồ thị tri thức trong Neo4j. Quy trình này được minh họa ở Hình 11.



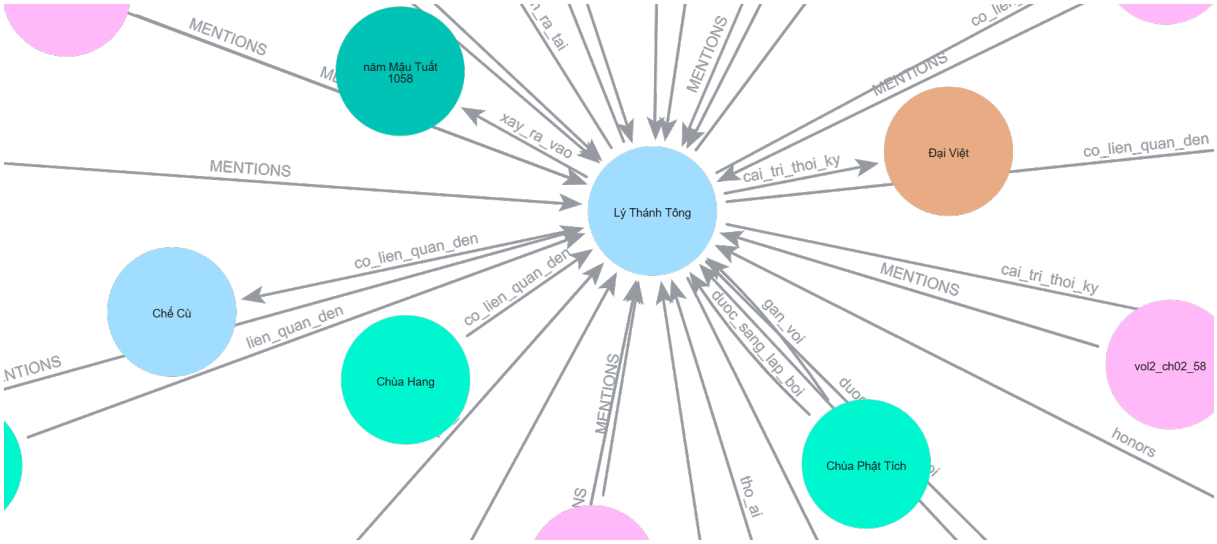
Hình 11. Các bước xây dựng đồ thị

Cụ thể:

- Thiết lập kết nối với Neo4j thông qua driver phù hợp (ví dụ: neo4j.GraphDatabase trong Python).
- Mỗi thực thể (subject, object) sẽ được biểu diễn dưới dạng một nút (node) trong đồ thị; quan hệ (relation) sẽ được biểu diễn dưới dạng cạnh có hướng (directed edge) nối giữa hai thực thể. Đặc biệt với cạnh "MENTIONS"thể hiện thực thể được nhắc đến trong nội dung thực thể khác (đoạn văn, chunk).
- Mỗi thực thể có thuộc tính type, dùng để phân loại node (ví dụ: Nhân vật, Sự kiện, Địa điểm, Thời gian...).
- Trước khi tạo node hoặc edge, hệ thống kiểm tra sự tồn tại để tránh trùng lặp. Nếu node đã tồn tại (cùng tên và cùng type), thì không tạo mới mà chỉ thêm quan hệ.

- Sau khi hoàn tất, toàn bộ đồ thị tri thức sẽ được lưu trữ trong Neo4j dưới dạng các cặp node–edge–node với thông tin đầy đủ về loại thực thể và quan hệ giữa chúng.
- Dữ liệu lịch sử và dữ liệu du lịch sẽ được liên kết qua thực thể (location, place).

Đồ thị tri thức sau khi xây dựng thành công:



Hình 12. Trực quan hóa đồ thị tri thức

Đánh giá đồ thị tri thức

- Số lượng triples: 40057
- Số lượng nút (nodes): 16042
- Số loại quan hệ: 95
- Số loại thực thể: 32

3.2.4 Thiết kế mô hình hỏi đáp

Trong dự án này, nhóm hướng tới việc thiết kế hệ thống QA sử dụng RAG kết hợp Knowledge Graph (KG) và so sánh hiệu quả giữa hai biến thể: Pretrained và Fine-tuned.

Kiến trúc hệ thống:

- LLM sử dụng Phi-3-mini-4k-instruct.
  - Kích thước: 3.8 tỷ tham số.
  - Giới hạn context: 4.096 token.
  - Kiểu: *Instruction-tuned*, tối ưu cho các tác vụ hỏi đáp và hội thoại.

  - Ưu điểm: nhẹ, dễ triển khai trên GPU 16GB (Kaggle T4), hỗ trợ tiếng Việt ở mức khá tốt, phù hợp cho cả inference (pretrained) và fine-tune LoRA.

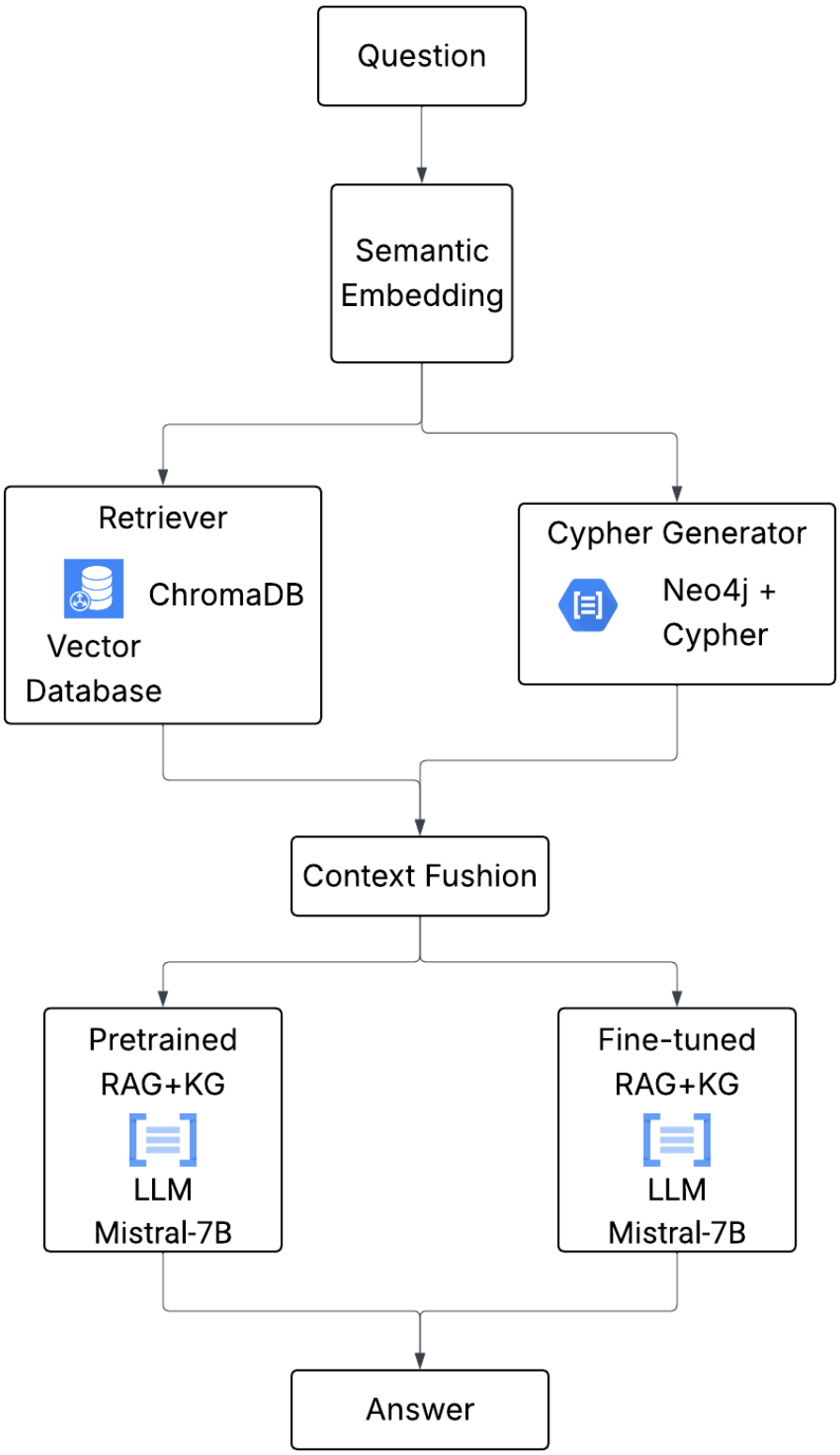
- RAG phần Retriever gồm:
  - Vector retriever từ ChromaDB (cho dữ liệu văn bản).
  - Graph retriever từ Neo4j (cho truy hồi tri thức từ KG).

- Tích hợp retriever + KG:
  - Sử dụng KG để mở rộng truy vấn, sinh chunk bổ sung từ các node có liên hệ (bằng related_to, happened_in,...).
  - Vector hóa các chunk liên quan từ KG, trộn vào truy hồi Top-K.

Quy trình xử lý RAG:
Quy trình vận hành của hệ thống được thiết kế như sau (tham khảo Hình 13):

1. **Nhận câu hỏi** từ người dùng.
2. **Truy xuất dữ liệu từ KG** bằng Cypher để lấy các triple liên quan trực tiếp đến câu hỏi.
3. **Chuẩn bị dữ liệu văn bản**:
   - Chia nhỏ (*chunking*) tài liệu lịch sử thành các đoạn 500–800 token.
   - Sinh vector embedding cho mỗi chunk bằng mô hình `dangvantuan/vietnamese-embedding`.
4. **Truy xuất theo ngữ nghĩa**:
   - So khớp câu hỏi với các vector embedding để lấy ra top-k đoạn liên quan nhất (**retriever**).
   - Xếp hạng lại kết quả bằng **cross-encoder re-ranking** để tăng độ chính xác.
5. **Kết hợp ngữ cảnh** từ KG và các đoạn văn bản đã được chọn.
6. **Sinh câu trả lời** bằng Phi-3-mini-4k-instruct, truyền toàn bộ ngữ cảnh vào prompt.

Sơ đồ hệ thống QA:



Hình 13. Hệ thống QA cho 2 trường hợp: Pretrained RAG+KG và Fine-tuned RAG+KG

3.2.5 Tối ưu và huấn luyện mô hình trả lời

Trường hợp 1: Pretrained RAG + KG

- Không fine-tune LLM, chỉ dùng RAG pipeline.
- Dữ liệu huấn luyện dưới dạng prompt-completion: prompt: phần chứa câu hỏi (hoặc ngữ cảnh) và completion: phần chứa câu trả lời mong muốn.
- Dữ liệu được nhúng bằng mô hình sentence-transformers (phobert) được lấy từ model vietnamese-embedding của "dangvantuan"rồi được lưu vào ChromaDB.
- Dữ liệu KG được truy hồi từ Neo4j, node liên quan được chunk sau đó embedding và nhập vào context.
- LLM tạo ra câu trả lời từ truy vấn và context.

Trường hợp 2: Fine-tuned RAG + KG:

- LLM (Mistral) được fine-tune trên tập dữ liệu lịch sử-du lịch theo định dạng prompt-completion.
- Dữ liệu vector embedding giữ nguyên như pipeline trên.
- Generator sau khi fine-tune sẽ có khả năng sinh văn bản sát ngữ cảnh lịch sử hơn.

Công nghệ sử dụng:

- Embedding: sentence-embedding (PhoBERT)
- Vector Store: ChromaDB
- KG: Neo4j
- LLM: Mistral-7B

3.2.6 Xây dựng giao diện người dùng

...

3.3 Minh hoạ

...

4 Thực nghiệm

4.1 Dữ liệu thực nghiệm

Dữ liệu thực nghiệm được xây dựng từ các nguồn tài liệu lịch sử và du lịch chính thống nhằm đảm bảo độ tin cậy và tính học thuật của hệ thống hỏi đáp. Quá trình chuẩn bị dữ liệu được chia thành hai phần chính: (1) dữ liệu tri thức nền cho truy hồi (retrieval) và (2) dữ liệu huấn luyện mô hình sinh ngôn ngữ (language generation).

4.1.1 Dữ liệu tri thức – Neo4j Knowledge Graph

Hệ thống sử dụng một đồ thị tri thức (KG) được xây dựng từ sách "Lịch sử Việt Nam" (15 tập) và các nguồn dữ liệu về di sản – địa điểm du lịch Việt Nam. Các thực thể và quan hệ được trích xuất thủ công và bán tự động, lưu trữ trong cơ sở dữ liệu đồ thị Neo4j. Tổng số triples: hơn 40,000.

4.1.2 Dữ liệu văn bản – Semantic Retrieval

Bên cạnh đồ thị tri thức, hệ thống còn sử dụng các đoạn văn bản (text chunk) từ cùng nguồn tài liệu để phục vụ cho module truy hồi ngữ nghĩa (semantic retriever).

- Tổng số đoạn văn bản: khoảng 30,000–35,000
- Mỗi đoạn có độ dài từ 300 đến 700 từ, được chia dựa trên ngữ nghĩa và dấu câu.
- Các đoạn này được vector hoá bằng mô hình intfloat/multilingual-e5-base và lưu trong ChromaDB để thực hiện tìm kiếm theo ngữ nghĩa (semantic search).

4.1.3 Dữ liệu huấn luyện mô hình sinh (Prompt–Completion)

Để huấn luyện mô hình ngôn ngữ lớn (LLM) cho khả năng sinh phản hồi tự nhiên, hệ thống sử dụng tập dữ liệu gồm hơn 30,000 cặp hỏi – đáp (prompt–completion).
Dữ liệu chia làm 3 phần:

- Train: 80%(khoảng 24,000 mẫu)
- Validation: 10% (khoảng 3,000 mẫu)
- Test: 10%(khoảng 3,000 mẫu)

4.2 Công cụ và tiêu chí đánh giá

...

4.3 Kết quả thực nghiệm

...

4.4 Phân tích kết quả

...

5 Ứng dụng minh họa

...

6 Kết luận và hướng phát triển

...

Tài liệu tham khảo

[1] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[2] D. Sanmartin, "Kg-rag: Bridging the gap between knowledge and creativity," 2024. arXiv preprint arXiv:2405.12035.

[3] X. Zhu, Y. Xie, Y. Liu, Y. Li, and W. Hu, "Knowledge graph-guided retrieval-augmented generation," 2025. arXiv preprint arXiv:2502.06864.

[4] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, and A. Zimmermann, "Knowledge graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, 2021.

[5] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[6] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, pp. 593–607, Springer, 2018.

[7] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc, "Foundations of modern query languages for graph databases," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–40, 2017.

[8] A. Hur, N. Janjua, and M. Ahmed, "A survey on state-of-the-art techniques for knowledge graphs construction and challenges ahead," *arXiv preprint arXiv:2110.08012*, 2021.

[9] M. Mukherjee, S. Kim, X. Chen, *et al.*, "From documents to dialogue: Building kg-rag enhanced ai assistants," 2025. arXiv preprint.

[10] W. Chen, T. Bai, J. Su, *et al.*, "Kg-retriever: Efficient knowledge indexing for retrieval-augmented large language models," 2024. arXiv preprint.

[11] Z. Xu, M. J. Cruz, *et al.*, "Retrieval-augmented generation with knowledge graphs for customer service question answering," 2024. arXiv preprint.

[12] M. Böckling, H. Paulheim, and A. Iana, "Walk&retrieve: Simple yet effective zero-shot retrieval-augmented generation via knowledge graph walks," 2025. arXiv preprint.

[13] C. Mavromatis and G. Karypis, "Gnn-rag: Graph neural retrieval for large language model reasoning," 2024. arXiv preprint.

[14] "Knowledge graph-guided retrieval augmented generation," 2025. NAACL.

[15] "Knowledge graph retrieval-augmented generation for llm-based recommendation," 2025. ACL long.

[16] J. Swacha and M. Gracel, "Retrieval-augmented generation (rag) chatbots for education: A survey of applications," 2025. Applied Sciences.

[17] "Systematic analysis of retrieval-augmented generation-based llms for medical chatbot applications," 2025. MDPI.

[18] "Document graphrag: Knowledge graph enhanced retrieval augmented generation for document question answering within the manufacturing domain," 2024. MDPI.

[19] Nhóm nghiên cứu Đại học Bách Khoa TP.HCM, "Hệ thống qa giáo dục kết hợp rag và knowledge graph," 2024. Truy cập nội bộ, không công bố chính thức.

[20] H. L. D, "Graph-rag: Qa hệ thống lịch sử việt nam dựa trên kg và rag." `https://github.com/Huy-L-D/graph-rag`, 2024. Dự án thực nghiệm trên GitHub.

[21] H. Guinness, "The best large language models (llms) in 2025," *Zapier Blog*, May 27 2025. Truy cập qua bản dịch: translate.goog/blog/best-llm/.

[22] Wikipedia contributors, "Knowledge graph — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Knowledge_graph`, 2025. Truy cập ngày 25/08/2025.

[23] S. Knollmeyer, O. Caymazer, and D. Grossmann, "Document graphrag: Knowledge graph enhanced retrieval-augmented generation for document question answering within the manufacturing domain," *Electronics*, vol. 14, no. 11, p. 2102, 2025. Open Access (CC BY).

[24] 0123, “ raggraphrag.” Blog trên 51CTO, 2025. Truy cập ngày 25/08/2025.

[25] A. Balaguer, V. Benara, R. L. de Freitas Cunha, R. de M. Estevão Filho, T. Hendry, D. Holstein, J. Marsman, N. Mecklenburg, S. Malvar, L. O. Nunes, R. Padilha, M. Sharp, B. Silva, S. Sharma, V. Aski, and R. Chandra, “Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture,” *arXiv preprint arXiv:2401.08406*, 2024. Accessed: 2025-08-25.