RESEARCH ARTICLE

VẬN DỤNG LLAMAINDEX VÀ GRAPHRAG CHO XÂY DỰNG VÀ TRUY VẤN ĐỒ THỊ TRI THỨC

Phạm Ngọc Bảo[1], Đinh Hùng[1], Nguyễn Quang Tấn[2], Nguyễn Hoàng Minh Nhật[2], Nguyễn Thị Thúy A[1]*

[1] Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ-Tin học TP.HCM
[2] Khoa Công nghệ thông tin, Trường Đại học Văn Hiến TPHCM
antt@huflit.edu.vn, baopn@huit.edu.vn, hungd@huflit.edu.vn, TanNQ@vhu.edu.vn, NhatNHM@vhu.edu.vn

TÓM TẮT— Bài báo này trình bày phương pháp kết hợp LLaMA Index và GraphRAG để xây dựng và truy vấn đồ thị tri thức từ dữ liệu văn bản thuần. Chúng tôi giới thiệu quy trình tự động hóa trích xuất thực thể, quan hệ và tạo bản tóm tắt cộng đồng, kết hợp mô hình ngôn ngữ lớn (LLM) với thuật toán phát hiện cộng đồng. Hệ thống cho phép xử lý truy vấn ngôn ngữ tự nhiên hiệu quả. Thực nghiệm trên tập dữ liệu du lịch tại TP. Hồ Chí Minh minh họa khả năng trả lời chính xác các truy vấn phức tạp. Qua kết quả thực nghiệm cho thấy các chỉ số về khả năng đáp ứng yêu cầu truy vấn của phương pháp chúng tôi đề xuất tăng khoản 4% so với chỉ sử dụng mỗi LLM để truy vấn thông tin. Các thách thức về chi phí tính toán và yêu cầu dữ liệu chất lượng cao cũng được thảo luận, cùng với tiềm năng ứng dụng trong quản lý tri thức và tìm kiếm ngữ nghĩa.

Từ khóa— Mô hình ngôn ngữ lớn, Đồ thị tri thức, GraphRAG, LlamaIndex

GIỚI THIỆU

Trong bối cảnh bùng nổ thông tin hiện nay, việc tổ chức và khai thác hiệu quả khối lượng dữ liệu phức tạp là một thách thức lớn đối với các hệ thống trí tuệ nhân tạo và tìm kiếm thông tin. Đồ thị tri thức (Knowledge Graph - KG), với khả năng biểu diễn tri thức dưới dạng các thực thể và mối quan hệ ngữ nghĩa, đã nổi lên như một công cụ mạnh mẽ để quản lý và truy xuất dữ liệu có cấu trúc [1]. Các ứng dụng thực tiễn của đồ thị tri thức tiêu biểu như Google Knowledge Graph – đã chứng minh tiềm năng của KG trong việc cung cấp câu trả lời trực tiếp và liên kết thông tin theo ngữ cảnh cho các truy vấn người dùng [2]. Điều này cho thấy cả ý nghĩa thực tiễn lẫn tầm quan trọng khoa học của việc nghiên cứu và phát triển đồ thị tri thức trong kỷ nguyên dữ liệu lớn.

Sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn (Large Language Models – LLMs) trong những năm gần đây, chẳng hạn như GPT-4 và LLaMA, cùng với sự ra đời của các công cụ tiên tiến như LlamaIndex và GraphRAG, đã mở ra những cơ hội mới để tự động hóa quá trình xây dựng và truy vấn đồ thị tri thức [3] [4]. Các nghiên cứu cho thấy LLMs có khả năng hiểu và sinh ngôn ngữ tự nhiên ở mức độ cao [3], cho phép chúng tham gia vào việc phân tích và tổng hợp thông tin phức tạp. Các công cụ như LlamaIndex giúp kết nối LLM với các nguồn dữ liệu bên ngoài (bao gồm cả KG), trong khi GraphRAG (Graphs + Retrieval-Augmented Generation) kết hợp quá trình truy xuất thông tin KG nhằm tăng cường thông tin cho LLM để nâng cao độ chính xác cho câu trả lời khi truy vấn [4]. Sự hội tụ của công nghệ LLM và KG hứa hẹn nâng cao hiệu quả cũng như tính linh hoạt của các hệ thống quản lý tri thức, cho phép chúng xử lý tốt hơn các truy vấn đa dạng từ người dùng.

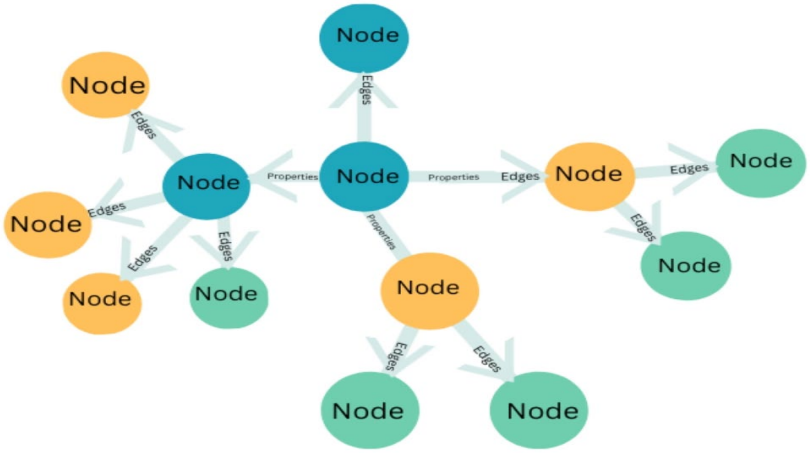
Xây dựng một đồ thị tri thức thường đòi hỏi trích xuất thực thể và mối quan hệ từ nhiều nguồn dữ liệu khác nhau (bao gồm cả văn bản phi cấu trúc), thông qua các kỹ thuật xử lý ngôn ngữ tự nhiên như nhận dạng thực thể (Named Entity Recognition - NER) và trích xuất quan hệ [5]. Tuy nhiên, quá trình này gặp nhiều thách thức khi dữ liệu đầu vào có thể không đồng nhất và chứa đựng ngữ cảnh phức tạp. Ở khía cạnh truy vấn, các phương pháp truyền thống dựa trên ngôn ngữ truy vấn có cấu trúc như SPARQL [6] (cho đồ thị RDF) hoặc Cypher [7] (cho đồ thị thuộc tính) tỏ ra hiệu quả với câu hỏi đơn giản, nhưng lại hạn chế khi đối mặt với những truy vấn phức tạp hoặc được diễn đạt bằng ngôn ngữ tự nhiên. Điều này tạo ra khoảng trống giữa nhu cầu truy vấn linh hoạt của người dùng và khả năng đáp ứng của các hệ thống KG hiện tại. Gần đây, LlamaIndex và GraphRAG đã tạo ra bước đột phá khi kết hợp khả năng hiểu ngôn ngữ tự nhiên mạnh mẽ của LLM với tính chặt chẽ của cấu trúc đồ thị. Cách tiếp cận này cho phép hệ thống truy xuất thông tin một cách linh hoạt, chính xác hơn và sinh câu trả lời có ngữ cảnh phù hợp với ý định người dùng . Sự kết hợp mới mẻ này không chỉ cải thiện hiệu suất truy vấn mà còn mở ra hướng đi mới trong việc xây dựng đồ thị tri thức tự động từ khối lượng dữ liệu khổng lồ một cách hiệu quả và ít tốn công sức thủ công.

Trong bài báo này, chúng tôi giới thiệu về GraphRAG, LlamaIndex cũng như là cách dùng chúng để tạo ra và truy vấn đồ thị tri thức. Các đóng góp chính của bài báo bao gồm: *(i) Cung cấp một cái nhìn tổng quan cũng như kiến trúc của kỹ thuật GraphRAG với LlamaIndex; (ii) giới thiệu về phương pháp xây dựng KG và tạo bản tóm tắt cộng đồng tự động bằng LLM; (iii) kết hợp LlamaIndex và GraphRAG đề xuất pipe line ba bước để truy vấn thông tin theo thứ tự truy xuất bản tóm tắt cộng đồng, tạo câu trả lời trung gian và tổng hợp câu trả lời.*

Phần còn lại của bài báo được trình bày như sau: Mục II giới thiệu tổng quan về cơ sở lý thuyết và các công nghệ liên quan. Ở mục III, trình bày về phương pháp đề xuất dùng GraphRAG với LlamaIndex. Mục IV giới thiệu các kết quả thực nghiệm trong việc xây dựng và truy vấn đồ thị tri thức, tạo nên hệ thống hỏi đáp mô phỏng. Cuối cùng là phần kết luận ở mục V.

I. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ LIÊN QUAN

A. ĐỒ THỊ TRI THỨC



Hình 1. Đồ thị tri thức gồm các nút (node) liên kết bởi các cạnh (edge)

Đồ thị tri thức (Knowledge Graph - KG) là một mô hình biểu diễn dữ liệu theo dạng đồ thị, trong đó các nút đại diện cho các thực thể (entities) và các cạnh biểu thị mối quan hệ (relationships) giữa chúng. Với cấu trúc ngữ nghĩa, đồ thị tri thức cho phép tổ chức thông tin phức tạp và hỗ trợ các ứng dụng như tìm kiếm ngữ nghĩa, trả lời câu hỏi, và phân tích dữ liệu [1]. Một ví dụ điển hình là Google Knowledge Graph, ra mắt năm 2012, đã cải thiện khả năng truy xuất thông tin liên quan và cung cấp câu trả lời trực tiếp cho các truy vấn người dùng [2]. Các nghiên cứu gần đây đã ứng dụng các kỹ thuật học sâu để tăng cường khả năng biểu diễn và suy luận trong đồ thị tri thức, đặc biệt trong việc xử lý dữ liệu không cấu trúc [8].

B. XÂY DỰNG VÀ TRUY VẤN ĐỒ THỊ TRI THỨC

Quá trình xây dựng đồ thị tri thức bao gồm trích xuất thực thể (Named Entity Recognition - NER), phân tích cú pháp, và trích xuất quan hệ (Relation Extraction) từ các nguồn dữ liệu có cấu trúc, bán cấu trúc, hoặc không cấu trúc [5]. Các phương pháp truyền thống thường dựa vào các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) thủ công, đòi hỏi nhiều công sức và khó mở rộng cho dữ liệu lớn. Sự ra đời của các mô hình ngôn ngữ lớn (Large Language Models - LLMs) như GPT4, LLaMA đã thay đổi cục diện, cung cấp khả năng tự động trích xuất và biểu diễn thông tin với độ chính xác cao [3]. LLaMA, với hiệu suất vượt trội trong xử lý văn bản, hỗ trợ xây dựng các mối quan hệ ngữ nghĩa phức tạp, từ đó tối ưu hóa việc tạo đồ thị tri thức từ dữ liệu khối lượng lớn.

Về truy vấn, các phương pháp truyền thống thường sử dụng ngôn ngữ truy vấn như SPARQL cho các đồ thị RDF [6]. Tuy nhiên, SPARQL yêu cầu người dùng có kiến thức chuyên môn và gặp khó khăn trong việc xử lý các truy vấn ngôn ngữ tự nhiên hoặc phức tạp. Để khắc phục, các công nghệ mới như LLaMA Index và GraphRAG đã được phát triển, kết hợp sức mạnh của LLMs với cấu trúc đồ thị để cải thiện khả năng truy vấn [4].

C. GRAPHRAG

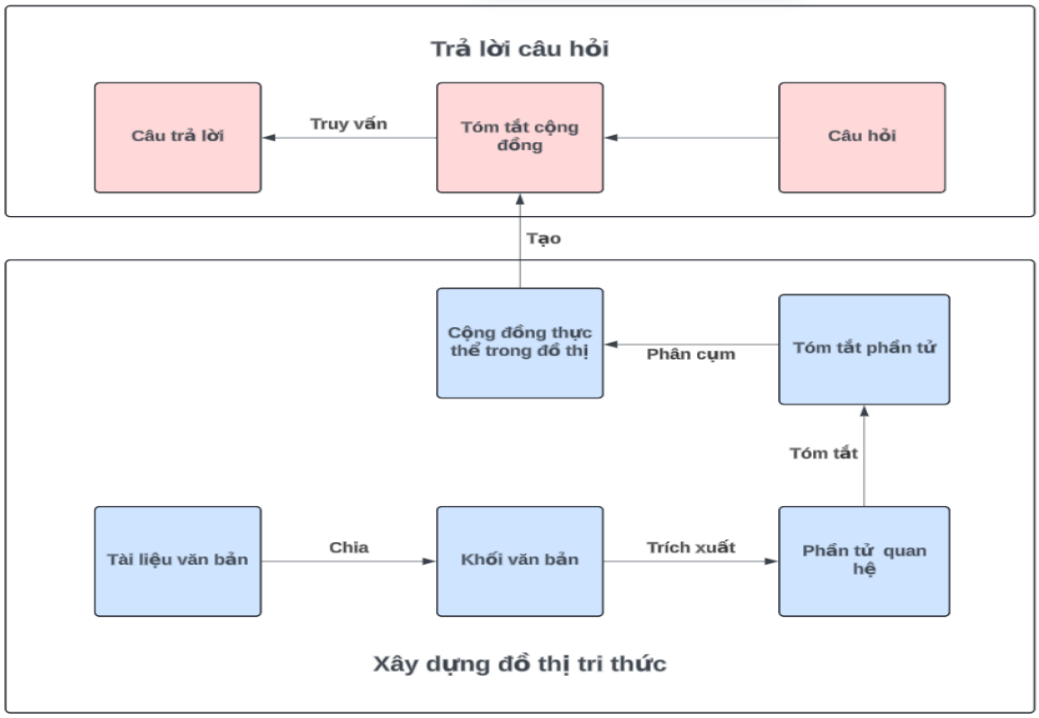
GraphRAG (Retrieval-Augmented Generation) là một phương pháp tiên tiến kết hợp truy xuất thông tin (retrieval) và tạo sinh (generation) để trả lời các truy vấn phức tạp trên đồ thị tri thức. Không giống các phương pháp truy xuất thông tin truyền thống, GraphRAG sử dụng LLMs để phân tích cấu trúc đồ thị và tạo câu trả lời dựa trên cả thông tin trong đồ thị và dữ liệu bổ sung từ các nguồn bên ngoài [4]. Điều này giúp GraphRAG vượt trội trong việc xử lý các truy vấn yêu cầu suy luận ngữ nghĩa sâu, ví dụ như các câu hỏi liên quan đến mối quan hệ đa tầng giữa các thực thể [4]. Sự kết hợp giữa LLaMA Index và GraphRAG tạo ra một hệ thống mạnh mẽ, cho phép xây dựng và truy vấn đồ thị tri thức một cách hiệu quả và linh hoạt.

D. LLAMA INDEX

LLaMA Index là một khung (framework) làm việc cho phép lập chỉ mục và truy vấn dữ liệu dựa trên các mô hình ngôn ngữ lớn. Công cụ này tổ chức dữ liệu thành các chỉ mục (indices) ngữ nghĩa, giúp truy xuất thông tin nhanh chóng và tạo câu trả lời dựa trên ngữ cảnh từ các nguồn dữ liệu phân tán [4]. LLaMA Index đặc biệt hiệu quả trong việc xử lý dữ liệu không cấu trúc, hỗ trợ xây dựng KG bằng cách trích xuất thực thể và quan hệ một cách tự động. Ngoài ra, nó cung cấp giao diện linh hoạt cho các ứng dụng truy vấn, từ tìm kiếm đơn giản đến trả lời câu hỏi phức tạp [9].

II. PHƯƠNG PHÁP ĐỀ XUẤT

Để cải thiện hiệu năng của các hệ thống truy vấn thông tin chúng tôi đề xuất phương pháp kết hợp LLaMA Index và GraphRAG để xây dựng và truy vấn KG từ dữ liệu văn bản không cấu trúc. Quy trình bao gồm hai giai đoạn chính: (1) xây dựng đồ thị tri thức bằng cách trích xuất tự động thực thể và quan hệ, tổ chức thành cấu trúc đồ thị, và tạo bản tóm tắt cộng đồng; (2) truy vấn KG bằng cách sử dụng các bản tóm tắt cộng đồng để trả lời các câu hỏi ngôn ngữ tự nhiên. Phương pháp này tận dụng LLM và thuật toán phát hiện cộng đồng để nâng cao hiệu quả truy xuất và suy luận ngữ nghĩa so với các phương pháp truy vấn truyền thống [4] [9], chi tiết như hình 2.



Hình 2. Quy trình xây dựng và truy vấn KG sử dụng LLaMA Index và GraphRAG.

A. XÂY DỰNG ĐỒ THỊ TRI THỨC

Quá trình xây dựng đồ thị tri thức bao gồm các bước sau:

1. PHÂN ĐOẠN VĂN BẢN

Một bước tiền xử lý quan trọng trong xây dựng đồ thị tri thức, nhằm chia nhỏ tài liệu đầu vào thành các đơn vị ngữ nghĩa độc lập, tạo điều kiện thuận lợi cho việc trích xuất thực thể và quan hệ. Phương pháp đề xuất trong nghiên cứu này sử dụng một chiến lược phân đoạn linh hoạt, kết hợp các ranh giới tự nhiên của văn bản (như dấu xuống dòng hoặc dấu câu) với các ràng buộc tối ưu hóa độ dài đoạn, nhằm đảm bảo các đoạn vừa đủ ngắn để xử lý hiệu quả, vừa giữ được tính liên kết ngữ nghĩa.

Quá trình phân đoạn được thực hiện qua hai giai đoạn chính: (1) tách văn bản dựa trên các ranh giới ngữ nghĩa tự nhiên, chẳng hạn như các đoạn văn hoặc câu có ý nghĩa hoàn chỉnh; (2) tinh chỉnh độ dài đoạn để đáp ứng các ngưỡng tối thiểu và tối đa.

Phương pháp này đảm bảo các đoạn văn bản không chỉ phù hợp cho các thuật toán xử lý ngôn ngữ tự nhiên (NLP) mà còn hỗ trợ tối ưu hóa việc trích xuất thực thể và quan hệ trong các bước tiếp theo. So với các phương pháp phân đoạn đơn giản như chia theo số từ cố định, chiến lược này vượt trội nhờ khả năng duy trì ngữ cảnh và giảm thiểu mất mát thông tin, đặc biệt khi xử lý dữ liệu không cấu trúc phức tạp. Tuy nhiên, việc xác định các ngưỡng độ dài tối thiểu và độ dài tối đa đòi hỏi cân nhắc cẩn thận để tránh phân đoạn không đồng đều, đặc biệt với các văn bản có cấu trúc đa dạng.

2. TRÍCH XUẤT THỰC THỂ VÀ QUAN HỆ

Một bước cốt lõi trong quá trình xây dựng đồ thị tri thức, nhằm nhận diện các thành phần ngữ nghĩa quan trọng và các mối liên kết giữa chúng từ dữ liệu văn bản không cấu trúc. Phương pháp đề xuất sử dụng *GraphRAGExtractor*, một thành phần tùy chỉnh được tích hợp trong khung làm việc LLaMA Index, để tự động trích xuất các bộ ba tri thức (subject-relation-object), hình thành nền tảng cho cấu trúc đồ thị tri thức. Quá trình này

tận dụng sức mạnh của LLM để phân tích văn bản, đảm bảo khả năng nhận diện chính xác các thực thể và quan hệ trong bối cảnh ngữ nghĩa phức tạp [4].

Quy trình trích xuất được thực hiện qua các bước chính sau:

Phân tích văn bản bằng LLM: Một mô hình ngôn ngữ lớn, chẳng hạn như GPT-4o-mini, được sử dụng để xử lý từng đoạn văn bản đã phân đoạn. LLM nhận diện các thực thể (entities) – những đối tượng có ý nghĩa cụ thể như địa danh, tổ chức, hoặc sự kiện – cùng với loại thực thể (entity type) và mô tả chi tiết (entity description). Quá trình này dựa trên khả năng hiểu ngữ cảnh của LLM, cho phép phân biệt các thực thể cụ thể với các khái niệm chung chung, từ đó nâng cao chất lượng trích xuất.

Định dạng và lưu trữ dữ liệu: Các thực thể được biểu diễn dưới dạng *EntityNode*, trong khi các quan hệ giữa chúng được lưu dưới dạng *Relation*. Mỗi thực thể và quan hệ được gắn với các thuộc tính bổ sung (metadata), bao gồm mô tả chi tiết, được lưu trữ dưới các khóa như *KG_NODES_KEY* và *KG_RELATIONS_KEY*. Cấu trúc này không chỉ đảm bảo tính tổ chức của dữ liệu mà còn hỗ trợ việc truy xuất và suy luận trong các bước sau của quy trình xây dựng đồ thị tri thức.

Phân tích cú pháp phản hồi LLM: Để chuyển đổi đầu ra của LLM thành dữ liệu có cấu trúc, một hàm phân tích cú pháp (**parse_fn**) được áp dụng. Hàm này sử dụng các kỹ thuật như biểu thức chính quy để tách biệt các thực thể và quan hệ từ phản hồi văn bản của LLM, đảm bảo các bộ ba tri thức được định dạng chính xác và nhất quán. Quá trình này giảm thiểu lỗi trích xuất và tăng cường độ tin cậy của dữ liệu đầu ra.

Hướng dẫn trích xuất bằng Prompt: Một mẫu prompt (mô tả ở phần PHỤ LỤC A.1) được thiết kế cẩn thận được sử dụng để chỉ đạo LLM trong việc trích xuất các bộ ba tri thức. Prompt này xác định rõ ràng các tiêu chí trích xuất, chẳng hạn như giới hạn số lượng bộ ba tối đa cho mỗi đoạn văn bản và yêu cầu tập trung vào các quan hệ có ý nghĩa ngữ nghĩa, như "nằm tại", "gần", hoặc "liên quan đến". Mẫu prompt cũng yêu cầu mô tả chi tiết cho mỗi thực thể và quan hệ, nhằm cung cấp thông tin bổ sung có giá trị cho việc xây dựng đồ thị tri thức. Cấu trúc của prompt được tối ưu hóa để đảm bảo LLM tạo ra kết quả nhất quán và phù hợp với mục tiêu của hệ thống [9].

Phương pháp trích xuất này mang lại nhiều ưu điểm so với các kỹ thuật truyền thống, chẳng hạn như trích xuất thực thể dựa trên quy tắc hoặc các mô hình học máy đơn giản. Bằng cách tận dụng khả năng hiểu ngữ cảnh sâu sắc của LLM, GraphRAGExtractor có thể xử lý các văn bản phức tạp và đa dạng, đồng thời giảm thiểu sự phụ thuộc vào các bộ dữ liệu huấn luyện lớn. Hơn nữa, việc tích hợp các mô tả chi tiết vào metadata cho phép đồ thị tri thức lưu giữ thông tin ngữ cảnh phong phú, hỗ trợ tốt hơn cho các ứng dụng truy vấn ngôn ngữ tự nhiên và suy luận ngữ nghĩa.

Tuy nhiên, quá trình trích xuất cũng đặt ra một số thách thức. Sự phụ thuộc vào LLM có thể dẫn đến chi phí tính toán cao, đặc biệt khi xử lý khối lượng văn bản lớn. Ngoài ra, chất lượng trích xuất phụ thuộc vào thiết kế của prompt và khả năng phân tích của mô hình, đòi hỏi sự tinh chỉnh liên tục để đạt hiệu quả tối ưu. Các nghiên cứu tương lai có thể tập trung vào việc cải thiện hiệu suất của GraphRAGExtractor, chẳng hạn bằng cách tích hợp các kỹ thuật nén ngữ nghĩa hoặc giảm thiểu sai số trong phân tích cú pháp, nhằm nâng cao độ chính xác và khả năng mở rộng của phương pháp [10].

Lưu trữ đồ thị: Việc lưu trữ các thực thể và quan hệ là một bước quan trọng để tổ chức dữ liệu tri thức dưới dạng cấu trúc đồ thị, tạo điều kiện cho việc truy xuất và suy luận hiệu quả. Phương pháp đề xuất sử dụng *GraphRAGStore*, một lớp mở rộng của *SimplePropertyGraphStore*, được thiết kế để quản lý các thành phần của đồ thị tri thức một cách có hệ thống [4]. GraphRAGStore tận dụng cơ sở dữ liệu đồ thị, chẳng hạn như Neo4j, để biểu diễn các thực thể dưới dạng nút (nodes) và các quan hệ dưới dạng cạnh (edges), hình thành một cấu trúc liên kết ngữ nghĩa mạnh mẽ.

Quá trình lưu trữ bao gồm việc ánh xạ các thực thể và quan hệ đã trích xuất vào một mô hình đồ thị có hướng, trong đó mỗi nút đại diện cho một thực thể với các thuộc tính như tên, loại thực thể, và mô tả chi tiết, còn mỗi cạnh biểu thị một quan hệ với các thuộc tính bổ sung như loại quan hệ và ngữ cảnh liên kết. GraphRAGStore hỗ trợ lưu trữ metadata phong phú, cho phép gắn các thông tin ngữ cảnh vào từng nút và cạnh, từ đó nâng cao khả năng truy vấn và phân tích đồ thị.

Để đảm bảo tính toàn vẹn và hiệu quả, GraphRAGStore sử dụng các kỹ thuật tối ưu hóa như lập chỉ mục nút và cạnh, giúp tăng tốc độ truy xuất dữ liệu, đặc biệt với các đồ thị tri thức quy mô lớn. So với các phương pháp lưu trữ truyền thống như cơ sở dữ liệu quan hệ, cách tiếp cận này vượt trội nhờ khả năng biểu diễn các mối quan hệ phức tạp và hỗ trợ các thuật toán phân tích đồ thị tiên tiến. Tuy nhiên, thách thức nằm ở việc quản lý tài nguyên tính toán khi đồ thị mở rộng, đòi hỏi các chiến lược nén dữ liệu hoặc phân vùng đồ thị để duy trì hiệu suất.

Để tăng cường độ tin cậy của quá trình trích xuất thực thể và quan hệ, chúng tôi đề xuất tích hợp các cơ chế kiểm soát lỗi và đánh giá độ tin cậy vào quy trình trích xuất dựa trên LLM. Các cơ chế này bao gồm:

Kiểm soát lỗi trong phân tích cú pháp: Để xử lý các đầu ra không hợp lệ từ LLM, một hàm phân tích cú pháp nâng cao (parse_fn) được sử dụng, kết hợp các biểu thức chính quy và kiểm tra tính hợp lệ của dữ liệu. Các thực thể hoặc quan hệ không đáp ứng các tiêu chí định dạng (ví dụ: tên thực thể rỗng hoặc quan hệ không có nguồn/đích) sẽ bị loại bỏ, đồng thời được ghi lại vào một tệp nhật ký (error_log.txt) để theo dõi và phân tích lỗi. Điều này đảm bảo rằng chỉ các bộ ba tri thức hợp lệ được đưa vào đồ thị tri thức, giảm thiểu nguy cơ sai lệch dữ liệu.

Đánh giá độ tin cậy của triplet: Mỗi thực thể và quan hệ trích xuất được gán một điểm độ tin cậy dựa trên các tiêu chí sau: (1) sự xuất hiện của thực thể trong văn bản gốc, (2) độ dài và chi tiết của mô tả, và (3) tính phù hợp với ngữ cảnh sinh học (trong trường hợp này là về cá). Một hàm đánh giá (score_triplet_reliability) tính toán điểm độ tin cậy, trong đó các thực thể và quan hệ có điểm cao hơn được ưu tiên sử dụng trong đồ thị. Ngoài ra, các thực thể được so sánh với một danh sách các thực thể đã xác minh (ví dụ: “saumon”, “anguille européenne”) để đảm bảo tính chính xác. Các triplet có độ tin cậy thấp được đánh dấu trong metadata để hỗ trợ việc tinh chỉnh sau này.

Xử lý lỗi khi gọi LLM: Để xử lý các lỗi trong quá trình gọi API LLM (ví dụ: lỗi kết nối hoặc quá tải), chúng tôi áp dụng cơ chế thử lại (retry) với số lần thử tối đa là 3 và ghi log lỗi chi tiết. Các node không trích xuất được triplet sẽ được gắn thẻ lỗi trong metadata (error: "Không tìm thấy triplet tri thức") để đảm bảo quy trình tiếp tục mà không bị gián đoạn.

Phát hiện cộng đồng: Phát hiện cộng đồng là một bước quan trọng để tổ chức các thực thể trong đồ thị tri thức thành các nhóm có ý nghĩa ngữ nghĩa, từ đó tăng cường khả năng truy xuất và suy luận. Phương pháp đề xuất sử dụng thuật toán **Hierarchical Leiden**, một kỹ thuật phân cụm dựa trên đồ thị, để phân chia các nút liên quan thành các cộng đồng [11]. Thuật toán này, được triển khai thông qua thư viện **graspologic**, cải tiến từ thuật toán Leiden bằng cách áp dụng phân cụm phân cấp, cho phép phát hiện các cấu trúc cộng đồng ở nhiều cấp độ chi tiết.

So với các phương pháp phân cụm truyền thống như K-means hoặc DBSCAN, Hierarchical Leiden vượt trội nhờ khả năng xử lý đồ thị không đồng nhất mà không yêu cầu xác định trước số lượng cụm. Việc áp dụng cấu trúc phân cấp cho phép khám phá các cộng đồng ở các cấp độ khác nhau, từ các nhóm nhỏ chặt chẽ đến các nhóm lớn hơn, bao quát hơn. Tuy nhiên, thuật toán này có thể gặp thách thức về chi phí tính toán khi xử lý các đồ thị lớn, đòi hỏi các kỹ thuật tối ưu hóa như phân vùng đồ thị hoặc giảm độ phức tạp thuật toán trong các ứng dụng thực tiễn [12].

Tạo bản tóm tắt cộng đồng: Tạo bản tóm tắt cộng đồng là một bước quan trọng để tổng hợp thông tin từ các nhóm thực thể trong đồ thị tri thức, cung cấp cái nhìn tổng quan về cấu trúc ngữ nghĩa và hỗ trợ truy vấn hiệu quả. Phương pháp này sử dụng mô hình ngôn ngữ lớn (LLM) để tạo ra các bản tóm tắt ngắn gọn nhưng toàn diện cho mỗi cộng đồng, dựa trên các mô tả quan hệ giữa các thực thể trong cộng đồng đó. Các bản tóm tắt được lưu trữ trong **community_summary** của **GraphRAGStore**, đóng vai trò như một lớp tri thức cô đọng để tăng tốc độ xử lý truy vấn [4].

Quá trình tạo bản tóm tắt bắt đầu bằng việc thu thập thông tin từ các quan hệ trong mỗi cộng đồng, bao gồm các cặp thực thể, loại quan hệ, và mô tả chi tiết của chúng. LLM, chẳng hạn như GPT-4o-mini, được sử dụng để phân tích và tổng hợp các thông tin này, tạo ra một đoạn văn xuôi mạch lạc, tránh lặp lại và ưu tiên các khía cạnh ngữ nghĩa quan trọng. Để đảm bảo chất lượng, một mẫu prompt (mô tả ở phần PHỤ LỤC A.2) được thiết kế để hướng dẫn LLM tập trung vào việc gộp các quan hệ tương tự và diễn đạt theo cách tự nhiên, phù hợp với các ứng dụng hỏi đáp [9].

Để đảm bảo độ tin cậy của các bản tóm tắt cộng đồng, chúng tôi bổ sung một bước đánh giá chất lượng đầu ra của LLM trong quá trình tổng hợp. Cụ thể:

Kiểm tra tính nhất quán: Các bản tóm tắt được kiểm tra để đảm bảo rằng chúng không chứa thông tin mâu thuẫn với các quan hệ trong cộng đồng. Một hàm kiểm tra tính nhất quán (validate_community_summary) so sánh bản tóm tắt với các triplet gốc, loại bỏ các thông tin không phù hợp hoặc không có trong dữ liệu nguồn.

Đánh giá độ tin cậy của bản tóm tắt: Mỗi bản tóm tắt được gán một điểm độ tin cậy dựa trên mức độ bao quát của thông tin và sự phù hợp với ngữ cảnh của cộng đồng. Điểm này được tính dựa trên tỷ lệ các quan hệ được sử dụng trong bản tóm tắt và độ dài của văn bản tóm tắt. Các bản tóm tắt có điểm thấp hơn ngưỡng (ví dụ: 0.5) sẽ được tạo lại bằng cách điều chỉnh prompt hoặc sử dụng một mẫu LLM khác.

Các cải tiến này đảm bảo rằng bản tóm tắt cộng đồng không chỉ ngắn gọn và mạch lạc mà còn đáng tin cậy, giảm thiểu nguy cơ truyền bá thông tin sai lệch trong quá trình truy vấn.

3. TRUY VẤN ĐỒ THỊ TRI THỨC

Quá trình truy vấn được thực hiện bởi **GraphRAGQueryEngine**, một query engine tùy chỉnh, với các bước sau:

a) Truy xuất bản tóm tắt cộng đồng

Truy xuất bản tóm tắt cộng đồng là bước khởi đầu trong quy trình truy vấn đồ thị tri thức, nhằm thu thập các thông tin cô đọng đại diện cho các nhóm thực thể có liên kết ngữ nghĩa. Phương pháp đề xuất sử dụng một cơ chế truy xuất tích hợp trong **GraphRAGQueryEngine**, tận dụng **GraphRAGStore** để lấy các bản tóm tắt cộng đồng đã được xây dựng trước đó [4]. Các bản tóm tắt này, được lưu trữ dưới dạng các đoạn văn xuôi ngữ nghĩa, cung cấp một cái nhìn tổng quan về cấu trúc và mối quan hệ giữa các thực thể trong từng cộng đồng, từ đó hỗ trợ việc xử lý các truy vấn phức tạp.

Quá trình truy xuất được thiết kế để đảm bảo tính hiệu quả và toàn vẹn thông tin. Mỗi bản tóm tắt cộng đồng được tổ chức như một đơn vị tri thức độc lập, chứa các mô tả tổng hợp về các thực thể và quan hệ trong cộng đồng đó. Để tối ưu hóa hiệu suất, hệ thống sử dụng các kỹ thuật lập chỉ mục ngữ nghĩa, cho phép truy xuất nhanh các bản tóm tắt phù hợp với ngữ cảnh truy vấn.

So với các phương pháp truy xuất truyền thống, cách tiếp cận này vượt trội nhờ khả năng xử lý các truy vấn ngôn ngữ tự nhiên và khai thác thông tin ngữ nghĩa từ các bản tóm tắt đã được cô đọng. Tuy nhiên, thách thức nằm ở việc đảm bảo rằng các bản tóm tắt được truy xuất bao quát toàn bộ thông tin liên quan, đặc biệt khi đồ thị tri thức chứa nhiều cộng đồng với các mức độ chi tiết khác nhau. Các cải tiến tương lai có thể tập trung vào việc tích hợp các kỹ thuật xếp hạng ngữ nghĩa hoặc học sâu để nâng cao độ chính xác của quá trình truy xuất [11] .

b) Tạo câu trả lời trung gian

Việc tạo câu trả lời trung gian là một bước quan trọng để sinh ra các phản hồi cụ thể cho truy vấn người dùng dựa trên các bản tóm tắt cộng đồng. Phương pháp này sử dụng một mô hình ngôn ngữ lớn (LLM), chẳng hạn như GPT-4o-mini, để phân tích từng bản tóm tắt và tạo ra một câu trả lời phù hợp với nội dung truy vấn [4]. Quá trình này tận dụng khả năng hiểu ngữ cảnh sâu sắc của LLM, cho phép hệ thống sinh ra các câu trả lời mang tính ngữ nghĩa cao, thay vì chỉ dựa trên việc khớp từ khóa đơn giản.

Quy trình tạo câu trả lời trung gian bắt đầu bằng việc kết hợp bản tóm tắt cộng đồng với truy vấn người dùng thông qua một mẫu prompt được thiết kế cẩn thận (mô tả ở phần PHỤ LỤC A.3). Prompt này chỉ đạo LLM tập trung vào việc sử dụng thông tin trong bản tóm tắt, loại bỏ các suy đoán bên ngoài, và tạo ra câu trả lời ngắn gọn nhưng đầy đủ.

Ưu điểm của phương pháp này nằm ở khả năng xử lý các truy vấn phức tạp yêu cầu suy luận ngữ nghĩa, chẳng hạn như các câu hỏi đòi hỏi tổng hợp thông tin từ nhiều khía cạnh. So với các hệ thống truy vấn truyền thống, việc sử dụng LLM cho phép tạo ra các câu trả lời có tính tự nhiên và phù hợp với ngôn ngữ người dùng. Tuy nhiên, thách thức bao gồm sự phụ thuộc vào chất lượng của bản tóm tắt cộng đồng và thiết kế prompt, cũng như chi phí tính toán liên quan đến việc gọi LLM nhiều lần. Các nghiên cứu tương lai có thể khám phá các kỹ thuật nén prompt hoặc tối ưu hóa hiệu suất LLM để cải thiện khả năng mở rộng.

c) Tổng hợp câu trả lời

Tổng hợp câu trả lời là bước cuối cùng trong quy trình truy vấn, nhằm kết hợp các câu trả lời trung gian từ các cộng đồng thành một phản hồi cuối cùng mạch lạc, ngắn gọn, và bao quát. Phương pháp này sử dụng LLM để phân tích và gộp các câu trả lời trung gian, loại bỏ thông tin lặp lại và đảm bảo rằng phản hồi cuối cùng phản ánh đầy đủ các khía cạnh quan trọng của truy vấn [4]. Quá trình này không chỉ tăng cường tính chính xác mà còn cải thiện trải nghiệm người dùng bằng cách cung cấp một câu trả lời thống nhất, dễ hiểu.

Quy trình tổng hợp bao gồm các bước sau: (1) lọc các câu trả lời trung gian để loại bỏ các phản hồi không liên quan hoặc thiếu thông tin; (2) sử dụng LLM để phân tích và gộp các câu trả lời còn lại, ưu tiên các thông tin bổ sung lẫn nhau; (3) tạo ra một câu trả lời cuối cùng thông qua một mẫu prompt (mô tả ở phần PHỤ LỤC A.4) yêu cầu tính ngắn gọn và mạch lạc.

So với các phương pháp tổng hợp truyền thống, chẳng hạn như ghép nối thủ công hoặc dựa trên quy tắc, cách tiếp cận này vượt trội nhờ khả năng xử lý các câu trả lời trung gian có độ phức tạp và độ dài khác nhau. Việc sử dụng LLM cho phép hệ thống tự động xác định các thông tin quan trọng và loại bỏ các yếu tố dư thừa, từ đó cải thiện chất lượng phản hồi. Tuy nhiên, thách thức nằm ở việc đảm bảo tính nhất quán giữa các câu trả lời trung gian và quản lý chi phí tính toán khi xử lý nhiều cộng đồng. Các hướng nghiên cứu tương lai có thể tập trung vào việc phát triển các kỹ thuật tổng hợp hiệu quả hơn, chẳng hạn như sử dụng các mô hình nhẹ hoặc các phương pháp học tăng cường để tối ưu hóa quá trình này [11].

d) Thang đo đánh giá

Để đánh giá toàn diện hiệu suất của hệ thống hỏi đáp du lịch dựa trên GraphRAG và LlamaIndex, chúng tôi đề xuất ba độ đo bổ sung: Khả năng trả lời (Answerability) và Đáp ứng yêu cầu truy vấn (Question Relevance/Requirement

Satisfaction). Các độ đo này được thiết kế để đánh giá khả năng phản hồi, tính chính xác và mức độ phù hợp của câu trả lời đối với truy vấn người dùng.

Khả năng trả lời (Answerability/Answer Coverage): Đây là tỷ lệ truy vấn mà hệ thống có thể trả lời được, tức là không trả về phản hồi "Không tìm thấy thông tin". Độ đo này phản ánh khả năng của hệ thống trong việc cung cấp câu trả lời cho các truy vấn dựa trên cơ sở tri thức đã xây dựng. Công thức: Answerability = (Số truy vấn có câu trả lời) / (Tổng số truy vấn). Với mỗi truy vấn, kiểm tra phản hồi của hệ thống. Nếu hệ thống có thể trả lời, thì tính là có trả lời. Tính tỷ lệ phần trăm số truy vấn được trả lời trên tổng số truy vấn.

Đáp ứng yêu cầu truy vấn (Question Relevance/Requirement Satisfaction): Độ đo này đánh giá mức độ câu trả lời đáp ứng đúng trọng tâm và yêu cầu của truy vấn, đảm bảo câu trả lời không lan man, không thiếu ý quan trọng và phù hợp với ý định của người dùng. Công thức: Requirement Satisfaction = (Tổng điểm đáp ứng yêu cầu) / (Số câu trả lời đã trả lời). Đánh giá thủ công từng câu trả lời, chấm điểm theo thang 0-1, trong đó 1 là đáp ứng hoàn toàn và 0 là không đáp ứng. Tùy chọn, có thể chi tiết hóa thành thang 1-5, dựa trên các tiêu chí: đúng trọng tâm câu hỏi, đủ ý cần thiết, không thừa, không thiếu thông tin quan trọng.

Đối với việc đánh giá, chúng tôi dùng cả đánh giá thủ công và đánh giá tự động với LLM. Cụ thể, ở độ đo Answer Coverage, chúng tôi sẽ thực hiện giá thủ công trực tiếp, bằng việc tinh chỉnh Prompt trong quá trình test, để khi câu hỏi không thể trả lời được thì sẽ trả về một ký hiệu đã xác định trước, nhằm dễ dàng đếm được các câu mà hệ thống không thể trả lời với các dữ liệu hiện có. Với độ đo Requirement Satisfaction, chúng tôi cũng dùng Prompt đưa vào API lại câu hỏi gốc và câu trả lời để nhờ đánh giá xem câu trả lời có đã đáp ứng được yêu cầu của câu hỏi hay chưa. Ở mọi trường hợp đều đánh giá bằng các phương pháp, Prompt như nhau để đảm bảo được tính công bằng trong đánh giá.

III. KẾT QUẢ THỰC NGHIỆM

Dataset: Chúng tôi tiến hành thực nghiệm công việc xây dựng, truy xuất đồ thị tri thức bằng GraphRAG với LlamaIndex để xây dựng 3 hệ hỏi đáp nhỏ được giới thiệu ở Bảng 1.

Bảng 1. Giới thiệu chi tiết về các hệ hỏi đáp thực nghiệm

Hệ hỏi đáp	Lĩnh vực	Ngôn ngữ	Các chủ đề
1	Du lịch	Tiếng Việt	Quận 1, Quận 3, Quận 5, Quận 10, Quận 11
2	Thể thao	Tiếng Anh	Bóng đá, Bóng rỗ, Quần vợt, Bóng chuyền, Golf
3	Sinh học	Tiếng Pháp	Nấm, Bò sát, Cây thân gỗ, Chim, Cá

Với mỗi hệ hỏi đáp, chúng tôi sẽ sử dụng 5 bài văn bản khác nhau về 5 chủ đề khác nhau trong lĩnh vực để test. Thử nghiệm luân phiên dữ liệu văn bản và tập câu hỏi của 5 chủ đề. Mỗi chủ đề sẽ thử nghiệm với 100 câu hỏi, tổng cộng 500 câu tất cả. Thử nghiệm chất lượng trả lời của hệ hỏi đáp trên từng tập dữ liệu - câu hỏi của từng chủ đề, câu hỏi nằm trong phạm vi của văn bản đã cung cấp.

Ngoài ra chúng tôi cũng tiến hành truy xuất và trả lời câu hỏi kiểm tra với hai phương pháp khác để so sánh với hệ thống GraphRAG+LlmaIndex, cho bởi Bảng 2.

Bảng 2. Các phương pháp sử dụng thực nghiệm

STT	Phương pháp	Mô tả
1	LLM	Sử dụng API của OpenAI với mô hình GPT-4o-mini, đưa thẳng câu hỏi test để nhận câu trả lời (Câu trả lời hoàn toàn dựa vào GPT)
2	BM25	Sử dụng thuật toán BM25 (Best Match 25) để lựa chọn phần thông tin liên quan nhất, sau đó truyền vào API OpenAI GPT-4o-mini (Chỉ trả lời dựa trên kiến thức truy vấn)
3	GraphRAG + LlamaIndex	Trích xuất từ văn bản đầu vào tạo đồ thị tri thức, sau đó truy xuất để trả lời. Sử dụng GPT-4o-mini để trả lời (Chỉ trả lời dựa trên kiến thức truy vấn)

Kết quả: Kết quả thực nghiệm được trình bày ở các bảng sau:

Bảng 3. Kết quả thực nghiệm hệ thống hỏi đáp du lịch tiếng Việt

	Quận 11	**Quận 10**	**Quận 5**	**Quận 3**	**Quận 1**	**Trung bình**
Số ký tự của văn bản	4379	4491	4076	4950	4113	4402
LLM's Answer Coverage	**0.92**	**0.73**	**0.88**	**0.71**	**0.90**	**0.83**
BM25 Answer Coverage	**0.91**	**0.84**	**0.84**	**0.79**	**0.86**	**0.85**
Our Answer Coverage	**0.95**	**0.96**	**0.89**	**0.93**	**0.92**	**0.93**
LLM's Requirement Satisfaction	**0.90**	**0.73**	**0.85**	**0.69**	**0.88**	**0.81**
BM25 Requirement Satisfaction	**0.86**	**0.90**	**0.79**	**0.87**	**0.82**	**0.85**
Our Requirement Satisfaction	**0.92**	**0.94**	**0.86**	**0.91**	**0.89**	**0.89**

Bảng 4. Kết quả thực nghiệm hệ thống hỏi đáp thể thao tiếng Anh

	Bóng đá	Bóng rổ	Quần vợt	Bóng chuyền	Golf	**Trung bình**
Số ký tự của văn bản	4754	4823	4342	4157	4278	4,471
LLM's Answer Coverage	**0.75**	**0.86**	**0.70**	**0.85**	**0.94**	**0.82**
BM25 Answer Coverage	**0.62**	**0.64**	**0.74**	**0.75**	**0.72**	**0.69**
Our Answer Coverage	**0.84**	**0.87**	**0.87**	**0.86**	**0.94**	**0.88**
LLM's Requirement Satisfaction	**0.86**	**0.78**	**0.83**	**0.81**	**0.93**	**0.84**
BM25 Requirement Satisfaction	**0.62**	**0.65**	**0.72**	**0.65**	**0.60**	**0.65**
Our Requirement Satisfaction	**0.82**	**0.84**	**0.74**	**0.85**	**0.94**	**0.84**

Bảng 5. Kết quả thực nghiệm hệ thống hỏi đáp sinh học tiếng Pháp

	Nấm	Bò sát	Cây thân gỗ	Chim	Cá	**Trung bình**
Số ký tự của văn bản	4577	4736	4616	4683	4429	4,608
LLM's Answer Coverage	**0.77**	**0.80**	**0.74**	**0.79**	**0.75**	**0.77**
BM25 Answer Coverage	**0.65**	**0.64**	**0.62**	**0.71**	**0.62**	**0.65**
Our Answer Coverage	**0.83**	**0.84**	**0.75**	**0.76**	**0.75**	**0.78**
LLM's Requirement Satisfaction	**0.74**	**0.76**	**0.83**	**0.81**	**0.73**	**0.77**
BM25 Requirement Satisfaction	**0.63**	**0.64**	**0.72**	**0.71**	**0.70**	**0.68**
Our Requirement Satisfaction	**0.81**	**0.76**	**0.76**	**0.77**	**0.78**	**0.78**

Kết quả cho thấy phương pháp GraphRAG + LlamaIndex (gọi tắt là Ours) vượt trội hơn cả về độ phủ câu trả lời (Answer Coverage) lẫn mức độ thỏa mãn yêu cầu (Requirement Satisfaction) so với hai cách tiếp cận còn lại, bất kể lĩnh vực hay ngôn ngữ đầu vào. Trung bình trên ba lĩnh vực, Our đạt độ phủ 0.86 so với 0.81 của LLM thuần túy và 0.73 của BM25+LLM; tương tự, mức độ thỏa mãn yêu cầu đạt 0.87, cao hơn đáng kể so với 0.81 (LLM) và 0.73 (BM25) (xem Bảng 3–5). Sự ổn định về hiệu năng của GraphRAG cũng được thể hiện qua hệ số biến thiên thấp giữa các chủ đề, chứng tỏ khả năng khái quát tốt hơn khi làm việc với các văn bản có độ dài và cấu trúc nội dung khác nhau.

Nhìn chung, các kết quả minh chứng cho hai ưu điểm chủ yếu của GraphRAG + LlamaIndex:

Tăng cường độ phủ thông tin: bằng cách xây dựng đồ thị tri thức từ các đoạn văn bản, hệ thống có khả năng khái quát và truy cập đa hướng các thực thể, mối quan hệ, nên ít bỏ sót thông tin hơn so với việc chỉ dựa vào BM25 hay để LLM "tự do" sinh câu trả lời.

Giảm thiểu hiện tượng sai lệch và không nhất quán: BM25 và LLM đôi khi dẫn đến mất ngữ cảnh hoặc tập trung quá hẹp vào một vài câu chứa từ khóa, trong khi LLM thuần túy dễ đưa ra thông tin không chính xác nếu thiếu dữ liệu nền. Trong khi đó, GraphRAG giúp định hướng truy vấn vào các cấu trúc đã được đánh chỉ mục và nối kết, từ đó cải thiện đáng kể tính chính xác và nhất quán của câu trả lời.

Tuy nhiên, phương pháp này cũng đòi hỏi bước xử lý tiền đề – trích xuất và xây dựng đồ thị – khiến chi phí tính toán và độ phức tạp hệ thống tăng lên. Trong tương lai, cần tối ưu tiếp thuật toán trích xuất quan hệ và nén đồ thị để cân bằng giữa chất lượng trả lời và hiệu năng triển khai thực tế.

IV. KẾT LUẬN

LLaMA Index và GraphRAG đại diện cho một bước tiến quan trọng trong việc tự động hóa xây dựng và truy vấn đồ thị tri thức, mang lại hiệu quả vượt trội trong quản lý và truy xuất thông tin phức tạp. Phương pháp đề xuất trong bài báo cho phép trích xuất thực thể, quan hệ và xử lý truy vấn ngôn ngữ tự nhiên một cách linh hoạt, vượt xa các phương pháp truyền thống như SPARQL. Thực nghiệm trên tập dữ liệu du lịch Quận 10 (cũ), TP. Hồ Chí Minh đã chứng minh tính khả thi và độ chính xác của hệ thống. Tuy nhiên, các thách thức về chi phí tính toán và chất lượng dữ liệu vẫn cần được giải quyết.

Nghiên cứu tương lai sẽ tập trung vào tối ưu hóa hiệu suất tính toán và mở rộng ứng dụng của LLaMA Index và GraphRAG trong các lĩnh vực như y học, giáo dục và thương mại điện tử, nhằm nâng cao khả năng quản lý tri thức và tìm kiếm ngữ nghĩa.

V. LỜI CẢM ƠN

Nghiên cứu được tài trở bởi Trường Đại học Ngoại ngữ - Tin học Thành phố Hồ Chí Minh trong khuôn khổ Đề tài mã số HSV2024-06

VI. TÀI LIỆU THAM KHẢO

[1] A. Hogan *et al.*, "Knowledge Graphs," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, Jul. 2021, doi: 10.1145/3447772.
[2] A. Singhal, "Introducing the knowledge graph: things, not strings," Official google blog, 5(16), 3.
[3] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *Adv Neural Inf Process Syst*, vol. 33, pp. 1877–1901, 2020, Accessed: Jun. 17, 2025. [Online]. Available: https://commoncrawl.org/the-data/
[4] H. Han *et al.*, "Retrieval-Augmented Generation with Graphs (GraphRAG)," Dec. 2024, Accessed: Jun. 17, 2025. [Online]. Available: https://arxiv.org/pdf/2501.00309
[5] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, Aug. 2007, doi: 10.1075/LI.30.1.03NAD/CITE/REFWORKS.
[6] A. Seaborne and Eric Prud'hommeaux, "SPARQL query language for RDF," W3C Recommendation, W3C (2008).
[7] N. Francis *et al.*, "Cypher: An evolving query language for property graphs," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 1433–1445, May 2018, doi: 10.1145/3183713.3190657;CTYPE:STRING:BOOK.
[8] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications," *IEEE Trans Neural Netw Learn Syst*, vol. 33, no. 2, pp. 494–514, Feb. 2022, doi: 10.1109/TNNLS.2021.3070843.
[9] LLaMA Index Documentation, "LLaMA Index: A Framework for Contextual Data Querying," LLaMA Index Documentation.
[10] J. Z. Pan *et al.*, "Large Language Models and Knowledge Graphs: Opportunities and Challenges," *Transactions on Graph Data and Knowledge*, vol. 1, no. 1, pp. 2:1-2:38, 2023, doi: 10.4230/TGDK.1.1.2.
[11] F. Chung and L. Lu, "Connected Components in Random Graphs with Given Expected Degree Sequences," *Annals of Combinatorics 2002 6:2*, vol. 6, no. 2, pp. 125–145, Nov. 2002, doi: 10.1007/PL00012580.
[12] S. Sahu, K. Kothapalli, and D. S. Banerjee, "Fast Leiden Algorithm for Community Detection in Shared Memory Setting," *ACM International Conference Proceeding Series*, pp. 11–20, Aug. 2024, doi: 10.1145/3673038.3673146;WGROUP:STRING:ACM.

PHỤ LỤC

1. Prompt dùng cho trích xuất thực thể - quan hệ từ văn bản

Bảng 6. Prompt dùng cho trích xuất thực thể - quan hệ từ văn bản

Thành phần	**Nội dung**
Mục tiêu	Dựa trên văn bản đầu vào, trích xuất các thực thể là địa danh hoặc nơi chốn cụ thể tại Quận 10, Thành phố Hồ Chí Minh, cùng với các mối quan hệ rõ ràng giữa chúng. Chỉ các địa danh có tên riêng (ví dụ: "Chợ hoa Hồ Thị Kỷ", "Vạn Hạnh Mall") mới được trích xuất; các khái niệm chung chung như "Du lịch", "Ẩm thực" hoặc "Khu vực" sẽ bị loại bỏ.
Ràng buộc	Tối đa {max_knowledge_triplets} bộ ba thực thể – quan hệ được trích xuất, tập trung vào thông tin hữu ích cho lĩnh vực du lịch Quận 10.
Trích xuất thực thể	Mỗi thực thể bao gồm: (i) entity_name – tên địa danh cụ thể, viết hoa chữ cái đầu; (ii) entity_type – loại địa danh (ví dụ: chợ, trung tâm thương mại, đền chùa, quán cà phê, công viên); (iii) entity_description – mô tả chi tiết liên quan đến hoạt động du lịch và trải nghiệm của du khách.
Định dạng thực thể	("entity"\<$entity_name$ <entity_type>$$$$<entity_description>)
Trích xuất quan hệ	Xác định các cặp thực thể nguồn – đích có mối quan hệ rõ ràng như "nằm tại", "gần", "thuộc khu vực" hoặc "có hoạt động". Mỗi quan hệ đi kèm mô tả nhằm làm rõ giá trị du lịch.
Định dạng quan hệ	("relationship"\<$source_entity$ <target_entity>\<$relation$ <relationship_description>)
Dữ liệu đầu vào	Văn bản: {text}
Kết quả đầu ra	Danh sách các thực thể và mối quan hệ liên quan duy nhất đến các địa danh du lịch cụ thể tại Quận 10.

2. Prompt dùng cho việc tạo các bản tóm tắt cộng đồng

Bảng 7. Prompt dùng cho việc tạo các bản tóm tắt cộng đồng

Thành phần	**Nội dung**
Vai trò	Bạn là một chuyên gia phân tích đồ thị tri thức trong lĩnh vực du lịch.
Dữ liệu đầu vào	Tập hợp các quan hệ có dạng: địa danh 1 → địa danh 2 → loại quan hệ → mô tả chi tiết.
Nhiệm vụ	Tạo một bản tóm tắt ngắn gọn nhưng đầy đủ, phản ánh toàn diện nội dung và cấu trúc của cộng đồng du lịch.
Yêu cầu	(1) Sử dụng đầy đủ thông tin thực thể – quan hệ; (2) gộp các mối quan hệ tương tự để tránh lặp lại; (3) diễn đạt tự nhiên, logic và liền mạch; (4) ưu tiên thông tin giúp người đọc hiểu rõ chức năng và vai trò của các điểm đến trong cộng đồng.
Mục đích	Bản tóm tắt được sử dụng làm đầu vào cho hệ thống hỏi–đáp ở các bước tiếp theo.
Phong cách viết	Văn xuôi, ngắn gọn, rõ ràng; hạn chế liệt kê cứng nhắc, ưu tiên tổng hợp và khái quát.

3. Prompt tạo câu trả lời trung gian của mỗi cộng đồng

Bảng 8. Prompt tạo câu trả lời trung gian của mỗi cộng đồng

Thành phần	**Nội dung**
Dữ liệu đầu vào	Bản tóm tắt cộng đồng: {community_summary}
Câu hỏi	{query}
Nhiệm vụ	Trả lời câu hỏi chỉ dựa trên nội dung của bản tóm tắt cộng đồng được cung cấp.
Ràng buộc	Không sử dụng kiến thức bên ngoài; câu trả lời phải chính xác, ngắn gọn và đúng trọng tâm.
Trường hợp thiếu thông tin	Nếu không có đủ thông tin để trả lời, xuất ra: "Không tìm thấy thông tin trong bản tóm tắt"
Kết quả đầu ra	Câu trả lời ngắn gọn, phản ánh đúng nội dung của cộng đồng tương ứng.

4. Prompt dùng để tạo câu trả lời toàn cục cuối cùng

Bảng 9. Prompt dùng để tạo câu trả lời toàn cục cuối cùng

Thành phần	**Nội dung**
Dữ liệu đầu vào	Tập hợp các câu trả lời trung gian từ nhiều cộng đồng khác nhau: {filtered_answers}
Nhiệm vụ	Tổng hợp các câu trả lời trung gian thành một câu trả lời cuối cùng, ngắn gọn, rõ ràng và đầy đủ thông tin.
Ràng buộc	Không lặp lại thông tin; không suy đoán hoặc bổ sung kiến thức mới; chỉ sử dụng nội dung đã có trong các câu trả lời trung gian; gộp các ý tương đồng.
Kết quả đầu ra	Câu trả lời toàn cục, súc tích và không dư thừa.

APPLYING LLAMAINDEX AND GRAPHRAG FOR KNOWLEDGE GRAPH CONSTRUCTION AND QUERYING

Pham Ngoc Bao, Dinh Hung, Nguyen Quang Tan, Nguyen Hoang Minh Nhat, Thuy-A Nguyen *

ABSTRACT— This paper presents a method that combines LLaMA Index and GraphRAG to construct and query knowledge graphs from plain-text data. We introduce an automated pipeline for entity and relation extraction as well as community-based summarization, integrating a large language model (LLM) with a community-detection algorithm. The resulting system supports efficient processing of natural-language queries. Experiments on a tourism dataset from Ho Chi Minh City demonstrate its ability to answer complex queries accurately. Empirical results show that our approach improves query-response effectiveness by approximately 4% compared with using an LLM alone. We also discuss challenges related to computational cost and the need for high-quality data, along with the method's potential applications in knowledge management and semantic search.

Keywords — Large Language Models, Knowledge Graph, GraphRAG, LLaMAIndex

ThS. Nguyễn Thị Thúy A nhận bằng thạc sĩ ngành Khoa học máy tính tại trường ĐH Khoa học tự nhiên, ĐHQG-TP.HCM vào năm 2018. Hiện tại Thạc sĩ Thúy A là giảng viên tại khoa Công nghệ thông tin, trường ĐH Ngoại ngữ -Tin học TP. Hồ Chí Minh (HUFLIT).

Hướng nghiên cứu chính: Xử lý ngôn ngữ tự nhiên, Trí tuệ nhân tạo.

TS. Đinh Hùng nhận bằng Thạc sĩ Công nghệ thông tin từ Trường Đại học Khoa học tự nhiên (ĐHQG-TP.HCM) năm 2003, bằng MBA từ Đại học Lincoln năm 2014, và bằng Tiến sĩ từ Trường Đại học Lạc Hồng năm 2020. Hiện ông là Trưởng ngành Thương mại điện tử tại Trường Đại học Ngoại ngữ – Tin học TP.HCM (HUFLIT).

Hướng nghiên cứu chính: Thương mại điện tử, Quản trị kinh doanh, và Quản lý công nghệ thông tin.

Ông Nguyễn Hoàng Minh Nhật nhận bằng Thạc sĩ Khoa học máy tính từ Học viện Kỹ thuật quân sự (MTA) vào năm 2016. Hiện ông là giảng viên Bộ môn Khoa học máy tính, Khoa Công nghệ thông tin, Trường Đại học Văn Hiến (VHU).

Hướng nghiên cứu chính: Trí tuệ nhân tạo, Máy học, Xử lý ngôn ngữ tự nhiên và Hệ thống thông minh.

Phạm Ngọc Bảo tốt nghiệp chuyên ngành Khoa học dữ liệu, ngành Công nghệ thông tin tại Trường ĐH Ngoại ngữ –Tin học TP. Hồ Chí Minh (HUFLIT). Hiện đang học thạc sĩ tại trường ĐH Công Thương TPHCM (HUIT)

Hướng nghiên cứu chính: Xử lý ngôn ngữ tự nhiên, Trí tuệ nhân tạo.

Ông Nguyễn Quang Tấn nhận bằng Tiến sĩ (Ph.D.) từ Trường Đại học Quốc gia Belorussia (Liên xô cũ) vào năm 1983. Hiện tại là Trưởng Bộ môn thuộc Khoa Công nghệ Thông tin, trường Đại học Văn Hiến (VHU). Hướng nghiên cứu chính: Trí tuệ nhân tạo, Khai phá dữ liệu lớn, eLearning, Xử lý Ngôn ngữ tự nhiên và Hệ thống thông minh.