NGHIÊN CỨU VÀ PHÁT TRIỂN KHUNG LÀM VIỆC LẶP THÍCH ỨNG CHO BÀI TOÁN HỎI ĐÁP TRÊN TÀI LIỆU DÀI ĐA PHƯƠNG THỨC

Lê Nguyễn Anh Khoa - 23520742

Cáp Kim Hải Anh - 23520036

Tóm tắt

- Lớp: CS519.Q11.KHTN
- Link Github của nhóm: https://github.com/LeNguyenAnhKhoa/CS519.Q11
- Link YouTube video: https://www.youtube.com/watch?v=kIEodl1Kpog
- Họ tên các thành viên:

Lê Nguyễn Anh Khoa - 23520742

Cáp Kim Hải Anh - 23520036

Giới thiệu

- Bối cảnh: Sự bùng nổ dữ liệu phi cấu trúc (pdf, slide) chứa văn bản lẫn nhiều hình ảnh hoặc biểu đồ phức tạp
- Vấn đề: Các mô hình RAG (Retrieval-Augmented Generation) hiện tại chỉ tìm kiếm một lần, dễ bỏ sót thông tin
- Hạn chế: RAG thường thất bại khi thông tin nằm rải rác ở nhiều trang hoặc cần suy luận chéo giữa ảnh và text
- Giải pháp đề xuất: Áp dụng cơ chế Lặp thích ứng để tự động đánh giá và tìm kiếm lại những gì còn thiếu sót, thay vì trả lời ngay lập tức dựa trên dữ liệu không đủ.

Giới thiệu

- Bài toán: Xây dựng hệ thống hỏi đáp tự động trên tài liệu dài
- Input:
  - Tập dữ liệu D gồm N trang (chứa text và ảnh hoặc biểu đồ)
  - Câu truy vấn ngôn ngữ tự nhiên Q từ người dùng
- Output:
  - Câu trả lời A chính xác dạng văn bản
  - Trích dẫn nguồn (số trang, vùng ảnh) dùng để trả lời
- Tính chất: Input đa dạng, phi cấu trúc; Output đòi hỏi tính tổng hợp và suy luận logic cao
- Cách tiếp cận: Coi việc tìm kiếm là bài toán tối ưu hoá

Mục tiêu

- Đề tài tập trung vào 3 mục tiêu cụ thể:
  1. Nghiên cứu lý thuyết: Phân tích cơ chế React Reasoning trong RAG.
  2. Xây dựng pipeline hoàn chỉnh: Hiện thực hóa quy trình hỏi đáp lặp, tự động sinh câu hỏi phụ để tìm tin thiếu.
  3. Đánh giá thực nghiệm: Kiểm chứng tính khả thi và độ chính xác trên tập dữ liệu chuẩn (SlideVQA/MMLongBench) so với RAG cơ bản.
- Mục tiêu chung: Mô hình chạy thực tế, áp dụng triệt để tư duy phản biện để phân tích sâu các trường hợp mô hình thất bại.

Nội dung và Phương pháp

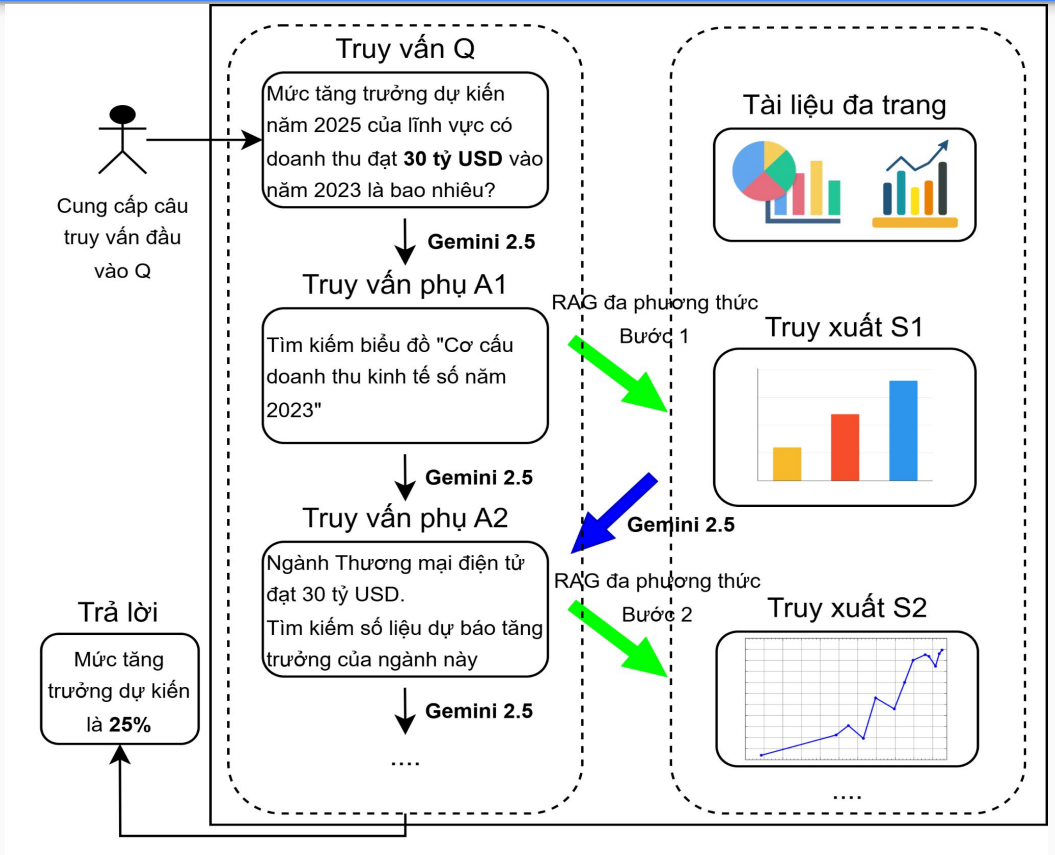
Áp dụng Computational Thinking:

- **Decomposition:** Chia hệ thống thành 2 module chính độc lập:
  - Model truy xuất: Dùng mô hình embedding đa phương thức (ColPali hoặc tương đương) để tìm trang tài liệu phù hợp.
  - Model đánh giá: Dùng LLM đóng vai trò bộ não để xem thông tin tìm được đã đủ để trả lời chưa.
- **Abstraction:** So sánh thông tin cần thiết để trả lời câu hỏi với thông tin đã có. Mô hình hóa sự "thiếu hụt thông tin" thành các biến số để tối ưu hóa.
- **Algorithm Design:** Quy trình xử lý lặp
  - Bước 1: Nhận câu hỏi Q, sinh ra sub-query để tìm ra thông tin ban đầu liên quan đến sub-query từ tập **D,** thêm vào tập **S**

Nội dung và Phương pháp

- **Algorithm Design:** Quy trình xử lý lặp
  - Bước 2: LLM đánh giá phần thông tin này, đưa ra **information gain** nếu nó vượt qua ngưỡng có thể giữ lại, ngược lại bỏ qua thông tin này để tránh việc bị nhét thông tin không cần thiết
  - Bước 3 (Pattern Recognition): Sinh câu hỏi phụ (sub-query) tiếp theo dựa trên phần thiếu.
  - Bước 4: Dùng sub-query tiếp theo này để tìm kiếm lại chính xác hơn, cập nhật tập **S** mới và LLM tiếp tục đánh giá.
  - Bước 5: Cập nhật thông tin và lặp lại B2 cho đến khi đủ tin hoặc đạt giới hạn lặp tối đa.
- **Công cụ:** Python, PyTorch, LangChain

Nội dung và Phương pháp



Kết quả dự kiến

- Báo cáo kỹ thuật: So sánh độ chính xác (Accuracy/F1) giữa phương pháp đề xuất và Baseline (RAG truyền thống).
- Phân tích lỗi: Đánh giá các trường hợp thất bại để hiểu rõ giới hạn của phương pháp xấp xỉ thông tin.
- Mức độ khả thi: Dựa trên framework có sẵn, tập trung vào tinh chỉnh logic.

Tài liệu tham khảo

- S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” ICRL, 2023
- M. Faysse et al., “ColPali: Efficient document retrieval with vision language models,” arXiv, 2024
- R. Tanaka et al., “SlideVQA: A dataset for document visual question answering,” AAAI 2023
- P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” NeurIPS, 2020