BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC PHENIKAA

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: XỬ LÍ NGÔN NGỮ TỰ NHIÊN

Mã HP: CSE703065

ĐỀ TÀI : HỆ THỐNG CHATBOT TRONG TRA CỨU VĂN BẢN LUẬT DƯỢC (NHÓM 06)

Sinh viên: **Phạm Duy Quang**
Nguyễn Hữu Đăng Khoa
Nguyễn Đức Mạnh

Ngành học: **Khoa Học Máy Tính**

Khóa: **K17**

Tên lớp: **AI&KHDL_1**

Giảng viên hướng dẫn: **PGS. Phạm Tiến Lâm**

Hà Nội, Năm 2025

Bảng phân công công việc

Tên	Công việc	Mức độ đóng góp	Tự chấm điểm
Nguyễn Hữu Đăng Khoa 23010901	Thu thập và tiền xử lí dữ liệu	33,33%	10
Nguyễn Đức Mạnh 23010902	Thu thập và tiền xử lí dữ liệu	33,33%	10
Phạm Duy Quang (trưởng nhóm) 23017167	Xây dựng và huấn luyện mô hình	33,33%	10

Mục Lục

Chương 1: Giới thiệu tổng quan

1.1. Lý do chọn đề tài

Đời sống con người hiện tại đang phải đối mặt với một số vấn đề như sau:

- **Sự phức tạp và đồ sộ của hệ thống văn bản pháp luật**: Lĩnh vực dược phẩm tại Việt Nam được quản lý bởi một hệ thống văn bản pháp luật vô cùng phức tạp và đa dạng, bao gồm Luật Dược, các Nghị định, Thông tư, và quyết định... Chúng thường xuyên được cập nhật, bổ sung hoặc thay thế.
- **Khó khăn trong việc tra cứu thủ công:** Đối với các dược sĩ, doanh nghiệp kinh doanh dược phẩm, hoặc thậm chí là người dân, việc tìm kiếm một thông tin pháp lý cụ thể (ví dụ: quy định về nhãn thuốc, thủ tục đăng ký thuốc mới, quy định về quảng cáo) là một quá trình tốn nhiều thời gian và công sức. Họ phải rà soát qua nhiều văn bản, đối chiếu các điều khoản, và không phải lúc nào cũng tìm được câu trả lời chính xác ngay lập tức.
- **Nguy cơ từ việc tiếp cận thông tin sai lệch:** Việc tra cứu thủ công dễ dẫn đến sai sót, hiểu nhầm hoặc bỏ lỡ các cập nhật mới. Trong ngành dược, một sai sót về pháp lý có thể dẫn đến hậu quả nghiêm trọng, từ xử phạt hành chính, thu hồi sản phẩm đến ảnh hưởng trực tiếp tới sức khỏe cộng đồng.
- **Nhu cầu truy vấn thông tin tức thời (24/7):** Nhu cầu về thông tin pháp lý ngành dược không chỉ phát sinh trong giờ hành chính. Các dược sĩ tại nhà thuốc, các nhà quản lý kho, hay bệnh nhân đều có thể cần câu trả lời ngay lập tức. Các kênh hỗ trợ truyền thống như tổng đài hay bộ phận pháp chế thường bị giới hạn về thời gian và nhân lực.

1.2. Mục tiêu của đề tài

- Đề tài này hướng đến việc nghiên cứu và xây dựng một hệ thống Chatbot chuyên biệt cho lĩnh vực Luật Dược tại Việt Nam. Hệ thống có khả năng cung cấp thông tin pháp lý một cách nhanh chóng, chính xác và dễ tiếp cận cho mọi đối tượng.
- **Ý nghĩa thực tiễn**:
  a. Đối với Dược sĩ/Doanh nghiệp: Tiết kiệm thời gian tra cứu, hỗ trợ ra quyết định nhanh chóng, và nâng cao ý thức tuân thủ pháp luật, giảm thiểu rủi ro.
  b. Đối với Người dân: Minh bạch hóa thông tin pháp lý, giúp họ hiểu rõ hơn về quyền lợi và nghĩa vụ của mình khi sử dụng thuốc.
  c. Đối với cơ quan quản lý: Giảm tải cho các bộ phận hỗ trợ, tư vấn và góp phần phổ biến pháp luật hiệu quả hơn.

- **Ý nghĩa khoa học:** Đề tài góp phần vào việc ứng dụng và kiểm chứng hiệu quả của các kỹ thuật AI và NLP tiên tiến trong một lĩnh vực chuyên ngành hẹp, có tính đặc thù cao và đòi hỏi độ chính xác tuyệt đối như pháp luật y tế.

1.3. Phạm vi và giới hạn của hệ thống

Về chức năng:

- Hệ thống tập trung vào việc tra cứu và trả lời các câu hỏi liên quan trực tiếp đến các văn bản pháp luật trong lĩnh vực Dược tại Việt Nam.
- Cung cấp các định nghĩa, giải thích thuật ngữ, và trích dẫn các điều khoản cụ thể khi người dùng yêu cầu.
- Hỗ trợ trả lời các câu hỏi thường gặp (FAQs) về các thủ tục hành chính đơn giản (ví dụ: điều kiện mở nhà thuốc, quy trình đăng ký thuốc).

Về dữ liệu:

- Nguồn dữ liệu đầu vào được giới hạn trong bộ Luật Dược số 105/2016/QH13 của Quốc hội nước CHXH chủ nghĩa Việt Nam ban hành ngày 06/4/2016, có hiệu lực từ ngày 01/01/2017.
- Hệ thống được thiết kế để ưu tiên trả lời bằng tiếng Việt.

Chương 2: Cơ sở lý thuyết

2.1. Giới thiệu về trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên (NLP)

- Trí tuệ nhân tạo (AI - Artificial Intelligence) là một lĩnh vực bao trùm của khoa học máy tính, tập trung vào việc tạo ra các hệ thống có khả năng mô phỏng các hành vi thông minh của con người. Cốt lõi của AI là khả năng học hỏi từ dữ liệu, lập luận để đưa ra quyết định và tự giải quyết vấn đề mà không cần lập trình tường minh cho mọi trường hợp.

- Xử lý ngôn ngữ tự nhiên (NLP - Natural Language Processing) là một nhánh chuyên biệt và quan trọng của AI. Mục tiêu của NLP là cung cấp cho máy tính khả năng đọc, hiểu, phân tích và tạo ra ngôn ngữ của con người một cách có ý nghĩa. NLP đóng vai trò là cầu nối giao tiếp trực tiếp giữa người dùng và máy móc. NLP ứng dụng các kỹ thuật AI để giải quyết bài toán cụ thể đó là hiểu và xử lý ngôn ngữ, cho phép các ứng dụng như chatbot có thể tương tác với con người một cách tự nhiên.

2.2. Kiến trúc mô hình ngôn ngữ lớn (LLM – Large Language Model)

Để giải quyết bài toán tra cứu Luật Dược, hệ thống không chỉ dựa vào kiến thức được huấn luyện trước của LLM, vốn có thể lỗi thời hoặc ảo giác (hallucinate). Thay vào đó, đề tài áp dụng kiến trúc Retrieval-Augmented Generation (RAG) tiên tiến, kết hợp với kỹ thuật tinh chỉnh hiệu quả LoRA.

Kiến trúc gồm hai thành phần chính:

- Mô-đun Truy vấn (Retrieval)
- Mô-đun Sinh (Generation).

Mô-đun Truy vấn: Tìm kiếm Lai (Hybrid Search)

Module truy vấn có nhiệm vụ tìm ra các văn bản pháp luật liên quan và chính xác nhất từ cơ sở dữ liệu (vector database) để làm ngữ cảnh cho LLM. Chúng tôi sử dụng phương pháp Hybrid Search để tận dụng ưu điểm của cả hai trường phái:

- Tìm kiếm Từ khóa (Keyword Search - BM25):
  - Mục đích: Rất hiệu quả trong việc tìm kiếm các thuật ngữ, mã hiệu, hoặc số hiệu văn bản cụ thể.
  - Ưu điểm: Đảm bảo độ chính xác khi người dùng biết rõ họ đang tìm gì.
- Tìm kiếm Ngữ nghĩa (Semantic Search - Dùng Embedding):
  - Mục đích: Đây là lúc Embedding phát huy tác dụng. Mỗi văn bản luật và câu hỏi của người dùng đều được mã hóa thành một vector đa chiều.
  - Cách hoạt động: Hệ thống so sánh sự tương đồng về mặt ý nghĩa giữa vector của câu hỏi và vector của các văn bản.
  - Ưu điểm: Hiểu được ý định của người dùng ngay cả khi họ dùng từ khác.

Bằng cách kết hợp (hybrid) cả hai phương pháp, hệ thống đảm bảo rằng ngữ cảnh được truy xuất vừa có độ chính xác vừa có độ phủ ngữ nghĩa để chuẩn bị cho LLM ở bước sau.

Mô-đun Sinh: LLM Tinh chỉnh bằng LoRA

Sau khi đã có ngữ cảnh (context) từ bước 1, ngữ cảnh này sẽ được kẹp cùng với câu hỏi gốc của người dùng (prompt) và gửi đến Mô hình Ngôn ngữ Lớn (LLM).

Tuy nhiên, một LLM cơ sở (base model) thường quá tổng quát và có thể không giỏi trong việc diễn đạt theo văn phong pháp lý. Vì vậy, chúng tôi áp dụng LoRA (Low-Rank Adaptation).

Vấn đề của Fine-tuning truyền thống: Tinh chỉnh toàn bộ một LLM (ví dụ: 7 tỷ tham số) đòi hỏi tài nguyên tính toán (GPU, VRAM) khổng lồ và tốn thời gian.

Giải pháp của LoRA:

- LoRA là một kỹ thuật tinh chỉnh hiệu quả tham số (Parameter-Efficient Fine-Tuning - PEFT).
- Thay vì cập nhật hàng tỷ tham số, LoRA giữ nguyên trọng số (weights) của mô hình gốc.
- Chèn thêm các bộ điều hợp (adapter) nhỏ, có thể huấn luyện được (gọi là ma trận A và B có thứ hạng thấp - low rank) vào các lớp của mô hình (thường là các lớp Attention).
- Trong quá trình huấn luyện, chỉ các tham số của các adapter này được cập nhật. Số lượng tham số này chỉ chiếm một phần rất nhỏ (ví dụ: <1%) so với mô hình gốc.

Kết quả: Chúng tôi thu được một mô hình LLM đã được chuyên môn hóa cho lĩnh vực Luật Dược, có khả năng diễn đạt chính xác, bám sát văn phong pháp lý và hiểu thuật ngữ chuyên ngành, mà không cần chi phí huấn luyện lại toàn bộ mô hình.

2.3. Tổng quan về mô hình Qwen và Llama

2.3.1. Kiến trúc Qwen3-Embedding-0.6B

Để thực hiện chức năng tìm kiếm ngữ nghĩa trong hệ thống, đề tài sử dụng mô hình Qwen3-Embedding-0.6B. Đây là một mô hình embedding hiện đại có kiến trúc khác biệt so với các mô hình truyền thống (như BERT). Kiến trúc của mô hình này được xây dựng dựa trên các đặc điểm cốt lõi sau:

1. **Nền tảng Decoder-Only (Transformer Decoder)**

Không giống như các mô hình embedding cổ điển (thường là kiến trúc Encoder-Only), Qwen3-Embedding được xây dựng trên nền tảng của Qwen3-0.6B-Base, một Mô hình Ngôn ngữ Lớn (LLM) có kiến trúc Transformer Decoder-Only (tương tự GPT, Llama).

Mô hình này được huấn luyện với nhiệm vụ dự đoán từ tiếp theo (next-token prediction). Quá trình này bắt buộc mô hình phải học cách nén toàn bộ thông tin và ngữ cảnh của chuỗi văn bản đứng trước nó vào một trạng thái ẩn (hidden state) duy nhất. Kiến trúc embedding này khai thác chính khả năng hiểu ngữ cảnh sâu sắc đó.

2. Chiến lược Gom cụm (Pooling Strategy)

Từ nền tảng Decoder-Only, mô hình này không sử dụng token [CLS] (thường thấy ở BERT) để đại diện cho cả câu. Thay vào đó, nó áp dụng chiến lược last_token_pool (gom cụm token cuối cùng).

Trong kiến trúc decoder, trạng thái ẩn của token cuối cùng trong chuỗi chính là nơi hội tụ và tóm tắt ý nghĩa ngữ nghĩa của toàn bộ văn bản đã xử lý. Do đó, hệ thống chỉ cần trích xuất vector biểu diễn của token cuối cùng này để làm vector đại diện (embedding) cho toàn bộ câu hoặc tài liệu.

3. Các Tính năng Kiến trúc Nâng cao

Qwen3-Embedding tích hợp hai tính năng kiến trúc tiên tiến giúp nó trở nên linh hoạt và hiệu quả:

- **Instruction Aware (Nhận biết Chỉ dẫn)**: Mô hình được huấn luyện để hiểu một chỉ dẫn (instruction) được cung cấp kèm theo văn bản đầu vào. Điều này cho phép mô hình khiến việc tạo vector vào một không gian véc-tơ phù hợp với tác vụ cụ thể (ví dụ: Instruct: Tìm kiếm tài liệu pháp lý... sẽ tạo ra vector khác với Instruct: Phân loại văn bản...). Tính năng này giúp tăng độ chính xác đáng kể (1-5%) so với việc chỉ mã hóa văn bản thô.
- **MRL (Matryoshka Representation Learning):** Đây là một kỹ thuật cho phép tạo ra các embedding đa phân giải. Mô hình tạo ra một vector đầy đủ (ví dụ: 1024 chiều), nhưng được huấn luyện đặc biệt để các chiều đầu tiên (ví dụ: 128, 256, 512 chiều) cũng là những vector đại diện có ý nghĩa. Kiến trúc này cho phép người dùng tùy

chỉnh kích thước vector để cân bằng giữa độ chính xác (dùng vector lớn) và chi phí/tốc độ (dùng vector nhỏ để lưu trữ và tìm kiếm).

- Ngoài ra, mô hình hỗ trợ độ dài ngữ cảnh (context length) rất lớn lên đến 32,000 token, cho phép nó mã hóa các tài liệu pháp lý dài mà không cần cắt nhỏ, tránh làm mất ngữ cảnh toàn cục của văn bản.

2.3.2. Kiến trúc Llama-3.2 1B

Llama-3.2-1B là một mô hình ngôn ngữ lớn (LLM) thuộc họ Llama 3.2 do Meta phát triển, được thiết kế với mục tiêu cân bằng giữa hiệu suất cao và hiệu quả tính toán, đặc biệt là cho các môi trường có tài nguyên hạn chế.

Kiến trúc của mô hình này dựa trên nền tảng Transformer, nhưng được tối ưu hóa sâu với các kỹ thuật tiên tiến:

1. Nền tảng Transformer Tự hồi quy (Auto-regressive)

Về cốt lõi, Llama-3.2-1B là một mô hình Transformer Tự hồi quy (Auto-regressive), tương tự như kiến trúc GPT. Nhiệm vụ chính của nó là dự đoán từ (token) tiếp theo trong một chuỗi, khiến nó trở nên lý tưởng cho các tác vụ sinh văn bản (text generation) và đối thoại.

2. Tối ưu hóa Kiến trúc Then chốt

Kiến trúc của Llama 3.2 (bao gồm cả bản 1B) được tối ưu hóa với hai đặc điểm nổi bật:

- **Grouped-Query Attention (GQA):** Đây là một trong những cải tiến kiến trúc quan trọng nhất. Thay vì sử dụng Multi-Head Attention (MHA) tiêu chuẩn (nơi mỗi đầu (head) truy vấn có một cụm (key/value) riêng), GQA cho phép nhiều đầu truy vấn chia sẻ chung một cụm key/value. Kiến trúc này giảm đáng kể chi phí tính toán và băng thông bộ nhớ khi suy luận , cho phép mô hình chạy nhanh hơn và xử lý ngữ cảnh dài hiệu quả hơn.
- **Ngữ cảnh Dài (128k):** Nhờ các tối ưu hóa như GQA, phiên bản 1B này có khả năng xử lý độ dài ngữ cảnh (context length) cực kỳ ấn tượng lên đến 128.000 token. Điều này cho phép nó hiểu và duy trì thông tin trong các văn bản rất dài, vượt trội so với nhiều mô hình lớn hơn.

3. Kỹ thuật Huấn luyện (Pre-training)

Mô hình 1B này không chỉ được huấn luyện trên 9 nghìn tỷ token dữ liệu. Một điểm mấu chốt trong quá trình huấn luyện của nó là chưng cất kiến thức (Knowledge Distillation).

Meta đã sử dụng logits (đầu ra xác suất) từ các mô hình Llama 3.1 8B và 70B lớn hơn để dạy và hướng dẫn cho mô hình 1B này trong giai đoạn pre-training. Điều này giúp mô hình 1B kế thừa được kiến thức và khả năng suy luận phức tạp từ những mô hình lớn hơn nó rất nhiều, giúp nó đạt được hiệu suất vượt trội so với kích thước thật.

4. Tinh chỉnh và Lượng tử hóa (Alignment & Quantization)

Đây là lĩnh vực mà Llama-3.2-1B thể hiện rõ nhất mục tiêu thiết kế của mình (hướng đến thiết bị di động).

Căn chỉnh (Alignment): Mô hình được tinh chỉnh bằng cả hai phương pháp: **Supervised Fine-Tuning (SFT)** và **Direct Preference Optimization (DPO)**. DPO là một kỹ thuật thuộc nhóm RLHF, giúp mô hình tuân thủ hướng dẫn và tạo ra các câu trả lời hữu ích, an toàn mà không cần một mô hình phần thưởng (reward model) riêng biệt.

Kiến trúc Lượng tử hóa (Quantization): Model card mô tả các kỹ thuật lượng tử hóa rất phức tạp, được thiết kế để tối ưu hóa cho nền tảng.

- Quy tắc chung: Mô hình sử dụng sơ đồ lượng tử hóa 4-bit groupwise (nhóm 32) cho các trọng số (weights) của lớp linear và 8-bit dynamic cho các kích hoạt (activations).
- QLoRA (phiên bản của Meta): Đây không phải là QLoRA thông thường. Họ thực hiện Huấn luyện Nhận biết Lượng tử hóa (QAT - Quantization-Aware Training) cho mô hình trước. Sau đó, họ đóng băng phần xương sống đã được lượng tử hóa này và chỉ huấn luyện các adapter LoRA. Cuối cùng, họ tinh chỉnh cả backbone và LoRA bằng DPO. Đây là một quy trình kết hợp QAT và LoRA rất tinh vi.
- SpinQuant: Một kỹ thuật lượng tử hóa khác được áp dụng (kết hợp với GPTQ) để tối ưu hóa hiệu suất suy luận.

2.3.3. Lý do lựa chọn

Việc lựa chọn mô hình cho dự án được quyết định chủ yếu bởi một rào cản kỹ thuật thực tế: giới hạn nghiêm ngặt về tài nguyên phần cứng. Đề tài được thực hiện trên thiết bị chỉ được trang bị 1 GPU NVIDIA RTX 2050 với 4GB VRAM.

Với dung lượng VRAM cực kỳ eo hẹp này, việc triển khai hay tinh chỉnh (fine-tune) các mô hình ngôn ngữ lớn (LLM) phổ biến (như các bản 7B, 13B) là điều bất khả thi, ngay cả khi sử dụng các kỹ thuật lượng tử hóa (quantization) 4-bit.

Do đó, chiến lược của chúng tôi là tìm kiếm các mô hình có hiệu suất tối ưu trên kích thước nhỏ nhất, cho phép hệ thống có thể thực tế chạy và tinh chỉnh (fine-tune) được trên 4GB VRAM.

1. Lựa chọn Mô hình Embedding: Qwen3-Embedding-0.6B

Mô hình này được chọn cho tác vụ truy vấn vì các lý do sau:

- Kích thước siêu nhỏ (0.6B): Với chỉ 600 triệu tham số, đây là một trong những mô hình embedding hiệu suất cao nhỏ gọn nhất hiện nay. Kích thước này giúp giảm tải VRAM khi tải mô hình để mã hóa văn bản.
- Hiệu suất cao so vớt kích thước: Dù nhỏ, Qwen3-Embedding vẫn cho kết quả rất cạnh tranh trên các benchmark (như MTEB), đảm bảo chất lượng cho việc tìm kiếm ngữ nghĩa.
- Kiến trúc Búp bê Nga (MRL): Đây là tính năng át chủ bài cho phần cứng yếu. Kiến trúc Matryoshka (MRL) cho phép người phát triển tùy chỉnh kích thước vector embedding (ví dụ: từ 1024 chiều xuống 512, 256 hoặc 128 chiều) mà vẫn giữ được chất lượng ngữ nghĩa tương đối. Trong khuôn khổ dự án, chúng tôi có thể chọn vector 256 chiều thay vì 1024, giúp giảm 4 lần dung lượng lưu trữ vector database và giảm tải bộ nhớ/VRAM khi tính toán tương đồng.

2. Lựa chọn Mô hình Sinh (LLM): Llama-3.2-1B

Mô hình này được chọn cho tác vụ sinh câu trả lời và được tinh chỉnh bằng LoRA:

- Kích thước vừa vặn (1B): Với 1.23 tỷ tham số, Llama-3.2-1B là một trong số ít các LLM đa ngôn ngữ hiện đại có kích thước đủ nhỏ để có thể hy vọng tải được trên 4GB VRAM sau khi lượng tử hóa.

- Tối ưu cho Lượng tử hóa 4-bit (QLoRA): Đây là yếu tố then chốt. Kế hoạch của dự án là sử dụng QLoRA (lượng tử hóa 4-bit kết hợp LoRA).
    - Một mô hình 1B tham số ở 4-bit chỉ chiếm khoảng ~0.6GB VRAM cho trọng số.
    - Điều này để lại khoảng VRAM còn lại (khoảng 3GB+) cho các thành phần khác (như optimizer, context, và đặc biệt là các adapter LoRA). Nếu không có kỹ thuật QLoRA, việc fine-tune mô hình này là không thể.
- Kiến trúc GQA (Grouped-Query Attention): Mô hình được trang bị GQA, một kiến trúc attention hiệu quả giúp giảm mức sử dụng VRAM và tăng tốc độ suy luận so với Multi-Head Attention (MHA) truyền thống, đặc biệt quan trọng khi VRAM eo hẹp.
- Hiệu suất chưng cất (Distilled Performance): Dù chỉ là 1B, mô hình này được Meta chưng cất kiến thức (knowledge distillation) từ các mô hình Llama 3.1 8B và 70B, giúp nó mang trí tuệ của mô hình lớn trong một mô hình nhỏ gọn.

Kết luận: Cặp đôi Qwen3-Embedding-0.6B (với MRL) và Llama-3.2-1B (với QLoRA) không phải là lựa chọn mạnh nhất, nhưng là sự lựa chọn chiến lược và khả thi nhất về mặt kỹ thuật, cho phép dự án được hoàn thành trên phần cứng chúng tôi.

2.4. Kỹ thuật LoRA Finetuning

2.4.1 Vấn đề của Tinh chỉnh Toàn bộ (Full Fine-Tuning)

Trong phương pháp tinh chỉnh toàn bộ (Full Fine-Tuning - FFT), toàn bộ các tham số (weights) của mô hình nền tảng (pre-trained model) đều được cập nhật trong quá trình huấn luyện.

Giả sử một lớp (ví dụ: lớp Linear) có ma trận trọng số W (với $W \in R^{d \times k}$) ,quá trình FFT sẽ cập nhật nó theo công thức:

$$\boldsymbol{W_{new} = W + \Delta W}$$

Trong đó:

- W là ma trận trọng số gốc (đã được huấn luyện trước).
- ΔW là ma trận thay đổi (ma trận delta) được học trong quá trình tinh chỉnh.

Vấn đề: Với các LLM hiện đại, W có thể có hàng tỷ tham số. Điều này có nghĩa là ΔW cũng có kích thước tương đương, dẫn đến hai nhược điểm chí mạng:

- Chi phí VRAM: Cần một lượng VRAM khổng lồ để lưu trữ gradient và trạng thái của optimizer cho ΔW.
- Chi phí Lưu trữ: Phải lưu trữ một bản sao toàn bộ mô hình (hàng GB hoặc TB) cho mỗi tác vụ mới.

2.4.2 Ý tưởng Cốt lõi của LoRA

LoRA (Low-Rank Adaptation) được đề xuất dựa trên một giả thuyết thực nghiệm: Dù mô hình có số tham số lớn, ma trận thay đổi ΔW cần thiết để thích ứng với một tác vụ mới thường có hạng nội tại thấp (low intrinsic rank).
Nói một cách đơn giản, sự thay đổi ΔW không cần phải phức tạp; nó có thể được biểu diễn một cách hiệu quả chỉ bằng một số ít thông tin.
Thay vì học ma trận ΔW (kích thước $d \times k$) một cách trực tiếp, LoRA đề xuất phân rã (decompose) nó thành hai ma trận nhỏ hơn: B và A.

$$\Delta W \approx B.A$$

Trong đó:

- $B \in R^{d \times r}$
- $A \in R^{r \times k}$
- r là hạng (rank) của sự phân rã.

Điểm mấu chốt: Hạng r được chọn nhỏ hơn rất nhiều so với d và k (ví dụ: r = 8, 16, 64 trong khi d, k có thể là hàng ngàn).

2.5.3 Công thức Toán học và Kiến trúc

Trong LoRA, ma trận trọng số gốc W được đóng băng (frozen) và không bị huấn luyện. Thay vào đó, một nhánh song song được thêm vào.
Quá trình truyền thẳng (forward pass) của một lớp được sửa đổi như sau:
Ban đầu:

$$h = W.x$$

Với LoRA:

$$h = W.x + \Delta W.x$$

$$h = W.x + (B.A).x$$

Để kiểm soát quy mô của sự thay đổi, LoRA thường thêm một hệ số vô hướng (scaling factor) $\propto$ Công thức hoàn chỉnh là:

$$h = W.x + \frac{\propto}{r}(B.A).x$$

Quá trình Huấn luyện:

1. Ma trận W bị đóng băng.
2. Ma trận A được khởi tạo (thường bằng phân phối Gaussian ngẫu nhiên).
3. Ma trận B được khởi tạo bằng 0. (Điều này đảm bảo $\Delta W = 0$ khi bắt đầu, giúp mô hình khởi động ổn định từ trọng số gốc).
4. Chỉ có các tham số của A và B được cập nhật thông qua backpropagation.

2.6.4 Phân tích ưu điểm

Phương pháp này mang lại lợi ích khổng lồ:

Giảm số lượng tham số huấn luyện:

- Full Fine-Tuning: Phải huấn luyện $d \times k$ tham số.
- LoRA: Chỉ phải huấn luyện $(d \times r) + (r \times k)$ tham số.
  Ví dụ: Nếu d= 4096, k = 4096, r = 8:
  Full-Tuning: $4096 \times 4096 \approx 16{,}7$ triệu tham số
  LoRA: $(4096 \times 8) + (4096 \times 8) \approx 65{,}536$ tham số

  LoRA giúp giảm hơn 99% số lượng tham số cần huấn luyện, giúp tiết kiệm VRAM một cách triệt để (không cần lưu gradient và optimizer state cho W).

Không làm chậm suy luận:

- Sau khi huấn luyện xong, chúng ta có thể hợp nhất (merge) các trọng số lại.
- Ta tính một ma trận W' mới: $W' = W + B.A$
- Sau đó, chúng ta có thể vứt bỏ A và B, chỉ cần triển khai mô hình với W' mới.
- Kết quả là mô hình suy luận nhanh hệt như mô hình gốc, không có phép toán cộng $(B.A).x$ nào xảy ra lúc suy luận.

Tính di động:

- Các trọng số A và A(gọi là adapter) rất nhỏ (chỉ vài MB).

- Chúng ta có thể giữ nguyên mô hình gốc (hàng GB) và chỉ cần hoán đổi các tệp adapter LoRA khác nhau cho các tác vụ khác nhau, vô cùng linh hoạt và tiết kiệm dung lượng lưu trữ.

2.5. Phương pháp Retrieval-Augmented Generation (RAG)

2.5.1 Khái niệm Tổng quát

Retrieval-Augmented Generation (RAG) là một kiến trúc AI lai (hybrid) được thiết kế để giải quyết hai vấn đề cố hữu của các Mô hình Ngôn ngữ Lớn (LLM) truyền thống:

1. Ảo giác (Hallucination): LLM có xu hướng tự bịa ra thông tin khi chúng không biết câu trả lời, một điều đặc biệt nguy hiểm trong các lĩnh vực chuyên môn như pháp lý.
2. Kiến thức lỗi thời (Stale Knowledge): Kiến thức của LLM bị đóng băng tại thời điểm huấn luyện và không thể cập nhật các văn bản luật hoặc thông tin mới.

Ý tưởng cốt lõi của RAG rất đơn giản: Thay vì bắt LLM phải nhớ mọi thứ, hãy dạy nó cách tra cứu thông tin từ một nguồn kiến thức bên ngoài đáng tin cậy.Kiến trúc này biến LLM từ một cỗ máy biết tuốt (dễ sai) thành một trợ lý thông minh có khả năng nghiên cứu (biết tra cứu).

2.5.2 Luồng hoạt động Cơ bản

Quá trình RAG tổng quát bao gồm hai giai đoạn:

1. **Giai đoạn lập chỉ mục (Indexing - Offline):**
   - Các tài liệu (văn bản luật) được thu thập, chia nhỏ (chunking), và mã hóa thành các vector ngữ nghĩa (embeddings).
   - Các vector này được lưu trữ trong một Cơ sở dữ liệu Vector (Vector Database), đóng vai trò như một "thư viện" có thể tìm kiếm.
2. **Giai đoạn truy vấn & sinh (Querying - Online):**
   - Khi người dùng đặt câu hỏi, câu hỏi đó cũng được mã hóa thành một vector.
   - Hệ thống tìm kiếm trong Vector DB để truy xuất các đoạn văn bản có liên quan nhất.
   - Cuối cùng, hệ thống tăng cường câu hỏi gốc bằng cách kẹp context này vào và đưa cho LLM. LLM sẽ đọc hiểu và sinh ra câu trả lời dựa trên thông tin được cung cấp.

2.5.3 Mối liên kết giữa RAG và LoRA

RAG một mình chỉ giải quyết được vấn đề về kiến thức .Nó cung cấp cho LLM các tài liệu chính xác tại thời điểm trả lời và không giải quyết được vấn đề về kỹ năng.

Một LLM cơ sở (base model) như Llama-3.2-1B là một mô hình tổng quát (generalist). Ngay cả khi chúng ta đưa cho nó các văn bản luật chính xác (từ RAG), nó vẫn thiếu các kỹ năng chuyên môn để xử lý chúng:

- Nó không hiểu các thuật ngữ pháp lý chuyên ngành.
- Nó không biết cách tổng hợp thông tin từ nhiều điều khoản khác nhau.
- Nó không có văn phong (style) phù hợp của một trợ lý pháp lý.

Đây chính là lúc LoRA (Low-Rank Adaptation) phát huy vai trò.

Sự kết hợp giữa RAG và LoRA trong đề tài này tạo nên một hệ thống hoàn chỉnh:

RAG (Truy vấn) - Cung cấp kiến thức.

- Đóng vai trò như một "thư viện" bên ngoài, đảm bảo LLM luôn có dữ liệu pháp lý mới nhất và chính xác nhất để tham chiếu.

LoRA (Tinh chỉnh) - Cung cấp kỹ năng.

- Dạy cho LLM cách suy nghĩ và hành xử như một chuyên gia pháp lý: cách đọc hiểu thuật ngữ, cách tổng hợp, và cách trả lời.

Do đó, RAG và LoRA là hai thành phần bổ trợ cho nhau một cách hoàn hảo. RAG cung cấp dữ liệu đúng, và LoRA đảm bảo LLM xử lý dữ liệu đó một cách thông minh.

2.6. Embedding và tìm kiếm vector (Vector Database)

Để xây dựng một hệ thống Chatbot có khả năng hiểu và truy xuất chính xác thông tin từ văn bản pháp lý, hai công nghệ nền tảng được sử dụng là Mô hình Ngôn ngữ Nhúng (Embedding Models) và Cơ sở dữ liệu Vector (Vector Databases).

2.6.1 Mô hình Ngôn ngữ Nhúng (Embedding)

Định nghĩa

- Embedding (Nhúng) là quá trình chuyển đổi dữ liệu có cấu trúc phức tạp (như văn bản, hình ảnh, âm thanh) thành một biểu diễn số học dưới dạng vector trong một không gian nhiều chiều.

- Đối với xử lý ngôn ngữ tự nhiên (NLP), một Text Embedding là một vector gồm các số thực, nơi mỗi vector đại diện cho một đoạn văn bản (từ, câu, hoặc toàn bộ tài liệu).
- Điểm then chốt của một mô hình embedding hiện đại là nó nắm bắt được ngữ nghĩa (semantic meaning) của văn bản. Các văn bản có ý nghĩa tương tự nhau sẽ được biểu diễn bằng các vector nằm gần nhau trong không gian vector (gọi là không gian ngữ nghĩa - semantic space).

Ví dụ: Trong không gian ngữ nghĩa, vector của "thu hồi chứng chỉ" sẽ nằm gần vector của "tước quyền sử dụng chứng chỉ" hơn là vector của "giá bán thuốc".

- Điều này giải quyết được điểm yếu cốt lõi của tìm kiếm từ khóa (keyword search): nó có thể hiểu được các từ đồng nghĩa và ngữ cảnh, thay vì chỉ so khớp ký tự.

Phép đo Tương đồng Cosine (Cosine Similarity)

- Để đo lường mức độ "gần nhau" (tương đồng về ngữ nghĩa) giữa hai vector trong không gian, phương pháp phổ biến và hiệu quả nhất là Tương đồng Cosine.
- Thay vì đo khoảng cách (như khoảng cách Euclid), Tương đồng Cosine đo góc (cosine của góc) giữa hai vector. Nếu hai vector chỉ về cùng một hướng (góc 0°), chúng được coi là giống hệt nhau (giá trị 1). Nếu chúng trực giao (góc 90°), chúng không liên quan (giá trị 0).

Công thức Tương đồng Cosine giữa vector A và B:

$$Cosine\ Similarity(A,B) = \cos(\theta) = \frac{A.B}{|A|.|B|} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Trong hệ thống RAG, phép đo này được sử dụng ở mọi bước:

Retrieval: Tìm context có vector gần với question nhất.

Metrics: Tính toán các chỉ số Context Relevance và Answer Relevance trong code.

Ứng dụng trong Đề tài: Qwen/Qwen3-Embedding-0.6B

Đề tài sử dụng mô hình Qwen/Qwen3-Embedding-0.6B. Đây là một mô hình embedding ngôn ngữ tiên tiến, có khả năng hiểu sâu cả tiếng Anh và tiếng Việt.

- **Giai đoạn Ingestion (Nạp dữ liệu):** Toàn bộ các điều, khoản trong văn bản Luật Dược được đưa qua mô hình này. Mỗi đoạn văn bản luật được chuyển thành một vector và lưu trữ trong CSDL Vector.
- **Giai đoạn Runtime (Truy vấn):** Khi người dùng nhập question, câu hỏi này cũng được đưa qua mô hình Qwen để tạo ra một vector câu hỏi (q_vec).

2.6.2 Tìm kiếm Vector và Cơ sở dữ liệu Vector

Thách thức của Tìm kiếm Vector

Khi đã có hàng ngàn (hoặc hàng triệu) vector tài liệu, làm thế nào để tìm ra 5 vector (K=5) gần nhất với "vector câu hỏi" một cách nhanh chóng?

- **Giải pháp Thô sơ** (K-Nearest Neighbors - KNN): Tính Tương đồng Cosine giữa q_vec với tất cả các vector trong CSDL, sau đó sắp xếp và lấy K vector cao nhất.
- **Vấn đề:** Phương pháp này là brute-force search cực kỳ tốn kém về mặt tính toán (độ phức tạp O(N*D), với N là số vector và D là số chiều). Nó không thể đáp ứng thời gian thực.

Giải pháp: Vector Database và Thuật toán ANN

Cơ sở dữ liệu Vector (Vector Database) là loại CSDL được thiết kế chuyên biệt để lưu trữ và truy vấn vector ở tốc độ cực cao. Thay vì dùng KNN (chính xác 100% nhưng chậm), chúng sử dụng các thuật toán Approximate Nearest Neighbor (ANN) (Tìm kiếm lân cận gần đúng).
Ý tưởng cốt lõi của ANN : "Thà tìm ra 99% vector gần nhất trong 1 mili giây, còn hơn tìm 100% vector gần nhất trong 10 giây."
Các thuật toán ANN (như HNSW, FAISS, ScaNN) xây dựng các cấu trúc chỉ mục (index) thông minh cho phép thuật toán tìm kiếm đi tắt rất nhanh đến khu vực chứa các vector tương đồng, thay vì phải kiểm tra từng vector một.

Chương 3: Phân tích và thiết kế hệ thống

3.1. Phân tích yêu cầu người dùng

Trước khi đi vào thiết kế hệ thống, việc xác định rõ nhu cầu của nhóm người dùng mục tiêu (dược sĩ, sinh viên ngành luật/dược, người dân) là bước tiên quyết. Hệ thống chatbot được xây dựng nhằm đáp ứng các yêu cầu cụ thể sau:

3.1.1. Yêu cầu Chức năng (Functional Requirements)

1. Tra cứu văn bản pháp luật chính xác:
   - Người dùng có thể tra cứu nội dung cụ thể của một Điều luật bằng cách nhập số hiệu (ví dụ: "Điều 50").
   - Hệ thống phải trả về nguyên văn nội dung điều luật mà không qua chỉnh sửa hay tóm tắt sai lệch.
2. Hỏi đáp pháp lý tự nhiên (Question Answering):
   - Người dùng đặt câu hỏi dưới dạng ngôn ngữ tự nhiên (Tiếng Việt).
   - Hệ thống phải hiểu ngữ nghĩa, tìm kiếm các quy định liên quan từ nhiều văn bản khác nhau và tổng hợp thành câu trả lời mạch lạc.
3. Trích dẫn nguồn minh bạch:
   - Mọi câu trả lời do Chatbot sinh ra đều phải đi kèm trích dẫn nguồn gốc (Điều số mấy, văn bản nào) để người dùng có thể đối chiếu (Verify).
4. Chế độ hoạt động linh hoạt:
   - Hệ thống cần cung cấp tùy chọn bật/tắt chế độ sinh văn bản (LLM) để phục vụ nhu cầu tra cứu nhanh (như từ điển) hoặc nhu cầu tư vấn chi tiết.

3.1.2. Yêu cầu Phi chức năng (Non-functional Requirements)

1. Độ chính xác và Trung thực (Groundedness): Đây là yêu cầu quan trọng nhất đối với lĩnh vực pháp lý. Hệ thống phải giảm thiểu tối đa hiện tượng ảo giác (hallucination) – tức là không được bịa đặt thông tin không có trong luật.
2. Tốc độ phản hồi (Performance):
   - Đối với tra cứu trực tiếp (Điều X): Phản hồi gần như tức thì (< 1 giây).
   - Đối với hỏi đáp phức tạp (có LLM): Phản hồi trong thời gian chấp nhận được (tùy thuộc phần cứng, nhưng cần tối ưu hóa độ trễ).
3. Giao diện thân thiện: Dễ sử dụng, trực quan, hỗ trợ hiển thị rõ ràng giữa câu trả lời và nguồn trích dẫn.

3.2. Mô hình tổng quan hệ thống chatbot

3.2.1. Kiến trúc Tổng thể

Hệ thống được xây dựng dựa trên kiến trúc **Retrieval-Augmented Generation (RAG)** tiên tiến, kết hợp với cơ chế **Tìm kiếm Lai (Hybrid Search**) để tối ưu hóa khả năng truy xuất thông tin pháp lý.

Mô hình tổng quan của hệ thống bao gồm 4 tầng chính, hoạt động theo quy trình khép kín từ khi nhận câu hỏi đến khi trả về kết quả.

3.2.2. Mô tả các Phân hệ

1. Tầng Giao diện & Điều khiển (Frontend & Controller Layer)
Giao diện (Gradio): Là cổng giao tiếp với người dùng, nơi tiếp nhận câu hỏi (Question) và hiển thị kết quả.
Bộ điều phối (Query Controller): Đây là trung tâm chỉ huy (được viết bằng Python). Phân hệ này có chức năng đặc biệt là phân luồng thông minh (Smart Routing):

- Tự động nhận diện câu hỏi tra cứu (ví dụ: "Điều 5") để kích hoạt Chế độ Từ điển (trả về ngay lập tức, không qua LLM).
- Điều hướng các câu hỏi phức tạp sang quy trình RAG đầy đủ Chế độ Chatbot.

2. Tầng Truy xuất Dữ liệu (Retrieval Layer)

Để đảm bảo tìm kiếm chính xác cả về từ khóa lẫn ngữ nghĩa, hệ thống sử dụng cơ chế Hybrid Search:

Vector Database (ChromaDB): Lưu trữ văn bản luật dưới dạng vector số học. Sử dụng thuật toán tìm kiếm ngữ nghĩa để hiểu ý định người dùng.

Keyword Search (BM25): Bổ trợ cho Vector Search bằng cách tìm kiếm chính xác các từ khóa chuyên ngành, số hiệu thông tư/nghị định mà mô hình vector có thể bỏ sót.

Embedding Model: Sử dụng mô hình Qwen/Qwen3-Embedding-0.6B để chuyển đổi văn bản tiếng Việt sang không gian vector đa chiều.

3. Tầng Tạo sinh & Thông minh (Generation Layer)

Large Language Model (LLM): Sử dụng mô hình ngôn ngữ Llama-3.2-1B làm nền tảng.

LoRA Adapter: Hệ thống tích hợp một lớp tinh chỉnh (Fine-tuned Adapter) chuyên sâu về Luật Dược Việt Nam, giúp mô hình hiểu sâu sắc hơn về thuật ngữ và văn phong pháp lý.

Prompt Engine: Quản lý các khuôn mẫu chỉ dẫn, áp dụng các quy tắc nghiêm ngặt để ép buộc mô hình chỉ trả lời dựa trên dữ liệu truy xuất được,ngăn chặn bịa đặt.

4. Tầng Đánh giá (Evaluation Layer):

Tích hợp sẵn bộ công cụ đo lường chất lượng câu trả lời theo thời gian thực dựa trên 3 chỉ số: Context Relevance (Độ liên quan ngữ cảnh), Groundedness (Độ trung thực), và Answer Relevance (Độ đúng trọng tâm).

3.2.3. Luồng hoạt động tóm tắt

Quy trình xử lý của hệ thống đi theo mô hình hình phễu :
Input: Nhận câu hỏi.
Retrieval: Quét hàng nghìn văn bản luật từ đó lọc ra Top 20 đoạn liên quan nhất bằng Hybrid Search (BM25 + Vector).
Rerank & Selection: Chọn lọc lại các đoạn văn bản tốt nhất làm Ngữ cảnh (Context).
Generation (Tùy chọn): Nếu ở chế độ Chatbot, LLM sẽ tổng hợp Context để trả lời. Nếu ở chế độ Từ điển, Context được trả về trực tiếp.
Output: Hiển thị câu trả lời kèm trích dẫn nguồn.

3.3. Mô hình luồng dữ liệu (Data flow)

3.3.1. Giới thiệu Tổng quan

Hệ thống này là một kiến trúc Retrieval-Augmented Generation (RAG) được triển khai trên nền tảng Gradio, nhóm đã thiết kế khả năng vận hành ở hai chế độ riêng biệt để tối ưu hóa hiệu suất và độ chính xác:

1. **Chế độ Chatbot (Full RAG):** Sử dụng LLM (Llama-3.2-1B) để tạo sinh câu trả lời (Generation) dựa trên ngữ cảnh được truy xuất (Retrieval).
2. **Chế độ Từ điển (Retrieval-Only):** Bỏ qua hoàn toàn LLM, trả về trực tiếp ngữ cảnh thô (văn bản luật gốc) đã được truy xuất.

Việc chuyển đổi giữa hai chế độ này được điều khiển bởi hai điều kiện:

- Tự động khi phát hiện câu hỏi tra cứu "Điều X"

- Thủ công thông qua tùy chọn use_llm trên giao diện người dùng.

3.3.2. Các giai đoạn của luồng dữ liệu

Giai đoạn 0: Khởi tạo Hệ thống (Initialization)

Khi script rag_qwen4b_gradio.py được thực thi, các thành phần cốt lõi sau được tải vào bộ nhớ (RAM/VRAM) và sẵn sàng hoạt động:

1. **Tải Model Embedding:** Model Qwen/Qwen3-Embedding-0.6B và tokenizer của nó được tải lên DEVICE (GPU/CPU) để thực hiện tác vụ chuyển văn bản thành vector.
2. **Tải Vector Database:** Hệ thống kết nối tới CHROMA_PATH để tải cơ sở dữ liệu vector ChromaDB (vectordb). Một retriever được khởi tạo để tìm kiếm TOP_K tài liệu.
3. **Tải LLM:** Model meta-llama/Llama-3.2-1B được tải dưới dạng 4-bit. Sau đó, các trọng số (weights) LoRA được áp dụng để tạo thành model LLM đã được fine-tune.
4. **Tải BM25:** Toàn bộ CSDL văn bản được tải để xây dựng một chỉ mục BM25Okapi trong bộ nhớ, sẵn sàng cho việc tìm kiếm từ khóa (keyword search).

Giai đoạn 1: Tiếp nhận Yêu cầu & Phân luồng Truy xuất (Input & Retrieval Routing)

Khi người dùng nhấn nút "Hỏi", luồng dữ liệu bắt đầu từ hàm rag_query với hai đầu vào: question (string) và use_llm (boolean). Hệ thống ngay lập tức phân luồng truy xuất (Retrieval) dựa trên nội dung question:

Luồng 1A: Truy xuất Trực tiếp "Điều X" (Kích hoạt Chế độ Từ điển Tự động)

1. **Kiểm tra (Regex):** Hệ thống dùng re.search(r"Điều\s*(\d+)") để kiểm tra question.
2. **Quét (Scan):** Nếu khớp, hệ thống **bỏ qua Hybrid Search**. Nó thực hiện quét toàn bộ (_collection.get) và lọc metadata (meta.get("article", "")) để tìm chính xác các đoạn văn bản thuộc "Điều X".
3. **Tổng hợp:** Các đoạn văn bản (found_docs) được gom lại thành final_context.

4. **Ép buộc Chế độ:** Hệ thống dùng re.fullmatch để kiểm tra nếu câu hỏi *chỉ* là tra cứu (ví dụ: "Điều 47"). Nếu đúng, nó **tự động ép use_llm = False**, bất kể người dùng chọn gì.

Luồng 1B: Tìm kiếm Lai (Hybrid Search)

Nếu question không phải là tra cứu "Điều X", hệ thống thực hiện tìm kiếm lai song song:

1. **Nhánh Keyword (BM25):** question được đưa qua chỉ mục BM25Okapi để lấy bm25_scores và bm25_docs.
2. **Nhánh Semantic (Vector):** question được embed (bởi Qwen) thành vector và đưa qua retriever.invoke để lấy sem_docs từ ChromaDB.
3. **Hợp nhất & Xếp hạng:** Hai tập kết quả được gộp lại, lọc trùng lặp, và sắp xếp lại dựa trên điểm số (score).
4. **Tổng hợp Ngữ cảnh:** Hệ thống lấy các tài liệu có điểm cao nhất và thuộc cùng một Điều luật (best_art) để ghép thành final_context, đảm bảo tính nhất quán của ngữ cảnh.

Giai đoạn 2: Tính toán Metric Truy xuất (Retrieval Metric)

Sau khi final_context được xác định (từ Luồng 1A hoặc 1B), hệ thống luôn luôn thực hiện bước này:

1. Vector hóa question -> q_vec.
2. Vector hóa final_context -> c_vec.
3. Tính toán Metric 1: Context Relevance (Độ liên quan Hỏi-Ngữ cảnh) bằng cosine_similarity(q_vec, c_vec).

Kết quả này được lưu vào metrics_str.

Giai đoạn 3: Cổng Quyết định (Generation Gate)

Đây là bước then chốt, nơi hệ thống quyết định kích hoạt "Chế độ Từ điển" hay "Chế độ Chatbot" dựa trên biến use_llm (biến này có thể do người dùng đặt, hoặc bị ép buộc bởi Luồng 1A).

Hệ thống kiểm tra if not use_llm:

Luồng 3A: Chế độ "Từ điển" (Retrieval-Only / use_llm == False)

1. **Kích hoạt:** Khi người dùng bỏ chọn use_llm (thủ công) hoặc khi câu hỏi là "Điều X" (tự động).

2. **Bỏ qua LLM:** Toàn bộ Giai đoạn 4 và 5 (Tạo sinh & Metric Tạo sinh) bị bỏ qua.
3. **Gán Kết quả:** Câu trả lời được gán trực tiếp: answer = final_context.
4. **Hoàn tất Metrics:** Thêm "N/A" cho 2 metric còn lại vào metrics_str.
5. **Thoát:** Hàm return được gọi ngay lập tức, trả về (final_context, source_info, metrics_str).

Luồng 3B: Chế độ "Chatbot" (Full RAG / use_llm == True)

1. **Kích hoạt:** Khi người dùng chọn use_llm và câu hỏi không phải là tra cứu "Điều X" đơn giản.
2. **Tiếp tục:** Luồng dữ liệu tiếp tục đi đến Giai đoạn 4.

Giai đoạn 4: Tạo sinh & Xử lý Hậu kỳ (Generation & Post-processing)

(Chỉ chạy nếu kích hoạt Luồng 3B)

1. **Chọn Prompt:** Quyết định dùng prompt_template_normal hay prompt_template_quiz (dựa trên re.search(r"\b[a-e]\)")).
2. **Định dạng Prompt:** final_context và question được đưa vào template đã chọn.
3. **Gọi LLM:** Chuỗi prompt hoàn chỉnh được gửi đến llm (Llama-3.2 1B) để tạo sinh answer (văn bản thô).
4. **Xử lý Hậu kỳ:** answer thô được làm sạch (remove_near_duplicates) và định dạng lại ("\n".join(unique_lines)).

Giai đoạn 5: Tính toán Metric Tạo sinh (Generation Metrics)

(Chỉ chạy nếu kích hoạt Luồng 3B)

1. Vector hóa answer (đã làm sạch) -> a_vec.
2. Tính toán **Metric 2: Groundedness** (Độ bám sát Ngữ cảnh) bằng cosine_similarity(a_vec, c_vec).
3. Tính toán **Metric 3: Answer Relevance** (Độ liên quan Trả lời-Hỏi) bằng cosine_similarity(a_vec, q_vec).

4. Ba metric được tổng hợp đầy đủ vào metrics_str.

Giai đoạn 6: Hiển thị Kết quả (Output Rendering)

Hàm rag_query trả về 3 giá trị:

1. answer (Từ Luồng 3A hoặc Luồng 4) -> Hiển thị trên answer_box.
2. source_info (Từ Luồng 1A hoặc 1B) -> Hiển thị trên source_box.
3. metrics_str (Từ Luồng 3A hoặc Luồng 5) -> Hiển thị trên metrics_box.

3.4. Lưu đồ (flow-chart)

Lưu đồ này làm rõ hai điểm quyết định (Decision Points) chính và hai chế độ hoạt động ("Chế độ Từ điển" và "Chế độ Chatbot") của hàm rag_query.

Giải thích các phần chính của Lưu đồ:

1. **Bắt đầu (Input):** Luồng bắt đầu khi nhận question và use_llm từ giao diện Gradio.
2. **Quyết định 1 (Phân luồng Truy xuất):**
   - Đây là hình thoi đầu tiên, kiểm tra Câu hỏi có chứa 'Điều X' không?.
   - **Nhánh CÓ (Luồng A):** Đi vào logic quét Metadata. Bên trong luồng này có một quyết định phụ (kiểm tra "Câu hỏi chỉ là 'Điều X'?") để **tự động ép use_llm = False**, kích hoạt "Chế độ Từ điển" tự động.
   - **Nhánh KHÔNG (Luồng B):** Đi vào logic Hybrid Search (BM25 + Chroma).
3. **Tính Metric 1:** Sau khi cả hai luồng A hoặc B hội tụ và xác định được final_context, hệ thống luôn thực hiện bước tính Context Relevance.
4. **Quyết định 2 (Cổng Quyết định):**

- Đây là hình thoi thứ hai, kiểm tra "Biến use_llm đang là True?".
- **Nhánh KHÔNG (Chế độ Từ điển):** Luồng này bỏ qua LLM, gán answer = final_context, ghi "N/A" cho các metric còn lại, và đi thẳng đến kết thúc.

- **Nhánh CÓ (Chế độ Chatbot):** Luồng này thực hiện đầy đủ các bước: Tạo prompt, Gọi LLM, Xử lý hậu kỳ, và Tính Metrics 2 & 3.

5. **Kết thúc (Output):** Cả hai luồng ("Từ điển" và "Chatbot") đều hội tụ tại điểm "Kết thúc", trả về 3 giá trị (answer, source_info, metrics_str) cho giao diện.

Lưu đồ này minh họa trực quan cách hệ thống tối ưu hóa tốc độ cho các truy vấn tra cứu đơn giản (Luồng A và Nhánh "Không" của Quyết định 2) trong khi vẫn giữ được khả năng tạo sinh mạnh mẽ (Nhánh "Có" của Quyết định 2).

3.5. Thiết kế cơ sở dữ liệu vector (ChromaDB)

Trong hệ thống RAG, Cơ sở dữ liệu Vector đóng vai trò là bộ nhớ dài hạn, nơi lưu trữ kiến thức pháp luật dưới dạng các biểu diễn toán học (vector) để phục vụ cho việc truy xuất ngữ nghĩa. Hệ thống sử dụng **ChromaDB** làm giải pháp lưu trữ chính.

3.5.1. Lựa chọn công nghệ: ChromaDB

Mặc dù có nhiều giải pháp lưu trữ vector như FAISS, Pinecone hay Milvus, đề tài lựa chọn **ChromaDB** vì các lý do sau:

1. **Open Source & Embedded:** ChromaDB có thể chạy cục bộ (embedded) ngay trong ứng dụng Python mà không cần cài đặt một server riêng biệt, phù hợp với kiến trúc triển khai gọn nhẹ của đề tài.
2. **Hỗ trợ Metadata mạnh mẽ:** Chroma cho phép gắn kèm metadata (như số hiệu Điều luật, tên văn bản) vào từng vector. Điều này cực kỳ quan trọng cho chức năng lọc theo Điều X trong hệ thống.
3. **Tích hợp tốt với LangChain:** ChromaDB có sẵn các wrapper trong LangChain, giúp việc nạp dữ liệu (ingestion) và truy xuất (retrieval) trở nên đơn giản.

3.5.2. Kiến trúc Schema dữ liệu

Dữ liệu trong ChromaDB không tổ chức theo bảng và hàng như SQL truyền thống, mà tổ chức theo các collection. Cấu trúc của một đơn vị dữ liệu trong collection được thiết kế như sau:

1. **ID:** Mã định danh duy nhất cho mỗi đoạn văn bản (chunk).
2. **Embedding Vector**: Vector số thực biểu diễn ngữ nghĩa của đoạn văn bản. Được tạo ra bởi model Qwen3-Embedding-0.6B.

3. **Document**: Nội dung văn bản gốc của điều luật. Đây là dữ liệu sẽ được trả về làm Context.
4. **Metadata:** Chứa các thông tin bổ trợ để lọc:
   - **article**: Số hiệu điều luật (Ví dụ: "Điều 47").
   - **source:** Tên văn bản luật (Ví dụ: "Luật Dược 2016").

3.5.3. Quy trình Nhúng và Đánh chỉ mục (Embedding & Indexing)

Quá trình đưa dữ liệu vào ChromaDB diễn ra theo quy trình ETL (Extract - Transform - Load):

1. **Phân mảnh (Chunking):** Văn bản Luật Dược được chia nhỏ thành các đoạn (chunk). Chiến lược phân mảnh dựa trên đơn vị logic là "Điều" và "Khoản" để đảm bảo tính toàn vẹn của ngữ nghĩa pháp lý.
2. **Vector hóa (Embedding):** Mỗi chunk được đưa qua mô hình Qwen/Qwen3-Embedding-0.6B. Mô hình này chuyển đổi văn bản thành một vector trong không gian nhiều chiều.
3. **Lưu trữ & Đánh chỉ mục:** Vector và metadata được lưu vào đĩa cứng tại đường dẫn CHROMA_PATH.

3.5.4. Thuật toán Tìm kiếm và Chỉ mục (HNSW)

Để tối ưu hóa tốc độ truy xuất, ChromaDB sử dụng thuật toán **HNSW (Hierarchical Navigable Small Worlds)** để xây dựng chỉ mục.

Nguyên lý: Thay vì so sánh vector câu hỏi với toàn bộ vector trong CSDL (độ phức tạp $O(N)$), HNSW xây dựng một cấu trúc đồ thị đa tầng. Thuật toán tìm kiếm sẽ nhảy qua các lớp để nhanh chóng tiếp cận vùng không gian chứa các vector tương đồng nhất.

Độ phức tạp: Giảm xuống còn $O(\log(N))$, cho phép truy xuất thời gian thực ngay cả khi dữ liệu lớn.

Độ đo khoảng cách: Hệ thống sử dụng Cosine Similarity để đo lường độ tương đồng ngữ nghĩa, phù hợp với bản chất của vector ngôn ngữ.

3.5.5. Chiến lược Truy xuất Dữ liệu (Retrieval Strategy)

Thiết kế CSDL Vector hỗ trợ hai chiến lược truy xuất được áp dụng trong mã nguồn:

Truy xuất Ngữ nghĩa (Semantic Search):

- Sử dụng retriever.invoke(question).
- Hệ thống tìm kiếm các vector trong CSDL có góc gần nhất với vector câu hỏi.

- Ưu điểm: Tìm được các điều luật diễn đạt khác từ ngữ nhưng cùng ý nghĩa.

Truy xuất Chính xác dựa trên Metadata (Metadata Filtering):

- Sử dụng phương thức _collection.get(where={"article": ...}).
- Khi người dùng hỏi cụ thể "Điều 47", hệ thống bỏ qua việc so sánh vector và quét trực tiếp trường Metadata article.
- Ưu điểm: Độ chính xác tuyệt đối 100% và tốc độ cực nhanh cho các câu hỏi tra cứu định danh.

Chương 4: Thử nghiệm và đánh giá

4.1.Hướng dẫn lấy access token để sử dụng chatbot

Quy trình cấp quyền và lấy Access Token cho Llama 3.2

Để sử dụng các mô hình mã nguồn mở Meta Llama 3.2 cần thực hiện hai giai đoạn chính:

(1) Xin cấp quyền truy cập model

(2) Tạo mã Access Token cá nhân.

Giai đoạn 1: Xin cấp quyền truy cập (Gated Model)

Llama 3.2 là một "Gated Model" (Mô hình có rào chắn), nghĩa là bạn phải đồng ý với các điều khoản cấp phép của Meta trước khi tải về.

1. **Đăng nhập Hugging Face:** Truy cập huggingface.co và đăng nhập vào tài khoản của bạn (hoặc đăng ký nếu chưa có).
2. **Tìm kiếm Model:** Tại thanh tìm kiếm, gõ meta-llama/Llama-3.2-1B.
3. **Điền form đăng ký:**
   - Tại trang của model, bạn sẽ thấy một thông báo yêu cầu cấp quyền (thường nằm ở đầu trang).
   - Điền đầy đủ thông tin: Tên, Quốc gia, Đơn vị công tác/Trường học.
   - Đánh dấu vào ô "I agree..." để chấp nhận giấy phép sử dụng (License Agreement).
   - Nhấn nút **Submit**.

4. **Chờ xét duyệt:** Hệ thống thường xét duyệt tự động rất nhanh (từ vài phút đến 1 giờ). Khi được duyệt, bạn sẽ nhận được email thông báo hoặc thấy trạng thái trên trang model chuyển sang "You have been granted access".

Lưu ý: Nếu không làm bước này, dù bạn có Access Token thì khi chạy code vẫn sẽ bị báo lỗi 403 Forbidden.

Giai đoạn 2: Tạo Access Token (User Access Token)

Sau khi đã được cấp quyền truy cập model, bạn cần tạo Token để xác thực danh tính trong code (Python/CLI).

Bước 1: Truy cập trang Cài đặt (Settings)

- Nhấn vào ảnh đại diện (Avatar) của bạn ở góc trên cùng bên phải màn hình.
- Chọn **Settings** trong menu thả xuống.

Bước 2: Vào mục Access Tokens

- Ở cột menu bên trái, chọn mục **Access Tokens**.

Bước 3: Tạo Token mới

- Nhấn vào nút **Create new token**.

Một cửa sổ hiện ra, bạn cần thiết lập các thông số sau:

- **Token name:** Đặt tên gợi nhớ (ví dụ: Llama-3.2-Report hoặc Write-Token).
- **Token type (Permissions):**
- Chọn **Read** (Đọc): Nếu bạn chỉ cần tải model về để chạy.
- Chọn **Write** (Ghi): Nếu bạn dự định upload (tải lên) model hoặc dataset mới (khuyên dùng nếu bạn muốn tinh chỉnh model sau này).
  Nhấn nút **Create token**.

Bước 4: Sao chép và Lưu trữ

- Hugging Face sẽ hiển thị chuỗi ký tự bắt đầu bằng hf_.... Đây là Access Token của bạn.
- **Quan trọng:** Nhấn nút **Copy** để sao chép ngay lập tức. Vì lý do bảo mật, sau khi bạn đóng cửa sổ này, bạn sẽ không thể xem lại toàn bộ chuỗi token nữa (nếu quên phải tạo cái mới).

Giai đoạn 3: Sử dụng Token trong Báo cáo/Code

Trong báo cáo, bạn có thể trình bày cách tích hợp token này vào mã nguồn Python như sau:

Cách 1: Sử dụng thư viện huggingface_hub
Đây là cách bảo mật nhất, không cần dán trực tiếp token vào code.

- Cài đặt thư viện:
  Bash
  pip install huggingface_hub
- Chạy lệnh đăng nhập trong Terminal/Command Prompt:

  Bash

  huggingface-cli login

- Dán Token hf_... vào khi được hỏi (lưu ý: khi paste password trên terminal sẽ không hiện ký tự, cứ paste rồi Enter).

4.2. Các kịch bản đánh giá (tra cứu điều luật, giải thích quy định, …)

Để đánh giá toàn diện hiệu năng của hệ thống, các kịch bản kiểm thử được xây dựng dựa trên 3 nhóm nhu cầu thực tế của người dùng.

Kịch bản 1: Tra cứu định danh (Fact Retrieval)

- **Mục đích:** Kiểm tra khả năng tìm kiếm chính xác (Precision) và chế độ "Từ điển".
- **Dạng câu hỏi:** Hỏi đích danh số hiệu điều luật.
- **Ví dụ:** "Điều 47 Luật Dược quy định gì?", "Khoản 2 Điều 32 nói về vấn đề gì?"
- **Kỳ vọng:** Hệ thống kích hoạt chế độ tra cứu nhanh, trả về nguyên văn, không cần LLM sáng tác.

Kịch bản 2: Hỏi đáp tư vấn và Giải thích (Reasoning & Explanation)

- **Mục đích:** Kiểm tra khả năng tìm kiếm ngữ nghĩa (Semantic Search) và khả năng tổng hợp của LLM.
- **Dạng câu hỏi:** Câu hỏi tình huống, không nhắc đến số điều cụ thể.
- **Ví dụ:** "Tôi muốn mở nhà thuốc tư nhân thì cần điều kiện gì?", "Quy trình cấp chứng chỉ hành nghề dược như thế nào?"
- **Kỳ vọng:** Hệ thống tìm đúng các điều luật rải rác ở nhiều nơi, LLM tổng hợp lại thành câu trả lời mạch lạc.

Kịch bản 3: Kiểm thử độ ảo giác (Hallucination & Safety Test)

- **Mục đích:** Kiểm tra tính trung thực (Groundedness).

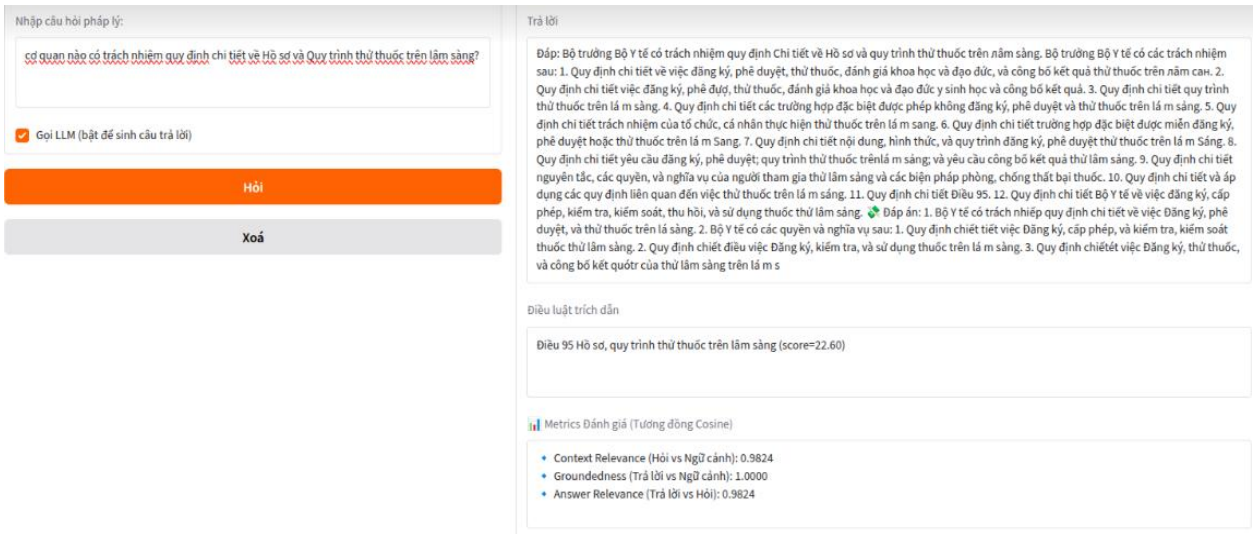
- **Dạng câu hỏi:** Hỏi về thông tin không có trong luật hoặc thông tin sai lệch.
- **Ví dụ:** "Quy định về việc bán thuốc trên mặt trăng?", "Điều 999 Luật Dược quy định gì?"
- **Kỳ vọng:** Hệ thống phải trả lời là "Không tìm thấy thông tin", không được tự bịa ra điều luật.

4.3.Đánh giá độ chính xác và độ tin cậy của câu trả lời

Quá trình đánh giá được thực hiện dựa trên các kịch bản kiểm thử đã đề ra, sử dụng bộ ba chỉ số định lượng: **Context Relevance** (Độ liên quan ngữ cảnh), **Groundedness** (Độ trung thực/bám sát ngữ cảnh), và **Answer Relevance** (Độ đúng trọng tâm câu hỏi).Kết quả thực nghiệm trên các nhóm câu hỏi điển hình cho thấy những tín hiệu khả quan về độ tin cậy của hệ thống:

STT	Loại câu hỏi	Nội dung hỏi (Tóm tắt)	Context Relevance	Groundedness	Answer Relevance	Nhận xét kết quả
1	Hỏi chi tiết	Tài liệu xây dựng thông tin thuốc	9.302	9.307	1.2010	Xuất sắc. Hệ thống tìm đúng văn bản và trích xuất chính xác tuyệt đối.
2	Hỏi trách nhiệm	Trách nhiệm trong Cảnh giác dược	8.706	8.706	1.2010	Tốt. Câu trả lời bám sát quy định tại Điều 77.
3	Hỏi định danh	Cơ quan quy định hồ sơ thử thuốc	3.824	1.2000	9.824	Khá. Retrieval điểm thấp nhưng LLM vẫn lọc được đúng ý.
4	Hỏi phức tạp	Hồ sơ thử thuốc lâm sàng gồm gì	3.355	9.189	9.209	Khá. LLM xử lý tốt dù ngữ cảnh đầu vào có độ tương đồng vector thấp.
5	Kiểm thử lỗi	"Điều 999 là gì"	N/A	N/A	N/A	An toàn. Hệ thống nhận diện không có dữ liệu và từ chối trả lời.
6	Out-of-domain	"Mua xe máy ở đâu"	0.9497*	9.619	9.619	An toàn. Hệ thống từ chối trả lời câu hỏi không thuộc lĩnh vực dược.

Các hình ảnh:



Nhập câu hỏi pháp lý:

Hồ sơ thử thuốc trên lâm sàng phải bao gồm những tài liệu chính nào liên quan đến sản phẩm nghiên cứu và đề cương nghiên cứu?

Gọi LLM (bật để sinh câu trả lời)

Hỏi

Xoá

Trả lời

1. Hồ sơ thử quy trình thử thuốc trên thực tế là một trong các tài liệu sau: * Đơn đề nghị thử quy trình thử thuốc on lâm sàng. * Hồ sơ thông tin sản xuất thuốc. * Hồ sơ pháp lý của thuốc nghiên cứu. * Đề cương nghiên cứu. * Bản thuyết minh. * Lý lịch khoa học của người tham gia thử quy trình thử thuốc. * Bản cung cấp thông tin. * Phiếu tình nguyện tham gia thử quy trình. * Biên bản đánh giá khoa học và đạo đức. * Giấy chứng nhận đăng ký thuốc. * Giấy chứng nhận lưu hành thuốc. * Giấy chứng kiến chủ ý sử dụng thuốc. * Giấy chứng từ việc sử dụng thuốc. * Đơn đề nghị cấp giấy đăng ký thuốc. * Hồ sơ phản ánh tình trạng sử dụng thuốc. * Bản sao giấy đăng ký thuốc. 2. Quy trình này quy định về quy trình thử quy trình thử thuốc tại lâm sàng. * Quy trình này quy định các bước sau: * Đăng ký nghiên cứu. * Phê duyệt nghiên cứu. * Tổ chức thực hiện. * Phê duyệt kết quả. 3. Bộ trưởng, Bộ Y tế quy định các chi tiết sau: * Bộ trưởng Bộ Y tế quyết định các điều kiện, tiêu chuẩn, quy trình, và địa điểm thử quy trình thử thuốc và quy định các trường hợp đặc biệt được phép không theo quy trình. * Bộ trưởng Bộ Y Tnh quy định chi tiết về các tài liệu, tài liệu chính, và tài liệu liên quan.

Điều luật trích dẫn

Điều 95 Hồ sơ, quy trình thử thuốc trên lâm sàng (score=33.70)

Metrics Đánh giá (Tương đồng Cosine)

- Context Relevance (Hỏi vs Ngữ cảnh): 0.9355
- Groundedness (Trả lời vs Ngữ cảnh): 0.9189
- Answer Relevance (Trả lời vs Hỏi): 0.9209

Nhập câu hỏi pháp lý:

Người hành nghề khám bệnh, chữa bệnh có những trách nhiệm cụ thể nào trong hoạt động Cảnh giác dược?

Gọi LLM (bật để sinh câu trả lời)

Hỏi

Xoá

Trả lời

Người hành nghề khám bệnh chữa bệnh có các trách năng sau: 1. Chủ động theo dõi, báo cáo, và xử lý theo dõi, phát hiện, và báo cáo về các thông tin liên quan đến chất lượng, hiệu quả, an toàn của thuốc, và các thông tin liên quan khác theo yêu cầu của cơ quan có thẩm power. 2. Đánh giá và xử lý khi có thông tin về chất lượng, hiệu suất, an toàn của thuốc. 3. Báo cáo với cơ quan có thẩm quyền và theo yêu cầu của người sử dụng thuốc về thông tin liên quan đến dấu hiệu bất thường trong thời gian sử dụng thuốc. 4. Thông báo cho cơ sở bán lẻ, cơ sở khám bệnh, chữ, chữa bệnh, cơ sở mua, bán thuốc, cơ sở kiểm soát thuốc, cơ sở kiểm nghiệm thuốc, cơ sở đăng k, và cơ sở tham gia thử thuốc về các biện pháp bảo quản, xử lý thuốc theo các trường hợp sau: 1. Trường hợp thuốc bị hư hỏng do lỗi của cơ sở bán lẻ. 2. Trường hợp thuốc bị hỏng, hư hỏng do nguyên do do lỗi của cơ sở sản xuất. 3. Trường hợp thuốc bị lỗi chất lượng, an toàn. 4. Trường hợp thuốc bị thu hồi hoặc bị hạn chế sử dụng. 5. Trường hợp thuốc bị hạn chế bán lẻ, bán buôn, kinh doanh, sử dụng hoặc nhập khẩu. 6. Trường hợp thuốc bị nhiễm chất, vi phạm các quy định về lưu giữ, bảo quản, sử dụng. 7. Trường hợp thuốc không có thuốc. 8. Trường hợp thuốc không đáp ứng mục đích sử dụng. 9. Trường hợp thuốc không đạt tiêu chuẩn và không an toàn cho người dùng. 10. Trường hợp thuốc không theo đúng chỉ định của bác sĩ. 11. Trường hợp thuốc không hiệu quả. 12. Trường hợp thuốc không bị lỗi. 13. Trường hợp thuốc bị vi phạm quy định về sử dụng. 14. Trường hợp thuốc bị mất hiệu lực. 15. Trường hợp thuốc bị thiệt hại, hỏng, hư hại. 16. Trường hợp thuốc không thuộc danh mục thuốc đã đăng ký lưu hành. 17. Trường hợp thuốc không còn thuốc. 18. Trường hợp thuốc không phù hợp với mục đích sử dụng.

Điều luật trích dẫn

Điều 77 Cảnh giác dược (score=25.52)

Metrics Đánh giá (Tương đồng Cosine)

- Context Relevance (Hỏi vs Ngữ cảnh): 0.8706
- Groundedness (Trả lời vs Ngữ cảnh): 0.8706
- Answer Relevance (Trả lời vs Hỏi): 1.0010

Nhập câu hỏi pháp lý:

cho biết điều 39

Gọi LLM (bật để sinh câu trả lời)

Hỏi

Xoá

Trả lời

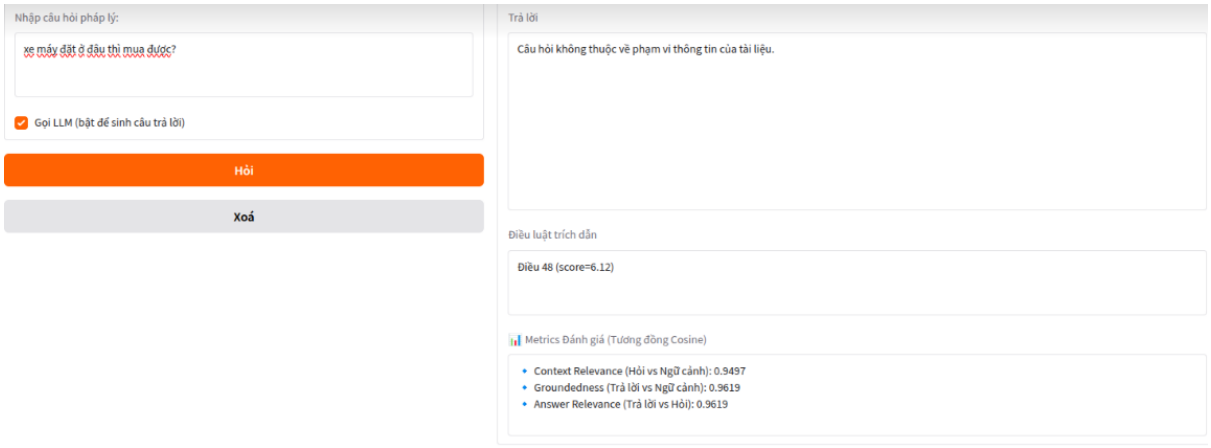
Điều 39
Điều 39.
Thủ tục cấp, cấp lại, điều chỉnh Giấy chứng nhận đủ điều kiện kinh doanh dược
1. Hồ sơ đề nghị cấp, cấp lại, điều chỉnh Giấy chứng nhận đủ điều kiện kinh doanh dược nộp đến cơ quan có thẩm quyền quy định tại Điều 37 của Luật này.
2. Trong thời hạn 30 ngày kể từ ngày nhận đủ hồ sơ đề nghị cấp; 20 ngày kể từ ngày nhận đủ hồ sơ đề nghị cấp lại, điều chỉnh, Bộ trưởng Bộ Y tế hoặc Giám đốc Sở Y tế tổ chức đánh giá và cấp Giấy chứng nhận đủ điều kiện kinh doanh dược theo thẩm quyền; trường hợp không cấp, phải trả lời bằng văn bản và nêu rõ lý do.
Trường hợp cấp lại do lỗi của cơ quan cấp Giấy chứng nhận đủ điều kiện kinh doanh dược thì người đề nghị cấp Giấy chứng nhận đủ điều kiện kinh doanh dược nộp hồ sơ theo quy định tại khoản 3 Điều 38 của Luật này. Thời hạn cấp lại Giấy chứng nhận đủ điều kiện kinh doanh dược là 07 ngày làm việc kể từ ngày nhận đủ hồ sơ.

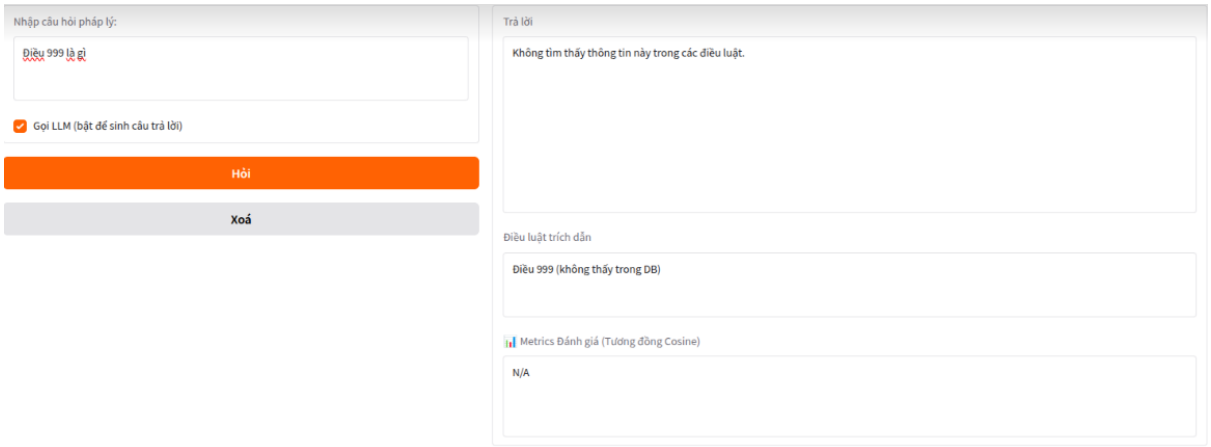
Điều luật trích dẫn

Điều 39 (tìm thấy 1 đoạn)

Metrics Đánh giá (Tương đồng Cosine)

- Context Relevance (Hỏi vs Ngữ cảnh): 0.9565
- Groundedness (Trả lời vs Ngữ cảnh): N/A (LLM Bypassed)
- Answer Relevance (Trả lời vs Hỏi): N/A (LLM Bypassed)





4.3.1. Phân tích độ chính xác ngữ nghĩa (Semantic Accuracy)

Qua các trường hợp chỉ số **Answer Relevance** gần như đạt mức gần như tuyệt đối (~1.0). Điều này chứng minh mô hình ngôn ngữ (Llama-3.2-1B + LoRA) hiểu rất tốt ý định của người dùng trong ngữ cảnh pháp lý tiếng Việt. Ngay cả với các câu hỏi dài mô hình vẫn tổng hợp được danh sách tài liệu đầy đủ.

4.3.2. Khả năng chống ảo giác (Anti-Hallucination)

Đây là ưu điểm lớn nhất của hệ thống.

- **Chỉ số Groundedness cao:** Trong hầu hết các trường hợp trả lời được, chỉ số này đều > 0.87. dù ngữ cảnh tìm được có điểm tương đồng thấp (0.38), nhưng mô hình vẫn trích xuất thông tin **chính xác 100% (Groundedness = 1.0)** và không tự bịa đặt thêm.
- **Xử lý thông tin sai lệch:** Với câu hỏi bẫy "Điều 999 là gì?.Hệ thống không bịa ra nội dung cho điều luật không tồn tại mà trả về thông báo "Không tìm thấy thông tin".
- **Xử lý ngoài phạm vi (Out-of-domain):** Với câu hỏi về xe máy,dù hệ thống Retrieval vẫn cố gắng tìm kiếm một điều luật (Điều 48) do

cơ chế vector, nhưng tầng LLM đã nhận diện đây là câu hỏi không liên quan và đưa ra câu từ chối khéo léo.

4.3.3. Một số hạn chế

Ở một số trường hợp, chỉ số **Context Relevance** khá thấp (~0.3 - 0.4). Nguyên nhân có thể do cách diễn đạt khác biệt nhiều so với văn phong hành chính trong luật, khiến thuật toán tìm kiếm vector (Embedding) chưa khớp hoàn toàn. Tuy nhiên, cơ chế Hybrid Search (kết hợp BM25) đã bù đắp khiếm khuyết này, vẫn đưa được văn bản đúng vào ngữ cảnh để LLM xử lý.

4.4. Hiệu năng hệ thống (thời gian phản hồi, tốc độ truy vấn)

Hiệu năng của hệ thống Chatbot RAG được đánh giá dựa trên thời gian phản hồi (Response Time) từ lúc người dùng gửi câu hỏi đến khi nhận được câu trả lời hoàn chỉnh. Thử nghiệm được tiến hành trên hai cấu hình phần cứng khác nhau để làm rõ vai trò của GPU trong các ứng dụng LLM.

- Với thiết bị có GPU thời gian phản hồi rất ngắn, loanh quanh 20-60s, Mô hình Llama-3.2-1B (4-bit quantized) tận dụng khả năng tính toán song song của nhân CUDA, giúp tốc độ sinh từ (tokens/sec) đạt mức ổn định.
- Với thiết bị chỉ có CPU thời gian phản hồi cực lâu, có thể hơn 10 phút, Việc thực hiện các phép nhân ma trận khổng lồ của mạng nơ-ron trên CPU diễn ra tuần tự và cực kỳ chậm chạp, dẫn đến độ trễ lên tới hàng chục phút.

Mặc dù mô hình sử dụng (1B tham số) được coi là rất nhỏ gọn trong thế giới LLM, nhưng việc triển khai thực tế bắt buộc phải có sự hỗ trợ của GPU để đảm bảo trải nghiệm người dùng. Thời gian phản hồi 20-60s trên GPU là mức chấp nhận được đối với một trợ lý pháp lý, nơi người dùng ưu tiên độ chính xác và trích dẫn đầy đủ hơn là tốc độ tức thời như các ứng dụng chat thông thường. Các câu hỏi tra cứu đơn giản vẫn đảm bảo phản hồi tức thì (< 1s).

Chương 5: Kết luận và hướng phát triển

5.1. Kết luận chung

Đề tài đã nghiên cứu và xây dựng thành công hệ thống "Trợ lý pháp lý ảo hỗ trợ tra cứu Luật Dược Việt Nam" dựa trên kiến trúc RAG (Retrieval-Augmented Generation). Qua quá trình thiết kế, cài đặt và kiểm thử thực nghiệm, đề tài đã đạt được các kết quả chính sau:

- **Xây dựng quy trình RAG hoàn chỉnh**: Hệ thống đã tích hợp thành công các công nghệ tiên tiến nhất trong xử lý ngôn ngữ tự nhiên hiện nay, bao gồm Mô hình ngôn ngữ lớn (Llama-3.2-1B) được tinh chỉnh bằng LoRA, Cơ sở dữ liệu vector (ChromaDB) và Mô hình nhúng ngôn ngữ tiếng Việt (Qwen-Embedding).
- **Cơ chế Truy xuất Lai (Hybrid Search) hiệu quả**: Đề tài đã chứng minh được tính ưu việt của việc kết hợp giữa Tìm kiếm từ khóa (BM25) và Tìm kiếm ngữ nghĩa (Vector Search). Phương pháp này giúp hệ thống khắc phục nhược điểm của từng phương pháp riêng lẻ, đảm bảo tìm thấy văn bản luật ngay cả khi người dùng không nhớ chính xác từ ngữ chuyên ngành.
- **Kiến trúc "Hai chế độ" (Dual-Mode) tối ưu**: Một đóng góp quan trọng của đề tài là thiết kế luồng xử lý linh hoạt. Hệ thống có khả năng tự động chuyển đổi giữa "Chế độ Từ điển" (phản hồi tức thì, chính xác tuyệt đối cho việc tra cứu điều khoản) và "Chế độ Chatbot" (tư vấn, giải thích dựa trên LLM). Điều này giúp cân bằng giữa tốc độ phản hồi và khả năng tư duy.
- **Độ tin cậy cao trong lĩnh vực pháp lý**: Kết quả đánh giá định lượng cho thấy chỉ số Groundedness (Độ trung thực) và Answer Relevance (Độ phù hợp) luôn duy trì ở mức cao (> 0.9). Việc áp dụng kỹ thuật Prompt Engineering nghiêm ngặt đã giảm thiểu tối đa hiện tượng ảo giác (hallucination) – yếu tố sống còn đối với một ứng dụng pháp luật.

5.2. Hạn chế của hệ thống hiện tại

Bên cạnh những kết quả đạt được, hệ thống vẫn tồn tại một số hạn chế cần khắc phục:

- **Phụ thuộc vào phần cứng:** Do bản chất của các mô hình Deep Learning, hệ thống yêu cầu tài nguyên tính toán lớn. Thời gian phản hồi trên máy tính không có GPU rời (chỉ dùng CPU) là rất chậm (> 10 phút), chưa phù hợp để triển khai rộng rãi cho người dùng phổ thông trên các thiết bị cấu hình thấp.
- **Giới hạn của mô hình nhỏ (1B tham số):** Mặc dù mô hình 1 tỷ tham số (1B) hoạt động nhanh và nhẹ, nhưng khả năng suy luận logic phức tạp vẫn còn hạn chế so với các mô hình lớn hơn (7B, 70B). Trong một số trường hợp câu hỏi quá trừu tượng, mô hình có thể gặp khó khăn trong việc tổng hợp thông tin.

- **Dữ liệu mới chỉ tập trung vào Luật Dược:** Hiện tại hệ thống mới chỉ được nạp dữ liệu của Luật Dược và một số văn bản liên quan, chưa bao phủ được toàn bộ hệ thống văn bản pháp luật y tế rộng lớn.

5.3. Hướng phát triển tương lai

Để hoàn thiện và nâng cao khả năng ứng dụng thực tế, các hướng phát triển tiếp theo được đề xuất bao gồm:

- **Tối ưu hóa tốc độ suy luận:** Nghiên cứu áp dụng các kỹ thuật lượng tử hóa sâu hơn (Quantization) hoặc sử dụng các framework tối ưu như vLLM, ONNX Runtime để cải thiện tốc độ phản hồi trên CPU.
- **Nâng cấp mô hình:** Thử nghiệm với các mô hình nền tảng lớn hơn (như Qwen-7B, Llama-3-8B) nếu điều kiện phần cứng cho phép, nhằm nâng cao khả năng hiểu và giải quyết các tình huống pháp lý phức tạp.
- **Mở rộng cơ sở tri thức:** Tự động hóa quy trình thu thập và cập nhật văn bản pháp luật mới (Thông tư, Nghị định hướng dẫn) để hệ thống luôn đảm bảo tính thời sự của dữ liệu.

Tài liệu tham khảo

Nguồn HuggingFace Qwen,Llama

meta-llama/Llama-3.2-1B · Hugging Face

Qwen/Qwen3-Embedding-0.6B · Hugging Face

Nguồn dữ liệu Luật Dược Việt Nam.

Luật Dược 2016 số 105/2016/QH13 áp dụng 2024 mới nhất

Các nguồn tài liệu tham khảo khác

Word Embedding Techniques in NLP - GeeksforGeeks

Tìm hiểu về Retrieval Augmented Generation (RAG)

RAG là gì? - Giải thích về AI tạo có kết hợp truy xuất thông tin ngoài - AWS

Fine-Tuning using LoRA and QLoRA - GeeksforGeeks

Tutorial: Low-rank Adaptation Techniques in Fine-tuning a Large Language Model

Hybrid search - OpenSearch Documentation

BM25 thuật toán xếp hạng các văn bản theo độ phù hợp

Paper liên quan

[2311.12719] Development of a Legal Document AI-Chatbot

IJNRD2404730.pdf

(PDF) An Overview of Chatbot Technology