**ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**Logo

Description automatically generated**

**LUẬN VĂN TỐT NGHIỆP**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Đề tài**

**XÂY DỰNG TRỢ LÝ ẢO**

**HỖ TRỢ HOẠT ĐỘNG TIẾP XÚC CỬ TRI**

**DEVELOPING A VIRTUAL ASSISSTANT**

**TO SUPPORT VOTER ENGAGEMENT MEETINGS**

**Sinh viên: Lê Tuấn Đạt**

**Mã số: B2113328**

**Khóa: 47**

**Giảng viên hướng dẫn: TS. Trần Nguyễn Minh Thư**

**Cần Thơ, 07/2025**

**ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**Logo

Description automatically generated**

**LUẬN VĂN TỐT NGHIỆP**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Đề tài**

**XÂY DỰNG TRỢ LÝ ẢO**

**HỖ TRỢ HOẠT ĐỘNG TIẾP XÚC CỬ TRI**

**DEVELOPING A VIRTUAL ASSISSTANT**

**TO SUPPORT VOTER ENGAGEMENT MEETINGS**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn**  **TS. Trần Nguyễn Minh Thư** | **Sinh viên thực hiện**  **Họ và tên: Lê Tuấn Đạt Mã số: B2113328 Khóa: 47** |

***Cần Thơ, 07/2025***

# LỜI CẢM ƠN

Để có được bài báo cáo này, em xin được bày tỏ lòng biết ơn chân thành và sâu sắc đến Cô Trần Nguyễn Minh Thư – người đã trực tiếp tận tình hướng dẫn, giúp đỡ em. Trong suốt quá trình thực hiện niên luận, nhờ những sự chỉ bảo và hướng dẫn quý giá đó mà bài niên luận này được hoàn thành một cách tốt nhất.

Em cũng xin gửi lời cám ơn chân thành đến các Thầy Cô Giảng viên Đại học Cần Thơ, đặc biệt là các Thầy Cô ở Trường CNTT & TT, những người đã truyền đạt những kiến thức quý báu trong thời gian qua.

Em cũng xin chân thành cảm ơn bạn bè cùng với gia đình đã luôn động viên, khích lệ và tạo điều kiện giúp đỡ trong suốt quá trình thực hiện để em có thể hoàn thành bài niên luận một cách tốt nhất.

Tuy có nhiều cố gắng trong quá trình thực hiện niên luận, nhưng không thể tránh khỏi những sai sót. Em rất mong nhận được sự đóng góp ý kiến quý báu của quý Thầy Cô và các bạn để bài niên luận hoàn thiện hơn.

|  |  |
| --- | --- |
|  | *Cần Thơ, ngày 19 tháng 08 năm 2025*  ***Người viết***  *(Ký và ghi họ tên)*  Lê Tuấn Đạt |

# MỤC LỤC

[LỜI CẢM ƠN ii](#_Toc31933)

[MỤC LỤC iii](#_Toc5542)

[DANH MỤC HÌNH ẢNH i](#_Toc6140)

[DANH MỤC BẢNG BIỂU ii](#_Toc7676)

[DANH MỤC TỪ VIẾT TẮT iii](#_Toc18532)

[ABSTRACT v](#_Toc10657)

[TÓM TẮT vi](#_Toc25472)

[I. PHẦN GIỚI THIỆU 1](#_Toc24655)

[1. Đặt vấn đề 1](#_Toc24848)

[2. Lịch sử giải quyết vấn đề 2](#_Toc21998)

[2.1. Truy xuất thông tin dựa trên từ khóa 2](#_Toc15329)

[2.2. Học máy cổ điển và trích xuất thông tin: 3](#_Toc32407)

[2.3. Học sâu và các mô hình ngôn ngữ ngữ cảnh: 3](#_Toc21047)

[2.4. Thế hệ mới - Truy xuất tăng cường cho sinh ngôn ngữ (RAG) 4](#_Toc22468)

[3. Mục tiêu đề tài 5](#_Toc2759)

[4. Đối tượng và phạm vi nghiên cứu 6](#_Toc6124)

[4.1. Đối tượng nghiên cứu: 6](#_Toc27228)

[4.2. Phạm vi nghiên cứu: 6](#_Toc33)

[5. Phương pháp nghiên cứu 6](#_Toc26504)

[6. Kết quả đạt được: 6](#_Toc1889)

[7. Bố cục quyển báo cáo 6](#_Toc29981)

[II. PHẦN NỘI DUNG 7](#_Toc5567)

[CHƯƠNG 1. MÔ TẢ BÀI TOÁN 7](#_Toc16712)

[1.1. Mô tả chi tiết bài toán 7](#_Toc30098)

[1.2. Vấn đề và giải pháp liên quan đến bài toán 8](#_Toc22131)

[CHƯƠNG 2. THIẾT KẾ VÀ CÀI ĐẶT GIẢI PHÁP 15](#_Toc4932)

[2.1. Thiết kế hệ thống 15](#_Toc29797)

[2.2. Cài đặt giải pháp 16](#_Toc10364)

[CHƯƠNG 3. KIỂM THỬ VÀ ĐÁNH GIÁ 22](#_Toc27446)

[3.1. Giao diện sản phẩm (nếu có) 22](#_Toc5560)

[3.2. Kết quả thực nghiệm 22](#_Toc20242)

[3.3. Thảo luận về kết quả đạt được 22](#_Toc1426)

[III. PHẦN KẾT LUẬN 23](#_Toc21215)

[1. TÀI LIỆU THAM KHẢO 24](#_Toc3901)

[2. PHỤ LỤC 1](#_Toc18753)

# DANH MỤC HÌNH ẢNH

[Hình 1. Đặc trưng về vị trí của một lát cắt MRI trong một bộ ảnh MRI](#_Toc187925773) **[Error! Bookmark not defined.](#_Toc187925773)**

[Hình 2. Minh họa một nơron nhân tạo](#_Toc187925774) **[Error! Bookmark not defined.](#_Toc187925774)**

[Hình 3. Tổ chức dữ liệu thực nghiệm](#_Toc187925775) **[Error! Bookmark not defined.](#_Toc187925775)**

# DANH MỤC BẢNG BIỂU

[Bảng 1. ………….. 3](#_Toc182489515)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt (Abbreviation)** | **Từ viết đầy đủ (Origin word)** |
| 1 | VPPL | Văn bản quy phạm pháp luật |
| 1 | LLMs | Large Language Models |
| 2 | QA | Question Answering |
| 3 | NLP | Natural Language Processing |
| 4 | GPT | Generative Pre-trained Transformer |
| 5 | BERT | Bidirectional Encoder Representations from Transformers |
| 6 | T5 | Text-To-Text Transfer Transformer |
| 7 | PDF | Portable Document Format |
| 8 | SVM | Support Vector Machine |
| 9 | CRF | Conditional Random Fields |
| 10 | RNN | Recurrent Neural Network |
| 11 | LSTM | Long-Short Term Memory |
| 12 | TF-IDF | Term Frequency-Inverse Document Frequency |
| 13 | BiDAF | Bi-Directional Attention Flow |
| 14 | DrQA | Document Reader for Question Answering |
| 15 | ViT5 | Vietnamese Text-To-Text Transfer Transformer |
| 17 |  |  |

# ABSTRACT

# TÓM TẮT

# I. PHẦN GIỚI THIỆU

## 1. Đặt vấn đề

Trong bối cảnh dữ liệu văn bản ngày càng gia tăng về mặt số lượng và độ phức tạp, việc truy xuất thông tin một cách hiệu quả từ các tài liệu như văn bản quy phạm pháp luật (VPPL) trở thành một yêu cầu thiết yếu. Các văn bản như nghị định, nghị quyết, thông tư,… thường mang tính chất kỹ thuật, có độ dài lớn, cấu trúc phức tạp, và đòi hỏi độ chính xác cao trong quá trình tiếp cận và trích xuất thông tin. Điều này khiến cho không chỉ người dân mà cả cán bộ hành chính cũng gặp nhiều khó khăn trong việc tra cứu, hiểu và sử đúng nội dung các văn bản này. Việc ghi nhớ toàn bộ nội dung, hoặc nhanh chóng tìm ra phần thông tin phù hợp khi cần thiết, vẫn là thách thức lớn trong thực tiễn áp dụng.

Khó khăn này càng thể hiện rõ hơn trong bối cảnh tiếp xúc cử tri - một hoạt động chính trị định kỳ, nơi đại biểu Quốc hội hoặc Hội đồng Nhân dân (HĐND) tiếp nhận ý kiến, phản ánh và kiến nghị từ người dân. Cử tri thường đặt ra nhiều câu hỏi liên quan đến các quy định pháp luật, đặc biệt xoay quanh những vấn đề thời sự có sự ảnh hưởng sâu rộng, như việc sáp nhập đơn vị hành chính. Hiện tại, đây là chủ đề nóng, thu hút nhiều sự quan tâm, kéo theo hàng loạt câu hỏi như: “*Sổ đỏ có phải làm lại không?*”, “*Tôi đang hưởng chính sách ở phường cũ thì sau sáp nhập thế nào?*”, “*Có cần phải đi làm giấy tờ lại hay không?*”. Những câu hỏi này đòi hỏi phản hồi chính xác, căn cứ vào văn bản pháp luật cụ thể. Tuy nhiên, không phải lúc nào đại biểu hay cán bộ hỗ trợ cũng có đủ thời gian và công cụ để nhanh chóng tra cứu, đối chiếu, và trích dẫn chính xác các điều khoản pháp lý liên quan. Điều này làm cho người hỗ trợ không đáp ứng ngay tức thì câu hỏi của người dân vô tình chung làm cho buổi tiếp xúc không được hiệu quả.



Hình 1. Quá trình tiếp xúc cử tri (Nguồn: baochinhphu.vn)

Song song đó, sự phát triển vượt bậc của lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) đã mở ra những hướng đi mới. Các công trình tiên phong như “Attention is All You Need” của Vaswani và cộng sự (2017) đã giới thiệu kiến trúc Transformer, tạo nền tảng cho một thế hệ mô hình ngôn ngữ lớn (LLMs). Đặc biệt, kỹ thuật “Retrieval-Augmented Generation” (RAG), được giới thiệu trong công trình của Lewis và các cộng sự vào năm 2020, đã giải quyết bài toán tạo sinh thông tin dựa trên một tri thức bên ngoài. Hướng tiếp cận này giúp nâng cao đáng kế độ chính xác và cho phép hệ thống trích dẫn nguồn gốc của câu trả lời, giải quyết những hạn chế cốt lõi của các mô hình ngôn ngữ lớn trước đó.

Từ thực tiễn đó, đề tài “Xây dựng trợ lý ảo hỗ trợ hoạt động tiếp xúc cử tri” ra đời với mục tiêu ứng dụng các kỹ thuật mới nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP - Natural Language Processing), nhằm hỗ trợ người dùng - đặc biệt là đại biểu, cán bộ hoặc người dân - có thể truy xuất thông tin pháp luật một cách thông minh, chính xác và thuận tiện, phục vụ tốt hơn cho công tác đối thoại chính sách, tiếp xúc cử tri và áp dụng pháp luật vào thực tiễn.

## 2. Lịch sử giải quyết vấn đề

Việc tra cứu, trích xuất và hỏi đáp thông tin từ văn bản pháp luật luôn là một thách thức lớn, đặc biệt đối với đại biểu Quốc hội, Hội đồng Nhân dân và cán bộ hỗ trợ trong quá trình tiếp xúc cử tri. Các văn bản pháp luật thường có độ dài lớn, cấu trúc phức tạp và ngôn ngữ chuyên ngành, khiến việc tìm kiếm, đối chiếu chính xác các điều khoản liên quan để trả lời câu hỏi của người dân trở nên khó khăn và tốn nhiều thời gian. Trong thực tế, đại biểu và cán bộ thường phải đọc kỹ từng văn bản dài để tìm ra thông tin phù hợp, điều này ảnh hưởng trực tiếp đến hiệu quả và độ chính xác của các buổi tiếp xúc.

### 2.1. Truy xuất thông tin dựa trên từ khóa

Đây là hướng đi sơ khởi và phổ biến nhất, dựa trên các hệ thống truy xuất thông tin (Information Retrieval - IR) truyền thống. Nguyên lý hoạt động của các hệ thống này là lập chỉ mục (indexing) toàn bộ kho văn bản và sử dụng các thuật toán như TF-IDF để tìm kiếm sự trùng khớp của từ khóa do người dùng cung cấp. Tại Việt Nam, các cổng thông tin pháp luật lớn như Cơ sở dữ liệu quốc gia về văn bản pháp luật (vbpl.vn), Thư Viện Pháp Luật hay Luật Việt Nam là những ví dụ điển hình cho hướng tiếp cận này.



Hình 2. Giao diện tìm kiếm văn bản pháp luật với bộ lọc từ khóa và

thông tin hành chính (Nguồn: vbpl.vn)

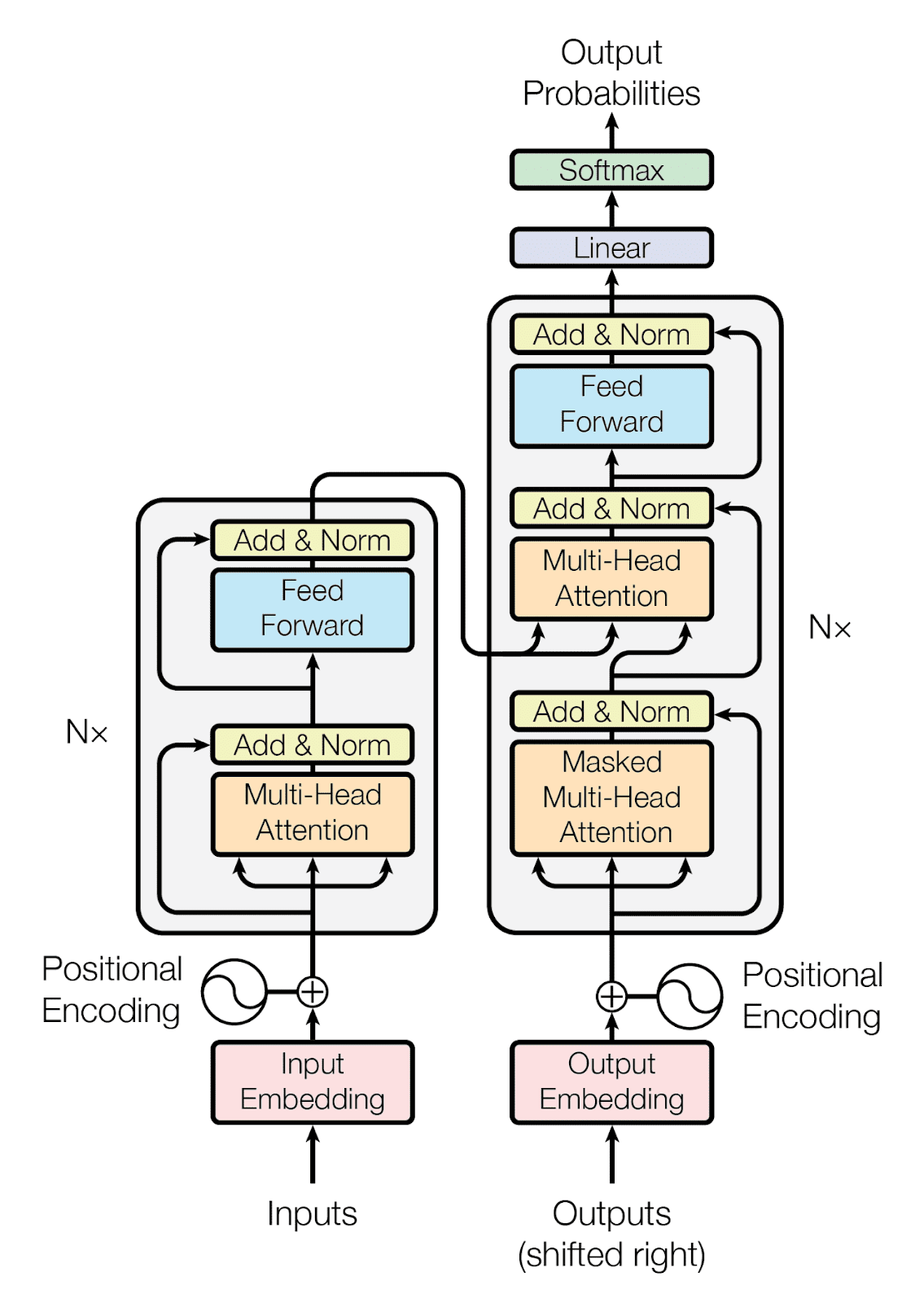
Tuy nhiên, hạn chế cố hữu của phương pháp này là không có khả năng “hiểu” được ngữ nghĩa hay ý định của người dùng. Hệ thống chỉ trả về các văn bản có từ khóa, và người dùng cuối - dù là cán bộ hay người dân - vẫn phải tự đọc và phân tích để tìm ra câu trả lời chính xác, gây mất nhiều thời gian và có thể dẫn đến sai sót.

### 2.2. Học máy cổ điển và trích xuất thông tin:

Để khắc phục những hạn chế trên, các nhà nghiên cứu bắt đầu áp dụng những mô hình học máy cổ điển. Các phương pháp dựa trên luật định (rule-based) sử dụng biểu thức chính quy để nhận diện thông tin ban đầu. Tiếp đó, các mô hình như Naive Bayes, Support Vector Machine (SVM) và Conditional Random Fields (CRF) được ứng dụng để tự động phân loại văn bản theo chủ đề hoặc trích xuất các thực thể có cấu trúc như tên điều khoản, ngày ban hành, v.v.. Mục tiêu của hướng đi này là biến các văn bản phi cấu trúc thành dữ liệu có cấu trúc, từ đó hỗ trợ việc xây dựng đặc trưng thủ công, đồng thời gặp khó khăn trong việc nắm bắt các mối quan hệ ngữ nghĩa phức tạp trong các văn bản pháp luật dài.

### 2.3. Học sâu và các mô hình ngôn ngữ ngữ cảnh:

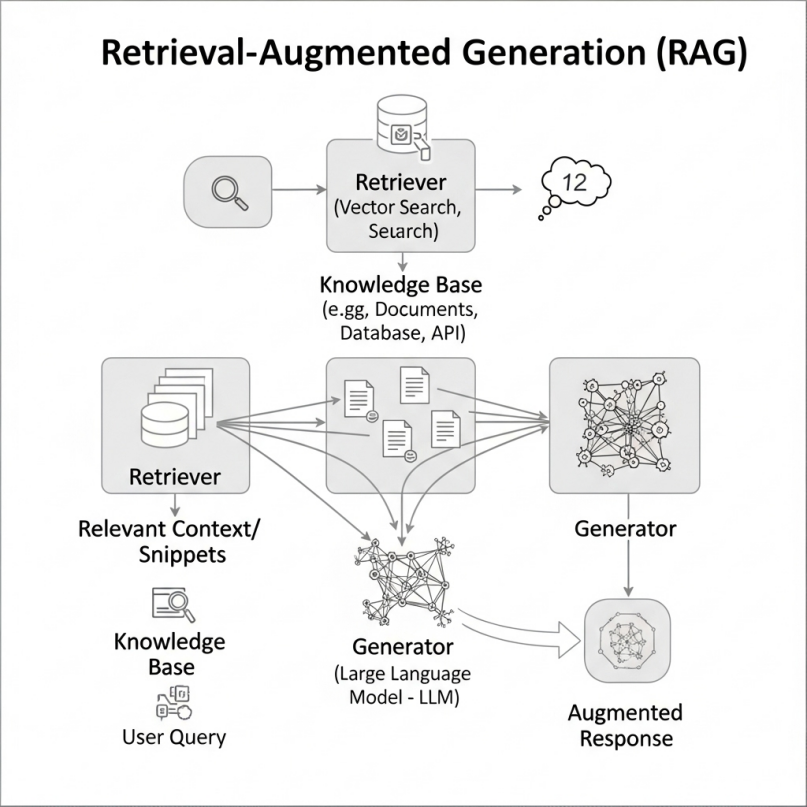
Một cuộc cách mạng thực sự đã đến cùng với sự phát triển của học sâu (Deep Learning) và kiến trúc Transformer, được giới thiệu lần đầu trong công trình “Attention is All You Need” (Vaswani et al., 2017). Sự ra đời của các mô hình ngôn ngữ được huấn luyện trước (pre-trained language models - PLMs) như BERT (Devlin et al., 2018) với cơ chế mã hóa hai chiều đã giúp máy tính lần đầu tiên có thể “hiểu” được một từ trong mối quan hệ ngữ cảnh với toàn bộ câu văn xung quanh nó. Các mô hình thế hệ sau như GPT và T5 tiếp tục đẩy xa giới hạn của việc sinh ngôn ngữ tự nhiên, mở ra tiềm năng khổng lồ cho các hệ thống hỏi-đáp tự động.



Hình 3. Sơ đồ kiến trúc Transformer (Nguồn: Vaswani et al., 2017)

### 2.4. Thế hệ mới - Truy xuất tăng cường cho sinh ngôn ngữ (RAG)

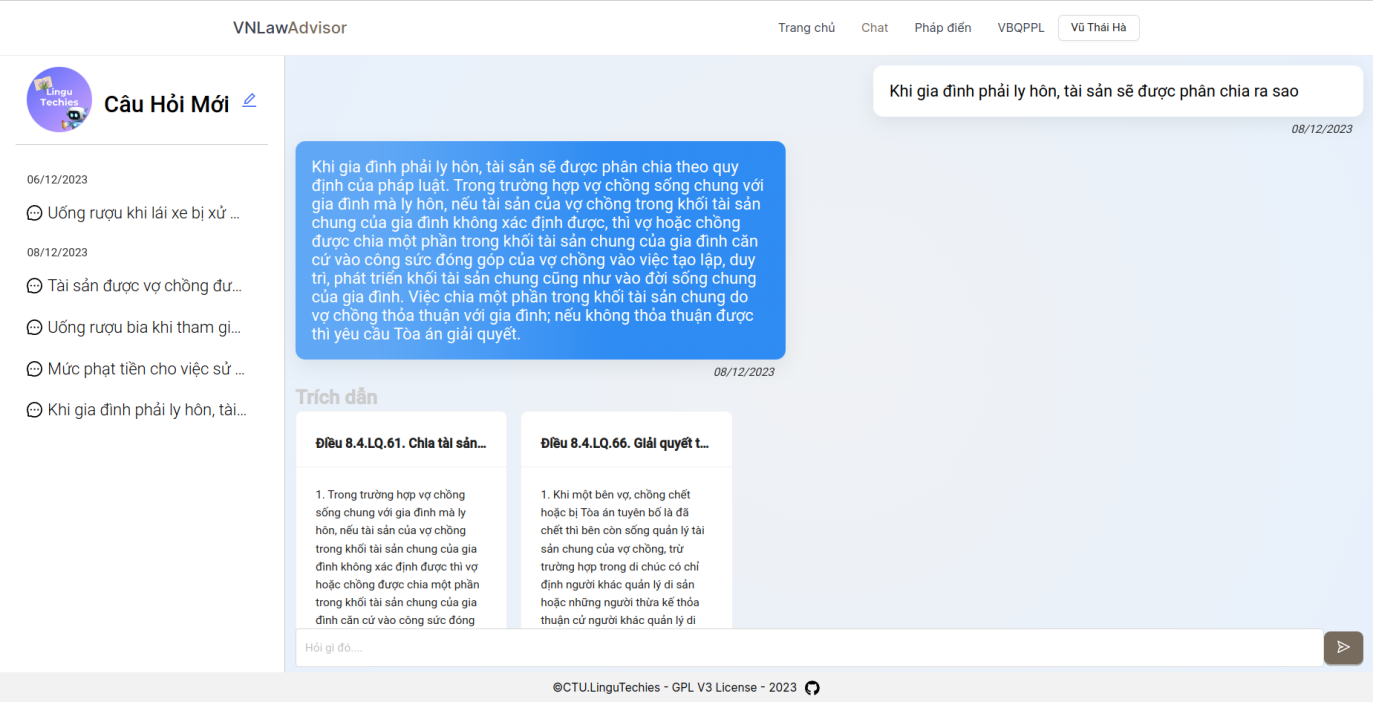
Mặc dù các mô hình ngôn ngữ lớn rất mạnh mẽ, chúng vẫn gặp phải vấn đề ảo giác (hallucination) - tức là tạo ra thông tin không chính xác. Để giải quyết vấn đề này, đặc biệt trong các lĩnh vực yêu cầu tính chính xác tuyệt đối như pháp luật, kỹ thuật Retrieval-Augmented Generation (RAG), được giới thiệu trong bài báo của Lewis và cộng sự (2020), đã trở thành một hướng tiếp cận đột phá. Nguyên tắc của RAG là kết hợp một module truy xuất (Retriever) để tìm các đoạn văn bản liên quan và một module sinh (Generator) để tạo ra câu trả lời dựa trên thông tin đã truy xuất được. Cách tiếp cận này giúp “neo” câu trả lời vào các bằng chứng xác thực, giảm thiểu thông tin sai lệch và cho phép hệ thống trích dẫn nguồn gốc rõ ràng.



Hình 4. Sơ đồ kỹ thuật RAG

Trên thế giới, nhiều dự án và hệ thống AI hỗ trợ pháp luật đã được phát triển nhằm tự động hóa quá trình tra cứu và phân tích văn bản pháp luật. Ví dụ, Siren Analytics sử dụng kỹ thuật RAG kết hợp mô hình ngôn ngữ lớn để tự động trích xuất và phân tích các văn bản luật quốc tế, hỗ trợ luật sư và chuyên gia pháp lý trong việc nghiên cứu và tư vấn. Các AI Expert Agents chuyên sâu về luật quốc tế cũng được triển khai để phân tích công ước, án lệ và tài liệu pháp lý phức tạp, giúp nâng cao hiệu quả nghiên cứu và ra quyết định. Ngoài ra, các hội thảo quốc tế như Legal Information Retrieval meets Artificial Intelligence (LIRAI) tập trung phát triển các phương pháp kết hợp AI và truy xuất thông tin pháp lý, giải quyết các thách thức về độ chính xác và khả năng giải thích trong môi trường phức tạp. Một số chatbot pháp lý tiên tiến cũng áp dụng kỹ thuật RAG để cung cấp câu trả lời chính xác dựa trên tài liệu pháp luật chính thống, đồng thời hạn chế lỗi do mô hình sinh thông tin không chính xác (hallucination).

Tại Việt Nam, các mô hình ngôn ngữ thuần Việt như PhoBERT, ViT5 và Vietnamese-SBERT đã được phát triển, hỗ trợ hiệu quả trong các tác vụ xử lý văn bản tiếng Việt. Một ví dụ điển hình là dự án VN-Law-Advisor do nhóm CTU-LinguTechies phát triển, sử dụng mô hình phoGPT kết hợp kỹ thuật RAG và cơ sở dữ liệu vector Chroma để cung cấp hệ thống hỏi đáp pháp luật tự động, hỗ trợ đại biểu và cán bộ tra cứu, đối chiếu thông tin pháp luật một cách nhanh chóng và chính xác, đồng thời trích dẫn rõ ràng các điều khoản liên quan.



Hình 4. Giao diện ứng dụng tra cứu, hỏi đáp pháp luật của dự án VN-Law-Advisor

(Nguồn: https://github.com/CTU-LinguTechies/VN-Law-Advisor)

Dự án VN-Law-Advisor đã xây dựng thành công một hệ thống tra cứu pháp luật toàn diện dựa trên Bộ pháp điển và các văn bản quy phạm pháp luật từ năm 1990 đến nay, được cập nhật liên tục. Hệ thống sử dụng kiến trúc microservices hiện đại, tích hợp các công cụ xử lý PDF (PyMuPDF), tiền xử lý ngôn ngữ tiếng Việt (VnCoreNLP, Underthesea) và lưu trữ embedding trong cơ sở dữ liệu vector Chroma để truy xuất nhanh và chính xác. Người dùng có thể đặt câu hỏi tự nhiên,, hệ thống sẽ tự động truy xuất thông tin liên quan và sinh câu trả lời mạch lạc, kèm trích dẫn điều khoản cụ thể, giúp đại biểu và cán bộ hỗ trợ trong các buổi tiếp xúc cử tri có thể phản hồi nhanh chóng, chính xác và có căn cứ pháp lý vững chắc.

## Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một hệ thống hỏi - đáp tự động có khả năng tiếp nhận đầu vào là các văn bản pháp luật tiếng Việt dưới định dạng tập tin PDF, phân tích nội dung bằng các kỹ thuật xử lý ngôn ngữ tự nhiên, và sinh ra câu trả lời phù hợp với câu hỏi người dùng. Hệ thống hướng đến việc xử lý tốt các đặc thù của văn bản pháp lý như ngôn ngữ chuyên ngành, cấu trúc phi chuẩn, độ dài lớn và yêu cầu chính xác cao. Đồng thời, đề tài cũng tập trung vào việc tích hợp các mô hình ngôn ngữ tiên tiến nhằm nâng cao khả năng truy xuất thông tin theo ngữ cảnh, phục vụ hiệu quả cho các tình huống thực tiễn như tra cứu quy định pháp luật, tham vấn chính sách hoặc hỗ trợ công tác tiếp xúc cử tri. Mục tiêu cuối cùng là góp phần tạo ra một giải pháp có tính ứng dụng cao, giúp rút ngắn thời gian tìm kiếm và cải thiện khả năng tiếp cận thông tin pháp luật cho người dân và cán bộ chuyên môn.

## 4. Đối tượng và phạm vi nghiên cứu

### 4.1. Đối tượng nghiên cứu:

Đối tượng nghiên cứu của đề tài là các phương pháp và mô hình trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đặc biệt là tập trung vào bài toán hỏi - đáp tự động (Question Answering) trên ngữ liệu tiếng Việt. Ngoài ra, đề tài còn nghiên cứu kỹ thuật xử lý tài liệu đầu là các tập tin PDF văn bản pháp luật, bao gồm trích xuất thông tin và lưu trữ có cấu trúc để phục vụ cho việc truy xuất và sinh câu trả lời.

### 4.2. Phạm vi nghiên cứu:

Đề tài tập trung vào việc xây dựng một hệ thống có khả năng:

* Nhận diện và trích xuất nội dung từ tài liệu văn bản pháp luật tiếng Việt dưới định dạng tập tin PDF.
* Lưu trữ và tổ chức thông tin pháp lý đã trích xuất để hỗ trợ tìm kiếm và truy vấn hiệu quả.
* Phát triển một chatbot có khả năng tự động sinh câu trả lời cho các câu hỏi pháp luật dựa trên dữ liệu đã lưu trữ.

Hệ thống được triển khai thử nghiệm trên tập hợp các văn bản quy phạm pháp luật phổ biến (như nghị định, nghị quyết, thông tư, luật, …) với phạm vi ngôn ngữ tiếng Việt. Đề tài không đi sâu vào các vấn đề như diễn giải pháp lý hoặc tư vấn luật chuyên sâu.

## 5. Phương pháp nghiên cứu

## 6. Kết quả đạt được:

## Bố cục quyển báo cáo

**Phần 1.** Phần giới thiệu

**Phần 2.** Phần nội dung

**Chương 1:** Mô tả bài toán.

**Chương 2:** Thiết kế và cài đặt giải pháp

**Chương 3:** Kiểm thử và đánh giá

**Phần 3**. Phần kết luận và hướng phát triển.

# II. PHẦN NỘI DUNG

## CHƯƠNG 1. MÔ TẢ BÀI TOÁN

### 1.1. Mô tả chi tiết bài toán

Bài toán nghiên cứu được xác định từ nhu cầu thực tiễn trong hoạt động lập pháp và giám sát, đặc biệt khi xử lý các vấn đề chính sách phức tạp như việc sáp nhập đơn vị hành chính. Để hình dung, hãy đặt bối cảnh cụ thể: Một buổi tiếp xúc cử tri tại một phường sắp được sáp nhật. Không khí buổi họp rất nóng với hàng loạt câu hỏi cấp bách: *“Sổ đỏ của gia đình tôi có phải làm lại không?”*, “*Các chế độ chính sách tôi đang hưởng sẽ được giải quyết như thế nào?”*. Vị Đại biểu cần đưa ra những câu trả lời khổng chỉ để trấn an dư luận, mà còn phải chính xác và có căn cứ pháp lý rõ ràng.

Hình 5. Văn bản

Bài toán đặt ra là phải xây dựng một hệ thống trợ lý ảo pháp lý hoạt động như một “chuyên gia”, được trang bị một kho kiến thức chuyên sâu và toàn diện về vấn đề sáp nhập. Trải nghiệm của người dùng được thiết kế như một cuộc đối thoại trực tiếp, nơi họ có thể đặt câu hỏi và nhận về câu trả lời đánh tin cậy mà không cần thực hiện bất kỳ thao tác kỹ thuật phức tạp nào.

Hệ thống được trang bị sẵn một kho kiến thức thống nhất, đã được nạp và xử lý từ trước. Kho kiến thức này bao gồm các văn bản chung của nhà nước như Luật, Chỉ thị, và Công điện do các cơ quan trung ương ban hành. Điều này đảm bảo trợ lý ảo có khả năng cung cấp các câu trả lời dựa trên nền tảng pháp lý chung và chính thống của quốc gia. Người dùng cuối có thể tải lên các tập tin văn bản quy phạm pháp luật dạng PDF để truy vấn thông tin trong văn bản đó.

Khi người dùng nhập câu hỏi, hệ thống sẽ tự động phân tích ngữ nghĩa của câu hỏi đó. Thay vì tìm kiếm từ khóa một cách máy móc, nó sẽ thực hiện một cuộc tìm kiếm thông minh trong toàn bộ kho trí thức đã được số hóa để tìm ra những điều khoản, quy định chính xác nhất liên quan đến vấn đề người dùng quan tâm. Các ngữ cảnh phù hợp nhất sau đó sẽ được một mô hình ngôn ngữ lớn tổng hợp và diễn đạt lại thành một câu trả lời tự nhiên hơn, dễ hiểu thông qua kỹ thuật RAG.

Câu trả lời cuối cùng được trình bày như một bản tư vấn hoàn chỉnh, luôn bao gồm hai phần:

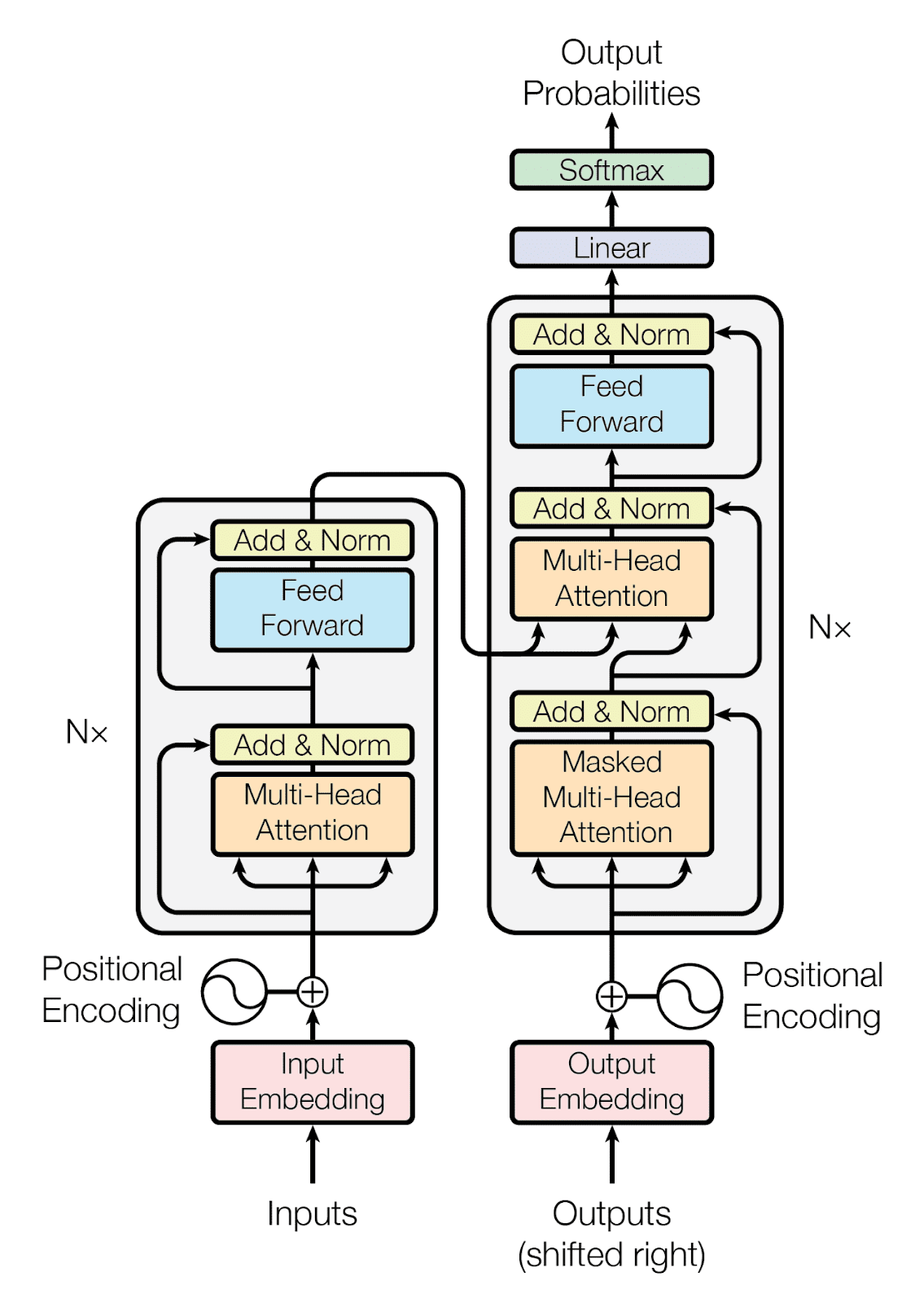
* Một phần giải thích chi tiết, cung cấp đầy đủ bối cảnh và lập luận.
* Một danh sách các nguồn trích dẫn, chỉ rõ thông tin được lấy từ điều khoản nào, của văn bản nào?

Phần trích dẫn này cho phép người dùng xem và kiểm chứng ngay lập tức với văn bản gốc, tạo ra sự minh bạch và tin cậy tuyệt đối. Trong trường hợp không tìm thấy thông tin phù hợp, hệ thống được thiết kế để phản hồi một cách trung thực, đảm bảo không “sáng tạo” thông tin sai lệch.

### 1.2. Vấn đề và giải pháp liên quan đến bài toán

#### 1.2.1. Kiến trúc Transformer:

Kiến trúc Transformer được giới thiệu lần đầu tiên trong bài báo nổi tiếng “Attention is All You Need” của Vaswani và cộng sự năm 2017. Nó đã tạo nên một cuộc cách mạng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và nhanh chóng trở thành nền tảng cho hầu hết các mô hình ngôn ngữ lớn (LLMs) hiện đại như BERT, GPT, T5 hay ViT5. Điểm đặc biệt là nó hoàn toàn dựa trên một cơ chế gọi là “Attention” (Chú ý), loại bỏ hoàn toàn sự cần thiết của các thành phần tuần tự (recurrent layers) như trong RNN hay LSTM, vốn là kiến trúc chủ đạo trước đó cho các bài toán xử lý chuỗi. Điều này cho phép Transformer xử lý song song các phần của chuỗi đầu vào, giúp tăng tốc độ huấn luyện và giải quyết hiệu quả vấn đề phụ thuộc xa (long-range dependencies).



Hình 6. Kiến trúc Transformer (Nguồn: Vaswani et al., 2017)

Transformer ban đầu được thiết kế cho bài toán dịch máy (sequence-to-sequence task) và bao gồm hai thành phần chính: Khối mã hóa (Encoder) và khối giải mã (Decoder). Cả Encoder và Decoder đều được cấu tạo từ nhiều tầng (layers) giống hệt nhau được xếp chồng lên nhau. Trong bài báo gốc, số lượng lớp cho cả Encoder và Decoder là sáu (N=6). Khối mã hóa tiếp nhận một chuỗi đầu vào và tạo ra một chuỗi các biểu diễn liên tục, giàu ngữ cảnh (contextualized representations) cho chuỗi đó. Những biểu diễn này nắm bắt ý nghĩa của từng từ trong mối quan hệ với các từ khác trong câu. Khối giải mã tiếp nhận đầu ra của khối mã hóa (các biểu diễn ngữ cảnh) cùng với chuỗi đầu ra đã được tạo ra ở bước trước đó để tạo ra từ tiếp theo trong chuỗi đầu ra một cách tự hồi quy.

#### 1.2.2. Mô hình ngôn ngữ lớn (Large Language Models - LLMs):

Mô hình ngôn ngữ lớn (LLMs) là một loại mô hình trí tuệ nhân tạo được huấn luyện trên một khối lượng dữ liệu văn bản khổng lồ, cho phép chúng có khả năng hiểu, tóm tắt, tạo sinh và dự đoán nội dung ngôn ngữ của con người một cách tinh vi. Nền tảng kiến trúc của hầu hết các LLM hiện đại đều dựa trên kiến trúc Transformer, giúp chúng xử lý hiệu quả các mối quan hệ ngữ nghĩa phức tạp và các phụ thuộc xa trong văn bản.

Về cơ bản, LLM hoạt động bằng cách học các mẫu (patterns), cấu trúc ngữ pháp và các mối liên hệ về ngữ nghĩa từ hàng tỷ câu văn. Dựa trên những gì đã học được, nhiệm vụ cốt lõi của chúng là dự đoán từ tiếp theo có khả năng xuất hiện cao nhất trong một chuỗi văn bản. Bằng cách lặp lại quá trình này, chúng có thể tạo ra các đoạn văn, bài luận, mã nguồn, hoặc các câu trả lời mạch lạc và tự nhiên.

Dựa trên việc sử dụng các thành phần của kiến trúc Transformer, các LLM thường được chia thành ba loại chính:

* **Mô hình chỉ bao gồm khối mã hóa (Encoder-only)**: Tiêu biểu là BERT và các biến thể của nó. Các mô hình này được thiết kế để xây dựng một biểu diễn ngữ nghĩa sâu sắc cho toàn bộ câu văn. Chúng có khả năng "nhìn" được cả ngữ cảnh bên trái và bên phải của một từ, do đó cực kỳ mạnh mẽ trong các tác vụ yêu cầu sự "hiểu" sâu như phân loại văn bản, nhận dạng thực thể, hoặc phân tích cảm xúc.
* **Mô hình chỉ bao gồm khối giải mã (Decoder-only)**: Tiêu biểu là dòng mô hình GPT. Các mô hình này hoạt động theo cơ chế tự hồi quy, tức là chúng dự đoán từ tiếp theo dựa trên các từ đã được sinh ra trước đó. Điều này làm cho chúng trở nên vượt trội trong các tác vụ tạo sinh văn bản một cách tự do, chẳng hạn như viết truyện, soạn email, hoặc đóng vai trò là một chatbot đối thoại.
* **Mô hình bao gồm cả khối mã hóa và giải mã (Encoder-Decoder)**: Tiêu biểu là T5 hay ViT5. Các mô hình này được thiết kế cho các bài toán "đầu vào-đầu ra" (sequence-to-sequence), nơi chúng nhận một chuỗi đầu vào, mã hóa nó thành một biểu diễn ngữ nghĩa, sau đó giải mã biểu diễn đó thành một chuỗi đầu ra mới. Chúng rất hiệu quả trong các tác vụ như dịch máy, tóm tắt văn bản, và trả lời câu hỏi.

Trong khuôn khổ của luận văn này, mô hình ngôn ngữ lớn được lựa chọn để đóng vai trò “Generator” tron hệ thống RAG là **Gemini**, một mô hình tiên tiến do Google phát triển. Cụ thể, phiên bản được sử dụng là **Gemini 2.5 Flash**, được chọn do các ưu điểm sau:

* **Hiệu suất cao**: Gemini Flash được tối ưu để có tốc độ phản hồi nhanh trong khi vẫn duy trì chất lượng cao, rất phù hợp cho một ứng dụng hỏi-đáp tương tác.
* **Cửa sổ ngữ cảnh lớn (Large Context Window):** Mô hình có khả năng xử lý một lượng lớn văn bản đầu vào cùng lúc, cho phép nó tổng hợp thông tin từ nhiều đoạn ngữ cảnh được truy xuất một cách hiệu quả.
* **Hiệu quả về chi phí:** Là một mô hình nhỏ gọn và hiệu quả, việc sử dụng Gemini Flash giúp giảm chi phí tính toán, phù hợp với các dự án nghiên cứu và phát triển sản phẩm ở giai đoạn đầu.

Trong hệ thống RAG của tôi, Gemini sẽ không trả lời dựa trên kiến thức nội tại của nó. Thay vào đó, nó sẽ nhận câu hỏi từ người dùng cùng với các đoạn văn bản pháp luật liên quan được truy xuất từ cơ sở dữ liệu vector. Nhiệm vụ của nó là đóng vai trò như một “bộ não suy luận”, đọc hiểu các ngữ cảnh được cung cấp và tổng hợp chúng thành một câu trả lời cuối cùng, mạch lạc, chính xác và bám sát vào nguồn thông tin pháp lý đã cho.

#### 1.2.3. Hệ thống hỏi đáp (Question Answering - QA):

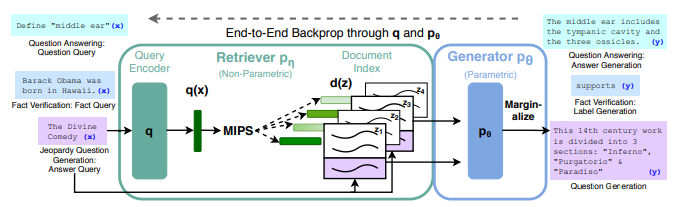
Hệ thống Hỏi-đáp (Question Answering - QA) là một lĩnh vực chuyên sâu trong Xử lý Ngôn ngữ Tự nhiên (NLP) với mục tiêu chính là xây dựng các hệ thống có khả năng tự động trả lời các câu hỏi do con người đặt ra bằng ngôn ngữ tự nhiên. Khác với các công cụ tìm kiếm truyền thống chỉ trả về một danh sách các tài liệu có thể liên quan, một hệ thống QA lý tưởng sẽ cung cấp một câu trả lời ngắn gọn, chính xác và trực tiếp cho câu hỏi, giúp người dùng tiết kiệm thời gian và công sức phân tích.

Các hệ thống QA ban đầu thường dựa trên các phương pháp trích xuất thông tin (Information Extraction) và các mẫu được định sẵn. Cùng với sự phát triển của học máy, các mô hình phức tạp hơn đã ra đời, có khả năng đọc và hiểu một đoạn văn bản để tìm ra câu trả lời. Các kiến trúc tiên phong trong giai đoạn này, chẳng hạn như DrQA (Document Reader for Question Answering) hay BiDAF (Bi-Directional Attention Flow), đã đạt được những thành công đáng kể trong việc xác định chính xác "span" (một đoạn văn bản) chứa câu trả lời trong một tài liệu cho trước. Dạng QA này được gọi là "Hỏi-đáp trích xuất" (Extractive QA).

Tuy nhiên, hạn chế của các hệ thống trích xuất là chúng chỉ có thể lấy ra thông tin đã tồn tại y nguyên trong văn bản mà không có khả năng tổng hợp, diễn giải hay trả lời các câu hỏi phức tạp đòi hỏi suy luận từ nhiều nguồn. Một bước đột phá thực sự đã đến cùng với sự trỗi dậy của các Mô hình Ngôn ngữ Lớn (LLMs). Các hệ thống QA hiện đại không còn chỉ trích xuất, mà đã có thể tạo sinh ra các câu trả lời hoàn toàn mới, mạch lạc và tự nhiên hơn.

Kỹ thuật Retrieval-Augmented Generation (RAG) chính là một kiến trúc tiêu biểu cho thế hệ hệ thống QA này. Bằng cách kết hợp một module truy xuất thông tin (Retriever) và một module tạo sinh (Generator) dựa trên LLM, hệ thống RAG có thể cung cấp những câu trả lời vừa bám sát vào nguồn kiến thức xác thực, vừa được diễn đạt một cách tự nhiên, phù hợp với ngữ cảnh của câu hỏi.

#### 1.2.4. Retrieval-augmented Generation (RAG):



Hình 8. Kiến trúc kỹ thuật RAG (Nguồn: Lewis et al., 2020)

Retrieval-Augmented Generation (RAG) là một khung trí tuệ nhân tạo hoặc một kỹ thuật học máy được giới thiệu lần đầu tiên trong bài báo “*Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”* (Lewis et al., 2020) nhằm nâng cao khả năng của các mô hình ngôn ngữ lớn (LLM) bằng cách tích hợp chúng với khả năng truy xuất thông tin từ bên ngoài. Điều này có nghĩa là LLM không còn chỉ dựa vào dữ liệu huấn luyện tĩnh của chúng nữa. Nguyên tắc cốt lõi của RAG là “neo” các LLM vào thông tin chính xác, cập nhật nhất từ một cơ sở tri thức bên ngoài trước khi tạo ra phản hồi. Cách tiếp cận này giúp giải quyết “khoảng trống kiến thức” trong cách hoạt động của các LLM truyền thống.

Động lực chính đằng sau sự phát triển của RAG bao gồm việc cho phép tạo ra văn bản chính xác, cập nhật và có thể kiểm chứng được. Quan trọng hơn, RAG giải quyết một số hạn chế chính của LLM, chẳng hạn như việc tạo ra “ảo giác” (halluciation) - tức là thông tin nghe có vẻ hợp lý nhưng thực tế lại không chính xác hoặc gây hiểu lầm. Bằng cách neo quá trình tạo sinh vào các tài liệu được truy xuất và đã được xác minh, RAG giảm thiểu việc tạo ra nội dung sai lệch. Ngoài ra, RAG còn cung cấp cho người dùng cái nhìn sâu sắc về quy trình tạo sinh của LLM bằng cách liên kết các phản hồi với thông tin nguồn.

Hơn nữa, động lực đằng sau RAG không chỉ là dừng lại ở việc đạt được độ chính xác thực tế; nó còn hướng đến việc xây dựng niềm tin và khả năng kiểm chứng trong nội dung do AI tạo ra, điều này rất quan trọng đối với việc áp dụng trong doanh nghiệp và các ứng dụng quan trọng. Hiện tượng ảo giác của LLM là một rào cản lớn đối với niềm tin. RAG nhằm mục đích giảm thiểu những điều này bằng cách đưa các phản hồi vào các tài liệu có thể kiểm chứng. Khả năng truy xuất nguồn gốc của nội dung được tạo ra từ một nguồn sự thật là một kết quả then chốt. Khả năng truy vết này giúp củng cố niềm tin của người dùng vào giải pháp AI.

Từ góc độ vĩ mô, các hệ thống RAG có thể được chia thành hai thành phần chính:

- Phần truy xuất (Retriever): Bao gồm các hoạt động đa dạng như tiền xử lý dữ liệu bên ngoài, các cơ chế truy xuất dày đặc (dense) hoặc thưa thớt (sparse) để tìm kiếm tài liệu liên quan, và có thể bao gồm cả việc xếp hạng lại (reranking) và tỉa bớt (pruning) các tài liệu này. Thành phần truy xuất sẽ tìm kiếm trong các cơ sở dữ liệu / nguồn bên ngoài dựa trên truy vấn người dùng.

- Phần tạo sinh: Bao gồm các thành phần như lập kế hoạch truy xuất (quyết định cái gì / làm thế nào để truy xuất), tích hợp kiến thức từ nhiều nguồn, và suy luận logic bởi LLM để tổng hợp một phản hồi dựa trên câu lệnh đã được tăng cường. Mô-đun tạo sinh xử lý thông tin được truy xuất để tạo ra văn bản giống như con người.

Ngoài ra, các yếu tố liên kết khác bao gồm việc phân đoạn tài liệu (document chunking), tạo vector nhúng (embedding generation), và các cơ chế đảm bảo an ninh và độ tin cậy.

Bảng 1. Các thành phần cốt lỗi của hệ thống RAG

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả / Chức năng** |
| Nguồn dữ liệu | Kho lưu trữ kiến trức bên ngoài mà RAG sẽ truy xuất thông tin. |
| Bộ chỉ mục / Bộ nhúng  (Indexer / Embedder) | Xử lý và chuyển đổi dữ liệu nguồn thành các biểu diễn số (vector nhúng) và lưu trữ chúng trong một cơ sở dữ liệu vector để tìm kiếm hiệu quả. |
| Bộ truy xuất  (Retriever) | Tìm kiếm và lấy các đoạn thông tin liên quan nhất từ cơ sở dữ liệu vector dựa trên truy vấn của người dùng. |
| Bộ tăng cường  (Augmenter) | Kết hợp thông tin được truy xuất với truy vấn ban đầu của người dùng để tạo ra một câu lệnh tăng cường cho LLM. |
| Bộ tạo sinh / LLM  (Generator / LLM) | Nhận câu lệnh tăng cường và tạo ra một phản hồi mạch lạc, dựa trên ngữ cảnh, kết hợp cả kiến thức được truy xuất và kiến thức nội tại của nó. |

RAG mang lại nhiều ưu điểm đáng kể, nổi bật nhất là khả năng tăng cường độ chính xác và giảm thiểu tình trạng “ảo giác” của mô hình ngôn ngữ lớn bằng cách truy xuất thông tin thực tế từ các nguồn dữ liệu bên ngoài. Điều này cũng cho phép RAG linh hoạt cập nhật kiến thức mới mà không cần huấn luyện lại toàn bộ mô hình, đồng thời tăng tính minh bạch khi câu trả lời thường đi kèm nguồn trích dẫn để người dùng kiểm chứng, và cải thiện khả năng xử lý kiến thức chuyên ngành hiệu quả. Tuy nhiên, RAG cũng tồn tại một số hạn chế. Hiệu quả của nó phụ thuộc lớn vào chất lượng của nguồn dữ liệu được truy xuất; nếu dữ liệu không tốt, kết quả cũng sẽ bị ảnh hưởng. Bên cạnh đó, việc triển khai RAG làm tăng độ phức tạp của hệ thống và có thể gây ra độ trễ trong thời gian phản hồi do quá trình truy xuất. Cuối cùng, luôn có nguy cơ cơ chế truy xuất không chọn được thông tin tối ưu hoặc bị ảnh hưởng bởi các thiên kiến tiềm ẩn trong dữ liệu. Mặc dù vậy, RAG vẫn là một giải pháp mạnh mẽ để nâng cao chất lượng và độ tin cậy của LLM, nhưng cần cân nhắc kỹ về nguồn dữ liệu và độ phức tạp khi triển khai.

#### 1.2.5. Nhúng vector (Vector embedding):

Vector embedding (Nhúng vector) là một kỹ thuật nền tảng trong Xử lý Ngôn ngữ Tự nhiên (NLP) hiện đại, với nhiệm vụ chuyển đổi các đơn vị văn bản (như từ, câu, hoặc đoạn văn) thành các biểu diễn dạng số, cụ thể là các vector trong một không gian đa chiều. Mục tiêu chính của kỹ thuật này là mã hóa **ngữ nghĩa** của văn bản vào một dạng mà máy tính có thể hiểu và xử lý được về mặt toán học.

Trong không gian vector này, các từ hoặc các câu có ý nghĩa tương tự nhau sẽ được biểu diễn bởi các vector nằm gần nhau. Ví dụ, vector của từ "luật sư" sẽ nằm gần vector của từ "thẩm phán" hơn là so với vector của từ "bác sĩ". Mối quan hệ ngữ nghĩa này không chỉ giới hạn ở sự tương đồng, mà còn có thể thể hiện các mối quan hệ phức tạp khác như quan hệ nhân quả hay đối lập. Quá trình tạo ra các vector này thường được thực hiện bởi các mô hình ngôn ngữ lớn đã được huấn luyện trước trên một kho dữ liệu khổng lồ, cho phép chúng học được các sắc thái tinh tế của ngôn ngữ.

Trong kiến trúc RAG của luận văn này, vector embedding đóng hai vai trò then chốt. **Thứ nhất**, trong giai đoạn xử lý dữ liệu, toàn bộ các đoạn văn bản từ kho kiến thức pháp luật sẽ được một mô hình embedding chuyển đổi thành các vector và lưu trữ trong cơ sở dữ liệu vector ChromaDB. **Thứ hai**, trong giai đoạn vận hành, khi người dùng đặt một câu hỏi, câu hỏi đó cũng sẽ được chuyển đổi thành một vector bằng chính mô hình embedding đó. Vector của câu hỏi sau đó sẽ được sử dụng để tìm kiếm các vector văn bản gần nhất (tương đồng nhất về mặt ngữ nghĩa) trong ChromaDB, từ đó truy xuất ra những ngữ cảnh phù hợp nhất để trả lời.

Hệ thống sử dụng mô hình text-embedding-004 của Google, một mô hình mạnh mẽ và hiệu quả được thiết kế để tạo ra các vector embedding chất lượng cao cho các tác vụ tìm kiếm và truy xuất thông tin.

#### 1.2.6. Cơ sở dữ liệu vector (Vector database/stores):

Cơ sở dữ liệu vector (vector database/stores) là một loại cơ sở dữ liệu được thiết kế chuyên biệt để lưu trữ, quản lý và truy vấn các vector embedding một cách hiệu quả. Không giống như các cơ sở dữ liệu quan hệ truyền thống lưu trữ dữ liệu có cấu trúc trong các bảng, cơ sở dữ liệu vector lưu trữ dữ liệu dưới dạng các vector đa chiều trong một không gian toán học, nơi khoảng cách và hướng của các vector thể hiện mối quan hệ ngữ nghĩa giữa các đối tượng dữ liệu.

Chức năng cốt lõi của một cơ sở dữ liệu vector là khả năng thực hiện các truy vấn tìm kiếm tương đồng (similarity search) với tốc độ cực nhanh. Thay vì tìm kiếm dựa trên sự trùng khớp từ khóa chính xác, nó tìm kiếm các vector "láng giềng" gần nhất với vector của câu hỏi truy vấn. Quá trình này, thường được thực hiện bằng các thuật toán như Approximate Nearest Neighbor (ANN), cho phép hệ thống tìm ra các đoạn văn bản có ý nghĩa tương tự nhau, ngay cả khi chúng không chia sẻ chung một từ khóa nào. Đây chính là nền tảng của tìm kiếm ngữ nghĩa (semantic search), giúp máy tính "hiểu" được ý định của người dùng thay vì chỉ khớp chuỗi ký tự.

Trong hệ thống RAG, cơ sở dữ liệu vector đóng vai trò như một "bộ não" lưu trữ toàn bộ kiến thức đã được số hóa. Mỗi đoạn văn bản từ các tài liệu pháp luật sau khi được chuyển đổi thành vector embedding sẽ được lưu trữ và lập chỉ mục trong cơ sở dữ liệu này. Khi người dùng đặt câu hỏi, câu hỏi đó cũng được chuyển thành một vector và được dùng để truy vấn, tìm ra các đoạn văn bản có nội dung liên quan nhất để làm ngữ cảnh cho mô hình ngôn ngữ lớn (LLM).

#### 1.2.7. Phương pháp đánh giá hệ thống - RAGAS:

Việc đánh giá chất lượng của một hệ thống Hỏi-đáp dựa trên kiến trúc RAG là một bài toán phức tạp, bởi nó không chỉ phụ thuộc vào chất lượng của câu trả lời cuối cùng mà còn cả chất lượng của bước truy xuất thông tin. Để giải quyết vấn đề này, RAGAS (Retrieval-Augmented Generation Assessment) đã được phát triển như một framework chuyên biệt, cung cấp một bộ các chỉ số đo lường khách quan và toàn diện cho hiệu suất của pipeline RAG.

Điểm đặc biệt của RAGAS là nó hoạt động mà không cần đến một bộ dữ liệu "vàng" (golden dataset) với các câu trả lời mẫu do con người viết tay. Thay vào đó, nó sử dụng chính các Mô hình Ngôn ngữ Lớn (LLMs) để thực hiện việc đánh giá một cách tự động dựa trên câu hỏi, ngữ cảnh được truy xuất và câu trả lời do hệ thống tạo ra. Điều này giúp quá trình đánh giá trở nên linh hoạt và có khả năng mở rộng cao.

Framework RAGAS tập trung vào việc đo lường hai khía cạnh chính của hệ thống: chất lượng của module truy xuất (Retriever) và chất lượng của module sinh (Generator). Các chỉ số đánh giá cốt lõi bao gồm:

* **Faithfulness (Tính trung thực):** Đây là chỉ số quan trọng nhất để chống lại hiện tượng "ảo giác". Nó đo lường mức độ mà câu trả lời do LLM tạo ra hoàn toàn bám sát và có thể suy ra từ các đoạn ngữ cảnh đã được cung cấp. Một điểm số Faithfulness cao cho thấy hệ thống đang trả lời dựa trên bằng chứng xác thực, chứ không bịa đặt thông tin.
* **Answer Relevancy (Độ liên quan của câu trả lời):** Chỉ số này đánh giá xem câu trả lời được tạo ra có thực sự là câu trả lời cho câu hỏi của người dùng hay không. Đôi khi, một câu trả lời có thể trung thực (bám sát ngữ cảnh) nhưng lại không đi thẳng vào vấn đề mà người dùng quan tâm.
* **Context Precision và Context Recall (Độ chính xác và Độ phủ của ngữ cảnh):** Hai chỉ số này dùng để đánh giá trực tiếp chất lượng của module Retriever. **Context Precision** đo lường xem các ngữ cảnh được truy xuất có chứa thông tin thực sự liên quan đến câu hỏi hay không (giảm thiểu "nhiễu"). Trong khi đó, **Context Recall** đo lường xem liệu tất cả các thông tin cần thiết để trả lời câu hỏi đã được truy xuất đầy đủ hay chưa.

Việc cung cấp RAGAS sẽ cung cấp những số liệu định lượng, khách quan để đánh giá hiệu quả của hệ thống đã xây dựng. Các điểm số từ RAGAS sẽ là bằng chứng khoa học vững chắc để kết luận về mức độ thành công của mô hình trong việc giải quyết bài toán hỏi-đáp văn bản pháp luật một cách chính xác và đáng tin cậy.

#### 1.2.8. Các thư viện và công cụ hỗ trợ:

##### 1.2.8.1. Requests:

Thư viện **Requests** là một công cụ HTTP client mạnh mẽ và phổ biến bậc nhất trong hệ sinh thái Python, được sử dụng để gửi các loại yêu cầu HTTP một cách đơn giản. Trong luận văn này, thư viện Requests đóng vai trò then chốt trong quy trình thu thập dữ liệu, cụ thể là để thực hiện việc tải về các tập tin văn bản pháp luật dạng PDF từ Cổng Thông tin điện tử Chính phủ. Sau khi đã xác định được đường dẫn URL chính xác của tệp tin, một hàm chuyên dụng sẽ gọi đến Requests để gửi yêu cầu GET đến máy chủ. Đặc biệt, việc sử dụng tham số *steam=True* khi tải file cho phép xử lý các tệp có dung lượng lớn một cách hiệu quả, bằng cách đọc dữ liệu theo từng khối nhỏ thay vì tải toàn bộ về bộ nhớ cùng lúc, giúp tối ưu hóa tài nguyên hệ thống.

##### 1.2.8.2. Selenium:

**Selenium** là một bộ công cụ mã nguồn mở mạnh mẽ, được thiết kế để tự động hóa các trình duyệt web. Vai trò chính của nó là mô phỏng các hành động của người dùng như nhấp chuột, điền biểu mẫu, và điều hướng qua các trang. Trong khuôn khổ luận văn này, Selenium được sử dụng để điều khiển trình duyệt Chrome, tự động duyệt qua các trang danh sách văn bản trên Cổng Thông tin điện tử Chính phủ, xử lý việc chuyển trang (pagination), và truy cập vào từng trang chi tiết của mỗi văn bản pháp luật. Nó chịu trách nhiệm tải về toàn bộ mã nguồn HTML của mỗi trang, đảm bảo lấy được nội dung đầy đủ trước khi chuyển cho BeautifulSoup để tiến hành bóc tách dữ liệu.

##### 1.2.8.3. BeautifulSoup:

**BeautifulSoup** là một thư viện Python được thiết kế chuyên biệt cho việc bóc tách dữ liệu từ các tệp HTML và XML. Nhiệm vụ chính của nó là phân tích (parse) một tài liệu HTML thô và tạo ra một cây cấu trúc đối tượng, cho phép các lập trình viên điều hướng và tìm kiếm trong cây phân tích một cách dễ dàng. Trong quy trình thu thập dữ liệu của luận văn, sau khi Selenium đã tải về mã nguồn trang web, BeautifulSoup đảm nhận vai trò then chốt trong việc phân tích cây cấu trúc này. Nó được sử dụng để tìm kiếm và trích xuất các thông tin quan trọng như tiêu đề, số hiệu, ngày ban hành và đường dẫn tải tệp PDF bằng cách định vị các thẻ HTML cụ thể dựa trên thuộc tính của chúng, qua đó đơn giản hóa đáng kể quá trình trích xuất siêu dữ liệu một cách chính xác từ cấu trúc HTML phức tạp.

##### 1.2.8.4. PyMuPDF:

**PyMuPDF** là một thư viện Python hiệu suất cao, được sử dụng để truy cập và xử lý các tài liệu PDF. Trong luận văn này, nó đóng một vai trò thiết yếu trong giai đoạn tiền xử lý dữ liệu. Sau khi các văn bản pháp luật được thu thập dưới dạng tệp PDF, PyMuPDF được sử dụng để mở và trích xuất toàn bộ nội dung văn bản từ các tệp này một cách chính xác. Nhờ tốc độ xử lý nhanh và độ tin cậy cao, thư viện này đảm bảo rằng dữ liệu văn bản thô được cung cấp cho các bước tiếp theo như phân đoạn và nhúng vector là dữ liệu sạch và toàn vẹn, tạo nền tảng vững chắc cho toàn bộ hệ thống RAG.

##### 1.2.8.5. LangChain:

**LangChain** là một framework mã nguồn mở mạnh mẽ, được thiết kế để đơn giản hóa việc xây dựng các ứng dụng phức tạp dựa trên Mô hình Ngôn ngữ Lớn (LLMs). Nó hoạt động như một lớp điều phối trung tâm, cung cấp một bộ công cụ và các module được tiêu chuẩn hóa để kết nối các thành phần khác nhau một cách liền mạch. Mục tiêu chính của LangChain là cho phép các nhà phát triển tạo ra các ứng dụng nhận biết được ngữ cảnh và có khả năng suy luận, thay vì chỉ thực hiện các lệnh gọi API đơn lẻ đến LLM.

##### 1.2.8.6. RAGAS:

**RAGAS (Retrieval-Augmented Generation Assessment)** là một framework mã nguồn mở được thiết kế chuyên biệt để đánh giá và kiểm thử chất lượng của các hệ thống AI xây dựng theo kiến trúc RAG. Nó được sử dụng để đo lường hiệu suất của toàn bộ pipeline một cách khách quan thông qua các chỉ số quan trọng như **Faithfulness** (đo lường câu trả lời có trung thực với ngữ cảnh không) và **Answer Relevancy** (đo lường câu trả lời có đúng trọng tâm câu hỏi không). Điểm đặc biệt của RAGAS là khả năng hoạt động mà không cần đến bộ dữ liệu câu trả lời mẫu do con người tạo ra; thay vào đó, nó sử dụng chính các mô hình ngôn ngữ lớn để làm "giám khảo", giúp quá trình kiểm thử trở nên linh hoạt, tự động và có khả năng mở rộng cao.

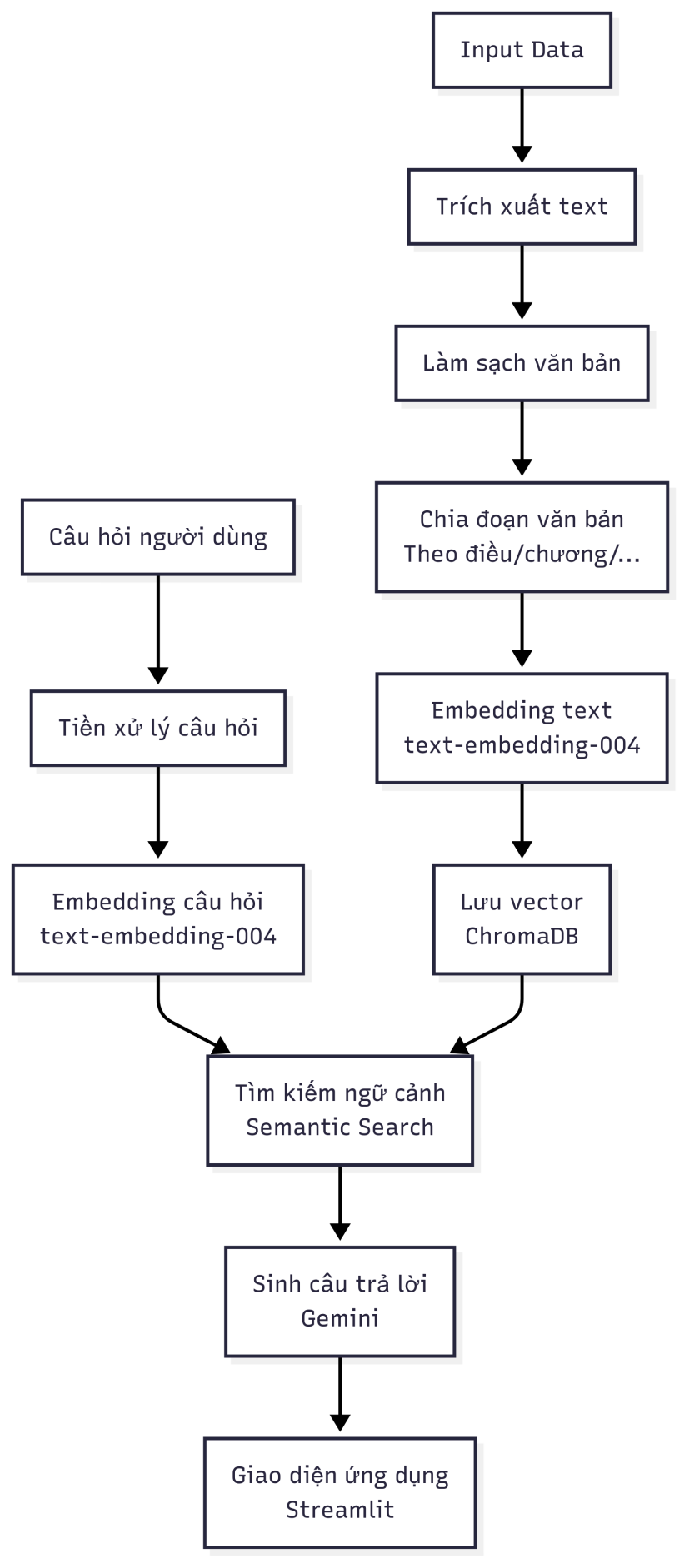
##### 1.2.8.7. Streamlit:

**Streamlit** là một framework mã nguồn mở miễn phí, được viết bằng Python, cho phép các nhà phát triển và nhà khoa học dữ liệu xây dựng và chia sẻ các ứng dụng web tương tác một cách nhanh chóng. Nó được sử dụng để biến các kịch bản phân tích dữ liệu và mô hình học máy thành các giao diện người dùng đồ họa (GUI) mà không yêu cầu kiến thức chuyên sâu về phát triển web front-end (HTML, CSS, JavaScript). Điểm mạnh lớn nhất của Streamlit là sự đơn giản; chỉ với vài dòng lệnh Python, người dùng có thể tạo ra các thành phần giao diện như thanh trượt, nút bấm, ô nhập liệu và hiển thị kết quả. Điều này giúp việc tạo một sản phẩm demo, một công cụ trực quan hóa, hay một giao diện đơn giản cho mô hình AI trở nên hiệu quả và tốn ít công sức.

## CHƯƠNG 2. THIẾT KẾ VÀ CÀI ĐẶT GIẢI PHÁP

### 2.1. Thiết kế hệ thống

Kiến trúc tổng thể của hệ thống được thiết kế dựa trên mô hình RAG tiên tiến. Mục tiêu của thiết kế này là xây dựng một trợ lý ảo không chỉ có khả năng sinh ngôn ngữ một cách tự nhiên mà còn phải đảm bảo các câu trả lời luôn được “neo” vào một nguồn kiến thức pháp lý xác thực, giảm thiểu tối đa hiện tượng “ảo giác” (hallucination). Như được minh họa trong hình 13, quy trình hoạt động của hệ thống được chia thành hai luồng chính, song song và bổ trợ cho nhau: Luồng xử lý và lập chỉ mục dữ liệu và luồng xử lý truy vấn và sinh câu trả lời.



Hình 13. Quy trình tổng thể của hệ thống

#### 2.1.1. Luồng xử lý và lập chỉ mục dữ liệu

Đây là luồng xử lý ngoại tuyến, có nhiệm vụ xây dựng nên “bộ não” hay kho kiến thức trợ lý ảo. Luồng này chỉ cần thực hiện một lần hoặc mỗi khi có dữ liệu mới cần cập nhật, và bao gồm các bước tuần tự sau:

1. Tiếp nhận dữ liệu đầu vào (Input Data): Dữ liệu đầu vào của hệ thống là tập hợp các văn bản quy phạm pháp luật bao gồm các loại chỉ thị, công điện và luật. Các văn bản này đã được thu thập từ trang Cổng Thông tin điện tử Chính phủ.
2. Trích xuất và làm sạch văn bản: Ở bước này, hệ thống sử dụng thư viện *PyMuPDF* để mở từng tệp PDF và trích xuất toàn bộ nội dung văn bản thơ. Sau khi trích xuất, một quy trình làm sạch cơ bản được áp dụng để loại bỏ những yếu tố gây nhiễu như thông tin thừa ở đầu trang (headers), chân trang (footers), số trang, và các khoảng trắng không cần thiết, đảm bảo dữ liệu văn bản là sạch và nhất quán.
3. Phân đoạn văn bản: Do các mô hình ngôn ngữ có giới hạn về độ dài ngữ cảnh đầu vào, việc đưa toàn bộ một văn bản dài hàng chục trang vào xử lý là không khả thi. Do đó, văn bản sau khi được làm sạch sẽ được phân chia thành các đoạn nhỏ hơn (chunks). Thay vì chia theo số lượng từ nhất định, hệ thống áp dụng phương pháp phân đoạn theo ngữ nghĩa, dựa trên cấu trúc logic của văn bản quy phạm pháp luật như Chương, Điều, Khoản. Cách tiếp cận này giúp mỗi đoạn văn bản chứa một ý nghĩa trọn vẹn và độc lập, làm tăng độ chính xác cho bước tìm kiếm về sau.
4. Tạo vector nhúng (vector embedding): Mỗi đoạn văn bản (chunk) sau khi được phân chia sẽ được đưa vào mô hình nhúng text-embedding-004 của Google. Mô hình này sẽ chuyển đổi mỗi đoạn văn bản thành một vector, tức một chuỗi các con số trong không gian đa chiều. Vector này mã hóa ngữ nghĩa của đoạn văn bản, giúp máy tính có thể “hiểu” và so sánh ý nghĩa giữa các đoạn văn với nhau.
5. Lưu trự vào cơ sở dữ liệu vector: Các vector nhúng cùng với nội dung văn bản gốc và các siêu dữ liệu (metadata) liên quan sẽ được lưu trữ và lập chỉ mục trong cơ sở dữ liệu vector ChromaDB. ChromaDB được tối ưu cho việc tìm kiếm tương đồng trên các vector với tốc độ cao. Tại thời điểm này, kho kiến thức của trợ lý ảo đã được xây dựng xong và sẵn sàng cho việc tra cứu.

#### 2.1.2. Luồng xử lý truy vấn và sinh câu trả lời:

Đây là luồng xử lý “trực tuyến”, được kích hoạt mỗi khi người dùng đặt một câu hỏi.

1. Tiếp nhận và tiền xử lý câu hỏi: Người dùng nhập câu hỏi bằng ngôn ngữ tự nhiên thông qua giao diện ứng dụng Streamlit. Câu hỏi này sẽ được tiền xử lý cơ bản để loại bỏ các yếu tố không cần thiết.
2. Nhúng câu hỏi: Câu hỏi đã được xử lý sau đó được đưa vào chính mô hình nhúng **text-embedding-004** đã dùng ở luồng trước. Việc sử dụng cùng một mô hình là bắt buộc để đảm bảo vector của câu hỏi và vector của các đoạn văn bản nằm trong cùng một không gian ngữ nghĩa.
3. Tìm kiếm ngữ cảnh (Semantic Search): Vector của câu hỏi được sử dụng để thực hiện một cuộc tìm kiếm tương đồng trong cơ sở dữ liệu **ChromaDB**. Hệ thống sẽ tìm và trả về một số lượng k (ví dụ: k=3) đoạn văn bản có vector gần với vector câu hỏi nhất. Đây là những đoạn văn bản được cho là có nội dung liên quan nhất để trả lời câu hỏi.
4. Sinh câu trả lời: Các đoạn văn bản (ngữ cảnh) được truy xuất ở bước trên, cùng với câu hỏi gốc của người dùng, sẽ được kết hợp lại thành một câu lệnh (prompt) hoàn chỉnh. Câu lệnh này sau đó được gửi đến mô hình ngôn ngữ lớn **Gemini**. Dựa trên prompt này, Gemini sẽ tổng hợp thông tin, suy luận và tạo ra một câu trả lời cuối cùng bằng ngôn ngữ tự nhiên.
5. Hiển thị kết quả: Câu trả lời do Gemini tạo ra được gửi trở lại **giao diện Streamlit** và hiển thị cho người dùng.

### 2.2. Cài đặt giải pháp

#### 2.2.1. Thu thập dữ liệu

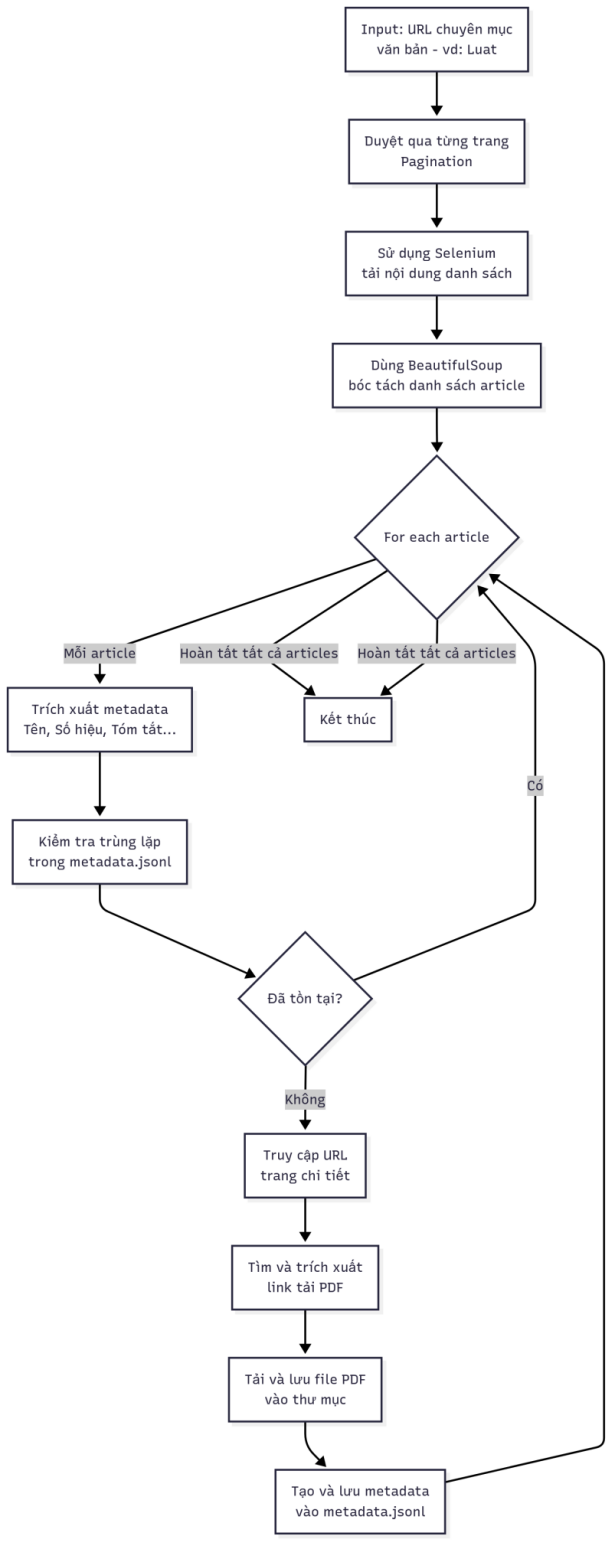


Hình 14. Trang Cổng Thông tin điện tử Chính phủ (Nguồn: https://congbao.chinhphu.vn/)

Nguồn dữ liệu được thu thập chủ yếu tại Cổng Thông tin điện tử Chính phủ, cụ thể là chuyên mục Công Báo (https://congbao.chinhphu.vn/). Đây là kênh thông tin điện tử chính thức của Chính phủ Việt Nam, đóng vai trò công khai các loại văn bản quy phạm pháp luật và văn bản hành chính do các cơ quan nhà nước ban hành. Trang Công Báo đảm bảo tính minh bạch và khả năng tiếp cận thông tin pháp luật cho người dân, doanh nghiệp và các tổ chức. Các văn bản đăng tải trên đây có độ tin cậy cao, phản ánh các quyết sách, chỉ đạo và quy định hiện hành của nhà nước. Một đặc điểm kỹ thuật quan trọng của nguồn này là hầu hết các văn bản được cung cấp dưới dạng tập tin PDF có chứa văn bản (text-based PDF), cho phép việc trích xuất nội dung text một cách trực tiếp và hiệu quả bằng các công cụ phần mềm.

Từ Cổng Thông tin điện tử Chính phủ - Công Báo, tôi đã thu thập được số lượng khá lớn tập tin văn bản (tổng cộng … tập tin) tính đến ngày 01/07/2025. Hầu hết các tập tin này đều ở định dạng PDF text-based, thuận lợi cho quá trình xử lý và phân tích nội dung. Dữ liệu loại văn bản “*Chỉ thị”*, cụ thể như sau:

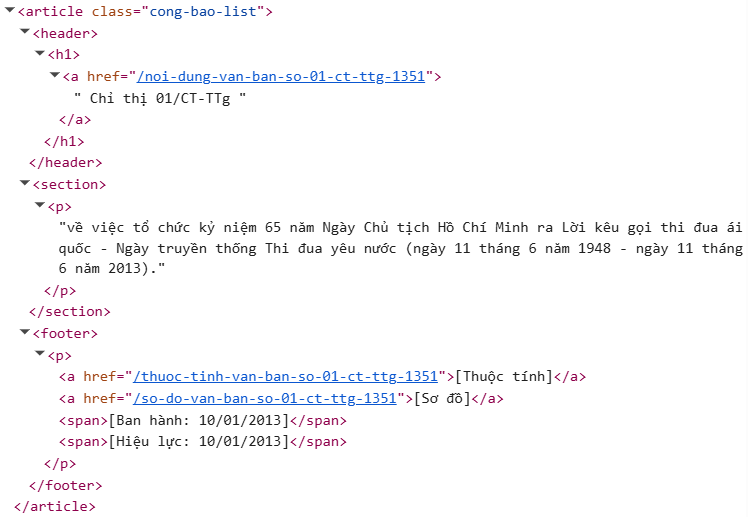
* Chỉ thị (40 tập tin): Thường do Thủ tướng Chính phủ hoặc các Bộ trưởng ban hành để chỉ đạo, đôn đốc việc thực hiện các chủ trương, chính sách, pháp luật hoặc các nhiệm vụ cụ thể trong phạm vị quản lý của mình.



Hình 15. Quy trình cào dữ liệu

Trước khi tiến hành thu thập dữ liệu, một bước phân tích cấu trúc HTML của trang nguồn đã được thực hiện nhằm xác định các mẫu cấu trúc ổn định để bóc tách dữ liệu. Kết quả cho thấy trang web trình bày mỗi văn bản như một mục tin riêng biệt, đồng nhất và các cấu trúc mã HTML của các trang nguồn chứa các loại văn bản đều giống nhau. Mỗi mục tin được bọc hoàn toàn trong một thẻ HTML là <article> với thuộc tính class=“cong-bao-list”. Thẻ <article> này đóng vai trò là một “container” chính, chứa toàn bộ các thông tin liên quan đến một văn bản cục thể như tiêu đề, tóm tắt và các siêu dữ liệu khác. Việc xác định container chung này là bước đầu tiên và quan trọng nhất để có thể duyệt qua và xử lý từng văn bản một cách độc lập.

Bên trong mỗi container <article>, thông tin được tổ chức một cách có cấu trúc và phân cấp rõ ràng. Cụ thể **tên văn bản** và **đường dẫn đến trang chi tiết** được tìm thấy trong thẻ <a> nằm bên trong thẻ <h1> của phần <header>. Phần **trích yếu** được đặt trong một thẻ <p> bên trong thẻ <section>. Cuối cùng, các thông tin về **ngày ban hành** và **ngày hiệu lực** được chứa trong thẻ <span> riêng biệt, nằm trong phần <footer> của <article>. Bằng cách nhận diện các quy tắc cấu trúc HTML cố định và có tính lặp lại này, việc lập trình để điều hướng một cách chính xác trong cây DOM và trích xuất từng mảng dữ liệu một cách đáng tin cậy.



Ngày ban hành và   
ngày hiệu lực của văn bản

Trích yếu của văn bản

Tên văn bản và đường dẫn đến trang tải văn bản

Hình 16. Cấu trúc HTML của trang chứa loại văn bản “Chỉ thị”.

Trong hình là ví dụ mã nguồn của trang chứa chỉ thị 01/CT-TTg năm 2013.

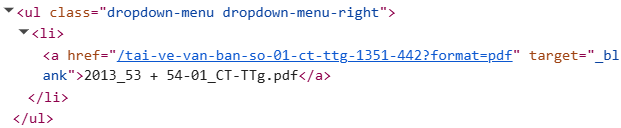
Để xây dựng cơ sở dữ liệu cho hệ thống, một quy trình thu thập dữ liệu được thực hiện tự động từ Cổng Thông tin điện tử Chính phủ. Quy trình này được thực hiện bằng một đoạn mã được lập trình bằng ngôn ngữ Python, kết hợp các thư viện chuyên dụng để đảm bảo hiệu suất và độ tin cậy.

Đầu tiên, một môi trường làm việc được thiết lập bằng thư viện **Selenium** cùng với **WebDriver Manager**, giúp tự động hóa việc điều khiển trình duyệt Chrome. Trình duyệt được cấu hình để hoạt động ở chế độ nền (headless), cho phép đoạn mã chạy hiệu quả mà không cần hiển thị giao diện đồ họa. Hai thực thể (instance) trình duyệt riêng biệt được khởi tạo đảm nhận các nhiệm vụ khác nhau: **driver\_1** đảm nhận việc duyệt qua các trang danh sách văn bản, trong khi **driver\_2** được dành riêng để truy cập vào từng trang chi tiết của mỗi văn bản. Cách này giúp quản lý các luồng điều hướng một cách độc lập và hiệu quả.

Quy trình chính được thực hiện trong một vòng lặp, tự động duyệt qua tất cả các trang của một chuyên mục văn bản cụ thể. Tại mỗi trang danh sách, **driver\_1** sẽ tải về toàn bộ mã nguồn HTML. Sau đó, mã nguồn này được đưa vào thư viện **BeautifulSoup** để tiến hành phân tích và tạo ra một cây cấu trúc dữ liệu. Dựa trên cấu trúc này, tôi đã lập trình để tìm kiếm và trích xuất tất cả các thẻ <article> có thuộc tính class=“cong-bao-list”, mỗi thẻ này tương ứng với thông tin của một văn bản.

Bên trong vòng lặp duyệt qua từng mục <article> đã tìm thấy, một quy trình bóc tách và chuẩn hóa dữ liệu chi tiết được thực hiện ở cấp độ mã nguồn thông qua **driver\_2**. Mỗi mục được lưu vào biến **item**. Để lấy **Tên văn bản**, tôi đã sử dụng phương thức item.find(“header h1 a”).get\_text(strip=True) để điều hướng chính xác qua cây cấu trúc HTML và trích xuất nội dung text thô. Ngay sau đó, biểu thức chính quy r“\s+” được áp dụng thông qua hàm re.sub() để thay thế một hoặc nhiều ký tự khoảng trắng liên tiếp bằng một khoảng trắng duy nhất, nhằm chuẩn hóa và làm sạch tiêu đề. Đồng thời, **đường dẫn** (URL) của trang chi tiết được lấy ra bằng cách truy cập trực tiếp vào thuộc tính *href* của thẻ <a>.

Tuy nhiên, các trường dữ liệu khác đòi hỏi xử lý phức tạp hơn. Do **Số hiệu** của mỗi văn bản nằm trong trường **Tên văn bản** được lưu trước đó, nên tôi đã xây dựng hàm get\_so\_hieu() để tách **Số hiệu** ra khỏi chuỗi tên văn bản. Hàm này sử dụng biểu thức chính quy r“\d” để thực thi lệnh re.search(), với mục đích tìm kiếm vị trí chỉ mục (index) của ký tự số đầu tiên xuất hiện trong chuỗi. Do số hiệu văn bản luôn bắt đầu là một con số, hàm sẽ thực hiện cắt chuỗi (string slicing) từ chỉ số đầu tiên đó cho đến hết chuỗi, trả về phần số hiệu đã được tách riêng một cách chính xác. Ví dụ, với tên văn bản là “Chỉ thị 01/CT-TTg” thì số hiệu của văn bản sau khi đưa vào hàm get\_so\_hieu() sẽ là “01/CT-TTg”. Tương tự, để xử lý **Ngày ban hành** và **Ngày hiệu lực**, đoạn mã trước hết sẽ tìm tất cả thẻ <span> trong phần <footer> của mục tin, sau đó lặp qua từng thẻ và kiểm tra nội dung text của nó có chứa từ khóa “ban hành” hay “hiệu lực” hay không. Khi đã xác định đúng thẻ, nội dung text thô sẽ được đưa vào hàm get\_ngaybh\_ngayhl(). Hàm này cũng áp dụng biểu thức r“\d” để xác định điểm bắt đầu của chuỗi ngày tháng, sau đó thực hiện cắt chuỗi đến kí tự áp chót nhằm loại bỏ các văn bản mô tả ở phía trước và các kí tự không cần thiết ở phía sau. Ví dụ như trong hình , có hai thẻ <span> trong phần <footer>, kết quả sau khi đưa nội dung text thô vào hàm get\_ngaybh\_ngayhl() thì ngày ban hành sẽ là “10/01/2013” và ngày hiệu lực là “10/01/2013”.



Hình 17. Mã nguồn trang chi tiết chứa đường dẫn tải văn bản trong thẻ <ul>

Tại trang chi tiết của văn bản, tiếp tục phân tích cấu trúc HTML để xác định vị trí của liên kết tải về. Cụ thể, nó sẽ tìm đến thẻ <ul> có thuộc tính class=“dropdown-menu” và duyệt qua các thẻ con <li> bên trong. Sau đó, đoạn mã sẽ bắt đầu xem các thẻ <a> trong mỗi thẻ <li> để tìm ra thẻ có thuộc tính *href* chứa chuỗi “format=pdf”. Khi đã có được URL chính xác của tập tin PDF, hàm download\_pdf() sẽ được gọi. Hàm này được cài đặt để sử dụng thư viện **requests** với tham số **stream=True**, cho phép tải các tập tin có dung lượng lớn một cách hiệu quả bằng cách đọc dữ liệu theo từng khối nhỏ (chunk) thay vì tải toàn bộ về bộ nhớ cùng lúc. Tập tin PDF sau đó được lưu vào thư mục cục bộ đã được cấu hình trước.

Song song với việc lưu tập tin PDF, một từ điển (dictionary) chứa toàn bộ siêu dữ liệu của văn bản được tạo ra. Từ điển này sau đó được chuyển cho hàm append\_metadata() để ghi vào tệp metadata.jsonl. Hàm này mở tệp ở chế độ ghi nối tiếp với mã hóa utf-8 và sử dụng lệnh json.dump() để chuyển đổi từ điển thành một chuỗi JSON. Mỗi bản ghi được ghi trên một dòng riêng biệt, theo đúng chuẩn định dạng JSON Lines. Cách lưu trữ này không chỉ đảm bảo tính toàn vẹn của dữ liệu mà còn cực kỳ hiệu quả khi cần bổ sung thông tin mới vào một tệp lớn mà không cần phải đọc và ghi lại toàn bộ tệp.

Mỗi tập tin metadata lưu mỗi loại văn bản với các trường dữ liệu sau:

* Tên tập tin: Tên của tập tin văn bản đã tải. Vì tên văn bản và tên tập tin không giống nhau nên sẽ lưu trữ tên tập tin để thuận tiện cho việc tìm kiếm.
* Tên văn bản: Tên của văn bản đã tải.
* Số hiệu: Số hiệu của văn bản sau khi đã xử lý.
* Ngày ban hành: Ngày ban hành của văn bản.
* Ngày hiệu lực: Ngày có hiệu lực chính thức của văn bản.
* Loại văn bản: Loại của văn bản. Vì các văn bản được xử lý có ba loại khác nhau là chỉ thị, công điện, luật nên việc lưu loại văn bản sẽ giúp chương trình biết được đang xử lý loại văn bản nào và thực hiện quy trình gì.
* Nguồn: Nơi tải văn bản.
* Trạng thái tải: “Tải thành công” và “Không tải được file”. Trường hợp “Không tải được file” là do trong quá trình cào dữ liệu có thể đường truyền không ổn định ảnh hưởng đến việc tài văn bản nên nếu trường hợp này xảy ra. Tôi sẽ tải văn bản này thủ công để thu được toàn bộ phần văn bản cần thu thập.

Cuối cùng, sau khi hoàn thành quá trình thu thập dữ liệu, tôi đã thu thập tổng cộng 48 tập tin thuộc loại văn bản “Chỉ thị”.

#### 2.2.2. Tiền xử lý dữ liệu

## CHƯƠNG 3. KIỂM THỬ VÀ ĐÁNH GIÁ

### 3.1. Giao diện sản phẩm (nếu có)

### 3.2. Kết quả thực nghiệm

### 3.3. Thảo luận về kết quả đạt được

# III. PHẦN KẾT LUẬN

**1. Kết quả đạt được**

**2. Hướng phát triển**

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N, Kaiser, L., & Polosukhin, I. (2017). ***Attention is All You Need***. arXiv preprint arXiv:1706.03762. |
| [2] | Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktashcel, Sebastian Riedel, Douwe Kiela, (2020). ***Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks***. arXiv preprint:2005.11401 |
| [3] | CTU-LinguTechies (2023) ***VN-Law-Advisor: Hệ thống hỏi đáp pháp luật dựa trên mô hình ngôn ngữ lớn.*** |
| [4] |  |
| [5] |  |
| [6] |  |
| [7] |  |
| [8] |  |
| [9] |  |
| [10] |  |
| [11] |  |

# PHỤ LỤC