

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐẶNG NGUYỄN QUANG HUY - 21133036  
HUỲNH GIA HÂN – 21133031  
NGUYỄN TRỌNG DŨNG - 21133021**

Đề Tài:

**TÌM HIỂU HỆ THỐNG TRẢ LỜI CÂU HỎI VỀ  
KIẾN THỨC PHÁP LUẬT DỰA TRÊN DỮ LIỆU  
PHÁP LUẬT VIỆT NAM VÀ BỘ PHÁP ĐIỀN**

**TIỂU LUẬN CHUYÊN NGÀNH NGÀNH KTDL**

**GIÁO VIÊN HƯỚNG DẪN**

**ThS. TRẦN TRỌNG BÌNH**

**Tp. Hồ Chí Minh, tháng 12 năm 2024**

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM  
KHOA CÔNG NGHỆ THÔNG TIN**

\*\*\*\*\*

**CỘNG HÒA XHCN VIỆT NAM  
Độc lập – Tự do – Hạnh phúc**

\*\*\*\*\*

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên 1: **Đặng Nguyễn Quang Huy** MSSV 1: **21133036**

Họ và tên Sinh viên 2: **Huỳnh Gia Hân** MSSV 2: **21133031**

Họ và tên Sinh viên 3: **Nguyễn Trọng Dũng** MSSV 3: **21133021**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Tìm hiểu hệ thống trả lời câu hỏi về kiến thức pháp luật dựa trên dữ liệu pháp luật việt nam và bộ pháp điển**

Họ và tên giáo viên hướng dẫn: **ThS. Trần Trọng Bình**

**NHẬN XÉT**

1. Về nội dung đề tài và khối lượng thực hiện:

2. Ưu điểm:

3. Khuyết điểm:

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

## LỜI CẢM ƠN

Đầu tiên, nhóm chúng em xin bày tỏ lòng biết ơn sâu sắc đến Ban Giám hiệu Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh đã tạo điều kiện học tập và nghiên cứu trong môi trường chất lượng, giúp chúng em có nền tảng vững chắc để thực hiện Tiểu Luận Chuyên Ngành.

Chúng em cũng chân thành cảm ơn Ban Chủ nhiệm khoa Công nghệ Thông tin cùng các thầy cô trong khoa, những người đã tận tình chỉ dạy và trang bị cho chúng em những kiến thức quý giá, là hành trang quan trọng trong suốt những năm tháng học tập tại trường. Sự tận tâm của các thầy cô là nguồn cảm hứng lớn lao để chúng em cố gắng không ngừng.

Đặc biệt, nhóm xin gửi lời cảm ơn sâu sắc đến thầy Trần Trọng Bình, giảng viên phụ trách hướng dẫn Tiểu Luận Chuyên Ngành. Trong suốt quá trình thực hiện, nhóm đã nhận được sự chỉ bảo tận tình và hỗ trợ quý báu từ thầy. Thầy không chỉ cung cấp tài liệu hữu ích mà còn đóng góp nhiều ý kiến chuyên môn giúp nhóm hoàn thiện nội dung tiểu luận. Sự nhiệt huyết và tinh thần trách nhiệm của thầy trong việc giảng dạy đã truyền cảm hứng mạnh mẽ, giúp nhóm có thêm động lực để vượt qua những khó khăn.

Dựa trên những kiến thức được thầy truyền đạt và sự tìm hiểu thêm từ các tài liệu, công cụ mới, nhóm đã nỗ lực hết mình để hoàn thành Tiểu Luận Chuyên Ngành. Tuy nhiên, bài tiểu luận chắc chắn vẫn còn những hạn chế và sai sót. Nhóm rất mong nhận được những ý kiến đóng góp quý báu từ thầy để cải thiện, hoàn thiện hơn trong các nghiên cứu và dự án tương lai.

Nhóm chúng em xin chân thành cảm ơn!

Trường ĐH Sư Phạm Kỹ Thuật TP.HCM

Khoa: Công nghệ thông tin

### ĐỀ CƯƠNG TIỂU LUẬN CHUYÊN NGÀNH

Họ và tên Sinh viên 1: **Đặng Nguyễn Quang Huy** MSSV 1: **21133036**

Họ và tên Sinh viên 2: **Huỳnh Gia Hân** MSSV 2: **21133031**

Họ và tên Sinh viên 3: **Nguyễn Trọng Dũng** MSSV 3: **21133021**

Chuyên ngành: **Kỹ thuật dữ liệu**

Tên đề tài: **Tìm hiểu hệ thống trả lời câu hỏi về kiến thức pháp luật dựa trên dữ liệu pháp luật việt nam và bộ pháp điển**

Họ và tên giáo viên hướng dẫn: **ThS. Trần Trọng Bình**

**Nhiệm vụ của tiểu luận chuyên ngành:**

1. Lý thuyết:

2. Thực hành:

### KẾ HOẠCH THỰC HIỆN

STT	Thời gian	Công việc	Ghi chú
1	8/9-15/9	Tìm hiểu về kiến trúc của Transfomer Tìm hiểu về mô hình Bert của Google được xây dựng dựa trên kiến trúc transfomer	
2	16/9-/23/9	Tìm hiểu về các phương pháp làm chatbot Tìm hiểu bài toán trích xuất thông tin từ đoạn văn	
3	24/9-1/10	Tìm hiểu công nghệ VectorDataBase Tìm hiểu về RAG được áp dụng cho bài toán truy xuất thông tin,các phương pháp RAG bao gồm : Navie RAG và Advanced RAG Bắt đầu ghi các lý thuyết tổng quan cho báo cáo	
4	2/10-9/10	Tìm hiểu về kỹ thuật Lora Fine-Tuning và Full Fine-Tuning	

		Thu thập,xử lý,tổ chức dữ liệu cho bài toán trích xuất thông tin bằng mô hình Bert Bổ sung các lý thuyết vào báo cáo.	
5	10/10-17/10	Áp dụng 2 kỹ thuật Fine-Tuning cho Bert là Lora Fine-Tuning và Full-Fine-Tuning để giải quyết bài toán trích xuất thông tin Nghiên cứu về các loại độ đo để đánh giá cho bài toán trích xuất thông tin Bổ sung các lý thuyết vào báo cáo.	
6	18/10-25/10	Tìm hiểu mô hình T5 Thu thập,xử lý,tổ chức dữ liệu cho bài toán tóm tắt văn bản bằng T5 Bổ sung các lý thuyết vào báo cáo.	
7	26/10-2/11	Áp dụng kỹ thuật Full Fine-Tuning cho mô hình T5 để giải quyết nhiệm vụ tóm tắt văn bản Cải thiện kỹ thuật chunking và sinh thêm các câu hỏi tương đồng để truy xuất thêm các ngữ cảnh liên quan(RAG) Bổ sung các lý thuyết vào báo cáo.	
8	3/11-10/11	Tiếp tục xử lý dữ liệu và điều chỉnh các siêu tham số cho bài toán trích xuất thông tin của mô hình Bert Tiếp tục xử lý dữ liệu và điều chỉnh các siêu tham số cho bài toán tóm tắt văn bản của mô hình T5 Tìm hiểu các độ đo cho bài toán tóm tắt văn bản Bổ sung các lý thuyết vào báo cáo.	
9	11/11-18/11	Kết hợp giữa search similarity và filter theo các metadata để truy xuất đúng hơn cho các ngữ cảnh phức tạp hơn	

		<p>Đánh giá lại các tài liệu(re-rank) bằng cách kết hợp giữa độ đo cosine và bm25</p> <p>Tích hợp API gemini để trả lời bằng cách suy luận và trích xuất các thông tin liên quan</p> <p>Bổ sung các lý thuyết vào báo cáo</p>	
10	19/11-26/11	<p>Tích hợp API bên thứ 3 là cohore để đánh giá lại các tài liệu(re-rank) liên quan nhất đến câu hỏi</p> <p>Xây dựng giao diện người dùng phù hợp với bài toán chatbot.</p> <p>Kết hợp bài toán RAG, trích xuất thông tin (BERT), và tóm tắt văn bản (T5) vào website sử dụng Flask.</p> <p>Bổ sung các lý thuyết vào báo cáo.</p>	
11	27/11-4/12	<p>Bổ sung thêm một số chức năng cho website bao gồm: đoạn chat mới, đổi mô hình, lịch sử chat, xoá lịch sử chat, hỏi đáp với chatbot</p> <p>Hoàn thiện báo cáo tiêu luận chuyên ngành.</p>	

**Ngày 07 tháng 12 năm 2024**

**Ý kiến của giáo viên hướng dẫn**

**Người viết đề cương**

**Trần Trọng Bình**

**Đặng Nguyễn Quang Huy**

**Huỳnh Gia Hân**

**Nguyễn Trọng Dũng**

## MỤC LỤC

MỞ ĐẦU .....	1
1. VẤN ĐỀ ĐẶT RA.....	1
2. MỤC TIÊU NGHIÊN CỨU .....	2
3. PHẠM VI NGHIÊN CỨU.....	2
4. TẦM QUAN TRỌNG CỦA ĐỀ TÀI.....	3
5. PHƯƠNG PHÁP NGHIÊN CỨU .....	3
CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN TRẢ LỜI CÂU HỎI .....	5
1.1. GIỚI THIỆU VỀ BÀI TOÁN TRẢ LỜI CÂU HỎI .....	5
1.2. XỬ LÝ NGÔN NGỮ TỰ NHIÊN (NLP) TRONG BÀI TOÁN TRẢ LỜI CÂU HỎI .....	5
1.2.1. Các phương pháp xử lý ngôn ngữ trong NLP .....	6
1.2.2. Ứng dụng của NLP trong bài toán trả lời câu hỏi .....	6
1.2.3. Những thách thức trong NLP cho bài toán trả lời câu hỏi.....	7
1.2.4. Những thách thức khi xây dựng bài toán trả lời câu hỏi .....	7
1.3. CÁC PHƯƠNG PHÁP XÂY DỰNG BÀI TOÁN TRẢ LỜI CÂU HỎI .....	7
1.3.1. Dựa trên kịch bản.....	7
1.3.2. Dựa trên biểu diễn tri thức .....	9
1.3.3. Dựa trên các mô hình LLM .....	10
1.4. CƠ CHẾ HOẠT ĐỘNG CỦA BÀI TOÁN TRẢ LỜI CÂU HỎI .....	10
CHƯƠNG 2: CÁC MÔ HÌNH LARGE LANGUAGE MODEL CHO BÀI TOÁN TRẢ LỜI CÂU HỎI .....	11
2.1. TỔNG QUAN VỀ TRANSFORMER.....	11
2.1.1. Giới thiệu .....	11
2.1.2. Kiến trúc Transformer .....	12
2.1.3. Kiến trúc và cách thức hoạt động .....	13

2.1.3.1. Input Embedding .....	13
2.1.3.2. Positional Encoding .....	13
2.1.3.3. Encoder .....	14
2.1.3.4. Decoder .....	15
2.1.3.5. Cơ chế Attention .....	18
2.1.3.6. Hoạt động của Scaled Dot Product Attention .....	20
2.1.3.7. Thành phần Multi-Head Attention .....	22
2.1.3.8. Thành phần Add và Normalize .....	23
2.1.3.9. Thành phần Feed Forward .....	25
2.1.3.10. Masked Multi-Head Attention .....	25
2.2. GIỚI THIỆU VỀ MÔ HÌNH BERT .....	26
2.2.1. Giới thiệu sơ lược về BERT .....	26
2.2.2. Quá trình pre-train BERT .....	27
2.2.3. Cấu tạo của BERT .....	27
2.3. GIỚI THIỆU VỀ MÔ HÌNH T5 .....	29
2.3.1 Giới thiệu sơ lược về T5 .....	29
2.3.2. Quá trình pre-train T5 .....	29
2.3.3. Cấu tạo của T5 .....	30
2.4. SƠ LUẬC VỀ PRE-TRAIN VÀ FINE-TUNING .....	31
2.5. CÁC KỸ THUẬT FINE-TUNING .....	33
2.5.1. Full Fine-Tuning .....	34
2.5.2. Lora Fine-tuning .....	34
2.6. CÁC PHƯƠNG PHÁP ĐÁNH GIÁ MÔ HÌNH VÀ ĐÁNH GIÁ TÀI LIỆU	37
2.6.1. Độ đo F1-score .....	37
2.6.2. Độ đo Exact Match (EM) .....	39

2.6.3. Độ đo ROUGE [18] .....	39
2.6.4. Độ đo BM25 .....	41
2.6.5. Độ đo Cosine Similarity .....	42
<b>CHƯƠNG 3: TỔNG QUAN VỀ RETRIEVAL-AUGMENTED GENERATION ..</b>	<b>44</b>
3.1. KHÁI NIỆM VỀ RETRIEVAL - AUGMENTED GENERATION (RAG) ...	44
3.2. QUY TRÌNH HOẠT ĐỘNG CƠ BẢN.....	44
3.3. CÁC MÔ HÌNH RAG .....	45
3.3.1. Naive RAG .....	46
3.3.2. Advanced RAG.....	47
3.3.3. Ưu và nhược điểm của các mô hình RAG.....	48
3.3.3.1. Naive RAG .....	48
3.3.3.2. Advanced RAG.....	48
3.4. VECTOR DATABASE .....	49
3.4.1. Khái niệm về Vector Database .....	49
3.4.2. Khái niệm về Vector và Embedding.....	49
3.4.2.1. Vector .....	49
3.4.2.2. Embedding .....	49
3.4.3. Ứng dụng của Vector Database .....	49
3.5. PROMPTING .....	50
3.5.1. Giới thiệu về Prompt .....	50
3.5.2. Các cách tạo Prompt.....	50
3.5.2.1. Prompt chỉ dẫn trực tiếp (Zero-shot Prompt) .....	50
3.5.2.2. Prompt đưa ra vài ví dụ (Few-shot Prompt) .....	50
3.5.2.3. Prompt đưa ra vai trò cụ thể (Role-based Prompt) .....	51
<b>CHƯƠNG 4 : XÂY DỰNG MÔ HÌNH LLM VÀ TRUY XUẤT TĂNG CƯỜNG (RAG) CHO BÀI TOÁN TRẢ LỜI CÂU HỎI .....</b>	<b>52</b>

4.1. BÀI TOÁN TRÍCH XUẤT CÂU TRẢ LỜI TỪ ĐOẠN VĂN BẢN.....	52
4.1.1. Giới thiệu về bài toán .....	52
4.1.1.1. Sơ lược về bài toán trích xuất câu trả lời từ đoạn văn .....	52
4.1.1.2. Mục tiêu hướng tới .....	52
4.1.2. Xây dựng bộ dữ liệu cho bài toán Machine Reading Comprehension.....	52
4.1.2.1. Quá trình thu thập dữ liệu .....	52
4.1.2.2. Tổ chức dữ liệu .....	54
4.1.3. Quá trình tiền xử lý dữ liệu và huấn luyện mô hình.....	55
4.1.3.1. Quá trình tiền xử lý dữ liệu cho việc fine-tuning .....	55
4.1.3.2. Quá trình huấn luyện mô hình BERT cho bài toán trả lời câu hỏi ....	59
4.1.4. Đánh giá mô hình cho bài toán trích xuất thông tin .....	67
4.1.4.1. Phương pháp đánh giá .....	67
4.1.4.2. Kết quả thu được.....	68
4.2. BÀI TOÁN TÓM TẮT VĂN BẢN .....	71
4.2.1. Giới thiệu về bài toán .....	71
4.2.2. Xây dựng bộ dữ liệu cho bài toán tóm tắt văn bản.....	71
4.2.2.1. Quá trình thu thập dữ liệu .....	71
4.2.2.2. Tổ chức dữ liệu .....	71
4.2.3. Quá trình tiền xử lý dữ liệu và huấn luyện mô hình.....	73
4.2.3.1. Quá trình tiền xử lý dữ liệu cho việc fine-tuning .....	73
4.2.3.2. Quá trình huấn luyện mô hình T5 cho bài toán tóm tắt văn bản .....	74
4.2.4. Đánh giá mô hình cho bài toán tóm tắt văn bản .....	79
4.2.4.1. Phương pháp đánh giá .....	79
4.2.4.2. Kết quả thu được.....	79
4.3. BÀI TOÁN TRUY XUẤT TĂNG CUỐNG.....	80

4.3.1. Giới thiệu về bài toán .....	80
4.3.2. Xây dựng bộ dữ liệu cho bài toán truy xuất tăng cường .....	81
4.3.2.1. Quá trình thu thập dữ liệu.....	81
4.3.2.2. Quá trình tiền xử lý dữ liệu.....	81
4.3.2.3. Quá trình tổ chức dữ liệu .....	82
4.3.2.4. Quá trình đưa dữ liệu vào vector database .....	85
4.3.4. Quá trình thực hiện truy xuất tăng cường.....	87
4.3.4.1. Giai đoạn sinh thêm câu hỏi tương tự.....	87
4.3.4.2. Giai đoạn truy xuất .....	88
4.3.4.3. Giai đoạn phản hồi.....	89
4.4. ÚNG DỤNG RAG VÀ CÁC MÔ HÌNH ĐÃ FINE-TUNING: T5, BERT VÀO WEBSITE HỎI ĐÁP PHÁP LUẬT .....	90
4.4.1. Ý tưởng sản phẩm.....	90
4.4.2. Công nghệ sử dụng cho website .....	91
4.4.3. Các chức năng và màn hình giao diện .....	91
4.4.3.1. Tạo đoạn chat mới .....	91
4.4.3.2. Đổi mô hình .....	92
4.4.3.3. Hỏi đáp với chatbot .....	93
4.4.3.4. Chức năng lịch sử chat .....	94
4.4.3.5. Chức năng xóa lịch sử chat.....	94
CHƯƠNG 5: KẾT LUẬN .....	96
5.1. KẾT QUẢ ĐẠT ĐƯỢC .....	96
5.2. HẠN CHẾ .....	96
5.3. HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI.....	97
TÀI LIỆU THAM KHẢO .....	99

## **DANH MỤC HÌNH ẢNH**

Hình 1.1. Mô tả hoạt động của Scripted Chatbot[8].....	8
Hình 1.2. Mô tả hoạt động của Clicking Bot[8] .....	9
Hình 2.1. Mô hình kiến trúc Transformer[1].....	12
Hình 2.2. Encoders Input[1] .....	14
Hình 2.3. Các thành phần của Encoder[1].....	15
Hình 2.4. Quá trình xử lý một câu đối với Encoder và Decoder[1] .....	15
Hình 2.5. Các thành phần của Decoder[1].....	16
Hình 2.6. Cơ chế self-Attention[1] .....	19
Hình 2.7. Cơ chế self-Attention trong transformer[1] .....	19
Hình 2.9. Scaled Dot-Product Attention[1] .....	22
Hình 2.10. Cơ chế Multi-head Attention[1] .....	22
Hình 2.11. Biểu diễn trực quan cho 2 bước add và normalize[1] .....	24
Hình 2.12. Quá trình pre-train và fine-tuning của mô hình BERT[9] .....	27
Hình 2.13. Biểu diễn đầu vào của BERT[9].....	28
Hình 2.14. Kiến trúc mô hình T5[17].....	31
Hình 2.15. Sự hình thành các mô hình LLM qua các năm[10] .....	32
Hình 2.16. Biểu diễn quá trình pre-train và fine-tuning[11] .....	33
Hình 2.17. Mô tả về quá trình fine-tuning[12] .....	34
Hình 2.18. Mô tả quá trình cập nhật trọng số Lora fine -tuning[12] .....	35
Hình 2.19. Quá trình cập nhật trọng số Full Fine Tuning và Lora fine tuning[12]....	36
Hình 2.20. Mô tả rõ hơn về việc phân tách ma trận[13].....	37
Hình 2.21. Biểu diễn tỉ lệ trùng lặp của ba thông số[15].....	38
Hình 3.1. Sơ đồ quy trình hoạt động cơ bản của RAG[2] .....	44
Hình 3.2. Mô hình Naive RAG và mô hình Advanced RAG[2] .....	45

Hình 3.3. Sơ đồ quy trình thực hiện của mô hình Naive RAG[3] .....	46
Hình 3.4. Sơ đồ quy trình thực hiện của mô hình Advanced RAG[3] .....	47
Hình 4.1. Giao diện trang web Giải đáp chính sách Online .....	53
Hình 4.2. Nội dung Data Frame của dữ liệu sau khi thu thập .....	53
Hình 4.3. Data Frame sau khi thực hiện tổ chức dữ liệu .....	54
Hình 4.4. Biểu đồ Phân bố dữ liệu: Train, Validation, Test.....	56
Hình 4.5. Cấu trúc của Bert trong quá trình full fine-tuning .....	61
Hình 4.6. Biểu đồ learning curve mô tả loss của quá trình full-tuning .....	63
Hình 4.7. Cấu trúc của Bert trong quá trình lora fine-tuning .....	64
Hình 4.8. Biểu đồ learning curve mô tả loss của quá trình lora fine-tuning .....	66
Hình 4.9. Biểu đồ phân bố độ dài câu theo từ theo 3 thuộc tính của tệp dữ liệu luật pháp.....	72
Hình 4.10. Biểu đồ phân bố độ dài câu theo từ theo 3 thuộc tính của tệp dữ liệu luật pháp.....	73
Hình 4.11. Biểu đồ Phân bố dữ liệu: Train, Validation, Test .....	74
Hình 4.12. Sơ đồ quá trình Full Fine-Tuning T5.....	76
Hình 4.13. Biểu đồ learning curve mô tả loss của quá trình full fine-tuning .....	78
Hình 4.14. Sơ đồ quá trình thu thập dữ liệu cho bài toán truy xuất tăng cường .....	81
Hình 4.15. File văn bản sau khi tiền xử lý .....	82
Hình 4.16. Sơ đồ quá trình tổ chức dữ liệu.....	83
Hình 4.17. Sơ đồ quá trình lưu dữ liệu vào vector database.....	85
Hình 4.18. Các collections sau khi upload vào Qdrant .....	86
Hình 4.19. Sơ đồ quy trình thực hiện truy xuất tăng cường .....	87
Hình 4.20. Quy trình hoạt động của hệ thống Hỏi đáp Pháp luật.....	90
Hình 4.21. Giao diện mặc định khi mới truy cập hoặc làm mới trang web .....	92
Hình 4.22. Giao diện mặc định khi bắt đầu một phiên Chat mới .....	92

Hình 4.23. Thay đổi trên giao diện sau khi người dùng lựa chọn Đổi Mô Hình .....	92
Hình 4.24. Giao diện hệ khi nhận được phản hồi từ hệ thống.....	93
Hình 4.25. Giao diện khi người dùng click vào Trích dẫn tham khảo .....	93
Hình 4.26. Giao diện khi người dùng truy cập lại vào một phiên Chat cũ.....	94
Hình 4.27. Giao diện khi người dùng thực hiện thao tác Xóa lịch sử chat .....	95

## **DANH MỤC BẢNG**

Bảng 4.1. Bảng mô tả cấu hình siêu tham số cho quá trình huấn luyện Bert Full-fine-tuning .....	62
Bảng 4.2. Bảng kết quả thu được sau quá trình Bert Full-fine-tuning .....	62
Bảng 4.3. Bảng mô tả cấu hình siêu tham số cho quá trình huấn luyện Bert lora fine tuning .....	65
Bảng 4.4. Bảng kết quả thu được sau quá trình Lora-fine-tuning .....	65
Bảng 4.5. Phân cứng cần thiết cho quá trình Bert Full-fine-tuning và Lora-fine-tuning .....	67
Bảng 4.6. Kết quả Full tuning của BERT .....	69
Bảng 4.7. Kết quả Lora fine-tuning của BERT .....	70
Bảng 4.8. Tiêu chí so sánh giữa hai kỹ thuật.....	70
Bảng 4.9. Bảng mô tả độ dài của các câu trong tệp dữ liệu luật .....	72
Bảng 4.10. Bảng cấu hình siêu tham số cho quá trình huấn luyện T5 .....	76
Bảng 4.11. Bảng kết quả thu được sau quá trình T5 Full-fine-tuning.....	77
Bảng 4.12. Các chỉ số ROUGE trên tệp dữ liệu pháp luật .....	80

## DANH MỤC CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Mô tả
1	AI	Artificial Intelligence
2	API	Application Programming Interface
3	BERT	Bidirectional Encoder Representations from Transformers
4	CNN	Convolutional Neural Network
5	CSS	Cascading Style Sheets
6	GLUE	General Language Understanding Evaluation
7	GPT	Chat Generative Pre-training Transformer
8	GPU	Graphics Processing Unit
9	HTML	Hypertext Markup Language
10	LLM	Large Language Model
11	LoRA	Low-Rank Adaptation
12	ML	Machine Learning
13	MRC	Machine Reading Comprehension
14	MultiNLI	Multi-Genre Natural Language Inference
15	NLP	Natural Language Processing
16	QA	Question Answering
17	RAG	Retrieval - Augmented Generation
18	RNNs	Recurrent Neural Networks
19	ROUGE	Recall-Oriented Understudy for Gisting Evaluation
20	SQuAD	Stanford Question Answering Dataset
21	T5	Text-to-Text Transfer Transformer
22	TF-IDF	Term Frequency-Inverse Document Frequency

## MỞ ĐẦU

### 1. VẤN ĐỀ ĐẶT RA

Trong bối cảnh xã hội hiện đại phát triển mạnh mẽ, nhu cầu tìm kiếm thông tin chính xác và nhanh chóng trong lĩnh vực pháp lý ngày càng trở nên cần thiết. Trước đây, để giải quyết các câu hỏi pháp lý, chúng ta thường phải dựa vào các chuyên gia hoặc tra cứu văn bản pháp luật trên các nguồn thông tin chính thống. Tuy nhiên, việc này đòi hỏi nhiều thời gian và công sức, đặc biệt là khi đối mặt với các vấn đề pháp lý phức tạp, cần sự hiểu biết sâu rộng về luật pháp.

Bên cạnh đó, sự phát triển không ngừng của công nghệ thông tin, đặc biệt là trí tuệ nhân tạo (AI) và xử lý ngôn ngữ tự nhiên (NLP), đã mở ra nhiều cơ hội mới để cải thiện cách chúng ta tiếp cận thông tin pháp lý. Tuy nhiên, việc ứng dụng các công nghệ này vào lĩnh vực pháp lý vẫn gặp nhiều thách thức. Một mặt, các văn bản pháp luật thường mang tính kỹ thuật cao với cấu trúc phức tạp, sử dụng ngôn ngữ chuyên môn và chứa đựng nhiều trường hợp ngoại lệ. Mặt khác, các tình huống pháp lý thực tế thường đa dạng và mang tính ngữ cảnh cao, đòi hỏi hệ thống không chỉ cung cấp thông tin đúng mà còn phải hiểu và áp dụng đúng ngữ cảnh của câu hỏi.

Hơn nữa, khi khối lượng thông tin pháp lý ngày càng tăng lên, từ các quy định mới đến các phán quyết pháp lý và các tài liệu liên quan khác, việc khai thác và quản lý hiệu quả thông tin này trở thành một vấn đề cấp thiết. Người dùng thường cảm thấy choáng ngợp trước số lượng lớn tài liệu cần tham khảo, dẫn đến việc xử lý thông tin trở nên chậm trễ và thiếu hiệu quả.

Trong bối cảnh đó, việc xây dựng một hệ thống tự động trả lời câu hỏi pháp lý không chỉ giúp giảm tải công việc cho các chuyên gia pháp lý mà còn đáp ứng tốt hơn nhu cầu thông tin của người dân và các tổ chức. Hệ thống này không chỉ cần khả năng hiểu ngôn ngữ tự nhiên mà còn phải có khả năng truy xuất, lọc và tổ chức thông tin pháp luật phù hợp với yêu cầu của người dùng. Đây là một bước tiến quan trọng hướng đến việc hiện đại hóa dịch vụ pháp lý, đồng thời tạo điều kiện cho việc tiếp cận công lý và thông tin pháp luật trở nên công bằng và hiệu quả hơn.

## **2. MỤC TIÊU NGHIÊN CỨU**

Mục tiêu chính của nghiên cứu này là áp dụng kiến trúc Transformer để phát triển một hệ thống trả lời câu hỏi pháp lý, kết hợp hai kỹ thuật Fine-Tuning và Retrieval - Augmented Generation (RAG).

Về mặt lý thuyết, nghiên cứu sẽ phân tích chi tiết cấu trúc của Transformer, bao gồm các thành phần Encoder và Decoder, và cách chúng phối hợp trong việc xử lý ngôn ngữ tự nhiên. Việc hiểu rõ cơ chế hoạt động của Transformer, đặc biệt là cơ chế Attention, sẽ giúp mô hình nắm bắt ngữ cảnh tốt hơn, từ đó cải thiện độ chính xác trong việc trả lời các câu hỏi.

Về mặt ứng dụng, nghiên cứu sẽ triển khai các mô hình ngôn ngữ lớn (Large Language Model - LLM) dựa trên kiến trúc Transformer, kết hợp với hai kỹ thuật Fine-Tuning và RAG để giải quyết bài toán trả lời câu hỏi trong lĩnh vực pháp lý. Cụ thể, thông qua RAG, hệ thống sẽ truy xuất các thông tin từ các cơ sở dữ liệu vector chứa các văn bản quy phạm pháp luật nhằm bổ sung ngữ cảnh cho mô hình. Các ngữ cảnh này sau đó sẽ là đầu vào cho mô hình LLM đã được tinh chỉnh bằng bộ dữ liệu gồm các câu hỏi và tình huống pháp lý thực tế, được thu thập thông qua kỹ thuật web scraping.

Mục tiêu cuối cùng là phát triển một chatbot tư vấn pháp luật có khả năng hiểu ngữ cảnh sâu sắc và truy xuất thông tin chính xác, giúp người dùng nhận được những giải pháp pháp lý nhanh chóng và hiệu quả. Những ứng dụng này không chỉ nâng cao tốc độ trả lời mà còn cải thiện đáng kể độ chính xác và khả năng xử lý các tình huống pháp lý phức tạp, góp phần hỗ trợ tư vấn pháp luật cho cộng đồng một cách thiết thực.

## **3. PHẠM VI NGHIÊN CỨU**

Nghiên cứu này tập trung vào kiến trúc Transformer và các mô hình ngôn ngữ lớn (Large Language Models - LLMs) được phát triển dựa trên kiến trúc này, nhằm giải quyết bài toán trả lời câu hỏi trong lĩnh vực pháp lý. Phương pháp đề xuất sẽ triển khai các kỹ thuật Fine-Tuning và Retrieval-Augmented Generation (RAG) trên các mô hình ngôn ngữ lớn, với mục tiêu cải thiện khả năng hiểu ngữ cảnh và đưa ra các câu trả lời chính xác, phù hợp với yêu cầu pháp lý.

Phạm vi nghiên cứu bao gồm việc sử dụng tập dữ liệu tiếng Việt, tập trung vào các văn bản pháp luật Việt Nam và các câu hỏi tình huống liên quan. Tập dữ liệu sẽ được thu thập từ các nguồn chính thống như Cơ sở dữ liệu Quốc gia về Văn bản Pháp luật và Giải đáp chính sách Online, đảm bảo tính chính xác và đầy đủ của thông tin pháp lý. Nghiên cứu giới hạn trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) cho tiếng Việt, tập trung vào các vấn đề pháp lý phổ biến tại Việt Nam, nhằm phát triển một chatbot tư vấn pháp luật chính xác và hiệu quả.

#### **4. TẦM QUAN TRỌNG CỦA ĐỀ TÀI**

Kiến trúc Transformer đã nỗi lèn như một giải pháp hiện đại và tiên tiến trong việc giải quyết bài toán trả lời câu hỏi tự động, đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Với khả năng nắm bắt ngữ cảnh và tạo ra các câu trả lời có độ chính xác cao, kiến trúc này không chỉ đáp ứng được yêu cầu về tốc độ và độ chính xác mà còn có thể xử lý tốt các vấn đề phức tạp trong việc tra cứu pháp luật.

Việc hiểu sâu về các thành phần trong kiến trúc Transformer cùng với việc áp dụng hai kỹ thuật quan trọng: Fine-Tuning và Retrieval-Augmented Generation (RAG), giúp tăng cường hiệu suất trong việc trả lời các câu hỏi pháp lý phức tạp. Fine-Tuning mô hình trên tập dữ liệu pháp luật Việt Nam sẽ giúp cá nhân hóa và tối ưu hóa mô hình cho bối cảnh cụ thể, trong khi RAG giúp tăng cường khả năng truy xuất thông tin và tạo ra câu trả lời chính xác hơn từ nguồn dữ liệu lớn.

Tầm quan trọng của nghiên cứu này không chỉ dừng lại ở việc cải tiến hiệu quả và độ chính xác của hệ thống trả lời câu hỏi pháp lý tự động, mà còn mở ra khả năng nâng cao trải nghiệm người dùng khi tương tác với các hệ thống AI tư vấn pháp luật. Điều này đặc biệt hữu ích trong bối cảnh nhu cầu tiếp cận và tra cứu thông tin pháp luật ngày càng tăng, giúp mọi người có thể dễ dàng nhận được các giải đáp nhanh chóng và chính xác mà không cần phải qua trung gian hoặc tốn nhiều thời gian.

#### **5. PHƯƠNG PHÁP NGHIÊN CỨU**

Phương pháp khảo sát và tổng hợp các tài liệu liên quan đến kiến trúc Transformer và ứng dụng của nó trong bài toán trả lời câu hỏi (QA). Điều này bao gồm việc phân tích các bài báo khoa học, tài liệu nghiên cứu từ các ứng dụng thực tiễn, cũng như tham khảo các khóa học trực tuyến về chủ đề này. Mục tiêu của phần nghiên

cứu lý thuyết là xây dựng một nền tảng kiến thức toàn diện về kiến trúc Transformer, bao gồm các đặc điểm kỹ thuật, cơ chế hoạt động (như Attention và Self-Attention), và các ứng dụng tiềm năng của nó trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và pháp lý. Qua đó, có thể nắm rõ những cải tiến gần đây và các phương pháp hiệu quả trong lĩnh vực này, bao gồm cả việc tối ưu hóa mô hình bằng các kỹ thuật tiên tiến như Low-Rank Adaptation (LoRA) để giảm chi phí tính toán mà không ảnh hưởng đáng kể đến hiệu suất.

**Thực nghiệm:** Sau khi hoàn tất phần lý thuyết, sẽ triển khai các thí nghiệm thực tế với Các văn bản quy phạm pháp luật hiện hành tại Việt Nam và Các câu hỏi tình huống thực tế sẽ là bộ dữ liệu chính. Các câu hỏi tình huống thực tế sẽ cung cấp nguồn dữ liệu pháp lý quan trọng để xây dựng và huấn luyện mô hình. Chúng tôi sẽ sử dụng các câu hỏi tình huống thực tế này để huấn luyện và kiểm thử mô hình Transformer. Mục tiêu của các thí nghiệm thực nghiệm là đánh giá khả năng của mô hình Transformer trong việc xử lý và trả lời các câu hỏi pháp lý liên quan đến các văn bản quy phạm pháp luật hiện hành. Bên cạnh đó, cũng sẽ đánh giá khả năng của mô hình khi áp dụng kỹ thuật Retrieval - Augmented Generation (RAG), kết hợp truy vấn thông tin từ các văn bản pháp lý và tạo ra các câu trả lời có chất lượng cao hơn cho các câu hỏi pháp lý. Ngoài ra sẽ đánh giá hiệu quả của mô hình dựa trên các tiêu chí đánh giá bài toán trả lời câu hỏi, với trọng tâm là độ chính xác và khả năng xử lý các tình huống phức tạp liên quan đến pháp lý. Kết quả thực nghiệm sẽ giúp chúng tôi điều chỉnh và tối ưu hóa mô hình, từ đó nâng cao hiệu quả ứng dụng trong các hệ thống tra cứu pháp luật và hỗ trợ giải đáp tình huống thực tiễn.

# **CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN TRẢ LỜI CÂU HỎI**

## **1.1. GIỚI THIỆU VỀ BÀI TOÁN TRẢ LỜI CÂU HỎI**

Bài toán trả lời câu hỏi (Chatbot) là một chương trình máy tính kết hợp trí tuệ nhân tạo. Nó được sinh ra để tương tác với người dùng và thay thế một phần cho nhân viên tư vấn hỏi đáp, thắc mắc của người dùng. Thông qua hình thức tin nhắn (textual) hoặc âm thanh (auditory), Bài toán trả lời câu hỏi (chatbot) có thể tương tác với người dùng theo thời gian thực. Do được thiết kế để mô phỏng cách con người tương tác với nhau nên các hệ thống chatbot thường phải được điều chỉnh và thử nghiệm một cách liên tục và lâu dài.

## **1.2. XỬ LÝ NGÔN NGỮ TỰ NHIÊN (NLP) TRONG BÀI TOÁN TRẢ LỜI CÂU HỎI**

Xử lý Ngôn ngữ Tự nhiên (NLP) là một lĩnh vực con của trí tuệ nhân tạo, nghiên cứu và ứng dụng các phương pháp giúp máy tính có thể hiểu, phân tích và phản hồi ngôn ngữ tự nhiên của con người. Trong bài toán trả lời câu hỏi, NLP đóng vai trò quan trọng trong việc phân tích và hiểu các câu hỏi từ người dùng, trích xuất thông tin từ tài liệu hoặc cơ sở dữ liệu, và tạo ra các câu trả lời phù hợp.

Sự phát triển của các mô hình học sâu (Deep Learning), đặc biệt là các mô hình ngôn ngữ lớn (Large Language Models - LLMs) như BERT và GPT, đã giúp bài toán trả lời câu hỏi đạt được những bước tiến vượt bậc. Các mô hình này có khả năng giải quyết các vấn đề ngữ nghĩa và ngữ cảnh phức tạp, từ đó cải thiện hiệu quả trả lời câu hỏi trong các tình huống yêu cầu độ chính xác cao.

Để giải quyết bài toán trả lời câu hỏi, các hệ thống NLP thường thực hiện các bước phân tích chính như tiền xử lý ngôn ngữ, phân tích ngữ pháp và cú pháp, phân tích ngữ nghĩa, truy xuất thông tin và sinh câu trả lời. Tiền xử lý ngôn ngữ là bước đầu tiên, bao gồm tách từ (tokenization), loại bỏ từ không có nghĩa (stop words), và chuẩn hóa văn bản. Sau đó, mô hình sẽ phân tích ngữ pháp và cú pháp để xác định mối quan hệ giữa các từ trong câu hỏi. Tiếp theo, mô hình sẽ phân tích ngữ nghĩa để hiểu ngữ cảnh và mục đích của câu hỏi. Cuối cùng, hệ thống sẽ truy xuất thông tin từ cơ sở dữ liệu hoặc tài liệu liên quan và sinh ra câu trả lời phù hợp.

### **1.2.1. Các phương pháp xử lý ngôn ngữ trong NLP**

Trong bài toán trả lời câu hỏi, các phương pháp xử lý ngôn ngữ có thể được chia thành hai nhóm chính: phương pháp truyền thống và phương pháp hiện đại.

Phương pháp truyền thống chủ yếu sử dụng các kỹ thuật dựa trên quy tắc (rule-based) hoặc mô hình thống kê như N-gram. Tuy nhiên, phương pháp này gặp khó khăn trong việc giải quyết những câu hỏi phức tạp hoặc thiếu cấu trúc rõ ràng. Đồng thời, hiệu suất của phương pháp này không cao khi phải đối mặt với các câu hỏi yêu cầu sự hiểu ngữ cảnh sâu sắc.

Phương pháp hiện đại dựa trên mô hình Transformer như BERT và GPT mang lại khả năng hiểu ngữ cảnh và ngữ nghĩa mạnh mẽ hơn. Các mô hình này sử dụng cơ chế Attention để tập trung vào các phần quan trọng trong câu hỏi và tài liệu, giúp cải thiện độ chính xác trong việc trả lời câu hỏi. Việc sử dụng các mô hình ngôn ngữ lớn (LLMs) đã giúp nâng cao chất lượng trả lời câu hỏi, đặc biệt trong các lĩnh vực chuyên môn như pháp lý, y tế hay kỹ thuật, nơi yêu cầu kiến thức chuyên sâu.

### **1.2.2. Ứng dụng của NLP trong bài toán trả lời câu hỏi**

NLP không chỉ được áp dụng trong bài toán trả lời câu hỏi liên quan đến các lĩnh vực chuyên môn mà còn được ứng dụng rộng rãi trong nhiều ngành nghề khác nhau. Một trong những ứng dụng quan trọng của NLP trong bài toán trả lời câu hỏi là trong lĩnh vực pháp lý, nơi các hệ thống cần tìm kiếm và cung cấp các câu trả lời chính xác từ cơ sở dữ liệu pháp lý lớn.

Ngoài ra, NLP còn đóng vai trò quan trọng trong các hệ thống chăm sóc khách hàng, khi được tích hợp vào các chatbot để trả lời tự động các câu hỏi phổ biến từ khách hàng. Điều này giúp tiết kiệm thời gian và công sức cho nhân viên hỗ trợ.

Các ứng dụng khác của NLP bao gồm trong các hệ thống học tập thông minh, giúp học sinh, sinh viên trả lời các câu hỏi giáo dục trong môi trường học trực tuyến, từ đó nâng cao hiệu quả học tập. Trong các môi trường tương tác người-máy, NLP giúp tạo ra những trải nghiệm người dùng mượt mà hơn, ví dụ như trợ lý ảo giúp người dùng tìm kiếm thông tin, điều khiển thiết bị hoặc giải đáp các câu hỏi hằng ngày.

### **1.2.3. Những thách thức trong NLP cho bài toán trả lời câu hỏi**

Mặc dù đã có những tiến bộ đáng kể trong NLP, nhưng việc áp dụng các phương pháp NLP vào bài toán trả lời câu hỏi vẫn gặp phải một số thách thức. Một trong những vấn đề lớn là hiểu ngữ cảnh câu hỏi, đặc biệt là khi câu hỏi có ngữ cảnh phức tạp hoặc thiếu rõ ràng. Trong những trường hợp này, mô hình có thể gặp khó khăn trong việc xác định chính xác mục đích của người hỏi.

Một vấn đề khác là xử lý ngôn ngữ chuyên ngành. Trong các lĩnh vực như pháp lý, y tế hay kỹ thuật, ngôn ngữ chuyên ngành có thể rất phức tạp và yêu cầu mô hình phải có kiến thức sâu về lĩnh vực đó. Điều này làm tăng độ khó trong việc phát triển các hệ thống trả lời câu hỏi hiệu quả.

Ngoài ra, dữ liệu không đầy đủ hoặc không cấu trúc cũng là một thách thức lớn. Các tài liệu nguồn có thể không đồng nhất, khiến mô hình không thể truy xuất thông tin một cách chính xác hoặc đầy đủ. Điều này đòi hỏi phải có những kỹ thuật xử lý dữ liệu tiên tiến hơn, chẳng hạn như việc sử dụng các mô hình học sâu để cải thiện khả năng trích xuất thông tin từ dữ liệu không cấu trúc.

### **1.2.4. Những thách thức khi xây dựng bài toán trả lời câu hỏi**

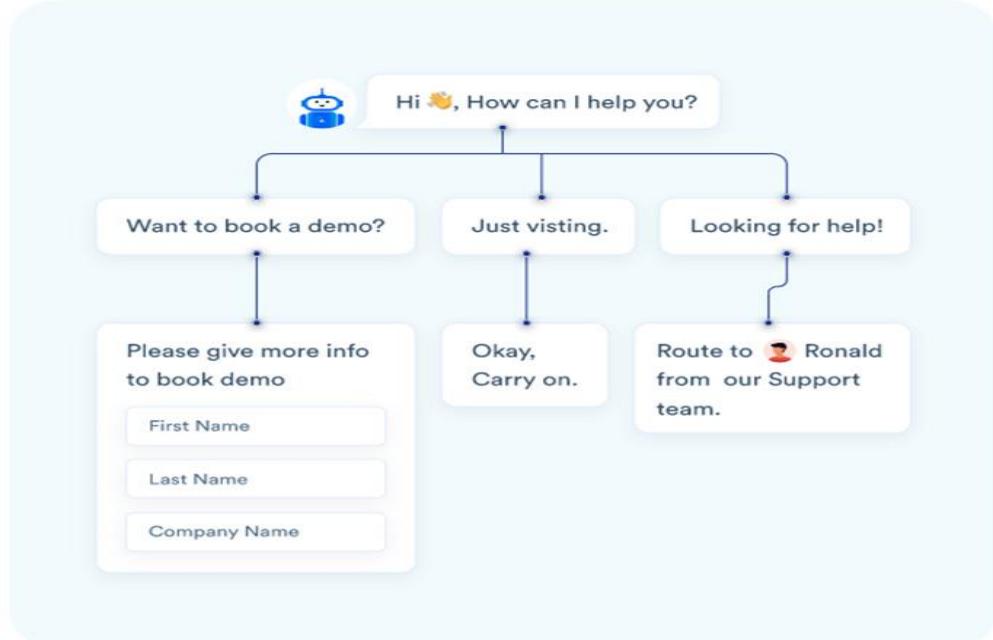
Việc xây dựng hệ thống trả lời câu hỏi đòi hỏi sự kết hợp giữa nhiều yếu tố như phân tích ngữ nghĩa, truy xuất thông tin và sinh câu trả lời. Đặc biệt trong các lĩnh vực chuyên môn, mô hình không chỉ cần hiểu câu hỏi mà còn phải có khả năng tìm kiếm thông tin từ các nguồn tài liệu chuyên ngành. Hơn nữa, việc xử lý lượng dữ liệu lớn và đa dạng trong thời gian ngắn là một thách thức không nhỏ, yêu cầu hệ thống phải có khả năng xử lý nhanh chóng và chính xác. Những công cụ hiện đại như mô hình Transformer và cơ sở dữ liệu bên ngoài sẽ giúp cải thiện chất lượng câu trả lời, nhưng điều này đòi hỏi một hệ thống tính toán mạnh mẽ và dữ liệu chất lượng cao.

## **1.3. CÁC PHƯƠNG PHÁP XÂY DỰNG BÀI TOÁN TRẢ LỜI CÂU HỎI**

### **1.3.1. Dựa trên kịch bản**

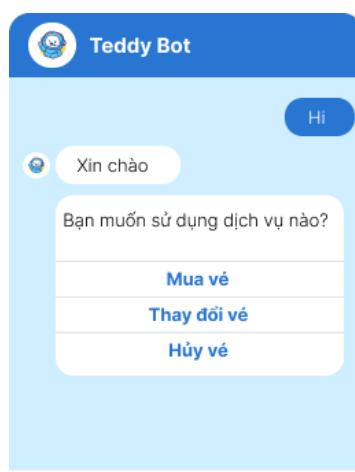
Scripted Chatbots hoạt động dựa trên các quy tắc cố định để trả lời những câu hỏi cụ thể. Phương pháp này có ưu điểm dễ triển khai và phù hợp với những tình huống có cấu trúc rõ ràng hoặc yêu cầu nội dung đơn giản. Tuy nhiên, nhược điểm lớn của mô hình này là khả năng hạn chế trong việc xử lý các câu hỏi phức tạp hoặc không

nằm trong phạm vi các kịch bản đã được định nghĩa trước. Điều này dẫn đến việc giảm hiệu quả, đặc biệt khi chatbot cần xử lý các tình huống mang tính tùy biến cao hoặc đòi hỏi hiểu biết ngữ cảnh sâu sắc.



**Hình 1.1. Mô tả hoạt động của Scripted Chatbot[8]**

Một dạng triển khai cụ thể của Scripted Chatbot là Clicking Bot cho phép người dùng tương tác thông qua các nút bấm được thiết kế sẵn [8]. Thay vì nhập liệu, người dùng chỉ cần thực hiện các lựa chọn qua các nút, từ đó chatbot sẽ phản hồi thông tin phù hợp với tùy chọn được chọn. Cách tiếp cận này mang lại sự tiện lợi, đặc biệt trong các tình huống có tính lặp lại hoặc yêu cầu xử lý các câu hỏi phổ biến, ví dụ như tra cứu giờ làm việc, hướng dẫn sử dụng, hoặc cung cấp thông tin cơ bản.



### **Hình 1.2. Mô tả hoạt động của Clicking Bot[8]**

Tuy nhiên, giới hạn của Clicking Bot nằm ở chính cách thức hoạt động theo kịch bản. Trong các trường hợp đòi hỏi sự hiểu biết ngữ cảnh hoặc các câu hỏi nằm ngoài phạm vi thiết kế, chatbot có thể trở nên kém hiệu quả, thậm chí không thể phản hồi chính xác. Người dùng cũng có thể cảm thấy bất tiện khi phải thực hiện nhiều lần nhấn nút để đạt được thông tin mong muốn. Điều này làm giảm trải nghiệm người dùng, đặc biệt khi so sánh với các chatbot hiện đại hơn dựa trên trí tuệ nhân tạo.

#### **1.3.2. Dựa trên biểu diễn tri thức**

Bài toán trả lời câu hỏi dựa trên biểu diễn tri thức hoạt động bằng cách tận dụng biểu đồ tri thức (Knowledge Graph) hoặc các mô hình dữ liệu được tổ chức có hệ thống để trả lời câu hỏi. Phương pháp này xây dựng dựa trên các thực thể (entities), mối quan hệ (relationships), và thuộc tính (attributes) của dữ liệu để cung cấp câu trả lời chính xác và logic.

Cách hoạt động của hệ thống dựa trên tri thức:

- Xây dựng biểu đồ tri thức: Tri thức được biểu diễn dưới dạng đồ thị với các thực thể (ví dụ: luật, bệnh lý, khái niệm) và mối quan hệ giữa chúng.
- Ánh xạ câu hỏi: Ánh xạ câu hỏi người dùng sang các thực thể và mối quan hệ trong biểu đồ.
- Truy vấn tri thức: Sử dụng các ngôn ngữ truy vấn tri thức như SPARQL để tìm kiếm câu trả lời.
- Trả lời người dùng: Biến đổi kết quả truy vấn thành câu trả lời tự nhiên.

Ví dụ cụ thể:

Câu hỏi: "Ai là tác giả của cuốn sách 'Sông Đông Em Đèm'?"

Biểu đồ tri thức:

Thực thể: "Sông Đông Em Đèm"

Thuộc tính: Tác giả

Giá trị: Mikhail Sholokhov

Trả lời: "Tác giả của cuốn sách 'Sông Đông Em Đèm' là Mikhail Sholokhov."

Hệ thống này cũng có khả năng suy diễn mạnh mẽ, cho phép xử lý các truy vấn phức tạp thông qua việc suy luận dựa trên các mối quan hệ trong biểu đồ tri thức.

Điều này đặc biệt hiệu quả trong các lĩnh vực chuyên môn như pháp luật, y tế, giáo dục hoặc khoa học, nơi dữ liệu thường đã được tổ chức và cấu trúc sẵn.

Tuy nhiên, phương pháp này không phải không có những hạn chế. Một trong những thách thức lớn nhất là chi phí xây dựng tri thức cao, vì quá trình thu thập, làm sạch và tổ chức dữ liệu để tạo biểu đồ tri thức đòi hỏi nhiều công sức và thời gian. Ngoài ra, hệ thống có thể gặp khó khăn trong việc mở rộng tri thức, vì việc cập nhật hoặc mở rộng cơ sở tri thức thường đòi hỏi thao tác thủ công và không tự động hóa hoàn toàn. Hơn nữa, phương pháp này có khả năng xử lý ngôn ngữ tự nhiên hạn chế, vì nó yêu cầu hệ thống có khả năng ánh xạ câu hỏi của người dùng sang tri thức đã có một cách chính xác.

### 1.3.3. Dựa trên các mô hình LLM

Bài toán trả lời câu hỏi dựa trên trí tuệ nhân tạo (AI Chatbots): sử dụng các mô hình Xử lý Ngôn ngữ Tự nhiên (NLP) hiện đại kết hợp với cơ sở dữ liệu để tạo ra câu trả lời. Phương pháp này mang lại sự linh hoạt và khả năng cung cấp câu trả lời chính xác cao, nhất là trong các tình huống yêu cầu sự hiểu biết sâu sắc về ngữ cảnh hoặc các lĩnh vực chuyên môn. Tuy nhiên, AI Chatbots đòi hỏi tài nguyên tính toán lớn và cần dữ liệu chất lượng cao để hoạt động hiệu quả, điều này có thể là một thách thức đối với các hệ thống nhỏ hoặc không có đủ cơ sở dữ liệu phong phú.

## 1.4. CƠ CHẾ HOẠT ĐỘNG CỦA BÀI TOÁN TRẢ LỜI CÂU HỎI

Cơ chế hoạt động của bài toán trả lời câu hỏi thường trải qua ba bước chính. Đầu tiên là phân tích câu hỏi, nơi hệ thống cần hiểu được ý định và ngữ cảnh của câu hỏi để xác định thông tin cần thiết. Tiếp theo, hệ thống thực hiện tìm kiếm thông tin, tức là truy xuất dữ liệu từ cơ sở dữ liệu hoặc các tài liệu liên quan để lấy thông tin cần thiết cho câu trả lời. Cuối cùng, xử lý và tạo câu trả lời là bước hệ thống sử dụng các thông tin truy xuất được để sinh ra câu trả lời phù hợp, có thể là một câu trả lời trực tiếp hoặc dựa trên kiến thức đã học từ các dữ liệu huấn luyện.

## CHƯƠNG 2: CÁC MÔ HÌNH LARGE LANGUAGE MODEL CHO BÀI TOÁN TRẢ LỜI CÂU HỎI

### 2.1. TỔNG QUAN VỀ TRANSFORMER

#### 2.1.1. Giới thiệu

Năm 2017, Google công bố bài báo “Attention Is All You Need” thông tin về Transformer như tạo ra bước ngoặt mới trong lĩnh vực NLP: “*Transformer Neural Network is the current state-of-art technique in the field of NLP*” [1]. Trước khi Transformer xuất hiện, các tác vụ xử lý ngôn ngữ tự nhiên (NLP), đặc biệt trong lĩnh vực dịch máy (Machine Translation), chủ yếu dựa vào kiến trúc Recurrent Neural Networks (RNNs). Tuy nhiên, do đặc tính phải xử lý dữ liệu theo thứ tự tuần tự, RNNs gặp phải nhiều hạn chế như hiệu suất xử lý thấp, khó khăn trong việc biểu diễn sự phụ thuộc dài hạn giữa các từ, và dễ bị ảnh hưởng bởi vấn đề gradient tiêu biến hoặc phát nổ trong quá trình tối ưu hóa.

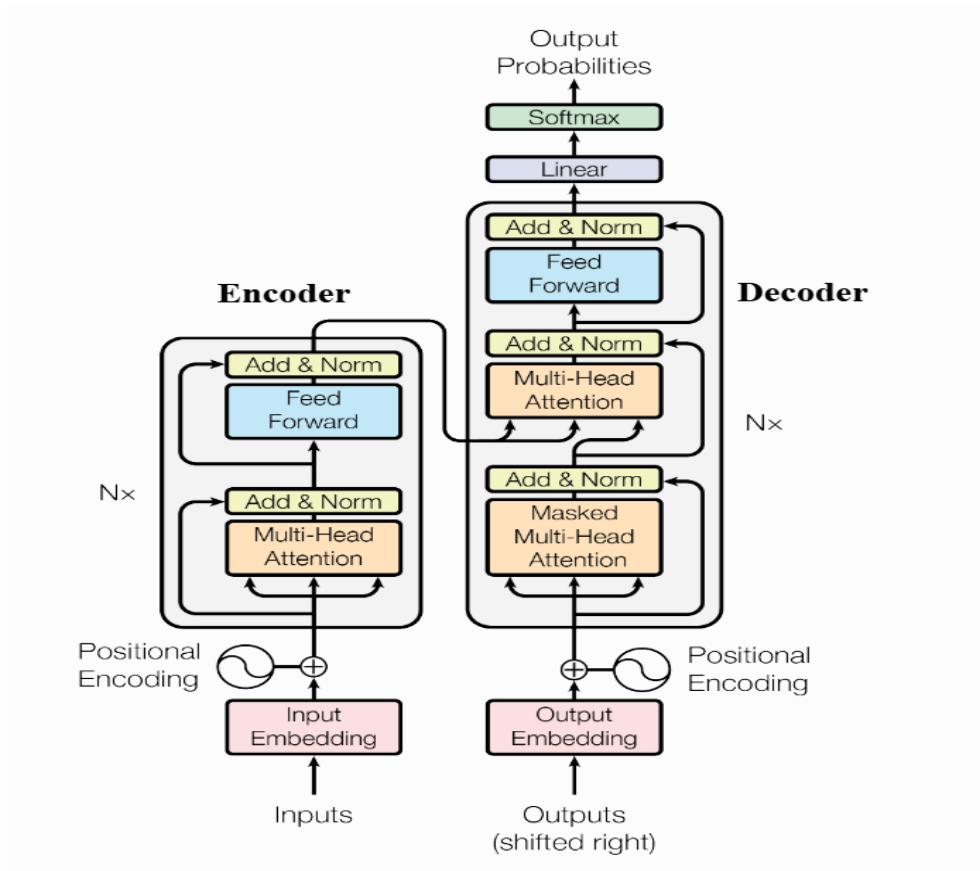
Trước khi Transformer xuất hiện, các tác vụ xử lý ngôn ngữ tự nhiên (NLP), đặc biệt trong lĩnh vực dịch máy (Machine Translation), chủ yếu dựa vào kiến trúc Recurrent Neural Networks (RNNs). Tuy nhiên, do đặc tính phải xử lý dữ liệu theo thứ tự tuần tự, RNNs gặp phải nhiều hạn chế như hiệu suất xử lý thấp, khó khăn trong việc biểu diễn sự phụ thuộc dài hạn giữa các từ, và dễ bị ảnh hưởng bởi vấn đề gradient tiêu biến hoặc phát nổ trong quá trình tối ưu hóa. Khi Transformer ra đời, những hạn chế vốn tồn tại trong Recurrent Neural Networks (RNNs) gần như đã được khắc phục hoàn toàn. Không chỉ giải quyết triệt để các vấn đề về hiệu suất xử lý tuần tự và khó khăn trong việc mô hình hóa mối quan hệ dài hạn giữa các từ, Transformer còn trở thành nền tảng cho nhiều mô hình học sâu mang tính cách mạng, chẳng hạn như BERT (Bidirectional Encoder Representations from Transformers, 2019), mô hình được Google tích hợp vào hệ thống tìm kiếm của mình, hay AlphaStar của DeepMind – một hệ thống trí tuệ nhân tạo đã đánh bại các cao thủ hàng đầu trong trò chơi Starcraft.

Với thiết kế hiện đại, Transformer đã mở ra một kỷ nguyên mới trong việc giải quyết các bài toán ngôn ngữ và xử lý tiếng nói. Các ứng dụng điển hình bao gồm dịch máy, sinh ngôn ngữ tự nhiên, phân loại văn bản, nhận dạng thực thể, nhận dạng giọng nói và chuyển đổi văn bản thành giọng nói. Không giống như RNNs, Transformer

không yêu cầu xử lý dữ liệu theo thứ tự tuần tự, cho phép mô hình tập trung vào các mối quan hệ quan trọng giữa các từ trong chuỗi thông qua cơ chế Self-Attention. Điều này giúp giảm thiểu đáng kể các vấn đề như gradient tiêu biến hoặc phát nổ trong quá trình tối ưu hóa, đồng thời đảm bảo không bỏ sót các thông tin quan trọng từ ngữ cảnh. Nhờ khả năng xử lý song song mạnh mẽ, Transformer tận dụng hiệu quả sức mạnh tính toán của GPU, giúp tăng tốc độ xử lý và cải thiện hiệu quả tổng thể, trở thành một trong những kiến trúc nền tảng quan trọng nhất của học sâu hiện nay.

### 2.1.2. Kiến trúc Transformer

Transformer gồm hai thành phần chính là Encoder và Decoder như trong hình 1, cả hai thành phần này đều có cấu tạo tương tự nhau gồm nhiều Block, mỗi Block được thiết kế với nhiều Sub-Layer. Sub-Layer đầu tiên là Multi-Head Attention Layer, Sub-Layer tiếp theo là Feedforward Neural Network Layer.



Hình 2.1. Mô hình kiến trúc Transformer[1]

### **2.1.3. Kiến trúc và cách thức hoạt động**

#### *2.1.3.1. Input Embedding*

Tất cả các thuật toán học sâu đều yêu cầu đầu vào là dữ liệu dạng số. Tuy nhiên, trong ngôn ngữ tự nhiên, các từ không phải là các giá trị số. Vì vậy, chúng ta cần phải chuyển đổi các từ sang dạng số để có thể đưa chúng vào mô hình học sâu. Một trong những cách cổ điển nhất là sử dụng TF-IDF để biểu diễn trọng số của một từ trong tài liệu, dựa trên thống kê mức độ quan trọng của từ đó trong một tài liệu so với toàn bộ tập hợp các tài liệu. Tuy nhiên, cách này vẫn còn hạn chế vì không xét đến mối quan hệ giữa các từ trong một câu.

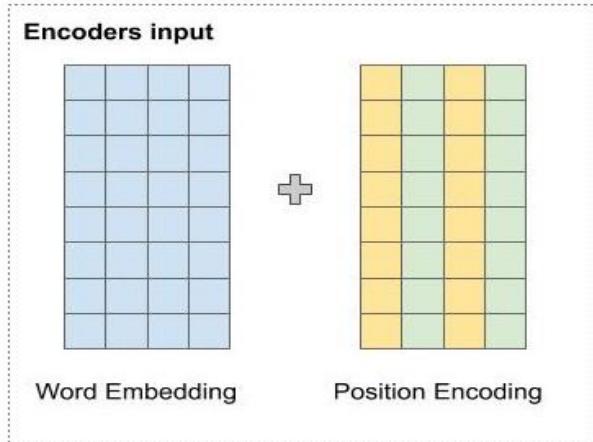
Trong các mạng nơ-ron, có một lớp nhúng từ (Word Embedding Layer) được sử dụng để giải quyết vấn đề này. Các từ sẽ được đi qua lớp nhúng và embedding layer sẽ học các mối quan hệ của từ trong câu sau đó lấy ra một biểu diễn vector đã được học cho mỗi từ. Mỗi từ sẽ được ánh xạ sang một vector có các giá trị liên tục để biểu diễn từ đó.

Ví dụ, các từ như “học sinh” và “giáo viên” có thể được biểu diễn bằng các vector khác nhau nhưng vẫn duy trì được mối quan hệ giữa chúng. Điều này cho phép mô hình học các đặc trưng của từ và sử dụng chúng để thực hiện các nhiệm vụ như phân loại văn bản, dịch máy, phân tích cảm xúc, nhận dạng thực thể, ... .

#### *2.1.3.2. Positional Encoding*

Trước khi đi vào Encoder, do cơ chế xử lý song song các từ trong câu của kiến trúc Transformer, nếu không biết được vị trí của các từ trong câu thì mô hình sẽ gặp khó khăn trong việc xử lý [1]. Để giải quyết vấn đề này, một cơ chế rất thú vị gọi là Positional Encoding, được sử dụng để đánh dấu vị trí của các từ trong câu và truyền vào pha Encoder.

Sau khi các từ được biểu diễn thông qua ma trận Word Embedding (với số dòng bằng kích thước của tập từ vựng và số cột là số đặc trưng), vị trí của các từ sẽ được mã hóa bằng một ma trận có cùng kích thước với ma trận Word Embedding. Sau đó, ma trận Positional Encoding này sẽ được cộng trực tiếp vào ma trận Word Embedding, tương tự như phép cộng hai ma trận, như minh họa trong Hình 2.2.



**Hình 2.2. Encoders Input[1]**

Positional Encoding sẽ tính giá trị cho các vị trí lẻ bằng hàm *Cos* và cho các vị trí chẵn bằng hàm *Sin*, công thức được minh họa như bên dưới:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Trong đó:

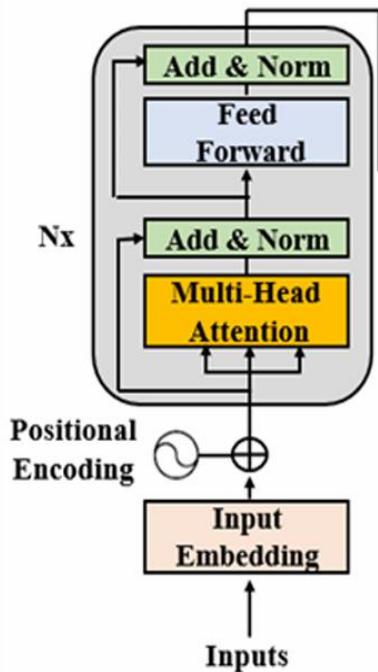
- pos: vị trí của một từ trong câu đầu vào.
- $d_{model}$ : số chiều của không gian Embedding đầu ra.
- $PE_{(pos,2i)}$  /  $PE_{(pos,2i+1)}$ : biểu diễn giá trị Positional Encoding tại vị trí pos trong câu đầu vào và tại cột thứ i của ma trận vị trí.
- i: được sử dụng để ánh xạ tới các chỉ số cột  $0 \leq i \leq \frac{d_{model}}{2}$  với một giá trị i được ánh xạ cho cả hàm Sin và Cos.

Bằng cách sử dụng các hàm *Sin* và *Cos* có các bước sóng khác nhau ở các chiều khác nhau, điều này giúp mỗi chiều của Positional Encoding có bước sóng riêng biệt, từ đó có thể phân biệt được các vị trí khác nhau trong câu.

#### 2.1.3.3. Encoder

Encoder có nhiệm vụ phát hiện và nắm bắt mối quan hệ giữa các từ trong câu. Mỗi Encoder bao gồm hai thành phần cốt lõi là Multi-Headed Attention và Feed Forward Network. Bên cạnh đó, Encoder còn có Skip Connection và Normalization

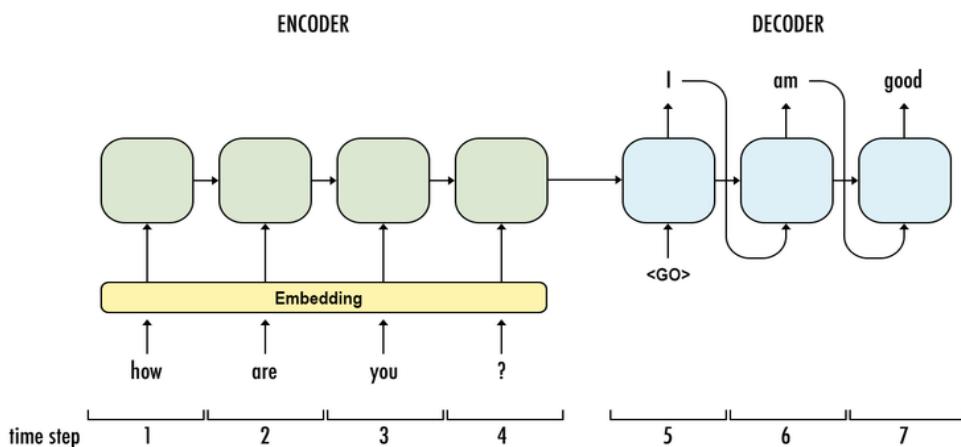
để đảm bảo tính ổn định và hiệu quả trong quá trình xử lý. Hình 2.3 bên dưới thể hiện cấu trúc của Encoder.



Hình 2.3. Các thành phần của Encoder[1]

#### 2.1.3.4. Decoder

##### a) Giới thiệu về Decoder



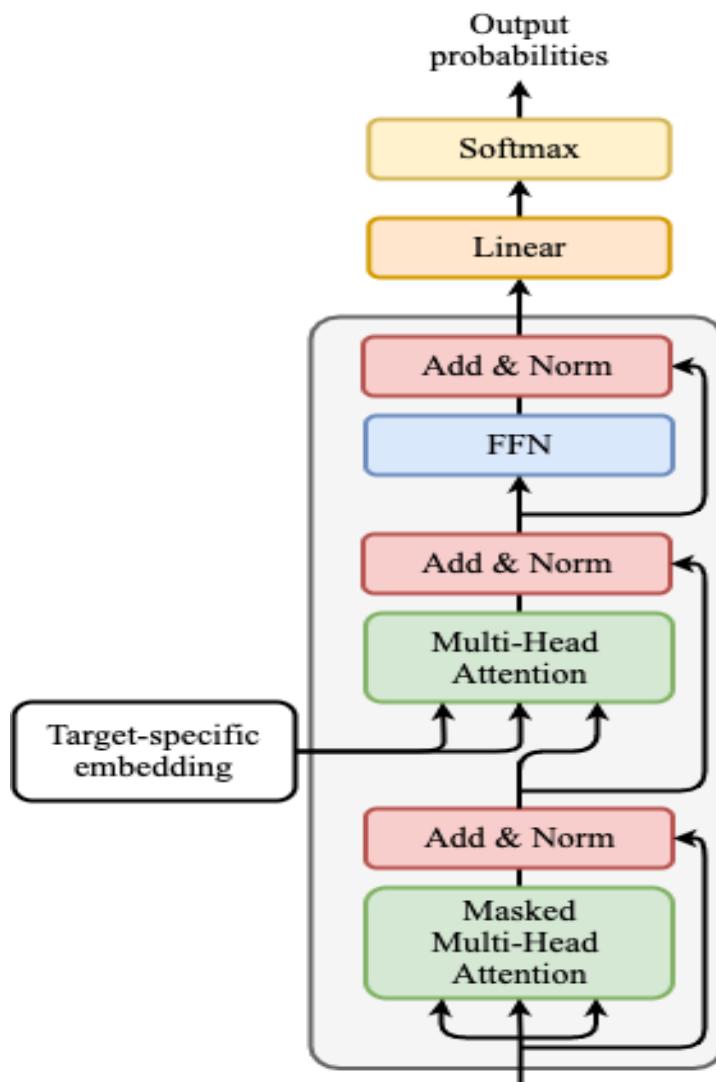
Hình 2.4. Quá trình xử lý một câu đối với Encoder và Decoder[1]

Decoder là một thành phần sử dụng biểu diễn ẩn (features) do Encoder tạo ra để dự đoán phân phối xác suất của đầu ra. Kết quả đầu ra của Decoder phụ thuộc vào đầu vào của Encoder; nó có thể là một nhãn đối với mô hình phân loại hoặc là một chuỗi theo thứ tự thời gian đối với mô hình Sequence To Sequence (Seq2Seq).

## b) Quá trình Decoder

Ở Hình 2.5, mỗi Decoder sẽ gồm 3 thành phần cốt lõi:

- Multi-Head Attention
- Feed-forward Neural Network
- Masked Multi-Head Attention



**Hình 2.5. Các thành phần của Decoder[1]**

Ngoài ra còn có Skip Connection và Normalization để đảm bảo tính ổn định và hiệu quả trong quá trình xử lý.

Để tránh việc mô hình "nhìn thấy" các từ phía sau trong câu khi dự đoán từ tiếp theo (trong quá trình huấn luyện), Masked Multi-Head Attention là thành phần quan trọng đầu tiên trong Decoder [1]. Thành phần này sử dụng một masking đặc biệt, chỉ

cho phép mỗi từ trong câu đầu ra tập trung vào các từ trước đó (bao gồm từ hiện tại) và không thể "nhìn thấy" các từ phía sau trong câu. Điều này đảm bảo quá trình dự đoán từ tiếp theo không bị ảnh hưởng bởi thông tin từ các từ chưa được dự đoán.

Tương tự như ở Encoder, thành phần Multi-Head Attention trong Decoder cho phép mỗi từ trong câu đầu ra tập trung vào các từ khác trong cùng câu đó, từ đó tính toán trọng số của các từ này để hình thành vector biểu diễn. Mục tiêu của Multi-Head Attention trong Decoder cũng giống như ở Encoder, đó là giúp mô hình hiểu được mối liên kết giữa các từ trong câu đầu ra, đảm bảo rằng các từ được sắp xếp đúng thứ tự và ngữ cảnh phù hợp.

Sau khi hoàn tất việc tính toán trọng số cho các từ trong câu đầu vào, vector biểu diễn của từ tiếp theo trong câu đầu ra được tính toán bằng cách kết hợp trọng số của các vector biểu diễn từ câu đầu vào và các vector của các từ đã được dự đoán trước đó trong câu đầu ra. Sau khi quá trình tính toán này hoàn tất, các vector sẽ được đưa vào Feed-Forward Neural Network để tiếp tục xử lý.

Trong Decoder, phần Feed-Forward Neural Network có nhiệm vụ tương tự như ở Encoder, đó là xử lý các vector biểu diễn của các từ và trích xuất những đặc trưng quan trọng.

Trong quá trình này, Transformer học cách dự đoán từ tiếp theo trong câu đầu ra dựa trên ngữ cảnh của câu đầu vào và các từ đã được xử lý trong câu đầu ra. Khi mô hình đã học được các trọng số và đặc trưng ngữ cảnh quan trọng của các từ trong câu, nó có khả năng dự đoán chính xác các từ tiếp theo trong câu đầu ra.

Sau khi các từ trong câu đầu ra được dự đoán, Decoder của Transformer sẽ tạo ra các vector biểu diễn cho toàn bộ câu đầu ra. Vector này sau đó được sử dụng để tính toán xác suất cho toàn bộ câu và đưa ra dự đoán cuối cùng.

Tóm lại, nhiệm vụ của Decoder trong Transformer là tạo ra chuỗi các vector biểu diễn cho các từ trong câu đầu ra, dựa trên vector biểu diễn của câu đầu vào và thông tin ngữ cảnh từ các lớp Encoder trước đó [1]. Các từ trong câu đầu ra được dự đoán dựa trên các đặc trưng quan trọng của câu đầu vào và ngữ cảnh đã được dự đoán trong câu đầu ra. Decoder giúp mô hình nắm bắt mối quan hệ giữa các từ trong câu

đầu ra, từ đó đưa ra dự đoán chính xác cho toàn bộ câu, đảm bảo rằng các từ được kết nối đúng thứ tự và ngữ cảnh.

#### 2.1.3.5. Cơ chế Attention

Trong những năm gần đây, cơ chế Attention đã mang lại những bước tiến lớn trong các mô hình học sâu, đặc biệt là trong các bài toán liên quan đến dữ liệu chuỗi như dịch máy, nhận dạng thực thể, và các mô hình ngôn ngữ. Cơ chế này cho phép mô hình học được mức độ quan trọng của mỗi phần tử trong chuỗi dữ liệu, từ đó có thể tập trung vào các phần quan trọng hơn.

Cơ chế Attention hoạt động bằng cách tính toán một trung bình có trọng số của các phần tử trong chuỗi, với các trọng số được điều chỉnh động dựa trên một truy vấn đầu vào và các khóa tương ứng của các phần tử. Mục tiêu của cơ chế này là tính toán trung bình các đặc trưng của nhiều phần tử, nhưng thay vì gán trọng số đồng đều cho tất cả các phần tử, nó điều chỉnh trọng số dựa trên giá trị thực tế của chúng. Nói cách khác, cơ chế này sử dụng thông tin đầu vào để quyết định mức độ “chú ý” cần thiết cho từng phần tử so với các phần tử khác. Một cơ chế Attention thường bao gồm ba phần chính:

- Truy vấn (Query): Truy vấn là một vector đặc trưng mô tả mục tiêu mà chúng ta đang tìm kiếm trong chuỗi, tức là điều chúng ta muốn chú ý đến.
- Khóa (Key): Mỗi phần tử đầu vào có một khóa tương ứng, là một vector đặc trưng mô tả nội dung hoặc tầm quan trọng của phần tử đó.
- Giá trị (Value): Mỗi phần tử đầu vào cũng có một vector giá trị, là thông tin mà chúng ta muốn tính toán.

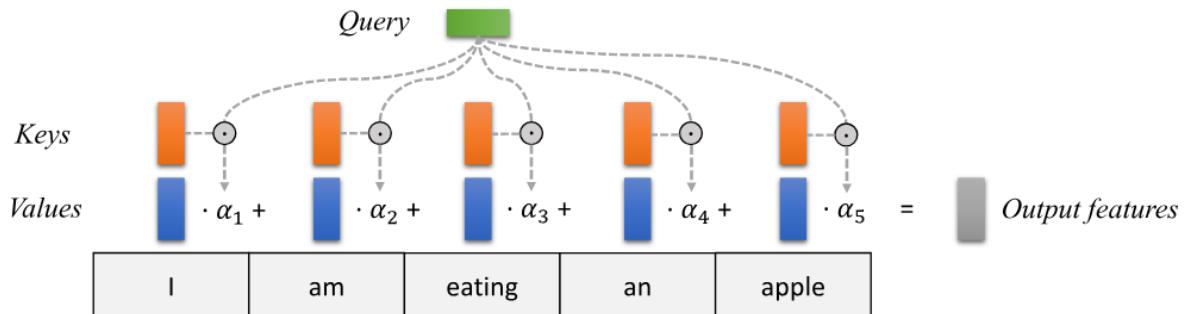
Các trọng số trong cơ chế Attention được tính toán bằng cách sử dụng hàm Softmax trên tất cả các đầu ra của phép toán điểm (dot product) giữa Query và Key. Kết quả là các vector giá trị (Value) sẽ được gán trọng số cao hơn nếu khóa tương ứng của chúng có độ tương đồng (được đo bằng dot product) gần gũi hơn với truy vấn (Query). Điều này có nghĩa là những phần tử trong chuỗi có Key gần gũi với Query sẽ có ảnh hưởng lớn hơn đến đầu ra của mô hình. Sau đây chính là công thức cho hàm Softmax:

$$\alpha_i = \frac{\exp(f_{attn}(key_i, query))}{\sum_j \exp(f_{attn}(key_j, query))}, \quad out = \sum_i \alpha_i \cdot value_i$$

Trong đó:

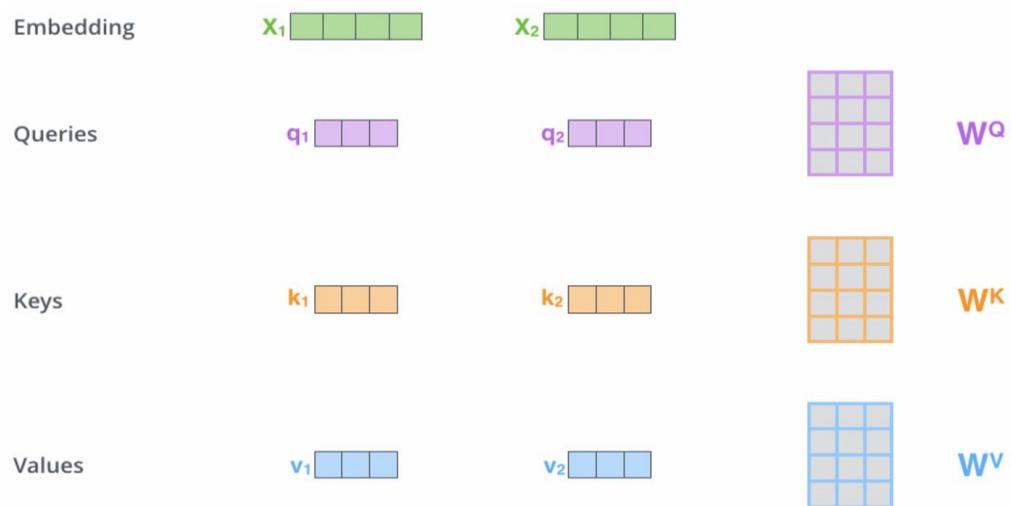
- $f_{attn}$ : là hàm điểm (dot product).
  - $\alpha_i$ : là đầu ra tại vị trí thứ  $i$ .
  - $out$ : là đầu ra cuối cùng.

Cơ chế self-Attention được trực quan hóa trên một câu như Hình 2.6:



**Hình 2.6. Cơ chế self-Attention[1]**

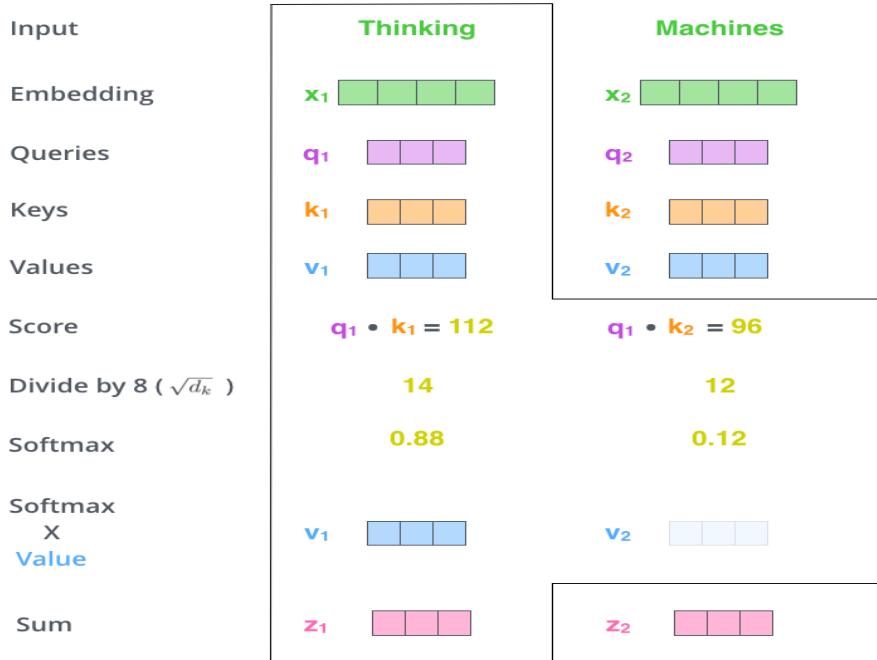
Cơ chế Self-Attention, như được mô tả trong Hình 2.6, là một phần quan trọng trong mô hình Transformer. Nó giúp mô hình hiểu được các mối quan hệ giữa các từ trong câu, điều này rất quan trọng để xử lý ngữ nghĩa trong các câu phức tạp [1]. Các hình minh họa dưới đây giúp giải thích rõ hơn về cách hoạt động của cơ chế này.



Hình 2.7. Cơ chế self-Attention trong transformer[1]

Trong cơ chế Self-Attention, mỗi từ trong câu sẽ được ánh xạ vào ba vector: Key, Query, và Value thông qua các ma trận tương ứng. Cụ thể, mỗi từ đầu vào sẽ

được nhân với các ma trận Key, Query, và Value để tạo ra các vector Key, Query, và Value tương ứng.



**Hình 2.8. Cơ chế self-Attention trong transformer tiếp theo[1]**

Tiếp theo, vector Query sẽ được so sánh với tất cả các vector Key bằng cách sử dụng phép toán Scaled Dot Product. Trong đó, vector Query được nhân với từng vector Key để tính toán mức độ tương đồng (trọng số). Những trọng số này sau đó sẽ được chuẩn hóa bằng hàm Softmax, giúp xác định trọng số tương đối của mỗi vector Key so với Query. Hàm Softmax sẽ quyết định mức độ "chú ý" mà mô hình cần dành cho từng vector Key, và chuẩn hóa các trọng số sao cho tổng của chúng bằng 1.

Cuối cùng, kết quả của phép nhân giữa vector Query và Key, sau khi qua hàm Softmax, sẽ được nhân với vector Value tương ứng. Các giá trị này sẽ được cộng lại (tổng có trọng số), từ đó tạo ra đầu ra của cơ chế Self-Attention.

#### 2.1.3.6. Hoạt động của Scaled Dot Product Attention

Hàm Scaled Dot Product Attention có mục đích tạo ra một cơ chế Attention mà giúp mỗi phần tử trong câu có thể "chú ý" đến tất cả các phần tử khác. Cơ chế này rất quan trọng trong việc xác định mối quan hệ giữa các từ trong câu, giúp mô hình học được ngữ cảnh của từng từ trong mối quan hệ với các từ khác. Không gian biểu diễn cho các ma trận trọng số  $Q$ ,  $K$ ,  $V$  được thể hiện như sau:

$$Queries: Q \in R^{T \times d_k}$$

$$Keys: K \in R^{T \times d_k}$$

$$Values: V \in R^{T \times d_v}$$

Trong đó:

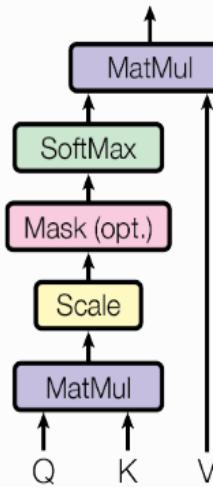
- $T$  là độ dài của câu.
- $d_k$ : là chiều ẩn cho vector Query, Key.
- $d_v$ : là chiều ẩn cho vector Value.

Chúng ta sẽ bỏ qua số chiều của batch để đơn giản hóa. Giá trị Attention giữa hai phần tử  $i$  và  $j$  được tính dựa trên mức độ tương đồng giữa Query  $Q_i$  và Key  $K_j$ . Sử dụng tích vô hướng (Dot Product) làm đo lường độ tương đồng, trong toán học, chúng ta tính toán Scaled Dot Product Attention theo công thức:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Phép nhân giữa ma trận  $Q$  với ma trận chuyển vị  $K$  thực hiện phép tính tích vô hướng giữa mỗi cặp Query và Key. Kết quả thu được là một ma trận kích thước  $T \times T$ , trong đó mỗi hàng thể hiện các giá trị Logits Attention từ một phần tử  $i$  tới tất cả các phần tử khác trong câu. Sau đó, ta áp dụng hàm Softmax lên các giá trị Logits này để chuẩn hóa chúng thành các trọng số. Quá trình chuẩn hóa này giúp đảm bảo rằng tổng của các trọng số trong mỗi hàng bằng 1, đồng thời xác định mức độ “chú ý” mà mỗi phần tử  $i$  dành cho các phần tử khác trong câu. Cuối cùng, những trọng số này sẽ được nhân với các vector Value tương ứng. Mỗi giá trị trong ma trận Value sau khi nhân với trọng số sẽ tạo ra một ma trận đầu ra, trong đó mỗi phần tử là một giá trị có trọng số. Các giá trị này sẽ được cộng lại (tổng có trọng số), từ đó tạo ra đầu ra của cơ chế Self-Attention. Một góc nhìn khác về cơ chế Attention này được minh họa qua đồ thị tính toán trong Hình 2.9 dưới đây:

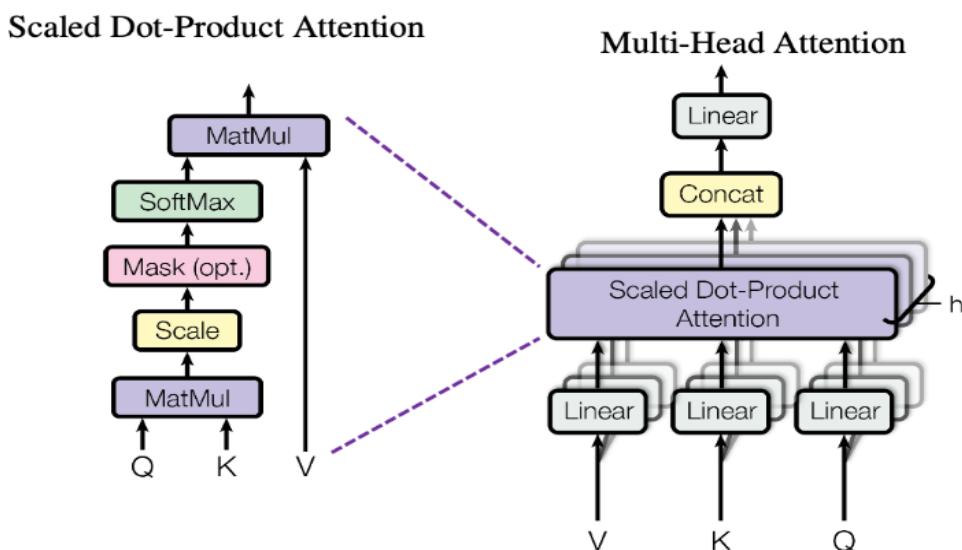
## Scaled Dot-Product Attention



**Hình 2.9. Scaled Dot-Product Attention[1]**

### 2.1.3.7. Thành phần Multi-Head Attention

Cơ chế Attention dựa trên Scaled Dot Product giúp mạng nơ-ron tập trung vào một phần cụ thể của chuỗi dữ liệu. Tuy nhiên, chỉ sử dụng một phép tính trung bình có trọng số duy nhất không đủ để mô hình khai thác hết các mối quan hệ phức tạp giữa các phần tử trong chuỗi, đặc biệt là khi một phần tử cần chú ý đến nhiều khía cạnh khác nhau của các phần tử khác. Để giải quyết vấn đề này, chúng ta mở rộng cơ chế Attention thành Multi-Head Attention.



**Hình 2.10. Cơ chế Multi-head Attention[1]**

Multi-Head Attention cho phép mô hình sử dụng nhiều bộ Query-Key-Value khác nhau để xử lý đồng thời nhiều khía cạnh của dữ liệu. Cụ thể, các ma trận Query, Key, và Value ban đầu sẽ được chia thành các ma trận con, gọi là Sub-Query, Sub-Key, và Sub-Value. Mỗi cặp Sub-Query, Sub-Key, và Sub-Value sẽ được đưa qua quá trình Scaled Dot Product Attention độc lập. Kết quả của các quá trình này là một loạt các vector Attention, mỗi vector tương ứng với một head trong Multi-Head Attention. Cuối cùng, nối các kết quả này lại với nhau và áp dụng một ma trận trọng số để kết hợp chúng thành đầu ra cuối cùng của lớp Attention. Có thể biểu diễn phép toán này như bên dưới:

$$\begin{aligned} \text{Multihead}(Q, K, V) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \\ \text{where } \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

Đây là lớp Multi-Head Attention với các tham số có thể học được. Quan sát công thức bên dưới:

$$W_{1\dots h}^Q \in R^{D \times d_k} W_{1\dots h}^K \in R^{D \times d_k} W_{1\dots h}^V \in R^{D \times d_v} W^O \in R^{h \cdot d_v \times d_{out}} D$$

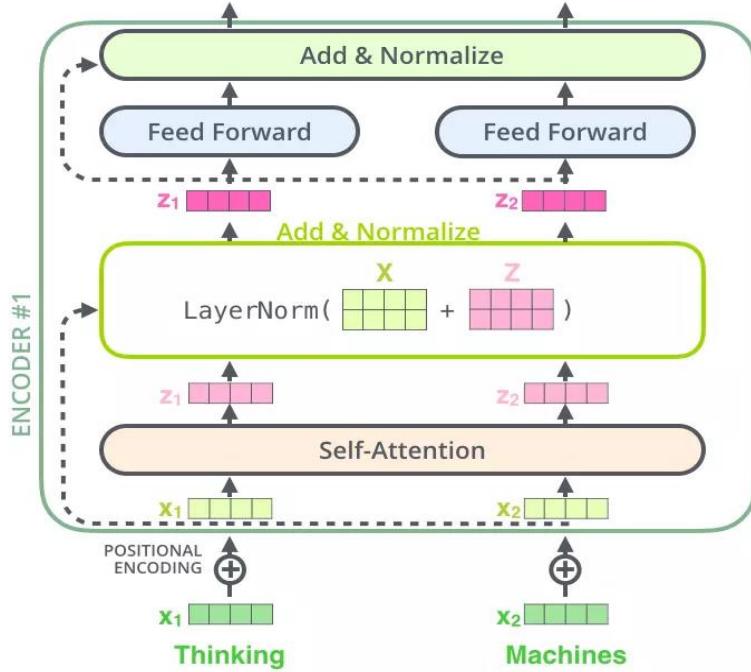
Việc sử dụng Multi-Head Attention giúp mô hình học được các mối quan hệ từ nhiều góc độ khác nhau, tăng cường khả năng hiểu và phân tích dữ liệu đầu vào [1]. Điều này đặc biệt hữu ích trong các bài toán phức tạp như dịch máy, phân loại văn bản, và nhận dạng thực thể, nơi mà một từ có thể mang nhiều nghĩa khác nhau tùy theo ngữ cảnh.

#### 2.1.3.8. Thành phần Add và Normalize

Add và Normalize là một kỹ thuật then chốt trong kiến trúc Transformer, giúp mô hình không chỉ học được thông tin mới một cách hiệu quả mà còn bảo toàn những đặc trưng quan trọng từ đầu vào ban đầu. Kỹ thuật này được áp dụng sau mỗi lần mô hình tính toán đầu ra từ lớp Attention hoặc Feed-Forward Neural Network, nhằm tối ưu hóa quá trình huấn luyện và cải thiện khả năng diễn giải của mô hình. Kỹ thuật này bao gồm hai bước:

Bước 1: Add (Cộng) - Đầu ra từ lớp Attention hoặc Feed-Forward Neural Network sẽ được cộng với đầu vào ban đầu của lớp đó. Mục tiêu của bước này là giữ lại thông tin nguyên bản từ đầu vào, đồng thời kết hợp với thông tin đã được mô hình học thêm ở lớp hiện tại. Điều này đảm bảo rằng không có thông tin quan trọng nào bị

mất đi trong quá trình truyền qua các lớp. Công thức cho bước này được minh họa trong Hình 17, trong đó mỗi vector đầu ra được cộng với vector đầu vào tương ứng để tạo ra một vector mới, phong phú hơn về mặt thông tin.



Hình 2.11. Biểu diễn trực quan cho 2 bước add và normalize[1]

Trong đó:

- X là đầu vào ban đầu của lớp đó.
- Z là đầu ra tại mỗi lớp Attention hoặc feedforward.

Bước 2: Normalize (Chuẩn hóa) - Sau khi thực hiện phép cộng, vector mới sẽ được chuẩn hóa để điều chỉnh lại các giá trị, tránh tình trạng gradient biến mất hoặc trở nên quá lớn. Bước chuẩn hóa này giúp mô hình hoạt động ổn định hơn và tăng cường khả năng học tập, bằng cách giữ các giá trị trong phạm vi tối ưu. Quá trình chuẩn hóa cũng làm giảm sự khác biệt giữa các vector, khiến việc huấn luyện mô hình trở nên dễ dàng và hiệu quả hơn.

Khi kết hợp hai bước Add và Normalize, mô hình Transformer không chỉ học được thông tin về vị trí và thứ tự các từ trong câu, mà còn giúp quá trình huấn luyện diễn ra nhanh chóng, tiết kiệm tài nguyên và ngăn ngừa các vấn đề như trọng số tăng đột biến (weight explosion) hoặc trọng số giảm quá nhỏ (vanishing weights) trong quá trình huấn luyện.

Với phương pháp Batch Normalization, dữ liệu sẽ được chuẩn hóa sao cho trung bình bằng 0 và phương sai khoảng 1, trong đó các giá trị đầu ra của một batch dữ liệu được chuẩn hóa theo trung bình và độ lệch chuẩn của batch đó [1].

#### 2.1.3.9. Thành phần Feed Forward

Feed Forward, còn gọi là Position-Wise Feed-Forward Network (Mạng chuyên tiếp theo vị trí), là một thành phần quan trọng khác trong kiến trúc Transformer [1]. Cơ chế Feed Forward trong Transformer cung cấp cấu trúc tuyến tính giúp mô hình xử lý từng phần tử đầu vào độc lập nhưng vẫn giữ mối liên hệ với các phần tử khác trong chuỗi, tương tự như hai lớp Convolutional 1x1 để trích xuất đặc trưng mà không tăng độ phức tạp đáng kể. Lựa chọn lớp Fully-Connected Feed Forward và Self-Attention làm thành phần chính của Transformer giúp xử lý chuỗi dài hiệu quả, vượt trội hơn so với CNN hay RNN truyền thống, nhờ khả năng duy trì thông tin về thứ tự và ngữ cảnh mà các mô hình cũ hay gặp khó khăn.

#### 2.1.3.10. Masked Multi-Head Attention

Masked Multi-Head Attention là một phiên bản đặc biệt của Multi-Head Attention mà chúng ta đã thảo luận trước đó, với sự khác biệt quan trọng là nó ngăn chặn việc sử dụng các từ trong tương lai khi dự đoán các từ tiếp theo. Thành phần này rất quan trọng trong quá trình giải mã của Transformer, nơi nó giúp tạo ra các trọng số Attention cho từng từ trong câu đang được giải mã. Những trọng số này quyết định mức độ quan trọng của từng từ khi dự đoán từ tiếp theo.

Dưới đây là các bước của Masked Multi-Head Attention:

Bước 1: Tạo Mask - Trước khi tính toán Attention, mô hình sẽ tạo ra một ma trận mask vuông có kích thước  $[L, L]$  tương ứng với chiều dài của chuỗi đầu vào. Mục đích của mask là che giấu các từ trong tương lai. Ví dụ, nếu ta đang dự đoán từ thứ  $i$ , các từ từ vị trí  $i + 1$  đến  $L$  sẽ bị che giấu bằng cách gán giá trị không trong ma trận mask.

Bước 2: Tính Attention Scores - Sau khi tạo ma trận mask, mô hình tính toán Attention scores bằng cách nhân các vector Query ( $Q$ ) với vector Key ( $K$ ) theo công thức được mô tả như bên dưới:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Trong đó,  $Q$  là ma trận Query có kích thước  $(L, d_{model})$ ,  $K$  là ma trận Key có kích thước  $(seq_L, d_{model})$ , và  $d_k$  là kích thước của mỗi vector Query/Key. Việc chia tỉ lệ cho  $d_k$  giúp kiểm soát độ lớn của Attention scores, đảm bảo rằng các giá trị này không trở nên quá lớn.

Bước 3: Áp dụng Mask - Sau khi tính toán Attention scores, ta áp dụng ma trận mask bằng cách nhân phần tử-wise giữa Attention scores và ma trận mask. Điều này giúp gán giá trị không cho các vị trí tương ứng với các từ trong tương lai, đảm bảo rằng các từ này không được sử dụng trong quá trình dự đoán từ tiếp theo.

Bước 4: Tính Weighted Sum - Cuối cùng, ta tính trọng số weighted sum của vector Value ( $V$ ) bằng cách nhân Attention scores với ma trận Value và cộng tổng các giá trị. Kết quả là một vector weighted sum, chứa thông tin quan trọng từ các từ đã quan sát trong quá trình tính toán Attention.

## 2.2. GIỚI THIỆU VỀ MÔ HÌNH BERT

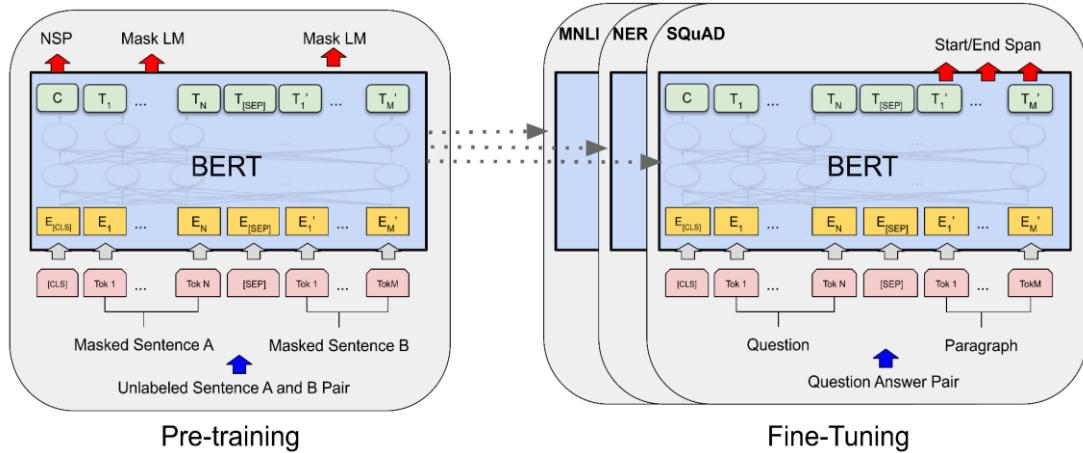
### 2.2.1. Giới thiệu sơ lược về BERT

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình NLP dựa trên kỹ thuật Transformer, được Google công bố vào tháng 11 năm 2018 do Jacob Devlin và cộng sự phát triển[9]. BERT có khả năng xử lý ngôn ngữ theo cả hai chiều trái và phải, giúp cải thiện khả năng hiểu ngữ cảnh. Mô hình này được thiết kế để huấn luyện trước các biểu diễn từ và có thể được điều chỉnh với thêm một tầng đầu ra để thực hiện các nhiệm vụ NLP hiện đại như suy luận ngôn ngữ và trả lời câu hỏi mà không cần thay đổi lớn trong kiến trúc.

Mô hình đạt được kết quả tiên tiến nhất trên 11 việc về xử lý ngôn ngữ tự nhiên, bao gồm cả việc đẩy điểm GLUE lên 80,5% (7,7 điểm cải thiện tuyệt đối), độ chính xác của MultiNLI đến 86,7% (4,6% tuyệt đối cải thiện), trả lời câu hỏi SQuAD v1.1 mô hình đạt kết quả trên tập F1 lên 93,2 và kết quả với tập test trên bộ dữ liệu SQuAD v2.0 F1 lên 83,1 [9].

## 2.2.2. Quá trình pre-train BERT

Quá trình pre-train BERT được huấn luyện với các liệu không dán nhãn, còn đối với quá trình fine-tuning, mô hình BERT đầu tiên được khởi tạo với tất cả các tham số trong quá trình pre-train và các tham số này được điều chỉnh với dữ liệu được dán nhãn từ các Downstream Tasks [9]. Mỗi Downstream Task có các mô hình tinh chỉnh riêng biệt, mặc dù chúng được khởi tạo với cùng các tham số trong quá trình pre-train.



**Hình 2.12. Quá trình pre-train và fine-tuning của mô hình BERT[9]**

Trong quá trình pre-training, BERT sử dụng hai nhiệm vụ chính:

- Masked Language Model (MLM): Trong nhiệm vụ này, một số từ trong câu được ngẫu nhiên che giấu (masked), và mục tiêu là dự đoán các từ đó dựa trên ngữ cảnh của các từ xung quanh. Việc này giúp mô hình học được cách hiểu ngữ cảnh và mối quan hệ giữa các từ trong câu.

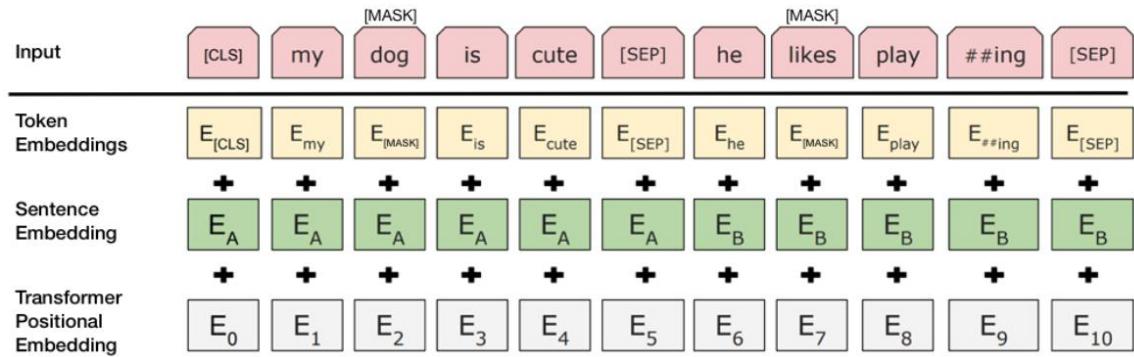
- Next Sentence Prediction (NSP): Nhiệm vụ này yêu cầu mô hình xác định xem một câu có phải là câu tiếp theo của một câu khác hay không. Điều này giúp mô hình hiểu được mối quan hệ giữa các câu, rất quan trọng trong việc xử lý các tác vụ như trả lời câu hỏi và suy luận ngôn ngữ.

## 2.2.3. Cấu tạo của BERT

BERT sử dụng các token đặc biệt [CLS] và [SEP] để xử lý các câu và đoạn văn.

- Token [CLS]: Được đặt ở đầu mỗi chuỗi đầu vào và đại diện cho toàn bộ văn bản. Đầu ra từ token này thường được sử dụng cho các tác vụ phân loại như phân loại cảm xúc, suy luận ngôn ngữ, và các nhiệm vụ dựa trên văn bản khác.

- Token [SEP]: Dùng để phân tách các câu hoặc đoạn văn trong chuỗi đầu vào. Trong trường hợp BERT xử lý cặp câu (sentence A và sentence B), token [SEP] sẽ được đặt giữa hai câu để đánh dấu sự kết thúc của câu đầu tiên và bắt đầu của câu thứ hai. Token này cũng được đặt ở cuối chuỗi đầu vào để biểu thị sự kết thúc của chuỗi.



Hình 2.13. Biểu diễn đầu vào của BERT[9]

Tóm lại, với các token [CLS] và [SEP], cùng các loại embeddings, BERT có thể phân tích các đoạn văn bản theo hướng hai chiều một cách hiệu quả để giải quyết các tác vụ NLP phức tạp [9].

Mô hình BERT chỉ có thành phần Encoder được chia thành 2 phiên bản chính dựa theo kích thước [9]:

- BERTBASE (L=12, H=768, A=12): Tổng tham số 110 triệu.
- BERTLARGE (L=24, H=1024, A=16): Tổng tham số 340 triệu.

Hiện nay có rất nhiều phiên bản khác nhau của BERT đã ra đời, các phiên bản này đều có một điểm chung đó là đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở ba tham số [9].

Các thành phần chính của mô hình Transformer bao gồm:

- L (Số lượng các khôi Encoder Layer): Lượng lớp Encoder giúp mô hình học ngữ cảnh song hướng, tăng cường khả năng hiểu mối quan hệ giữa các từ.
- H (Kích thước của Embedding Vector hay Hidden Size): Kích thước của vector biểu diễn từ, quyết định khả năng biểu diễn thông tin của mô hình.

- A (Số lượng Attention Head trong Multi-Head Attention): Số lượng Head giúp mô hình tính toán Self-Attention độc lập, học nhiều mối quan hệ ngữ cảnh khác nhau, cải thiện khả năng hiểu ngữ nghĩa.

### **2.3. GIỚI THIỆU VỀ MÔ HÌNH T5**

#### **2.3.1 Giới thiệu sơ lược về T5**

T5 - (viết tắt của cụm từ của Text-to-text Transfer Transformers) là một mô hình học sâu do Google phát triển và công bố vào năm 2020, dựa trên kiến trúc Transformers. Mô hình này có đặc điểm đặc biệt là chuyển tất cả các nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP) thành bài toán text-to-text. Điều này giúp cho T5 trở thành một mô hình linh hoạt và có thể áp dụng nhiều nhiệm vụ khác nhau mà không cần phải điều chỉnh kiến trúc của mô hình quá nhiều.

T5 được huấn luyện với mục tiêu tối ưu hóa các tác vụ như dịch máy, tóm tắt văn bản, trả lời câu hỏi, phân loại văn bản và nhiều nhiệm vụ khác. Điểm mạnh của T5 nằm ở khả năng tổng hợp kiến thức từ nhiều loại nhiệm vụ, điều này giúp mô hình đạt được hiệu quả cao trên các bộ dữ liệu benchmark nổi tiếng, chẳng hạn như GLUE, SQuAD và SuperGLUE. T5 đã chứng tỏ sự vượt trội trong nhiều thử thách, với khả năng chuyển giao học tập mạnh mẽ và hiệu quả, cho phép nó giải quyết nhiều bài toán NLP một cách đồng nhất mà không cần thay đổi cấu trúc cơ bản.

#### **2.3.2. Quá trình pre-train T5**

Quá trình pre-train của mô hình T5 được thực hiện với các dữ liệu không nhãn dán, tương tự như BERT. Tuy nhiên, T5 có sự khác biệt lớn ở cách thức thiết kế, khi mô hình dữ liệu này được huấn luyện với mục tiêu chuyển đổi tất cả các tác vụ NLP thành bài toán “text-to-text”. Điều này có nghĩa là mọi tác vụ, từ phân loại văn bản cho đến dịch máy hay tạo câu trả lời cho câu hỏi, đều được chuyển thành nhiệm vụ dự đoán một đoạn văn bản.

Quá trình Pre-train T5 gồm hai giai đoạn chính:

Giai đoạn 1: Tạo nhiệm vụ Text-to-Text - Trong quá trình pre-train, mô hình T5 được huấn luyện trên một loạt các nhiệm vụ NLP dưới dạng text-to-text. Điều này có nghĩa là đầu vào của mô hình là một văn bản, và đầu ra cũng là một văn bản.

Giai đoạn 2: Denosing Auto encoding - Mô hình T5 sử dụng một phương pháp được gọi là denoising auto encoding trong quá trình pre-train. Trong nhiệm vụ này, một phần của đầu vào (có thể là một câu hoặc một đoạn văn bản) sẽ bị che giấu hoặc bị nhiễu loạn, và mục tiêu của mô hình là dự đoán lại phần bị mất hoặc bị nhiễu loạn. Mục tiêu chính là giúp mô hình học cách phục hồi và hiểu nội dung văn bản.

### 2.3.3. Cấu tạo của T5

T5 không sử dụng các token đặc biệt như [CLS] và [SEP] trong cách thức xử lý dữ liệu, nhưng vẫn sử dụng một phương pháp tương tự để mã hóa và phân tách các thông tin.

Token <pad>: T5 sử dụng token <pad> để thực hiện việc padding cho các chuỗi văn bản sao cho chúng có độ dài bằng nhau trong quá trình huấn luyện. Token này chỉ được sử dụng để cân bằng độ dài chuỗi và nó không có giá trị ngữ nghĩa.

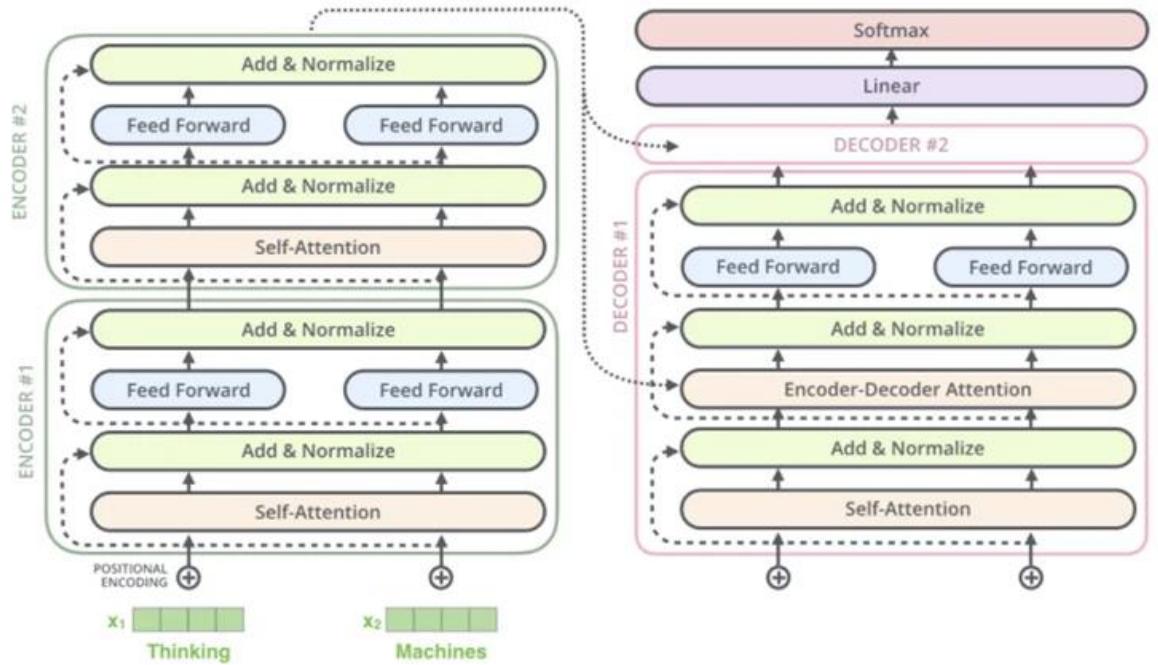
Token <extra\_id\_0>, <extra\_id\_1>, ... : các token này đóng vai trò xử lý các dữ liệu đặc biệt hoặc phân biệt các tác vụ, có thể được sử dụng để biểu diễn các phần khác nhau trong một nhiệm vụ text-to-text.

T5 sử dụng ba loại embeddings tương tự như BERT, nhưng có sự điều chỉnh phù hợp với mô hình text-to-text:

- Token Embeddings: Mỗi từ hoặc phần tử trong văn bản đầu vào được ánh xạ thành một vector nhúng (embedding) tương ứng, dựa trên mô hình subword tokenization.
- Position Embeddings: T5 sử dụng embeddings để mã hóa thông tin về vị trí của các từ trong chuỗi đầu vào. Điều này giúp mô hình nhận diện được thứ tự các từ trong câu.
- Task-Specific Embeddings: T5 có một tập hợp các embeddings đặc biệt để phân biệt các tác vụ khác nhau. Các nhiệm vụ như tóm tắt văn bản (summarization), dịch máy (translation), hoặc trả lời câu hỏi (question answering) đều sẽ có đầu vào bắt đầu với một token đặc biệt để chỉ rõ loại tác vụ mà mô hình cần thực hiện.

Với kiến trúc Transformer (Encoder và Decoder) đầy đủ, T5 sử dụng phần này để thực hiện các tác vụ chuyển đổi văn bản từ dạng này sang dạng khác. Kiến trúc này

giúp linh hoạt hơn trong việc thực hiện các tác vụ như sinh văn bản (text generation), tóm tắt (summarization), và dịch máy (translation).



Hình 2.14. Kiến trúc mô hình T5[17]

T5 có một số phiên bản khác nhau tùy vào các tham số như sau:

- T5-Base ( $L=12$ ,  $H=768$ ,  $A=12$ ): Tổng tham số là 220 triệu.
- T5-Large ( $L=24$ ,  $H=1024$ ,  $A=16$ ): Tổng tham số là 770 triệu.
- T5-3B ( $L=24$ ,  $H=1024$ ,  $A=32$ ): Tổng tham số là 3 tỷ.
- T5-11B ( $L=24$ ,  $H=1024$ ,  $A=64$ ): Tổng tham số là 11 tỷ.

Trong đó:

- $L$ : số lượng các lớp (layers) trong phần Encoder và Decoder.
- $H$ : kích thước của vector ẩn (hidden size).
- $A$ : số lượng attention heads trong mỗi layer của Multi-Head Attention.

## 2.4. SƠ LUẬC VỀ PRE-TRAIN VÀ FINE-TUNING

Large Language Models (LLMs) là các mô hình xác suất có khả năng hiểu và sinh ngôn ngữ tự nhiên dựa trên kiến thức được học từ các tập dữ liệu cực lớn. LLMs là một trong những ứng dụng thành công nhất của các mô hình Transformer [10]. Bên cạnh việc đẩy mạnh các ứng dụng xử lý ngôn ngữ tự nhiên như dịch thuật, chatbot, và

trợ lý ảo AI, LLMs còn được sử dụng trong nhiều lĩnh vực khác như chăm sóc sức khỏe và phát triển phần mềm.

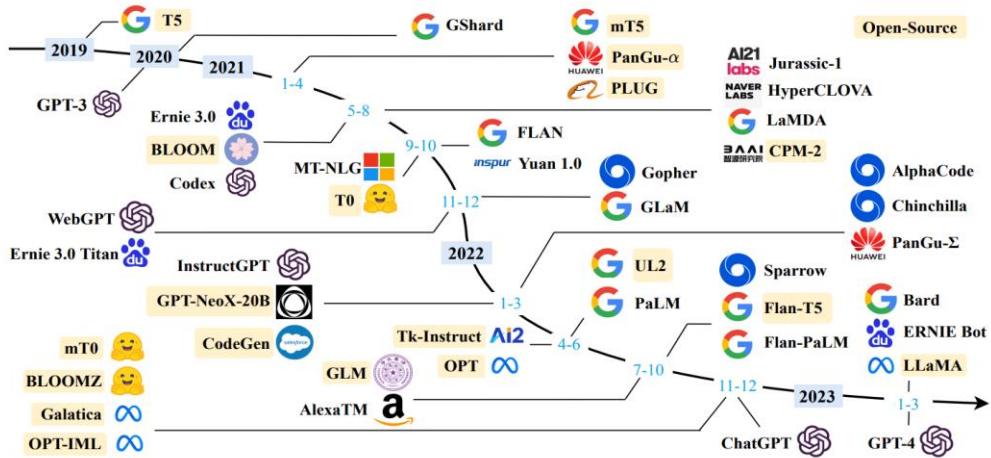


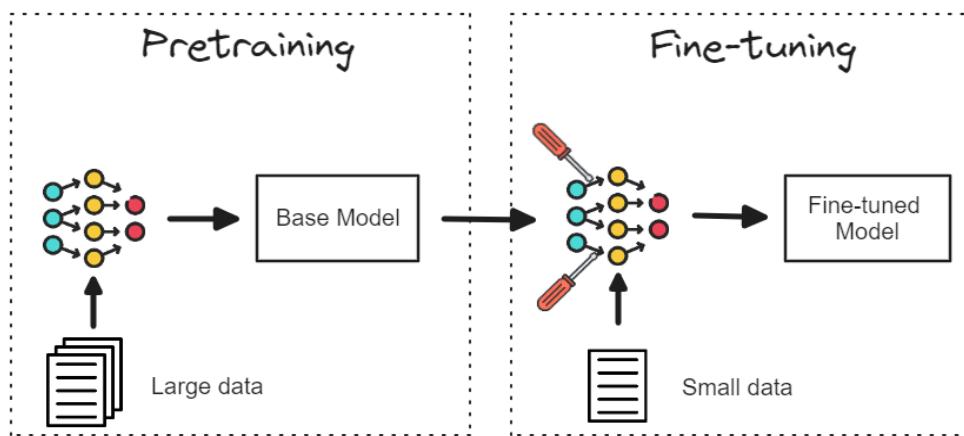
Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

### Hình 2.15. Sự hình thành các mô hình LLM qua các năm[10]

Mô hình tiền huấn luyện (Pre-trained model) là các mô hình học máy đã được huấn luyện trên một tập dữ liệu lớn cho một hoặc nhiều tác vụ như: dự đoán từ tiếp theo, phân loại cảm xúc, nhận diện thực thể, và nhiều tác vụ khác. Sau khi được tiền huấn luyện, các mô hình này có thể được tinh chỉnh (fine-tuned) để phù hợp với một nhiệm vụ cụ thể. Pre-trained models thường được sử dụng làm điểm khởi đầu cho việc phát triển các mô hình ML vì chúng cung cấp một tập hợp các trọng số và độ lệch ban đầu, giúp quá trình huấn luyện cho nhiệm vụ mới trở nên nhanh chóng và hiệu quả hơn.

Việc sử dụng mô hình tiền huấn luyện có nhiều lợi ích như tận dụng kiến thức và kinh nghiệm từ các mô hình đã được huấn luyện trước đó, tiết kiệm thời gian và tài nguyên, và cải thiện hiệu suất của mô hình [12]. Các mô hình này thường được huấn luyện trên các tập dữ liệu lớn và đa dạng, giúp nhận diện các mẫu và đặc trưng khác nhau một cách hiệu quả.

# Large Language Model



**Hình 2.16. Biểu diễn quá trình pre-train và fine-tuning[11]**

Fine-tuning (tinh chỉnh) trong học máy là một kỹ thuật quan trọng giúp điều chỉnh các mô hình đã được huấn luyện trước để phù hợp với các nhiệm vụ hoặc lĩnh vực sử dụng cụ thể [12]. Kỹ thuật này thuộc về khái niệm chuyển giao học (transfer learning), nơi mà kiến thức của mô hình đã học được từ dữ liệu lớn sẽ được áp dụng cho các nhiệm vụ nhỏ hơn. Bằng cách sử dụng các trọng số của mô hình đã được huấn luyện trước làm điểm khởi đầu, fine-tuning tiết kiệm thời gian và tài nguyên so với việc huấn luyện một mô hình từ đầu.

Quá trình fine-tuning bao gồm việc cập nhật các tham số của mô hình dựa trên một tập dữ liệu nhỏ hơn, thường cụ thể cho nhiệm vụ mong muốn. Các phương pháp fine-tuning có thể bao gồm cập nhật toàn bộ trọng số của mô hình, chỉ cập nhật một phần trọng số quan trọng nhất, hoặc thêm các lớp adapter mới vào mô hình. Nhờ đó, fine-tuning không chỉ giúp cải thiện độ chính xác của mô hình trên dữ liệu cụ thể mà còn giảm thiểu nguy cơ overfitting, cho phép mô hình tổng quát hóa tốt hơn trên dữ liệu mới.

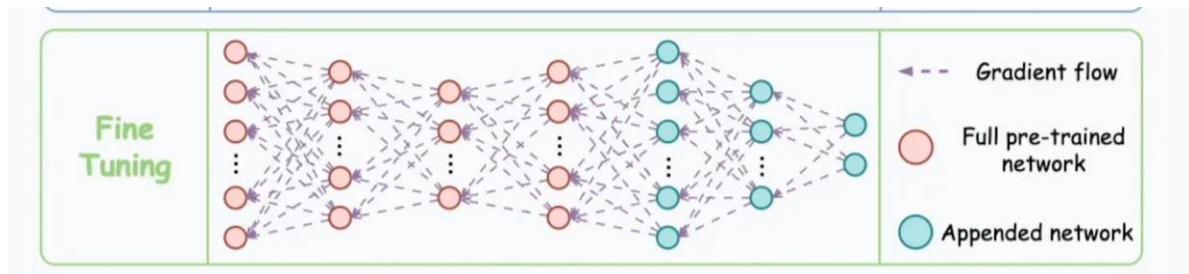
## 2.5. CÁC KỸ THUẬT FINE-TUNING

Có nhiều phương pháp để fine-tuning một mô hình Large Language Models (LLM), nhưng trong phần này, chúng tôi sẽ giới thiệu hai phương pháp chính và dễ tiếp cận.

### 2.5.1. Full Fine-Tuning

Tinh chỉnh đầy đủ là một trong những phương pháp chính để điều chỉnh mô hình ngôn ngữ lớn cho các mục tiêu cụ thể. Phương pháp này bao gồm việc huấn luyện toàn bộ mô hình trên dữ liệu dành riêng cho nhiệm vụ.

Điều này có nghĩa là tất cả các lớp của mô hình đều được điều chỉnh trong quá trình đào tạo. Tinh chỉnh đầy đủ đặc biệt hữu ích khi tập dữ liệu dành riêng cho nhiệm vụ lớn và khác biệt đáng kể so với dữ liệu trước khi mô hình được huấn luyện.



Hình 2.17. Mô tả về quá trình fine-tuning[12]

Việc fine-tuning toàn bộ mô hình giúp cải thiện hiệu suất cho các nhiệm vụ cụ thể nhưng có một số vấn đề cần lưu ý:

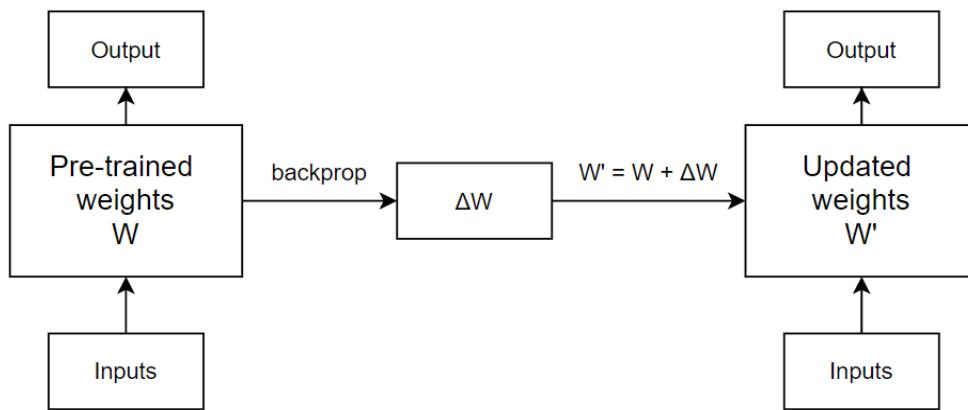
- Tốn chi phí tài nguyên: Fine-tuning toàn bộ tham số yêu cầu nhiều tài nguyên tính toán, đặc biệt với các mô hình lớn như Transformer, dẫn đến chi phí cao về thời gian và điện năng.
- Thời gian fine-tuning lâu: Quá trình fine-tuning trên các mô hình lớn có thể mất thời gian dài, khó triển khai nhanh cho các ứng dụng thực tế.
- Quên kiến thức đã học: Việc fine-tuning có thể dẫn đến hiện tượng "catastrophic forgetting", khiến mô hình mất khả năng thực hiện tốt các nhiệm vụ đã được huấn luyện từ trước.

### 2.5.2. Lora Fine-tuning

LoRA Fine-Tuning (Low-Rank Adaptation) là một phương pháp tinh chỉnh (fine-tuning) hiệu quả được giới thiệu bởi nhóm nghiên cứu Microsoft vào năm 2021 [12]. Đây là một kỹ thuật thuộc nhóm Parameter-efficient Fine-tuning (PEFT), tức là tinh chỉnh mô hình với sự tiết kiệm tài nguyên và tham số. LoRA đã trở thành một phương pháp phổ biến trong việc tinh chỉnh các mô hình ngôn ngữ lớn (large language

models), mô hình khuếch tán (diffusion models) như mô hình tạo hình ảnh, và các loại mô hình AI khác.

Mục tiêu chính của LoRA là tinh chỉnh mô hình hiệu quả mà không cần phải điều chỉnh tất cả các tham số của mô hình, giúp giảm bớt nhu cầu về bộ nhớ và tài nguyên tính toán [12]. Phương pháp này rất quan trọng vì quá trình tinh chỉnh toàn bộ tham số của mô hình lớn sẽ đòi hỏi rất nhiều tài nguyên (chủ yếu là GPU và bộ nhớ), làm cho việc tinh chỉnh trở nên khó khăn và tốn kém đối với nhiều người dùng. LoRA giúp giải quyết vấn đề này, cho phép tinh chỉnh các mô hình lớn một cách nhanh chóng và hiệu quả hơn mà không làm giảm chất lượng kết quả.



**Hình 2.18. Mô tả quá trình cập nhật trọng số Lora fine -tuning[12]**

Quá trình điều chỉnh trọng số của một lớp trong quá trình fine-tuning được minh họa trong hình trên. Trọng số đã được huấn luyện trước  $W$  của mô hình sẽ được chuyển đổi thành trọng số cập nhật  $W'$  dựa trên giá trị thay đổi trọng số  $\Delta W$  thu được từ quá trình backpropagation [12]. Và ở bước tiếp theo  $W'$  điều chỉnh với một  $\Delta W$  khác.

Quá trình forward sau mỗi step như sau:

$$\text{Iteration 0: } y = Wx$$

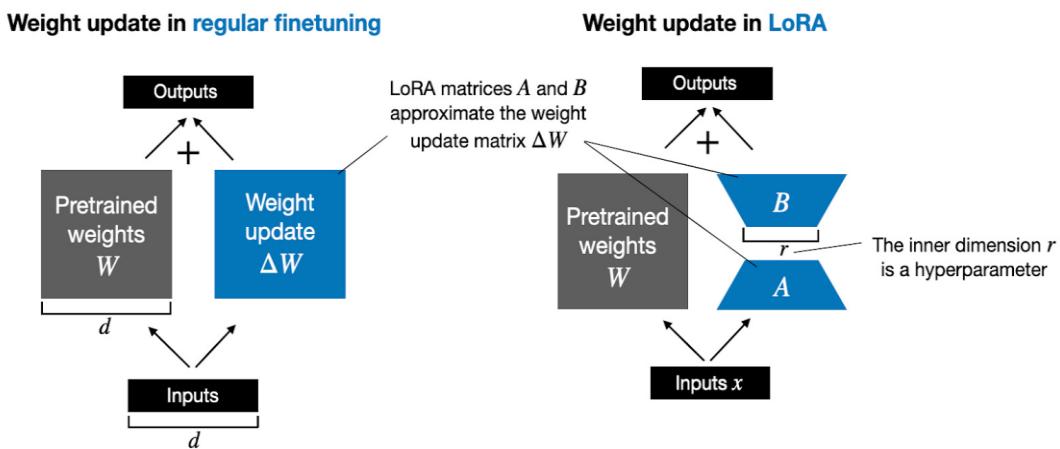
$$\text{Iteration 1: } y = W'x = (W + \Delta W)x$$

$$\text{Iteration 2: } y = y = W''x = (W' + \Delta W')x = (W + (\Delta W + \Delta W'))x$$

Xem xét quy trình cập nhật trong fine-tuning từ một góc độ khác (như thể hiện trong Hình 2.18). Tại mỗi bước, thay vì cập nhật trọng số  $W$  chúng ta sẽ cập nhật  $\Delta W$  (như đã biểu diễn ở step 2 phía trên). Lúc này, các trọng số đã được huấn luyện trước

$W$  sẽ được giữ nguyên (freeze), chúng ta chỉ cần xác định toàn bộ  $\Delta W$  là để có thể tính toán trọng số của mô hình sau khi fine-tune bằng cách cộng  $W$  với  $\Delta W$ .

Mục tiêu của LoRA là tìm ra cách biểu diễn ma trận  $\Delta W$  thành một dạng biểu diễn nhẹ hơn. Năm 2020, có một bài báo nói rằng những mô hình ngôn ngữ pre-trained có intrinsic dimension (hay intrinsic rank) cực kì thấp, tức là, mô hình này có thể được biểu diễn sử dụng số chiều ít hơn rất nhiều so với số chiều gốc của mô hình, mà vẫn giữ được hiệu suất khi đem đi fine-tune.



Hình 2.19. Quá trình cập nhật trọng số Full Fine Tuning và Lora fine tuning[12]

Tận dụng ý tưởng này, nhóm tác giả của LoRA cũng cho rằng,  $\Delta W$  có thể được biểu diễn bằng một số chiều ít hơn nhiều so với kích thước gốc của  $\Delta W$ . LoRA là kỹ thuật quyết định sử dụng phân rã ma trận (Matrix decomposition) để biểu diễn ma trận  $\Delta W$  thông qua tích của các ma trận con, giúp giảm độ phức tạp tính toán so với việc xử lý trên ma trận gốc. Có nhiều phương pháp phân rã ma trận khác nhau (như LU decomposition, Singular Value Decomposition, ...), và phương pháp lora đã được chọn cho mục đích này.

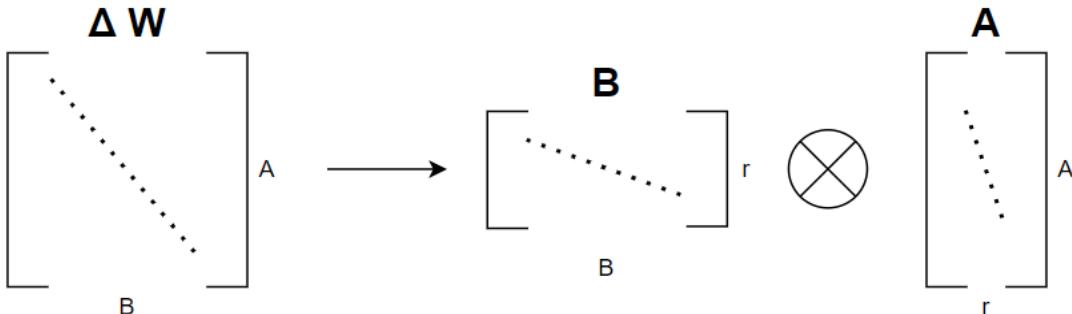
LoRA phân tách ma trận  $\Delta W$  thành 2 ma trận con  $A$  và  $B$  với số rank thấp hơn rất nhiều ma trận ban đầu. Cụ thể,  $\Delta W = BA$  với  $\Delta W \in R^{d \times k}, B \in R^{d \times r}, A \in R^{r \times k}$  và số rank  $r \ll \min(d, k)$ . Lúc này, output của layer đó sẽ trở thành:

$$y = W_0x + \Delta Wx = W_0x + BAx$$

Ma trận  $A$  được khởi tạo bằng phương pháp khởi tạo Gaussian ngẫu nhiên, trong khi ma trận  $B$  được khởi tạo với tất cả các phần tử bằng 0. Do đó, giá trị của  $\Delta W = BA$

có giá trị 0 khi bắt đầu fine-tuning. Và quá trình fine-tuning sẽ tối ưu để tìm ra ma trận  $A$  và  $B$ .

Chi tiết được mô tả như Hình 2.19 bên dưới:



**Hình 2.20. Mô tả rõ hơn về việc phân tách ma trận[13]**

Nhìn vào Hình 2.19, số phần tử mà ma trận  $\Delta W$  ban đầu có là  $A \times B$  còn số phần tử mà sau khi phân rã ma trận  $\Delta W$  thành 2 ma trận  $A$  và  $B$  với kích thước lần lượt là  $A \times r$  và  $r \times B$  [13]. Và khi ta tính tích 2 ma trận này sẽ tạo ra kết quả gần giống với ma trận ban đầu. Thông thường lora chúng ta sẽ cấu hình ở thành phần self attention bởi vì hầu hết tất cả bộ trọng số của mô hình điều nằm ở thành phần này.

Ví dụ chọn  $A = B = 100$ ,  $r = 4$ . Số phần tử của ma trận  $\Delta W$  trước phân rã là:  $100 \times 100 = 10000$  số phần tử sau phân rã là  $100 \times 4 + 100 \times 4 = 800$  ít hơn tới 12.5 lần.

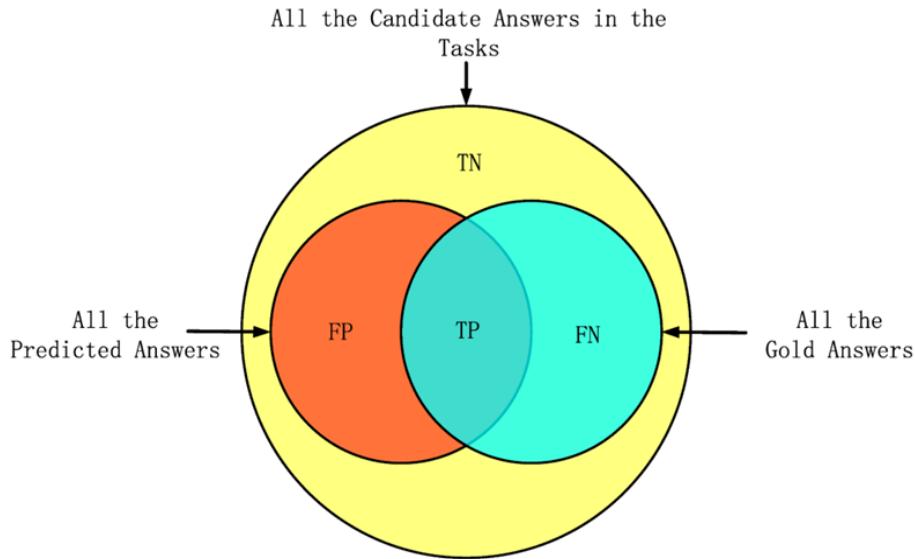
## 2.6. CÁC PHƯƠNG PHÁP ĐÁNH GIÁ MÔ HÌNH VÀ ĐÁNH GIÁ TÀI LIỆU

### 2.6.1. Độ đo F1-score

Độ đo F1-Score là một chỉ số tổng hợp giữa Precision và Recall, được sử dụng để đánh giá sự chồng chéo giữa câu trả lời dự đoán và câu trả lời đúng [15]. Theo phương pháp đánh giá trong bộ SQuAD, câu trả lời dự đoán và câu trả lời đúng được coi là túi token (bag of tokens), bỏ qua các dấu câu và các từ như "để", "đã", "mà" là các từ dừng. Mục tiêu của F1-Score là đo lường sự cân bằng giữa độ chính xác và độ bao phủ trong kết quả dự đoán.

F1-Score được tính thông qua hai độ đo phụ: Precision và Recall, chi tiết các độ đo này được trình bày như sau:

Precision (độ chính xác): đo lường phần trăm các từ trùng lặp giữa câu trả lời đúng và câu trả lời dự đoán. Để tính được Precision ở cấp độ từ, ta sử dụng các khái niệm:



**Hình 2.21. Biểu diễn tỉ lệ trùng lặp của ba thông số[15]**

- True Positive (TP): Số lượng từ giống nhau giữa câu trả lời dự đoán và câu trả lời đúng.
- False Positive (FP): Số lượng từ xuất hiện trong câu trả lời dự đoán nhưng không có trong câu trả lời đúng.
- False Negative (FN): Số lượng từ xuất hiện trong câu trả lời đúng nhưng không có trong câu trả lời dự đoán.

Công thức tính Precision như sau:

$$Precision = \frac{TP}{TP + FP}$$

Recall (độ bao phủ) đo lường tỷ lệ phần trăm từ trong câu trả lời đúng đã được mô hình dự đoán chính xác. Công thức tính Recall như sau:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score là một độ đo tổng hợp được tính từ Precision và Recall, phản ánh sự cân bằng giữa độ chính xác và độ bao phủ của mô hình. Công thức tính F1-Score như sau:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1-Score cho phép đánh giá mô hình một cách tổng thể, đặc biệt hữu ích trong các trường hợp dữ liệu mảnh cân bằng (ví dụ, câu trả lời đúng ít hơn câu trả lời sai).

### 2.6.2. Độ đo Exact Match (EM)

Nếu câu trả lời đúng cho câu hỏi là một câu hoặc một cụm từ, có thể một số từ trong câu trả lời do hệ thống tạo ra là câu trả lời đúng và các từ khác không phải là câu trả lời đúng. Trong trường hợp này, Exact Match (EM) đại diện cho tỷ lệ phần trăm các câu hỏi mà câu trả lời do hệ thống tạo ra hoàn toàn khớp với câu trả lời đúng, có nghĩa là mọi từ trong câu trả lời đều giống nhau hoàn toàn, không có sự sai lệch về từ ngữ.

Exact Match (EM) thường được viết tắt là EM và là một độ đo quan trọng trong các bài toán Question Answering (QA), giúp xác định độ chính xác của câu trả lời do hệ thống cung cấp [15].

Ví dụ: Giả sử một hệ thống QA có chứa N câu hỏi, mỗi câu hỏi có một câu trả lời đúng, câu trả lời có thể là một từ, một cụm từ, hoặc một câu. Nếu mô hình trả lời đúng cho M câu hỏi, thì tỷ lệ Exact Match (EM) có thể được tính như sau:

$$EM = \frac{M}{N} \times 100\%$$

Trong đó:

- M: Số câu hỏi mà câu trả lời do hệ thống hoàn toàn trùng khớp với câu trả lời đúng.
- N: Tổng số câu hỏi.

### 2.6.3. Độ đo ROUGE [18]

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) là một tập hợp các số liệu được dùng để đánh giá mô hình khởi tạo ngôn ngữ (tóm tắt văn bản hoặc dịch văn bản). ROUGE được dựa trên việc đo lường những điểm giống nhau giữa văn bản được tạo ra bởi model và văn bản được tạo ra từ con người.

a) ROUGE-N

ROUGE-N đo lường sự trùng lặp giữa n-grams trong văn bản cần đánh giá và văn bản được tham chiếu để đánh giá. Trong đó n-grams là một tập hợp n từ, số, ký hiệu hoặc dấu câu liên tiếp trong tài liệu văn bản.

ROUGE-N có công thức tính như sau:

$$ROUGE - N = \frac{\sum_{r_i \in \text{reference}} \sum_{n\text{-gram} \in r_i} \text{Count}(n\text{-gram}, \text{candidate})}{\sum_{r_i \in \text{reference}} \text{numNgram}(r_i)}$$

Trong đó:

- $r_i$  là một câu trong đoạn văn bản.
- $\text{Count}(n\text{-gram}, \text{candidate})$  là số lần nhân tố n-gram xuất hiện trong văn bản tóm tắt của máy.
- $\text{numNgram}(r_i)$  là số lần nhân tố n-gram xuất hiện trong câu  $r_i$ .

### b) ROUGE-L

ROUGE-L đo lường độ trùng lặp dựa trên độ dài chuỗi chung dài nhất của văn bản cần được đánh giá và văn bản được tham chiếu.

Để có thể tính được ROUGE-L, ta cần tính recall và precision trước:

$$\text{Recall} = \frac{\text{LCS}(\text{candidate}, \text{reference})}{\text{numOfWords}(\text{reference})}$$

$$\text{Precision} = \frac{\text{LCS}(\text{candidate}, \text{reference})}{\text{numOfWords}(\text{candidate})}$$

Trong đó:

- LCS là độ dài của chuỗi con chung dài nhất giữa văn bản cần đánh giá và văn bản được tham chiếu.
- $\text{numOfWords}(\text{reference})$  là số lượng từ trong văn bản được tham chiếu.
- $\text{numOfWords}(\text{candidate})$  là số lượng từ trong văn bản cần được đánh giá.

Công thức tính của ROUGE-L như sau:

$$ROUGE - L = \frac{(1 + \beta^2) \times \text{Recall} \times \text{Precision}}{\text{Recall} + \beta^2 \times \text{Precision}}$$

với  $\beta = \frac{\text{Precision}}{\text{Recall}}$

### c) ROUGE-W

ROUGE-W đo lường độ tương tự của văn bản cần đánh giá và văn bản được tham chiếu thông qua chuỗi con chung dài nhất liên tiếp.

Giả sử ta có ba chuỗi sau:

S1: “A B C D E F G”

S2: “A B C D H I K”

S3: “A T B V C O D”

Trong đó S1 là chuỗi được tham chiếu, S2 và S3 là hai chuỗi cần được đánh giá. Các chữ cái “A”, “B,”, “C”,... được xem là một từ. Có thể thấy rằng, S2 và S3 có chung điểm ROUGE-L. Nhưng trong trường hợp này S2 lại là một sự lựa chọn tốt hơn nếu so với S3. Đó là lý do phương pháp đánh giá ROUGE-W ra đời, đối với ROUGE-W, điểm của chuỗi S2 sẽ cao hơn so với S3 vì chuỗi S2 có các từ trùng nhau liên tiếp còn S3 bị đứt đoạn nếu so với S1.

### d) ROUGE-S

ROUGE-S dựa trên thống kê sự xuất hiện của skip-bigram, trong đó bigram là n-grams với n bằng 2. Nó đo lường độ trùng lặp của các từ không liên tiếp trong văn bản cần được đánh giá, được phân tách bởi một hoặc nhiều từ, so với văn bản được tham chiếu.

Giả sử ta có hai chuỗi tóm tắt: S1: “The cat is on the mat.” và S2: “The gray cat and the dog.”

Trong đó S1 là chuỗi cần đánh giá và S2 là chuỗi được tham chiếu. Lúc này nếu chúng ta xem bigram là “the cat”, thì đối với ROUGE-2 nó sẽ tìm kiếm chính xác “the cat” trong S2. Tuy nhiên đối với ROUGE-S thì “the gray cat” vẫn sẽ được chấp nhận là giống với “the cat”.

### e) ROUGE-SU

ROUGE-SU là một bản mở rộng của ROUGE-S. Nó tương tự như ROUGE-S nhưng nó sẽ đếm thêm cả các unigram, trong đó unigram là n-grams với n bằng 1.

#### 2.6.4. Độ đo BM25

BM25 là một thuật toán tính điểm dựa trên tần suất xuất hiện của các từ trong tài liệu và truy vấn [16]. Được sử dụng phổ biến trong các hệ thống tìm kiếm thông

tin, BM25 đánh giá mức độ liên quan giữa một tài liệu và một truy vấn dựa trên sự xuất hiện của các từ trong tài liệu, có điều chỉnh trọng số theo độ dài của tài liệu và mức độ quan trọng của từ (through qua IDF - Inverse Document Frequency).

BM25 không đơn thuần tính tần suất mà còn xử lý tốt các vấn đề như:

- Từ khóa xuất hiện nhiều lần trong một tài liệu (giảm thiểu ảnh hưởng của việc lặp từ quá nhiều).
- Độ dài tài liệu ảnh hưởng đến điểm số (dài quá hoặc ngắn quá).

Công thức:

$$BM25(q, d) = \sum_{t \in q} IDF(t) \cdot \frac{f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{\text{len}(d)}{\text{avgdl}}\right)}$$

Trong đó:

- $q$ : truy vấn.
- $d$ : tài liệu.
- $t$ : một từ trong truy vấn  $q$ .
- $f(t, d)$ : số lần từ  $t$  xuất hiện trong tài liệu  $d$ .
- $\text{len}(d)$ : độ dài của tài liệu  $d$ .
- $\text{avgdl}$ : độ dài trung bình của các tài liệu.
- $k_1$ : tham số điều chỉnh, thường nằm trong khoảng [1.2, 2].
- $b$ : tham số điều chỉnh độ nhạy với độ dài tài liệu, thường là 0.75.
- $IDF(t)$ : độ quan trọng của từ  $t$ , được tính như sau:

$$IDF(t) = \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

Trong đó:

- $N$ : tổng số tài liệu trong tập dữ liệu.
- $n_t$ : số tài liệu chứa từ  $t$ .

### 2.6.5. Độ đo Cosine Similarity

Cosine Similarity đo lường độ tương đồng giữa hai vector dựa trên góc giữa chúng trong không gian vector. Giá trị cosine similarity dao động từ -1 đến 1, trong đó:

- 1: Hai vector trùng khớp (tương tự hoàn toàn).

- 0: Hai vector vuông góc (không liên quan).

- 1: Hai vector đối ngược (cực kỳ khác biệt).

Cosine Similarity được ứng dụng rộng rãi trong tìm kiếm văn bản, phân cụm dữ liệu, và xử lý ngôn ngữ tự nhiên (NLP), nơi mà các văn bản được biểu diễn dưới dạng vector (như TF-IDF hoặc word embeddings).

Công thức:

$$\text{Cosine Similarity} = \text{Cos}(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Trong đó :

-  $\mathbf{A} \cdot \mathbf{B}$  : Là tích vô hướng của hai vectơ.

-  $\|\mathbf{A}\| \|\mathbf{B}\|$ : Là tích độ dài của 2 vector.

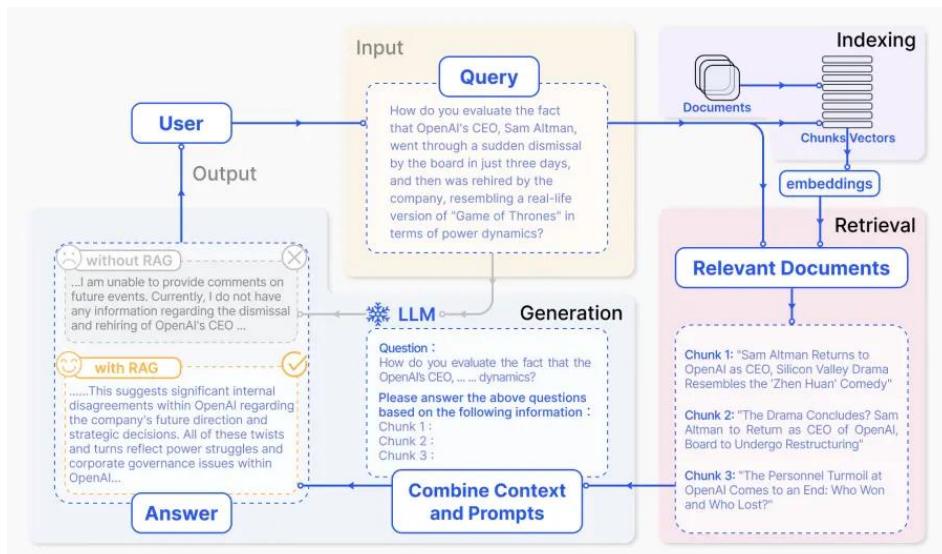
## CHƯƠNG 3: TỔNG QUAN VỀ RETRIEVAL-AUGMENTED GENERATION

### 3.1. KHÁI NIỆM VỀ RETRIEVAL - AUGMENTED GENERATION (RAG)

Các mô hình ngôn ngữ hiện nay có thể tinh chỉnh để thực hiện các nhiệm vụ như phân tích cảm xúc hoặc nhận diện, nhưng để xử lý các nhiệm vụ phức tạp đòi hỏi kiến thức sâu rộng, cần kết hợp với nguồn kiến thức bên ngoài. Phương pháp RAG (Retrieval Augmented Generation) cho phép kết hợp truy xuất thông tin và mô hình tạo sinh, giúp mô hình ngôn ngữ đáp ứng các nhiệm vụ cần kiến thức chuyên môn mà không cần huấn luyện lại. RAG tìm kiếm thông tin từ các nguồn, rồi đưa vào mô hình tạo sinh để tạo câu trả lời chính xác và cập nhật. Cách tiếp cận này tăng độ tin cậy và khả năng thích ứng của mô hình với các sự thay đổi thông tin, làm cho RAG trở thành phương pháp phổ biến trong việc hỗ trợ các mô hình mạnh như ChatGPT, Gemini, ... cung cấp thông tin thực tế hơn.

### 3.2. QUY TRÌNH HOẠT ĐỘNG CƠ BẢN

Hình dưới là giải thích đơn giản về quy trình hoạt động của RAG (Tạo sinh với Truy xuất) và từng bước trong đó:



Hình 3.1. Sơ đồ quy trình hoạt động cơ bản của RAG[2]

Bước 1: Input (Đầu vào) - Đây là câu hỏi hoặc yêu cầu mà người dùng gửi đến hệ thống. Nếu không dùng RAG, mô hình ngôn ngữ (LLM) sẽ trực tiếp tạo câu trả lời dựa trên những gì nó đã học được.

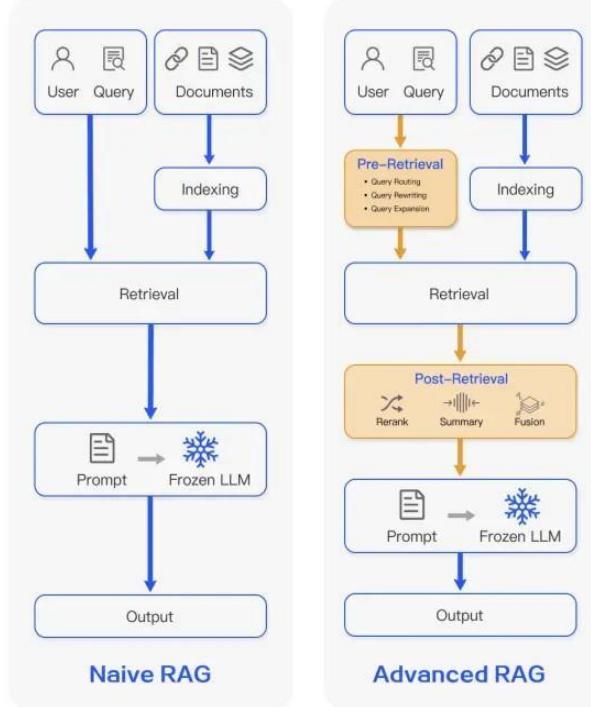
Bước 2: Indexing (Lập chỉ mục) - Hệ thống sẽ sắp xếp một bộ tài liệu liên quan. Các tài liệu này được cắt nhỏ thành từng đoạn nhỏ (chunks) rồi tạo các “embeddings” – dạng mã hóa để hệ thống hiểu nội dung. Sau đó, các đoạn mã hóa này được lưu trong một kho lưu trữ vector để dễ dàng truy xuất khi cần.

Bước 3: Retrieval (Truy xuất) - Khi có yêu cầu từ người dùng, hệ thống sẽ so sánh câu hỏi với các vectors đã lưu trước đó để tìm ra những tài liệu liên quan nhất.

Bước 4: Generation (Tạo câu trả lời) - Các tài liệu liên quan này sẽ được kết hợp với câu hỏi ban đầu để cung cấp thêm bối cảnh. Văn bản kết hợp này được đưa vào mô hình để tạo ra câu trả lời cuối cùng mà hệ thống sẽ gửi cho người dùng.

### 3.3. CÁC MÔ HÌNH RAG

Dưới đây là các mô hình của RAG qua từng giai đoạn phát triển và lý do vì sao chúng được cải tiến:

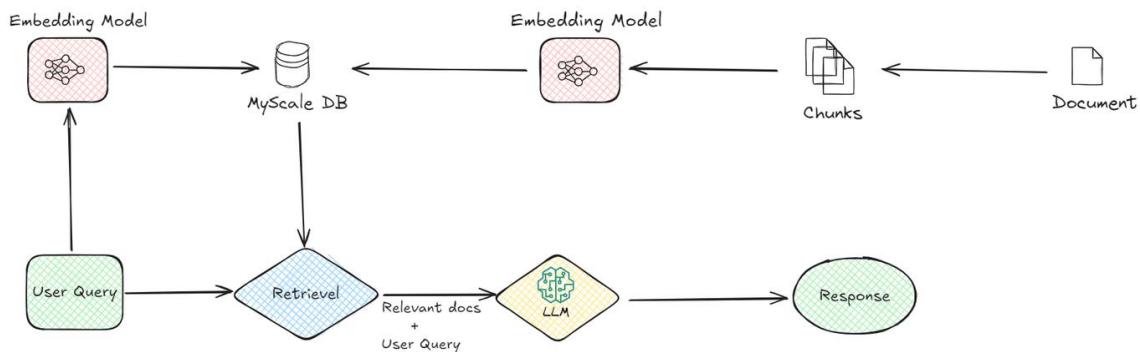


**Hình 3.2. Mô hình Naive RAG và mô hình Advanced RAG[2]**

**Naive RAG:** Đây là mô hình RAG ban đầu, trong đó quá trình truy xuất và tạo sinh diễn ra nhưng còn khá cơ bản. Naive RAG thường gặp giới hạn về hiệu suất, chi phí và độ chính xác. Hệ thống này không có nhiều tối ưu hóa nên thường cần nhiều tài nguyên hơn để đạt được kết quả.

Advanced RAG: Để khắc phục các hạn chế của Naive RAG, Advanced RAG đã được phát triển. Mô hình này đã tích hợp thêm các phương pháp cải tiến để tăng cường độ chính xác và hiệu quả, giảm thiểu chi phí mà vẫn duy trì chất lượng phản hồi. Advanced RAG có thể xử lý lượng dữ liệu lớn hơn, tối ưu hóa quá trình truy xuất và tạo sinh nhằm cải thiện khả năng trả lời chính xác hơn.

### 3.3.1. Naive RAG



Hình 3.3. Sơ đồ quy trình thực hiện của mô hình Naive RAG[3]

Bước 1: Chia nhỏ tài liệu (Document Chunking) - Quy trình bắt đầu bằng cách chia tài liệu lớn thành các đoạn nhỏ hơn. Điều này giúp hệ thống dễ dàng quản lý và xử lý thông tin.

Bước 2: Mô hình chuyển đổi ngữ nghĩa (Embedding Model) - Đây là thành phần quan trọng của RAG. Mô hình này chuyển đổi các đoạn tài liệu và câu hỏi của người dùng thành dạng số, gọi là embeddings. Khi người dùng đặt câu hỏi, mô hình sẽ chuyển đổi câu hỏi này thành các vector số, giúp hệ thống dễ dàng so sánh với các đoạn tài liệu đã được chuyển đổi.

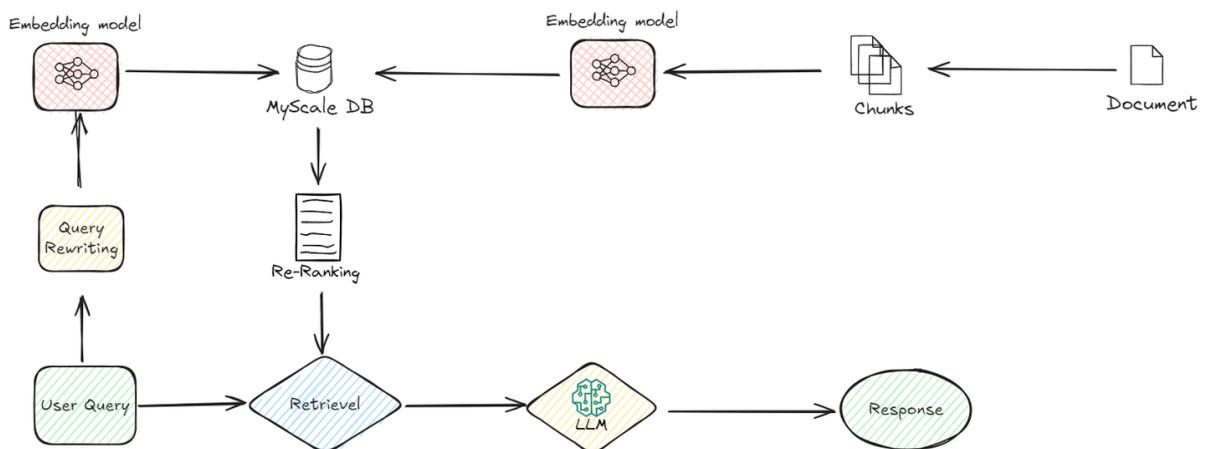
Bước 3: Cơ sở dữ liệu vector (Vector Database) - Sau khi các đoạn tài liệu được chuyển đổi thành embeddings, chúng được lưu trữ trong một cơ sở dữ liệu vector, ví dụ như Milvus, Pine Cone, Weaviate, ... . Cơ sở dữ liệu vector được thiết kế để lưu trữ và truy xuất embeddings một cách hiệu quả. Khi có câu hỏi từ người dùng, hệ thống sẽ dùng cơ sở dữ liệu vector này để tìm các đoạn tài liệu có nội dung tương tự nhất với câu hỏi.

Bước 4: Truy xuất (Retrieval) - Sau khi cơ sở dữ liệu xác định các đoạn tài liệu liên quan, chúng sẽ được truy xuất. Đây là bước quan trọng vì nó giúp thu hẹp thông tin xuống chỉ còn những dữ liệu phù hợp nhất, giúp tối ưu hóa độ chính xác của câu trả lời.

Bước 5: Mô hình ngôn ngữ lớn (LLM) - Sau khi nhận các đoạn tài liệu phù hợp, LLM sẽ tiếp nhận và xử lý, hiểu rõ thông tin từ các đoạn tài liệu đó và cả câu hỏi ban đầu để tạo ra câu trả lời phù hợp cho người dùng.

Bước 6: Tạo câu trả lời (Response Generation) - Cuối cùng, hệ thống sẽ tạo ra câu trả lời dựa trên thông tin đã được LLM xử lý và gửi câu trả lời này lại cho người dùng.

### 3.3.2. Advanced RAG



Hình 3.4. Sơ đồ quy trình thực hiện của mô hình Advanced RAG[3]

Ở Advanced RAG, hệ thống cải tiến thêm các chức năng ở phần trước và sau khi truy xuất.

Các Bước Cải Tiết Trước Khi Truy Xuất (Pre-Retrieval Optimizations): Trong Advanced RAG, quy trình truy xuất dữ liệu được tối ưu hóa ngay từ trước khi bắt đầu truy xuất. Cụ thể:

Bước 1: Viết lại câu hỏi (Query Rewriting) - Trước khi bắt đầu truy xuất, câu hỏi của người dùng được tinh chỉnh thông qua các kỹ thuật như viết lại câu hỏi, mở rộng và chuyển đổi câu hỏi, ... để làm cho câu hỏi rõ ràng hơn.

Bước 2: Sử dụng embeddings động (Dynamic Embeddings) - Trong Naive RAG, mô hình embeddings thường giống nhau cho mọi loại dữ liệu, nhưng Advanced

RAG điều chỉnh embeddings sao cho phù hợp với từng loại nhiệm vụ hoặc lĩnh vực, giúp hệ thống hiểu rõ hơn về ngữ cảnh của câu hỏi, từ đó cải thiện độ chính xác và hiệu quả của quá trình truy xuất.

Bước 3: Tìm kiếm kết hợp (Hybrid Search) - Advanced RAG kết hợp nhiều phương pháp tìm kiếm khác nhau để nâng cao hiệu suất truy xuất, bao gồm tìm kiếm từ khóa, tìm kiếm theo ngữ nghĩa và tìm kiếm bằng mạng nơ-ron.

Các Bước xử lý sau truy xuất (Post-Retrieval Processing): Sau khi truy xuất, Advanced RAG tiếp tục tinh chỉnh thông tin để đảm bảo chất lượng và độ chính xác của câu trả lời cuối cùng:

Bước 1: Xếp hạng lại các tài liệu liên quan (Re-ranking) - Tiến hành sắp xếp lại thông tin để ưu tiên các dữ liệu hữu ích nhất sau khi hệ thống lấy ra nhiều đoạn thông tin có thể liên quan. Quá trình re-ranking đánh giá và sắp xếp lại các tài liệu dựa trên các yếu tố như mức độ liên quan và tính phù hợp với ngữ cảnh.

Bước 2: Nén ngữ cảnh (Context Compression) - Là bước giúp loại bỏ các chi tiết thừa trong tài liệu trước khi chuyển đến mô hình ngôn ngữ (LLM). Điều này đảm bảo LLM chỉ nhận thông tin chính xác và hữu ích nhất để tạo ra câu trả lời tối ưu.

### 3.3.3. Ưu và nhược điểm của các mô hình RAG

#### 3.3.3.1. Naive RAG

**Ưu điểm:** Đơn giản, dễ triển khai, tiết kiệm tài nguyên và chi phí, phù hợp với yêu cầu ít phức tạp và độ chính xác trung bình.

**Nhược điểm:** Độ chính xác và hiệu suất hạn chế, khả năng tìm kiếm thông tin chính xác thấp, không hiệu quả với dữ liệu lớn hoặc yêu cầu phức tạp.

#### 3.3.3.2. Advanced RAG

**Ưu điểm:** Tối ưu hóa quy trình truy xuất và tạo sinh, giúp cải thiện độ chính xác và liên quan của câu trả lời nhờ các kỹ thuật như query-rewriting, re-ranking.

**Nhược điểm:** Phức tạp trong triển khai và duy trì, yêu cầu tài nguyên tính toán cao và tốn kém hơn.

## 3.4. VECTOR DATABASE

### 3.4.1. Khái niệm về Vector Database

Cơ sở dữ liệu vector là một loại cơ sở dữ liệu lưu trữ dữ liệu dưới dạng các đại diện toán học (vectors) trong không gian đa chiều, thay vì lưu trữ dữ liệu theo cách truyền thống (chẳng hạn như các bản ghi trong bảng với các giá trị cụ thể). Mỗi vector này biểu diễn các đặc tính của một đối tượng trong dữ liệu, giúp mô hình học máy có thể hiểu và xử lý dữ liệu một cách dễ dàng hơn.

### 3.4.2. Khái niệm về Vector và Embedding

#### 3.4.2.1. Vector

Một vector là một mảng các giá trị số thể hiện vị trí của một điểm trong không gian nhiều chiều. Mỗi giá trị trong vector đại diện cho tọa độ của điểm đó theo một chiều không gian cụ thể.

Ví dụ, một vector có thể là một danh sách các số như: {12, 13, 19, 8, 9}. Trong đó, mỗi số là một giá trị trong một chiều không gian xác định. Các vector thường được sử dụng trong toán học, khoa học máy tính và học máy để biểu diễn dữ liệu có nhiều thuộc tính.

#### 3.4.2.2. Embedding

Embeddings là các vector được sinh ra từ các mạng neural, giúp biểu diễn các đối tượng dữ liệu trong không gian đa chiều. Khi được huấn luyện đúng cách, mô hình học sâu có thể tự động tạo ra các embeddings, hỗ trợ các ứng dụng như tìm kiếm tương đồng, phân tích ngữ cảnh và AI tạo sinh. Các embeddings này đóng vai trò quan trọng trong việc tối ưu hóa quá trình xử lý dữ liệu, đặc biệt trong các bài toán về nhận diện đối tượng và phân tích dữ liệu lớn.

### 3.4.3. Ứng dụng của Vector Database

Vector database hỗ trợ các ứng dụng trong việc tìm kiếm tương đồng (similarity search) và tìm kiếm ngữ nghĩa (semantic search). Các vector gần nhau trong không gian vector có sự tương đồng và có thể liên quan đến nhau, giúp các ứng dụng kết nối thông tin phù hợp. Điều này không chỉ giúp người dùng tìm kiếm thông tin liên quan (ví dụ như tìm kiếm hình ảnh) mà còn giúp các ứng dụng đề xuất các sản phẩm tương tự, gợi ý bài hát, phim, chương trình truyền hình, hoặc các hình ảnh, video liên quan.

## 3.5. PROMPTING

### 3.5.1. Giới thiệu về Prompt

Một prompt là một đoạn văn bản hoặc tập hợp các chỉ dẫn cung cấp cho mô hình ngôn ngữ lớn (LLM) để giúp mô hình hiểu và tạo ra các điều ra như văn bản, hình ảnh, âm nhạc, và nhiều dạng khác. Việc sử dụng prompt hiệu quả giúp khai thác tối đa khả năng của mô hình ngôn ngữ lớn trong các ứng dụng trí tuệ nhân tạo. Quy trình hoạt động của prompt là như sau:

Bước 1: Input (Đầu vào) - là cách người dùng giao tiếp với LLM, giúp mô hình hiểu ta muốn nó làm gì.

Bước 2: Processing (Xử lý) - LLM phân tích prompt, dựa trên cơ sở dữ liệu không lò của LLM và khả năng hiểu các ngôn ngữ mà người dùng sử dụng.

Bước 3: Output (Đầu ra) - Dựa trên prompt, LLM tạo ra phản hồi, chẳng hạn như hoàn thành văn bản, dịch thuật, tóm tắt thông tin, ...

Về cơ bản, Prompt là cách “ra lệnh” cho mô hình LLM thực hiện nhiệm vụ. Ta cần đưa vào chỉ dẫn để LLM xử lý, và trả lại kết quả phù hợp ví dụ như viết, dịch, tóm tắt, hoặc sáng tạo nội dung.

### 3.5.2. Các cách tạo Prompt

#### 3.5.2.1. Prompt chỉ dẫn trực tiếp (Zero-shot Prompt)

Mục đích: Đưa ra yêu cầu hoặc nhiệm vụ mà không cung cấp bất kỳ ví dụ nào trước đó. LLM sẽ dựa vào kiến thức và khả năng của mình để hoàn thành nhiệm vụ.

Ví dụ:

"Hãy giải thích lý do tại sao trời mưa."

"Tạo một đoạn văn giới thiệu về một công ty khởi nghiệp công nghệ."

#### 3.5.2.2. Prompt đưa ra vài ví dụ (Few-shot Prompt)

Mục đích: Cung cấp một vài ví dụ về định dạng input - output để hướng dẫn LLM thực hiện nhiệm vụ thông qua các ví dụ mẫu.

Ví dụ:

Input: "Hôm nay là một ngày tuyệt vời! Tôi đã gặp lại bạn bè cũ và chúng tôi cùng nhau trò chuyện rất vui vẻ."

Output: Tích cực

Input: "Tôi cảm thấy rất buồn vì hôm nay công việc không như ý muốn và tôi đã thất bại trong cuộc họp quan trọng."

Output: Tiêu cực

Input: "Mẹ tôi đã nấu một bữa ăn thật ngon cho cả gia đình. Chúng tôi đã ăn tối và cười đùa thật vui vẻ."

Output: ?

Bạn hãy dựa trên các ví dụ trên để nhận dạng các ví dụ mới xem rơi vào lớp nào ‘tích cực hay tiêu cực’.

### 3.5.2.3. *Prompt đưa ra vai trò cụ thể (Role-based Prompt)*

Mục đích: Giao nhiệm vụ cho LLM với một vai trò cụ thể, giúp tạo ra phản hồi phù hợp với ngữ cảnh.

Ví dụ:

"Bạn là một giáo viên toán. Hãy giải thích cách giải phương trình bậc hai cho học sinh lớp 10."

"Bạn là một nhà báo. Viết một bài báo ngắn về tình hình kinh tế hiện nay."

## CHƯƠNG 4 : XÂY DỰNG MÔ HÌNH LLM VÀ TRUY XUẤT TĂNG CUỜNG (RAG) CHO BÀI TOÁN TRẢ LỜI CÂU HỎI

### 4.1. BÀI TOÁN TRÍCH XUẤT CÂU TRẢ LỜI TỪ ĐOẠN VĂN BẢN

#### 4.1.1. Giới thiệu về bài toán

##### 4.1.1.1. Sơ lược về bài toán trích xuất câu trả lời từ đoạn văn

Bài toán trích xuất câu trả lời từ đoạn văn bản (Machine Reading Comprehension - MRC) là một thách thức quan trọng trong xử lý ngôn ngữ tự nhiên (NLP). Mục tiêu là phát triển hệ thống có khả năng:

- + Hiểu ngữ cảnh văn bản: Phân tích và nắm bắt thông tin chính để trả lời chính xác.
- + Phân tích câu hỏi: Xác định loại thông tin cần trích xuất từ văn bản.
- + Trích xuất câu trả lời: Tìm và trích xuất câu trả lời chính xác từ đoạn văn, thường là cụm từ hoặc đoạn trích trực tiếp.

MRC hướng tới việc giúp máy tính đọc hiểu văn bản và hỗ trợ con người trong các tác vụ đòi hỏi phân tích ngữ nghĩa sâu.

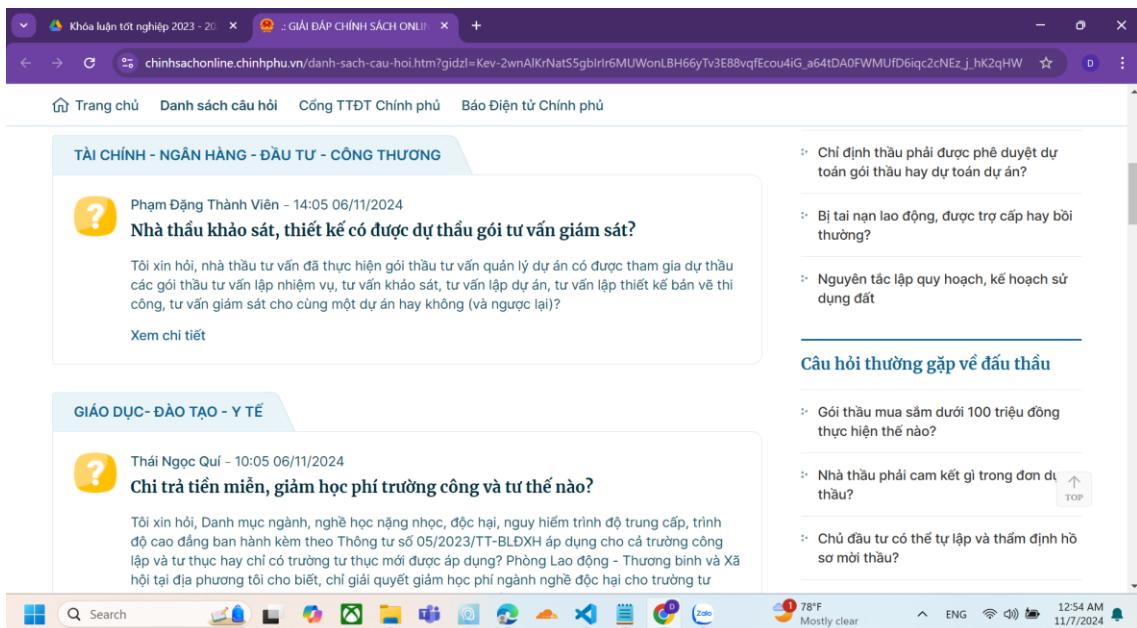
##### 4.1.1.2. Mục tiêu hướng tới

Mục tiêu của chúng tôi là xây dựng một hệ thống tối ưu nhất về mặt chi phí và hiệu suất cụ thể là sử dụng câu hỏi của người dùng và áp dụng kỹ thuật Retrieval Augmented Generation (RAG) để tìm ra các ngữ cảnh phù hợp nhất. Từ các ngữ cảnh đã được chọn lọc, mô hình sẽ tiến hành trích xuất câu trả lời chính xác và ngắn gọn, đáp ứng đầy đủ yêu cầu của câu hỏi. Đặc biệt, câu trả lời này sẽ được sử dụng làm đầu vào cho một mô hình tạo sinh tiếp theo, giúp tối ưu hóa lượng token khi sử dụng các API bên thứ ba.

#### 4.1.2. Xây dựng bộ dữ liệu cho bài toán Machine Reading Comprehension

##### 4.1.2.1. Quá trình thu thập dữ liệu

Để giải quyết bài toán MRC, chúng tôi sử dụng dữ liệu từ trang <https://chinh sach online.chinhphu.vn/>, nơi chuyên gia pháp lý tư vấn các tình huống pháp luật thực tế và trả lời các câu hỏi cụ thể. Đây là nguồn dữ liệu lý tưởng, vì nó cung cấp các câu hỏi rõ ràng, tình huống thực tế và câu trả lời kèm theo trích dẫn từ quy định pháp luật hiện hành.



**Hình 4.1. Giao diện trang web Giải đáp chính sách Online**

Quá trình thu thập dữ liệu được thực hiện bằng cách sử dụng thư viện BeautifulSoup trong Python, một công cụ hỗ trợ mạnh mẽ trong việc crawl dữ liệu từ các trang web. Các bước thực hiện bao gồm:

- Crawl nội dung từ trang web, lấy các thông tin chính như tên người hỏi, câu hỏi, tình huống pháp lý, thời gian hỏi và câu trả lời từ các chuyên gia.
- Tổ chức dữ liệu: Sử dụng thư viện Pandas, dữ liệu sau khi được thu thập sẽ được sắp xếp và chuyển đổi thành định dạng CSV, thuận tiện cho các bước xử lý và mô hình hóa tiếp theo.

Index	Name	Time	Question	Situation	Answer
0	Danh Ril	08:05 12/08/2024	Người dân tộc thiểu số xã nông thôn mới có đượ...	Tôi xin hỏi, một xã giai đoạn 2016-2020 là xã ...	Bộ Y tế trả lời vấn đề này như sau: Ngày 19/10/...
1	Ngô Đảm	09:05 02/08/2024	Có được truy lĩnh chế độ thai sản?	Tôi đóng BHXH từ tháng 6/2011 đến tháng 12/201...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
2	Nguyễn Thảo	07:05 02/08/2024	Khi nào cần xin thêm Giấy chứng nhận nghỉ việc...	Tôi xin hỏi, trên giấy ra viện có ghi chú ngườ...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
3	Trần Thị Hạnh	08:45 29/07/2024	Đóng BHXH tự nguyện bao lâu thì được lương hưu?	Tôi năm nay 53 tuổi, đóng BHXH tự nguyện được ...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
4	Nguyễn Hòa Điều	13:33 18/07/2024	Đóng BHXH bao lâu trước khi sinh thì được hưởng...	Tôi đóng BHXH từ tháng 1/2020 đến tháng 12/202...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...

**Hình 4.2. Nội dung Data Frame của dữ liệu sau khi thu thập**

File dữ liệu CSV cuối cùng sẽ bao gồm 30 460 mẫu và 5 thuộc tính chính, mỗi thuộc tính phản ánh một phần thông tin cần thiết cho mô hình MRC:

- Name: Tên người hỏi
- Time: Thời gian đặt câu hỏi
- Question: Câu hỏi của người dùng, đóng vai trò đầu vào chính cho mô hình.
- Situation: Tình huống thực tế mà người dùng mô tả, giúp mô hình có thêm bối cảnh để hiểu câu hỏi.
- Answer: Câu trả lời từ các chuyên gia pháp luật, thường bao gồm diễn giải chi tiết và các quy định pháp lý liên quan, là dữ liệu đầu vào để mô hình học cách tìm câu trả lời.

#### 4.1.2.2. Tổ chức dữ liệu

Sử dụng bộ dữ liệu đã thu thập từ trang Giải đáp chính sách Online làm bộ dữ liệu chính cho quá trình fine-tuning mô hình BERT. Đây là tập dữ liệu thực tế với các cặp câu hỏi và các đoạn văn bản liên quan, bao gồm cả tình huống và câu trả lời từ chuyên gia. Do đó, bộ dữ liệu này rất phù hợp để đánh giá hiệu quả của các mô hình Machine Reading Comprehension (MRC) trên tiếng Việt.

Bộ dữ liệu sau khi loại bỏ hai thuộc tính Name và Time. Bộ dữ liệu còn lại 30 460 mẫu với 3 thuộc tính chính giống hình bên dưới:

	<b>context</b>	<b>question</b>	<b>answer</b>
0	Tôi xin hỏi, một xã giai đoạn 2016-2020 là xã ...	Người dân tộc thiểu số xã nông thôn mới có được...	Bộ Y tế trả lời vấn đề này như sau: Ngày 19/10/...
1	Tôi đóng BHXH từ tháng 6/2011 đến tháng 12/201...	Có được truy lĩnh chế độ thai sản?	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
2	Tôi xin hỏi, trên giấy ra viện có ghi chú người...	Khi nào cần xin thêm Giấy chứng nhận nghỉ việc...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
3	Tôi năm nay 53 tuổi, đóng BHXH tự nguyện được ...	Đóng BHXH tự nguyện bao lâu thì được lương hưu?	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...
4	Tôi đóng BHXH từ tháng 1/2020 đến tháng 12/202...	Đóng BHXH bao lâu trước khi sinh thì được hưởng...	Bảo hiểm xã hội Việt Nam trả lời vấn đề này nh...

**Hình 4.3. Data Frame sau khi thực hiện tổ chức dữ liệu**

- Question: Câu hỏi của người dùng, đầu vào chính cho mô hình.
- Context: Đoạn văn bản liên quan, được tạo ra bằng cách gộp hai thuộc tính Situation và Answer để đảm bảo mô hình có ngữ cảnh đầy đủ.

- Answer: Câu trả lời chính xác được trích xuất từ Context.

Hình dưới đây minh họa một mẫu dữ liệu mẫu từ tập UTE-LAW: mỗi đoạn văn (passage hoặc context) đi kèm với các câu hỏi (question) và câu trả lời (answer) được trích dẫn từ đoạn văn.

**Context:** Điều 138. Thời gian làm việc và thời gian nghỉ ngơi.

1. Thời giờ làm việc bình thường không quá 08 giờ trong một ngày và không quá 48 giờ trong một tuần.

2. Người lao động được nghỉ ít nhất 24 giờ liên tục trong một tuần

**Question:** Thời giờ làm việc bình thường không quá bao nhiêu giờ trong một ngày?

**Answer:** Thời giờ làm việc bình thường không quá 08 giờ trong một ngày.

**Question:** Thời giờ làm việc bình thường không quá bao nhiêu giờ trong một tuần?

**Answer:** Thời giờ làm việc bình thường không quá 48 giờ trong một tuần.

#### 4.1.3. Quá trình tiền xử lý dữ liệu và huấn luyện mô hình

##### 4.1.3.1. Quá trình tiền xử lý dữ liệu cho việc fine-tuning

a) Kiểm tra các câu trả lời có nằm trong đoạn văn và phân chia bộ dữ liệu

Để sử dụng bộ dữ liệu này cho fine-tuning mô hình BERT trong bài toán MRC, chúng ta cần xử lý dữ liệu theo các bước sau:

- Tìm vị trí câu trả lời: Cần xác định vị trí bắt đầu của câu trả lời trong đoạn văn bản (Context). Nếu câu trả lời không có trong văn bản, trả về giá trị -1.

- Kiểm tra các mẫu: Những mẫu có câu trả lời không tồn tại trong văn bản (tức là có giá trị -1) sẽ bị loại bỏ khỏi quá trình huấn luyện và đánh giá mô hình.

Dưới đây là một ví dụ về một mẫu dữ liệu sau khi đã được xử lý:

{

'context': [

'Chương V của Bộ luật Lao động 2019 quy định về thời giờ làm việc, thời giờ nghỉ ngơi. Thời giờ làm việc bình thường không quá 8 giờ trong một ngày và không quá 48 giờ trong một tuần. Người lao động có thể làm thêm giờ khi được người sử dụng lao động yêu cầu và phải trả lương làm thêm giờ theo quy định. Tổng số giờ làm thêm không được vượt quá 200 giờ trong một năm, trường hợp đặc biệt không được vượt quá 300 giờ trong một năm. Người lao động được nghỉ ngơi giữa giờ làm việc

ít nhất 30 phút liên tục nếu làm việc 8 giờ liên tục trong ngày. Người lao động được nghỉ hằng tuần ít nhất 24 giờ liên tục. Người lao động được nghỉ lễ, Tết và nghỉ hằng năm có hưởng lương theo quy định của pháp luật. Thời gian làm việc ban đêm được tính từ 22 giờ đến 6 giờ sáng ngày hôm sau.'

],

'question

'Người lao động được nghỉ lễ, Tết và nghỉ hằng năm có hưởng lương theo quy định của pháp luật không?'

],

'answer

'answer\_start

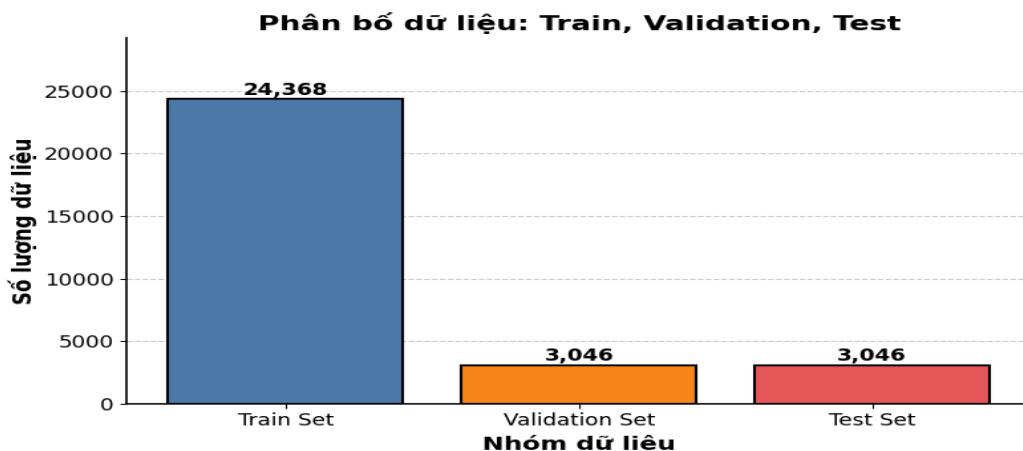
'text

}]

}

Bộ dữ liệu gồm 30.460 mẫu được chia ngẫu nhiên thành 3 tập: Train (80%), Validation (10%), và Test (10%) để đảm bảo tính đại diện và ngẫu nhiên.

- Tập Train: Dùng để huấn luyện mô hình, điều chỉnh trọng số và nhận diện mẫu trong dữ liệu. Tuy nhiên, nếu chỉ dựa vào nó, mô hình có thể bị overfitting.
- Tập Validation: Dùng để tinh chỉnh siêu tham số, đánh giá hiệu suất trong quá trình huấn luyện và giảm thiểu overfitting.
- Tập Test: Đánh giá khả năng tổng quát cuối cùng của mô hình trên dữ liệu mới, không tham gia vào quá trình huấn luyện.



**Hình 4.4. Biểu đồ Phân bố dữ liệu: Train, Validation, Test**

Tóm lại, sự phân chia này giúp mô hình học được từ một lượng dữ liệu lớn (tập Train), điều chỉnh các siêu tham số để tối ưu hóa hiệu suất (tập Validation), và cuối cùng kiểm tra khả năng dự đoán của mô hình trên dữ liệu chưa thấy (tập Test).

### b) Quá trình tạo nhãn dữ liệu và tokenizer

Để huấn luyện mô hình MRC, mỗi ví dụ cần có ba thành phần chính: đoạn văn chứa thông tin (context), câu hỏi về thông tin trong đoạn văn (question), và câu trả lời cụ thể kèm vị trí bắt đầu trong đoạn văn (answer). Để xử lý, các văn bản này được tokenizer biến đổi thành các token và sau đó mã hóa thành dạng số. Tokenizer không chỉ cắt nhỏ văn bản mà còn mã hóa đặc biệt để mô hình phân biệt giữa câu hỏi và ngữ cảnh chứa câu trả lời.

Ví dụ:

Context (Ngữ cảnh): “Trong Bộ luật Dân sự Việt Nam, quyền sở hữu tài sản là quyền của cá nhân, tổ chức đối với tài sản của mình. Người sở hữu có quyền sử dụng, chiếm hữu và định đoạt tài sản của mình theo quy định của pháp luật. Tuy nhiên, quyền sở hữu có thể bị hạn chế bởi các quy định của pháp luật như bảo vệ quyền lợi công cộng hoặc các quyền lợi của người khác.”

Question (Câu hỏi): “Quyền sở hữu tài sản bao gồm những quyền gì?”

Answer (Câu trả lời): “Quyền sử dụng, chiếm hữu và định đoạt tài sản”

Sau quá trình chuyển thành các token:

Context tokenized: ["Trong", "Bộ\_luật", "Dân\_sự", "Việt\_Nam", "quyền", "sở\_hữu", "tài\_sản", "là", "quyền", "của", "cá\_nhân", "tổ\_chức", "đối", "với", ...]

Question tokenized: ["Quyền", "sở\_hữu", "tài\_sản", "bao", "gồm", "những", "quyền", "gi", "?"]

Trong quá trình mã hóa context và question, các từ được chuyển thành số nguyên qua bộ từ điển của mô hình. Tham số max\_length giúp giới hạn độ dài chuỗi token, và khi đoạn ngữ cảnh quá dài, nó được chia thành các phần nhỏ. Để giữ lại thông tin quan trọng ở ranh giới các đoạn, tham số stride được sử dụng để tạo sự chồng

lắp giữa các phần mã hóa, đảm bảo thông tin không bị mất, đặc biệt khi câu trả lời nằm giữa hai đoạn.

Sau khi mã hóa context và question, bước tiếp theo là xác định vị trí của câu trả lời trong các chuỗi token. Mục tiêu ở đây là tìm ra các token tương ứng với vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn gốc, từ đó đặt các nhãn để mô hình biết phần nào là câu trả lời[14]. Để thực hiện điều này, tokenizer lưu lại thông tin về vị trí gốc của mỗi token trong đoạn văn bản ban đầu, nhờ đó có thể dễ dàng xác định các token trùng khớp với vị trí của câu trả lời.

Cụ thể:

- start\_position: xác định token đầu tiên trong chuỗi token mà câu trả lời bắt đầu.
- end\_position: xác định token cuối cùng trong chuỗi token mà câu trả lời kết thúc.

Quá trình này sẽ diễn ra qua các bước kiểm tra vị trí của từng token trong chuỗi mã hóa để xác định token nào nằm gần nhất với vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn gốc.

Trong một số trường hợp, câu trả lời nằm ngoài phạm vi của context đã được mã hóa (điều này xảy ra khi đoạn văn quá dài và tokenizer phải chia nhỏ nhiều lần)[14]. Để xử lý tình huống này, các ví dụ mà câu trả lời không xuất hiện đầy đủ trong đoạn mã hóa sẽ được gán nhãn mặc định là 0 hoặc bị loại bỏ để tránh gây nhiễu cho quá trình huấn luyện. Điều này đảm bảo rằng mô hình chỉ học từ các ví dụ có câu trả lời rõ ràng trong đoạn văn.

Việc chuẩn bị dữ liệu theo cách này giúp mô hình MRC học cách xác định câu trả lời chính xác trong văn bản, đồng thời xử lý hiệu quả các đoạn văn dài và phức tạp nhờ các thông số tokenizer như max\_length và stride.

Sau quá trình tokenizer và tạo nhãn cho bộ dữ liệu. Dữ liệu đầu vào đã được chuyển đổi thành các chỉ số số nguyên thông qua việc mã hóa từ vựng. Dữ liệu gồm năm thành phần chính: input\_ids, token\_type\_ids, và attention\_mask, cùng với các nhãn chỉ số vị trí của câu trả lời trong văn bản, được đánh dấu bằng start\_positions và end\_positions.

Trong đó

- input\_ids: Danh sách các chỉ số tương ứng với từng từ hoặc token trong văn bản, giúp mô hình xử lý văn bản dưới dạng số.
  - token\_type\_ids: Mảng chỉ ra phần nào của văn bản là câu hỏi (giá trị 0) và phần nào là đoạn văn (giá trị 1) để phân biệt giữa các phần văn bản.
  - attention\_mask: Mảng xác định các token nào cần được chú ý (giá trị 1) và token nào bị bỏ qua (giá trị 0), thường dùng để xử lý padding.
  - start\_positions và end\_positions: Chỉ vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn, là nhãn dùng trong bài toán trả lời câu hỏi.

#### 4.1.3.2. Quá trình huấn luyện mô hình BERT cho bài toán trả lời câu hỏi

Trong bài toán trả lời câu hỏi, mô hình BERT sử dụng hai input chính: câu hỏi và đoạn văn. Câu hỏi được chuyển thành embedding A, trong khi đoạn văn được chuyển thành embedding B. Mô hình học cách dự đoán vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn bằng cách sử dụng hai vecto ẩn: S cho vị trí bắt đầu và E cho vị trí kết thúc.

Mô hình được fine-tune thông qua việc sử dụng các vector bắt đầu (S) và kết thúc (E), trong đó S và E là các vectơ trong không gian ẩn của mô hình. Để tính toán xác suất của một từ tại vị trí thứ i (vị trí bắt đầu của câu trả lời), mô hình sử dụng công thức nhân điểm (Dot Product) giữa vector từ vựng của từ thứ i và vector S, sau đó áp dụng hàm Softmax trên tất cả các từ trong đoạn văn.

Công thức tính xác suất tại vị trí thứ  $i$  của nút bắt đầu:

$$P_{start}(i) = \text{Softmax}(S \cdot T_i) = \frac{e^{S \cdot T_i}}{\sum_{j=1}^n e^{S \cdot T_j}}$$

Trong đó:

- $S$  là vector bắt đầu (start vector).
- $T_i$  là vector của từ thứ  $i$  trong đoạn văn.
- $S \cdot T_i$  là phép nhân điểm (dot product\_ giữa  $S$  và  $T_i$  (vector của từ thứ  $i$ ).

Tương tự, một công thức được áp dụng cho vị trí kết thúc của câu trả lời:

$$P_{end}(i) = \text{Softmax}(E \cdot T_i) = \frac{e^{E \cdot T_i}}{\sum_{j=1}^n e^{E \cdot T_j}}$$

Trong đó:

- $E$  là vector kết thúc (end vector).
- $T_i$  là vector của từ thứ  $i$  trong đoạn văn.
- $E \cdot T_i$  là phép nhân điểm (dot product\_ giữa  $E$  và  $T_i$  (vector của từ thứ  $i$ ).

Điểm số của một ứng cử viên được trải dài từ vị trí  $i$  đến  $j$  được tính bằng tổng điểm số tại vị trí bắt đầu  $S_i$  và điểm số tại vị trí kết thúc  $E_j$ . Điều kiện là  $j \geq i$ , tức là vị trí kết thúc phải lớn hơn hoặc bằng vị trí bắt đầu. Mô hình sẽ chọn đoạn có điểm số cao nhất trong tất cả các đoạn  $i, j$  hợp lệ và mục tiêu của quá trình huấn luyện là tối đa hóa tổng Log-Likelihood của các dự đoán chính xác về vị trí bắt đầu và kết thúc của câu trả lời.

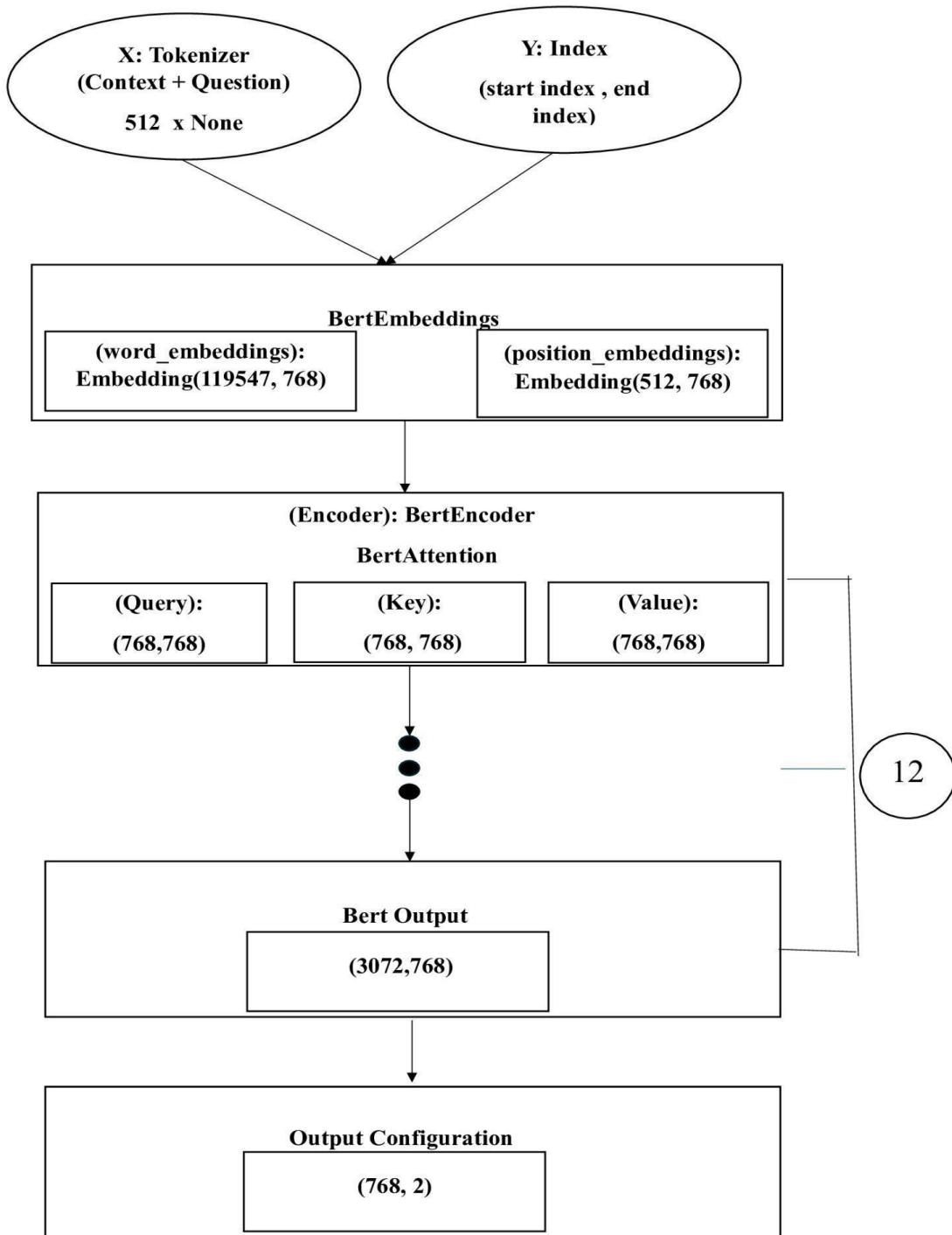
$$\text{Loss} = - \left( \sum_{i=1}^N y_{start}(i) \log(p_{start}(i)) + \sum_{j=1}^M y_{end}(j) \log(p_{end}(j)) \right)$$

Trong đó:

- $N$ : số lượng mẫu dữ liệu.
- $y_{start}(i)$ : là nhãn thực tế cho vị trí bắt đầu của câu trả lời.
- $y_{end}(j)$ : là nhãn thực tế cho vị trí kết thúc của câu trả lời.
- $p_{start}(i)$ : là xác suất mô hình dự đoán cho vị trí bắt đầu.
- $p_{end}(j)$ : là xác suất mô hình dự đoán cho vị trí kết thúc.

Trong nghiên cứu này triển khai hai phương pháp fine-tuning chính: Full-Fine-Tuning và LoRA Fine-Tuning, với mục tiêu phân tích hiệu quả của từng phương pháp và đưa ra kết luận rõ ràng dựa trên dữ liệu thực nghiệm.

### a) Quá trình Full Fine Tuning Bert



Hình 4.5. Cấu trúc của Bert trong quá trình full fine-tuning

Đối với kỹ thuật full-fine-tuning, đây là kỹ thuật fine-tuning truyền thống, trong đó tất cả các trọng số (weight) của mô hình BERT đều được cập nhật trong quá trình huấn luyện. Điều này có nghĩa là mọi lớp (layer) trong kiến trúc của BERT đều sẽ trải qua quá trình điều chỉnh để tối ưu hóa khả năng của mô hình đối với nhiệm vụ cụ thể, trong trường hợp này là dự đoán vị trí bắt đầu và kết thúc của câu trả lời trong đoạn văn bản.

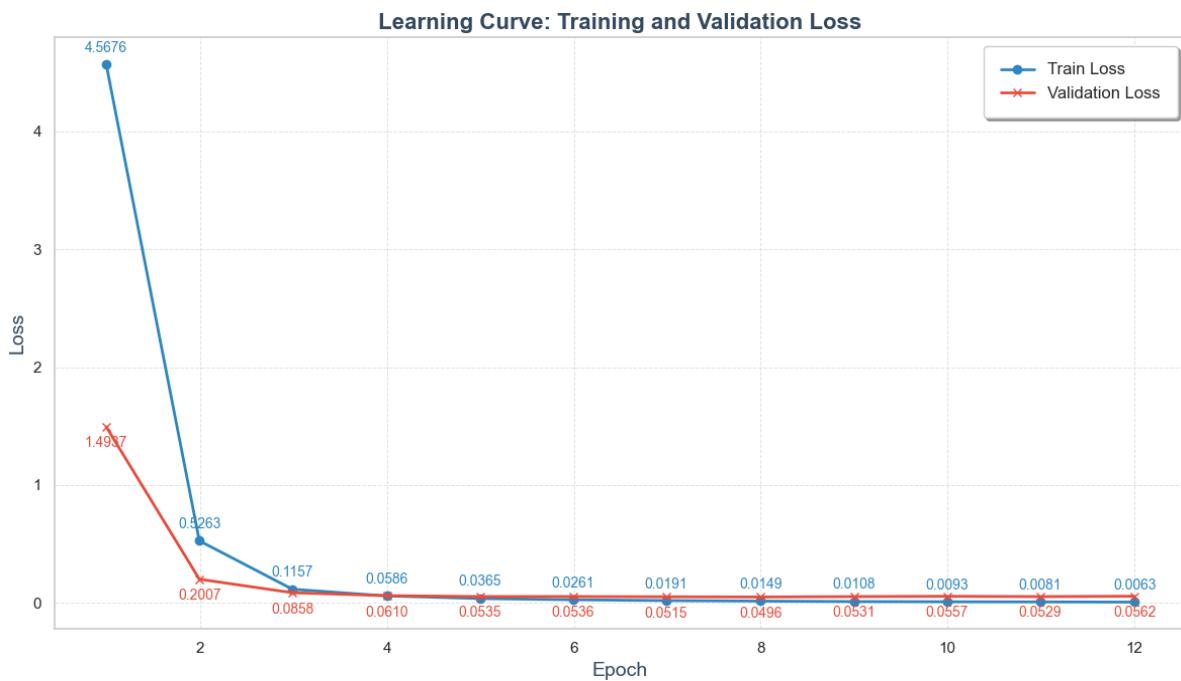
**Bảng 4.1. Bảng mô tả cấu hình siêu tham số cho quá trình huấn luyện Bert Full-fine-tuning**

Siêu tham số	Giá trị
Tốc độ học tập (learning_rate)	2e-5
Số mẫu trong mỗi batch huấn luyện (per_device_train_batch_size)	34
Số mẫu trong mỗi batch khi đánh giá (per_device_eval_batch_size)	34
Số bước huấn luyện để tích lũy gradient trước khi cập nhật trọng số (gradient_accumulation_steps)	12
Số lần toàn bộ tập dữ liệu được đưa qua mô hình (num_train_epochs)	15
Hệ số regularization để ngăn ngừa overfitting (weight_decay)	0.25
Giới hạn độ lớn gradient để tránh quá trình huấn luyện không ổn định (max_grad_norm)	0.6
Tỷ lệ bước huấn luyện để làm ám mô hình trước khi bắt đầu giảm tốc độ học (warmup_ratio)	0.2
Phương pháp điều chỉnh tốc độ học (lr_scheduler_type)	linear
Các nhãn được sử dụng trong nhiệm vụ huấn luyện (label_names)	['start_positions', 'end_positions']

**Bảng 4.2. Bảng kết quả thu được sau quá trình Bert Full-fine-tuning**

Thông số	Giá trị
----------	---------

Tổng số bước huấn luyện (global_step)	759/945
Mất mát huấn luyện (train/loss)	0.0063
Mất mát đánh giá (eval/loss)	0.0562
Tốc độ học tập (train/learning_rate)	4.920634920634921e-06
Độ lớn gradient (train/grad_norm)	42119.37109375
Số epoch (train/epoch)	12/15
Thời gian huấn luyện (train_runtime)	10 giờ 19 phút 17 giây

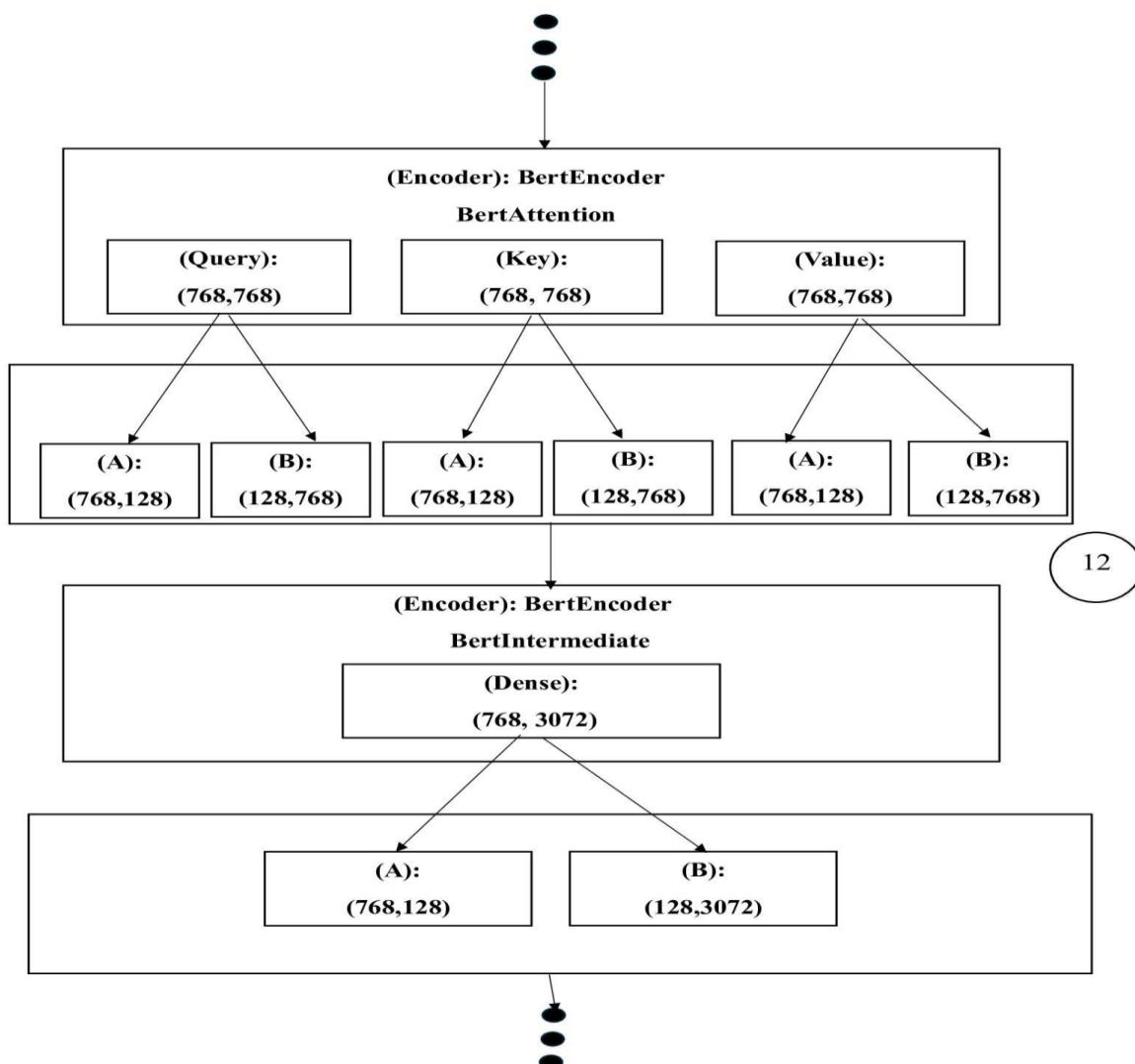


**Hình 4.6. Biểu đồ learning curve mô tả loss của quá trình full-tuning**

Quá trình fine-tuning mô hình MRC diễn ra trong 12 epoch và đạt được hiệu quả tối ưu hóa rõ rệt. Hàm loss giảm đáng kể trên cả tập huấn luyện và tập kiểm định, với loss trên tập kiểm định đạt 0.0562 và trên tập huấn luyện là 0.0063, cho thấy mô hình đạt được mức độ khai quát hóa tốt mà không bị overfitting. Biểu đồ learning curve cho thấy đường loss trên hai tập dữ liệu gần như song hành, chứng tỏ mô hình học tốt, ổn định và có thể áp dụng hiệu quả vào dữ liệu thực tế. Các tham số huấn luyện như learning rate, batch size, và warmup ratio đã được tối ưu hóa hợp lý, đảm bảo mô hình không bị overfitting và đạt được kết quả ổn định, tin cậy.

### b) Quá trình Lora Fine Tuning Bert

Kỹ thuật LoRA-fine-tuning là phương pháp tinh chỉnh nhẹ nhàng, chỉ cập nhật một phần nhỏ trọng số của mô hình, giúp tối ưu hóa huấn luyện nhanh và tiết kiệm tài nguyên. Trong LoRA, các trọng số ban đầu được giữ nguyên và chỉ tìm kiếm một tập hợp trọng số mới (delta W) để điều chỉnh, giúp mô hình thích nghi với nhiệm vụ mới mà không làm mất đi kiến thức đã học trước đó. Phương pháp này áp dụng vào thành phần multi-head attention và intermediate của BERT, giảm khói lượng tính toán và rủi ro overfitting khi huấn luyện với các nhiệm vụ khác nhau.



**Hình 4.7. Cấu trúc của Bert trong quá trình lora fine-tuning**

LoRA (Low-Rank Adaptation) áp dụng kỹ thuật phân rã ma trận để giảm thiểu độ phức tạp khi fine-tuning mô hình ngôn ngữ lớn (LLM). Thay vì cập nhật trực tiếp ma trận trọng số ban đầu, chẳng hạn như ma trận của lớp query có kích thước  $768 \times 768$ , LoRA chia ma trận này thành hai ma trận con: B ( $768 \times 128$ ) và A ( $128 \times 768$ ). Khi nhân hai ma trận này lại, ta thu được kết quả tương đương với ma trận trọng số gốc nhưng với chi phí tính toán thấp hơn, từ đó tiết kiệm đáng kể tài nguyên.

Kỹ thuật này không chỉ áp dụng cho lớp query mà còn được triển khai tương tự cho các thành phần khác như key, value trong Self-Attention và các thành phần trong Bert Intermediate Layer, như minh họa ở hình trên.

Một trong những điểm nổi bật của LoRA là nó giữ nguyên ma trận trọng số gốc, chỉ điều chỉnh thông qua delta W – phần trọng số được học thêm để thích nghi với dữ liệu mới. Nhờ vậy, mô hình tránh được hiện tượng "quên" kiến thức cũ (catastrophic forgetting) trong khi vẫn đảm bảo khả năng học hỏi và thích nghi với các nhiệm vụ mới.

Ta sẽ giữ nguyên toàn bộ cấu hình các siêu tham số chỉ cấu hình thêm lora vào quá trình fine-tuning

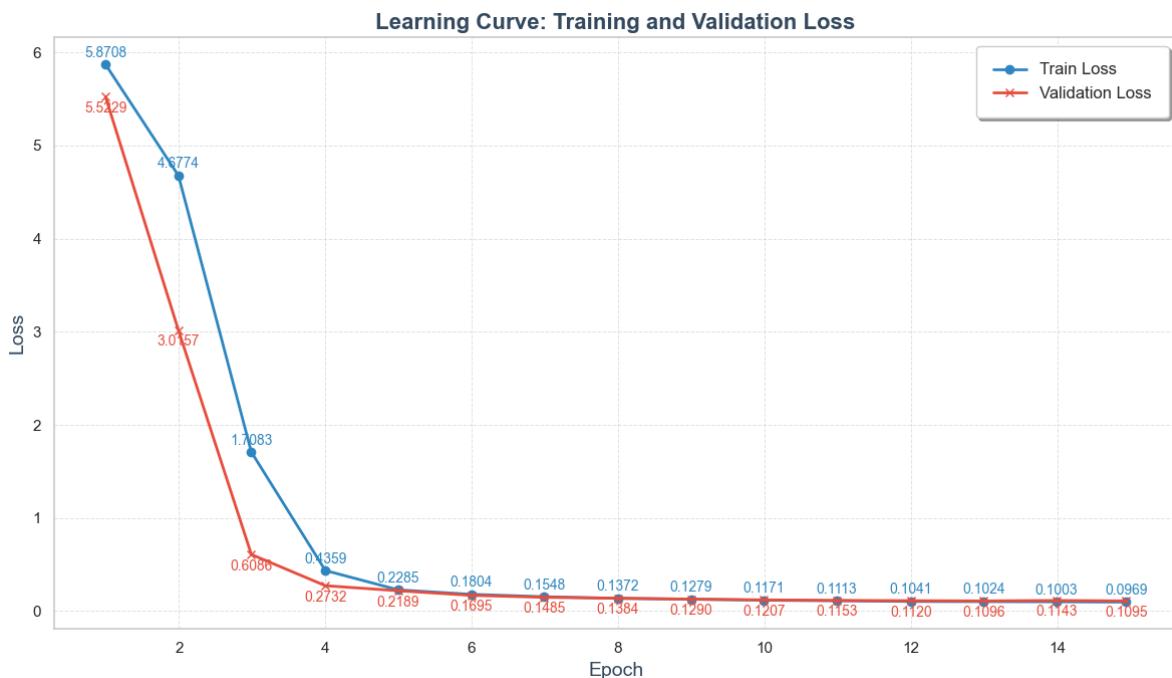
**Bảng 4.3. Bảng mô tả cấu hình siêu tham số cho quá trình huấn luyện Bert lora fine-tuning**

Siêu tham số	Giá trị
Điều chỉnh mức ảnh hưởng của LoRA (lora_alpha)	128
Giảm overfitting bằng dropout (lora_dropout)	0.1
Kích thước xấp xỉ low-rank (rank)	128
Chèn LoRA vào thành phần (target_modules)	["attention.self.query", "attention.self.key", "attention.self.value", "intermediate.dense"]

**Bảng 4.4. Bảng kết quả thu được sau quá trình Lora Full-fine-tuning**

Thông số	Giá trị
Tổng số bước huấn luyện (global_step)	945/945

Mất mát huấn luyện (train/loss)	0.0969
Mất mát đánh giá (eval/loss)	0.1094839796423912
Tốc độ học tập (train/learning_rate)	0
Độ lớn gradient (train/grad_norm)	35709.421875
Số epoch (train/epoch)	14.94/15
Thời gian huấn luyện (train_runtime)	11 giờ 32 phút 17 giây



**Hình 4.8. Biểu đồ learning curve mô tả loss của quá trình lora fine-tuning**

Quá trình fine-tuning bằng LoRA đã thành công trong việc điều chỉnh mô hình mà không làm mất kiến thức cũ, nhờ giữ nguyên trọng số gốc và thêm trọng số học mới. Huấn luyện đạt 945 bước, gần 15 epoch, với thời gian là 11 giờ 32 phút 17 giây, cho thấy quá trình huấn luyện đầy đủ và hợp lý. Mất mát huấn luyện (0.0969) và mất mát đánh giá (0.1095) đều thấp, chứng tỏ mô hình tổng quát hóa tốt mà không bị overfitting. Biểu đồ Learning Curve cho thấy sự ổn định và khả năng học tốt trên cả tập huấn luyện và kiểm định, cho thấy mô hình có thể áp dụng hiệu quả trên dữ liệu thực tế. LoRA giúp bảo toàn kiến thức gốc của mô hình và tăng cường khả năng học

đặc trưng mới, khăng định mô hình được fine-tuning hiệu quả, tổng quát hóa tốt và đạt được sự thành công trong tối ưu hóa.

**Bảng 4.5. Phần cứng cần thiết cho quá trình Bert Full-fine-tuning và Lora-fine-tuning**

Thông Số	Giá Trị
Số lượng CPU	2
Số lượng GPU	2 (Tesla T4)
Dung lượng đĩa (Total Disk)	8,656,922,775,552 bytes (~8.66 TB)
Dung lượng đĩa đã sử dụng	6,372,215,218,176 bytes (~6.37 TB)
Bộ nhớ RAM (Memory Total)	33,669,939,200 bytes (~33.67 GB)
Số lượng cores/1 GPU	2560

#### 4.1.4. Đánh giá mô hình cho bài toán trích xuất thông tin

##### 4.1.4.1. Phương pháp đánh giá

Trong bài toán Machine Reading Comprehension (MRC), mục tiêu chính là dự đoán vị trí bắt đầu và kết thúc của câu trả lời trong một đoạn văn bản dựa trên câu hỏi đầu vào. Để đánh giá độ chính xác của mô hình, các thước đo phổ biến như Exact Match (EM) và F1-score thường được sử dụng. Các thước đo này giúp xác định mức độ chính xác của câu trả lời mà mô hình dự đoán so với câu trả lời thực tế.

Quy trình đánh giá cho mô hình BERT với nhiệm vụ Question/ Answer bao gồm bốn bước chính:

1. Xác định các câu trả lời tiềm năng: Mô hình sẽ dựa vào các điểm số logits để xác định các vị trí có xác suất cao nhất cho vị trí bắt đầu và kết thúc của câu trả lời. Các vị trí này được cho là có khả năng chứa câu trả lời với độ chính xác cao nhất.
2. Lựa chọn câu trả lời tối ưu: Trong số các câu trả lời tiềm năng, mô hình sẽ chọn câu trả lời có tổng điểm số logit cao nhất (bao gồm cả vị trí bắt đầu và kết thúc) để làm dự đoán cuối cùng. Điều này đảm bảo rằng câu trả lời được chọn là dự đoán có khả năng chính xác nhất.

3. So sánh với câu trả lời thực tế: Câu trả lời dự đoán từ mô hình sẽ được so sánh với câu trả lời thực tế trong tập dữ liệu kiểm thử để tính toán điểm EM và F1-score. Mỗi câu trả lời được so sánh trực tiếp với mục tiêu để xác định điểm số EM và F1.
4. Tính tổng các điểm: Cuối cùng, điểm số EM và F1-score trên tất cả các mẫu trong tập kiểm thử được tính trung bình để đưa ra đánh giá tổng thể về hiệu suất của mô hình MRC. Điểm này phản ánh khả năng hoạt động của mô hình khi xử lý dữ liệu chưa được thấy trước.

#### *4.1.4.2. Kết quả thu được*

##### a) Đối với kỹ thuật Full-Fine-Tuning

Nhận dạng một ví dụ cơ bản với một đoạn văn và hai câu hỏi dưới đây:

- Đoạn văn

Điều 19. Người được cấp thẻ căn cước

1. Người được cấp thẻ căn cước là công dân Việt Nam.

2. Công dân Việt Nam từ đủ 14 tuổi trở lên phải thực hiện thủ tục cấp thẻ căn cước.

3. Công dân Việt Nam dưới 14 tuổi được cấp thẻ căn cước theo nhu cầu.

- Câu hỏi 1

Người bao nhiêu tuổi thì đăng ký căn cước công dân ?

- Câu trả lời cho câu hỏi 1

từ đủ 14 tuổi trở lên phải thực hiện thủ tục cấp thẻ căn cước.

- Câu hỏi 2:

Vậy người được cấp căn cước công là ai ?

- Câu trả lời cho câu hỏi 2:

Người được cấp thẻ căn cước là công dân Việt Nam.

Đối với kỹ thuật Full Fine Tuning ta thu được điểm F1 và Extract Match rất cao trên tập kiểm tra với kết quả như bảng dưới đây:

**Bảng 4.6. Kết quả Full Tuning của BERT**

F1 Score	Exact Match
89.03%	81.42%

Những chỉ số này chứng tỏ hiệu quả cao của mô hình trong việc trích xuất câu trả lời từ văn bản, đồng thời cho thấy khả năng áp dụng thành công vào các bài toán trích xuất thông tin thực tế.

Việc áp dụng kỹ thuật Full Fine-Tuning trên mô hình BERT đã cho thấy khả năng đạt được hiệu suất tốt trong các bài toán MRC. Các chỉ số F1-score và Exact Match cao cho thấy mô hình có khả năng hiểu và phân tích văn bản để trả lời câu hỏi một cách chính xác. Điều này mở ra triển vọng ứng dụng mô hình này trong các hệ thống xử lý ngôn ngữ tự nhiên phức tạp hơn, từ đó nâng cao chất lượng của các sản phẩm và dịch vụ AI.

b) Đối với kỹ thuật Lora Fine Tuning

Nhận dạng một ví dụ cơ bản với một đoạn văn và hai câu hỏi dưới đây:

- Đoạn văn

Điều 19. Người được cấp thẻ căn cước  
1. Người được cấp thẻ căn cước là công dân Việt Nam.  
2. Công dân Việt Nam từ đủ 14 tuổi trở lên phải thực hiện thủ tục cấp thẻ căn cước.  
3. Công dân Việt Nam dưới 14 tuổi được cấp thẻ căn cước theo nhu cầu.

- Câu hỏi 1

Người bao nhiêu tuổi thì đăng ký căn cước công dân ?

- Câu trả lời cho câu hỏi 1

từ đủ 14 tuổi trở lên phải thực hiện thủ tục cấp thẻ căn cước.

- Câu hỏi 2:

Vậy người được cấp căn cước công là ai ?

- Câu trả lời cho câu hỏi 2:

Người được cấp thẻ căn cước là công dân Việt Nam.

Đối với kỹ thuật Lora Fine Tuning ta thu được điểm F1 và Extract Match cũng khá ổn trên tập kiểm tra với kết quả như bảng dưới đây:

**Bảng 4.7. Kết quả Lora fine-tuning của BERT**

F1 Score	Exact Match
85.89 %	77.68%

c) So sánh giữa hai kỹ thuật full-fine-tuning và lora fine-tuning

Trong bài toán fine-tuning mô hình BERT để trích xuất câu trả lời từ đoạn văn, hai kỹ thuật phổ biến là Full Fine-Tuning và LoRA Fine-Tuning được so sánh dựa trên các tiêu chí sau: điểm F1 Score, Exact Match, tài nguyên GPU sử dụng, khả năng giữ lại kiến thức cũ, và hiệu suất đa tác vụ. Các kết quả được đánh giá theo ba mức độ: Cao, Trung Bình, và Thấp.

**Bảng 4.8. Tiêu chí so sánh giữa hai kỹ thuật**

Tiêu chí	Full Fine-Tuning	Lora-Fine-Tuning
Tài nguyên tính toán	Cao	Trung Bình
Hiệu suất cho 1 tác vụ	Cao	Trung Bình
Khả năng giữ lại kiến thức cũ	Thấp	Cao
Đa tác vụ	Thấp	Cao

Việc chọn giữa Full Fine-Tuning và LoRA Fine-Tuning phụ thuộc vào yêu cầu bài toán và hạ tầng tính toán. Full Fine-Tuning tối ưu hóa hiệu suất cao nhất cho nhiệm vụ cụ thể, đạt điểm F1 Score và Exact Match tốt hơn nhưng cần nhiều tài nguyên tính toán và dễ bị "quên" kiến thức đã học. LoRA Fine-Tuning chỉ cập nhật một phần nhỏ trọng số, giúp tiết kiệm tài nguyên và duy trì kiến thức đã học, phù hợp cho các bài toán đa tác vụ, nhưng hiệu suất trên từng nhiệm vụ thường thấp hơn. Nếu cần tối ưu hóa cho một nhiệm vụ duy nhất và có đủ tài nguyên, Full Fine-Tuning là tốt nhất. Ngược lại, nếu ưu tiên tiết kiệm tài nguyên và xử lý đa tác vụ, LoRA Fine-Tuning là lựa chọn phù hợp.

## 4.2. BÀI TOÁN TÓM TẮT VĂN BẢN

### 4.2.1. Giới thiệu về bài toán

Tóm tắt văn bản là việc tạo ra một bản tóm tắt ngắn bao gồm một số câu có thể nắm bắt được ý tưởng của toàn đoạn văn bản. Vượt qua được bài toán này có thể giúp ta hiểu hơn về ngôn ngữ tự nhiên. Không những thế, một văn bản tóm tắt được làm tốt có thể giúp ta nắm bắt được nội dung chính của văn bản chỉ trong thời gian ngắn.

### 4.2.2. Xây dựng bộ dữ liệu cho bài toán tóm tắt văn bản

#### 4.2.2.1. Quá trình thu thập dữ liệu

Nguồn dữ liệu cho bài toán tóm tắt văn được được lấy từ Cơ sở dữ liệu Quốc gia về văn bản pháp luật (vbpl.vn). Dữ liệu mang tính chất là văn bản pháp quy, được tổng hợp từ ba loại văn bản pháp quy phổ biến tại Việt Nam là luật, nghị định và thông tư. Tập dữ liệu gốc sẽ gồm 7 thuộc tính và 11936 bản ghi, trong đó các thuộc tính bao gồm:

Loại văn bản: Loại của văn bản pháp quy.

Số ký hiệu: id của văn bản.

Tiêu đề: Tên của văn bản.

Tình trạng: Tình trạng hiện tại của văn bản.

Ngày ban hành: Ngày văn bản được ban hành.

Ngày hiệu lực: Ngày văn bản bắt đầu có hiệu lực.

Nội dung: Nội dung của văn bản.

#### 4.2.2.2. Tổ chức dữ liệu

Sau khi đã có được tệp dữ liệu trên, chúng tôi tiến hành xử lý nhằm tạo ra một tệp dữ liệu mới. Trong quá trình xử lý này, các văn bản luật chỉnh sửa lại những văn bản luật cũ sẽ bị lược bỏ vì việc phân tách ra thành các điều luật một cách hoàn chỉnh sẽ gặp khó khăn. Tệp dữ liệu sau khi đã xử lý sẽ gồm 3 thuộc tính, thuộc tính đầu tiên sẽ là tiêu đề của văn bản luật, hai thuộc tính tiếp theo sẽ là điều luật tóm tắt và nội dung điều luật được lấy từ thuộc tính nội dung của tệp gốc. Thuộc tính nội dung sẽ được tách ra làm nhiều điều luật nhỏ vì trong một văn bản luật sẽ có nhiều điều luật nhỏ, sau đó tên của điều luật sẽ là nội dung tóm tắt của điều luật, còn phần còn lại sẽ

là nội dung điều luật. Tập dữ liệu mới khi này sẽ bao gồm 31659 bản ghi và có các thuộc tính sau:

Title: Tên của văn bản luật

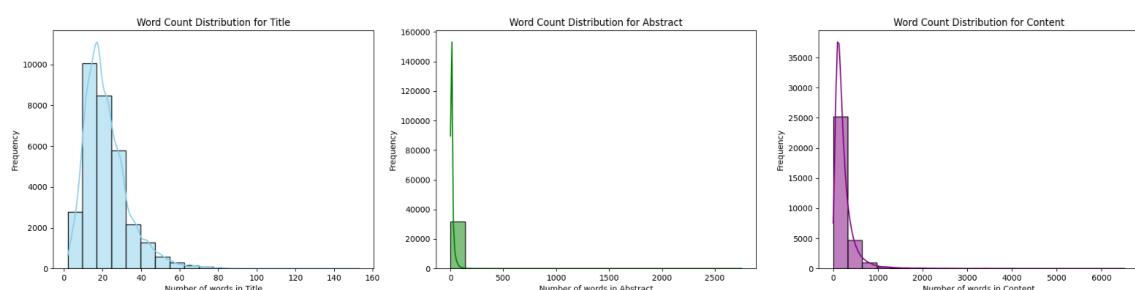
Abstract: Tên của điều luật

Content: Nội dung điều luật

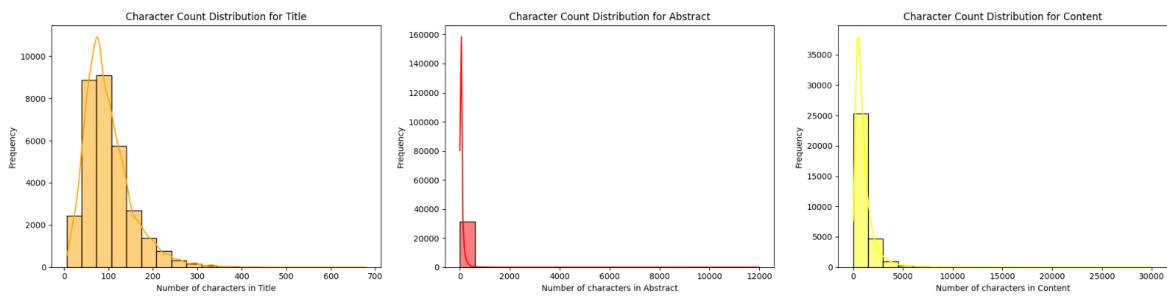
Tập dữ liệu sau đó cũng sẽ được xáo trộn ngẫu nhiên và chia ra làm 3 tập là tập train, tập test và tập validation với tỉ lệ là 70% cho tập train, 20% cho tập test và 10% cho tập validation.

**Bảng 4.9. Bảng mô tả độ dài của các câu trong tập dữ liệu luật**

	Word count			Character count		
	Title	Abstract	Content	Title	Abstract	Content
<b>count</b>	31659					
<b>mean</b>	21.924	15.561	250.389	98.855	72.225	1149.230
<b>std</b>	11.539	33.735	329.782	98.855	72.225	1149.231
<b>min</b>	2	0	0	6	0	1
<b>25%</b>	14	6	100	63	26	461
<b>50%</b>	20	10	165	87	45	759
<b>75%</b>	27	17	284	123	78	1304
<b>max</b>	153	2755	6512	679	11994	30077



**Hình 4.9. Biểu đồ phân bố độ dài câu theo 3 thuộc tính của tập dữ liệu luật pháp**



**Hình 4.10. Biểu đồ phân bố độ dài câu theo từ theo 3 thuộc tính của tệp dữ liệu luật pháp**

Thông qua biểu đồ trên ta có thể thấy được 75% độ dài của nội dung tiêu đề của điều luật rơi vào khoảng 78 ký tự, hay là 17 từ. Còn đối với phần phân nội dung của điều luật thì 75% độ dài lại rơi vào khoảng 1309 ký tự, cũng đồng thời là 285 từ.

#### 4.2.3. Quá trình tiền xử lý dữ liệu và huấn luyện mô hình

##### 4.2.3.1. Quá trình tiền xử lý dữ liệu cho việc fine-tuning

a) Xử lý dữ liệu, quá trình tạo nhãn dữ liệu phân chia bộ dữ liệu

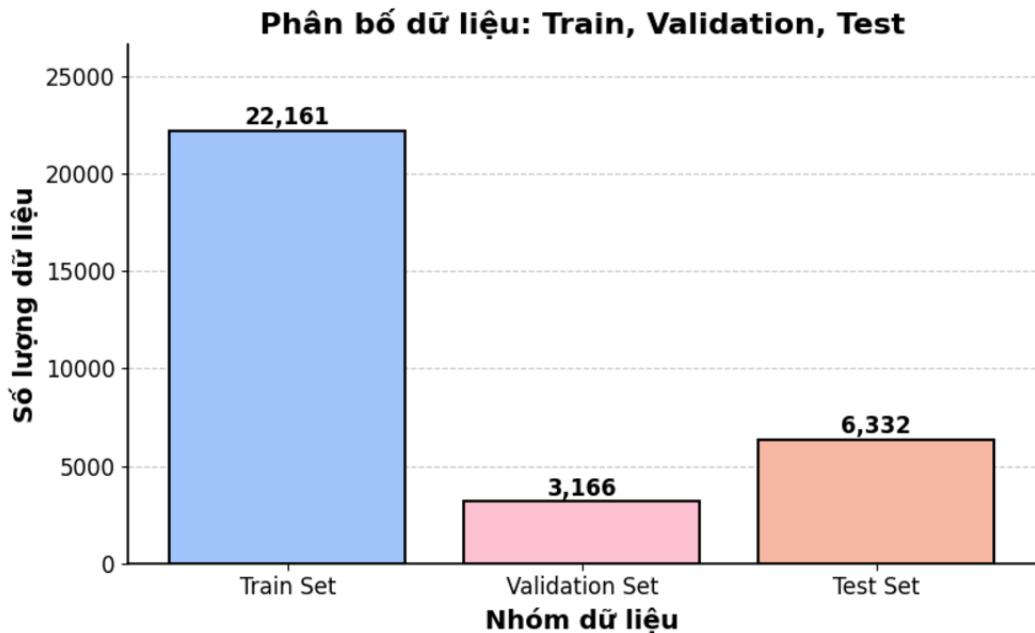
Do tập dữ liệu đầu vào là tập dữ liệu thô, chính vì vậy cần phải trải qua bước tiền xử lý dữ liệu (làm sạch dữ liệu) và tiến hành phân chia bộ dữ liệu để có thể đưa vào quá trình fine-tuning. Tiếp đó thực hiện quá trình tạo nhãn dữ liệu sau khi thực hiện xong tiền xử lý. Với cột “Content” được giữ nguyên và được trả về “input\_text”, cột “Title” và cột “Abstract” sẽ được sử dụng để tạo ra văn bản tóm tắt, chính vì vậy sau khi thực hiện tiền xử lý, hai cột này sẽ kết hợp lại và tạo nên “summary\_text”.

Dưới đây là một ví dụ về một mẫu dữ liệu sau khi đã được xử lý:

**input\_text:** ""Văn bản, phiếu yêu cầu cung cấp dữ liệu mà nội dung không rõ ràng, cụ thể; yêu cầu cung cấp dữ liệu thuộc phạm vi bí mật nhà nước không đúng quy định. Văn bản yêu cầu không có chữ ký của người có thẩm quyền và đóng dấu xác nhận đối với tổ chức; phiếu yêu cầu không có chữ ký, tên và địa chỉ cụ thể của cá nhân yêu cầu cung cấp dữ liệu. Mục đích sử dụng dữ liệu không phù hợp theo quy định của pháp luật. Không thực hiện nghĩa vụ tài chính theo quy định""

**summary\_text:** "Quy định về xây dựng, quản lý, khai thác hệ thống thông tin đất đai Những trường hợp không cung cấp dữ liệu"

Sau đó, tệp dữ liệu sẽ được xáo trộn ngẫu nhiên và chia ra làm 3 tệp là tệp train (tỉ lệ 70%), tệp test (tỉ lệ 20%) và tệp validation (tỉ lệ 10%).



**Hình 4.11. Biểu đồ Phân bố dữ liệu: Train, Validation, Test**

#### b) Quá trình tokenizer

Quá trình tokenization của T5 hoạt động gần như giống BERT. Ban đầu đoạn văn có giới hạn tối đa là 512 token, trường hợp nếu văn bản quá dài sẽ được cắt bớt. Nếu đoạn văn bản ngắn hơn 512 token, chúng sẽ được đếm thêm các token đặc biệt để đạt được độ dài tối đa. Trường hợp nếu đoạn văn bản quá dài sẽ cắt văn bản dài thành nhiều đoạn, stride xác định độ dài của đoạn chồng giữa các chunk (phản cắt) nhằm giữ lại thông tin trong các điểm nối giữa các đoạn.

#### 4.2.3.2. Quá trình huấn luyện mô hình T5 cho bài toán tóm tắt văn bản

Để thực hiện bài toán tóm tắt văn bản, mô hình T5 có với cấu trúc Encoder-Decoder. Với Encoder, sẽ nhận đầu vào là đoạn văn bản sau đó mã hóa thành các vector đặc trưng trong không gian ẩn. Lớp Encoder sẽ bao gồm Self-attention Layer (Cả self-attention và cross-attention) và Feed-forward layer (FF). Sau đó sẽ đến Decoder, sẽ nhận các mã hóa từ encoder và thực hiện dự đoán kết quả, đồng thời các lớp decoder có thêm cross-attention để kết hợp thông tin từ encoder.

Công thức tính Self-Attention (trong Encoder và Decoder) sử dụng dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

Trong đó:  $Q$  là Query matrix,  $K$  là Key matrix,  $V$  là Value matrix,  $d_k$  là Dimension của Key.

Cross-attention trong decoder cũng sử dụng công thức tương tự, nhưng với Query từ Decoder và Key, Value từ Encoder.

Sau mỗi lớp attention và feed-forward, Layer Normalization được áp dụng:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma + \epsilon}$$

Trong đó:

- $x$ : Đầu vào của lớp.
- $\mu$  và  $\sigma$ : Trung bình và độ lệch chuẩn của đầu vào.
- $\epsilon$ : Một hằng số nhỏ để tránh chia cho 0.

Sau mỗi lớp attention, dữ liệu sẽ đi qua một mạng nơ-ron feed-forward với ReLU activation:

$$FF(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

Trong đó:

- $W_1, W_2$ : Trọng số của mạng neural.
- $b_1, b_2$ : Bias.

Cuối cùng, đầu ra của Decoder sẽ được đưa qua một lớp Linear (fully connected layer) để tạo ra dự đoán:

$$P(x) = \text{Softmax}(W_{out} \cdot h + b_{out})$$

Trong đó:

- $h$ : là đầu ra từ decoder (embedding vector).
- $W_{out}$ : là trọng số của lớp Linear (thường có kích thước  $768 \times vocab_size$ )
- $b_{out}$ : Bias.

Khi mô hình huấn luyện, hàm loss được sử dụng là cross-entropy loss:

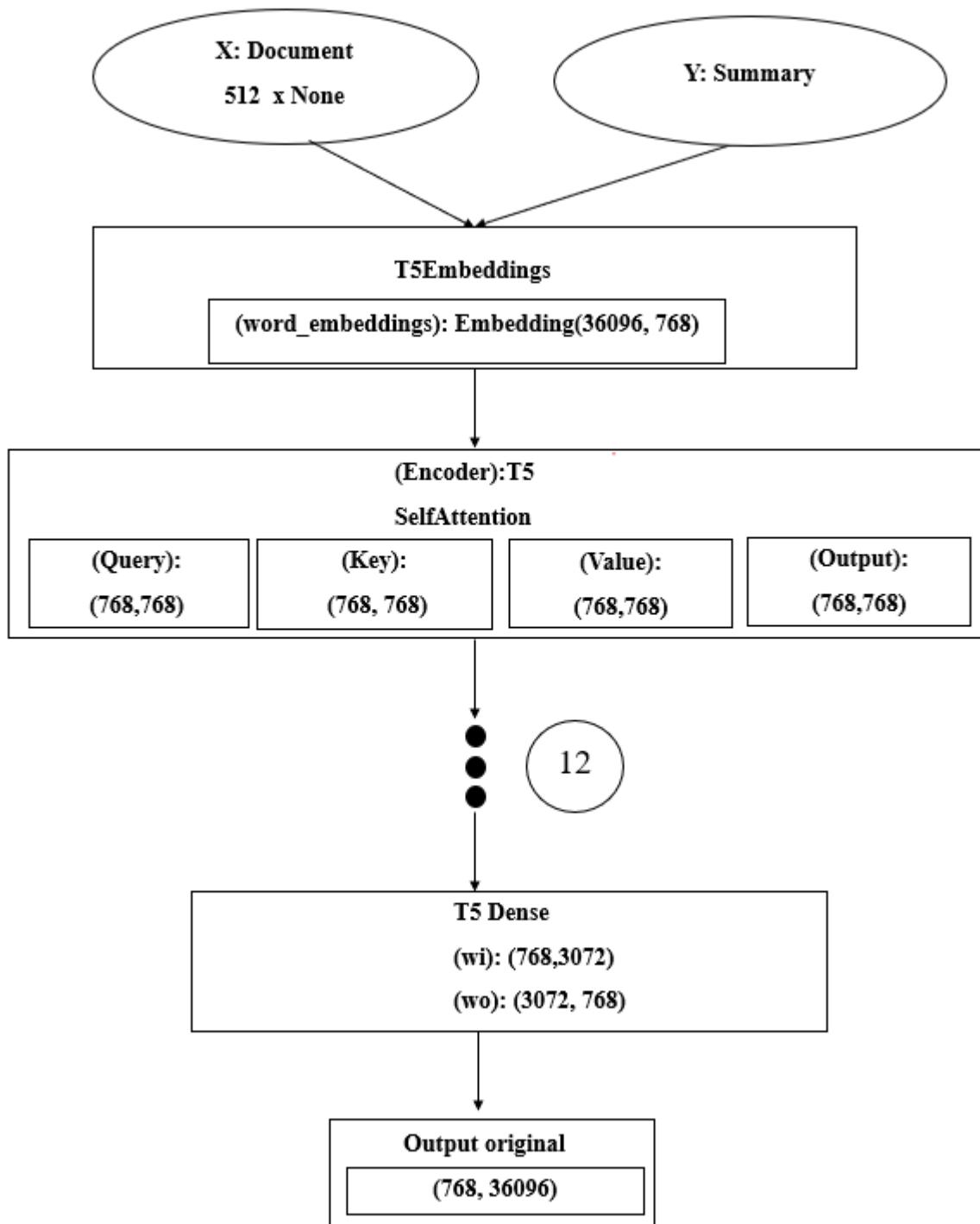
$$L = - \sum_i^N ((y_i \log(a_i)))$$

Trong đó:

- $N$ : số lượng mẫu dữ liệu.

-  $y_i$ : Chỉ mục của từ thực tế trong câu trả lời/tóm tắt.

-  $a_i$ : dự đoán từ vựng thứ  $i$  tại mỗi bước.



Hình 4.12. Sơ đồ quá trình Full Fine-Tuning T5

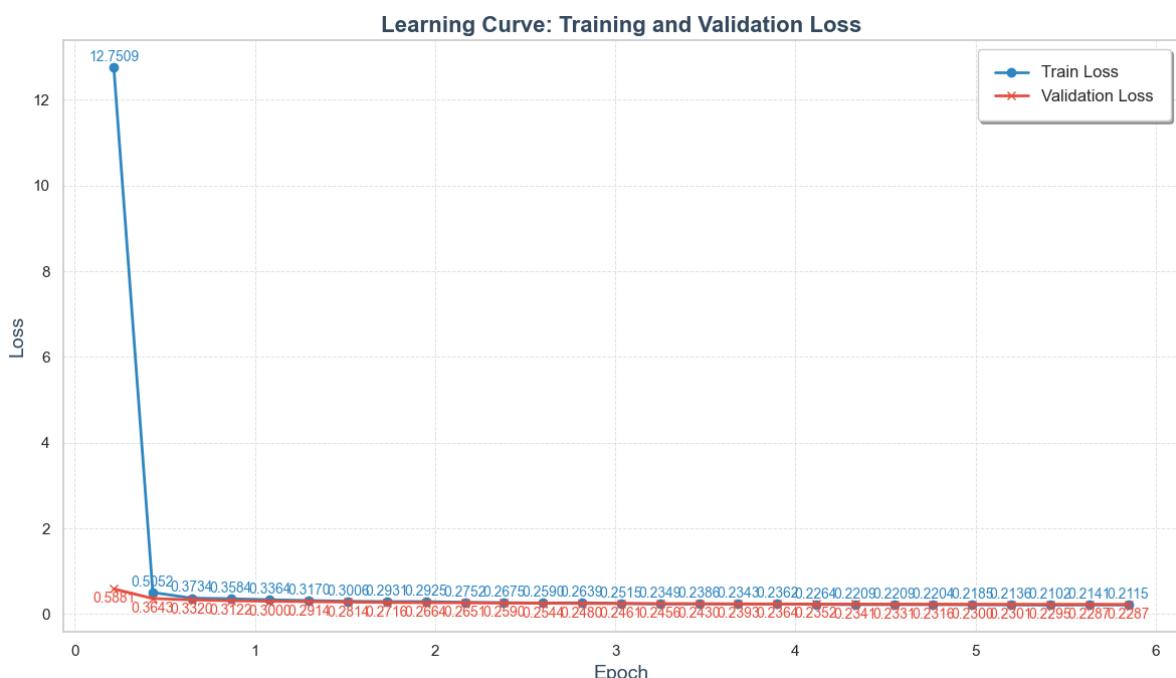
Bảng 4.10. Bảng cấu hình siêu tham số cho quá trình huấn luyện T5

<b>Siêu tham số</b>	<b>Giá trị</b>
Tốc độ học tập (learning_rate )	2e-5
Số mẫu trong mỗi batch huấn luyện (per_device_train_batch_size)	6
Số mẫu trong mỗi batch khi đánh giá (per_device_eval_batch_size)	6
Số bước huấn luyện để tích lũy gradient trước khi cập nhật trọng số (gradient_accumulation_steps)	4
Số lần toàn bộ tập dữ liệu được đưa qua mô hình (num_train_epochs)	6
Hệ số regularization để ngăn ngừa overfitting (weight_decay)	0.1
Giới hạn độ lớn gradient để tránh quá trình huấn luyện không ổn định (max_grad_norm)	0.4
Tỷ lệ bước huấn luyện để làm ám mô hình trước khi bắt đầu giảm tốc độ học (warmup_ratio)	0.1
Phương pháp điều chỉnh tốc độ học (lr_scheduler_type)	linear
Các nhãn được sử dụng trong nhiệm vụ huấn luyện (label_names)	['input_text', 'summary_text']

**Bảng 4.11. Bảng kết quả thu được sau quá trình T5 Full-fine-tuning**

<b>Thông số</b>	<b>Giá trị</b>
Tổng số bước huấn luyện (global_step)	2766

Mất mát huấn luyện (train/loss)	0.208
Mất mát đánh giá (eval/loss)	0.22687357664108276
Tốc độ học tập (train/learning_rate)	5.303334672559261e-07
Độ lớn gradient (train/grad_norm)	250543.234375
Số epoch (train/epoch)	5/6
Thời gian huấn luyện (train_runtime)	5 giờ 31 phút 43.81 giây



**Hình 4.13. Biểu đồ learning curve mô tả loss của quá trình full fine-tuning**

Nhận xét:

- Quá trình fine-tuning đã diễn ra trong vòng 5 epoch. Cụ thể, đến epoch thứ 5, hàm loss trên tập kiểm định đạt 0.229086, trong khi trên tập huấn luyện là 0.208000. Sự chênh lệch nhỏ giữa hai giá trị này cho thấy mô hình đạt được mức độ khai quát hóa tốt mà không rơi vào tình trạng overfitting.

- Biểu đồ learning cho thấy đường loss trên tập huấn luyện và tập kiểm định gần như song hành. Qua đó cho thấy chất lượng của quá trình tinh chỉnh và khả năng áp dụng của mô hình cho các dữ liệu thực tế.

Nhìn chung, kết quả cho thấy mô hình đạt được ổn định và hiệu quả. Việc mô hình không bị overfitting càng được chứng minh rõ ràng qua sự chênh lệch nhỏ giữa loss trên tập huấn luyện và tập kiểm định.

#### 4.2.4. Đánh giá mô hình cho bài toán tóm tắt văn bản

##### 4.2.4.1. Phương pháp đánh giá

Trong bài toán Summary Text, mục tiêu chính là tóm tắt được ý chính từ đoạn văn bản được cung cấp đầu vào. Để đánh giá độ chính xác của mô hình, sử dụng thước đo phổ biến như ROUGE. Thước đo này giúp xác định mức độ chính xác của câu trả lời mà mô hình dự đoán so với câu trả lời thực tế.

##### 4.2.4.2. Kết quả thu được

Tóm tắt một ví dụ cơ bản với một đoạn văn như sau:

- Đoạn văn

Vị trí xác định sản lượng bán buôn điện nông thôn Giá bán buôn điện nông thôn được áp dụng đối với các đơn vị bán lẻ điện nông thôn cho sản lượng điện mua buôn đếm được tại công tơ đo đếm tổng đặt tại trạm biến áp. Giá bán buôn điện sinh hoạt nông thôn như sau: Giá bán buôn điện sinh hoạt nông thôn và giá bán tại công tơ tổng do Tổng công ty Điện lực, Công ty Điện lực hoặc đơn vị được ủy quyền thuộc Tập đoàn Điện lực Việt Nam bán cho đơn vị bán lẻ điện nông thôn. Nguyên tắc xác định số định mức sử dụng điện cho hộ sử dụng điện sinh hoạt sau công tơ tổng được quy định tại điểm b khoản 1 mục IV Phụ lục Ban hành kèm Thông tư này. Giá bán buôn điện sử dụng cho mục đích khác (ngoài mục đích sinh hoạt tại công tơ tổng mua buôn điện nông thôn là 230 đồng/kWh

- Kết quả tóm tắt

Quy định về giá bán điện và hướng dẫn thực hiện Giá bán buôn điện nông thôn

Ta thu được điểm ROUGE với kết quả như bảng dưới đây:

**Bảng 4.12. Các chỉ số ROUGE trên tệp dữ liệu pháp luật**

<b>ROUGE - 1</b>	<b>ROUGE - 2</b>	<b>ROUGE - L</b>
0.6674	0.4969	0.578

ROUGE-1 đạt 0.6674, cho thấy mô hình có khả năng giữ lại các từ khóa và thông tin cốt lõi từ văn bản gốc. Tuy nhiên, ROUGE-2 với 0.4969 và ROUGE-L đạt 0.578 đã phản ánh mô hình gặp khó khăn trong việc duy trì các cụm từ liên tiếp, mối quan hệ ngữ nghĩa phức tạp và cần cải thiện về khả năng duy trì mối quan hệ câu và độ chính xác trong tóm tắt.

Nhìn chung, mô hình T5 cho thấy hiệu suất ổn định, nhưng vẫn có tiềm năng cải thiện, đặc biệt trong việc duy trì các mối quan hệ ngữ nghĩa phức tạp và cấu trúc câu để tạo ra các tóm tắt chính xác hơn và gần gũi hơn với văn bản nguồn.

### **4.3. BÀI TOÁN TRUY XUẤT TĂNG CƯỜNG**

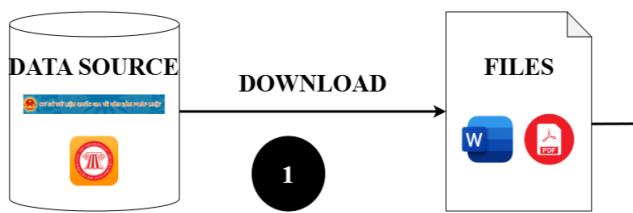
#### **4.3.1. Giới thiệu về bài toán**

Truy xuất tăng cường (RAG) là một phương pháp tiên tiến trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên, kết hợp giữa các mô hình truy xuất thông tin và mô hình sinh văn bản để tìm kiếm thông tin có liên quan từ một cơ sở tri thức lớn, tận dụng thông tin này làm ngữ cảnh đầu vào cho mô hình sinh văn bản. Việc này giúp tăng cường khả năng sinh câu trả lời chính xác và có tính tham chiếu cao, giảm thiểu rủi ro phát sinh thông tin sai lệch.

RAG đã được ứng dụng rộng rãi trong các lĩnh vực như trợ lý ảo, hỏi đáp thông minh (question answering), và tổng hợp nội dung. Điểm mạnh của RAG nằm ở khả năng tận dụng cơ sở tri thức có sẵn, kết hợp với sức mạnh xử lý của các mô hình ngôn ngữ lớn (LLM), để tạo ra các giải pháp linh hoạt và hiệu quả.

### 4.3.2. Xây dựng bộ dữ liệu cho bài toán truy xuất tăng cường

#### 4.3.2.1. Quá trình thu thập dữ liệu



**Hình 4.14. Sơ đồ quá trình thu thập dữ liệu cho bài toán truy xuất tăng cường**

Quá trình thu thập dữ liệu cho bài toán truy xuất tăng cường được thực hiện từ hai nguồn chính, đó là Thư viện Pháp luật (<https://thuvienphapluat.vn/>) và Cơ sở dữ liệu Quốc gia Văn bản pháp luật (<https://vbpl.vn/pages/portal.aspx>). Đây là các kho dữ liệu trực tuyến chứa các văn bản quy phạm pháp luật quan trọng của Việt Nam, bao gồm các Luật, Quyết định, Pháp lệnh, Nghị quyết và các văn bản pháp lý khác. Nhóm đã tiến hành tham khảo 26 loại văn bản pháp luật khác nhau, bao gồm những loại văn bản có ảnh hưởng lớn đến các quy trình pháp lý và quản lý hành chính, lựa chọn ra 99 file văn bản pháp lý để xây dựng kho dữ liệu cho bài toán truy xuất tăng cường. Các file này bao gồm 30 files Luật, 28 files Quyết định, 22 files Nghị quyết và 19 files Pháp lệnh. Quyết định chọn lựa các văn bản này được dựa trên tính đại diện của chúng trong hệ thống pháp lý hiện hành cũng như tính chất đa dạng của các loại văn bản pháp luật.

Các file này được tải xuống thủ công từ các trang web của Thư viện Pháp luật và Cơ sở dữ liệu Quốc gia Văn bản pháp luật. Sau khi tải về, các văn bản này được chuyển đổi sang định dạng .docx để thuận tiện cho quá trình tổ chức và xử lý dữ liệu. Việc chuyển đổi này đảm bảo rằng dữ liệu có thể được đọc và xử lý một cách dễ dàng hơn, đồng thời cũng giúp việc tạo ra các chỉ mục và tổ chức thông tin trở nên hiệu quả hơn trong quá trình truy xuất tăng cường.

#### 4.3.2.2. Quá trình tiền xử lý dữ liệu

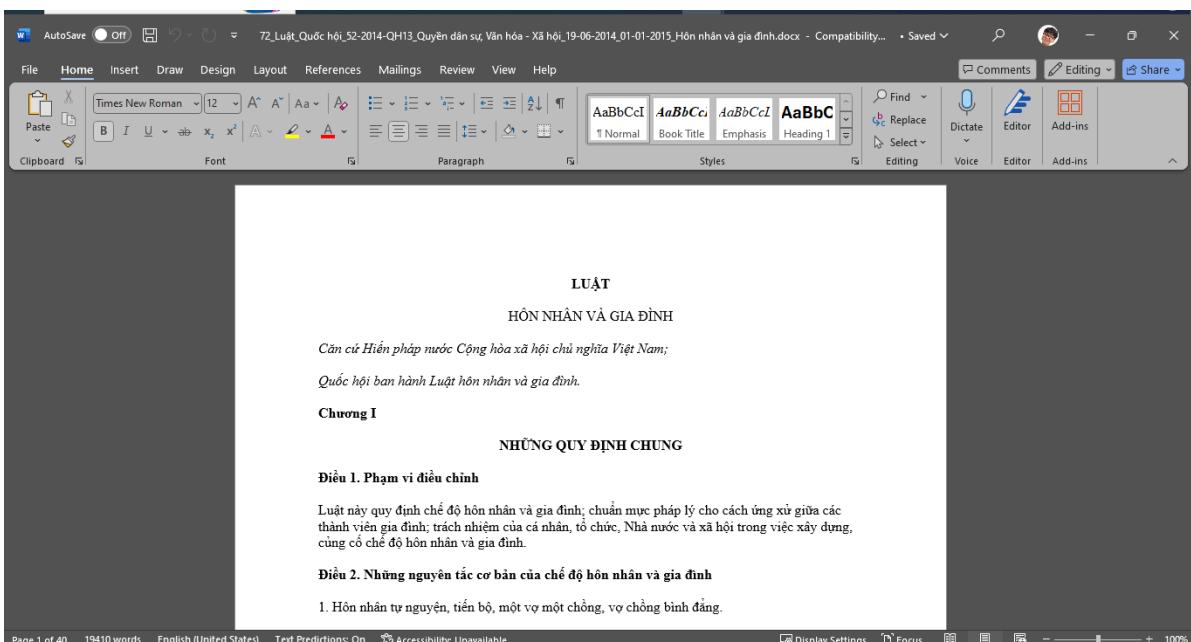
Sau khi hoàn tất thu thập các file văn bản, bước tiếp theo là thực hiện việc tiền xử lý dữ liệu. Đầu tiên, các file văn bản được đổi tên theo một cú pháp chuẩn hóa nhằm tạo thuận lợi cho quá trình tổ chức và tìm kiếm sau này. Cú pháp đặt tên file theo định

dạng: STT\_Loại văn bản\_Nơi ban hành\_Số hiệu\_Lĩnh vực – ngành\_Ngày ban hành\_Ngày hiệu lực\_Chủ đề. Ví dụ:

1\_Luật\_Quốc hội\_10-2022-QH15\_Quyền dân sự\_10-11-2022\_01-07-2023\_Thực hiện dân chủ ở cơ sở.docx

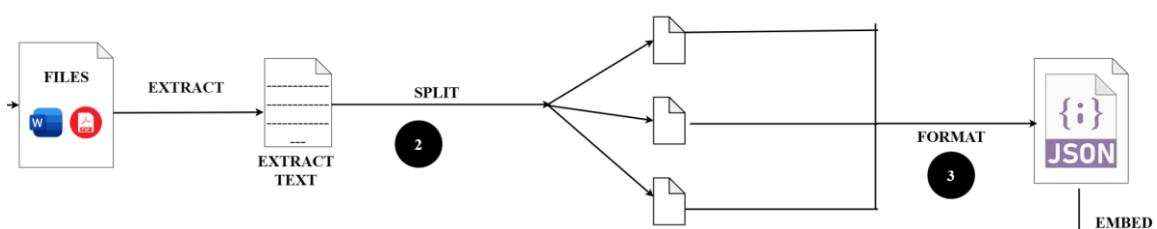
Mục đích của việc này là thiết lập metadata cho các văn bản, giúp tạo một bộ lọc trong quá trình tìm kiếm, từ đó nâng cao độ chính xác của kết quả truy vấn.

Sau khi hoàn thành việc đổi tên các file, tiến hành mở từng file và loại bỏ các thành phần không cần thiết có trong nội dung văn bản như header, footer, phụ lục, bảng biểu, v.v. Chỉ giữ lại các phần nội dung chính gồm Chương, Mục, Tiêu mục, và Điều. Những phần này sẽ được sử dụng để tạo metadata và làm cơ sở cho việc truy xuất dữ liệu sau này. Việc làm sạch dữ liệu này giúp đảm bảo rằng chỉ những thông tin quan trọng và có liên quan đến bài toán truy xuất mới được giữ lại, cải thiện hiệu quả và độ chính xác của hệ thống truy vấn văn bản.



**Hình 4.15. File văn bản sau khi tiền xử lý**

#### 4.3.2.3. Quá trình tổ chức dữ liệu



#### **Hình 4.16. Sơ đồ quá trình tổ chức dữ liệu**

Trong phần tổ chức dữ liệu, mục tiêu chính là chuẩn bị và xử lý dữ liệu từ các file .docx, sau đó lưu trữ kết quả dưới dạng file .json để phục vụ cho lưu trữ dữ liệu vào vector database. Trong văn bản quá trình tìm hiểu, nhóm nhận thấy các tình huống về Pháp luật thường được trả lời với 2 kiểu tham chiếu, trích dẫn: theo Điều và theo Khoản. Chính vì thế, nhóm quyết định tổ chức dữ liệu xử lý được từ các file .docx về 2 kiểu: mức Điều và mức Khoản.

##### a) Tổ chức dữ liệu từ tên file

Đầu tiên, các tên file .docx trong thư mục được xử lý để trích xuất các thông tin metadata quan trọng. Tên file được phân tách thành các phần dựa trên dấu gạch dưới ("\_"), và các thông tin như "số thứ tự", "loại văn bản", "ngày ban hành", "chủ đề", v.v. được trích xuất và xử lý. Các từ khóa từ tên file cũng được làm sạch và chuẩn hóa để dễ dàng sử dụng trong các bước tiếp theo.

##### b) Tổ chức dữ liệu ở mức Điều

Mỗi file .docx được xử lý để trích xuất nội dung các đoạn văn bản. Quá trình này gồm các bước:

Bước 1: Chương, Mục, Tiêu mục, Điều - Tiến hành duyệt qua từng đoạn văn trong file .docx, nếu đoạn văn đó bắt đầu với từ đầu tiên là “Chương”, “Mục”, “Tiêu mục” hay “Điều” thì tiến hành lưu lại đoạn văn này để đánh dấu tiêu đề.

Bước 2: Nội dung chi tiết - Mỗi đoạn văn bản sau các tiêu đề đã được đánh dấu lại sẽ được lưu thành nội dung của “Điều”. Trong trường hợp “Điều” có nội dung quá dài, vượt qua max\_tokens = 2048 token thì tiến hành tách ra thành các phần nhỏ hơn dựa trên số thứ tự (ví dụ: "1.", "2.", "3.", ...) để dễ dàng kiểm soát số lượng từ trong mỗi phần, đồng thời đảm bảo không vượt quá giới hạn max\_tokens. Việc tách ra này nhằm thỏa mãn số token đối đa mà mô hình embedding có thể thực hiện để lưu data vào vector database.

Bước 3: Xử lý các nội dung thành từ khóa (key words) - Sau khi đã lưu được các tiêu đề và nội dung kèm theo, tiến hành tạo từ khóa để bổ sung cho metadata sử dụng cho việc filter lúc truy vấn. Các bộ key words cho các tiêu đề và nội dung được thực hiện thông qua các bước:

- Trích xuất từ khóa từ các nội dung: Các từ khóa được trích xuất bằng cách loại bỏ các dấu câu, trừ các ký tự đặc biệt là “/” và “-”. Sau đó xử lý loại bỏ bớt các từ trùng lặp, chỉ còn các từ duy nhất.
- Xử lý chuyển đổi và lower case: Các từ khóa được chuyển sang chữ thường, ngoại trừ các từ có chứa ký tự đặc biệt như số La Mã hoặc các ký tự “/” và “-” bên trong nó.

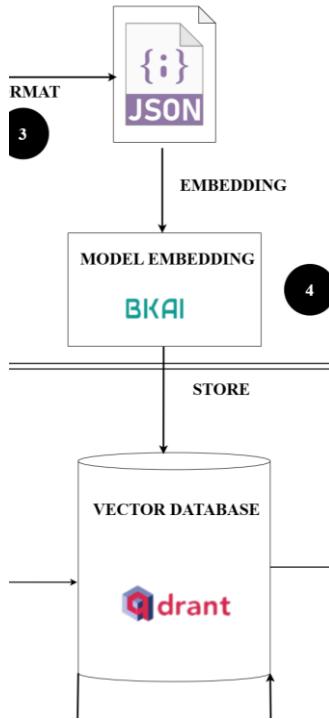
Bước 4: Tiến hành gộp metadata đã xử lý từ tên file và metadata đã xử lý từ nội dung file này thành bộ data cuối cùng.

#### c) Tổ chức dữ liệu ở mức Khoản

Đối với mức Khoản, cách tổ chức giống hệt như mức Điều, nhưng có một số thay đổi. Đó là:

- Không còn giới hạn max\_tokens cho nội dung Khoảng.
- Nội dung của "Content" lúc này lưu trữ nội dung của Khoản.
- Nội dung của "combine\_Article\_Content" lúc này là gộp từ Điều và Khoản.
- Tạo thêm thuộc tính "Article-Section" - Khoản, "Article-Section\_Keywords" - từ khóa được tạo từ nội dung Khoản, "combine\_Article\_Content\_Keywords" - các từ khóa được tạo từ nội dung gộp lại của Điều và Khoản.

#### 4.3.2.4. Quá trình đưa dữ liệu vào vector database



**Hình 4.17. Sơ đồ quá trình lưu dữ liệu vào vector database**

##### a) Lựa chọn vector database

Sau khi hoàn thành việc xử lý và tổ chức dữ liệu, bước tiếp theo là đưa dữ liệu vào một hệ cơ sở dữ liệu vector để phục vụ cho các tìm kiếm và phân tích hiệu quả hơn. Trong phần này, nhóm đã chọn Qdrant làm cơ sở dữ liệu vector. Qdrant là một cơ sở dữ liệu mã nguồn mở, chuyên dụng để lưu trữ và truy vấn các vector, hỗ trợ các tác vụ như tìm kiếm gần đúng (nearest neighbor search) và xử lý dữ liệu không cấu trúc như văn bản hoặc hình ảnh. Qdrant cung cấp khả năng mở rộng và hiệu suất cao, giúp tối ưu quá trình tìm kiếm và phân tích dữ liệu văn bản dưới dạng vector.

##### b) Lựa chọn model embedding

Để chuyển đổi văn bản thành các vector, nhóm đã sử dụng mô hình bkai-foundation-models/vietnamese-bi-encoder từ Hugging Face (<https://huggingface.co/bkai-foundation-models/vietnamese-bi-encoder>). Mô hình này là một bi-encoder chuyên dụng cho ngôn ngữ tiếng Việt, có khả năng tạo ra các vector biểu diễn cho văn bản một cách chính xác, phù hợp cho các tác vụ như tìm kiếm và phân loại văn bản. Việc sử dụng mô hình này giúp tạo ra các vector có tính biểu

cảm cao, phản ánh được ý nghĩa ngữ nghĩa trong từng câu hoặc đoạn văn bản, từ đó phục vụ cho việc truy vấn hiệu quả trong Qdrant.

### c) Lưu dữ liệu

Khi lưu vào vector database Qdrant, mỗi điểm dữ liệu đều có 2 thuộc tính:

- page\_content - phần nội dung được embedding để lưu và sử dụng để tìm kiếm.
- metadata - các metadata sẽ được sử dụng trong quá trình filter, góp phần mang về kết quả truy vấn chính xác hơn.

Với hai bộ dữ liệu đã được tổ chức ở các mức Điều và Khoản, nhóm tiến hành lưu trữ chúng vào Qdrant dưới hai collection riêng biệt:

- Luat\_bkai\_Article\_More\_Keywords: Collection này lưu trữ dữ liệu ở mức Điều. Phần nội dung page\_content được sử dụng để embedding là thuộc tính "Article" nhằm tránh tình trạng lỗi khi model embedding bị giới hạn ở 2048 tokens (vì nội dung tiêu đề của Điều luôn nhỏ hơn ngưỡng này).
- Luat\_bkai\_Article-Section\_More\_Keywords: Collection này chứa dữ liệu ở mức Khoản, Phần nội dung page\_content được sử dụng để embedding là thuộc tính "combine\_Article\_Content" nhằm gia tăng độ chính xác khi sử dụng tìm kiếm ngữ nghĩa (trong trường hợp câu hỏi về nội dung Khoản hoặc hỏi về nội dung Điều).

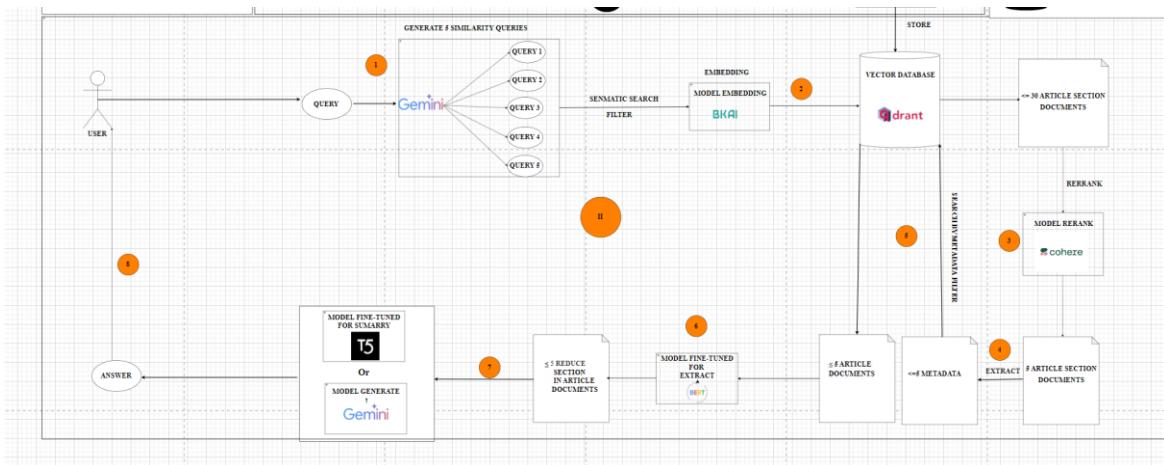
The screenshot shows the Qdrant application interface. On the left is a sidebar with icons for collections, search, and other functions. The main area is titled 'Collections' and contains a search bar. Below the search bar is a table with the following data:

Name	Status	Points (Approx)	Segments	Shards	Vectors Configuration (Name, Size, Distance)	Actions
Luat_bkai_Article_More_Keywords	● green	8757	8	1	default 768 Cosine	⋮
Luat_bkai_Article-Section_More_Keywords	● green	31834	8	1	default 768 Cosine	⋮

In the bottom left corner of the interface, it says 'v1.12.1'.

Hình 4.18. Các collections sau khi upload vào Qdrant

#### 4.3.4. Quá trình thực hiện truy xuất tăng cường



Hình 4.19. Sơ đồ quy trình thực hiện truy xuất tăng cường

##### 4.3.4.1. Giai đoạn sinh thêm câu hỏi tương tự

Sau khi nhận được câu hỏi từ người dùng, hệ thống tiến hành sinh thêm 5 câu hỏi tương tự về ngữ nghĩa thông qua mô hình **Gemini-1.5-pro**. Việc sinh ra các câu hỏi này mang lại nhiều lợi ích quan trọng:

- Mở rộng phạm vi tìm kiếm: Các câu hỏi tương tự giúp hệ thống có thể mở rộng phạm vi tìm kiếm, từ đó cung cấp nhiều câu trả lời chính xác hơn.
- Cải thiện độ chính xác: Việc sinh ra câu hỏi tương tự giúp hệ thống không bị giới hạn bởi cách diễn đạt cụ thể của người dùng, từ đó cải thiện độ chính xác của các câu trả lời.

Sau khi sinh ra 5 câu hỏi tương tự, hệ thống kết hợp chúng với câu hỏi gốc để tiếp tục thực hiện các bước tiếp theo trong quá trình xử lý và tìm kiếm thông tin, đảm bảo rằng kết quả trả về sẽ phong phú và chính xác hơn. Dưới đây là ví dụ:

#### Câu hỏi gốc:

người trên bao nhiêu tuổi thì phải đăng ký căn cước công dân?

#### Các câu hỏi tương tự được sinh ra:

1. Độ tuổi đăng ký căn cước công dân theo quy định pháp luật Việt Nam?
2. Quy định về độ tuổi làm căn cước công dân ở Việt Nam?
3. Khi nào công dân Việt Nam bắt buộc phải đăng ký căn cước công dân?
4. Trẻ em bao nhiêu tuổi phải làm căn cước công dân theo luật định?
5. Luật Căn cước công dân quy định độ tuổi nào phải đăng ký?

#### *4.3.4.2. Giai đoạn truy xuất*

##### a) Truy xuất ở mức Khoản

Sau khi có bộ 6 câu hỏi (bao gồm câu hỏi gốc và 5 câu hỏi tương tự), hệ thống tiến hành tìm kiếm kết quả liên quan trong cơ sở dữ liệu vector ở mức Khoản. Quá trình này được thực hiện như sau:

Bước 1: Sử dụng mô hình Embedding - Để tìm kiếm các tài liệu liên quan, mô hình bkai-foundation-models/vietnamese-bi-encoder (tương tự như khi thực hiện embedding dữ liệu vào Qdrant) được sử dụng. Mô hình này giúp chuyển các câu hỏi thành các vector, từ đó tìm kiếm các tài liệu tương thích trong cơ sở dữ liệu vector Qdrant.

Bước 2: Giảm thiểu tài liệu không liên quan - Filter metadata giúp loại trừ các kết quả không đáp ứng đủ các điều kiện nhất định, từ đó nâng cao độ chính xác của kết quả tìm kiếm.

Bước 3: Xử Lý và loại bỏ tài liệu trùng lặp - Với mỗi câu hỏi, hệ thống trả về 5 tài liệu liên quan, dẫn đến tổng cộng 30 tài liệu ( $6 \text{ câu hỏi} \times 5 \text{ tài liệu mỗi câu hỏi}$ ). Các tài liệu có nội dung trùng nhau sẽ được lọc bỏ. Kết quả nhận được không quá 30 tài liệu trả về.

Bước 4: Re-rank các tài liệu trả về: Đoạn rerank có thể sử dụng 2 phương pháp:

Phương pháp 1: Kết hợp giữa BM25 (câu hỏi, tài liệu) và biểu diễn nhúng (embedding). BM25 được sử dụng để đánh giá mức độ quan trọng của từng từ trong các tài liệu, với trọng số được gán là 0.3. Đồng thời, truy vấn và 30 tài liệu trả về được biểu diễn dưới dạng vector nhúng. Sử dụng độ đo cosine similarity giữa truy vấn và các vector tài liệu, một điểm số được tính toán với trọng số là 0.7. Kết hợp hai thang điểm này, tổng điểm của mỗi tài liệu được xác định, và top 5 tài liệu có điểm cao nhất được chọn làm kết quả.

Score =  $0.3 * \text{Bm25}(\text{câu hỏi}, \text{tài liệu}) + 0.7 * \text{Cosine similarity}(\text{câu hỏi}, \text{tài liệu})$

Phương pháp 2: Sử dụng dịch vụ API của bên thứ ba, chẳng hạn như Cohere, được pre-trained cho tác vụ reranking tài liệu. API này xếp hạng tài liệu dựa

trên mức độ liên quan nhất định đối với truy vấn đầu vào. Hệ thống tự động đánh giá và trả về top 5 tài liệu có độ phù hợp cao nhất.

#### b) Truy xuất ở mức điều

Sau khi hoàn tất quá trình truy xuất và re-rank các tài liệu ở mức Khoản, hệ thống tiếp tục bước xử lý ở mức Điều như sau:

Bước 1: Trích xuất metadata của tài liệu - Sau khi có kết quả cuối cùng ở mức Khoản, hệ thống tiến hành trích xuất các bộ metadata của từng tài liệu. Metadata này bao gồm các thông tin về "stt", "loai\_van\_ban", "noi\_ban\_hanh", "so\_hieu", "linhvuc\_nganh", "ngay\_ban\_hanh", "ngay\_hieu\_luc", "chu\_de", "Chapter", "Section", "Mini-Section", "Article".

Bước 2: Loại bỏ metadata trùng nhau - Trong trường hợp các tài liệu trả về ở mức Khoản đều thuộc cùng một Điều, tiến hành loại bỏ các bộ metadata trùng nhau giúp giảm thiểu việc trả về nhiều tài liệu giống nhau.

Bước 3: Sau khi xử lý metadata, sử dụng các bộ metadata để lọc ra các tài liệu liên quan trong bộ dữ liệu ở mức Điều. Với mỗi bộ lọc metadata, thực hiện chỉ trả về 1 tài liệu duy nhất. Điều này đảm bảo rằng mặc dù các tài liệu có thể đã được chia tách thành nhiều phần do giới hạn max\_tokens trong quá trình embedding lúc lưu trữ data, nhưng mỗi metadata vẫn đại diện cho một tài liệu hoàn chỉnh.

#### 4.3.4.3. Giai đoạn phản hồi

Sau khi đã có được các tài liệu liên quan từ kết quả truy xuất ở mức Điều, hệ thống tiến hành tạo phản hồi cho người dùng thông qua trình tự sau:

##### a) Sử dụng model BERT ở bài toán trả lời câu hỏi từ đoạn văn

Mô hình BERT, sau khi được fine-tuned cho bài toán trích xuất câu trả lời từ đoạn văn, được áp dụng để xác định các khoản trong điều luật có liên quan nhất đến câu hỏi của người dùng. Mục tiêu chính của bước này là giảm thiểu số lượng token cần xử lý trong các bước tiếp theo, giúp tối ưu hóa chi phí khi sử dụng các API bên thứ ba. Kết quả từ BERT là các đoạn văn bản ngắn gọn, có tính chọn lọc cao, cung cấp ngữ cảnh cần thiết để phục vụ cho các mô hình tiếp theo như T5 hoặc Gemini.

##### b) Sử dụng model T5 ở bài toán tóm tắt văn bản

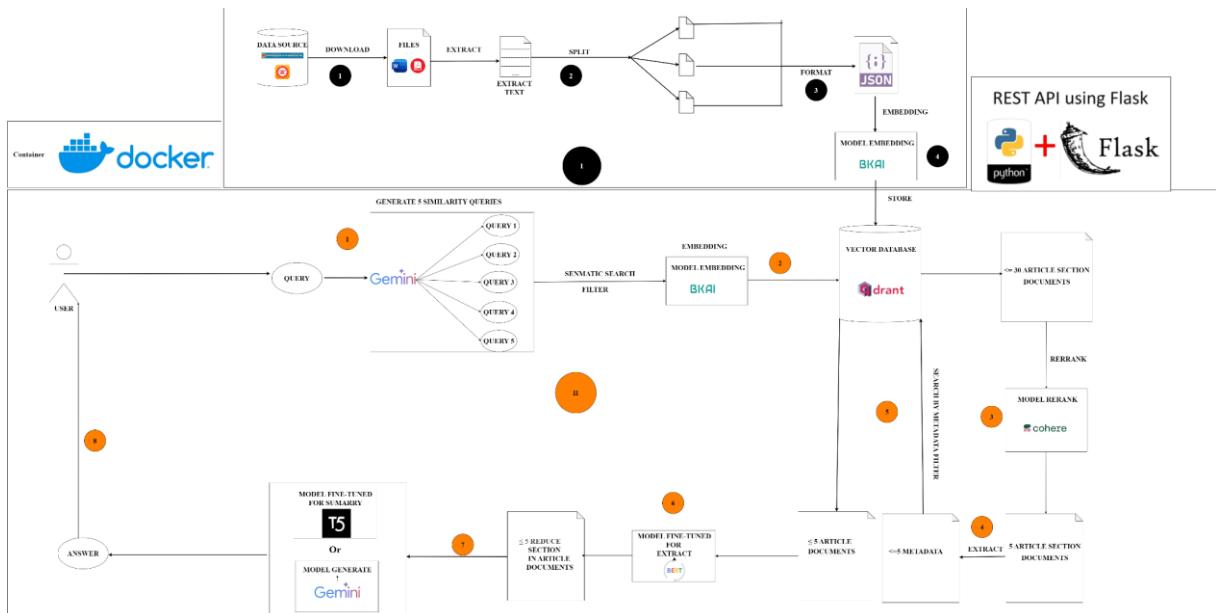
Mô hình T5 được fine-tuned đặc biệt cho nhiệm vụ tóm tắt văn bản, với mục tiêu xử lý và tổng hợp các ngữ cảnh đầu ra từ BERT. Các ngữ cảnh này, vốn là các khoán nhỏ được trích xuất từ điều luật, sẽ được T5 tóm tắt thành một đoạn văn ngắn gọn, dễ hiểu nhưng vẫn đảm bảo giữ lại đầy đủ thông tin cần thiết để trả lời câu hỏi của người dùng.

### c) Sử dụng model Gemini-1.5-pro

Gemini-1.5-pro, một API do Google cung cấp, được sử dụng để sinh câu trả lời dựa trên các ngữ cảnh đầu vào từ BERT. Do đây là một mô hình bên thứ ba không cho phép tinh chỉnh, việc tối ưu hóa kết quả tập trung vào tinh chỉnh prompt và điều chỉnh các tham số như top-p, top-k và temperature. Mô hình Gemini tận dụng đầu vào là các khoán nhỏ đã được BERT chọn lọc để sinh câu trả lời phù hợp nhất với câu hỏi. Quy trình này không chỉ cải thiện hiệu quả về mặt xử lý thông tin mà còn giúp giảm thiểu chi phí sử dụng API nhờ việc tối ưu hóa token qua bước xử lý ban đầu của BERT.

## 4.4. ÚNG DỤNG RAG VÀ CÁC MÔ HÌNH ĐÃ FINE-TUNING: T5, BERT VÀO WEBSITE HỎI ĐÁP PHÁP LUẬT

### 4.4.1. Ý tưởng sản phẩm



Hình 4.20. Quy trình hoạt động của hệ thống Hỏi đáp Pháp luật

Với mục tiêu xây dựng một chatbot hỗ trợ hỏi đáp và tư vấn pháp lý, nhóm đã nghiên cứu và fine-tune các mô hình ngôn ngữ lớn. Đặc biệt, mô hình BERT được lựa chọn để tối ưu hóa lượng token và làm rõ ngữ cảnh đầu vào, hỗ trợ hiệu quả cho các

mô hình như Gemini và T5 (fine-tuned) nhằm cải thiện chất lượng sinh câu trả lời. Bằng cách kết hợp với phương pháp RAG, nhóm đã phát triển một sản phẩm có giá trị thực tiễn, tích hợp trên nền tảng website, nhằm mang lại trải nghiệm tối ưu và hiệu quả cho người dùng.

#### 4.4.2. Công nghệ sử dụng cho website

Hệ thống được xây dựng bằng các công nghệ tiên tiến để đảm bảo hiệu suất và khả năng mở rộng. Docker được sử dụng để tạo môi trường triển khai đồng nhất, giúp quản lý dịch vụ và tối ưu hóa phát triển. Qdrant - một Vector Database, lưu trữ và truy vấn dữ liệu vector, hỗ trợ phương pháp Retrieval-Augmented Generation (RAG) để tìm kiếm thông tin nhanh chóng và chính xác. Backend sử dụng Flask để xây dựng API xử lý các chức năng chính, trong khi frontend được thiết kế bằng HTML, CSS và JQuery để tạo giao diện trực quan và kết nối với backend. Sự tích hợp này giúp hệ thống hoạt động hiệu quả và phù hợp với nhu cầu tư vấn pháp lý.

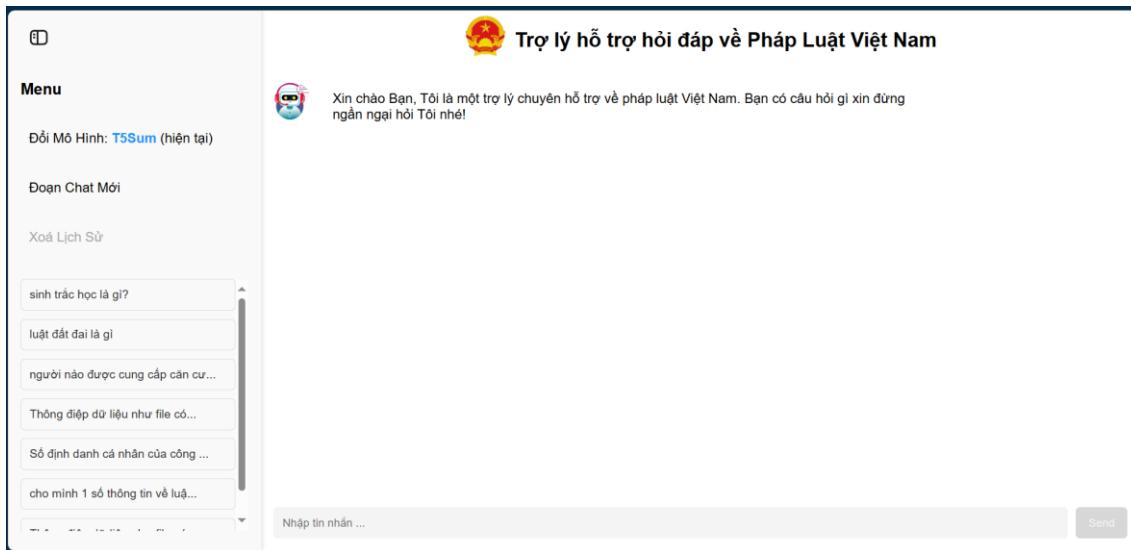
#### 4.4.3. Các chức năng và màn hình giao diện

##### 4.4.3.1. Tạo đoạn chat mới

Khi người dùng truy cập giao diện web lần đầu hoặc sau khi làm mới trang, giao diện sẽ trở về trạng thái mặc định và chức năng chat sẽ bị vô hiệu hóa, không cho phép nhập hay gửi tin nhắn. Để sử dụng chatbot, người dùng cần kích hoạt chức năng chat bằng cách nhấn nút “Đoạn Chat Mới”. Khi nút được chọn, một phiên trò chuyện mới sẽ được khởi tạo, cho phép người dùng bắt đầu nhập và gửi câu hỏi hoặc yêu cầu liên quan đến văn bản pháp luật.



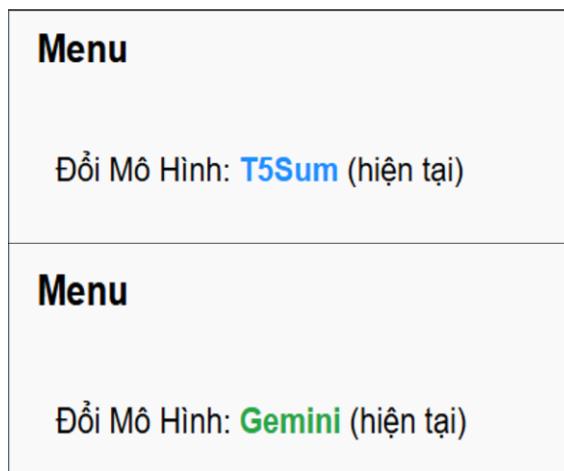
**Hình 4.21. Giao diện mặc định khi mới truy cập hoặc làm mới trang web**



**Hình 4.22. Giao diện mặc định khi bắt đầu một phiên Chat mới**

#### 4.4.3.2. *Đổi mô hình*

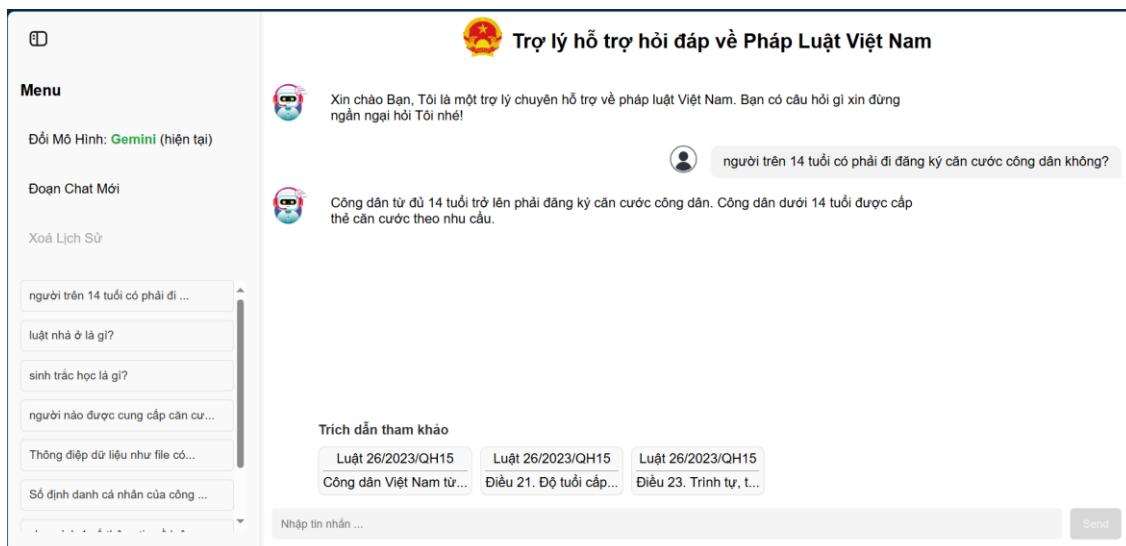
Mặc định, hệ thống sử dụng mô hình cuối là T5 ở bài toán tóm tắt văn bản để tạo phản hồi cho người dùng. Tuy nhiên, để đảm bảo tính linh hoạt và tối ưu hóa kết quả, người dùng có thể lựa chọn thay đổi mô hình khi cảm thấy phản hồi từ hệ thống chưa đáp ứng được yêu cầu. Việc này có thể được thực hiện thông qua thao tác nhấp vào nút "Đổi Mô Hình" trên giao diện. Khi lựa chọn này được kích hoạt, người dùng sẽ có tùy chọn thay đổi mô hình phản hồi giữa **T5** và **Gemini-1.5-pro**.



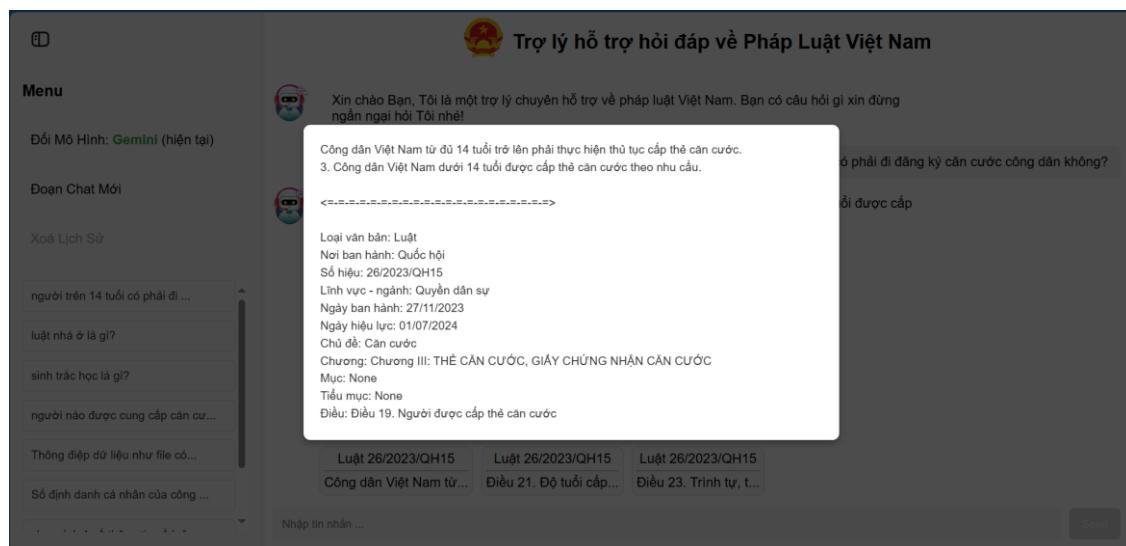
**Hình 4.23. Thay đổi trên giao diện sau khi người dùng lựa chọn Đổi Mô Hình**

#### 4.4.3.3. Hỏi đáp với chatbot

Khi người dùng gửi tin nhắn, hệ thống sẽ sử dụng mô hình đã chọn (T5 hoặc Gemini-1.5-pro) để tạo ra phản hồi chính xác và phù hợp. Nếu đây là tin nhắn đầu tiên trong phiên trò chuyện, hệ thống sẽ tự động lưu và thêm phiên chat này vào Lịch sử Chat của người dùng, giúp dễ dàng theo dõi và tìm lại các cuộc trò chuyện trước. Sau khi hệ thống xử lý xong, tin nhắn phản hồi sẽ hiển thị trên giao diện, kèm theo các tài liệu trích dẫn từ kho văn bản pháp luật. Người dùng có thể click vào các trích dẫn này để tham khảo chi tiết nguồn thông tin.



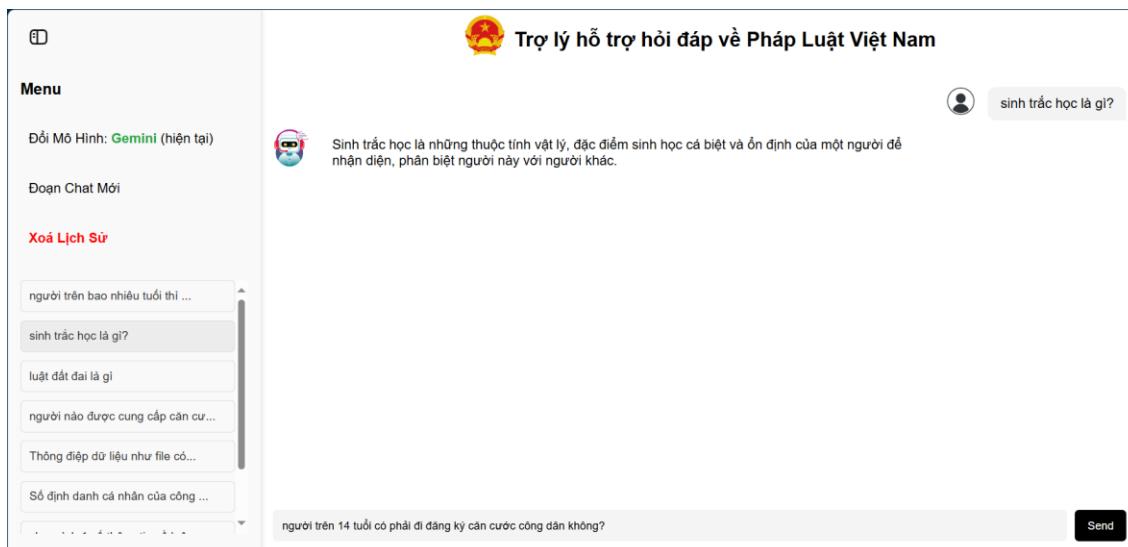
Hình 4.24. Giao diện hệ khi nhận được phản hồi từ hệ thống



Hình 4.25. Giao diện khi người dùng click vào Trích dẫn tham khảo

#### 4.4.3.4. Chức năng lịch sử chat

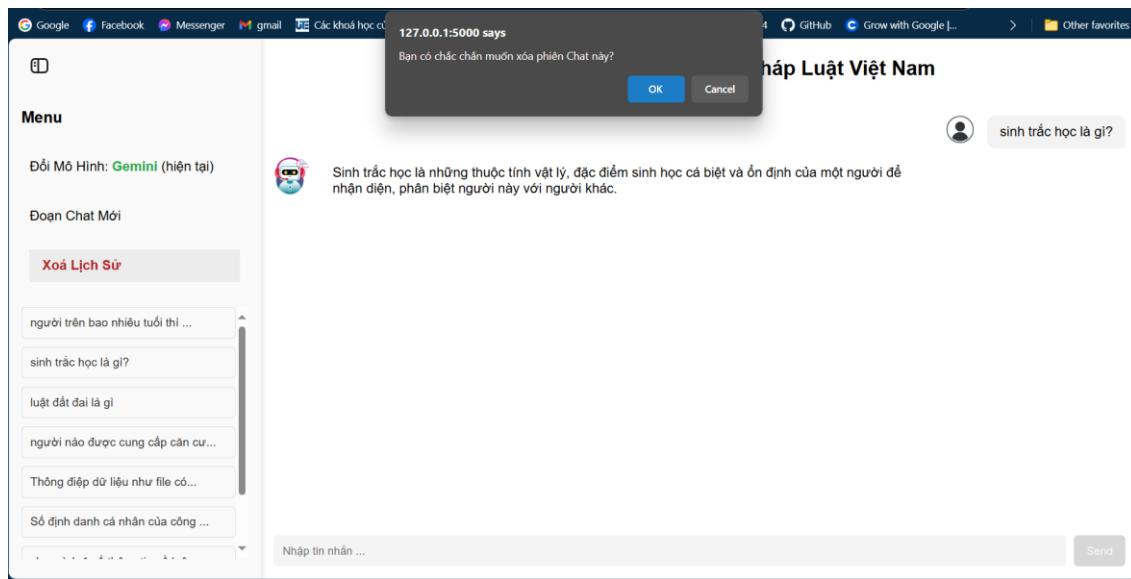
Hệ thống tự động cập nhật và lưu trữ các phiên chat vào Lịch sử Chat của người dùng, bao gồm các câu hỏi và phản hồi, giúp người dùng dễ dàng theo dõi và quản lý các cuộc trò chuyện trước. Người dùng có thể truy cập lại các phiên chat đã lưu bằng cách click vào chúng, hệ thống sẽ tải lại nội dung tin nhắn đã gửi, cho phép xem lại toàn bộ cuộc trò chuyện. Ngoài ra, người dùng có thể tiếp tục nhắn tin trong các phiên chat cũ để duy trì sự liên tục trong trao đổi thông tin.



Hình 4.26. Giao diện khi người dùng truy cập lại vào một phiên Chat cũ

#### 4.4.3.5. Chức năng xóa lịch sử chat

Chức năng xóa lịch sử chat cho phép người dùng xóa một phiên chat khỏi hệ thống khi họ click vào phiên chat trong Lịch sử Chat và chọn nút "Xóa Lịch Sử". Sau khi thao tác xóa được thực hiện, phiên chat sẽ bị loại bỏ vĩnh viễn, không thể truy cập lại. Chức năng này giúp người dùng bảo vệ sự riêng tư, tổ chức Lịch sử Chat một cách khoa học và ngăn ngừa việc truy cập vào các phiên chat không cần thiết.



**Hình 4.27. Giao diện khi người dùng thực hiện thao tác Xóa lịch sử chat**

## CHƯƠNG 5: KẾT LUẬN

### 5.1. KẾT QUẢ ĐẠT ĐƯỢC

Trong quá trình triển khai nghiên cứu và thực hiện tiểu luận chuyên ngành, nhóm đã đạt được những kết quả đáng khích lệ trong việc ứng dụng và đánh giá các mô hình ngôn ngữ lớn (LLMs) để giải quyết bài toán hỏi đáp pháp lý. Các mô hình như BERT và T5 đã được fine-tune thành công, nâng cao đáng kể khả năng hiểu ngữ cảnh và sinh câu trả lời. Những cải tiến này không chỉ giúp tối ưu hóa hiệu suất của hệ thống mà còn đạt được các chỉ số đánh giá tin cậy trong quá trình thử nghiệm.

Việc tích hợp phương pháp Retrieval-Augmented Generation (RAG) với các mô hình BERT và T5 đã cải thiện hiệu quả truy xuất và trích xuất ngữ cảnh pháp lý liên quan. Đồng thời, điều này giúp giảm chi phí vận hành thông qua việc tối ưu hóa truy vấn và giảm sự phụ thuộc vào các API từ bên thứ ba.

Dự án thể hiện khả năng tích hợp toàn diện các công nghệ tiên tiến, bao gồm Prompt Engineering, Vector Database (Qdrant) và các mô hình LLMs. Sự kết hợp này đã tạo nên một hệ thống hoàn chỉnh, có tiềm năng ứng dụng thực tiễn cao, hỗ trợ hiệu quả các nhu cầu tư vấn pháp lý cho cả cá nhân và tổ chức.

### 5.2. HẠN CHẾ

Một thách thức lớn là thời gian truy xuất dữ liệu trong cơ sở dữ liệu vector. Hiện tại, nhóm sử dụng các mô hình mã nguồn mở và tiến hành chunking dữ liệu thành các đoạn nhỏ trước khi lưu trữ vào Vector Database. Tuy nhiên, do chưa fine-tune mô hình cho tác vụ tìm kiếm tài liệu tương đồng, hệ thống đôi khi trả về kết quả không chính xác hoặc không liên quan chặt chẽ đến câu hỏi của người dùng.

Để khắc phục, nhóm đã áp dụng giải pháp tạm thời bằng cách bổ sung các bộ lọc dựa trên các trường thông tin như tên bộ luật, chương, mục, tiêu mục... Tuy giải pháp này giúp tăng độ chính xác của truy xuất, nhưng lại làm gia tăng thời gian xử lý, gây trở ngại khi mở rộng hoặc tích hợp thêm dữ liệu mới.

Một rào cản khác là hạn chế về dữ liệu và tài nguyên tính toán. Bộ dữ liệu cho bài toán tóm tắt văn bản bằng T5 còn hạn chế, dẫn đến hiệu suất mô hình chưa đạt mức tối ưu khi áp dụng vào thực tế. Đồng thời, cơ sở hạ tầng hiện có chưa cho phép nhóm

triển khai các mô hình lớn hơn như Gemma 2 2B của Google hay Llama-3.1-8B của Meta.

Hiện tại, nhóm chỉ thực hiện fine-tuning trên các mô hình như T5 Base (220 triệu tham số) và BERT Base (110 triệu tham số). Nếu có thêm tài nguyên như GPU hỗ trợ CUDA Framework của NVIDIA, quá trình fine-tuning và suy luận của mô hình có thể được cải thiện đáng kể cả về hiệu suất lẫn tốc độ.

Vì việc điều chỉnh siêu tham số (hyperparameters) vẫn là một thách thức lớn. Hiệu quả của mô hình phụ thuộc nhiều vào các bộ dữ liệu và siêu tham số, trong khi việc tối ưu hóa đòi hỏi thời gian và tài nguyên đáng kể.

Đặc biệt, trong các bài toán pháp lý phức tạp đòi hỏi suy luận sâu hoặc phân tích đa chiều, độ chính xác của hệ thống vẫn còn hạn chế. Mặc dù đã sử dụng BERT để tối ưu hóa việc phân tích token và làm rõ ngữ cảnh, kết hợp với T5 để tóm tắt nội dung, nhưng hệ thống chưa đáp ứng tốt các bài toán pháp lý yêu cầu sự suy luận logic cao.

### **5.3. HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI**

Nhằm khắc phục những hạn chế và phát triển hệ thống, nhóm sẽ tiếp tục tối ưu hóa các mô hình trích xuất thông tin và hướng tới fine-tuning các mô hình lớn hơn với nhiều tham số hơn. Trong các bước truy xuất tăng cường, nhóm dự kiến sẽ fine-tuning các mô hình embedding để tối ưu hơn cho việc tìm kiếm ngữ nghĩa, kết hợp với fine-tuning mô hình re-ranking để cải thiện chất lượng sắp xếp lại các ngữ cảnh liên quan.

Một hướng phát triển khác là tích hợp thêm các thành phần hỗ trợ, chẳng hạn như Agent-based Systems. Các agent này có khả năng tự động hóa việc gửi truy vấn của người dùng đến các nguồn dữ liệu phù hợp, giúp tăng cường khả năng suy luận thông minh. Ví dụ, khi người dùng đặt các câu hỏi phức tạp hoặc trừu tượng, hệ thống có thể kết hợp sử dụng search engine như Google để truy xuất thông tin từ bên ngoài, đồng thời kết hợp với dữ liệu trong cơ sở dữ liệu vector để tạo ra các ngữ cảnh liên quan và cần thiết nhất.

Ngoài ra, việc cải thiện hiệu suất và độ chính xác của hệ thống thông qua các mô hình ngôn ngữ lớn hơn như Llama 3 hoặc Gemma 2 cũng sẽ là mục tiêu dài hạn,

mở rộng khả năng ứng dụng của hệ thống trong thực tế, đặc biệt trong lĩnh vực tư vấn và hỗ trợ pháp lý thông minh.

## TÀI LIỆU THAM KHẢO

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin , “Attention Is All You Need ” ,arXiv:1706.03762,02-Aug-2023. [Online]. Available:  
<https://arxiv.org/pdf/1706.03762.pdf>. [Accessed: 1-Sep-2024].
- [2] “Retrieval Augmented Generation (RAG) for LLMs”, *Prompt Engineering Guide*, 28-Otc-2024. [Online]. Available: <https://www.promptingguide.ai/research/rag>. [Accessed: 1-Oct-2024].
- [3] “Advanced RAG Unveiled - I: Naive RAG Vs. Advanced RAG”, *MYSCALE*, 27-Aug-2024. [Online]. Available: <https://myscale.com/blog/naive-rag-vs-advanced-rag/>. [Accessed: 1-Oct-2024].
- [4] “What is a vector database?”, *Cloudflare*. [Online]. Available:  
<https://www.cloudflare.com/learning/ai/what-is-vector-database/>. [Accessed: 23-Nov-2024].
- [5] Sahin Ahmed, “LLM Prompt Engineering for Beginners: What It Is and How to Get Started”, *Medium*, 15-Apr-2024. [Online]. Available: <https://medium.com/thedepthub/llm-prompt-engineering-for-beginners-what-it-is-and-how-to-get-started-0c1b483d5d4f>. [Accessed: 1-Oct-2024].
- [6] “Qdrant Documentation”, *Qdrant*. [Online]. Available: <https://qdrant.tech/documentation/>. [Accessed: 30-July-2024].
- [7] “Qdrant”, *LangChain*. [Online]. Available:  
<https://python.langchain.com/docs/integrations/vectorstores/qdrant/>. [Accessed: 30-July-2024].
- [8] N.T.T Dương, “Chatbot và cách xây dựng chatbot ” , viblo,10-Jul-2024. [Online]. Available:  
<https://viblo.asia/u/Duongntt>. [Accessed: 11-Oct-2024].
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova , “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding ” ,arXiv:1810.04805,24-May-2019. [Online]. Available:  
<https://arxiv.org/pdf/1810.04805.pdf>. [Accessed: 23-Oct-2024].
- [10] Akshit Mehra,“8 Challenges Of Building Your Own Large Language Model ” , labellerr,19-Sep-2024. [Online]. Available:  
<https://www.labellerr.com/blog/challenges-in-development-of-langs/>. [Accessed: 22-Oct-2024].
- [11] Prasad Mahamulkar,“Fine-Tune Large Language Model in a Colab Notebook ” , medium,22-Jan-2024. [Online]. Available:  
<https://medium.com/@prasad.mahamulkar/fine-tune-large-language-model-in-a-colab-notebook-5a2a2a2a2a2a>. [Accessed: 25-Oct-2024].
- [12] Reece Shuttleworth, Jacob Andreas, Antonio Torralba, Pratyusha Sharma , “LORA VS FULL FINE-TUNING: AN ILLUSION OF EQUIVALENCE ” ,

- arXiv:2410.21228,28 Oct 2024 . [Online]. Available:  
<https://arxiv.org/pdf/2410.21228.pdf>. [Accessed: 4-Nov-2024].
- [13] Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, Tuo Zhao,“ LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models”, arXiv:2310.08659,28 Nov 2023 . [Online]. Available:  
<https://arxiv.org/pdf/2310.08659.pdf>. [Accessed: 8-Nov-2024].
- [14] Sahana Ramnath, Preksha Nema, Deep Sahni, Mitesh M. Khapra,“ Towards Interpreting BERT for Reading Comprehension Based QA”, arXiv:2010.08983,18 Oct 2020 . [Online]. Available:  
<https://arxiv.org/pdf/2010.08983.pdf>. [Accessed: 10-Sep-2024].
- [15] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang,“ SQuAD: 100,000+ Questions for Machine Comprehension of Text”, arXiv:1606.05250,11 Oct 2016 . [Online]. Available:  
<https://arxiv.org/pdf/1606.05250.pdf>. [Accessed: 15-Sep-2024].
- [17] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”, arXiv:1910.10683v4, 19 Sep 2023 . [Online]. Available:  
<https://arxiv.org/pdf/1910.10683.pdf>. [Accessed 17-Oct-2024]
- [18] Chin-Yew Lin, “ROUGE: A Package for Automatic Evaluation of Summaries”, July-2004 . [Online]. Available: <https://aclanthology.org/W04-1013.pdf>. [Accessed: 27-Oct-2024]