ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN

VŨ THỊ HẰNG

XÂY DỰNG TRỢ LÝ ẢO HỖ TRỢ TƯ VẤN TUYỂN SINH ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN NGHIÊN CỨU CỬ NHÂN

Chuyên ngành: TOÁN TIN

HÀ NỘI, 01/2025

ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN - TIN

XÂY DỰNG TRỢ LÝ ẢO HỖ TRỢ TƯ VẤN TUYỂN SINH ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN NGHIÊN CỨU CỬ NHÂN

Chuyên ngành: TOÁN TIN

Giảng viên hướng dẫn: PGS.TS. Lê Chí Ngọc
Sinh viên thực hiện: Vũ Thị Hằng
Mã sinh viên: 20206110

HÀ NỘI, 01/2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

(a) Mục tiêu: .....
.....
.....
.....

(b) Nội dung: .....
.....
.....
.....

2. Kết quả đạt được: .....
.....
.....
.....
.....
.....

3. Ý thức làm việc của sinh viên: .....
.....
.....
.....
.....
.....

Hà Nội, ngày tháng năm 2025

Giảng viên hướng dẫn

PGS.TS Lê Chí Ngọc

Lời cảm ơn

"Đường là do mình chọn, khó đến đâu cũng phải đi cho hết."

Câu nói ấy đã sống cùng với tôi suốt hơn 4 năm đại học qua. Tôi không nhớ đã rơi nước mắt bao nhiêu lần trên giấy, biết bao lần tôi đã muốn dừng lại nhưng cuối cùng tôi cũng đã đi đến chặng cuối cùng của chuyến hành trình này. Suốt hơn 4 năm đó, con xin cảm ơn bố mẹ luôn ở bên con, động viên và an ủi con giúp con có thể mạnh mẽ hơn để vượt qua biết bao khó khăn và gian nan, là điểm tự vững chắc cho con mỗi khi con gục ngã. Em xin cảm ơn PGS.TS Lê Chí Ngọc, người thầy luôn tạo động lực và chỉ dạy em từng tí một, người đã giúp em thay đổi tích cực hơn rất nhiều để hoàn thành đồ án tốt nghiệp này. Em xin cảm ơn các thầy cô Bách Khoa đã mang đến cho em thật nhiều kiến thức hay và bổ ích. Cảm ơn Bách Khoa đã cho tôi những người bạn tốt như bạn Văn, bạn Linh, chúng tôi luôn ở bên nhau đến tận bây giờ. Đặc biệt, cảm ơn chàng trai Bách Khoa CNTT Việt-Nhật đã đến bên đời tôi, giúp đỡ và đồng hành cùng tôi qua biết bao sóng gió của cuộc đời, giúp tôi nhận ra được chỉ cần dồng hành cùng nhau thì mọi khó khăn đều có thể vượt qua được.

Lời cuối cùng, tôi xin cảm ơn chính bản thân mình. Cảm ơn vì đã không bỏ cuộc, cảm ơn vì đã cố gắng mỗi ngày để trở thành phiên bản tốt hơn của chính mình. Cũng đã dài, tôi đã kịp viết ra những kỉ niệm, lời cảm ơn với những người tôi yêu thương nhất. Khoảng khắc cuối cùng, hi vọng đồ án này, tôi đã làm bằng cả tâm huyết của mình trong học kỳ vừa rồi sẽ nhận được những đánh giá tốt từ thầy cô. Chúc bản thân có những thành công mới trong một chặng đường mới. Cảm ơn Bách Khoa vì tất cả.

Hà Nội, ngày 4 tháng 1 năm 2025

Sinh viên thực hiện

Vũ Thị Hằng

Lời cam đoan

Em là Vũ Thị Hằng, sinh viên lớp Toán Tin 01 K65, khoa Toán Tin thuộc Khoa Toán Tin. Em xin cam đoan Đồ án "Xây dựng trợ lý ảo hỗ trợ tư vấn tuyển sinh Đại học Bách Khoa Hà Nội" là công trình nghiên cứu, xây dựng của riêng mình dưới sự hướng dẫn của PGS.TS. Lê Chí Ngọc Các nội dung nghiên cứu, kết quả trong đồ án là xác thực.

Các thông tin sử dụng trong đồ án là có cơ sở và không có nội dung nào sao chép từ các tài liệu mà không ghi rõ trích dẫn tham khảo. Em xin chịu trách nhiệm về lời cam đoan này.

Hà Nội, ngày 4 tháng 1 năm 2025

Sinh viên thực hiện

Vũ Thị Hằng

Tóm tắt nội dung đồ án

Mục tiêu của đồ án là xây dựng một trợ lý ảo hỗ trợ tư vấn tuyển sinh cho Đại học Bách Khoa Hà Nội, nhằm cung cấp những thông tin chính xác, nhanh chóng, tiện lợi cho thí sinh cũng như phụ huynh. Hệ thống được thiết kế dựa trên việc áp dụng các phương pháp xử lý ngôn ngữ tự nhiên (NLP) và học sâu (Deep Learning) để cung cấp câu trả lời chính xác và tối ưu hoá khả năng xử lý câu hỏi cho chatbot.

Đồ án tập trung vào việc nghiên cứu và so sánh độ chính xác khi trả lời của các mô hình: ChatGPT 4o với bộ dữ liệu tuyển sinh, Mô hình RAG (Retrieval-Augmented Generation) + Pre-trained model, Mô hình RAG + Fine-tuning model. Dữ liệu huấn luyện và đánh giá bao gồm: Bộ dữ liệu cho quá trình RAG, Bộ dữ liệu Q&A (chứa context) cho quá trình fine-tune mô hình. Độ chính xác được đánh giá dựa trên 3 chỉ số: Precision, Recall và F1.

Kết quả thực nghiệm cho thấy phương pháp RAG kết hợp với fine-tuning mô hình đạt hiệu suất cao nhất nhờ sự kết hợp giữa khả năng truy xuất và học sâu. Đóng góp chính của đồ án này là xây dựng được các bộ dữ liệu phù hợp với các bài toán cụ thể (nguồn dữ liệu: Trang tuyển sinh Đại học Bách Khoa Hà Nội) và các phương pháp huấn luyện có thể mở rộng và ứng dụng cho các bài toán tương tự trong các lĩnh vực khác.

Hà Nội, ngày 4 tháng 1 năm 2025

Sinh viên thực hiện

Vũ Thị Hằng

Mục lục

Lời cảm ơn **i**

Lời cam đoan **ii**

Tóm tắt nội dung đồ án **iii**

Mục lục **v**

Danh mục ký hiệu và chữ viết tắt **vi**

Danh sách hình vẽ **vii**

Danh sách bảng biểu **viii**

1 Giới thiệu chung **6**

1.1 Phương pháp học sâu 6

1.1.1 Giới thiệu về học sâu (Deep Learning) 6

1.2 Xử lý ngôn ngữ tự nhiên và LLMs 7

1.2.1 Tổng quan 7

1.2.2 Cơ chế Attention cho mô hình Seq2seq 10

1.2.3 Mô hình Transformers 12

1.2.4 Mô hình ngôn ngữ lớn LLM 19

1.3 Hệ thống chatbot 20

1.3.1 Tổng quan 20

1.3.2 So sánh AI chatbot và chatbot truyền thống 20

1.3.3 Quy trình xây dựng AI chatbot 23

2 Cơ sở lý thuyết **25**

2.1 Xây dựng kiến trúc của LLM chatbot 25

2.2 Bài toán xác định ý của người dùng 27

2.3 Bài toán trích xuất thông tin 28
2.3.1 Phương pháp Retrieval-Augmented Generation (RAG) . . 28
2.3.2 Lưu trữ dữ liệu 29
2.3.3 Tìm kiếm thông tin 33
2.4 Bài toán sinh dữ liệu 34
2.4.1 Mô hình Pre-trained 34
2.4.2 Mô hình Fine-tuning 34
2.5 Phương pháp đánh giá mô hình 38

3 Dữ liệu và thực nghiệm 40
3.1 Dữ liệu 40
3.1.1 Giới thiệu 40
3.1.2 Nguồn dữ liệu 42
3.1.3 Quá trình 43
3.1.4 Thống kê và đánh giá tập dữ liệu 44
3.2 Cài đặt thử nghiệm 45
3.2.1 Cài đặt RAG sử dụng ChromaDB và OpenAPI 45
3.2.2 Kết hợp RAG và Pre-train Model 47
3.2.3 Kết hợp RAG và Fine-tuning Model 48
3.2.4 Xây dựng giao diện cho chatbot 49
3.3 Đánh giá thực nghiệm 50

Kết luận 52

Danh mục tài liệu tham khảo 55

Phụ lục 58
A Lưu ý về mã nguồn chương trình và dữ liệu 58
B Quy cách đóng quyển đồ án tốt nghiệp 58

Danh mục ký hiệu và chữ viết tắt

Ký hiệu	**Ý nghĩa tiếng Anh**	**Ý nghĩa tiếng Việt**
AI	Artificial Intellegence	Trí tuệ nhân tạo
NLP	Natural language processing	Xử lý ngôn ngữ tự nhiên
LLM	Large Language Model	Mô hình ngôn ngữ lớn
API	Application Programming Interface	Giao diện chương trình ứng dụng
RAG	Retrieval Augmented Generation	Tạo sinh tăng cường bằng truy xuất
NMT	Neural Machine Translation	Dịch máy bằng mạng nơ-ron
Q&A	Question and Answer	Câu hỏi và câu trả lời
MHSA	Multi-Head Self-Attention	Cơ chế tự chú ý đa đầu
FFN	Feed-Forward Neural Network	Mạng nơ-ron truyền thẳng
LCS(X,Y)	Longest Common Subsequence	Chuỗi con chung dài nhất giữa hai chuỗi X và Y
$\|X\|$	Length of X string	Độ dài của chuỗi X
$\|Y\|$	length of Y string	Độ dài của chuỗi Y
GPT	Generative Pre-trained Transformer	Mạng nơ-ron biến đổi
BERT	Bidirectional Encoder Representation from Transformer	Mô hình biểu diễn từ theo 2 chiều ứng dụng kỹ thuật Transformer
DPR	Dense Passage Retrieval	Truy xuất Đoạn Văn Bản Dày Đặc
BART	Bidirectional and Auto-Regressive Transformers	Biến Mẫu Transformer Hai Chiều và Tự Hồi Quy
R2	R-squared	Hệ số xác định
KV	Key-Value	Khóa-Giá trị
D1	Dataset 1	Một tập dữ liệu cụ thể hoặc một phân đoạn dữ liệu trong một quá trình đánh giá mô hình.
LLaMA	Large Language Model Meta AI	Mô hình Ngôn ngữ Lớn của Meta AI
LoRA	Low-Rank Adaptation	Điều chỉnh Hạng Thấp
QLoRA	Quantized Low-Rank Adaptation	Điều chỉnh Hạng Thấp Lượng Tử Hóa

Danh sách hình vẽ

0.1 Kiến trúc của hệ thống Contact Center 3
0.2 Kiến trúc của hệ thống tư vấn tuyển sinh Đại học Bách Khoa Hà Nội 4

1.1 Ví dụ về dịch máy 9
1.2 Mô hình RNN và LSTM 10
1.3 Mô tả cơ chế chú ý[21] 11
1.4 Kiến trúc mô hình Transformers[6] 17
1.5 Khối Decoder-only transformers 18
1.6 Các điểm chính của mô hình ngôn ngữ lớn 19
1.7 Mô phỏng chatbot truyền thống 21

2.1 Quy trình thực hiện chatbot 27
2.2 Kiến trúc cơ bản của RAG 28
2.3 Fixed size chunking 31
2.4 Recursive chunking 31
2.5 Mô phỏng quá trình fine-tuning 34
2.6 3 phương pháp fine-tuning mô hình 35
2.7 Quy trình fine-tuning mô hình LLM 36
2.8 Mô phỏng quá trình hợp trọng số LoRA vào mô hình cơ sở 37
2.9 Các phương pháp tinh chỉnh khác nhau LoRA và QLoRA[22] . . 38

3.1 Trang web thông tin tuyển sinh HUST 42
3.2 Quá trình tạo xử lý tập dữ liệu 43
3.3 Hình ảnh mô tả Chroma 46
3.4 Cấu hình cho chatbot sử dụng mô hình GPT4o 47
3.5 Phần trả lời câu hỏi của chatbot 48
3.6 Cấu hình LoRA 49
3.7 Giao diện chatbot 50
.8 Quy cách đóng quyển 58

Danh sách bảng

1.1 So sánh các mô hình seq2seq, seq2seq với attention và transformer 13
1.2 So sánh giữa AI Chatbot và Chatbot truyền thống 23

3.1 Số lượng file DOCX và PDF trong các folder 44
3.2 Số lượng file DOCX và PDF trong các folder 45
3.3 Độ chính xác trích xuất top 1 theo từng miền 50
3.4 So sánh điểm Precision, Recall và F1 của các mô hình khác nhau 51
3.5 Độ chính xác của số câu trả lời đúng 51

Mở đầu

Lý do chọn đề tài

Một trong những mục tiêu quan trọng của các nghiên cứu trong lĩnh vực tương tác giữa con người và máy tính là thiết kế của tự nhiên và phương thức tương tác trực quan. Đặc biệt, nhiều nỗ lực đã được cống hiến cho sự phát triển của các hệ thống để tương tác với người dùng bằng ngôn ngữ tự nhiên. Đối với sự phát triển nhanh chóng của thời kỳ công nghiệp hoá, hiện đại hóa. Internet đang dần trở nên gần gũi và không thể thiếu trong cuộc sống hiện nay. Chatbot được ra đời dựa trên sự kết hợp với Trí Tuệ Nhân Tạo (Artificial Intellegence - AI) và có thể biểu diễn dưới dạng hình ảnh, âm thanh và giọng nói để tương tác với con người. Chatbot đã trở nên phổ biến hơn trong những năm gần đây vì chúng giúp giảm chi phí dịch vụ, xử lý nhiều người dùng cùng một lúc [1]. Chatbot sử dụng công nghệ xử lý ngôn ngữ tự nhiên (NLP) để hiểu người dùng và nhờ đó, có thể trả lời các câu hỏi phức tạp với chiều sâu và độ chính xác cao, nhưng nhiều hệ thống hiện nay áp dụng kỹ thuật quét từ khóa từ đầu vào, sau đó phản hồi bằng cách chọn ra từ khóa phù hợp nhất hoặc mẫu ngôn ngữ gần gũi nhất từ cơ sở dữ liệu.

Vài thập kỷ qua đã chứng kiến sự gia tăng dữ liệu, tạo thành nền tảng cho công nghệ AI dựa trên học tập thông qua dữ liệu lớn. Các tác nhân đàm thoại, thường được gọi là chatbot AI, dựa rất nhiều vào dữ liệu đó để đào tạo các mô hình ngôn ngữ lớn (LLM) và tạo nội dung mới (kiến thức) để phản hồi lời nhắc của người dùng [2]. Với sự ra đời ChatGPT của OpenAI, các chatbot dựa trên LLMs đã thiết lập các tiêu chuẩn mới trong cộng đồng AI. Trong năm 2024, có rất nhiều chatbot sử dụng công nghệ LLMs được ra đời với các ứng dụng khác nhau (trò chuyện, tư vấn, tạo sinh hình ảnh, tạo sinh âm thanh,...) ví dụ như ChatGPT, Gemini, Midjourney, Prezi,... Tuy nhiên, các công cụ AI này đều mang một mục đích là trở thành một trợ lý giúp giảm thiểu chi phí hệ thống, chăm sóc sức khỏe và đời sống con người.

Hiện nay, giáo dục đang ngày một thay đổi và công nghệ hóa hơn. Các ứng dụng AI đang dần được đưa vào các trường học để giúp tối ưu hóa nhiều tác vụ đặc thù. Trong các kì tư vấn tuyển sinh, các học sinh có thể tìm hiểu thông tin thông qua internet hoặc qua các chương trình hỗ trợ học sinh định hướng nguyện vọng đại học. Các chatbot hiện nay cũng có thể giúp tư vấn tuyển sinh cho các học sinh cần tìm hiểu. Tuy nhiên, những dữ liệu đặc thù mà các chatbot hiện nay được học còn hạn chế, sinh viên phải mất thời gian chọn lọc và xác minh thông tin. Cùng với đó các hệ thống tư vấn trực tiếp nhiều khi quá tải dẫn đến chậm trễ thông tin tới các em học sinh, các ngày hội định hướng chỉ diễn ra vào thời gian nhất định cũng không thể đáp ứng nhu cầu trong mọi thời điểm.

Với mong muốn tạo ra một hệ thống tư vấn tuyển sinh cho Đại học Bách Khoa Hà Nội nhằm nâng cao trải nghiệm của phụ huynh, học sinh khi sử dụng sản phẩm, tôi đề xuất đề tài:" Xây dựng trợ lý ảo tư vấn tuyển sinh Đại học Bách Khoa Hà Nội".

Mục tiêu đề tài

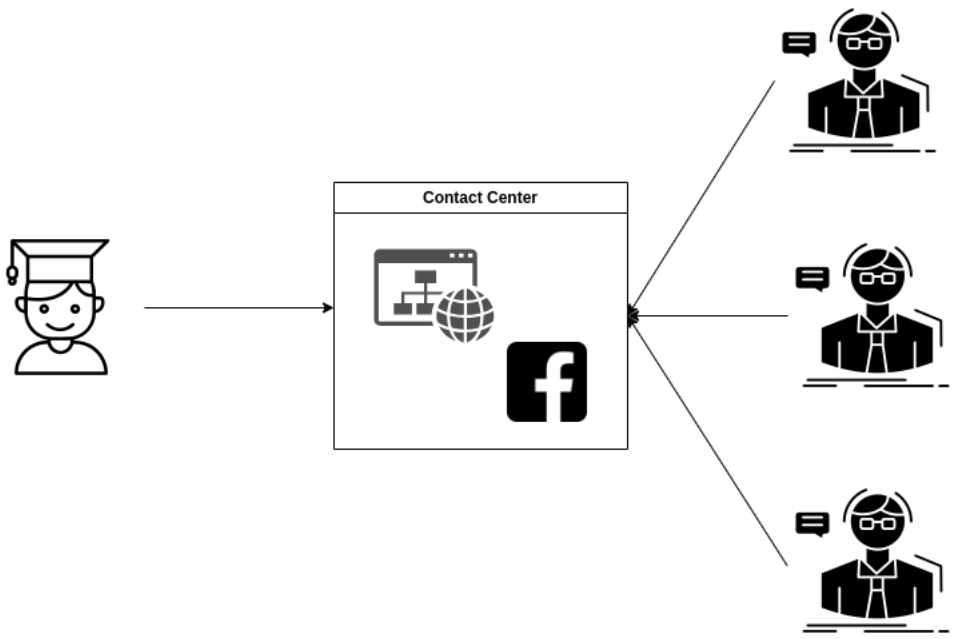
Trợ lý ảo hỗ trợ tư vấn tuyển sinh của Đại học Bách Khoa Hà Nội này nhằm cung cấp một chatbot với khả năng tư vấn những thông tin về tuyển sinh cho các học sinh và phụ huynh với dữ liệu tuyển sinh của trường. Chatbot ở đồ án này là một chương trình máy tính xử lý ngôn ngữ đầu tự nhiên từ người dùng, dựa vào thông tin truy xuất từ vector database để cung cấp cho người dùng những phản hồi thông minh và chính xác. Chatbot được xây dựng dựa trên các công cụ truy xuất và các công cụ thông minh nhằm tương tác với người dùng thông qua giao diện dựa trên văn bản là chủ yếu. Đây là chương trình độc lập, chạy ở máy chủ có thể được tích hợp lên bất kì nền tảng nhắn tin đã mở cho các nhà phát triển thông qua các API như Messenger, Telegram,...

Chatbot tuyển sinh này xây dựng một nền tảng chatbot hoàn chỉnh với 2 chức năng chính: cập nhật thủ công dữ liệu vào database và đưa ra tư vấn, giải đáp thắc mắc cho người dùng hệ thống. Nó có thể tư vấn về thông tin tuyển sinh, ngành tuyển sinh, quy chế tuyển sinh, điểm chuẩn các năm,...

Thực tế hiện nay, do quá nhiều kênh truyền thông chính thống và không chính thống, đồng thời lượng thông tin ngày một nhiều do các phương thức xét tuyển ngày một đổi mới dẫn đến gây rối loạn thông tin cho phụ huynh, học sinh. Cùng với đó, nhiều nguồn thông tin gây hoang mang dư luận và không được kiểm chứng bởi trường đại học dẫn đến ảnh hưởng đến việc xếp nguyện vọng và tư vấn chọn ngành phù hợp với các em học sinh. Ngoài ra, việc tư vấn kịp thời,

ngay lập tức rất khó khăn đối với các trường đại học hiện nay dẫn đến giảm chất lượng của dịch vụ chăm sóc khách hàng và tư vấn định hướng. Thậm chí nhiều câu hỏi không nhận được phản hồi từ văn phòng tuyển sinh.

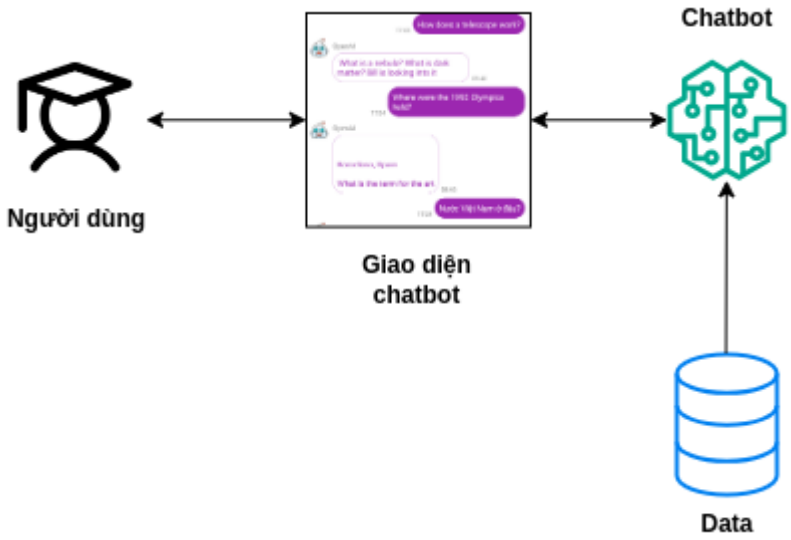
Nhu cầu ngày một tăng cao dẫn đến cần một hệ thống tự động hóa cập nhật thông tin và tư vấn tuyển sinh cho phụ huynh, học sinh. Thực trạng hiện nay ở các trường Đại học hay cụ thể là Đại học Bách Khoa Hà Nội là sử dụng các phần mềm, website hỗ trợ chat và gọi điện giữa các thầy cô của văn phòng tuyển sinh và phụ huynh, học sinh cần tư vấn được gọi là hệ thống Contact Center. Hệ thống này được vận hành bởi các tổng đài viên là các thầy cô phòng tuyển sinh, người sẽ trực tiếp giải đáp và trả lời những câu hỏi qua hình thức gọi điện trực tiếp hay trả lời bình luận ở website, mạng xã hội,... Dưới đây là kiến trúc của hệ thống Contact Center này:



Hình 0.1: Kiến trúc của hệ thống Contact Center

Vấn đề thứ nhất là dữ liệu ngày càng lớn, lượng tư vấn viên tại văn phòng tuyển sinh thì hữu hạn, ngoài ra, nhu cầu cần được tư vấn ngay tại thời điểm của khách hàng là một nhu cầu quan trọng, chưa kể đến những thời điểm sát với thời điểm điều chỉnh nguyện vọng thì nhu cầu ngày càng gia tăng dẫn đến không xử lý được kịp thời do các thầy cô chỉ làm việc giờ hành chính. Vấn đề thứ hai dữ liệu sửa đổi có thể điều chỉnh hàng ngày dẫn đến nhiều thí sinh bị trễ thông tin không nắm kịp, nhiều trang thông tin đưa tin không chính thống gây hoang mang dư luận. Như vậy, giải pháp thay thế các thầy cô tuyển sinh hiện nay bằng chatbot mang lại nhiều hiệu quả thiết thực. Bởi chatbot có thể

giải đáp thắc mắc ngay lập tức đảm bảo độ chính xác, tự nhiên, thông tin tin cậy. Dưới đây là kiến trúc tổng quan hệ thống:



Hình 0.2: Kiến trúc của hệ thống tư vấn tuyển sinh Đại học Bách Khoa Hà Nội

Phạm vi và cấu trúc đề tài

Mục tiêu và nhiệm vụ

Ứng dụng kỹ thuật RAG và Fine-tuning model vào xây dựng chatbot tư vấn tuyển sinh, so sánh kết quả khi sử dụng RAG kết hợp với mô hình pre-trained, mô hình fine-tuning với khi sử dụng mô hình RNN truyền thống. Như vậy, nhiệm vụ của đồ án bao gồm: tìm hiểu lý thuyết về học sâu và lý thuyết về NLP, LLM, tìm hiểu về fine-tuning, pre-trained model và cuối cùng là kĩ thuật RAG.

Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đồ án này là ứng dụng học sâu để xử lý ngôn ngữ tự nhiên trong chatbot. Đồ án sẽ giới thiệu kĩ thuật RAG, Fine-tuning model và các kỹ thuật xây dựng chatbot. Bộ dữ liệu thông tin về tuyển sinh được lấy từ website của nhà trường, bộ dữ liệu Q&A để fine-tuning do nhóm sinh viên chúng em thực hiện đã được chuẩn bị.

Phạm vi nghiên cứu là xây dựng chatbot tuyển sinh có thể thay thế các thầy cô trong việc tư vấn và định hướng người dùng sử dụng hệ thống.

Ý nghĩa khoa học và thực tiễn

Đồ án này đóng góp vào việc nghiên cứu và phát triển các mô hình ngôn ngữ lớn (LLM) trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đặc biệt là trong ứng dụng tư vấn và tương tác với người dùng và đưa ra được cách huấn luyện đạt hiệu quả tốt nhất với bộ dữ liệu tuyển sinh. Thông qua việc áp dụng kỹ thuật RAG (Retrieval-Augmented Generation) và fine-tuning, đồ án cung cấp một góc nhìn mới về cách kết hợp giữa tìm kiếm thông tin và sinh ngôn ngữ tự nhiên nhằm nâng cao độ chính xác và tính hiệu quả của chatbot. Nghiên cứu cũng đưa ra sản phẩm là một giao diện chatbot để thử nghiệm các mô hình sau khi xử lý với bộ dữ liệu tuyển sinh.

Đồ án này có tiềm năng ứng dụng thực tiễn cao khi xây dựng một hệ thống chatbot tư vấn tuyển sinh tự động cho Đại học Bách Khoa Hà Nội. Chatbot sẽ giúp giảm tải công việc cho các thầy cô, cung cấp thông tin kịp thời, chính xác cho thí sinh và người quan tâm đến tuyển sinh, từ đó nâng cao trải nghiệm người dùng. Hệ thống này không chỉ tiết kiệm nguồn lực mà còn tăng cường khả năng tiếp cận thông tin nhanh chóng, hỗ trợ đắc lực trong quá trình tư vấn và định hướng cho thí sinh.

Bố cục luận văn

- **Phần 1**: Giới thiệu chung
- **Phần 2**: Cơ sở lý thuyết
- **Phần 3**: Dữ liệu và thực nghiệm

Chương 1

Giới thiệu chung

1.1 Phương pháp học sâu

1.1.1 Giới thiệu về học sâu (Deep Learning)

Học sâu là một dạng học máy cho phép máy tính học hỏi từ kinh nghiệm và hiểu thế giới theo thứ bậc các khái niệm [4]. Các phương pháp này đã cải thiện đáng kể trình độ tiên tiến trong nhận dạng giọng nói, nhận dạng đối tượng trực quan, phát hiện đối tượng và nhiều lĩnh vực khác như khám phá thuốc và hệ gen. Học sâu khám phá cấu trúc phức tạp trong các tập dữ liệu lớn bằng cách sử dụng thuật toán lan truyền ngược để chỉ ra cách máy nên thay đổi các tham số bên trong được sử dụng để tính toán biểu diễn trong mỗi lớp từ biểu diễn trong lớp trước đó. Các mạng tích chập sâu đã mang lại những đột phá trong xử lý hình ảnh, video, giọng nói và âm thanh, trong khi các mạng hồi quy đã làm sáng tỏ dữ liệu tuần tự như văn bản và giọng nói. Công nghệ học máy hỗ trợ nhiều khía cạnh của xã hội hiện đại: từ tìm kiếm trên web đến lọc nội dung trên mạng xã hội đến các đề xuất trên các trang web thương mại điện tử và ngày càng hiện diện trong các sản phẩm tiêu dùng như máy ảnh và điện thoại thông minh. Các hệ thống học máy được sử dụng để xác định các đối tượng trong hình ảnh, chuyển lời nói thành văn bản, khớp các mục tin tức, bài đăng hoặc sản phẩm với sở thích của người dùng và chọn kết quả tìm kiếm có liên quan [3].

Học biểu diễn là tập hợp các phương pháp cho phép máy nhận dữ liệu thô và tự động tìm ra các biểu diễn cần thiết để phân loại hoặc phát hiện. Các phương pháp học sâu chính là học biểu diễn nhiều tầng, trong đó dữ liệu được biến đổi

qua nhiều lớp không tuyến tính. Mỗi lớp sẽ chuyển đổi biểu diễn từ mức cơ bản (từ đầu vào thô) lên các mức cao hơn, trừu tượng hơn. Nhờ chuỗi các phép biến đổi này, máy có thể học được các hàm rất phức tạp. Trong các bài toán phân loại, các biểu diễn ở mức cao sẽ nhấn mạnh những đặc trưng quan trọng của đầu vào để dễ dàng phân biệt, đồng thời loại bỏ các yếu tố không liên quan.

Mạng nơ-ron trong học sâu đòi hỏi khối lượng dữ liệu đào tạo lớn để học qua quá trình mắc lỗi và sửa sai, điều này chỉ khả thi khi các doanh nghiệp bắt đầu thu thập và phân tích dữ liệu lớn. Dữ liệu đào tạo cần được gán nhãn để mô hình có thể xác định chính xác dự đoán của mình, tuy nhiên, dữ liệu phi cấu trúc ít hữu ích và chỉ có thể được kiểm tra sau khi mô hình đã đạt độ chính xác cao. Học sâu cũng gặp nhiều thách thức khác, như chỉ học từ dữ liệu quan sát được, do đó khả năng khái quát hóa bị giới hạn nếu dữ liệu đào tạo không đầy đủ hoặc không đại diện. Hơn nữa, thiên vị trong dữ liệu là một vấn đề quan trọng, vì mô hình sẽ lặp lại những thiên kiến này trong dự đoán nếu không được xử lý đúng cách. Tốc độ học cũng ảnh hưởng lớn đến kết quả, với tốc độ quá cao dễ gây hội tụ không mong muốn, trong khi tốc độ quá thấp khiến mô hình gặp khó khăn khi tìm giải pháp.

Về mặt phần cứng, học sâu yêu cầu GPU hiệu suất cao, RAM lớn và bộ nhớ SSD, những thiết bị này đắt đỏ và tiêu tốn nhiều năng lượng. Ngoài ra, mô hình học sâu đòi hỏi lượng dữ liệu khổng lồ và nhiều tham số để đạt độ chính xác cao nhưng lại thiếu linh hoạt và chỉ giải quyết được một vấn đề cụ thể sau khi đào tạo. Dù có thể phân tích dữ liệu phức tạp, học sâu hiện vẫn chưa xử lý được các tác vụ đòi hỏi tư duy, khả năng lập kế hoạch dài hạn hay thao tác dữ liệu theo thuật toán.

1.2 Xử lý ngôn ngữ tự nhiên và LLMs

1.2.1 Tổng quan

Xử lý ngôn ngữ tự nhiên (NLP) là một lĩnh vực của khoa học máy tính và trí tuệ nhân tạo (AI) sử dụng học máy để cho phép máy tính hiểu và giao tiếp với ngôn ngữ của con người[14]. NLP cho phép máy tính và các thiết bị kỹ thuật số nhận dạng, hiểu và tạo ra văn bản và giọng nói bằng cách kết hợp ngôn ngữ học tính toán - mô hình hóa ngôn ngữ con người dựa trên quy tắc - cùng với mô hình thống kê, học máy và học sâu . Nghiên cứu NLP đã giúp tạo ra kỷ nguyên AI tạo sinh , từ các kỹ năng giao tiếp của các mô hình ngôn ngữ lớn (LLM) đến khả năng của các mô hình tạo hình ảnh để hiểu các yêu cầu. NLP đã trở thành một phần trong cuộc sống hàng ngày của nhiều người, cung cấp năng lượng cho

các công cụ tìm kiếm, nhắc nhở các chatbot để phục vụ khách hàng bằng các lệnh nói, hệ thống GPS điều khiển bằng giọng nói và trợ lý kỹ thuật số trả lời câu hỏi trên điện thoại thông minh[15] như Alexa của Amazon, Siri của Apple và Cortana của Microsoft. NLP cũng đóng vai trò ngày càng quan trọng trong các giải pháp doanh nghiệp giúp hợp lý hóa và tự động hóa các hoạt động kinh doanh, tăng năng suất của nhân viên và đơn giản hóa các quy trình kinh doanh.

Bên cạnh đó, NLP giúp con người dễ dàng giao tiếp và cộng tác với máy móc hơn bằng cách cho phép họ làm như vậy bằng ngôn ngữ tự nhiên của con người mà họ sử dụng hàng ngày. NLP bao gồm rất nhiều các bài toán, dưới đây là một trong số các bài toán cơ bản của NLP:

- Mô hình hóa ngôn ngữ (Language modelling)
- Phân loại văn bản (Text classification)
- Trích xuất thông tin (Information extraction)
- Truy xuất thông tin (Information retrieval)
- Tác tử phần mềm hội thoại (Conversational agent)
- Tóm tắt văn bản (Text summarization)
- Hỏi đáp (Question answering)
- Dịch máy (Machine translation)
- Mô hình hóa chủ đề (Topic modelling)

Xử lý ngôn ngữ tự nhiên (NLP) là một cách tiếp cận hiệu quả để mang lại sự cải thiện trong môi trường giáo dục[16]. Việc triển khai NLP liên quan đến việc bắt đầu quá trình học tập thông qua việc tiếp thu tự nhiên trong hệ thống giáo dục. Nó dựa trên các phương pháp tiếp cận hiệu quả để cung cấp giải pháp cho các vấn đề và vấn đề khác nhau trong giáo dục. Xử lý ngôn ngữ tự nhiên cung cấp giải pháp trong nhiều lĩnh vực khác nhau liên quan đến bối cảnh văn hóa và xã hội của việc học ngôn ngữ. Đây là một cách tiếp cận hiệu quả dành cho giáo viên, sinh viên, tác giả và nhà giáo dục trong việc hỗ trợ các thủ tục viết, phân tích và đánh giá. Xử lý ngôn ngữ tự nhiên được tích hợp rộng rãi với số lượng lớn bối cảnh giáo dục như nghiên cứu, khoa học, ngôn ngữ học, học tập điện tử, hệ thống đánh giá và góp phần mang lại kết quả tích cực trong các môi trường giáo dục khác như trường học, hệ thống giáo dục đại học và đại học.

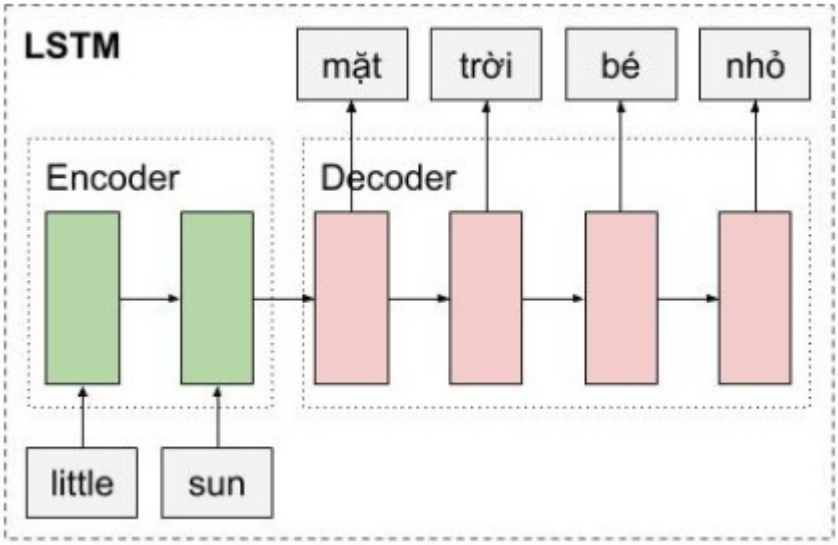
Vậy để có những ứng dụng như vậy, chúng ta làm thế nào để AI có thể hiểu được ngôn ngữ của chúng ta và có thể tìm kiếm và đưa ra dữ liệu chuẩn xác, tự nhiên nhất đến với con người. Thông qua các phần tiếp theo, chúng ta sẽ tìm

hiểu nó.

Dịch máy (Machine Translation)

Dịch máy là nhiệm vụ dịch một đoạn văn bản từ ngôn ngữ nguồn(source language) sang một ngôn ngữ đích(target language).

Ví dụ: Input: little sun, Output: Mặt trời bé con



Hình 1.1: Ví dụ về dịch máy

Dịch máy nơ-ron (NMT) là một phương pháp dịch máy sử dụng mạng nơ-ron nhân tạo để dự đoán khả năng xảy ra của một chuỗi từ, thường mô hình hóa toàn bộ câu trong một mô hình tích hợp duy nhất. Dạng cơ bản của NMT bao gồm 2 thành phần:

- Encoder: Bộ mã hóa lấy câu nguồn làm đầu vào của mô hình và encode nó thành một context-vector.
- Decoder: Bộ giải mã sử dụng context-vector của encoder làm khởi tạo cho hidden-state đầu tiên.

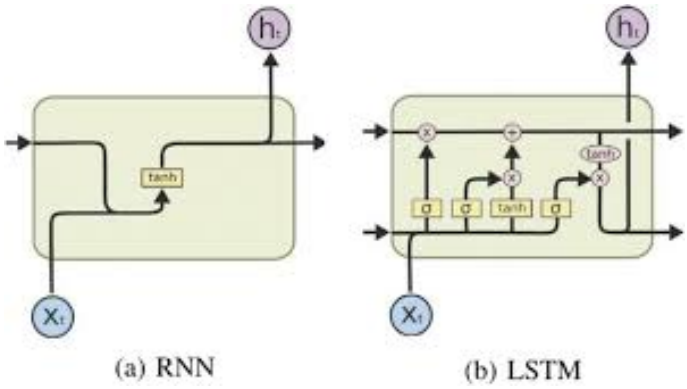
Các bài toán Seq2seq thường sử dụng kiến trúc mạng RNN, LSTM, GRU,... Với cách đào tạo tuần tự này, chúng ta sẽ không tận dụng được sức mạnh của GPU, không thể tính toán song song dẫn đến đào tạo mô hình bị chậm.

Thêm vào đó, với dữ liệu tiếng việt gặp một số khó khăn nhất định. Trong tiếng việt, với hiện tượng đồng âm khác nghĩa, dấu thanh hay với câu dài, phức tạp,...RNN không đáp ứng tốt việc duy trì ngữ cảnh xa dẫn đến việc dịch sai nghĩa của đầu vào. LSTM ra đời khắc phục nhược điểm trên một phần nhỏ do

có cổng thông tin đi qua giúp duy trì các thông tin khi qua các bước , tuy nhiên vẫn gặp các nhược điểm sau:

- Khó huấn luyện, thời gian huấn luyện lâu do gradient path rất dài (chuỗi 100 từ có gradient như là mạng 100 lớp)
- Transfer learning không hoạt động với LSTM, điều đồng nghĩa với một bài toán mới thì ta cần huần luyện lại mô hình với bộ dữ liệu riêng biệt cho nhiệm vụ đề ra (tốn kém)

Quay lại với mô hình Seq2seq với RNN thì encoder sẽ phải "nén" toàn bộ chuỗi thành một context vector duy nhất để lưu trữ thông tin toàn bộ câu đầu vào. Với các câu dài, context vector không thể chứa đủ thông tin, dẫn đến mất mát ngữ nghĩa và giảm chất lượng dịch. Để giải quyết hạn chế của mô hình encoder-



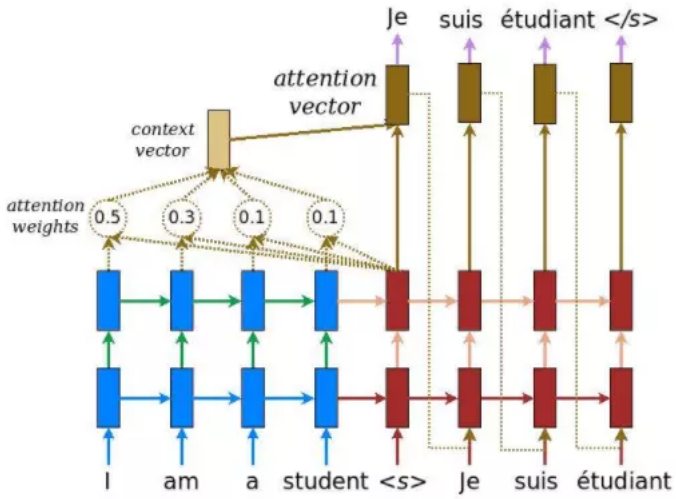
Hình 1.2: Mô hình RNN và LSTM

decoder dựa trên RNN, đặc biệt là việc phải nén toàn bộ thông tin của câu nguồn vào một vector cố định (context vector), với văn bản dài, việc này dẫn đến mất mát thông tin thì cơ chế attention ra đời. Attention cho phép mô hình tập trung vào các phần quan trọng khác nhau của câu nguồn khi tạo ra từng từ trong câu đích, thay vì chỉ dựa vào một vector duy nhất. Phần chi tiết về cơ chế attention được trình bày ở mục 1.2.2.

1.2.2 Cơ chế Attention cho mô hình Seq2seq

1. **Giới thiệu**

   Như vậy cơ chế attention được ra đời để giải quyết các vấn đề của mô hình seq2seq, với ý tưởng sử dụng một vector bối cảnh có thể tương tác với toàn bộ vector trạng thái ẩn của encoder thay vì chỉ sử dụng vector trạng thái ẩn cuối cùng để tạo ra vector biểu diễn cho decoder. Cụ thể hơn, mô hình seq2seq khi áp dụng cơ chế attention vào sẽ có cấu trúc như sau:



Hình 1.3: Mô tả cơ chế chú ý[21]

Trong ảnh trên, các khối màu xanh dương là khối encoder, khối màu đỏ là khối decoder. Miêu tả chi tiết các bước tại mỗi time-step t ở khối decoder như sau:

- **Bước 1**: Nhận vector trạng thái ẩn (Hidden state) của decoder h_t và tất cả các vector trạng thái ẩn (Hidden state) của encoder h_s.
- **Bước 2**: Tính điểm attention bằng cách: Với mỗi vector trạng thái ẩn của encoder ta sẽ tính điểm thể hiện sự liên quan với vector ẩn h_t của decoder. Cụ thể, ta sẽ tính tích vô hướng của h_t và h_s trong đó: h_t là vector trạng thái ẩn decoder, h_s là vector trạng thái ẩn của encoder.
- **Bước 3**: Tính trọng số Attention. Ta áp dụng hàm softmax với đầu vào là các điểm attention đã tính ở bước 2:

$$\alpha_{ts} = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'=1}^{S} \exp(\text{score}(h_t, \bar{h}_{s'}))}$$

- **Bước 4**: Tính toán vector bối cảnh c_t là tổng của các trọng số attention nhân với vector trạng thái ẩn của decoder tại time-step tương ứng.

$$c_t = \sum_{s'=1}^{S} \alpha_{ts} \bar{h}_{s'}$$

Cuối cùng, các vector attention a_t dùng để đưa ra đầu ra được tính dựa

trên vector bối cảnh c_t và vector trạng thái ẩn ở decoder h_t.

Kết luận:

2. **Các loại cơ chế attention**

   Sự ra đời của cơ chế attention đã loại bỏ sự phục thuộc về khoảng cách của chuỗi đầu vào và đầu ra. Với attention thì lĩnh vực dịch máy đã có nhiều cải tiến đáng kể, ngày càng có nhiều các dạng khác nhau của cơ chế attention. Cơ chế attention có thể được chia thành 3 loại [6][22] sau:

   - Self attention
   - Global/Soft attention
   - Local/Hard attention

Sau khi hiểu rõ về cơ chế Attention và cách nó cải thiện hiệu quả của mô hình Seq2seq, chúng ta có thể thấy rằng Attention không chỉ giải quyết được các hạn chế của Seq2seq mà còn mở ra những hướng tiếp cận mới trong việc xử lý dữ liệu tuần tự. Tuy nhiên, để vượt qua các giới hạn về hiệu suất và khả năng mở rộng khi áp dụng trên dữ liệu lớn và phức tạp, các nhà nghiên cứu đã đề xuất mô hình Transformer – một kiến trúc hoàn toàn dựa trên Attention mà không cần đến các cơ chế tuần tự như RNN hay LSTM. Tiếp theo, chúng ta sẽ đi sâu vào mô hình Transformer để hiểu cách nó tận dụng tối đa sức mạnh của Attention và mang lại những đột phá vượt bậc trong các tác vụ học sâu được trình bày ở phần 1.2.3

1.2.3 Mô hình Transformers

1. **Giới thiệu**

   Với sự ra đời của cơ chế Attention, bài báo Attention is All You Need năm 2017[6] đã giới thiệu một kiến trúc hoàn toàn mới - Transformer, đánh dấu bước ngoặt quan trọng trong các bài toán Xử lý Ngôn ngữ Tự nhiên (NLP). Điểm đặc biệt của Transformer là không còn sự phụ thuộc vào các mạng nơ-ron hồi tiếp (RNN, LSTM), mà thay vào đó hoàn toàn dựa trên cơ chế Attention.

   Như đã đề cập ở phần 1.2.1, trong các bài toán Seq2seq, các mô hình dựa trên RNN hoặc LSTM thường gặp hạn chế về khả năng xử lý chuỗi dài và thời gian huấn luyện. Transformer đã khắc phục được các vấn đề này bằng cách loại bỏ tính tuần tự trong quá trình tính toán, giúp tăng tốc độ huấn luyện đáng kể đồng thời cải thiện độ chính xác và hiệu quả của mô hình. Không chỉ dừng lại ở các bài toán Seq2seq, Transformer hiện nay đã được

phát triển thành nhiều biến thể mạnh mẽ, phục vụ đa dạng các ứng dụng như mô hình ngôn ngữ (Language Modeling), xử lý hình ảnh, và các lĩnh vực khác.

Bảng dưới đây sẽ thể hiện sự khác biệt của transformer với các mô hình trước đó, khi transformer hoàn toàn hoạt động dựa trên cơ chế attention

	seq2seq	**seq2seq với attention**	**transformer**
Encoder	RNN/CNN	RNN/CNN	attention
Decoder	RNN/CNN	RNN/CNN	attention
Tương tác giữa encoder và decoder	vector	attention	attention

Bảng 1.1: So sánh các mô hình seq2seq, seq2seq với attention và transformer

Một trong những lý do khiến Transformers mang lại hiệu quả tốt hơn RNN là Transformer không xử lý dữ liệu tuần tự, mà xử lý toàn bộ chuỗi song song. Điều này cho phép Transformer xử lý toàn bộ chuỗi dữ liệu cùng lúc mà không cần tuần tự, giúp giảm độ phức tạp tính toán từ $O(N)$ xuống chỉ còn $O(1)$.

Kết quả: Transformer không chỉ nhanh hơn mà còn có khả năng học được mối liên hệ dài hạn trong câu một cách hiệu quả, điều mà các mô hình RNN truyền thống thường gặp khó khăn. Đây là lý do khiến Transformer trở thành lựa chọn ưu việt trong nhiều bài toán NLP hiện đại.

2. **Cơ chế tự chú ý (Self Attention)**

   Thuật toán scale dot product[6]

   Mục tiêu của việc sử dụng thuật toán này là tạo ra một ma trận attention biểu diễn mối quan hệ của từng thành phần trong một câu với các thành phần khác để xác định nên tập trung vào phần/vùng nào, trong khi vẫn giữ được lợi thế về mặt tính toán.

   Dot product attention nhận đầu vào là 1 tập queries $Q \in \mathbb{R}^{T \times d_k}$, keys $K \in \mathbb{R}^{T \times d_k}$ và values $V \in \mathbb{R}^{T \times d_v}$, trong đó T là độ dài của câu và d_k, d_v là dimension của queries/keys và values. Dot product attention được biểu diễn dưới dạng toán học như sau:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V[6]$$

   Phép nhân ma trận QK^T sẽ thực hiện nhân mỗi cặp query-key với nhau để

tìm mối quan hệ giữa chúng, tạo thành ma trận có dạng $T \times T$. Sau đó ta sẽ áp dụng phép softmax lên ma trận vừa tìm được để chuẩn hóa trọng số attention về khoảng $(0, 1)$, nếu ta nhân ma trận này với V thì những từ cần được "chú ý" sẽ gần như giữ nguyên giá trị, những từ không quan trọng sẽ bị giảm giá trị. Hệ số scaling factor $\frac{1}{\sqrt{d_k}}$, hệ số này đóng vai trò khá quan trọng với nhiệm vụ giữ cho phương sai của phép attention không bị lệch.

Cơ chế tự chú ý (Self Attention)

Tự chú ý[6] là phương pháp mà mô hình transformer sử dụng để minh họa việc "hiểu" các từ có liên quan khác vào từ đang được xử lý.

Ví dụ: "The animal didn't cross the street because it was too tired"

Khi mô hình đang xử lý từ "it", khả năng tự chú ý cho phép mô hình liên kết "it" với "động vật".

Cơ chế self-attention mô tả trọng số đại diện cho độ quan trọng của mỗi thành phần trong input. Cơ chế Self-Attention cũng bao gồm ba thành phần chính là Query (Q), Key (K), và Value (V):

- **Query (Q)**: Là vector đại diện cho thông tin cần "tìm kiếm". Tại mỗi token, Query được dùng để so sánh với Key của tất cả các token khác trong chuỗi đầu vào nhằm xác định mức độ liên quan.
- **Key (K)**: Là vector đại diện cho thông tin "chìa khóa" của từng token, đóng vai trò xác định liệu nó có liên quan đến Query hay không.
- **Value (V)**: Là vector chứa toàn bộ thông tin của từng token. Sau khi tính toán trọng số Attention dựa trên Query và Key, các trọng số này được áp dụng lên Value để tạo ra vector bối cảnh (context vector).

Query được sử dụng khi một token "quan sát" những tokens còn lại, nó sẽ tìm kiếm thông tin xung quanh để hiểu được ngữ cảnh và mối quan hệ của nó với các tokens còn lại. Key sẽ phản hồi yêu cầu của Query và được sử dụng để tính trọng số attention. Cuối cùng, Value được sử dụng trọng số attention vừa rồi để tính ra vector đại diện (attention vector). Trong công thức dưới đây, ma trận W_Q, W_K, W_V chính là các hệ số mà mô hình cần huấn luyện.Cơ chế Self-Attention được tính toán theo thuật toán Scaled

Dot-Product Attention, với công thức[6]:

$$Q = X \times W^Q$$
$$K = X \times W^K$$
$$V = X \times W^V$$

Đầu ra cuối cùng của lớp self-attention được biểu diễn như sau:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V[6]$$

3. **Cơ chế Masked Self Attention**

   Cách hoạt động:

   - Masked Self-Attention hoạt động tương tự như cơ chế Self-Attention đã trình bày, nhưng có thêm một bước masking.
   - Masking là quá trình gán giá trị $-\infty$ hoặc một giá trị rất nhỏ cho các trọng số Attention tương ứng với các tokens ở tương lai, trước khi áp dụng hàm softmax. Điều này làm cho trọng số Attention của các tokens tương lai trở thành 0 sau khi chuẩn hóa.

4. **Cơ chế Multi-Head Attention**

   Thông thường, để hiểu được vai trò của một từ trong câu, mô hình cần đánh giá mức độ liên quan giữa từ đó với các thành phần khác trong câu. Vì vậy, thay vì chỉ sử dụng một cơ chế self-attention (hay còn gọi là một "head") để xác định mối liên hệ giữa các từ, mô hình được thiết kế với nhiều "heads". Mỗi "head" sẽ học cách tập trung vào một khía cạnh cụ thể của mối quan hệ giữa từ và ngữ cảnh, ví dụ như sự liên quan ngắn hạn, dài hạn, hoặc các đặc điểm ngữ nghĩa khác.

   Chính cơ chế này được gọi là Multi-Head Attention[6], và nó giúp mô hình không chỉ tập trung vào một loại thông tin duy nhất mà còn nắm bắt được nhiều mối liên hệ phức tạp hơn, từ đó cải thiện hiệu quả và độ chính xác trong các bài toán xử lý ngôn ngữ tự nhiên.

   Công thức chính của Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O[6]$$

Trong đó:

- $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$ là kết quả Attention ở từng "head".
- W^O là ma trận trọng số học được, dùng để chiếu kết quả cuối cùng về cùng kích thước với đầu vào.

Việc sử dụng nhiều head Attention giúp mô hình học được các mối quan hệ khác nhau giữa các token (ví dụ: quan hệ ngữ nghĩa ngắn hạn, dài hạn) và cung cấp một biểu diễn phong phú hơn so với việc sử dụng một Attention duy nhất. Concat kết quả từ các head và chiếu về kích thước ban đầu giúp đảm bảo tính nhất quán giữa đầu vào và đầu ra.

5. **Positional Encoding**

   Trong mô hình Transformer, do không có mạng hồi tiếp hay mạng tích chập để nhận biết thứ tự của các token đầu vào, positional encoding được sử dụng để cung cấp thông tin về vị trí của các token. Sau bước nhúng từ (embedding layers) ta thu được các tokens tsau đó ta cộng nó với các vector thể hiện vị trí của từ trong câu.

   Chúng ta sẽ sử dụng hàm *sin* hoặc *cosin* với các tần số khác nhau:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)[6]$$

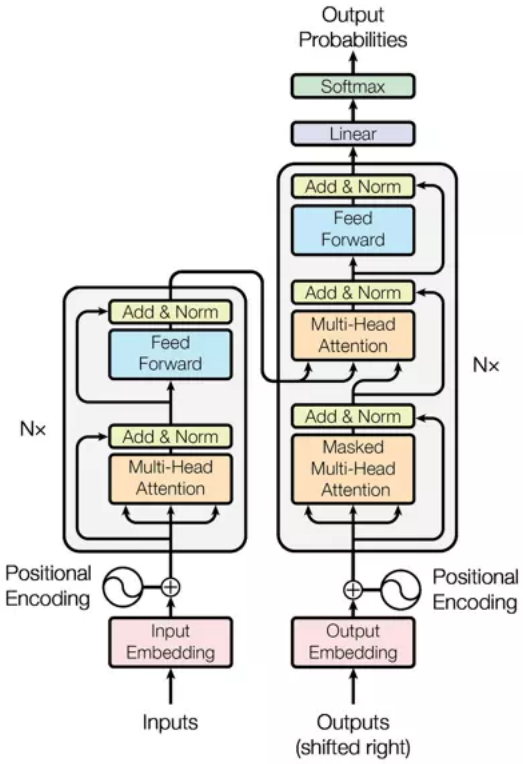
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)[6]$$

   Trong đó: pos là vị trí và i là kích thước.

6. **Kiến trúc mô hình transformers**

   Đây là cấu trúc của transformer được giới thiệu trong bài báo "Attention is all you need". Mô hình cũng gồm 2 thành phần tương tự như các mô hình khác:

   - Encoder: Bộ mã hóa là ánh xạ chuỗi đầu vào thành một chuỗi biểu diễn liên tục, sau đó được đưa vào bộ giải mã.
   - Decoder: Bộ giải mã nhận đầu ra của bộ mã hóa cùng với đầu ra của bộ giải mã ở bước thời gian trước đó để tạo ra chuỗi đầu ra.



Hình 1.4: Kiến trúc mô hình Transformers[6]

(a) **Thành phần mã hóa (Encoder)**

Transformer Encoder bao gồm N khối giống nhau được sắp xếp tuần tự. Mỗi khối Encoder gồm hai thành phần chính:

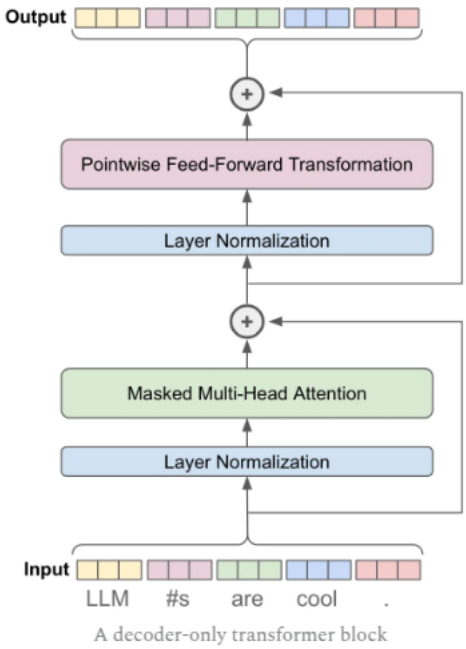
- Multi-Head Self-Attention (MHSA) với cơ chế Add & Norm.
- Feed-Forward Network (FFN) cũng đi kèm Add & Norm.

(b) **Thành phần giải mã (Decoder)**

Kiến trúc của Decoder trong Transformer tương tự với Encoder, bao gồm N khối giống nhau được sắp xếp tuần tự. Tuy nhiên, điểm khác biệt chính là Decoder có thêm một Masked Multi-Head Attention block ở vị trí đầu tiên trong mỗi khối. Thành phần này ngăn chặn việc một token hiện tại "nhìn" thấy các token ở tương lai trong chuỗi đầu ra (output sequence), đảm bảo quá trình sinh đầu ra tuân thủ thứ tự tuần tự.

Hiện nay, các mô hình LLM hầu hết sử dụng kiến trúc Decoder-Only Transformers. Decoder-Only Transformers được chế tạo bằng cách xếp chồng nhiều lớp, mỗi lớp bao gồm hai thành phần chính:

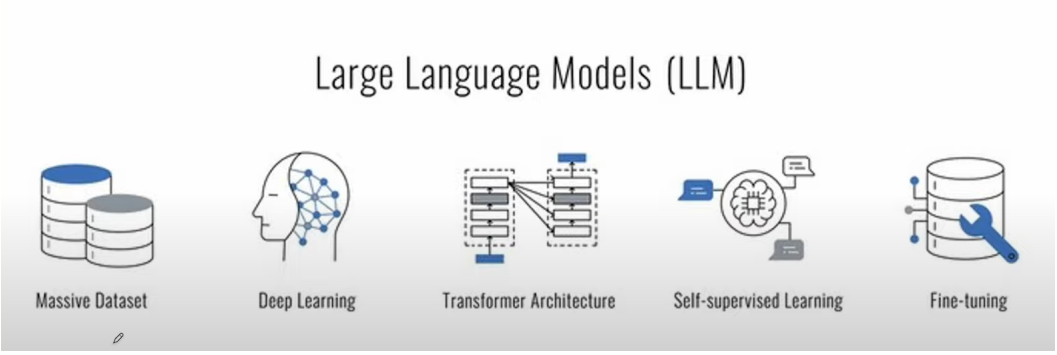
- Tự chú ý bị che giấu: lớp này cho phép các từ kết hợp thông tin theo ngữ cảnh từ các từ trước đó.
- Perceptron nhiều lớp (MLP): MLP là các mạng thần kinh chuyển tiếp nguồn cấp dữ liệu xử lý việc nhúng từng từ một cách độc lập. Chúng có thể được coi là "ngân hàng kiến thức" của máy biến áp, kết hợp kiến thức chung đã học được trong quá trình đào tạo để hoàn thiện hơn nữa các phần nhúng.



Hình 1.5: Khối Decoder-only transformers

1.2.4 Mô hình ngôn ngữ lớn LLM

Large language model (LLM): là một thuật toán học sâu được trang bị để tóm tắt, dịch, dự đoán và tạo ra văn bản giống con người để truyền đạt ý tưởng và khái niệm.



Hình 1.6: Các điểm chính của mô hình ngôn ngữ lớn

Hiện nay, các mô hình LLM hoạt động dựa trên kiến trúc Transformer[23], với khả năng học từ một lượng lớn dữ liệu văn bản để hiểu ngữ nghĩa, ngữ cảnh và tạo ra văn bản tự nhiên. Với các model GPT, LLMs hiện đại chỉ sử dụng phần Decoder để sinh văn bản gọi là Decoder-only. LLMs được huấn luyện trên dữ liệu lớn như sách, bài báo, trang web, nhưng có các bộ lọc để đảm bảo chất lượng và tính đạo đức.

Các mô hình LLM phổ biến

- GPT (Generative Pre-trained Transformer): Sử dụng kiến trúc Decoder-only, ví dụ: GPT-3, GPT-4.
- BERT (Bidirectional Encoder Representations from Transformers): Sử dụng kiến trúc Encoder để hiểu ngữ nghĩa, không tập trung vào sinh văn bản.
- T5 (Text-to-Text Transfer Transformer): Kiến trúc Encoder-Decoder dùng để xử lý các tác vụ text-to-text.
- PaLM, Bloom, LLaMA: Những mô hình ngôn ngữ lớn khác đang được sử dụng phổ biến.

  Mô hình ngôn ngữ lớn (LLM) đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực, mang lại giá trị vượt trội cho các tổ chức và cá nhân. Trong lĩnh vực chăm sóc khách hàng, LLM hỗ trợ xây dựng các chatbot và trợ lý ảo thông minh, giúp trả lời câu hỏi, giải quyết vấn đề nhanh chóng, nâng cao trải nghiệm người dùng. Trong giáo dục, mô hình này có thể tạo nội dung học tập, gợi ý ý tưởng và tóm tắt thông tin, giúp người học tiếp cận

kiến thức hiệu quả hơn. Trong ngành y tế, LLM hỗ trợ phân tích triệu chứng, cung cấp thông tin y khoa, giúp bác sĩ và bệnh nhân đưa ra quyết định chính xác. Bên cạnh đó, mô hình này còn giúp tự động hóa việc phân tích dữ liệu, tạo báo cáo, dịch thuật đa ngôn ngữ và hỗ trợ nghiên cứu khoa học. Với khả năng xử lý ngôn ngữ tự nhiên tiên tiến, LLM đang mở ra những cơ hội mới cho nhiều ngành nghề và cuộc sống hàng ngày.

1.3 Hệ thống chatbot

1.3.1 Tổng quan

Hệ thống chatbot là chương trình hoặc ứng dụng có thể trò chuyện với người dùng thông qua giọng nói hoặc văn bản. Kể từ thời điểm chatbot đầu tiên được phát triển vào những năm 1960, ứng dụng này đã không ngừng đổi mới công nghệ hỗ trợ[15]. Chatbot truyền thống sử dụng quy tắc định sẵn để trò chuyện với người dùng và cung cấp câu trả lời theo kịch bản. Chatbot hiện tại sử dụng công nghệ xử lý ngôn ngữ tự nhiên (NLP) để hiểu người dùng và nhờ đó, có thể trả lời các câu hỏi phức tạp với chiều sâu và độ chính xác cao.

Cách thức hoạt động của chatbot:

- **Translator**: Khi người dùng gửi thông tin hoặc yêu cầu, chatbot sẽ sử dụng một ngôn ngữ lập trình để "dịch" nội dung này thành dạng mà máy tính có thể hiểu. Bước này giúp chatbot xác định rõ các công việc hoặc hành động cần thực hiện dựa trên input từ người dùng.
- **Processor**: Sau khi thông tin đã được chuyển đổi, công nghệ AI sẽ tiến hành phân tích và xử lý yêu cầu của người dùng. AI sử dụng các thuật toán để xác định ý nghĩa của yêu cầu, tìm kiếm thông tin liên quan, và đưa ra phương án đáp ứng thích hợp.
- **Respondent**: Cuối cùng, máy tính nhận kết quả đầu ra từ quá trình xử lý và gửi phản hồi về cho người dùng trên nền tảng nhắn tin (messenger). Thông tin này có thể là câu trả lời, hướng dẫn hoặc bất kỳ nội dung nào mà người dùng cần, giúp hoàn thiện trải nghiệm giao tiếp của họ với chatbot.

1.3.2 So sánh AI chatbot và chatbot truyền thống

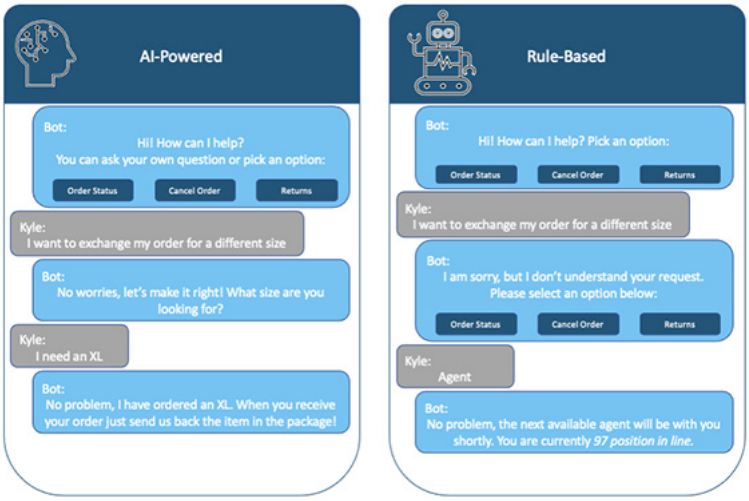
1. **Chatbot truyền thống**

   Chatbot truyền thống được xây dựng dựa trên nguyên tắc và luồng hội thoại được xác định từ trước. Với hầu hết các trường hợp sử dụng chatbot

truyền thống, người dùng sẽ chọn yêu cầu qua các nút bấm với các chức năng tùy chọn có sẵn.

Chatbot nhận diện được ý định người dùng thông qua những từ khóa được lập trình sẵn (keyword-based). Nhược điểm của những chatbot này là câu hỏi của người dùng cần phải trùng khớp với câu hỏi trong kịch bản. Nếu có bất kì những yêu cầu nào ngoài kịch bản đã lập trình sẵn, chatbot hoàn toàn không thể đáp ứng được.

Do việc phụ thuộc vào kịch bản có sẵn hay các luồng hội thoại được xây dựng từ trước khiến cho chatbot tương tác rập khuôn, thiếu tự nhiên với người dùng. Nhiều trường hợp chatbot không hiểu hoặc hiểu sai và đưa ra những phản hồi thiếu chính xác đối với những yêu cầu của người dùng.



Hình 1.7: Mô phỏng chatbot truyền thống

2. **AI chatbot**

   AI chatbot là một chương trình phần mềm được xây dựng và triển khai với sự trợ giúp của Trí tuệ nhân tạo (AI) giúp hỗ trợ người dùng và tạo ra những tương tác tự nhiên giữa bot và người dùng. Trong 2 năm trở về đây, AI đang trở thành vấn đề rất phổ biến trong xã hội cũng như trong giáo dục. AI có khả năng xử lý lượng dữ liệu rất lớn với tốc độ cao và từ đó sẽ đưa ra các phán đoán mà không cần sự can thiệp của con người.

   Với công cụ Machine Learning khớp các ý nghĩa đầu vào với cơ sở dữ liệu để tìm và nạp những thông tin liên quan. Cuối cùng chatbot sẽ xác định câu trả lời tốt nhất phù hợp với truy vấn của người dùng và tạo ra được phản hồi thân thiện và tự nhiên hơn với con người.

   Điều nổi trội nhất của các AI chatbot hiện nay là sẽ tự học thêm các từ mới cũng như hiểu ý định của người dùng để giữ ngữ cảnh cho các câu hỏi tiếp theo trong cùng 1 cuộc hội thoại. Từ đó sẽ tăng được trải nghiệm người dùng cũng như sẽ đáp ứng được những truy vấn ở những lần tiếp theo.

3. **So sánh giữa chatbot truyền thống và AI chatbot**

Tiêu chí	AI Chatbot	Chatbot truyền thống
Khả năng hiểu và xử lý ngôn ngữ	Sử dụng công nghệ xử lý ngôn ngữ tự nhiên (NLP). Hiểu và phản hồi câu hỏi tự nhiên, học hỏi từ các cuộc trò chuyện trước đó.	Dựa vào kịch bản và quy tắc định sẵn. Phản hồi dựa trên từ khóa, hạn chế trong việc xử lý ngôn ngữ phức tạp.
Khả năng tùy biến	Tự động cập nhật và thay đổi phản hồi dựa trên dữ liệu mới và mô hình học máy. Thích nghi tốt với các tình huống và câu hỏi mới.	Cần lập trình lại hoặc cập nhật kịch bản khi có thay đổi. Ít khả năng thích nghi với câu hỏi ngoài kịch bản.
Hiệu quả và trải nghiệm người dùng	Mang lại trải nghiệm người dùng tốt hơn nhờ hiểu ngữ cảnh và phản hồi chính xác. Xử lý nhiều yêu cầu phức tạp cùng lúc.	Gặp khó khăn với yêu cầu ngoài kịch bản hoặc câu hỏi phức tạp. Trải nghiệm người dùng bị gián đoạn nếu không có phản hồi phù hợp.
Ứng dụng và chi phí triển khai	Phù hợp với doanh nghiệp cung cấp dịch vụ khách hàng tiên tiến, hỗ trợ quy trình kinh doanh phức tạp. Chi phí triển khai và duy trì cao hơn.	Thích hợp cho doanh nghiệp nhỏ hoặc ứng dụng đơn giản. Chi phí triển khai thấp hơn và dễ bảo trì.

Bảng 1.2: So sánh giữa AI Chatbot và Chatbot truyền thống

1.3.3 Quy trình xây dựng AI chatbot

Các bước thực hiện bao gồm:

1. **Thu thập dữ liệu**

   Bước đầu tiên là thu thập dữ liệu văn bản từ các nguồn khác nhau như tài liệu, website, mạng xã hội hoặc dữ liệu có sẵn. Dữ liệu có thể gồm các câu, đoạn văn, cuộc hội thoại, hoặc tài liệu đầy đủ, tùy thuộc vào bài toán cần giải quyết.

2. **Tiền xử lý dữ liệu**

   Tokenization: Chia nhỏ văn bản thành các đơn vị cơ bản (tokens), có thể là từ hoặc ký tự.

   Áp dụng các kĩ thuật xử lý văn bản như loại bỏ các kỹ tự đặc biệt, chuyển đổi sang chữ thường, loại bỏ stopwords, ...

3. **Biểu diễn văn bản**

   Word Embedding (e.g., Word2Vec, GloVe): Kỹ thuật biểu diễn từ dưới

dạng vector có ý nghĩa ngữ nghĩa, giúp mô hình hiểu mối quan hệ giữa các từ và ngữ cảnh.

Sentence Embedding: Biểu diễn cả câu hoặc đoạn văn dưới dạng vector để mô hình nắm bắt được ngữ cảnh ở mức câu.

4. **Chọn mô hình**

   Chọn mô hình phù hợp dựa trên yêu cầu của bài toán. Có thể sử dụng các mô hình truyền thống như Naive Bayes, SVM hoặc các mô hình học sâu như RNN, LSTM, Transformer.

   Với các bài toán phức tạp, mô hình ngôn ngữ lớn (LLM) như GPT, BERT, hoặc Transformer-based models là lựa chọn tốt để xử lý ngữ nghĩa và ngữ cảnh phức tạp.

5. **Tinh chỉnh và đánh giá mô hình**

   Sau khi chọn mô hình, tiến hành huấn luyện và tinh chỉnh để đạt hiệu suất tối ưu cho bài toán cụ thể. Đánh giá mô hình thông qua các chỉ số như độ chính xác, F1-score, độ nhạy, độ đặc hiệu giúp đảm bảo mô hình đạt yêu cầu về chất lượng và khả năng dự đoán.

6. **Triển khai và tự động hóa quy trình**

   Khi mô hình đạt hiệu suất mong muốn, tiến hành triển khai vào ứng dụng thực tế, như chatbot, phân tích cảm xúc, hoặc dịch máy. Để đảm bảo tính ổn định và tiết kiệm công sức, nên tự động hóa quy trình từ việc thu thập dữ liệu mới, tiền xử lý, cho đến tinh chỉnh và cập nhật mô hình. Quy trình tự động này giúp mô hình luôn được cải thiện và thích ứng với dữ liệu thay đổi.

Chương 2

Cơ sở lý thuyết

2.1 Xây dựng kiến trúc của LLM chatbot

Để giải quyết bài toán được đặt ra trong đồ án này, tôi sẽ xây dựng mô hình chatbot tư vấn tuyển sinh sử dụng phương pháp Retrieval-Augmented Generation (RAG) kết hợp với các mô hình ngôn ngữ lớn (LLMs). Chatbot được chia thành 3 phần chính: *NLU (Natural Language Understanding)*, *RAG (Retrieval-Augmented Generation)*, *NLG (Natural Language Generation)*. Với mỗi phần ở trên sẽ đóng một vai trò riêng của mình.

1. **NLU (Hiểu ngôn ngữ tự nhiên)**
   - Phân loại ý định (Intent Classification): Xác định ý định của người dùng, chẳng hạn như hỏi về "thông tin tuyển sinh," "điều kiện xét tuyển," "học phí," v.v.
   - Trích chọn thông tin (Entity Extraction): Từ câu hỏi của người dùng, trích xuất các thực thể quan trọng, chẳng hạn như "ngành học", "kỳ tuyển sinh", "điểm xét tuyển", v.v. Phần này giúp hệ thống hiểu rõ hơn về những thông tin mà người dùng cần tìm.
2. **RAG (Truy xuất thông tin kết hợp sinh văn bản)**

   Kỹ thuật RAG là một kĩ thuật kết hợp hai phần:
   - Truy xuất (Retriever): Khi người dùng đặt câu hỏi, hệ thống sẽ truy xuất các đoạn văn bản hoặc tài liệu có liên quan từ cơ sở dữ liệu tuyển sinh của trường. Điều này có thể thực hiện bằng cách sử dụng các kỹ thuật truy xuất thông tin (IR) dựa trên mô hình nhúng ngữ nghĩa

(semantic embedding) như FAISS, hoặc Elasticsearch.

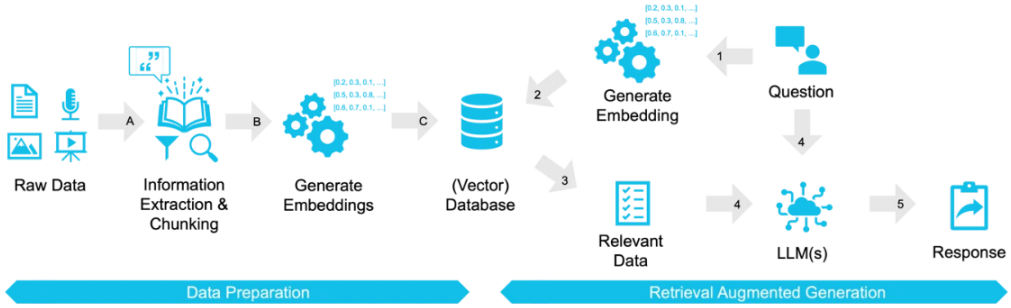
- Sinh văn bản (Generator): Sau khi truy xuất các đoạn tài liệu liên quan, hệ thống sẽ truyền chúng vào một mô hình ngôn ngữ lớn (LLM) để tạo câu trả lời tự nhiên dựa trên ngữ cảnh của câu hỏi. Phần sinh câu trả lời này giúp tạo ra một phản hồi mượt mà và phù hợp với ngữ điệu tự nhiên.

3. **NLG (Sinh ngôn ngữ tự nhiên)**

   Sinh câu trả lời: Sau khi RAG tạo ra câu trả lời dựa trên thông tin được truy xuất, mô hình LLM có thể cải tiến câu trả lời để đáp ứng phong cách ngôn ngữ tự nhiên và phù hợp với ngữ cảnh của người dùng. Phần này có thể dùng để cung cấp thêm thông tin phụ trợ hoặc điều chỉnh lại câu trả lời để nó dễ hiểu và gần gũi với ngôn ngữ hội thoại.

Quy trình xử lý của chatbot:

1. Phân loại ý định người dùng: Khi nhận câu hỏi, chatbot xác định ý định người dùng có phải là về thông tin tuyển sinh hay không và ý định cụ thể là gì.
2. Trích xuất thông tin quan trọng: Từ câu hỏi, chatbot trích xuất các thực thể quan trọng như ngành học, kỳ tuyển sinh, mức học phí, điểm chuẩn, v.v.
3. Truy xuất dữ liệu (Retriever): Dựa trên các thực thể đã trích xuất, chatbot truy vấn cơ sở dữ liệu để lấy các đoạn văn bản có liên quan. Điều này sẽ giúp đảm bảo chatbot có các thông tin chính xác và cập nhật nhất để trả lời câu hỏi.
4. Sinh câu trả lời (Generator): Sử dụng mô hình LLM, chatbot tạo câu trả lời mượt mà và tự nhiên dựa trên các đoạn văn bản đã truy xuất.
5. Phản hồi người dùng: Chatbot gửi câu trả lời cho người dùng, đảm bảo thông tin chính xác và đúng trọng tâm.



Hình 2.1: Quy trình thực hiện chatbot

2.2 Bài toán xác định ý của người dùng

Để có thể phân loại chính xác ý định của người dùng, chúng ta cần một mô hình biểu diễn ngôn ngữ dưới dạng vector số học – một phương pháp mã hóa giúp truyền tải được ý nghĩa và sắc thái ngữ nghĩa của từ ngữ vào các con số (vector hóa). Trong các phương pháp phổ biến hiện nay, word embedding nổi bật lên như một công cụ mạnh mẽ và thiết yếu trong xử lý ngôn ngữ tự nhiên (NLP).**Vậy embedding là gì?** Embedding là quá trình chuyển đổi các token (từ hoặc cụm từ) thành các vector số học, và vai trò của các vector này trong việc giúp mô hình hiểu ngữ nghĩa và ngữ cảnh[24]. Thay vì coi mỗi từ là một thực thể độc lập, word embedding cho phép các từ được mã hóa thành các vector số thực nằm trong một không gian ngữ nghĩa liên tục, nơi mối quan hệ giữa các từ được thể hiện bằng khoảng cách và hướng giữa các vector.

Một số phương pháp word embedding truyền thống nổi tiếng đã được phát triển và tối ưu hóa qua nhiều năm như Word2Vec của Google, GloVe của Stanford, và gần đây nhất là FastText của Facebook, một phương pháp cải tiến nhằm xử lý tốt hơn những ngôn ngữ có từ ghép phức tạp và từ không có trong từ điển.

Tuy nhiên, các kĩ thuật embedding truyền thống hoạt động ở cấp từ, nên nó không thể giải quyết được các trường hợp từ đa nghĩa (polysemy) hay ngữ cảnh của câu hỏi. Vì vậy, các phương pháp embedding dựa trên Transformer (BERT embedding, GPT embedding) hiện nay đang được sử dụng để hiểu ngữ cảnh và ngữ nghĩa sâu sắc hơn. Các mô hình này có khả năng phân tích câu hỏi ở mức độ tinh vi, giúp cải thiện đáng kể độ chính xác và hiệu quả của hệ thống hỏi đáp.

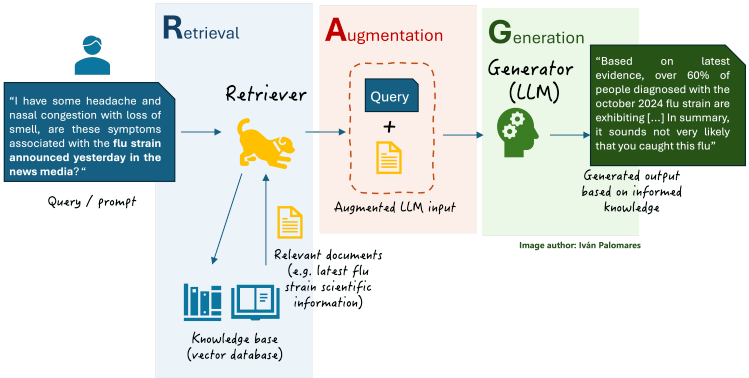
2.3 Bài toán trích xuất thông tin

2.3.1 Phương pháp Retrieval-Augmented Generation (RAG)

Trong vài năm trở lại đây, LLM phát triển nhanh chóng thì việc áp dụng RAG với các mô hình LLM đã khiến nâng cao khả năng phản hồi cho các mô hình ngôn ngữ lớn (LLM). Phương pháp này không chỉ giúp LLM xử lý thông tin mở rộng và cập nhật mà còn giảm thiểu chi phí tính toán và thời gian xử lý cho các truy vấn có ngữ cảnh dài[18]. Bằng cách sử dụng cơ sở dữ liệu ngoài để bổ sung kiến thức, RAG khắc phục các hạn chế vốn có của LLMs, chẳng hạn như khó khăn trong việc cập nhật kiến thức, hạn chế về ngữ cảnh, và chi phí tính toán cao. Dưới đây là các kiến trúc và cơ chế RAG nổi bật, cũng như các ứng dụng đặc thù của chúng.

Kiến trúc cơ bản của RAG:

Hệ thống RAG thường bao gồm ba thành phần chính là *Retriever (hệ thống tìm kiếm)*, *Retrieval fusion (cơ chế hợp nhất dữ liệu truy xuất)*, và *Generator (bộ sinh văn bản)*. Quá trình hoạt động của RAG diễn ra qua ba giai đoạn cơ bản



Hình 2.2: Kiến trúc cơ bản của RAG

- Giai đoạn truy xuất (Retrieval): Ở giai đoạn đầu tiên, retriever (hệ thống tìm kiếm) sẽ nhận truy vấn từ người dùng và thực hiện tìm kiếm trong cơ sở dữ liệu ngoài. Thông tin truy xuất thường là các đoạn dữ liệu ngắn có liên quan đến truy vấn, như các đoạn văn, câu trả lời tiềm năng, hoặc các thông tin bổ trợ. Đây là bước xác định và thu thập dữ liệu cần thiết nhằm giúp mô hình sinh văn bản có được nguồn dữ liệu bên ngoài hỗ trợ cho việc xây dựng câu trả lời hoặc thông tin đầu ra.
- Giai đoạn hợp nhất dữ liệu (Retrieval Fusion): Ở bước này, các thông tin

truy xuất được kết hợp với truy vấn ban đầu thông qua một trong ba kỹ thuật chính:

  - Query-based Fusion: Thông tin truy xuất được kết hợp trực tiếp vào truy vấn, và mô hình sẽ xử lý cả truy vấn cùng với các thông tin liên quan, đảm bảo rằng tất cả thông tin cần thiết đã có sẵn trước khi bắt đầu sinh văn bản.
  - Logits-based Fusion: Kết quả truy xuất không kết hợp vào truy vấn mà vào giai đoạn logits đầu ra của mô hình. Kỹ thuật này giúp tinh chỉnh độ chính xác bằng cách phối hợp các logits của thông tin truy xuất và logits của mô hình sinh văn bản, cho phép mô hình tạo ra các câu trả lời chính xác hơn.
  - Latent Fusion: Thông tin truy xuất được tích hợp vào các trạng thái ẩn (latent states) của mô hình, cải thiện khả năng hiểu ngữ cảnh trong suốt quá trình sinh văn bản.

- Giai đoạn sinh văn bản (Generation): Cuối cùng, sau khi hợp nhất dữ liệu từ giai đoạn trước, generator (bộ sinh văn bản) sẽ tạo ra nội dung đầu ra dựa trên truy vấn và các thông tin bổ trợ đã được hợp nhất. Kết quả cuối cùng là một phản hồi có độ chính xác và tính liên quan cao nhờ sự hỗ trợ từ các thông tin truy xuất trước đó[17]

Ứng dụng của RAG trong các mô hình xử lý văn bản dài

Với sự phát triển của các mô hình xử lý văn bản dài như GPT-4o và Gemini 1.5, RAG đã cho thấy lợi thế về hiệu quả chi phí khi xử lý các dữ liệu văn bản rất dài. Các nghiên cứu chỉ ra rằng mặc dù RAG có thể không mạnh bằng các mô hình văn bản dài hiện đại, nhưng nó lại có chi phí thấp hơn đáng kể do giảm được số lượng tokens đầu vào cho mô hình[20].

2.3.2 Lưu trữ dữ liệu

Quá trình Ingestion trong RAG: Biến đổi và lưu trữ dữ liệu

Ingestion là quy trình chuyển đổi và chuẩn bị dữ liệu văn bản để sẵn sàng cho truy vấn và sử dụng trong các hệ thống RAG. Quy trình này bao gồm ba bước chính:

- **Thu thập dữ liệu**: Dữ liệu văn bản được thu thập từ nhiều nguồn đa dạng, bao gồm các công cụ tìm kiếm, cơ sở dữ liệu, đồ thị tri thức, và nhiều định dạng khác nhau như PDF, Word, HTML, Markdown. Mục tiêu của bước này là tập hợp một lượng lớn thông tin từ các nguồn tin cậy và

phù hợp với mục tiêu ứng dụng.

- **Tiền xử lý dữ liệu**: Văn bản sau khi thu thập sẽ được xử lý để loại bỏ những thông tin không có giá trị tri thức, loại bỏ nhiễu và sắp xếp lại cấu trúc cho phù hợp. Văn bản sau đó được chia thành các phần nhỏ hơn (thường gọi là chunks) nhằm tối ưu hóa khả năng lưu trữ và truy xuất thông tin sau này.
- **Lập chỉ mục và lưu trữ vào cơ sở dữ liệu vector**: Bước cuối cùng là lựa chọn cơ sở dữ liệu vector phù hợp và phương án lập chỉ mục (indexing) tương ứng với từng loại dữ liệu. Dữ liệu sau khi được indexing sẽ lưu trữ trong vector database, cho phép truy xuất thông tin nhanh chóng, chính xác và hiệu quả khi hệ thống thực hiện quá trình truy vấn thông tin.

Phương pháp chia nhỏ tài liệu (Document chunking)

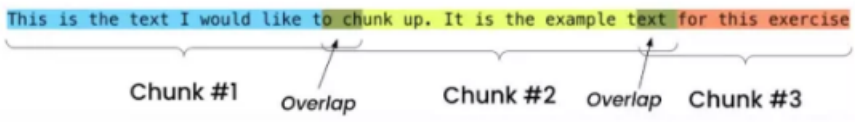
Document chunking là kỹ thuật chia nhỏ tài liệu lớn thành các đoạn văn bản nhỏ hơn, giúp dễ dàng quản lý và truy xuất trong các ứng dụng sử dụng mô hình ngôn ngữ lớn (LLM). Mục tiêu của kỹ thuật này là cung cấp chính xác lượng thông tin cần thiết cho nhiệm vụ cụ thể, đảm bảo tính hiệu quả và tránh quá tải dữ liệu cho mô hình.

Trong các hệ thống RAG (Retrieval-Augmented Generation), chunking đóng vai trò quan trọng, hỗ trợ quá trình lưu trữ và truy xuất thông tin nhanh chóng từ các tài liệu lớn[8].

Các cấp độ Chunking

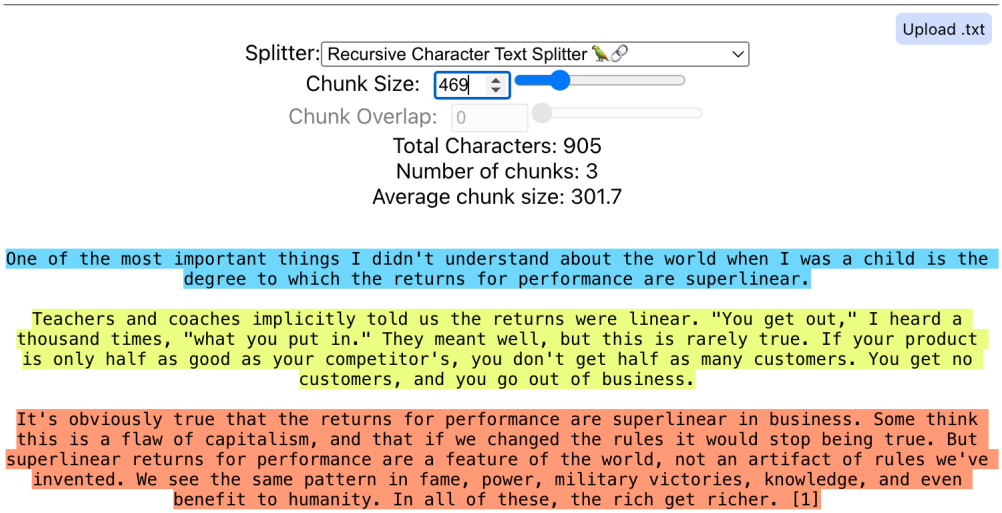
Có năm cấp độ chunking từ cơ bản đến nâng cao, dựa trên độ phức tạp và hiệu quả của phương pháp[8]:

- Fixed size chunking: Phương pháp đơn giản nhất, chia tài liệu thành các đoạn có kích thước cố định dựa trên số ký tự, mà không cần quan tâm đến nội dung hay cấu trúc của văn bản. Các framework như Langchain và Llamaindex cung cấp lớp CharacterTextSplitter để triển khai phương pháp này. Các yếu tố cần chú ý bao gồm:
  - Cách chia văn bản: theo ký tự.
  - Kích thước đoạn: số lượng ký tự trong mỗi đoạn.
  - Chunk_overlap: số ký tự bị trùng lặp giữa các đoạn liền kề.



Hình 2.3: Fixed size chunking

- Recursive chunking: Khắc phục nhược điểm của Fixed Size Chunking bằng cách xem xét cấu trúc của văn bản. Recursive Chunking phân đoạn tài liệu dựa trên các điểm phân tách nhất định như dấu ngắt dòng, dấu chấm, hoặc các ký tự đặc biệt. Phương pháp này sử dụng một chuỗi các bộ phân tách (separators) để chia nhỏ dần văn bản cho đến khi đạt kích thước mong muốn.



Hình 2.4: Recursive chunking

- Document-Based Chunking: Phương pháp này phân đoạn tài liệu dựa trên cấu trúc nội tại của nó, thích hợp cho các tài liệu có cấu trúc rõ ràng như Markdown, mã nguồn Python/JavaScript, bảng dữ liệu, hoặc các tài liệu đa phương tiện (multi-modal) chứa hình ảnh. Ví dụ:

  - Với tài liệu Markdown: sử dụng lớp MarkdownTextSplitter.

  - Với mã nguồn Python/JavaScript: sử dụng lớp PythonCodeTextSplitter để phân đoạn theo hàm hoặc lớp.

  - Đối với bảng dữ liệu: nên định dạng lại dưới dạng HTML hoặc CSV để giữ nguyên mối quan hệ dữ liệu giữa các hàng và cột.

  - Đối với tài liệu có chứa hình ảnh: sử dụng các mô hình đa phương thức để phân tích và lưu trữ nội dung hình ảnh.

- Semantic chunking: Ở cấp độ này, chunking không chỉ dựa vào cấu trúc nội dung mà còn xem xét mối quan hệ ngữ nghĩa giữa các đoạn văn bản. Phương pháp này sử dụng các phép nhúng (embeddings) để nhận diện ngữ cảnh và phân đoạn dựa trên sự tương đồng ngữ nghĩa. SemanticSplitterNodeParse trong Llamaindex cho phép phân đoạn dựa vào ngữ nghĩa với các tham số như buffer_size (kích thước cửa sổ ban đầu cho từng đoạn) và breakpoint_percentile_threshold (ngưỡng để quyết định điểm phân tách).
- Agentic Chunking: Đây là cấp độ cao nhất, cho phép mô hình ngôn ngữ lớn (LLM) tự quyết định số lượng và nội dung của các đoạn văn bản dựa trên ngữ cảnh. Bằng cách sử dụng các tuyên đề (propositions), LLM có thể quyết định đoạn nào nên được ghép nối hoặc tạo mới. Phương pháp này phù hợp với các tài liệu phức tạp và yêu cầu sự tinh chỉnh cao trong phân đoạn.

Document Chunking là một phương pháp hữu hiệu để xử lý dữ liệu văn bản lớn trong các hệ thống LLM, giúp cải thiện tốc độ và độ chính xác khi truy xuất thông tin. Mỗi cấp độ chunking đều có đặc điểm riêng, từ các phương pháp đơn giản như Fixed Size Chunking cho đến các phương pháp phức tạp hơn như Agentic Chunking, phù hợp với các loại tài liệu và yêu cầu ứng dụng khác nhau. Việc lựa chọn cấp độ chunking phù hợp giúp tối ưu hóa hệ thống và nâng cao chất lượng của các ứng dụng RAG dựa trên mô hình ngôn ngữ lớn.

Vector database

Vector database là một phần quan trọng trong việc xây dựng các ứng dụng AI có khả năng mở rộng. Cơ sở dữ liệu vector cung cấp bộ nhớ dài hạn, trên mô hình học máy hiện có. Nếu không sử dụng cơ sở dữ liệu vector, chúng ta cần phải đào tạo mô hình chạy lại trên tập dữ liệu ban đầu thông qua một mô hình trước khi truy vấn, điều này sẽ dẫn đến việc chậm và gây tốn kém tài nguyên sử dụng.

Cơ sở dữ liệu vector sẽ xác định dữ liệu nào được biểu diễn dưới dạng vector mà gần với truy vấn đầu vào của bài toán. Điều này cho phép xây dựng các trường hợp sử dụng khác nhau trên cơ sở dữ liệu vector bao gồm:

- Tìm kiếm ngữ nghĩa, được sử dụng để trả về kết quả tương tự như thông tin đầu vào của truy vấn.

- Phân loại, được sử dụng để trả về nhóm (hoặc các nhóm) gần nhất với truy vấn đầu vào.

- Công cụ đề xuất, được sử dụng để trả về nội dung tương tự như nội dung

đầu vào dựa trên các tiêu chí khác nhau .

- Phát hiện bất thường, được sử dụng để xác định xem các điểm dữ liệu cụ thể có giống với dữ liệu hiện có hay khác biệt.

Trong đồ án này, vector database hỗ trợ các tác vụ Retrieval Augmented Generation (RAG), cho phép đưa thêm ngữ cảnh vào LLM (Mô hình ngôn ngữ lớn) bằng cách sử dụng ngữ cảnh từ tìm kiếm vectơ để tăng cường lời nhắc của người dùng.

2.3.3 Tìm kiếm thông tin

Trong trường hợp sử dụng tìm kiếm vectơ truyền thống, các truy vấn được thực hiện trên cơ sở dữ liệu vectơ bằng cách truyền cho nó một vectơ truy vấn và để cơ sở dữ liệu vectơ trả về danh sách các vectơ có thể cấu hình được với khoảng cách ngắn nhất ("giống nhất") với vectơ truy vấn.

Các bước thực hiện tìm kiếm thông tin được sử dụng trong đồ án này như sau:

Bước 1: Chuyển đổi tập dữ liệu gồm các file .docx và pdf thành một tập hợp các vector embedding bằng cách sử dụng một mô hình học máy được đào tạo cho kiểu dữ liệu đó (mô hình được lấy từ Hugging Face).

Bước 2: Các nhúng đầu ra được chèn vào chỉ mục cơ sở dữ liệu.

Bước 3: Truy vấn tìm kiếm, yêu cầu phân loại hoặc truy vấn phát hiện bất thường cũng được truyền qua cùng một mô hình ML, trả về biểu diễn nhúng vectơ của truy vấn.

Bước 4: Vector được truy vấn bằng nhúng này và trả về tốp các nhúng vector giống nhất với truy vấn được cung cấp.

Bước 5: Các nhúng được trả về được sử dụng để truy xuất các đối tượng nguồn gốc từ bộ lưu trữ chuyên dụng (R2, KV và D1) và trả lại cho người dùng.

Lưu ý: Với quy trình không sử dụng cơ sở dữ liệu vector, chúng ta sẽ cần phải truyền toàn bộ tập dữ liệu cùng với truy vấn mỗi lần, điều này không thực tế (các mô hình có giới hạn về kích thước đầu vào) và sẽ tiêu tốn đáng kể tài nguyên và thời gian.

2.4 Bài toán sinh dữ liệu

2.4.1 Mô hình Pre-trained

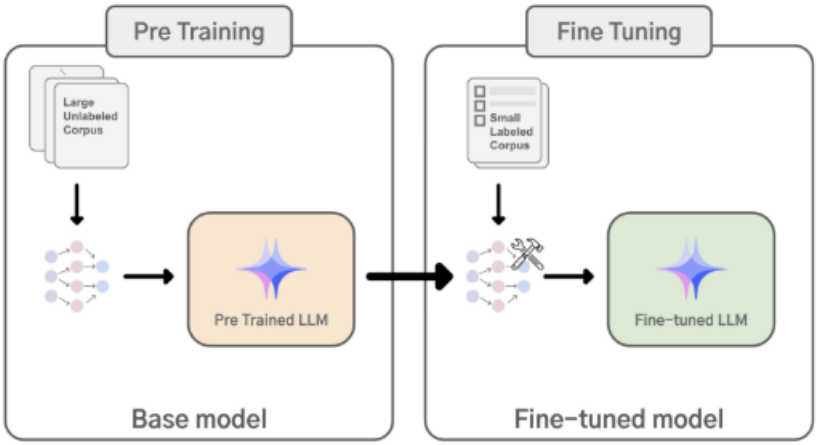
Pre-trained model là mô hình học máy đã được huấn luyện trên một tập dữ liệu trước đó và có thể được sử dụng lại cho các tác vụ tương tự mà không cần huấn luyện lại từ đầu.

Việc sử dụng pre-trained model trong đồ án này giúp tiết kiệm thời gian và tài nguyên sử dụng. Vì pre-trained model được huấn luyện trên một tập dữ liệu lớn trước đó nên việc phản hồi người dùng sẽ khá tự nhiên.

Với nghiên cứu này, em sẽ sử dụng model GPT4-o mini (sử dụng qua OpenAPI Key) kết hợp cùng với phương pháp RAG đã được trình bày ở trên để so sánh với các model đã được fine-tuning (được trình bày ở phần 3.4.2)

2.4.2 Mô hình Fine-tuning

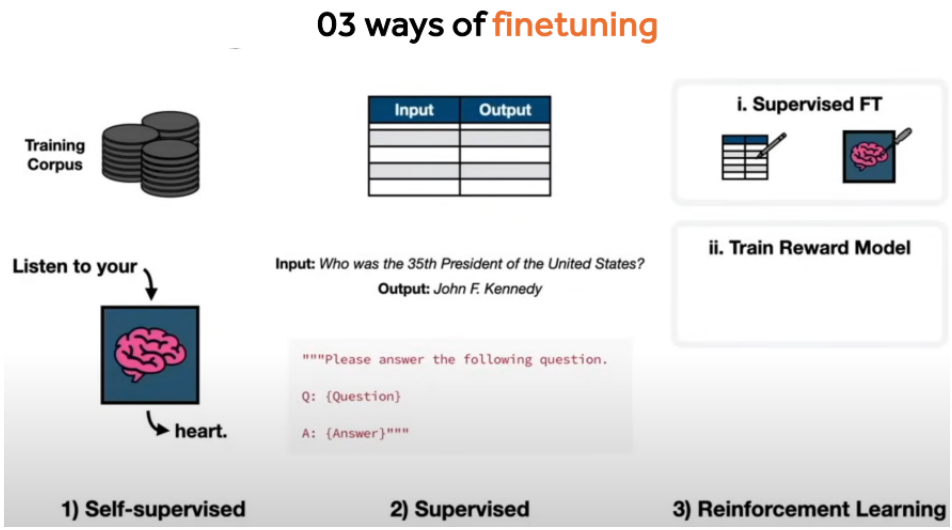
Fine-tuning là quá trình lấy một mô hình học máy được đào tạo trước (như mô hình LLM, mô hình thị giác máy tính hoặc các mạng nơ-ron khác) và điều chỉnh nó để thực hiện một nhiệm vụ cụ thể bằng cách đào tạo nó trên một tập dữ liệu nhỏ hơn để phục vụ cho nhiệm vụ đó[9]. Đây là một cách tiếp cận phổ biến của học chuyển giao, trong đó kiến thức thu được từ một nhiệm vụ được sử dụng lại để giải quyết một nhiệm vụ liên quan nhưng khác.



Hình 2.5: Mô phỏng quá trình fine-tuning

Hiện nay, có 3 phương pháp chính để fine-tuning mô hình:

- Self-supervised learning: Mô hình được tinh chỉnh dựa trên dữ liệu không có nhãn, bằng cách tận dụng các cấu trúc nội tại trong dữ liệu để tạo ra các bài toán học tự động.
- Supervised learning: Quá trình tinh chỉnh mô hình dựa trên dữ liệu có nhãn, trong đó mô hình học từ các cặp đầu vào và đầu ra đã biết.
- Reinforcement learning: Mô hình được tinh chỉnh thông qua cơ chế phần thưởng và hình phạt, nhằm tối ưu hóa hành vi để đạt được mục tiêu cụ thể trong một môi trường tương tác.



Hình 2.6: 3 phương pháp fine-tuning mô hình

Trong đồ án này, tôi sử dụng phương pháp supervised fine tuning. Quy trình fine-tuning như sau:

- Bước 1: Chọn một công việc để fine-tuning ví dụ như hỏi đáp về nội quy lao động, hỏi đáp về quy định sinh viên,...
- Bước 2: Chuẩn bị cơ sở dữ liệu bao gồm các bộ câu hỏi-câu trả lời về một chủ đề cụ thể.
- Bước 3: Chọn mô hình gốc ví dụ: GPT, VinaLlama, PhoGPT,...
- Bước 4: Fine-tune mô hình và đánh giá mô hình đó.

Hình 2.7: Quy trình fine-tuning mô hình LLM

Có 3 cách để đào tạo lại tham số cho mô hình: tinh chỉnh toàn bộ tham số (train all parameters), học chuyển giao (transfer learning), và PERT (Parameter-Efficient Transfer Learning). Trong PERT, một công nghệ nổi bật là LoRA Fine-Tuning (Low-Rank Adaptation), cho phép tối ưu hóa hiệu quả quá trình tinh chỉnh mô hình bằng cách chỉ điều chỉnh một tập hợp nhỏ các tham số dưới dạng ma trận hạng thấp, thay vì huấn luyện toàn bộ mô hình.

a. Phương pháp LoRA Fine-tuning

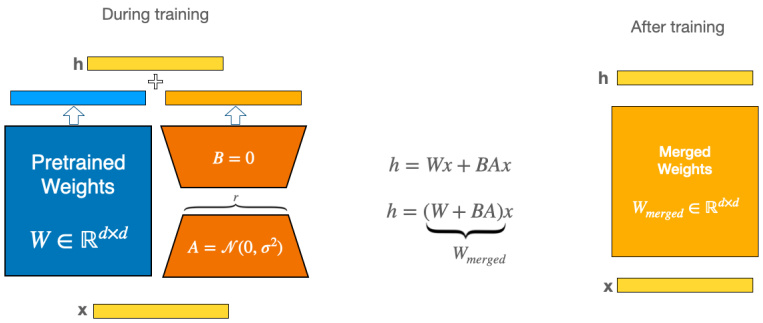
Phương pháp của LoRA là biểu diễn các bản cập nhật trọng số bằng hai ma trận nhỏ hơn (gọi là ma trận cập nhật) thông qua phân tích hạng thấp. Các ma trận mới này có thể được đào tạo để thích ứng với dữ liệu mới trong khi vẫn giữ tổng số thay đổi ở mức thấp. Ma trận trọng số ban đầu vẫn được đóng băng và không nhận thêm bất kỳ điều chỉnh nào nữa. Để tạo ra kết quả cuối cùng, cả trọng số ban đầu và trọng số đã điều chỉnh đều được kết hợp.

Ưu điểm:

- LoRA giúp việc tinh chỉnh hiệu quả hơn bằng cách giảm đáng kể số lượng tham số có thể đào tạo được.
- Trọng lượng được đào tạo trước ban đầu được giữ nguyên, nghĩa là bạn có thể có nhiều mô hình LoRA nhẹ và di động cho nhiều tác vụ tiếp theo được xây dựng dựa trên chúng.
- LoRA tương thích với nhiều phương pháp hiệu quả về tham số khác và có thể kết hợp với nhiều phương pháp trong số đó.
- Hiệu suất của các mô hình được tinh chỉnh bằng LoRA tương đương với hiệu suất của các mô hình được tinh chỉnh hoàn toàn.
- LoRA không thêm bất kỳ độ trễ suy luận nào vì trọng số bộ điều hợp có thể được hợp nhất với mô hình cơ sở.

Về nguyên tắc, LoRA có thể được áp dụng cho bất kỳ tập hợp con nào của ma trận trọng số trong mạng nơ-ron để giảm số lượng tham số có thể đào tạo. Tuy nhiên, để đơn giản và hiệu quả hơn nữa của tham số, trong các mô hình Transformer, LoRA thường chỉ được áp dụng cho các khối chú ý. Số lượng tham số có thể đào tạo được trong mô hình LoRA phụ thuộc vào kích thước của các ma trận cập nhật hạng thấp, chủ yếu được xác định bởi hạng r và hình dạng của ma trận trọng số ban đầu.

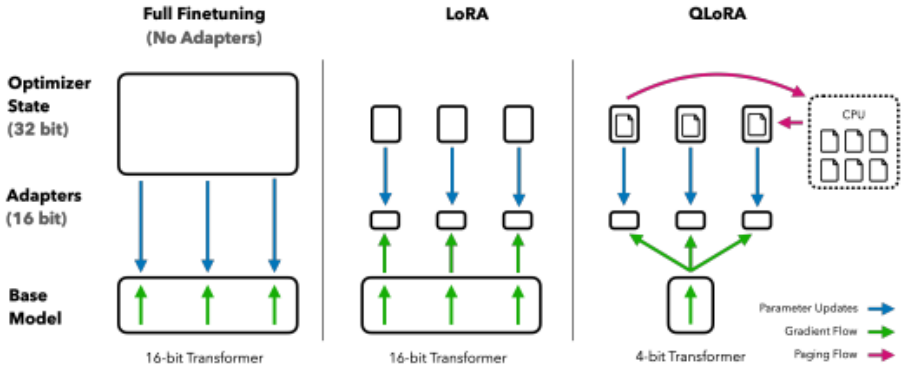
Mặc dù LoRA nhỏ hơn đáng kể và đào tạo nhanh hơn, chúng ta có thể gặp phải các vấn đề về độ trễ trong quá trình suy luận do tải riêng mô hình cơ sở và mô hình LoRA. Để loại bỏ độ trễ, em sử dụng hàm merge_and_unload() để hợp nhất trọng số bộ điều hợp với mô hình cơ sở, cho phép chúng ta sử dụng hiệu quả mô hình mới hợp nhất như một mô hình độc lập.



Hình 2.8: Mô phỏng quá trình hợp trọng số LoRA vào mô hình cơ sở

b. Phương pháp QLoRA Fine-tuning

Phương pháp của QLORA, sử dụng một kỹ thuật độ chính xác cao mới để lượng tử hóa một mô hình được đào tạo trước thành 4 bit, sau đó thêm một tập hợp nhỏ các trọng số Bộ điều hợp hạng thấp có thể học được được điều chỉnh bằng cách truyền ngược các gradient qua các trọng số được lượng tử hóa[20].



Hình 2.9: Các phương pháp tinh chỉnh khác nhau LoRA và QLoRA[22]

QLoRA hiện tại chỉ hỗ trợ cho Linear layer (vì LoRA cũng chỉ hộ trợ cho Linear). Một layer sẽ gồm 2 thành phần: thành phần pretrained (freeze) và thành phần LoRA (train).

Trong hầu hết các bài toán về LLMs hiện nay chúng ta sẽ quantize mô hình về dạng NF4. Trong quá trình tính toán output, thành phần pretrained sẽ được dequantize từ NF4 về FP16, rồi kết hợp tính toán với thành phần LoRA ở dạng FP16. Khi tính xong thì thành phàn pretrained lại được quantize lại về NF4. Vì vậy, QLoRA giúp chúng ta có thể đưa được model to vào vừa với VRAM để training, chứ không tăng tốc độ của model (do cứ đến layer nào cần tính toán thì ta lại phải dequantize nó về FP16).

2.5 Phương pháp đánh giá mô hình

Đánh giá chất lượng văn bản có thể cực kỳ khó khăn, nhưng lại là nền tảng cho bất kỳ mục đích sử dụng nào của các mô hình AI tạo sinh. Nhưng mặc dù việc đánh giá chất lượng văn bản có thể dễ dàng đối với con người, nhưng thường thì khó hơn nhiều đối với máy móc. Một lý do là các tiêu chí xác định chất lượng văn bản không được xác định rõ ràng. Hơn nữa, các tiêu chí đó có thể thay đổi, tùy thuộc vào mục đích và ngữ cảnh của văn bản.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) là thước đo để đánh giá độ tương đồng của văn bản bằng cách so sánh n-gram, chuỗi con chung dài nhất hoặc các từ chồng chéo giữa bản tóm tắt do máy tạo ra và bản tóm tắt tham chiếu[7]. Điểm ROUGE càng cao thì hai văn bản càng giống nhau.

Ví dụ: 1-gram là một từ đơn, 2-gram là một cặp từ, v.v. Điểm ROUGE có thể được tính cho các loại n-gram khác nhau, chẳng hạn như ROUGE-1, ROUGE-2

hoặc ROUGE-L (xem xét chuỗi con chung dài nhất).

Trong đồ án này, em sẽ sử dụng loại ROUGE-L để đánh giá độ tương đồng giữa câu trả lời của chatbot và câu trả lời được chuẩn bị sẵn trong bộ dataset.

- **ROUGE-L** là một thước đo đánh giá được sử dụng rộng rãi trong các tác vụ xử lý ngôn ngữ tự nhiên, đặc biệt là để tóm tắt và trả lời câu hỏi. Thước đo này đo độ tương đồng giữa văn bản được tạo ra và văn bản tham chiếu bằng cách tập trung vào Chuỗi con chung dài nhất (LCS) của các từ giữa hai văn bản.

  LCS cho phép ROUGE-L tính đến cả độ chính xác và thu hồi, cân bằng chúng thông qua điểm F1. Các công thức tính độ chính xác, thu hồi và Điểm F1 dựa trên LCS như sau:

$$\text{Precision}_{LCS} = \frac{LCS(X,Y)}{|X|}[7]$$

$$\text{Recall}_{LCS} = \frac{LCS(X,Y)}{|Y|}[7]$$

$$\text{F1}_{LCS} = \frac{(1+\beta^2)\cdot \text{Precision}_{LCS}\cdot \text{Recall}_{LCS}}{(\beta^2\cdot \text{Precision}_{LCS}) + \text{Recall}_{LCS}}[7]$$

  Trong đó:

  - $LCS(X,Y)$ là độ dài của dãy con chung dài nhất giữa văn bản được tạo X và văn bản tham chiếuY
  - $|X|$ là chiều dài của tạo ra văn bản và $|Y|$ là độ dài của văn bản tham chiếu.
  - Thông số β được đặt thành 1, làm cho điểm $F1$ trở thành giá trị trung bình hài hòa của độ chính xác và khả năng thu hồi

Chương 3

Dữ liệu và thực nghiệm

3.1 Dữ liệu

3.1.1 Giới thiệu

Trong đồ án này, em sử dụng hai bộ dữ liệu chính để cải thiện hiệu suất của các mô hình chatbot trong tư vấn tuyển sinh Đại học Bách Khoa Hà Nội: **Bộ dữ liệu cho RAG**, **Bộ dữ liệu cho Fine-tuning**. Mỗi bộ dữ liệu sẽ đóng vai trò quan trọng trong việc giúp mô hình tìm kiếm và sinh câu trả lời đúng với mục đích tuyển sinh.

1. Bộ dữ liệu cho RAG (Retrieval-Augmented Generation)

   Bộ dữ liệu sử dụng cho quá trình RAG chủ yếu được thiết kế để cung cấp thông tin từ các dữ liệu bên ngoài mô hình, giúp mô hình truy vấn và lấy thông tin từ Vector Database trước khi sinh phản hồi cho người dùng. Quá trình này nhằm nâng cao độ chính xác của chatbot với những dữ liệu đặc thù, ở đồ án này là dữ liệu về tuyển sinh.

   - Cấu trúc và nội dung bộ dữ liệu: Bộ dữ liệu bao gồm 7 folder với mục con của phần Tuyển sinh Đại học: thông tin chung, đề án tuyển sinh, xét tuyển tài năng, kỳ thi đánh giá tư duy, điểm chuẩn tuyển sinh, xác thực chứng chỉ ngoại ngữ, hướng nghiệp. Các file dữ liệu được lưu dưới định dạng .docx và pdf, được hệ thống hóa theo các chủ đề chính.
   - Mục đích sử dụng: Dữ liệu này được dùng để hỗ trợ quá trình Retrieval, trong đó mô hình RAG sẽ truy vấn các vector trong database sau quá trình chia chunks và embedding dữ liệu để trả lời chính xác các câu

hỏi phức tạp và ngoài phạm vi dữ liệu huấn luyện.

2. Bộ dữ liệu cho Fine-tuning

   Trả lời câu hỏi (QA) là một nhiệm vụ quan trọng trong Xử lý ngôn ngữ tự nhiên (NLP), trong đó các hệ thống được thiết kế để tự động cung cấp câu trả lời cho các câu hỏi được viết bằng ngôn ngữ hàng ngày. Trong QA miền mở, các câu hỏi có thể bao gồm nhiều chủ đề, do đó hệ thống cần có nền tảng kiến thức rộng hoặc khả năng tìm kiếm thông tin liên quan trên nhiều nguồn. Bộ dữ liệu cho fine-tuning được sử dụng để tinh chỉnh mô hình pre-trained với dữ liệu tuyển sinh cụ thể.

   - Ví dụ về bản ghi của bộ dữ liệu:

     Context: *Số lượng chương trình đào tạo: 64 chương trình, trong đó: • Số lượng chương trình đại trà (chương trình chuẩn): 36 chương trình; • Số lượng chương trình chất lượng cao - Elitech: 23 chương trình; • Số lượng chương trình PFIEV: 02 chương trình; • Số lượng chương trình liên kết đào tạo quốc tế: 03 chương trình.*

     Question: *Trường mình có tổng cộng bao nhiêu chương trình đào tạo và chi tiết từng loại chương trình như thế nào ạ?*

     Answer: *Trường mình có tổng cộng 64 chương trình đào tạo, trong đó có 36 chương trình đại trà, 23 chương trình chất lượng cao Elitech, 2 chương trình PFIEV và 3 chương trình liên kết quốc tế em nhé.*

     Context: *Những lưu ý khi thanh toán • Trường hợp thí sinh đã hoàn thành thao tác thanh toán, tài khoản đã trừ tiền nhưng hệ thống chưa hiển thị trạng thái Đã thanh toán, thí sinh gửi email tới địa chỉ tuyensinh@hust.edu.vn, cung cấp địa chỉ email, số CCCD/CMND, mật khẩu đăng nhập tài khoản, ảnh chụp minh chứng bị trừ tiền, phương thức thanh toán, số điện thoại,... hoặc liên hệ số hotline 024 3 868 3408 (trong giờ hành chính) để được hỗ trợ.*

     Question: *Nếu em đã thanh toán mà hệ thống chưa cập nhật thì em cần liên hệ với ai để được hỗ trợ ạ?*

     Answer: *Nếu em đã thanh toán mà hệ thống chưa cập nhật thì em hãy gửi email tới địa chỉ tuyensinh@hust.edu.vn, cung*

cấp địa chỉ email, số CCCD/CMND, mật khẩu đăng nhập tài khoản, ảnh chụp minh chứng bị trừ tiền, phương thức thanh toán, số điện thoại hoặc liên hệ số hotline 024 3 868 3408 để được hỗ trợ em nhé.

- Mục đích sử dụng: Dữ liệu fine-tuning giúp mô hình học và điều chỉnh các tham số của nó dựa trên các câu hỏi và tình huống cụ thể. Mục đích là giúp mô hình có thể hiểu và trả lời các câu hỏi tuyển sinh một cách chính xác sau khi kết hợp với RAG và sinh câu trả lời một cách tự nhiên hơn trong các tình huống thực tế.

3.1.2 Nguồn dữ liệu

Trong đồ án này, em sử dụng nguồn dữ liệu chính trên website Thông tin tuyển sinh Đại học Bách Khoa Hà Nội (*Link: https://ts.hust.edu.vn/*). Mỗi bộ dữ liệu đều có nguồn gốc và cách xử lý riêng, nhằm đáp ứng các yêu cầu cụ thể trong việc nâng cao hiệu quả và độ chính xác của hệ thống.

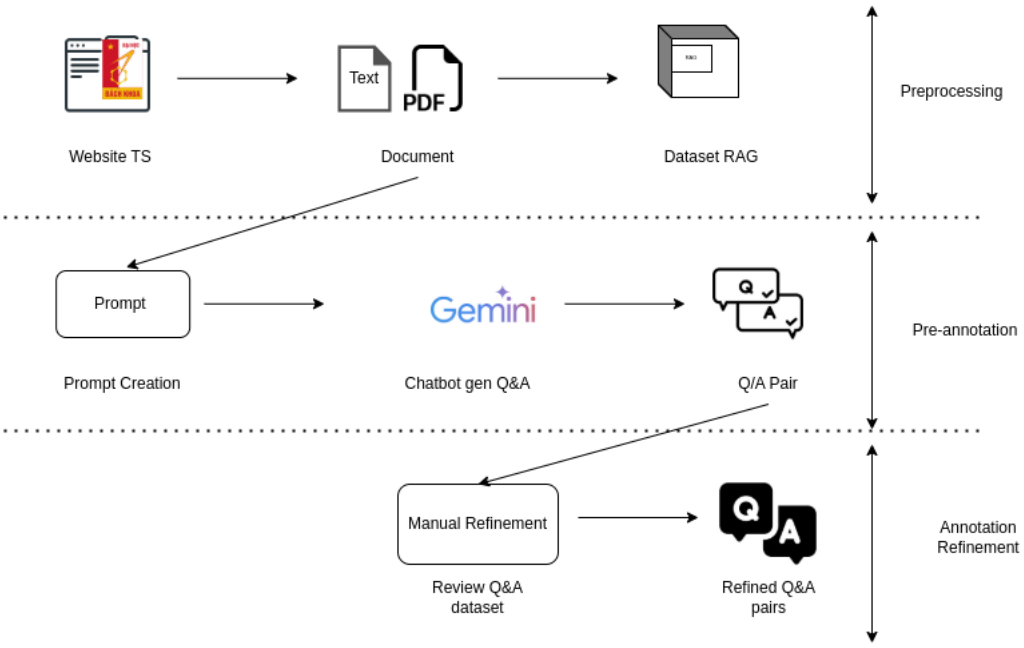
Hình 3.1: Trang web thông tin tuyển sinh HUST

3.1.3 Quá trình

Dưới đây là quy trình để lấy dữ liệu từ website đến bước tạo được 2 bộ dữ liệu để cho quá trình RAG và quá trình Fine-tuning.

Quá trình được chia làm 3 giai đoạn chính:

- **Pre-processing**: Crawl dữ liệu từ website tuyển sinh, chia ra các folder theo chủ đề chính, mỗi folder chứa những file text và pdf theo đúng chủ đề.
- **Pre-annotation**: Viết prompt cho ChatGPT4o và Gemini tạo ra những cặp câu hỏi Q&A theo dữ liệu đã thu thập.
- **Annotation Refinement**: Review lại nội dung và chỉnh sửa những câu trả lời bị sai, đơn giản để giúp bộ dữ liệu phong phú và chính xác hơn.



Hình 3.2: Quá trình tạo xử lý tập dữ liệu

3.1.4 Thống kê và đánh giá tập dữ liệu

1. Bộ dữ liệu cho RAG: Bộ dữ liệu gồm 7 folder bao gồm các file.docx và pdf.

Tên folder	**Số lượng file DOCX**	**Số lượng file PDF**
Thông tin chung	5	2
Đề án tuyển sinh	2	5
Xét tuyển tài năng	18	2
Điểm chuẩn tuyển sinh	22	0
Kỳ thi đánh giá tư duy	32	1
Xác thực chứng chỉ ngoại ngữ	2	0
Hướng nghiệp	9	0

Bảng 3.1: Số lượng file DOCX và PDF trong các folder

   Dữ liệu trong bộ dữ liệu này được lấy trực tiếp từ web tuyển sinh, sau đó chỉnh sửa lại các dữ liệu định dạng table về định dạng csv để nâng cao khả năng học của mô hình.

2. Bộ dữ liệu cho Fine-tuning:

   Cấu trúc và nội dung dữ liệu: Bộ dữ liệu chứa 1940 cặp ngữ cảnh, câu hỏi, câu trả lời liên quan đến các nội dung tuyển sinh đại học của trường.

	Train set	**Test set**	**%**
Số lượng câu hỏi	1746	194	80%
Số lượng câu trả lời	1746	194	20%

Bảng 3.2: Số lượng file DOCX và PDF trong các folder

Bộ dữ liệu được chia theo tỷ lệ 8:2 để sử dụng cho việc huấn luyện và kiểm tra. Với bộ dữ liệu để huấn luyện bao gồm 1746 cặp ngữ cảnh, câu hỏi, câu trả lời, với bộ dữ liệu kiểm tra bao gồm 194 cặp ngữ cảnh, câu hỏi, câu trả lời được tạo bằng ChatGPT4o và Google AI studio. Việc xem xét xét kỹ lưỡng tất cả các mẫu đã được tiến hành để đánh giá tính chính xác của các phản hồi do mô hình tạo ra.

3. Đánh giá tập dữ liệu:

   Việc đánh giá 3 bộ dữ liệu trên được thực hiện qua 3 giai đoạn: **Đánh giá riêng lẻ**, **Đánh giá chéo**, **Thảo luận nhóm và đồng thuận**

   - Đánh giá riêng lẻ: Một trong 4 người đánh giá độc lập phần phản hồi được giao cho người đó làm. Người đó sẽ đánh giá các câu trả lời dựa trên tiêu chí được xác định trước và chuyên môn tương ứng, tự so sánh với dữ liệu có sẵn để kiểm tra độ chính xác của câu trả lời.
   - Đánh giá chéo: Phần dữ liệu của người này sẽ được đổi chéo cho người kia kiểm tra và đánh giá dựa trên tiêu chí xác định trước và chuyên môn tương ứng và điền những nhận xét sang cột bên cạnh.
   - Đánh giá nhóm: Nhóm sẽ có cuộc meeting và cùng nhau đánh giá lại bộ dữ liệu về các câu hỏi và câu trả lời được đưa ra. Cuộc thảo luận tập thể này sẽ cho phép nhóm giải quyết bất kỳ sự khác biệt nào trong cách diễn giải hoặc phản hồi mâu thuẫn dẫn đến sự thống nhất về dữ liệu.

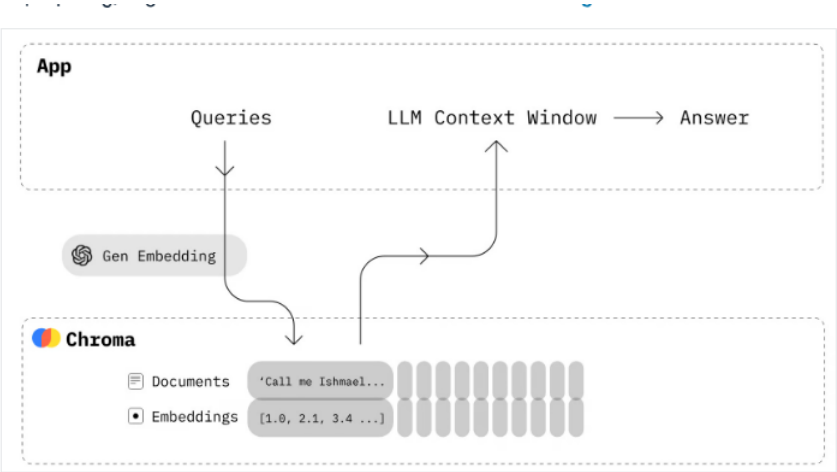
3.2 Cài đặt thử nghiệm

3.2.1 Cài đặt RAG sử dụng ChromaDB và OpenAPI

Chroma DB là một kho lưu trữ vector nguồn mở được sử dụng để lưu trữ và truy xuất các nhúng vector. Công dụng chính của nó là lưu các nhúng cùng với siêu dữ liệu để các mô hình ngôn ngữ lớn sử dụng sau này. Ngoài ra, nó cũng có thể được sử dụng cho các công cụ tìm kiếm ngữ nghĩa trên dữ liệu văn bản.

Tính năng của ChromaDB

- Đơn giản và mạnh mẽ: Cài đặt bằng lệnh đơn giản: `pip install chromadb`. Bắt đầu nhanh chóng với Python SDK, cho phép tích hợp liền mạch và thiết lập nhanh chóng.
- Đầy đủ tính năng: Tính năng truy xuất toàn diện : Bao gồm tìm kiếm vectơ, tìm kiếm toàn văn, lưu trữ tài liệu, lọc siêu dữ liệu và truy xuất đa phương thức.
- Khả năng mở rộng cao : Hỗ trợ nhiều nền tảng lưu trữ khác nhau như DuckDB để sử dụng cục bộ hoặc ClickHouse để mở rộng quy mô các ứng dụng lớn hơn.
- Hỗ trợ đa ngôn ngữ: Cung cấp SDK cho các ngôn ngữ lập trình phổ biến, bao gồm Python , JavaScript/TypeScript , Ruby , PHP và Java
- Tích hợp: Tích hợp gốc với các mô hình nhúng từ HuggingFace, OpenAI, Google, v.v. Tương thích với Langchain và LlamaIndex, nhiều công cụ tích hợp khác sẽ sớm ra mắt.
- Mã nguồn mở : Được cấp phép theo Apache 2.0 .
- Tốc độ và sự đơn giản: Tập trung vào sự đơn giản và tốc độ, được thiết kế để phân tích và truy xuất hiệu quả đồng thời trực quan khi sử dụng. Chroma DB cung cấp tùy chọn máy chủ tự lưu trữ.



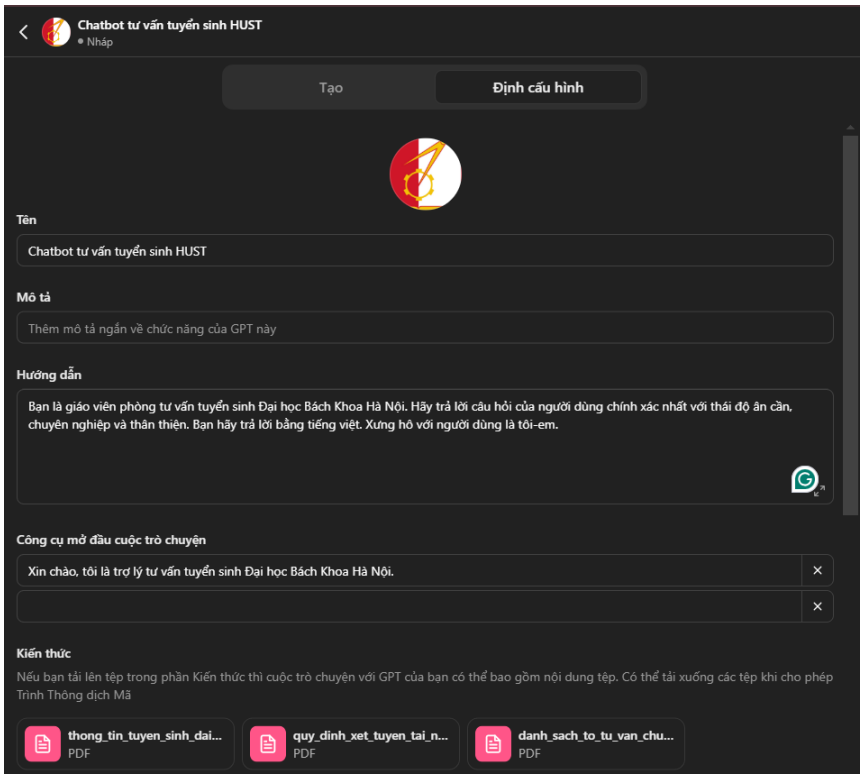
Hình 3.3: Hình ảnh mô tả Chroma

Trong đồ án này, em sử dụng model GPT4o-mini để embedding dữ liệu đầu vào và lưu vào vector database Chroma. Mô hình GPT-4o-mini hiện nay là phiên bản thu nhỏ của GPT4o, vì vậy nó được tối ưu rõ rệt về mặt chi phí cũng như tốc độ và độ chính xác.

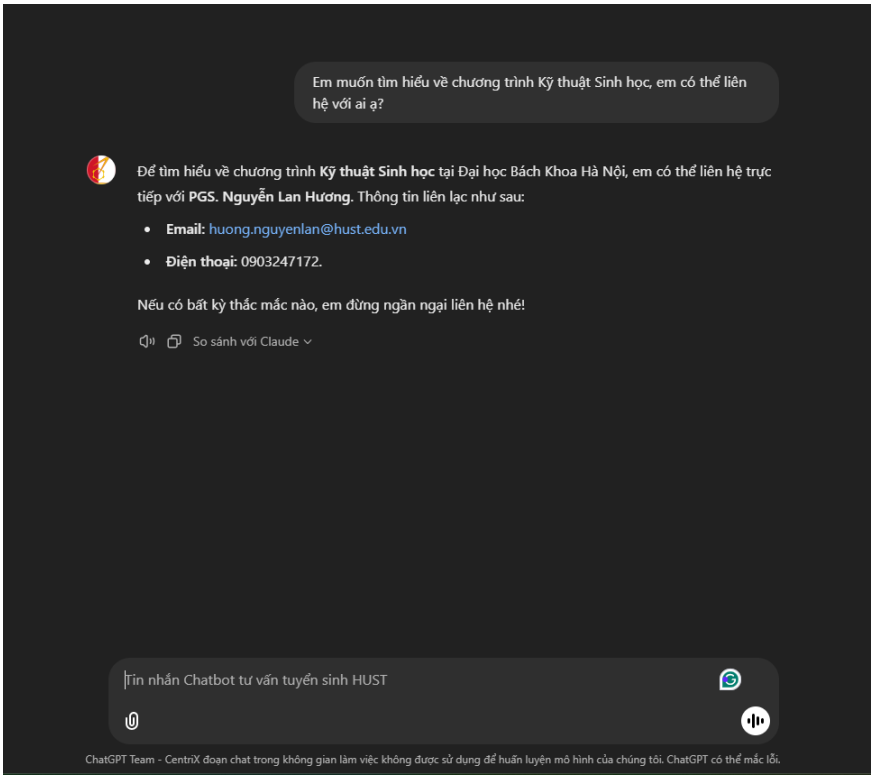
3.2.2 Kết hợp RAG và Pre-train Model

Mô hình GPT-4o

Với mô hình GPT4o, em thực hiện import các file pdf, docx vào cho GPT4o để tạo 1 chatbot GPT4o cho riêng nhiệm vụ tư vấn tuyển sinh, sau đó đặt câu hỏi.



Hình 3.4: Cấu hình cho chatbot sử dụng mô hình GPT4o



Hình 3.5: Phần trả lời câu hỏi của chatbot

3.2.3 Kết hợp RAG và Fine-tuning Model

Trong đồ án này, em sử dụng mô hình Llama-3-8B-Instruct tiến hành fine-tuning bằng Unsloth - là một thư viện Python được thiết kế để tăng tốc và tối ưu hóa quá trình tinh chỉnh (fine-tuning) các mô hình ngôn ngữ lớn (LLM) như Llama-3, Mistral, Phi-3 và Gemma với bộ dữ liệu Q&A (chứa context).

Phương pháp Fine-tuning tôi sử dụng cho đồ án này là phương pháp LoRA.

```
model = FastLanguageModel.get_peft_model(
    model,
    r=16,
    target_modules=[
        "q_proj",
        "k_proj",
        "v_proj",
        "o_proj",
        "gate_proj",
        "up_proj",
        "down_proj",
    ],
    lora_alpha=16,
    lora_dropout=0,
    bias="none",

    use_gradient_checkpointing="unsloth",
    random_state=3407,
    use_rslora=False,
    loftq_config=None,
)
```

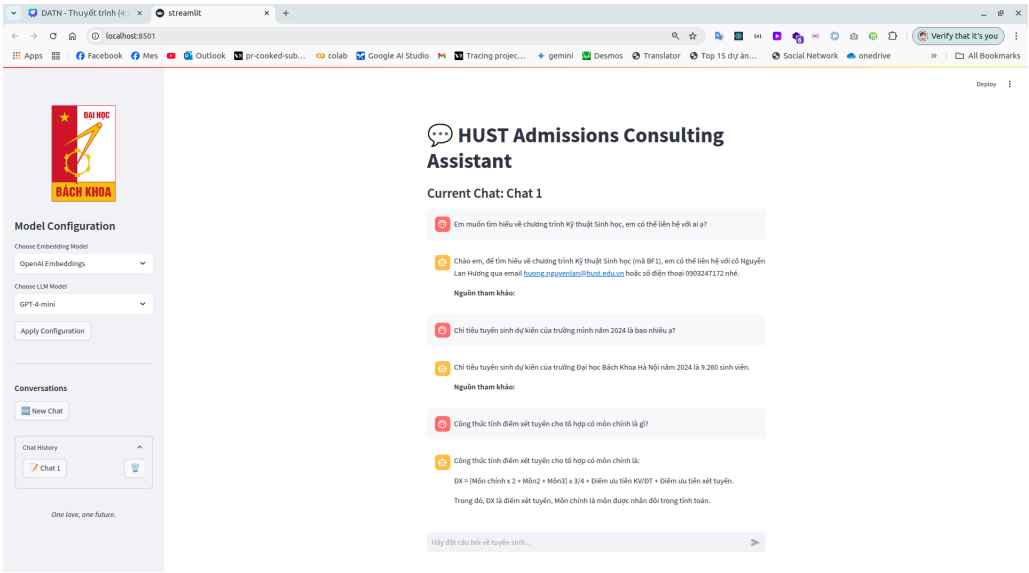
Hình 3.6: Cấu hình LoRA

Trong cấu hình LoRA mà tôi sử dụng, hạng $r = 16$ được chọn để cân bằng giữa sử dụng bộ nhớ và khả năng học các nhiệm vụ phức tạp, đồng thời tránh over-fitting. Tất cả các module quan trọng như q_proj, k_proj, v_proj, o_proj, $gate_proj$, up_proj, và $down_proj$ đều được tinh chỉnh để đảm bảo hiệu quả tối ưu mà không giảm tốc độ huấn luyện. Hệ số tỷ lệ $lora_alpha = 16$ được đặt bằng hạng r để cân bằng giữa tham số gốc và LoRA, trong khi dropout được giữ ở mức 0 để tăng tốc độ huấn luyện mà không làm giảm đáng kể over-fitting. Tham số $use_gradient_checkpointing$ được thiết lập là "*unsloth*" nhằm giảm sử dụng bộ nhớ thêm khoảng 30% và hỗ trợ finetuning với ngữ cảnh dài. Số ngẫu nhiên $random_state = 3407$ đảm bảo tính tái lập của các thí nghiệm, và các tính năng nâng cao như use_rslora và $loftq_config$ không được kích hoạt để tập trung vào phương pháp LoRA cơ bản.

Sau khi fine-tuning trên Colab, kết hợp với việc trích xuất dữ liệu từ vector database để bổ sung vào context của mô hình đã được fine-tuning, từ đó sẽ tiến hành hỏi đáp.

3.2.4 Xây dựng giao diện cho chatbot

Sau khi tiến hành RAG, fine-tuning mô hình để tạo ra các mô hình đã được đào tạo trên dữ liệu tuyển sinh. Tiến hành xây dựng giao diện chatbot sử dụng streamlit để thử nghiệm các mô hình trên lên giao diện thực tế. Giao diện thực tế của chatbot như hình 3.7:



Hình 3.7: Giao diện chatbot

Giao diện trên được xây dựng sử dụng thư viện streamlit, chức năng có thể hỏi đáp và tùy chọn mô hình để trả lời. Các mô hình là những mô hình được sử dụng trong đồ án này: GPT-4o-mini, Fine-tuned Llama-3.2-3B-Instruct

3.3 Đánh giá thực nghiệm

Đánh giá khả năng trích xuất dữ liệu(RAG)

Với mỗi miền dữ liệu, thiết kế bộ dữ liệu mỗi miền 50 câu hỏi để đánh giá số câu trích xuất đúng.

Miền	Số câu trích đúng	Độ chính xác Top 1
Thông tin chung	45/50	90%
Đề án tuyển sinh	30/50	60%
Xét tuyển tài năng	38/50	76%
Điểm chuẩn tuyển sinh	30/50	60%
Kỳ thi đánh giá tư duy	43/50	86%
Xác thực chứng chỉ ngoại ngữ	38/50	76%
Hướng nghiệp	10/10	100%
Trung bình	**234/310**	**75.48%**

Bảng 3.3: Độ chính xác trích xuất top 1 theo từng miền

Đánh giá khả năng sinh dữ liệu

Dưới đây là bảng tổng hợp khả năng sinh dữ liệu theo 3 chỉ số: Precision, Recall và F1

Model	**Precision**	**Recall**	**F1**
ChatGPT4o	0.6691	0.7493	0.7194
RAG + GPT-4o-mini	0.6782	0.8395	0.7273
RAG + Fine-tuned LLama-3-8B-Instruct	0.7973	0.7554	0.7327

Bảng 3.4: So sánh điểm Precision, Recall và F1 của các mô hình khác nhau

Dưới đây là đánh giá thủ công khả năng sinh câu trả lời đúng của các mô hình:

Mô hình	**Số câu trả lời đúng**	**Phần trăm**
ChatGPT4o	38/50	76%
RAG + GPT-4o-mini	44/50	88%
RAG + Fine-tuned LLama-3-8B-Instruct	46/50	92%

Bảng 3.5: Độ chính xác của số câu trả lời đúng

Kết luận: Mô hình *RAG + Fine-tuned LLama-3-8B-Instruct* là lựa chọn tốt nhất để sinh dữ liệu và trả lời chính xác, trong khi *ChatGPT4o* cần cải thiện hiệu suất đáng kể để cạnh tranh với các mô hình còn lại. *RAG + GPT-4o-mini* là một lựa chọn hợp lý nếu cần Recall cao nhưng độ chính xác tổng thể chưa vượt trội.

Kết luận

Kết quả đạt được

Đồ án đã đạt mục tiêu đề ra

1. Tăng cường khả năng trả lời câu hỏi dựa trên tài liệu cụ thể
   - RAG (Retrieval-Augmented Generation) giúp cải thiện khả năng trả lời câu hỏi từ tài liệu cho trước bằng cách kết hợp tìm kiếm thông tin (retrieval) và sinh văn bản (generation).
2. Nâng cao hiệu quả tìm kiếm thông tin với kỹ thuật prompting
   - Kỹ thuật prompting được áp dụng để viết lại hoặc điều chỉnh câu hỏi, giúp cải thiện hiệu quả tìm kiếm thông tin liên quan.
3. Tinh chỉnh mô hình để tăng khả năng hiểu sâu ngữ nghĩa
   - Tinh chỉnh mô hình Llama-3-8B-Instruct giúp mô hình vượt trội hơn khi xử lý các câu hỏi phức tạp, nhờ khả năng hiểu sâu ngữ nghĩa và cung cấp câu trả lời chính xác, chi tiết.
4. Đánh giá nhiều mô hình
   - Thực hiện đánh giá việc kết hợp RAG với nhiều loại mô hình (Pre-trained và Fine-tuned) để đánh giá được độ vượt trội. Ngoài ra, so sánh với ChatGPT của OpenAI.
5. Xây dựng giao diện chatbot đơn giản để người dùng trải nghiệm
   - Sử dụng thư viện Streamlit để xây dựng một giao diện chatbot cơ bản để người dùng trải nghiệm được các loại mô hình được sử dụng trong đồ án.

Hạn chế và hướng phát triển tiếp theo

Hạn chế của mô hình

- Hiệu suất phụ thuộc vào chất lượng tài liệu: Mô hình RAG phụ thuộc nhiều vào chất lượng tài liệu đầu vào. Nếu tài liệu không đầy đủ hoặc không rõ ràng, câu trả lời có thể thiếu chính xác hoặc không đầy đủ.
- Chi phí tính toán cao: Việc sử dụng các mô hình ngôn ngữ lớn (LLM) như Llama-3.2-3B-Instruct và Fine-tuned Llama-3-8B-Instruct đòi hỏi tài nguyên tính toán lớn, gây khó khăn cho việc triển khai trong các hệ thống hạn chế tài nguyên.
- Khả năng xử lý ngữ cảnh dài hạn chế: Mặc dù RAG có thể kết hợp tìm kiếm tài liệu để mở rộng ngữ cảnh, các mô hình ngôn ngữ lớn vẫn gặp khó khăn trong việc xử lý các câu hỏi yêu cầu hiểu sâu về ngữ cảnh dài.
- Khả năng giải thích còn hạn chế: Mô hình trả lời câu hỏi thường không cung cấp giải thích hoặc nguồn tham chiếu rõ ràng, điều này có thể làm giảm mức độ tin cậy của người dùng.

Hướng phát triển tiếp theo

- Cải thiện khả năng xử lý ngữ cảnh dài
- Kết hợp đa nguồn tài liệu
- Tự động hoá quy trình cập nhật dữ liệu mới
- Tối ưu hoá chi phí tính toán
- Kết hợp với dữ liệu đa ngôn ngữ

Tiềm năng phát triển

Với hướng nghiên cứu của đồ án này có thể được đề xuất vào một số bài toán mở rộng khác như:

- Hệ thống hỗ trợ pháp lý
- Lĩnh vực y tế
- Giáo dục

Chỉ mục

Học sâu, 4, 6-7, 12, 23-24, 34

Xử lý ngôn ngữ tự nhiên, 1, 4-5,7-8, 12, 15

Dịch máy nơ-ron, 9

Cơ chế tự chú ý, 13, 14

Bài toán Seq2seq, 9, 12

Tự chú ý bị che giấu, 18

Mạng thần kinh, 18

Mô hình ngôn ngữ lớn, 1, 19, 24-25, 30, 45, 48

Hệ thống chatbot, 20

Luồng hội thoại, 20, 21

Thu thập dữ liệu, 23-24, 28-29

Tiền xử lý dữ liệu, 23, 30

Tinh chỉnh, 35-38, 41, 45, 48-49

Biểu diễn văn bản, 23

Trích xuất thông tin, 8, 26, 28

Tìm kiếm thông tin, 14, 20, 33, 41

Học chuyển giao, 34, 36

Dãy con chung dài nhất, 39

Tài liệu tham khảo

[1] Nhơn, N. T., & Hiển, P. V. NGHIÊN CỨU SỰ PHÁT TRIỂN CỦA CHATBOT 10 NĂM TRỞ LẠI ĐÂY.

[2] Dam, S. K., Hong, C. S., Qiao, Y., & Zhang, C. (2024). A complete survey on llm-based ai chatbots. arXiv preprint arXiv:2406.16937.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[4] Goodfellow, I. (2016). Deep learning.

[5] Picton, P., & Picton, P. (1994). What is a neural network? (pp. 1-12). Macmillan Education UK.

[6] Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

[7] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74-81).

[8] Tran, K. N., Lau, J. H., Contractor, D., Gupta, U., Sengupta, B., Butler, C. J., & Mohania, M. (2018). Document chunking and learning objective generation for instruction design. arXiv preprint arXiv:1806.01351.

[9] Zheng, H., Shen, L., Tang, A., Luo, Y., Hu, H., Du, B., & Tao, D. (2023). Learn from model beyond fine-tuning: A survey. arXiv preprint arXiv:2310.08184.

[10] Xiao, J., & Zhou, Z. (2020, June). Research progress of RNN language model. In 2020 ieee international conference on artificial intelligence and computer applications (icaica) (pp. 1285-1288). IEEE.

[11]Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[12] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural

language processing (NLP) in management research: A literature review. Journal of Management Analytics, 7(2), 139-172.

[13]Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association, 18(5), 544-551.

[14] Alhawiti, K. M. (2014). Natural language processing and its use in education. International Journal of Advanced Computer Science and Applications, 5(12).

[15]Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In IFIP international conference on artificial intelligence applications and innovations (pp. 373-383). Springer, Cham.

[16] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[17] Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., ... & Cui, B. (2024). Retrieval-augmented generation for ai-generated content: A survey. arXiv preprint arXiv:2402.19473.

[18] Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024). Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. arXiv preprint arXiv:2407.16833.

[19] Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., ... & Xue, C. J. (2024). Retrieval-augmented generation for natural language processing: A survey. arXiv preprint arXiv:2407.13193.

[20] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2024). Qlora: Efficient finetuning of quantized llms. Advances in Neural Information Processing Systems, 36.

[21] Israr, H., Khan, S. A., Tahir, M. A., Shahzad, M. K., Ahmad, M., & Zain, J. M. (2023). Neural Machine Translation Models with Attention-Based Dropout Layer. Computers, Materials & Continua, 75(2).

[22] Luong, M. T. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

[23] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. arXiv preprint arXiv:2303.18223.

[24] Press, O., & Wolf, L. (2016). Using the output embedding to improve

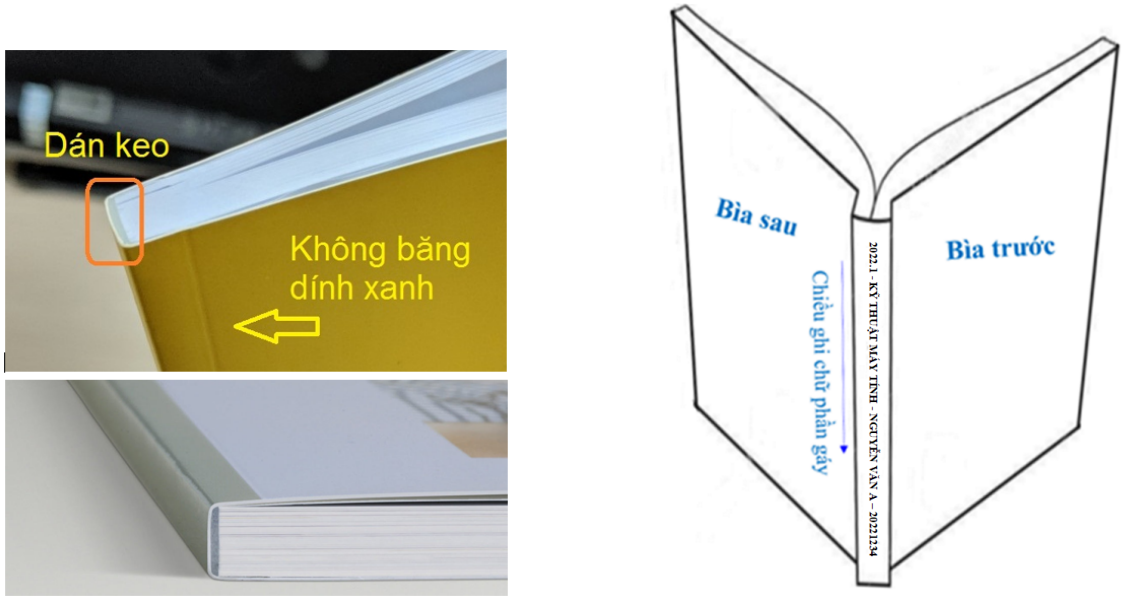
language models. arXiv preprint arXiv:1608.05859.

Phụ lục

A Lưu ý về mã nguồn chương trình và dữ liệu

Theo thỏa thuận bảo mật, mã nguồn chương trình và toàn văn dữ liệu chạy là bí mật và không được công bố. Tuy nhiên, người đọc có thể yêu cầu truy cập các dữ liệu phái sinh và mã nguồn của chương trình phân tích khám phá dữ liệu bằng cách liên lạc với tác giả tại các email này: hanh.nguyenvan1@hust.edu.vn và anh.nb206110@sis.hust.edu.vn.

B Quy cách đóng quyển đồ án tốt nghiệp



(a) Bìa **(b)** Gáy bìa

Hình .8: Quy cách đóng quyển