|  |  |
| --- | --- |
|  |  |
| **VŨ MINH ĐỨC** | **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
| **HAUI**  **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **NGÀNH KHOA HỌC MÁY TÍNH** |
|  |
| **NGHIÊN CỨU CÁC KỸ THUẬT TINH CHỈNH NHƯ PEFT, INSTRUCTION TUNING, RLHF ĐỂ TINH CHỈNH MÔ HÌNH NGÔN NGỮ LỚN LLM** |
|  |
| **KHOA HỌC MÁY TÍNH** | **CBHD : *TS. Đỗ Mạnh Hùng*** |
| **Sinh viên : Vũ Minh Đức**  **Mã số sinh viên : *2021606758*** |
|  |
| Hà Nội – Năm 2024 |
|  |

# LỜI CAM ĐOAN

Tôi xin cam đoan rằng đề án này là kết quả của quá trình nghiên cứu nghiêm túc và là công trình nghiên cứu của riêng tôi. Toàn bộ nội dung được trình bày trong đề án đều đảm bảo tính trung thực và chính xác, phản ánh đúng các kết quả mà tôi đã thực hiện trong quá trình làm việc.

Quá trình nghiên cứu và xây dựng nội dung đề án được thực hiện dưới sự hướng dẫn tận tình của TS. Đỗ Mạnh Hùng và TS. Trần Hùng Cường. Tôi đã dành nhiều thời gian tìm hiểu, nghiên cứu và trình bày các nội dung này với sự cẩn trọng và nghiêm túc cao nhất.

Các số liệu, bảng biểu, và thông tin phục vụ cho việc phân tích, minh họa trong đề án được tôi thu thập từ nhiều nguồn tài liệu đáng tin cậy. Tất cả các tài liệu tham khảo được ghi chú rõ ràng trong mục tài liệu tham khảo hoặc được chú thích ngay bên dưới các bảng biểu, hình ảnh để đảm bảo tính minh bạch.

Ngoài ra, các tài liệu bổ sung để diễn giải, làm rõ thêm các luận điểm phân tích trong đề án cũng được tôi trích dẫn nguồn gốc đầy đủ. Những tài liệu này được sắp xếp và trình bày trong phần phụ lục kèm theo.

Tôi hoàn toàn chịu trách nhiệm về tính trung thực và chính xác của các nội dung trong đề án này. Nếu có bất kỳ sai sót nào được phát hiện, tôi xin chịu hoàn toàn trách nhiệm trước cơ quan quản lý và hội đồng đánh giá.

Tôi xin chân thành cảm ơn.

*Hà Nội, ngày 27 tháng 12 năm 2024*

Sinh viên thực hiện

Đức

Vũ Minh Đức

# LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn sâu sắc tới TS. Đỗ Mạnh Hùng và TS. Trần Hùng Cường, người đã tin tưởng và tạo điều kiện để em có cơ hội lựa chọn, nghiên cứu đề tài “Nghiên cứu các kỹ thuật finetune để tinh chỉnh mô hình ngôn ngữ lớn”. Đề tài này không chỉ là một thử thách lớn đối với em mà còn là một hành trình khám phá đầy ý nghĩa, mang lại cho em nhiều trải nghiệm quý báu và những kiến thức vô cùng bổ ích trong lĩnh vực trí tuệ nhân tạo.

Trong suốt quá trình thực hiện đề án, em đã nhận được sự hướng dẫn tận tình và hỗ trợ đầy trách nhiệm từ các thầy. Những kiến thức chuyên môn sâu rộng, kinh nghiệm thực tiễn quý giá cùng những lời khuyên chân thành từ TS. Đỗ Mạnh Hùng và TS. Trần Hùng Cường đã giúp em vượt qua những khó khăn và hoàn thành tốt đề tài. Em cảm thấy vô cùng may mắn khi được làm việc dưới sự chỉ bảo của các thầy, những người không chỉ là những người hướng dẫn mà còn là nguồn cảm hứng to lớn để em tiếp tục theo đuổi đam mê nghiên cứu khoa học.

Bên cạnh đó, em cũng xin gửi lời cảm ơn chân thành đến tập thể giáo viên của Trường Đại học Công Nghiệp Hà Nội. Chính những kiến thức nền tảng mà các thầy cô đã truyền đạt qua các môn học và những lời động viên, khích lệ trong suốt quá trình học tập đã trở thành động lực để em thực hiện đề tài này.

Em cũng xin bày tỏ lòng biết ơn sâu sắc tới các học viên, bạn bè và đồng nghiệp, những người đã không ngần ngại chia sẻ ý kiến, góp ý và động viên em trong suốt hành trình nghiên cứu và triển khai đề án. Sự hỗ trợ từ mọi người đã giúp em thêm vững tin và nỗ lực để hoàn thiện công việc một cách tốt nhất.

Đề tài này không chỉ giúp em nâng cao kiến thức chuyên môn mà còn rèn luyện các kỹ năng nghiên cứu khoa học, kỹ năng phân tích và giải quyết vấn đề. Quan trọng hơn, em đã có cơ hội áp dụng các kiến thức học thuật vào thực tế, từ đó phát triển khả năng thực hành và tư duy sáng tạo. Những kinh nghiệm thu được trong quá trình thực hiện đề án chắc chắn sẽ trở thành hành trang quý giá, giúp em tự tin hơn trong công việc và các dự án nghiên cứu sau này.

Một lần nữa, em xin được gửi lời cảm ơn chân thành tới TS. Đỗ Mạnh Hùng và TS. Trần Hùng Cường, cùng toàn thể các thầy cô giáo và những người đã đồng hành, hỗ trợ em trong suốt quá trình thực hiện đề án này.

Trân trọng!

**MỤC LỤC**

[LỜI CAM ĐOAN i](#_Toc185927270)

[LỜI CẢM ƠN ii](#_Toc185927271)

[DANH MỤC TỪ VIẾT TẮT vii](#_Toc185927272)

[DANH MỤC CÁC BẢNG ix](#_Toc185927273)

[DANH MỤC HÌNH ẢNH x](#_Toc185927274)

[MỞ ĐẦU 1](#_Toc185927275)

[LÝ DO CHỌN ĐỀ TÀI 1](#_Toc185927276)

[MỤC TIÊU NGHIÊN CỨU 3](#_Toc185927277)

[PHẠM VI ĐỀ TÀI 4](#_Toc185927278)

[CHƯƠNG 1: TỔNG QUAN BÀI TOÁN 5](#_Toc185927279)

[1.1. Các nghiên cứu gần đây 5](#_Toc185927280)

[1.1.1 Tình hình nghiên cứu trong nước 5](#_Toc185927281)

[1.1.2 Tình hình nghiên cứu ở nước ngoài 6](#_Toc185927282)

[1.2. Bài toán xây dựng chatbot cho trường Đại học Công nghiệp Hà Nội 6](#_Toc185927283)

[1.2.1 Nhu cầu tra cứu thông tin về trường Đại học Công nghiệp Hà Nội 6](#_Toc185927284)

[1.2.2 Thực trạng các ứng dụng đang được sử dụng để tra cứu thông tin về trường Đại học Công Nghiệp Hà Nội 7](#_Toc185927285)

[1.3. Chatbot và ứng dụng của chatbot 8](#_Toc185927286)

[1.3.1 Khái niệm chatbot 8](#_Toc185927287)

[1.3.2 Các loại chatbot 9](#_Toc185927288)

[1.3.3 Ứng dụng của chatbot 9](#_Toc185927289)

[1.3.4 Lợi ích của chatbot 10](#_Toc185927290)

[1.3.5 Thách thức trong việc triển khai Chatbot 11](#_Toc185927291)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 12](#_Toc185927292)

[2.1. Mô hình ngôn ngữ lớn 12](#_Toc185927293)

[2.1.1 Khái niệm mô hình ngôn ngữ lớn 12](#_Toc185927294)

[2.1.2 Lịch sử hình thành 12](#_Toc185927295)

[2.1.3 Word Embedding 13](#_Toc185927296)

[2.1.4 Mạng Nơ-ron hồi tiếp truyền thống RNN 15](#_Toc185927297)

[2.1.5 Mạng Nơ-ron hồi tiếp hiện đại GRU và LSTM 18](#_Toc185927298)

[2.1.6 Kiến trúc mạng Transformer 24](#_Toc185927299)

[2.1.7 Mô hình BERT 29](#_Toc185927300)

[2.1.8 Mô hình GPT 31](#_Toc185927301)

[2.1.9 Mixture of Experts 32](#_Toc185927302)

[2.2. Các chiến lược tinh chỉnh mô hình ngôn ngữ lớn 33](#_Toc185927303)

[2.2.1 Pre-trained Language Model (PLM) 33](#_Toc185927304)

[2.2.2 Khái niệm tinh chỉnh 34](#_Toc185927305)

[2.2.3 Các dạng tinh chỉnh LLM 34](#_Toc185927306)

[2.3. Tinh chỉnh với tham số hiệu quả (PEFT) 35](#_Toc185927307)

[2.3.1 Khái niệm tinh chỉnh với tham số hiệu quả 35](#_Toc185927308)

[2.3.2 Adapters 36](#_Toc185927309)

[2.3.3 LoRA 37](#_Toc185927310)

[2.3.4 QLoRA 39](#_Toc185927311)

[2.3.5 DoRA 40](#_Toc185927312)

[2.3.6 Proximal Policy Optimization 43](#_Toc185927313)

[2.3.7 Direct Preference Optimization 44](#_Toc185927314)

[2.4. Retrieval Augmented Generation 45](#_Toc185927315)

[2.4.1 Khái niệm RAG 45](#_Toc185927316)

[2.4.2 Ứng dụng 45](#_Toc185927317)

[2.4.3 Các mô hình RAG 46](#_Toc185927318)

[CHƯƠNG 3: XÂY DỰNG HỆ THỐNG CHATBOT 50](#_Toc185927319)

[3.1. Tổng quan ý tưởng hệ thống 50](#_Toc185927320)

[3.2. Chuẩn bị dữ liệu 50](#_Toc185927321)

[3.2.1 Giới thiệu bộ dữ liệu 50](#_Toc185927322)

[3.2.2 Biến đổi dữ liệu 51](#_Toc185927323)

[3.3. Xây dựng back-end 52](#_Toc185927324)

[3.3.1 Xây dựng vector database 53](#_Toc185927325)

[3.3.2 Xây dựng Retriever 56](#_Toc185927326)

[3.4. Lựa chọn mô hình ngôn ngữ lớn 56](#_Toc185927327)

[3.4.1 Lượng tử hóa mô hình 57](#_Toc185927328)

[3.4.2 Huấn luyện mô hình 58](#_Toc185927329)

[3.4.3 Triển khai mô hình 58](#_Toc185927330)

[3.5. Xây dựng front-end 59](#_Toc185927331)

[CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 60](#_Toc185927332)

[4.1. Phương pháp đánh giá 60](#_Toc185927333)

[4.2. Kết quả thực nghiệm 60](#_Toc185927334)

[4.3. Đánh giá kết quả 62](#_Toc185927335)

[KẾT LUẬN 63](#_Toc185927336)

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| SGD | Stochastic Gradient Descent | Giảm độ dốc ngẫu nhiên |
| RNN | Recurrent Neural Network | Mạng nơ-ron hồi quy |
| CBOW | Continuous Bag-of-Words | Túi từ liên tục |
| DL | Deep Learning | Học sâu |
| CPU | Central Processing Unit | Bộ xử lý trung tâm |
| GPU | Graphics Processing Unit | Bộ xử lý đồ họa |
| LLM/LLMs | Large Language Model | Mô hình ngôn ngữ lớn |
| LSTM | Long Shot-Term Memory | Mạng bộ nhớ ngắn hạn dài hạn |
| GRU | Gated Recurrent Units | Nút hồi tiếp có cổng |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| CNN | Convolutional Neural Networks | Mạng nơ-ron tích chập |
| UI | User Interface | Giao diện người dùng |
| RAG | Retrieval-Augmented Generation | Tạo tăng cường truy xuất |
| LoRA | Low-Rank Adaptation | Thích ứng cấp thấp |
| QLoRA | Quantized Low-Rank Adaptation | Thích ứng cấp thấp với lượng tử hóa |
| DoRA | Weight-Decomposed Low-Rank Adaptation | Thích ứng trọng số phân rã cấp thấp |
| PEFT | Parameter-Efficient Fine-Tune | Tinh chỉnh hiệu quả tham số |
| RLHF | Reinforcement Learning from Human Feedback | Học tăng cường từ phản hồi người dùng |
| FT | Fine-Tune/ Fine-Tuning | Tinh chỉnh |
| SFT | Supervised Fine-Tuning | Tinh chỉnh có giám sát |
| HFT | Haft Fine-Tuning | Tinh chỉnh một nửa |
| FFT | Full Fine-Tuning | Tinh chỉnh đầy đủ |
| PPO | Proximal Policy Optimisation | Tối ưu hóa chính sách gần đúng |
| DPO | Direct Preference Optimisation | Tối ưu hóa sở thích trực tiếp |
| MoE | Mixture of Expert | Kết hợp các chuyên gia |
| MoA | Mixture of Agent | Kết hợp các tác nhân |
| BERT | Bidirectional Encoder Representations from Transformers | Biểu diễn mã hóa hai chiều từ Transformers |
| GPT | Generative Pre-trained Transformer | Transformer sinh tạo được huấn luyện trước |
| BPTT | Backpropagation through time | Lan truyền ngược theo thời gian |

# DANH MỤC CÁC BẢNG

[Bảng 2.1 Bảng so sánh LoRA và DoRA 43](#_Toc185240791)

# DANH MỤC HÌNH ẢNH

[Hình 2.1 Lược sử các mô hình từ năm 1920 đến nay 13](#_Toc185260501)

[Hình 2.2 Minh họa Skip-gram dưới dạng mạng nơ-ron 14](#_Toc185260502)

[Hình 2.3 Minh họa CBOW dưới dạng mạng nơ-ron 15](#_Toc185260503)

[Hình 2.4 Minh họa khác biệt giữa mạng truyền thẳng và mạng hồi quy 17](#_Toc185260504)

[Hình 2.5 Cổng xóa và cổng cập nhật trong GRU 19](#_Toc185260505)

[Hình 2.6 Tính toán của trạng thái tiềm ẩn trong GRU 20](#_Toc185260506)

[Hình 2.7 Tính toán trạng thái ẩn trong GRU 20](#_Toc185260507)

[Hình 2.8 Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra tại một đơn vị LSTM 22](#_Toc185260508)

[Hình 2.9 Các phép tính toán trong ô nhớ tiềm năng LSTM 23](#_Toc185260509)

[Hình 2.10 Các phép tính toán trong ô nhớ của LSTM 23](#_Toc185260510)

[Hình 2.11 Các phép tính của trạng thái ẩn 24](#_Toc185260511)

[Hình 2.12 Minh họa cơ chế Self-Attention 25](#_Toc185260512)

[Hình 2.13 Minh họa Q, K, V và ứng dụng của chúng trong Self-Attention 25](#_Toc185260513)

[Hình 2.14 Minh họa Masked Self-Attention 26](#_Toc185260514)

[Hình 2.15 Minh họa Multi-head Attention 27](#_Toc185260515)

[Hình 2.16 Minh họa kiến trúc Transformer 28](#_Toc185260516)

[Hình 2.17 Minh họa kiến trúc mô hình BERT 30](#_Toc185260517)

[Hình 2.18 Minh họa kiến trúc mô hình GPT 32](#_Toc185260518)

[Hình 2.19 Lược đồ kiến trúc mô hình Mixtral 8x7B 33](#_Toc185260519)

[Hình 2.20 Tổng quan các phương pháp PEFT 36](#_Toc185260520)

[Hình 2.21 Sơ đồ biểu diễn kiến trúc Adapter được sử dụng trong LLM 37](#_Toc185260521)

[Hình 2.22 Sơ đồ cập nhật trọng số trong LoRA 37](#_Toc185260522)

[Hình 2.23 Quy trình tối ưu hóa QLoRA 40](#_Toc185260523)

[Hình 2.24 Quá trình cập nhật trọng số DoRA 41](#_Toc185260524)

[Hình 2.25 Sơ đồ PPO 44](#_Toc185260525)

[Hình 2.26 Luồng xử lý Direct Preference Optimization 45](#_Toc185260526)

[Hình 2.27 Tổng quan kiến trúc RAG 46](#_Toc185260527)

[Hình 2.28 Các mô hình RAG 46](#_Toc185260528)

[Hình 3.1 Chia tập dữ liệu 51](#_Toc185260529)

[Hình 3.2 Thông tin chung về bộ dữ liệu vilm/OpenOcra-Viet 51](#_Toc185260530)

[Hình 3.3 Định dạng dữ liệu 52](#_Toc185260531)

[Hình 3.4 Dữ liệu sau khi định dạng 52](#_Toc185260532)

[Hình 3.5 Xây dựng back-end 53](#_Toc185260533)

[Hình 3.6 Import các thư viện cần dùng 54](#_Toc185260534)

[Hình 3.7 Load và chia nhỏ file dữ liệu 54](#_Toc185260535)

[Hình 3.8 Tạo embedding và lưu lại trên vector database server 55](#_Toc185260536)

[Hình 3.9 Tạo các Retriever 56](#_Toc185260537)

[Hình 3.10 Khởi tạo mô hình 57](#_Toc185260538)

[Hình 3.11 Hàm tính số lượng tham số có thể huấn luyện được 57](#_Toc185260539)

[Hình 3.12 Thiết lập cấu hình LoRA 58](#_Toc185260540)

[Hình 3.13 Thiết lập các siêu tham số 58](#_Toc185260541)

[Hình 3.14 Lưu trữ mô hình lên HuggingFace Hub 59](#_Toc185260542)

[Hình 3.15 Giao diện hệ thống 59](#_Toc185260543)

[Hình 4.1 Các metric đánh giá mô hình 60](#_Toc185260544)

[Hình 4.2 Sơ đồ phân bố của Precision 61](#_Toc185260545)

[Hình 4.3 Sơ đồ phân bố của Recall 61](#_Toc185260546)

[Hình 4.4 Sơ đồ phân bố F1 Score 62](#_Toc185260547)

# MỞ ĐẦU

# LÝ DO CHỌN ĐỀ TÀI

Trong thời đại công nghệ hiện nay, mô hình ngôn ngữ lớn (Large Language Models - LLMs) đang đóng vai trò quan trọng trong các ứng dụng trí tuệ nhân tạo, đặc biệt là trong xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Các mô hình như GPT, BERT, hay T5 đã đạt được những bước tiến vượt bậc trong việc hiểu và sinh ngữ liệu tự nhiên, cung cấp nền tảng mạnh mẽ cho nhiều lĩnh vực như dịch máy, tóm tắt văn bản, và chatbot. Tuy nhiên, để áp dụng LLMs hiệu quả vào các bài toán thực tế, việc fine-tuning (tinh chỉnh) là một yếu tố then chốt nhằm tối ưu hóa hiệu suất cho các tác vụ cụ thể.

Mặc dù các mô hình ngôn ngữ lớn đã được huấn luyện trên kho dữ liệu khổng lồ, khả năng ứng dụng của chúng không phải lúc nào cũng đạt hiệu quả cao trên các tác vụ đặc thù. Điều này đòi hỏi phải áp dụng các kỹ thuật fine-tuning để điều chỉnh trọng số của mô hình, giúp nó phù hợp với đặc trưng của dữ liệu và mục tiêu cụ thể. Nghiên cứu các kỹ thuật fine-tuning không chỉ mang lại kiến thức sâu hơn về cách sử dụng mô hình ngôn ngữ lớn mà còn giải quyết được các vấn đề thực tiễn như tối ưu hóa tài nguyên tính toán, tăng cường tính linh hoạt cũng như nâng cao hiệu suất và tính cá nhân hóa.

Bên cạnh đó, nghiên cứu về các kỹ thuật fine-tuning hiện đại như LoRA (Low-Rank Adaptation), Prompt Tuning, hay Adapter Modules đã mở ra những cách tiếp cận mới, giúp khai thác hiệu quả hơn sức mạnh của LLMs. Những kỹ thuật này không chỉ cải thiện chất lượng đầu ra mà còn tối ưu về tài nguyên, đáp ứng nhu cầu phát triển các hệ thống AI nhanh chóng và bền vững.

Với tầm quan trọng ngày càng tăng của LLMs và nhu cầu tối ưu hóa chúng trong thực tế, đề tài nghiên cứu này sẽ đóng góp vào việc hiểu rõ hơn các kỹ thuật fine-tuning, cung cấp các hướng dẫn, phương pháp và giải pháp ứng dụng hiệu quả cho bài toán xây dựng chatbot hỏi đáp về trường Đại học Công Nghiệp Hà Nội. Đồng thời, kết quả nghiên cứu sẽ mang lại giá trị thực tiễn cao trong việc phát triển các ứng dụng AI hiện đại, phục vụ nhu cầu của cộng đồng và các tổ chức doanh nghiệp.

# MỤC TIÊU NGHIÊN CỨU

Tìm hiểu các khái niệm cơ bản liên quan đến mô hình ngôn ngữ lớn (LLM) và các kỹ thuật fine-tuning. Đề tài tập trung nghiên cứu lý thuyết về mô hình ngôn ngữ lớn, cơ chế hoạt động của các thành phần cơ bản, đặc biệt là mô hình Transformers, cũng như phân loại các kỹ thuật fine-tuning hiện đại như Full Fine-Tuning, Prompt Tuning, LoRA (Low-Rank Adaptation), và Adapter Modules. Đồng thời, đề tài khảo sát ứng dụng của các mô hình ngôn ngữ lớn trong các lĩnh vực như hỏi đáp, tóm tắt văn bản, dịch máy, và hệ thống chatbot, qua đó đánh giá lợi ích và hạn chế của việc tinh chỉnh mô hình cho các miền dữ liệu cụ thể.

Bên cạnh việc nghiên cứu lý thuyết, đề tài tập trung xây dựng và tinh chỉnh mô hình ngôn ngữ lớn nhằm tối ưu hóa hiệu suất trong các tác vụ cụ thể là việc hỏi đáp và tra cứu thông tin về Trường Đại học Công Nghiệp Hà Nội. Quá trình này bao gồm áp dụng các kỹ thuật fine-tuning để cải thiện hiệu quả trên tập dữ liệu thực tế và đánh giá các kết quả đạt được. Ngoài ra, đề tài cũng đề xuất các giải pháp tối ưu hóa quá trình fine-tuning để giảm thiểu chi phí tính toán, tài nguyên cần thiết, đồng thời nâng cao độ chính xác và khả năng áp dụng thực tiễn của mô hình.

Một trong những mục tiêu quan trọng của đề tài là xây dựng và tích hợp mô hình ngôn ngữ lớn đã được tinh chỉnh vào hệ thống chatbot thông minh. Hệ thống này sẽ hỗ trợ người dùng trong việc tra cứu, hỏi đáp thông tin một cách nhanh chóng, chính xác, và liên tục. Cuối cùng, hiệu năng của hệ thống sẽ được đánh giá dựa trên các tiêu chí như thời gian phản hồi, độ chính xác, và mức độ hài lòng của người dùng, nhằm đảm bảo rằng kết quả nghiên cứu mang lại giá trị thực tiễn cao, đặc biệt trong môi trường giáo dục.

# PHẠM VI ĐỀ TÀI

Phạm vi nghiên cứu của đề tài này tập trung vào việc khám phá và áp dụng các kỹ thuật hiện đại trong quá trình tinh chỉnh mô hình ngôn ngữ lớn (LLM). Đề tài nghiên cứu các khái niệm cơ bản liên quan đến mô hình ngôn ngữ lớn, đặc biệt là mô hình Transformers, cơ chế Self-Attention, và các cải tiến trong kiến trúc này.

Ngoài ra, nghiên cứu cũng khảo sát các kỹ thuật fine-tuning hiện đại, bao gồm Full Fine-Tuning, Adapter Modules, LoRA (Low-Rank Adaptation), và Prompt Tuning.

Phạm vi ứng dụng của đề tài bao gồm việc điều chỉnh mô hình ngôn ngữ lớn để phục vụ các tác vụ xử lý ngôn ngữ tự nhiên (NLP) như hỏi đáp, tra cứu thông tin, và tóm tắt văn bản, với tập dữ liệu cụ thể liên quan đến thông tin và tài liệu của Trường Đại học Công Nghiệp Hà Nội.

Dữ liệu nghiên cứu sẽ được thu thập từ các nguồn học thuật và quản lý của trường, đồng thời sử dụng các công cụ và thư viện mã nguồn mở như PyTorch và Hugging Face Transformers để thực hiện các thí nghiệm.

Tuy nhiên, nghiên cứu không đi sâu vào tối ưu hóa hạ tầng phần cứng mà tập trung chủ yếu vào các thuật toán và kỹ thuật tinh chỉnh mô hình. Các thí nghiệm sẽ chỉ thực hiện trên các mô hình ngôn ngữ lớn đã được huấn luyện trước (pre-trained models), không bao gồm việc huấn luyện từ đầu.

Phạm vi nghiên cứu này giúp đảm bảo đề tài tập trung vào việc áp dụng các kỹ thuật tinh chỉnh để giải quyết các vấn đề cụ thể và mang lại giá trị thực tiễn cao trong lĩnh vực giáo dục và ứng dụng xử lý ngôn ngữ tự nhiên.

# TỔNG QUAN BÀI TOÁN

Chương này sẽ khảo sát các nghiên cứu trong và ngoài nước gần đây về các phương pháp tinh chỉnh mô hình ngôn ngữ lớn. Đồng thời đặt ra vấn đề cho bài toán tinh chỉnh mô hình ngôn ngữ lớn để tạo ra chatbot giúp đưa ra thông tin về trường Đại học Công nghiệp Hà Nội một cách nhanh chóng và hiệu quả.

## Các nghiên cứu gần đây

### Tình hình nghiên cứu trong nước[10]

Trong những năm gần đây, các mô hình ngôn ngữ lớn (LLM) đã thu hút sự chú ý trong cộng đồng nghiên cứu Việt Nam nhờ khả năng xử lý ngôn ngữ tự nhiên (NLP). Mô hình Transformer, dựa trên kiến trúc học sâu, đã cách mạng hóa các nhiệm vụ NLP bằng cách nắm bắt hiệu quả mối quan hệ tầm xa trong văn bản. Tuy nhiên, LLM gặp khó khăn khi làm việc với các nhiệm vụ mới hoặc khi yêu cầu theo hướng dẫn không quen thuộc, làm hạn chế khả năng ứng dụng thực tế.

Để cải thiện, cộng đồng NLP đã nỗ lực đào tạo LLM để tuân theo hướng dẫn tốt hơn và trở nên hữu ích hơn. Các mô hình "điều chỉnh theo hướng dẫn" đã chứng minh khả năng khái quát hóa tốt hơn cho các nhiệm vụ mới, nhưng quá trình này tốn kém và giới hạn về tính đa dạng trong các nhiệm vụ. Phương pháp tự hướng dẫn đã được phát triển để cải thiện khả năng tuân theo hướng dẫn của LLM bằng cách tận dụng chính các thế hệ của mô hình.

Dù có sự thành công của các mô hình như GPT-4 và PaLM, việc tiếp cận chúng vẫn bị hạn chế do chỉ có thể sử dụng qua API. Ở Việt Nam, việc thiếu các mô hình ngôn ngữ được đào tạo chuyên sâu trên tập văn bản lớn dẫn đến sự hạn chế về tính năng và khả năng học tập. Thách thức khác là chi phí tinh chỉnh và triển khai LLM. Phương pháp LoRA (Thích ứng Xếp hạng Thấp) đã được áp dụng để tinh chỉnh LLM trong điều kiện tài nguyên thấp, duy trì hiệu suất và khả năng thích ứng cao.

### Tình hình nghiên cứu ở nước ngoài[11]

Các mô hình ngôn ngữ lớn (LLM) hiện đại có khả năng thực hiện các nhiệm vụ phức tạp nhưng gặp hạn chế về khả năng tự học hỏi và cập nhật kiến thức mới.[ Nghiên cứu của Bubeck và cộng sự (2023) cho thấy GPT-4 vượt trội hơn GPT-3 nhờ được đào tạo với lượng dữ liệu lớn hơn và tinh chỉnh kỹ lưỡng. Một số nghiên cứu khác, như của Zelikman và cộng sự (2022), đề xuất cải thiện LLM bằng cách huấn luyện trên các ví dụ từ cả thành công và thất bại, giúp tăng cường trí thông minh của mô hình, đặc biệt trong lĩnh vực toán học. Tuy nhiên, phương pháp này chỉ hiệu quả khi áp dụng cho các LLM tiên tiến như GPT-3. Một cách tiếp cận khác, Truy xuất thế hệ tăng cường (RAG), kết hợp bộ nhớ tham số của LLM với bộ nhớ không tham số từ cơ sở tri thức, đã chứng minh sự cải thiện trong các nhiệm vụ yêu cầu trả lời câu hỏi phức tạp (Khattab và cộng sự, 2021). Mặc dù các phương pháp này đều cải thiện hiệu suất LLM, nhưng chúng đòi hỏi nhiều dữ liệu và tài nguyên, và khi đã được thiết lập, chúng không thể tự học hỏi thêm theo thời gian.

## Bài toán xây dựng chatbot cho trường Đại học Công nghiệp Hà Nội

### Nhu cầu tra cứu thông tin về trường Đại học Công nghiệp Hà Nội

Trường Đại học Công nghiệp Hà Nội là trường đại học công lập trực thuộc Bộ Công Thương, có truyền thống đào tạo cán bộ khoa học kỹ thuật, cán bộ kinh tế, công nhân kỹ thuật lâu đời nhất Việt Nam. Tiền thân là Trường Chuyên nghiệp Hà Nội thành lập năm 1898 và Trường Chuyên nghiệp Hải Phòng thành lập năm 1913. Đây là một cơ sở đào tạo định hướng ứng dụng nhiều ngành, nhiều loại hình, nhiều cấp trình độ.

Với bề dày nhiều năm thành lập và hoạt động, Đại học Công nghiệp Hà Nội đã và đang là một trong những trường đại học có nhiều sinh viên và nghiên cứu sinh theo học. Từ đó như cầu tìm hiểu thông tin, hợp tác và kết nối,... cũng tăng lên chóng mặt; việc tra cứu thông tin đã chuyển dịch phù hợp với xu thế của sự phát triển của công nghệ thông tin. Việc ứng dụng khoa học công nghệ, đặc biệt là các ứng dụng của trí tuệ nhân tạo (Artificial Intelligence – AI) trở thành xu thế tất nhiều nhằm nâng cao hiệu suất giảm thiểu thời gian tra cứu thông tin của con người cũng như nâng cao uy tín của chính đơn vị sử dụng nó.

Nhu cầu tra cứu thông tin về trường Đại học Công nghiệp Hà Nội là vô cùng cần thiết thể hiện qua các nội dung như:

* Phụ huynh, học sinh tra cứu về thông tin trường trước kỳ thi đại học: Như ta thấy, trước mỗi kỳ thi tuyển sinh đại học, mỗi học sinh và phụ huynh của các em luôn phải tìm hiểu để chọn ra các ngành nghề cũng như trường phù hợp cho các em theo học. Mỗi năm có hàng triệu thí sinh tốt nghiệp trung học phổ thông và tuyển sinh đại học. Mỗi phụ huynh và học sinh đều muốn tìm hiểu về các thông tin cụ thể như loại hình đào tạo, cơ sở vật chất trang thiết bị, đội ngũ giảng viên,.. để có thể đưa ra lựa chọn phù hợp nhất.
* Tuyển dụng nhân lực: Không chỉ là cơ sở giáo dục đào tạo các hệ Cao đẳng, Đại học, Thạc sĩ, Tiến sĩ với nhiều ngành nghề xu hướng trong xã hội. Đại học Công nghiệp Hà Nội còn là đơn vị tin cậy cho việc tuyển dụng nhân lực cho các doanh nghiệp trong và ngoài nước.
* Thông tin tuyển dụng, cơ hội nghiên cứu: Đại học Công nghiệp Hà Nội hàng năm tuyển dụng nhiều các vị trí giảng viên, nhân viên, văn phòng khoa, cùng cơ hội nghiên cứu vô cùng hấp dẫn. Việc tiếp cận được thông tin một cách dễ dàng sẽ giúp không bỏ lỡ cơ hội việc làm cũng như nghiên cứu.

### Thực trạng các ứng dụng đang được sử dụng để tra cứu thông tin về trường Đại học Công Nghiệp Hà Nội

1. **Trang thông tin điện tử**

Nhà trường đã ứng dụng công nghệ web trong xây dựng trang thông tin điện tử về Trường tại địa chỉ “www.haui.edu.vn”. Đây là kênh cung cấp thông tin quan trọng và chính thống cho sinh viên, phụ huynh và các bên liên quan. Trang web được thiết kế để cung cấp đầy đủ các thông tin về tuyển sinh, chương trình đào tạo, lịch học, lịch thi, các thông báo từ nhà trường, và các hoạt động khác.

Tuy vậy việc tra cứu thông tin sẽ không dễ dàng đối với nhiều người dùng, nhất là đối với các tài liệu đã được đăng lên từ lâu. Thêm vào đó việc đọc các thông báo, các quyết định do trường đề ra sẽ gây mất thời gian với nhiều người dùng.

1. **Mạng xã hội**

Với xu thế và sự bùng phát của mạng xã hội, để quảng bá hình ảnh nhà trường và truyền tải thông tin tới nhiều người hơn, nhà trường đã và đang sử dụng các mạng xã hội như Facebook để kết nối chia sẻ thông tin thông qua các hội nhóm, hay các trang fanpage của trường. Các trang chính thức của trường và các nhóm sinh viên sẽ thường xuyên cập nhật các thông báo, sự kiện, và các hoạt động liên quan.

Tuy nhiên thông tin sẽ không chính xác nếu người dùng tham gia vào các nhóm không chính thức hoặc các trang cá nhân cung cấp thông tin giả mạo. Thêm vào đó việc trao đổi thông tin cá nhân qua mạng xã hội có thể tiềm ẩn nguy cơ lộ thông tin người dùng. Hơn nữa việc không đồng bộ với trang chính thức sẽ gây khó khăn trong việc kiểm chứng.

## Chatbot và ứng dụng của chatbot

### Khái niệm chatbot

Chatbot, viết tắt của "chat robot", là một phần mềm ứng dụng trí tuệ nhân tạo (AI) được thiết kế để mô phỏng và xử lý các cuộc trò chuyện của con người thông qua ngôn ngữ tự nhiên. Chatbot có khả năng tương tác với người dùng thông qua văn bản hoặc giọng nói, giúp tự động hóa các nhiệm vụ giao tiếp và cung cấp thông tin một cách hiệu quả. Những tiến bộ trong xử lý ngôn ngữ tự nhiên (NLP) và học sâu (deep learning) đã làm tăng cường khả năng của chatbot trong việc hiểu và phản hồi các truy vấn của người dùng một cách chính xác và tự nhiên.

### Các loại chatbot

Chatbot có thể được phân loại thành hai nhóm chính: chatbot dựa trên quy tắc (rule-based) và chatbot dựa trên AI.

* Chatbot dựa trên quy tắc: Những chatbot này hoạt động dựa trên các quy tắc và kịch bản được lập trình sẵn. Chúng phản hồi theo các mẫu câu và từ khóa đã được xác định trước. Ưu điểm của loại chatbot này là dễ triển khai và kiểm soát, nhưng nhược điểm là hạn chế trong việc xử lý các truy vấn phức tạp và linh hoạt.
* Chatbot dựa trên AI: Những chatbot này sử dụng công nghệ AI và học máy (machine learning) để học hỏi từ dữ liệu và cải thiện khả năng hiểu và phản hồi. Chúng có thể phân tích ngữ cảnh, học từ các tương tác trước đó và tự động điều chỉnh phản hồi. Nhược điểm của loại này là cần dữ liệu lớn và tài nguyên tính toán mạnh để huấn luyện và vận hành.

### Ứng dụng của chatbot

Chatbot đã được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, mang lại nhiều lợi ích đáng kể.

* Dịch vụ khách hàng: Chatbot có thể hoạt động 24/7 để hỗ trợ khách hàng, trả lời các câu hỏi thường gặp, giải quyết các vấn đề cơ bản và hướng dẫn khách hàng trong quá trình sử dụng sản phẩm hoặc dịch vụ. Điều này giúp giảm tải công việc cho nhân viên và cải thiện trải nghiệm khách hàng.
* Thương mại điện tử: Trong ngành thương mại điện tử, chatbot có thể tư vấn sản phẩm, hỗ trợ quy trình đặt hàng, cung cấp thông tin về tình trạng đơn hàng và giải quyết các khiếu nại của khách hàng. Chatbot cũng có thể phân tích dữ liệu khách hàng để đề xuất sản phẩm phù hợp.
* Giáo dục: Trong lĩnh vực giáo dục, chatbot có thể hỗ trợ học tập, giải đáp thắc mắc về bài học, cung cấp tài liệu học tập và nhắc nhở học sinh về lịch 9 học. Đặc biệt, chatbot có thể hỗ trợ trong việc tư vấn tuyển sinh, cung cấp thông tin về các khóa học và chương trình đào tạo.
* Y tế: Trong y tế, chatbot có thể tư vấn sức khỏe, đặt lịch hẹn khám bệnh, cung cấp thông tin về triệu chứng bệnh và thuốc, và nhắc nhở bệnh nhân uống thuốc đúng giờ. Chatbot giúp giảm bớt gánh nặng cho các nhân viên y tế và cải thiện chất lượng chăm sóc sức khỏe.
* Ngân hàng và tài chính: Chatbot có thể hỗ trợ khách hàng thực hiện các giao dịch ngân hàng, kiểm tra số dư tài khoản, tư vấn tài chính, và giải đáp các thắc mắc liên quan đến dịch vụ ngân hàng. Điều này giúp tăng cường hiệu quả dịch vụ và cải thiện trải nghiệm khách hàng.

### Lợi ích của chatbot

Chatbot mang lại nhiều lợi ích cho các tổ chức và doanh nghiệp, bao gồm:

* Tăng cường hiệu quả: Chatbot có thể xử lý một lượng lớn yêu cầu cùng lúc mà không gặp phải sự mệt mỏi, giúp tăng cường hiệu quả làm việc và giảm thiểu thời gian chờ đợi của khách hàng.
* Tiết kiệm chi phí: Sử dụng chatbot giúp giảm chi phí nhân công cho các công việc lặp đi lặp lại và tăng cường tự động hóa các quy trình kinh doanh.
* Nâng cao trải nghiệm khách hàng: Chatbot cung cấp dịch vụ liên tục 24/7, giúp giải đáp kịp thời các thắc mắc của khách hàng và cải thiện sự hài lòng của họ.
* Thu thập và phân tích dữ liệu: Chatbot có khả năng thu thập và phân tích dữ liệu từ các tương tác với khách hàng, giúp doanh nghiệp hiểu rõ hơn về nhu cầu và hành vi của khách hàng để đưa ra các chiến lược kinh doanh phù hợp.

### Thách thức trong việc triển khai Chatbot

Mặc dù có nhiều lợi ích, việc triển khai chatbot cũng đối mặt với một số thách thức:

* Hiểu ngữ cảnh: Việc hiểu và xử lý ngữ cảnh trong các cuộc trò chuyện phức tạp là một thách thức lớn đối với chatbot.
* Bảo mật và quyền riêng tư: Việc bảo vệ thông tin cá nhân và đảm bảo quyền riêng tư của người dùng là một vấn đề quan trọng khi triển khai chatbot.
* Chất lượng dữ liệu: Chatbot dựa trên AI cần dữ liệu lớn và chất lượng cao để huấn luyện. Dữ liệu không đầy đủ hoặc không chính xác có thể ảnh hưởng đến hiệu quả của chatbot.
* Tương tác tự nhiên: Để tạo ra trải nghiệm tương tác tự nhiên, chatbot cần được thiết kế sao cho phản hồi một cách linh hoạt và phù hợp với ngữ cảnh.

# CƠ SỞ LÝ THUYẾT

## Mô hình ngôn ngữ lớn

### Khái niệm mô hình ngôn ngữ lớn

Con người có khả năng đáng chú ý là có thể diễn đạt và giao tiếp thông qua ngôn ngữ, bắt đầu phát triển trong thời thơ ấu và tiếp tục phát triển trong suốt cuộc đời của họ. Tuy nhiên máy móc không có khả năng vốn có để hiểu và giao tiếp bằng ngôn ngữ như con người trừ khi chúng được trang bị các thuật toán AI mạnh mẽ. Mục tiêu đạt được các kĩ năng đọc, viết và giao tiếp giống con người ở máy móc đã là một thách thức nghiên cứu lâu dài.

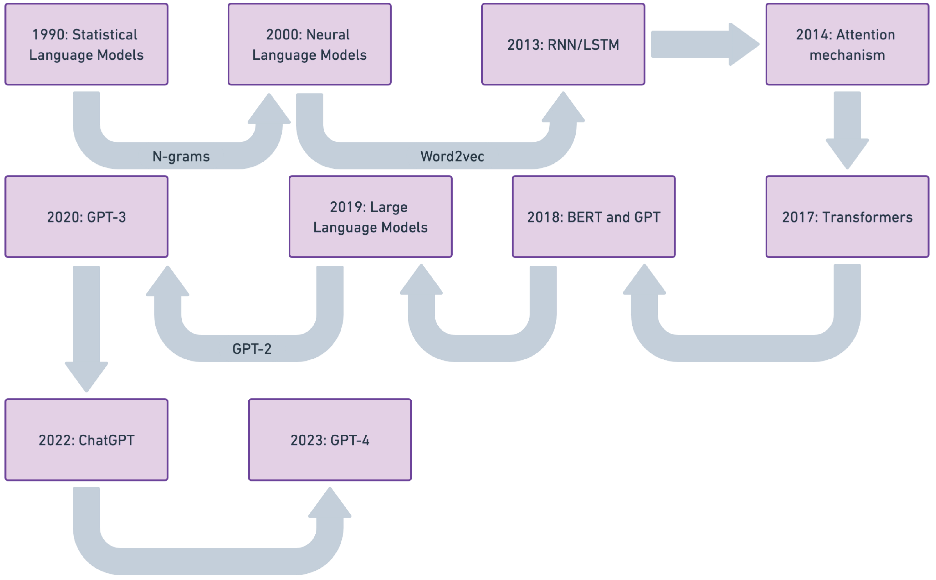
Sự ra đời của các mô hình ngôn ngữ lớn (LLM) có thể là do những tiến bộ trong các phương pháp học sâu (DL), sự sẵn có của một lượng lớn dữ liệu đào tạo. Những mô hình này, thường được đào tạo trước trên các tập hợp dữ liệu mở rộng từ website, có khả năng học các mẫu phức tạp, sắc thái ngôn ngữ và các mối quan hệ ngữ nghĩa. Việc tinh chỉnh các mô hình này trên các nhiệm vụ cụ thể đã cho thấy kết quả đầy hứa hẹn, đạt được hiệu suất tiên tiến trong nhiều chuẩn mực khác nhau.Mô hình ngôn ngữ (LM) là một phương pháp quan trọng để nâng cao trí thông minh của máy móc. LM liên quan đến việc mô hình hóa xác suất của các chiểu từ để dự đoán khả năng xảy ra trong tương lai.

Large language model (LLM) là một mô hình tính toán có khả năng tạo ngôn ngữ hoặc các tác vụ xử lý ngôn ngữ tự nhiên khác. Là mô hình ngôn ngữ, LLM có được những khả năng này bằng cách học các mối quan hệ thống kê từ một lượng lớn văn bản trong quá trình đào tạo tự giám sát và bán giám sát.

### Lịch sử hình thành

Mô hình ngôn ngữ lớn LLM đại diện cho một nhánh quan trọng trong hệ thống tính toán có khả năng hiểu và tạo ra ngôn ngữ loài người. Nó được xây dựng dựa trên các mô hình ngôn ngữ truyền thống như mô hình N-gram, LLM giải quyết các hạn chế vẫn còn tồn đọng như xử lý các từ hiếm gặp, overfitting và thấu hiểu được các mẫu ngôn ngữ phức tạp. Một ví dụ điển hình chẳng hạn như mô hình GPT-3 và GPT-4, áp dụng cơ chế tự tập trung (self-attention mechanism) trong kiến trúc Transformer để xử lý các dữ liệu tuần tự và hiểu được mối quan hệ giữa các từ trong một câu dài. Các tiến bộ chính bao gồm khả năng học ngữ cảnh để tạo ra văn bản mạch lạc từ các prompts và học tăng cường từ phản hồi của con người (Reinforcement Learning from Human Feedback – RLHF) để điều chỉnh mô hình sử dụng các phản hồi từ con người. Các kỹ thuật như xử lý yêu cầu (prompt engineering), hỏi – đáp, và tương tác đàm thoại được phát triển đáng kể trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

Mô hình ngôn ngữ (LM) là nền móng cho việc xử lý ngôn ngữ tự nhiên (NLP), bằng việc sử dụng các kỹ thuật toán học để tạo ra các quy luật và sự hiểu biết về ngôn ngữ cho các nhiệm vụ liên quan đến dự đoán và sinh văn bản. Qua nhiều thập kỷ, mô hình hóa ngôn ngữ đã được phát triển từ những mô hình ngôn ngữ dựa trên xác suất đơn giản đến nhưng mô hình ngôn ngữ lớn phức tạp như hiện nay. Sự phát triển nhanh chóng cho phép LLMs có thể xử lý, khái quát và tạo ra văn bản tương đương với khả năng của con người.



Hình 2.1 Lược sử các mô hình từ năm 1920 đến nay

### Word Embedding

Ngôn ngữ tự nhiên là một hệ thống phức tạp được sử dụng để bày tỏ ý nghĩa. Trong hệ thống này, các từ là đơn vị cơ bản của ý nghĩa. Như tên gọi của nó, word embedding là các vector được sử dụng để biểu diễn các từ, và có thể được xem là các vector đặc trưng hoặc các biểu diễn của từ. Kỹ thuật ánh xạ các từ thành các vector thực được gọi là word embedding. Trong những năm gần đây, word embedding ngày càng trở thành một kiến thức cơ bản trong việc xử lý ngôn ngữ tự nhiên.

1. **Skip-gram**

Skip-gram là một kiến trúc trong thuật toán Word2Vec được thiết kế để học biểu diễn từ vựng bằng cách khai thác ngữ cảnh của từ trong câu. Ý tưởng chính của Skip-gram là dự đoán các từ ngữ cảnh xung quanh một từ trung tâm.

Khi sử dụng Skip-gram, mô hình nhận đầu vào là một từ trung tâm và cố gắng dự đoán các từ xuất hiện trong cửa sổ ngữ cảnh xung quanh từ đó. Cửa sổ ngữ cảnh có kích thước cố định, xác định số lượng từ gần nhất được coi là ngữ cảnh.

Công thức xác suất để tối ưu hóa mô hình là cực đại hóa xác suất có điều kiện của các từ ngữ cảnh dựa trên từ trung tâm. Skip-gram hoạt động hiệu quả với dữ liệu lớn và đặc biệt tốt trong việc học biểu diễn từ vựng cho các từ hiếm. Kết quả là mỗi từ được biểu diễn dưới dạng vector trong không gian liên tục, giúp nắm bắt ý nghĩa ngữ nghĩa và mối quan hệ giữa các từ.

A black and white image of a row of circles

Description automatically generated

Hình 2.2 Minh họa Skip-gram dưới dạng mạng nơ-ron

1. **Continuous Bag of Words (CBOW)**

CBOW (Continuous Bag of Words) là một phương pháp học biểu diễn từ vựng (word embeddings) trong thuật toán Word2Vec, với mục tiêu chính là dự đoán một từ trung tâm (center word) dựa trên các từ ngữ cảnh (context words) xung quanh nó.

Trong CBOW, đầu vào của mô hình là các từ ngữ cảnh trong một cửa sổ ngữ cảnh cố định, và đầu ra là xác suất của từ trung tâm. Các từ ngữ cảnh được xử lý như một "túi từ" (bag of words), tức là thứ tự xuất hiện của chúng không quan trọng, mà chỉ xét đến sự hiện diện của chúng.

CBOW sử dụng mạng neural đơn giản với một lớp nhúng để ánh xạ từ ngữ cảnh vào không gian vector, sau đó tính trung bình các vector này để tạo ra một biểu diễn tổng hợp. Biểu diễn này được sử dụng để dự đoán từ trung tâm thông qua hàm softmax.

Phương pháp CBOW hoạt động hiệu quả với tập dữ liệu nhỏ và huấn luyện nhanh hơn so với Skip-gram. Tuy nhiên, nó có thể kém chính xác hơn với các từ hiếm, do phụ thuộc vào ngữ cảnh tổng hợp. Kết quả của CBOW là mỗi từ được ánh xạ vào một vector trong không gian liên tục, giúp biểu diễn ý nghĩa ngữ nghĩa và mối quan hệ giữa các từ.

A black and white image of a line

Description automatically generated

Hình 2.3 Minh họa CBOW dưới dạng mạng nơ-ron

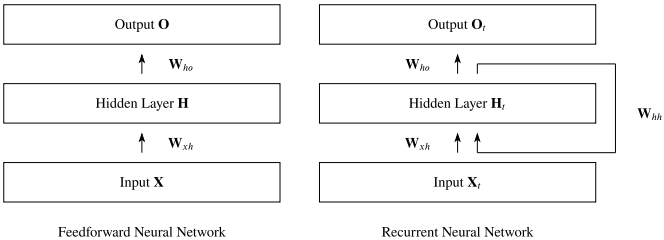
### Mạng Nơ-ron hồi tiếp truyền thống RNN

Mạng nơ-ron hồi tiếp tiếp truyền thống hay recurrent neural network (RNNs) là một lớp của các mô hình học sâu được thiết kế để xử lý dữ liệu dạng chuỗi. Không giống như các mạng truyền thẳng, RNNs thể hiện đặc trưng riêng biệt trong việc duy trì bộ nhớ với các đầu vào trước đó bằng việc sử dụng trạng thái bên trong (internal state – memory) của nó để xử lý chuỗi các đầu vào. Nó làm chúng phù hợp một cách lý tưởng cho các ứng dụng chẳng hạn như trong xử lý ngôn ngữ tự nhiên, nhận biết giọng nói, và dự đoán chuỗi thời gian, với ngữ cảnh và thứ tự của dữ liệu là quan trọng.

RNNs là một dạng kiến trúc mạng nơ-ron được sử dụng chủ yếu để phát hiện các mẫu trong chuỗi dữ liệu. Dữ liệu này có thể là chữ viết tay, bộ gen, văn bản hoặc chuỗi thời gian số học thường được tạo ra trong môi trường công nghiệp (ví dụ: thị trường chứng khoán hoặc cảm biến). Tuy nhiên, RNNs cũng có thể được áp dụng cho hình ảnh nếu hình ảnh được chia nhỏ thành một chuỗi các mảnh nhỏ và được xử lý như một chuỗi. Ở cấp độ cao hơn, RNNs được ứng dụng trong Mô hình hóa Ngôn ngữ & Sinh Văn bản, Nhận diện Giọng nói, Sinh Mô tả Hình ảnh, hoặc Gắn Thẻ Video.

Điểm khác biệt chính giữa RNNs và Mạng Nơ-ron Truyền Tiến (Feedforward Neural Networks), còn được gọi là Mạng Nhận Thức Đa Lớp (MLPs), là cách thông tin được truyền qua mạng. Trong khi Mạng Truyền Tiến truyền thông tin qua mạng mà không có chu trình, RNNs có các chu trình và truyền thông tin trở lại chính nó. Điều này cho phép RNNs mở rộng chức năng của Mạng Truyền Tiến bằng cách xem xét các đầu vào trước đó chứ không chỉ xem xét đầu vào hiện tại . Sự khác biệt này được minh họa như hình dưới.

Quá trình truyền thông tin từ vòng lặp trước đến lớp ẩn có thể được mô tả bằng ký hiệu toán học. Trong đó, trạng thái ẩn và đầu vào tại bước thời gian lần lượt được ký hiệu là và trong đó là số lượng mẫu, là số đầu vào của mỗi mẫu, là số đơn vị ẩn. Hơn nữa, ma trận trọng số , ma trận từ trạng thái ẩn sang trạng thái ẩn , và tham số bias . Cuối cùng tất cả thông tin này được truyền qua một hàm kích hoạt , thường là hàm sigmoid logistic hoặc tanh, để chuẩn bị gradient cho việc sử dụng trong lan truyền ngược. Khi kết hợp tất cả các ký hiệu này thu được các phương trình:



Hình 2.4 Minh họa khác biệt giữa mạng truyền thẳng và mạng hồi quy

Vì được tính đệ quy bao gồm , và quá trình này diễn ra ở mọi bước thời gian, RNN sẽ lưu giữ dấu vết của tất cả các trạng thái ẩn trước và chính .

Lan truyền ngược qua thời gian (Backpropagation through time) là phiên bản thích ứng của thuật toán lan truyền ngược (backpropagation) dành cho RNNs. Về mặt lý thuyết BPTT triển khai việc mở rộng RNN thành một Mạng Nơ-ron Truyền Tiến truyền thống để áp dụng lan truyền ngược. Để thực hiện điều này, chúng ta sử dụng các ký hiệu đã được định nghĩa trước đó cho RNN.

Khi thực hiện truyền xuôi đầu vào qua mạng, trạng thái ẩn và trạng thái đầu ra được tính toán từng bước một. Sau đó, hàm mất mát được định nghĩa để mô tả sự khác biệt giữa tất cả đầu ra và các giá trị mục tiêu . Về cơ bản, hàm này tổng hợp tất cả các thành phần mất mát tại mỗi bước cập nhật đến thời điểm hiện tại.

Thành phần mất mát có thể được định nghĩa theo nhiều cách khác nhau tùy thuộc vào bài toán cụ thể (ví dụ: mất mát Trung bình Bình phương – Mean Squared Error, mất mát Hinge – Hinge Loss, mất mát Cross Entropy – Cross Entropy Loss, v.v.).

### Mạng Nơ-ron hồi tiếp hiện đại GRU và LSTM

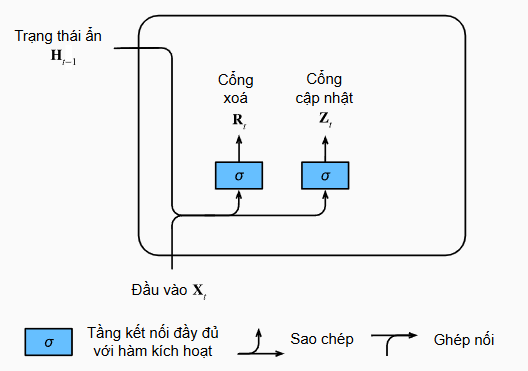
1. **GRU**

Gated Recurrent Units (GRU) là một loại kiến trúc mạng nơ ron hồi tiếp được đưa ra để giải quyết vấn đề tiêu biến hoặc bùng nổ gradient. Sự khác biệt chính giữa RNN thông thường và GRU là GRU hỗ trợ việc kiểm soát trạng thái ẩn. Điều này có nghĩa là sẽ có các cơ chế được học để quyết định khi nào nên cập nhật và khi nào nên xóa trạng thái ẩn. Ví dụ, nếu ký tự đầu tiên có mức độ quan trọng cao, mô hình sẽ học để không cập nhật trạng thái ẩn sau lần quan sát đầu tiên. Tương tự, mô hình sẽ học cách bỏ qua nhưng quan sát tạm thời không liên quan, cũng như cách xóa trạng thái ẩn khi cần thiết.

Cổng xóa và cổng cập nhật

Cổng xóa và cổng cập nhật được thiết kế thành các vector có các phần tử trong khoảng để có thể biểu diễn các tổ hợp lồi. Chẳng hạn, một biến xóa cho phép kiểm soát bao nhiêu phần của trác thái trước đây được giữ lại. Tương tự, một biến cập nhật cho phép kiểm soát bao nhiêu phần của trạng thái mới sẽ giống trạng thái cũ.

Bắt đầu bằng việc thiết kế cổng ra các biến này. Cổng xóa và cổng cập nhật trong GRU thể hiện ở hình dưới đây với đầu vào ở bước thời gian hiện tại và trạng thái ẩn ở bước thời gian trước đó . Đầu ra được tạo bởi một tầng kết nối đầy đủ với hàm kích hoạt sigmoid.



Hình 2.5 Cổng xóa và cổng cập nhật trong GRU

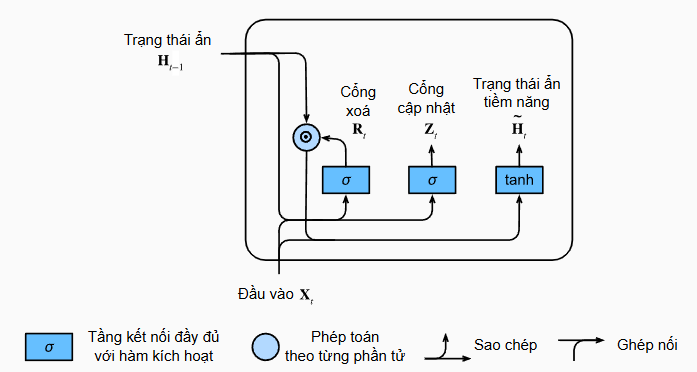
Tại bước thời gian , với đầu vào minibatch là (số lượng mẫu: , số lượng đầu vào: ) và trạng thái ẩn ở bước thời gian gần nhất là (số lượng trạng thái ẩn: ), cổng xóa và cổng cập nhật được tính như sau:

Ở đây, và là các tham số trọng số và là các hệ số điều chỉnh. Sigmoid sẽ được sử dụng để biến đổi các giá trị đầu vào nằm trong khoảng .

1. **Hoạt động của cổng xóa**

Cổng xóa được tích hợp với một cơ chế cập nhật trạng thái ẩn thông thường.

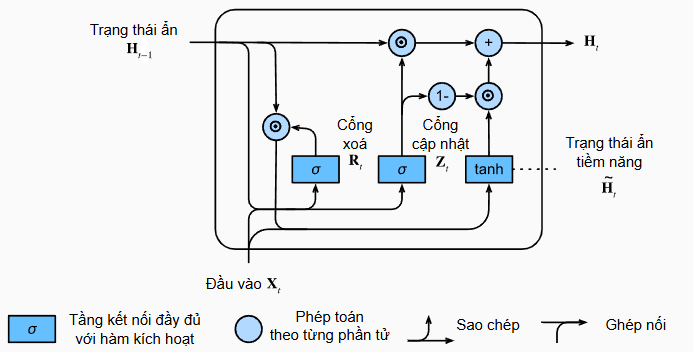
Hàm tanh giúp đảm bảo các giá trị trạng thái tiềm ẩn nằm trong khoảng . Nếu muốn giảm ảnh hưởng của các trạng thái trước đó, ta có thể nhân với có giá trị gần với 1, kết quả sẽ giống RNN thông thường. Nếu tất cả các phần tử của cổng xóa gần với 0, trạng thái ẩn sẽ là đầu ra của một perceptron đa tầng với đầu vào . Bất kỳ trạng thái ẩn nào tồn tại trước đó đều được đặt lại về giá trị mặc định. Tại đây nó được gọi là trạng thái tiềm ẩn tiềm năng, và chỉ tiềm năng vì ta vẫn cần kết hợp thêm đầu ra của cổng cập nhật:



Hình 2.6 Tính toán của trạng thái tiềm ẩn trong GRU

1. **Hoạt động của cổng cập nhật**

Tại đây hiệu ứng của cổng cập nhật sẽ được kết hợp. Cổng này xác định mức độ giống nhau giữa trạng thái mới và trạng thái cũ , cũng như mức độ trạng thái ẩn tiềm năng được sử dụng. Biến cổng (gating variable) được sử dụng cho mục đích này, bằng cách áp dụng tổ hợp lồi giữa trạng thái cũ và trạng thái tiềm năng. Ta có phương trình cập nhật cuối cùng cho GRU.



Hình 2.7 Tính toán trạng thái ẩn trong GRU

1. **LSTM**

Thách thức đối với việc lưu trữ những thông tin dài hạn và bỏ quả đầu vào ngắn hạn trong các mô hình biến tiềm ẩn đã tồn tại trong thời gian dài. Một trong những phương pháp tiếp cận sớm nhất để giải quyết vấn đề này là LSTM [Hochreiter & Schmidhuber, 1997]. Nó có nhiều tính chất tương tự Nút hồi tiếp có cổng (GRU). Điều thú vị là thiết kế của LSTM chỉ phức tạp hơn GRU một chút nhưng đã xuất hiện trước GRU gần hai thập kỷ.

Có thể cho rằng thiết kế này được lấy cảm hứng từ các cổng logic trong máy tính. Để kiểm soát một ô nhớ sẽ cần một số các cổng. một cổng để đọc các thông tin từ ô nhớ đó (trái với việc đọc từ các ô khác) được gọi cổng này là cổng đầu ra (output gate). Một cổng thứ hai để quyết định khi nào cần ghi dữ liệu vào ô nhớ, được gọi là cổng đầu vào. Cuối cùng là cơ chế để thiết lập lại nội dung chứa trong ô nhớ, được chi phối bởi một cổng quên (forget gate). Tác dụng của chúng là đưa ra quyết định khi nào cần nhớ và khi nào nên bỏ qua đầu vào trong trạng thái tiềm ẩn thông qua một cơ chế chuyên dụng.

1. **Cổng đầu vào, cổng quên, cổng đầu ra**

Dữ liệu được đưa vào các cổng LSTM là đầu vào ở bước thời gian hiện tại và trạng thái ở bước thời gian trước đó . Những đầu vào này được xử lý bởi một tầng kết nối đầy đủ và một hàm kích hoạt sigmoid để tính toán các giá trị của các cổng đầu vào, cổng quên và cổng đầu ra. Kết quả là, tất cả các giá trị đầu ra tại ba cổng đều nằm trong khoảng .

A diagram of a diagram

Description automatically generated

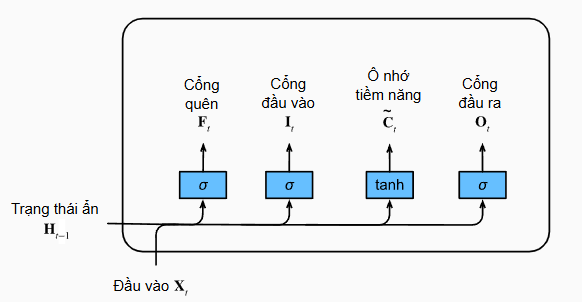
Hình 2.8 Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra tại một đơn vị LSTM

Giả sử rằng có nút ẩn, mỗi minibatch có kích thước n và kích thước đầu vào là . Như vậy đầu ra là và trạng thái ẩn của bước thời gian trước đó là . Tương tự, các cổng được định nghĩa như sau: cổng đầu vào là , cổng quên là , và cổng đầu ra là . Chúng được tính như sau:

Trong đó và là các trọng số và là các hệ số điều chỉnh.

1. **Ô nhớ tiềm năng**

Ô nhớ tiềm năng được thiết kế để lưu giữ là điều chỉnh thông tin trong thời gian dài. Nó được thiết kế để giải quyết vấn đề biến mất gradient mà các mạng RNN thông thường gặp phải khi học các phụ thuộc dài hạn.

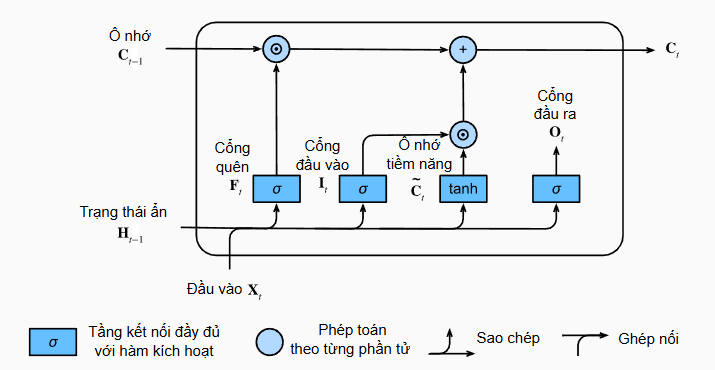


Hình 2.9 Các phép tính toán trong ô nhớ tiềm năng LSTM

1. **Ô nhớ**

Trong LSTM chúng ta có 2 tham số để quản lý việc nhớ và quên. điều chỉnh lượng dữ liệu mới lấy được thông qua và tham số quên chỉ định lượng thông tin cũ cần giữ lại trong ô nhớ . Sử dụng cùng một phép nhân theo từng điểm (pointwise), có phương trình cập nhật như sau:

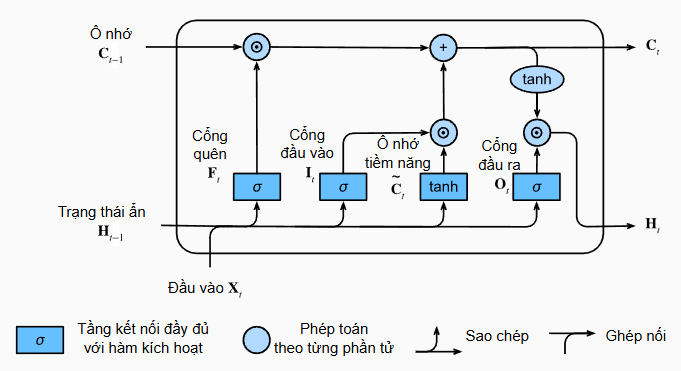
Nếu giá trị ở cổng quên luôn xấp xỉ bằng 1 và cổng đầu vào luôn xấp xỉ bằng 0, thì giá trị ô nhớ trong quá khứ sẽ được lưu lại qua thời gian và truyền tới bước thời gian hiện tại. Thiết kế này được giới thiệu nhằn giảm bớt vấn đề tiêu biến gradient cũng như nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian tốt hơn.



Hình 2.10 Các phép tính toán trong ô nhớ của LSTM

1. **Các trạng thái ẩn**

Cuối cùng cần phải xác định các trạng thái ẩn . Đây là nơi cổng đầu ra được sử dụng. Trong LSTM, đây chỉ đơn giản là một phiên bản có kiểm soát của hàm kích hoạt trong ô nhớ. Bất cứ khi nào giá trị của cổng đầu ra là 1, thức chất ta đang đưa toàn bộ thông tin trong ô nhớ tới bộ dự đoán. Ngược lại khi giá trị của cổng đầu ra là 0, tất cả các thông tin trong ô nhớ được giữ lại và không xử lý gì thêm.

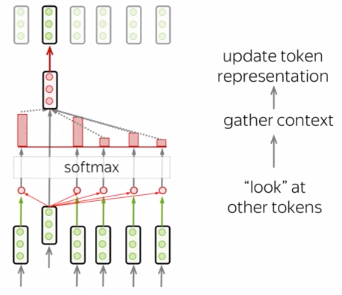


Hình 2.11 Các phép tính của trạng thái ẩn

### Kiến trúc mạng Transformer

1. **Cơ chế Self Attention**

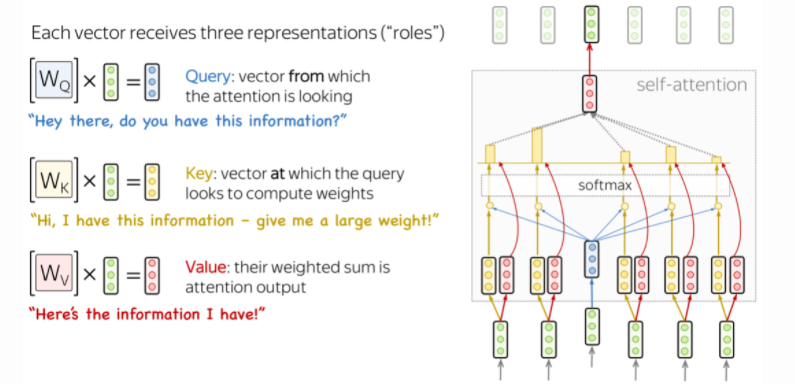
Self Attention là thành phần quan trọng nhất của transformer. Có thể hiểu self attention là sự chú ý trong một câu khi từng thành phần trong câu sẽ tương tác với nhau. Từng tokens sẽ “quan sát” các tokens còn lại trong câu, thu thập ngữ cảnh của câu và cập nhật vector biểu diễn.



Hình 2.12 Minh họa cơ chế Self-Attention

Để xây dựng cơ chế self attention, cần chú ý đến hoạt động của 3 vector biểu diễn cho mỗi từ lần lượt là Query: hỏi thông tin, Key: trả lời rằng nó có một số thông tin, và Value: trả về thông tin đó.

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).



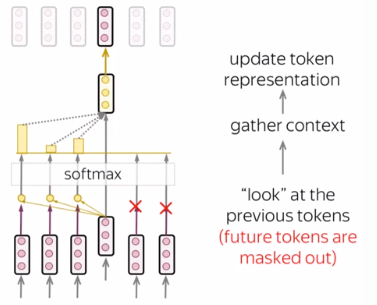
Hình 2.13 Minh họa Q, K, V và ứng dụng của chúng trong Self-Attention

Biểu thức tính toán attention vector như sau:

Với là số chiều của vector Key với mục đích tránh tràn luồng.

1. **Cơ chế masked self attention**

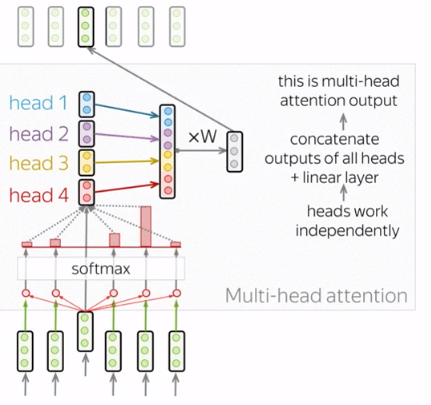
Đây là cơ chế được sử dụng cho decoder trong transformer, cụ thể nó thực hiện nhiệm vụ chỉ cho phép target token tại time-step hiện tại chỉ được phép dùng các tokens ở time-step trước đó. Về hoạt động nó cũng giống như đã giới thiệu ở trên, ngoại trừ việc nó không tính đến attention của những tokens trong tương lai.



Hình 2.14 Minh họa Masked Self-Attention

1. **Multi-Head Attention**

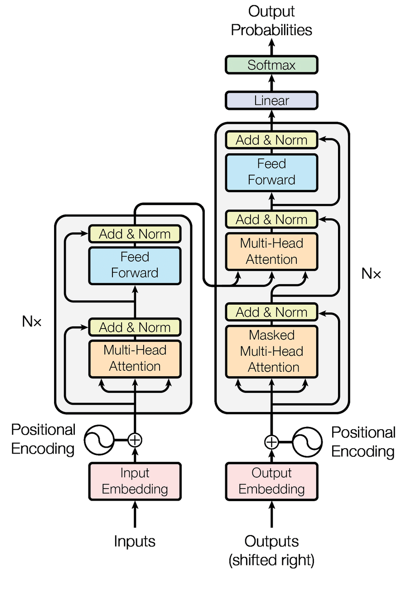
Thông thường, để có thể hiểu được vai trò của một từ trong một câu ta cần hiều được sự liên quan giữa từ đó và các thành phần còn lại trong câu. Điều này rất quan trọng trong quá trình xử lí câu đầu vào ở ngôn ngữ A và cả trong quá trình tạo ra câu ở ngôn ngữ B. Vì vậy, mô hình cần phải tập trung vào nhiều thứ khác nhau, cụ thể là thay vì chỉ có một cơ chế self attention như đã giới thiệu hay còn gọi là 1 "head" thì mô hình sẽ có nhiều "heads" mỗi head sẽ tập trung vào khía cạnh về sự liên quan giữa từ và các thành phần còn lại. Đó chính là multi-head attention.



Hình 2.15 Minh họa Multi-head Attention

1. **Kiến trúc của Transformer**

Kiến trúc transformer gồm 2 phần chính là Encoder và Decoder hay còn gọi là bộ mã hóa và bộ giải mã. Encoder thường có = 6 layers xếp chồng lên nhau. Mỗi layer sẽ có multi-head attention và khối feed-forward. Ngoài ra còn có các kết nối residual. Thêm vào đó là các khối Decoder có số lượng layers bằng với bộ Encoder xếp chồng lên nhau. Kiến trúc khá giống Encoder chỉ thêm khối masked multi-head attention ở ví trí đầu tiên.



Hình 2.16 Minh họa kiến trúc Transformer

1. **Positional encoding**

Bởi vì transformer không có các mạng hồi tiếp hay mạng tích chập nên nó sẽ không biết được thứ tự của các token đầu vào. Vì vậy, cần phải có cách nào đó để cho mô hình biết được thông tin này. Đó chính là nhiệm vụ của positional encoding. Như vậy, sau bước nhúng từ (embedding layers) để thu được các tokens thì ta sẽ cộng nó với các vector thể hiện vị trí của từ trong câu.

1. **Lớp Normalization**

Trong hình ảnh cấu trúc, có lớp "Add & Norm" thì từ Norm thể hiện cho lớp Normalization. Lớp này đơn giản là sẽ chuẩn hóa lại đầu ra của multi-head attention, mang lại hiệu quả cho việc nâng cao khả năng hội tụ.

1. **Kết nối residual**

Kết nối residual bản chất rất đơn giản: thêm đầu vào của một khối vào đầu ra của nó. Với kết nối này giúp mạng có thể chồng được nhiều layers. Như trên hình, kết nối residual sẽ được sử dụng sau các khối FFN và khối attention. Như trên hình từ "Add" trong "Add & Norm" sẽ thể hiện cho kết nối residual.

1. **Khối Feed-Forward**

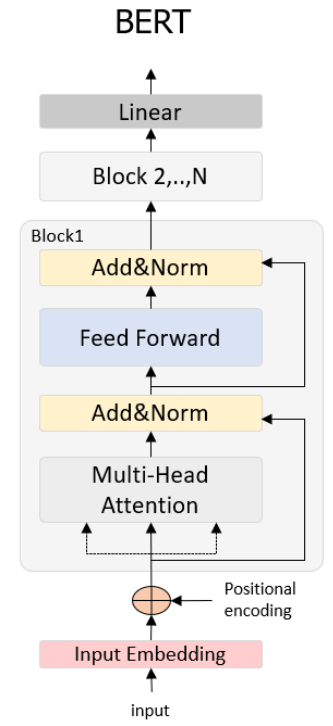
Đây là khối cơ bản, sau khi thực hiện tính toán ở khối attention ở mỗi lớp thì khối tiếp theo là FFN. Có thể hiểu là cơ chế attention giúp thu thập thông tin từ những tokens đầu vào thì FFN là khối xử lí những thông tin đó.

### Mô hình BERT

BERT là một mô hình được huấn luyện trước (pre-trained model), các vector đại diện theo ngữ cảnh 2 chiều cả từ, được sử dụng để transfer sang các bài toán khác trong lĩnh vực xử lý ngôn ngữ tự nhiên. BERT được áp dụng thành công trong việc xác định biểu diễn số hóa của từ trong không gian số hóa dựa trên ngữ cảnh mà từ đó xuất hiện.

BERT là mô hình được xây dụng và huấn luyện bởi Google trên tập dữ liệu lớn gồm Toronto Book Corpus và Wikipedia. BERT được thiết kế để huấn luyện trước các biểu diễn hai chiều từ văn bản không được gắn nhãn bằng cách điều hòa chung cả ngữ cảnh bên trái và bên phải trong tất cả các lớp. Do đó, mô hình BERT được huấn luyện trước có thể được tinh chỉnh chỉ với một lớp đầu ra bổ sung để tạo ra các mô hình hiện đại cho nhiều nhiệm vụ, chẳng hạn như trả lời câu hỏi và suy luận ngôn ngữ mà không cần phải thực hiện tác vụ quan trọng nào nhằm sửa đổi kiến trúc cụ thể. BERT được thiết kế để tạo ra các biểu diễn ngôn ngữ hai chiều từ văn bản chưa được gán nhãn thông qua việc đồng nhất thông tin từ cả hai phía ngữ cảnh, từ trái sang phải và ngược lại tại tất cả các lớp. Vì vậy BERT có khả năng học được các biểu diễn tự nhiên và phong phú của từ trong ngữ cảnh của chúng.

Mô hình BERT đã được huấn luyện trước có thể dễ dàng tinh chỉnh cho các nhiệm vụ cụ thể chỉ bằng cách thêm một lớp đầu ra mới, mà không cần phải điều chỉnh toàn bộ kiến trúc. Điều này mở ra khả năng xây dựng các mô hình ngôn ngữ lớn và hiện đại thực hiện các nhiệm vụ như trả lời câu hỏi và suy luận ngôn ngữ mà không yêu cầu quá nhiều công sức đối với việc sửa đổi kiến trúc.



Hình 2.17 Minh họa kiến trúc mô hình BERT

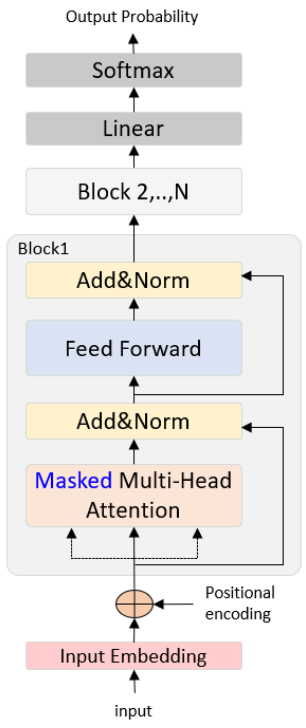
### Mô hình GPT

GPT (Generative Pre-trained Transformer) là một mô hình ngôn ngữ mạnh mẽ dựa trên kiến trúc Transformer, được phát triển bởi OpenAI. GPT được thiết kế để thực hiện các nhiệm vụ sinh văn bản và xử lý ngôn ngữ tự nhiên (NLP) một cách hiệu quả và chính xác. Mô hình này hoạt động theo cơ chế tiền huấn luyện và tinh chỉnh (pre-training và fine-tuning), cho phép nó học được ngữ nghĩa và ngữ cảnh của ngôn ngữ từ khối lượng lớn dữ liệu.

GPT sử dụng kiến trúc Transformer decoder với các thành phần chính như multi-head self-attention, feed-forward networks, và vị trí nhúng (positional embedding) để xử lý đầu vào là các chuỗi văn bản. Trong giai đoạn tiền huấn luyện, mô hình được huấn luyện trên tập dữ liệu văn bản khổng lồ bằng cách học cách dự đoán từ tiếp theo trong câu dựa trên từ đã cho trước. Quá trình này được gọi là học không giám sát. Sau khi được tiền huấn luyện, GPT có thể được tinh chỉnh trên các tác vụ cụ thể như tóm tắt văn bản, dịch thuật, trả lời câu hỏi và phân loại văn bản. Đây là bước học có giám sát, trong đó mô hình được cung cấp thêm các nhãn mục tiêu để cải thiện khả năng cho nhiệm vụ cụ thể.

GPT sở hữu nhiều ưu điểm vượt trội. Thứ nhất, mô hình có khả năng sinh văn bản tự nhiên, tạo ra nội dung mượt mà và có ý nghĩa nhờ cơ chế attention. Thứ hai, GPT rất linh hoạt trong nhiều tác vụ NLP khác nhau mà không cần thay đổi kiến trúc mô hình. Thứ ba, nhờ khả năng học ngữ cảnh dài hạn, GPT có thể nắm bắt mối liên hệ giữa các từ ở khoảng cách xa trong văn bản, giúp cải thiện đáng kể chất lượng đầu ra.

Mô hình GPT có nhiều ứng dụng trong thực tế, từ chatbot, trợ lý ảo, viết nội dung tự động đến phân tích ngữ nghĩa và dịch thuật. Với các phiên bản cải tiến như GPT-2, GPT-3, và GPT-4, GPT đã chứng minh khả năng vượt trội trong việc xử lý ngôn ngữ tự nhiên và tạo ra các văn bản chất lượng cao, gần như không thể phân biệt được với văn bản do con người viết.



Hình 2.18 Minh họa kiến trúc mô hình GPT

### Mixture of Experts

Mixture of Expert (MoE) là một kiến trúc được thiết kế cho mạng nơ ron mà chia các lớp tính toán hoặc các hoạt động tính toán (computation of layer or operation, eg. Linear layer, MLPs, attention projector) thành các mạng con chuyên dụng, được gọi là các experts. Mỗi Expert sẽ độc lập thực hiện tính toán, và kết quả được tổng hợp để tạo ra đầu ra cuối cùng của lớp MoE. Kiến trúc MoE có thể được phân loại hoặc dày đặc (dense), trong đó mọi expert đều tham gia cho từng đầu vào hoặc thưa thớt (sparse), trong đó chỉ có một tập con các expert được sử dụng cho mỗi đầu vào.

1. **Kiến trúc và hiệu suất của Mixtral 8x7B**

Mixtral 8x7B sử dụng kiến trúc Sparse Mixture of Experts, áp dụng cấu trúc của Mistral 7B những kết hợp với 8 khối feedforward (experts) trong 1 layer. Đối với mỗi token ở mỗi lớp, một mạng router sẽ chọn 2 experts để xử lý trạng thái hiện tại và kết hợp với đầu ra của chúng. Mặc dù mỗi token chỉ tương tác với 2 experts tại cùng một thời điểm, nhưng các experts được chọn sẽ khác nhau tại mỗi thời điểm. Do vậy, mỗi token sẽ truy cập vào 47 tỷ tham số nhưng chỉ sử dụng 13 tỉ tham số trong quá trình suy luận. Mixtral 8x7B không chỉ sánh ngang mà còn vượt qua Llama 2 70B và GPT 3.5 trên các tiêu chuẩn đánh giá.Hiệu suất của nó còn vượt trội hơn hẳn so với Llama 2 70B trong các tác vụ toán học và code và đa ngôn ngữ.

A diagram of a router network

Description automatically generated

Hình 2.19 Lược đồ kiến trúc mô hình Mixtral 8x7B

## Các chiến lược tinh chỉnh mô hình ngôn ngữ lớn

### Pre-trained Language Model (PLM)

PLM là mô hình được tiền huấn luyện dựa trên một bộ dữ liệu khổng lồ các văn bản không nhãn để hiểu các cấu trúc ngôn ngữ cơ bản. Chúng sau đó sẽ được tinh chỉnh trên bộ dữ liệu nhỏ hơn và tập trung vào các dữ liệu để xử lý các tác vụ nhất định. Các mô hình này mang lại tính linh hoạt cao qua việc được sử dụng làm điểm khởi đầu cho nhiều các tác vụ khác nhau như tinh chỉnh cũng như trích xuất đặc trưng áp dụng vào nhiều bài toán khác nhau. Thêm vào đó nó giúp tiết kiệm tài nguyên khi không phải huấn luyện lại từ đầu làm tốn tài nguyên và thời gian, thay vào đó sử dụng pre-trained model sẽ giúp giảm chi phí một cách đáng kể.

### Khái niệm tinh chỉnh

Tinh chỉnh hay fine-tuning là việc sử dụng các mô hình tiền huấn luyện (pre-trained model) chẳng hạn như các mô hình GPT của OpenAI như là điểm khởi đầu. Quá trình này liên quan đến việc huấn luyện mô hình dựa trên tập dữ liệu nhỏ hơn và liên quan mật thiết đến một lĩnh vực cụ thể. Hướng tiếp cận này được xây dựng dựa trên các hiểu biết trước đó của mô hình, nâng cao hiệu suất của mô hình trên các tác vụ cụ thể với ít hơn các yêu cầu về dữ liệu và tính toán.

Tinh chỉnh sẽ biến đổi các đặc trưng và mẫu đã được học của pre-trained model để phù hợp với các tác vụ mới, cải thiện hiệu suất và giảm thiểu yêu cầu độ lớn của bộ dữ liệu. Nó trở lên ngày càng phổ biến trong NPL về các tác vụ như phân loại văn bản, phân tích cảm xúc, và hỏi đáp.

### Các dạng tinh chỉnh LLM

1. **Tinh chỉnh không giám sát (Unsupervised Fine-Tuning)**

Tinh chỉnh không giám sát là phương pháp không yêu cầu dữ liệu có nhãn. Thay vào đó LLM được học trên một tập dữ liệu lớn các văn bản không có nhãn của một lĩnh vực cụ thể, giúp cải thiện sự hiểu biết của chúng về ngôn ngữ. Hướng tiếp cận này là hữu dụng cho các lĩnh vực mới như pháp lý hay y tế nhưng lại ít chính xác hơn trong các tác vụ cụ thể chẳng hạn như phân loại hoặc tóm tắt.

1. **Tinh chỉnh có giám sát (Supervised Fine-Tuning SFT)**

SFT liên quan đến việc cung cấp cho LLM dữ liệu có nhãn được thiết kế cho tác vụ mục tiêu. Ví dụ như việc tinh chỉnh LLM để phân loại trong ngữ cảnh kinh doanh sử dụng tập dữ liệu các đoạn văn bản có nhãn. Tuy là rất hữu ích nhưng phương pháp này yêu cầu một lượng dữ liệu có nhãn đáng kể, và nó có thể rất tốn kém và mất thời gian để tạo ra.

1. **Instruction Fine-Tuning thông qua Prompt Engineering**

Phương pháp này dựa trên việc cung cấp cho LLM các chỉ dẫn dựa trên ngôn ngữ tự nhiên, hữu dụng cho việc tạo ra các trợ lý chuyên ngành. Nó giúp giảm thiểu nhu cầu với lượng lớn các dữ liệu có nhãn nhưng lại phụ thuộc rất lớn vào chất lượng của các prompt.

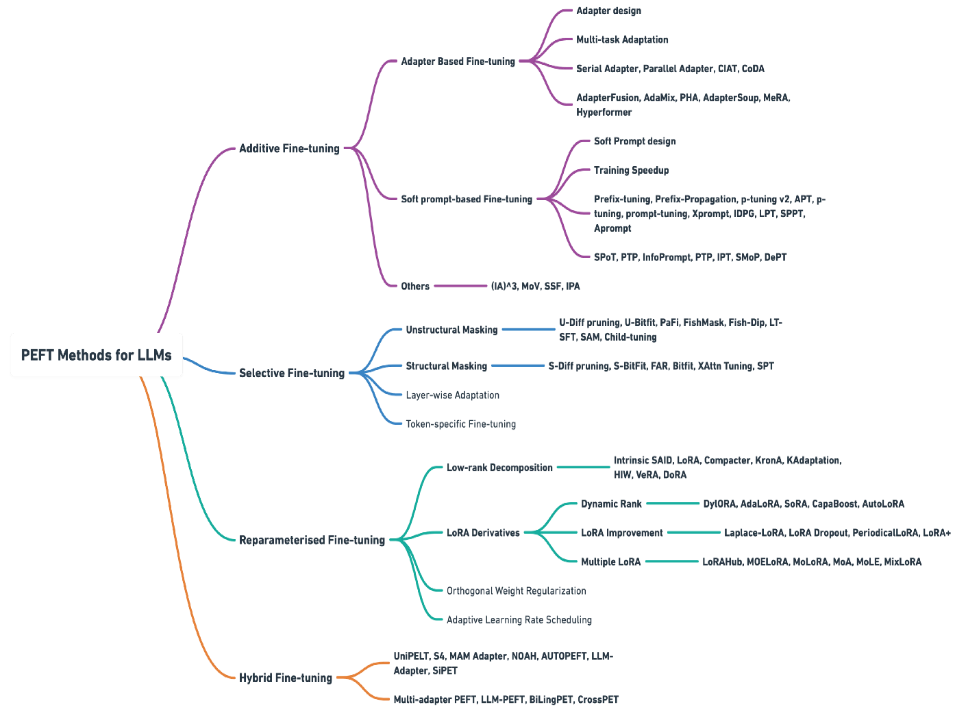
## Tinh chỉnh với tham số hiệu quả (PEFT)

### Khái niệm tinh chỉnh với tham số hiệu quả

Tinh chỉnh với tham số hiệu quả hay Parameter-Efficient Fine Tuning là một kỹ thuật NLP hiệu quả bằng việc điều chỉnh khéo léo các mô hình ngôn ngữ được đào tạo trước (pre-trained model) cho các ứng dụng khác nhau với hiệu quả vượt trội.

Phương pháp PEFT chỉ tinh chỉnh một lượng nhỏ các tham số mô hình trong khi vẫn giữ nguyên phần lớn các tham số từ pre-trained LLM, vì thế nó sẽ giảm đáng kể chi phí tính toán cũng như lưu trữ. Hướng tiếp cận này giúp giảm các vấn đề liên quan đến việc quên đáng kể là hiện tượng mạng nơ-ron quên đi các tri thức đã được học trước đó, vì vậy làm giảm hiệu suất của mô hình trên các tác vụ đã được học trước đó khi huấn luyện trên các bộ dữ liệu mới.

Phương pháp PEFT thể hiện hiệu suất vượt trội khi so sánh với tinh chỉnh toàn bộ, cụ thể trong các trường hợp có ít dữ liệu, thêm đó nó khái quát hóa một cách tốt hơn với các ngữ cảnh không thuộc lĩnh vực đã học. Kỹ thuật này có tính ứng dụng trong nhiều trường hợp chẳng hạn như phân loại tâm lý tài chính và thông dịch các thuật ngữ y tế.



Hình 2.20 Tổng quan các phương pháp PEFT

### Adapters

Phương pháp dựa trên adapter sẽ thêm vào mô hình các tham số có thể huấn luyện được vào sau lớp attention và fully connected của một mô hình pretrained đã được đóng băng, với mục tiêu là giảm bớt việc sử dụng bộ nhớ và giúp đẩy nhanh quá trình huấn luyện. Cách tiếp cận cụ thể sẽ khác nhau tùy thuộc vào các adapter, nó có thể liên quan đến thêm vào một lớp bổ sung hoặc biểu diễn độ khác nhau của tham số cập nhật như là một phân rã cấp thấp của ma trận trọng số. Bất kể phương pháp nào, adapters  thường nhỏ nhưng vẫn đạt được hiệu suất tương đương với các mô hình được tinh chỉnh hoàn toàn, cho phép huấn luyện các mô hình lớn hơn với ít tài nguyên hơn.

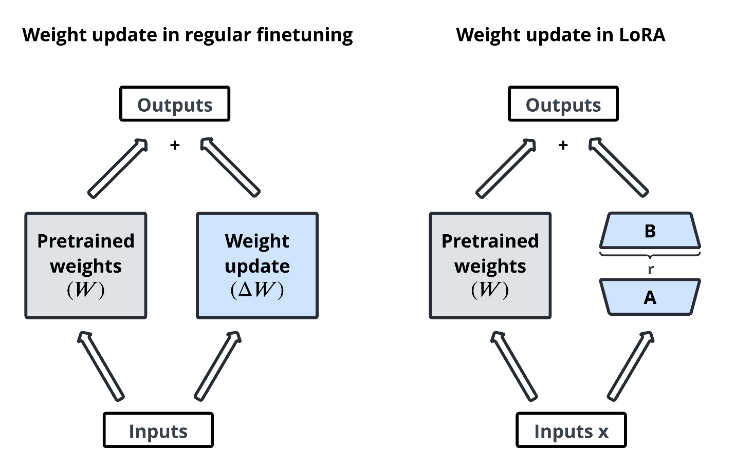
A diagram of a layer of information

Description automatically generated with medium confidence

Hình 2.21 Sơ đồ biểu diễn kiến trúc Adapter được sử dụng trong LLM

### LoRA

Low-Rank Adaptation là một kỹ thuật được thiết kế để fine-tuning mô hình ngôn ngữ lớn, sẽ sửa đổi quá trình fine-tuning bằng cách đóng băng các trọng số của mô hình gốc và áp dụng sự thay đổi cho một phần trong số riêng biệt, được thêm vào các tham số ban đầu. LoRA biến đổi tham số mô hình thành một không gian low-rank, giảm số lượng tham số có thể huấn luyện, đẩy nhanh việc xử lý, và giảm thiểu chi phí. Phương pháp này đặc biệt hữu dụng trong trường hợp nhiều khách hàng yêu cầu fine-tune mô hình cho các ứng dụng khác nhau, cho phép tạo các trọng số cụ thể cho từng trường hợp sử dụng mà không cần các mô hình riêng biệt. Bằng cách sử dụng phương pháp xấp xỉ thứ hạng thấp, làm giảm thiểu một cách hiệu quả yêu cầu về tài nguyên và tính toán trong khi vẫn giữ được khả năng thích ứng của mô hình pre-trained cho từng tác vụ cũng như lĩnh vực cụ thể.



Hình 2.22 Sơ đồ cập nhật trọng số trong LoRA

1. **Lợi ích của việc sử dụng LoRA**

* Hiệu quả tham số: LoRA giảm đáng kể số lượng tham số cần thiết cho quá trình huấn luyện bằng việc chỉ tập trung vào ma trận low-rank, làm cho yêu cầu về bộ nhớ và lưu trữ sẽ ít hơn khi so sánh với full fine-tuning.
* Hiệu quả lưu trữ: Lưu trữ một mô hình đã được huấn luyện là hiệu quả hơn vì nó chỉ yêu cầu lưu trữ các ma trận low-rank thay vì toàn bộ trọng số mô hình
* Giảm tải tính toán: Huấn luyện với ma trận low-rank yêu cầu ít tài nguyên tính toán hơn, giúp nó nhanh chóng và nhỏ gọn hơn.
* Bộ nhớ ít hơn: Cùng với việc ít các tham số được cập nhật hơn, bộ nhớ trong quá trình huấn luyện sẽ được giảm bớt, giúp cho việc sử dụng được batch size lớn hơn hoặc mô hình phức tạp hơn trong cùng một ràng buộc về phần cứng.
* Khả năng linh hoạt: LoRA có thể dễ dàng tích hợp với các mô hình pre-trained có sẵn mà không cần sự điều chỉnh quá lớn nào đối với kiến trúc mô hình
* Khả năng tương thích: Nó có thể được sử dùng theo cùng các kỹ thuật fine-tuning khác, chẳng hạn như các lớp adapters hoặc prompt-tuning, để nâng cao hiệu suất hơn nữa.
* Kết quả tương đương: Mặc dù có sự giảm thiểu về số lượng các tham số huấn luyện, LoRA vẫn thể hiện hiệu suất tương đương với full fine-tuning trong nhiều các tác vụ.
* Task-Specific Adaptation: Nó giúp pre-trained model thích ứng hiệu quả với các tác vụ cụ thể, bằng việc tận dụng các hiểu biết có sẵn từ mô hình gốc.
* Tránh overfitting: Bằng việc tập trung vào low-rank update, LoRA giúp giảm thiểu overfitting, đặc biệt là phải xử lý với bộ dữ liệu nhỏ hơn.

1. **Hạn chế của LoRA**

Trong khi LoRA vẫn được xem là rất mạnh mẽ trong fine-tuning, tuy nhiên nó vẫn gặp phải một số cách thách thức:

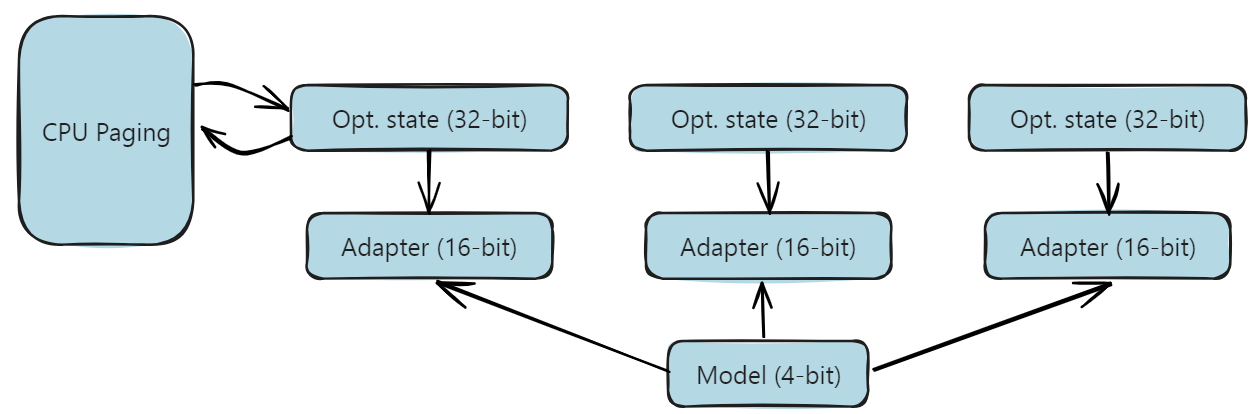
* Phạm vi fine-tuning (Fine-tuning Scope): LoRA có thể gặp khó khăn khi được áp dụng vào các tác vụ yêu cầu sự thay đổi đáng kể với các đại diện bên trong của mô hình pretrained.
* Tối ưu các siêu tham số (Hyperparameter Optimisation): Tinh chỉnh rank parameter ‘r’ yêu cầu điều chỉnh tỉ mỉ để tối ưu hiệu suất.
* Tiếp tục nghiên cứu (Ongoing Research): Tuy hứa hẹn, nhưng LoRA vẫn đang trong quá trình nghiên cứu, và ý nghĩa cuối cùng của nó vẫn chưa được khám phá hoàn toàn.

Mặc dù có những thách thức, nhưng LoRA vẫn là ứng viên tiềm năng cho việc tinh chỉnh các mô hình LLMs. Việc tiếp tục nghiên cứu và phát triển mạng lại triển vọng để khắc phục những hạn chế hiện tại và đạt được hiệu quả và khả năng thích ứng cao hơn nữa

### QLoRA

QLoRA là phiên bản mở rộng của LoRA được thiết thế mang lại bộ nhớ cao hơn trong các mô hình ngôn ngữ lớn bằng cách lượng tử hóa các tham số (quantising weight parameters) thành độ chính xác 4 bit. Thông thường, các tham số LLM được lưu ở định dạng 32 bit, nhưng QLoRA nén chúng thành 4 bit, giảm thiểu đáng kể dung lượng bộ nhớ. Nó cho phép fine-tuning trên phần cứng kém chất lượng hơn, bao gồm cả consumer GPUs. QLoRA cũng định lượng (quantises) trọng số của các LoRA adapters từ 8bit thành 4bit, giúp giảm hơn nữa yêu cầu về bộ nhớ và lưu trữ. Mặc dù độ chính xác của bit bị giảm, QLoRA vẫn duy trì hiệu suất ở mức tương đương với tỉnh chỉnh 16bit truyền thống.

Nó đạt được điều này bằng cách lan truyền ngược gradient thông qua mô hình ngôn ngữ pretrained được lượng tử hóa 4 bit bị đóng băng vào các Low-rank Adapters, làm cho việc fine-tuning trở lên hiệu quả trong khi vẫn giữ được hiệu suất mô hình. Cấu hình QLoRA được hỗ trợ bởi HuggingFace thông qua thư viện PEFT, sử dụng LoraConfig và BitsAndBytesConfig cho lượng từ hóa. Những cải tiến như dạng dữ liệu 4 bit tối ưu, lượng tử hóa kép các hằng số, và quản lý tăng đột biến bộ nhớ cho phép QLoRA giảm thiểu mức sử dụng bộ nhớ từ 96 bit cho mỗi tham số trong fine-tuning truyền thống thành 5.2 bit cho mỗi tham số, giảm 18 lần.

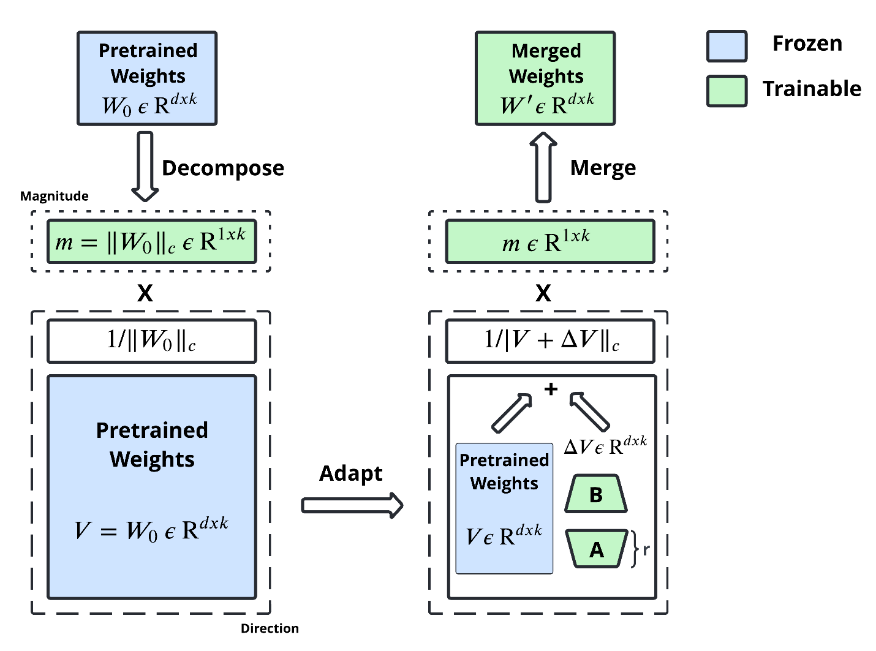


Hình 2.23 Quy trình tối ưu hóa QLoRA

### DoRA

Trong bối cảnh tối ưu hóa quá trình fine-tuning mô hình, phân tích LoRA và Full Fine-Tuning cho thấy sự khác biệt đáng kể trong hành vi học tập và cập nhật trọng số. LoRA sử dụng chiến lượng cập nhật pre-trained weight sử dụng tích của 2 ma trận low-rank, giúp giữ nguyên các trọng số gốc trong quá trình fine-tuning, cho phép suy diễn hiệu quả. Mặc dù có hiệu quả tính toán cao, các nghiên cứu trước đây lại cho rằng số lượng tham số tham gia huấn luyện  hạn chế của LoRA có thể góp phần vào việc làm khác biệt hiệu suất của nó so với full fine tuning.

Weight-Decomposed Low-Rank Adaption (DoRA) là một phương pháp fine-tuning mới được thiết thế để tối ưu mô hình pre-trained bằng việc phân rã các trọng số của chúng trở thành các thành phần biên độ và hướng. Cách tiếp cận này tận dụng độ hiệu quả của Low-Rank Adaptation (LoRA) cho hướng cập nhật, tạo điều kiện cho việc cập nhật một lượng đáng kể các tham số mà không thay đổi toàn bộ kiến trúc mô hình. DoRA giải quyết các thách thức về tính toán của các các fine-tuning truyền thống bằng cách duy trì tính đơn giản của mô hình và hiệu quả trong suy luận, đồng thời thu hẹp khoảng cách của hiệu suất thường thấy của LoRA và Full Fine Tuning. Các đánh giá theo lý thuyết và thực nghiệm đã chứng minh rằng DoRA không chỉ đạt được kết quả học đầu ra có thể so sánh được với Full Fine Tuning trong nhiệm tác vụ khác nhau – bao gồm xử lý ngôn ngữ tự nhiên và ứng dụng  ngôn ngữ thị giá, mà còn vượt qua LoRA về mặt hiệu suất, cung cấp giải pháp mạnh mẽ để nâng cao khả năng thích ứng và hiệu quả của các mô hình quy mô lớn.



Hình 2.24 Quá trình cập nhật trọng số DoRA

Tổng quan về DoRA (Decomposed Representation for Adapters), là một phương thức cho việc thích ứng hạng thấp phân tách trọng số. Hình minh họa cách các trọng số pre-trained phân tách và điều chỉnh trong quá trình fine-tuning. Ở phần bên trái các trọng số được tách thành các biên độ và hướng. Phần bên phải cho thấy cách các trong số đã phân tách này kết hợp cùng với các tham số có thể huấn luyện được trong quá trình fine-tuning, tạo ra các trọng số được cập nhật – bao gồm cả các thành phần đóng băng (màu xanh dương) và các thành phần có thể huấn luyện (màu xanh lá). Quy trình này nhấn mạng khả năng thích ứng bằng cách tập trung vào hướng quan trọng nhất trong không gian tham số, giúp fine-tuning hiệu quả trong khi vẫn duy trì được tính toàn vẹn của mô hình gốc.

* 1. **Lợi ích của DoRA**
* Cải thiện khả năng học: DoRA đạt được khả năng học gần như Full Fine-Tuning bằng cách phân tách trọng số pretrained thành các thành phần biên độ và hướng, cho phép cập nhật được nhiều sắc thái hơn.
* Fine-tuning hiệu quả: Bằng việc sử dụng các cấu trúc tiên tiến của Low-Rank Adaptation (LoRA) cho cập nhật theo hướng, DoRA cho phép fine-tuning hiệu quả mà không cần chỉnh sửa toàn bộ cấu trúc mô hình.
* Không có độ trễ khi suy luyện thêm: Mặc dù khả năng học của nó được cải thiện, DoRA vẫn không có thêm bất kì độ trễ nào trong quá trình suy luận như LoRA, duy trì tính đơn giản và hiệu quả của mô hình.
* Hiệu suất vượt trội : Các kết quả thực nghiệm đã cho thấy rằng DoRA thực sự vượt trội hơn LoRA trong phạm vi nhiệm vụ lớn, bao gồm xử lý ngôn ngữ tự nhiên , tinh chỉnh hướng dẫn bằng hình ảnh, hình ảnh/video – text understanding. Ví dụ, nó thể hiện tính cải thiện đáng kể trong các tiêu chuẩn điều chỉnh lý luận thông thường và hướng dẫn trực quan.
* Tính linh hoạt dựa trên backbone: DoRA đã được xác thực trên nhiêu backbone các mô hình khác nhau, bao gồm các mô hình ngôn ngữ lớn (LLM) và mô hình ngôn ngữ tâm nhìn cho thấy khả năng ứng dụng rộng rãi và mạnh mẽ của nó trong các lĩnh vực khác nhau.
* Phân tích đổi mới :Việc giới thiệu phân tích phân rã trọng số giúp khám phá những khác biệt cơ bản trong mô hình học tập của Full Fine Tuning và các phương pháp PEFT khác nhau, góp phần hiểu sâu hơn về sự linh hoạt trong cách fine-tune mô hình.
  1. **So sánh LoRA và DoRA**

Low-Rank Adaption (LoRA) và Weight-Decomposed Low-Rank Adaption (DoRA) cả 2 đều sử dụng các kỹ thuật được thiết kế nhằm cải thiện độ hiệu quả để fine-tune các mô hình pre-trained lớn. trong khi chúng chia sẻ mục tiêu chung là giảm thiểu khả năng tính toán, chúng đã sử dụng các kĩ thuật khác nhau để giải quyết vấn đề này

|  |  |  |
| --- | --- | --- |
| Tiêu chuẩn | LoRA | DoRA |
| Mục tiêu | Cung cấp một phương pháp hiệu quả để fine-tune mô hình pre-trained bằng việc sử dụng tích của ma trận low-rank để tính toán cập nhật trọng số mà không có độ trễ trong quá trình suy luận | Cải thiện khả năng học bằng việc bắt chước gần giống với việc học trong full fine-tuning, tối ưu các biên độ và hướng một cách riêng biệt |
| Hướng tiếp cận | Triển khai phân rã low-rank (low-rank decomposition) bằng việc cập nhật các trọng số được mô hình hóa dưới dạng tích 2 ma trận low rank (B và A), giữ nguyên trọng số ban đầu ở trạng thái tĩnh. | Sử dụng phân tích phân rã trọng số để tham số hóa lại các trọng số của ma trận thành các thành phần biên độ và hướng riêng biệt cho mỗi lần cập nhật khác nhau. |
| Kiến trúc mô hình | Giữ  nguyên các trọng số đã pretrained (W0) và áp dụng cập nhật bằng việc sử dụng ma trận low-rank (B và A). Ma trận A được tạo lập bằng phân phối Kaiming thống nhất, trong khi B được khởi tạo bằng ma trận 0. | Cấu trúc lại ma trận trọng số thành các thành phần biên độ và hướng, đảm bảo các vector hướng là các đơn vị chi tiết hơn cho quá trình điều chỉnh. |

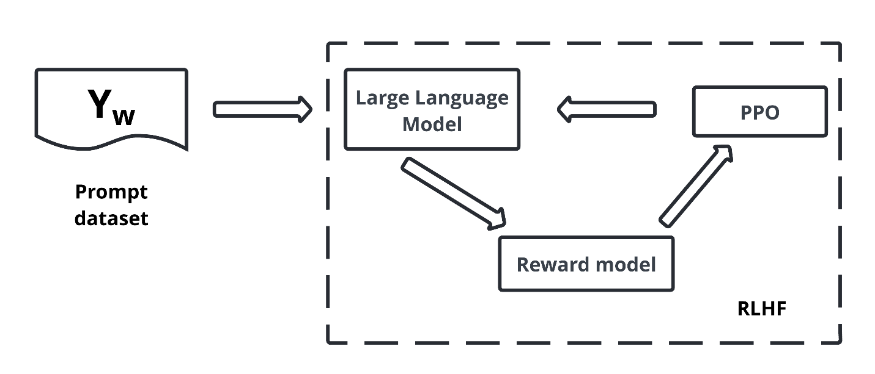
Bảng 2.1 Bảng so sánh LoRA và DoRA

### Proximal Policy Optimization

PPO là một thuật toán học tăng cường (reinforcement learning algorithm) được sử dụng để huấn luyện các agents thực hiện các tác vụ trong nhiều các môi trường đa dạng. Thuật toán này tận dụng phương pháp policy gradient trong đó các luật - được biểu diễn bằng các mạng nơ ron – giúp quyết định hành vi của agent dựa trên trạng thái hiện tại. PPO xử lý hiệu quá tính chất động của dữ liệu huấn luyện, được tạo ra thông qua tương tác liên tục giữa agents và môi trường, một đặc điểm nổi bật so với tập dữ liệu tĩnh được sử dụng trong học có giám sát.

Sự cải tiến của PPO nằm ở hàm mục tiêu “đại diện”, được tối ưu hóa thông qua phương pháp stochastic gradient. Cách tiếp cận này cho phép cập nhật nhiều lần từ cùng một lô dữ liệu ,nâng cao sự hiệu quả và sự ổn định của quá trình huấn luyện so với các phương pháp gradient chính sách truyền thống. Được phát triển bởi OpenAI, PPO được thiết kế nhằm cân bằng giữa tính dễ triển khai với đặc điểm hiệu suất mạnh mẽ của các thuật toán phức tạp như Trust Region Policy Optimization (TRPO), những không có độ phức tạp tính toán cao kèm theo.

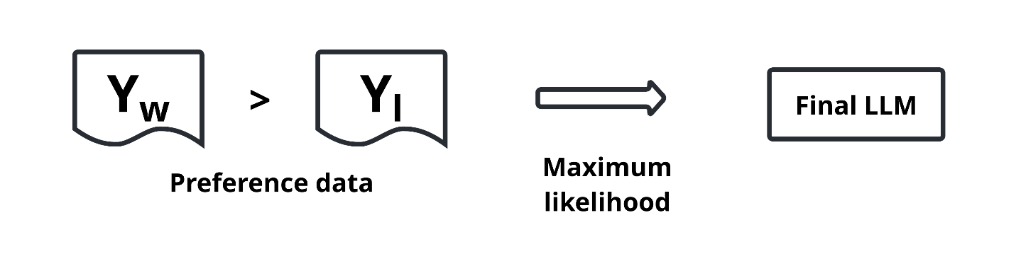
PPO hoạt động bằng cách tối đa hóa phần thưởng tích lũy kỳ vọng thông qua các điều chỉnh chính sách lặp đi lặp lại, làm tăng khả năng các hành động dẫn đến phần thưởng cao hơn. Một đặc điểm quan trọng của PPO là cơ chế clipping trong hàm mục tiêu, giúp hạn chế mức độ cập nhật chính sách, từ đó ngăn chặn các thay đổi đột ngột và duy trì sự ổn định trong quá trình huấn luyện.



Hình 2.25 Sơ đồ PPO

### Direct Preference Optimization

Direct Preference Optimisation (DPO) cung cấp một phương pháp đơn giản hóa để căn chỉnh LMs với sở thích của con người, bỏ qua sự phức tạp của RLHF. Các mô hình ngôn ngữ lớn không giám sát (Large-Scale unsupervised LMs) thường thiếu đi khả năng kiểm soát hành vi chính xác, do đó cần các phương pháp như LLM để finetune mô hình sử dụng phản hồi từ con người. Tuy nhiên RLHF khá phức tạp, liên quan đến việc tạo ra mô hình reward và fine-tune LM để đạt được các rewards ước lượng tối đa, điều này có thể không ổn định và yêu cầu tính toán cao. DPO giải quyết những thách thức này bằng cách trực tiếp tối ưu các mô hình LMs với một mục tiêu phân loại đơn giản, căn chỉnh các phản hồi với sở thích của con người. Hướng tiếp cận này loại bỏ nhu cầu về explicit reward model và điều chỉnh một lượng siêu tham số lớn, cải thiện tính ổn định và độ hiệu quả. DPO tối ưu các hành vi mong muốn bằng việc tăng các phản hồi có liên quan được ưa thích hơn, đồng thời tích hợp trọng số tầm quan trọng động để ngăn chặn sự suy thoái của mô hình. Do đó DPO đơn giản hóa quá trình học theo sở thích, trở thành một phương pháp hiệu quả cho quá trình huấn luyện LMs từ sở thích con người.



Hình 2.26 Luồng xử lý Direct Preference Optimization

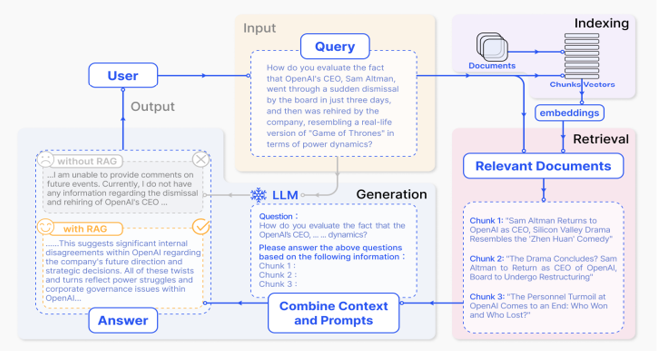
## Retrieval Augmented Generation

### Khái niệm RAG

Retrieval-Augmented Generation (RAG) là một kỹ thuật giúp nâng cao khả năng của mô hình sinh kết hợp với tri thức bên ngoài. Phương pháp này thực hiện bằng cách truy xuất thông tin liên quan từ kho tài liệu (tri thức) và sử dụng chúng cho quá trình sinh câu trả lời dựa trên LLMs.

### Ứng dụng

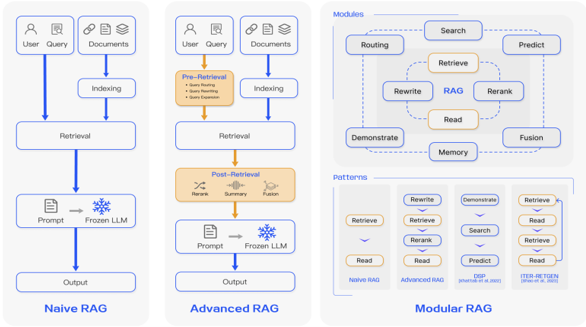
Một ứng dụng điểm hình của RAG là được áp dụng cho trả lời câu hỏi. Ví dụ như: khi người dùng đặt câu hỏi cho ChatGPT về một tin tức gần đây, được thảo luận rộng rãi. Do ChatGPT phụ thuộc vào dữ liệu đào tạo trước, ban đầu nó không có khả năng cung cấp thông tin cập nhật về các diễn biến gần đây. RAG thu hẹp khoảng cách thông tin này bằng cách tìm nguồn và kết hợp kiến thức về các cơ sở dữ liệu bên ngoài. Trong trường hợp này, nó thu thập các bài báo tin tức có liên quan đến truy vấn của người dùng. Các bài báo này, kết hợp với câu hỏi ban đầu tạo thành một lời nhắc toàn diện giúp LLM tạo ra câu trả lời có thông tin đầy đủ.



Hình 2.27 Tổng quan kiến trúc RAG

### Các mô hình RAG

Mô hình nghiên cứu RAG liên tục phát triển và nó được phân loại thành ba mô hình: Navie RAG, advanced RAG, modular RAG.



Hình 2.28 Các mô hình RAG

* 1. **Naïve RAG**

Naïve RAG tuân theo quy trình lập chỉ mục, truy xuất và tạo dữ liệu truyền thống đã đề cập ở trên. Tóm lại, dữ liệu đầu vào của người dùng được sử dụng để truy vấn các tài liệu có liên quan, sau đó được kết hợp với lời nhắc và chuyển đến mô hình để tạo phản hồi cuối cùng. Lịch sử hội thoại có thể được tích hợp vào lời nhắc nếu ứng dụng liên quan đến tương tác đối thoại nhiều lượt.

Naive RAG có những hạn chế như độ chính xác thấp (các khối lấy ra không thẳng hàng) và khả năng thu hồi thấp (không lấy ra được tất cả các khối có liên quan). Cũng có khả năng LLM được truyền thông tin lỗi thời, đây là một trong những vấn đề chính mà hệ thống RAG ban đầu nên hướng đến để giải quyết. Điều này dẫn đến các vấn đề ảo giác và phản hồi kém và không chính xác.

Khi áp dụng tăng cường, cũng có thể có vấn đề về sự trùng lặp và lặp lại. Khi sử dụng nhiều đoạn trích được lấy, xếp hạng và điều hòa phong cách/giọng điệu cũng là chìa khóa. Một thách thức khác là đảm bảo rằng tác vụ tạo không phụ thuộc quá nhiều vào thông tin tăng cường, điều này có thể dẫn đến việc mô hình chỉ lặp lại nội dung đã lấy.

* 1. **Advanced RAG**

RAG nâng cao giúp giải quyết các vấn đề tồn tại trong RAG ngây thơ như cải thiện chất lượng truy xuất có thể bao gồm việc tối ưu hóa các quy trình trước khi truy xuất, truy xuất và sau khi truy xuất.

Quá trình trước khi truy xuất bao gồm việc tối ưu hóa lập chỉ mục dữ liệu nhằm nâng cao chất lượng dữ liệu được lập chỉ mục thông qua năm giai đoạn: tăng cường độ chi tiết của dữ liệu, tối ưu hóa cấu trúc chỉ mục, thêm siêu dữ liệu, tối ưu hóa căn chỉnh và truy xuất hỗn hợp.

Giai đoạn truy xuất có thể được cải thiện hơn nữa bằng cách tối ưu hóa chính mô hình nhúng, tác động trực tiếp đến chất lượng của các khối tạo nên ngữ cảnh. Điều này có thể được thực hiện bằng cách tinh chỉnh nhúng để tối ưu hóa tính liên quan của truy xuất hoặc sử dụng nhúng động nắm bắt tốt hơn sự hiểu biết theo ngữ cảnh (ví dụ: mô hình nhúng-ada-02 của OpenAI).

Tối ưu hóa sau khi truy xuất tập trung vào việc tránh giới hạn cửa sổ ngữ cảnh và xử lý thông tin nhiễu hoặc có khả năng gây mất tập trung. Một cách tiếp cận phổ biến để giải quyết các vấn đề này là xếp hạng lại, có thể bao gồm các cách tiếp cận như di dời ngữ cảnh có liên quan đến các cạnh của lời nhắc hoặc tính toán lại độ tương đồng về mặt ngữ nghĩa giữa truy vấn và các khối văn bản có liên quan. Nén lời nhắc cũng có thể giúp giải quyết các vấn đề này.

* 1. **Modular RAG**

Như tên gọi của nó, Modular RAG tăng cường các mô-đun chức năng như kết hợp mô-đun tìm kiếm để truy xuất sự tương đồng và áp dụng tinh chỉnh trong trình truy xuất. Cả Naive RAG và Advanced RAG đều là các trường hợp đặc biệt của Modular RAG và được tạo thành từ các mô-đun cố định. Các mô-đun RAG mở rộng bao gồm tìm kiếm, bộ nhớ, hợp nhất, định tuyến, dự đoán và bộ điều hợp tác vụ giúp giải quyết các vấn đề khác nhau. Các mô-đun này có thể được sắp xếp lại để phù hợp với các bối cảnh vấn đề cụ thể. Do đó, Modular RAG được hưởng lợi từ tính đa dạng và linh hoạt hơn ở chỗ bạn có thể thêm hoặc thay thế các mô-đun hoặc điều chỉnh luồng giữa các mô-đun dựa trên yêu cầu tác vụ.

Với tính linh hoạt tăng lên trong việc xây dựng các hệ thống RAG, các kỹ thuật tối ưu hóa quan trọng khác đã được đề xuất để tối ưu hóa các đường ống RAG bao gồm:

* Khám phá tìm kiếm kết hợp: Phương pháp này tận dụng sự kết hợp của các kỹ thuật tìm kiếm như tìm kiếm dựa trên từ khóa và tìm kiếm ngữ nghĩa để truy xuất thông tin có liên quan và giàu ngữ cảnh; điều này hữu ích khi xử lý các loại truy vấn và nhu cầu thông tin khác nhau.
* Công cụ truy vấn và truy xuất đệ quy: Bao gồm quá trình truy xuất đệ quy có thể bắt đầu bằng các khối ngữ nghĩa nhỏ và sau đó truy xuất các khối lớn hơn làm phong phú thêm ngữ cảnh; điều này hữu ích để cân bằng hiệu quả và thông tin phong phú về ngữ cảnh.
* StepBack-prompt: [Một kỹ thuật nhắc nhở](https://arxiv.org/abs/2310.06117) cho phép LLM thực hiện trừu tượng hóa tạo ra các khái niệm và nguyên tắc hướng dẫn lý luận; điều này dẫn đến những phản hồi có cơ sở hơn khi áp dụng vào khuôn khổ RAG vì LLM tránh xa các trường hợp cụ thể và được phép lý luận rộng hơn nếu cần.
* Truy vấn phụ: Có nhiều chiến lược truy vấn khác nhau như truy vấn cây hoặc truy vấn tuần tự các khối có thể được sử dụng cho các tình huống khác nhau. LlamaIndex cung cấp một công cụ truy vấn câu hỏi cho phép chia nhỏ một truy vấn thành nhiều câu hỏi sử dụng các nguồn dữ liệu có liên quan khác nhau.
* Nhúng tài liệu giả định: [HyDE](https://arxiv.org/abs/2212.10496) tạo ra câu trả lời giả định cho truy vấn, nhúng câu trả lời đó và sử dụng nó để truy xuất các tài liệu tương tự như câu trả lời giả định thay vì sử dụng trực tiếp truy vấn.

# XÂY DỰNG HỆ THỐNG CHATBOT

## Tổng quan ý tưởng hệ thống

Hệ thống chatbot thông minh phục phụ trường đại học công nghiệp hướng tới tự động hóa các tác vụ hỗ trợ thông tin cho mọi người nhắn tra cứu các thông tin liên quan đến thông tin trường, thông tin tuyển sinh, các dự án tuyển sinh và các thông tin liên quan khác. Qua việc nghiên cứu các phương pháp tinh chỉnh mô hình ngôn ngữ lớn LLM, mục tiêu của bài báo cáo là hướng tới việc sử dụng phương pháp tinh chỉnh phù hợp nhằm tạo ra chatbot có khả năng hiểu và phản hồi trong ngữ cảnh tiếng việt.

Trọng tâm của hệ thống là sử dụng mô hình ngôn ngữ lớn đã được tiền huấn luyện, sau đó áp dụng các phương pháp tinh chỉnh phù hợp để điều chỉnh mô hình có thể trả lời được về các thông tin của trường Đại học Công nghiệp Hà Nội. Qua đó hệ thống sẽ là một trợ lý số đắc lực giúp cải thiện cho bộ phận hỗ trợ và tăng cường sự hài lòng cho người dùng trong môi trường đại học.

## Chuẩn bị dữ liệu

### Giới thiệu bộ dữ liệu

Dữ liệu phục vụ huấn luyện được là bộ dữ liệu vilm/OpenOcra-Viet, đây là bộ dữ liệu được phát triển nhằm hỗ trợ nghiên cứu và huấn luyện các mô hình ngôn ngữ lớn trong ngữ cảnh tiếng Việt. Nó là bộ dự liên tiên tiến được tạo ra thông qua sự hợp tác tiên phong giữa Virtual Interactive và Alignment Lab AI. Lấy cảm hứng và phương pháp luận từ bài báo nổi tiếng của Ocra và được mở rộng để chắt lọc kiến thức từ sự kết hợp của các LLM hàng đầu bao gồm GPT-4, PaLM-2 và Claude. Bộ dữ liệu này nhằm thúc đẩy hoạt động nghiên cứu và phát triển nằm nâng cao hiệu suất của mô hình ngôn ngữ Tiếng Việt và các lĩnh vực chưa được khám phá. Nguồn gốc chinh của bộ nhiệm vụ/câu hỏi là bản dịch của FLAN, vi-FLAN và được tăng cường trên các LLM hiện đại hơn.

Bộ dữ liệu gồm hơn 121,000 điểm dữ liệu. Tuy nhiên chỉ sử dụng 10,000 dữ liệu cho việc huấn luyện mô hình. Với 9,900 điểm dùng cho tập train và 100 điểm cho tập test.

A computer code with black text

Description automatically generated with medium confidence

Hình 3.1 Chia tập dữ liệu

Bộ dữ liệu có 3 trường bao gồm instruction (hướng dẫn), input (đầu vào) và output (đầu ra). Với instruction là phần chứa nội dung yêu cầu hoặc câu lệnh từ người dùng giúp mô hình hiểu được vai trò của mình trong câu hỏi, ví dụ như bảo mô hình là một trợ lý giúp hỗ trợ việc truy tìm dữ liệu (trường này có thể trống). Input là câu hỏi cần thiết để thực hiện câu lệnh trong trường instruction. Và cuối cùng là output là câu trả lời hoặc phản hồi dựa trên instruction và input.

A screenshot of a computer

Description automatically generated

Hình 3.2 Thông tin chung về bộ dữ liệu vilm/OpenOcra-Viet

### Biến đổi dữ liệu

Dữ liệu sau đó cần được định dạng lại sao cho nội dung của từng mẫu dữ liệu là có thể hiểu được ở các mô hình hội thoại.

A screenshot of a computer program

Description automatically generated

Hình 3.3 Định dạng dữ liệu

Trường “data[‘chat\_format’]” là trường mới được sinh ra qua việc ánh xạ danh sách “message” được định dạng theo vai trò của các trường trong dữ liệu ban đầu. Trường instruction sẽ có vai trò là system. Trường input sẽ có vai trò là user. Và trường output sẽ có vai trò là assistant.

A screenshot of a computer

Description automatically generated

Hình 3.4 Dữ liệu sau khi định dạng

## Xây dựng back-end

Sử dụng framework FastAPI với khả năng xử lý CORS (Cross-Origin Resource Sharing) để xây dựng API trong Python. Nó được thiết kế để hỗ trợ xây dựng các API RESTful nhanh, hiệu quả và dễ bảo trì, phù hợp với các ứng dụng hiện đại. FastAPI sử dụng các tính năng như type hints trong Python và dựa trên chuẩn ASGI để đảm bảo hiệu năng cao.

A screenshot of a computer program

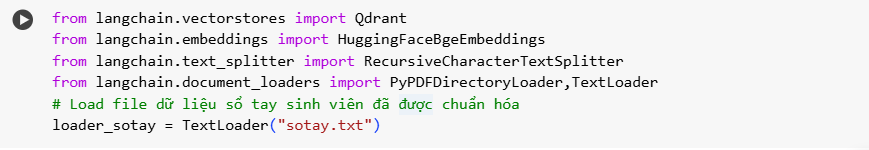
Description automatically generated

Hình 3.5 Xây dựng back-end

Ngoài ra, sử dụng ngrok để tạo một URL công khai cho server, cho phép việc truy cập từ bên ngoài mà không cần cấu hình mạng phức tạp. Cho phép việc phát triển và thử nghiệm ứng dụng back-end ở trong môi trường cục bộ.

### Xây dựng vector database

Vector database là ở đây là các thông tin liên quan đến trường Đại học Công nghiệp Hà Nội bao gồm về thông tin tuyển sinh, các quy chế, và các thông tin khác được thu thập từ trang thông tin điện tử của trường và được tích hợp vào 1 file có tên là “sotay.txt”. File này sau đó sẽ được cắt ra thành từng đoạn gọi là các “chunk” và được mã hóa bởi mô hình embedding của HuggingFace rồi được lưu lại trên vector database server.



Hình 3.6 Import các thư viện cần dùng

Qdrant là một cơ sở dữ liệu vector để lưu trữ các embedding của tài liệu. Giúp cho việc truy xuất tài liệu nhanh chóng dựa trên độ tương đồng của vector. Sử dụng HuggingFaceBgeEmbeddings để mã hóa các đoạn dữ liệu thành các vector embedding. ReCursiveCharacterTextSplitter giúp chia nhỏ file dữ liệu thành các “chunk” nhỏ để xử lý embeddings.

A screen shot of a computer code

Description automatically generated

Hình 3.7 Load và chia nhỏ file dữ liệu

Chia dữ liệu thành các chunk có độ dài là 800 tokens và overlap là 400 tokens sẽ giúp không bị mất đi nghĩa của các câu.

A screen shot of a computer program

Description automatically generated

Hình 3.8 Tạo embedding và lưu lại trên vector database server

Sử dụng mô hình “sentence-transformers/paraphrase-multilingual-mpnet-base-v2” để embeddings các chunk thành các vector. Rồi lưu lại trên vector database server.

### Xây dựng Retriever

Tạo các lớp Retriever bao gồm “VectorStoreRetriever” và “RerankWikiRetriever” giúp tìm kiếm các đoạn văn bản có nội dung liên quan nhất đến truy vấn. Bằng việc sử dụng các thư viện của LangChain để tính toán mức độ tương đồng trong câu truy vấn và vector database gồm vector database trong file “sotay.txt” của trường Đại học Công nghiệp Hà Nội và vector database của Wikipedia.

A screen shot of a computer code

Description automatically generated

Hình 3.9 Tạo các Retriever

## Lựa chọn mô hình ngôn ngữ lớn

Sử dụng thư viện Transformer của HuggingFace để gọi các lớp cần thiết để import mô hình ngôn ngữ lớn. Mô hình ngôn ngữ lớn được dùng ở đây để tinh chỉnh là mô hình “zephyr-7b-beta” là một trong chuỗi các mô hình ngôn ngữ được huấn luyện để hoạt động như một người trợ lý hữu ích. Nó là phiên bản đã được tinh chỉnh của mô hình “mistral/Mistral-7B-v0.1”, được huấn luyện bằng sự kết hợp của các dữ liệu tổng hợp công khai và sử dụng Direct Preference Optimization. Mô hình có 7 tỷ tham số được thiết kế giống với mô hình GPT.

A screenshot of a computer program

Description automatically generated

Hình 3.10 Khởi tạo mô hình

### Lượng tử hóa mô hình

Sử dụng QLoRA một kỹ thuật tinh chỉnh giúp tối ưu hóa bộ nhớ cho mô hình bằng việc giảm thiểu biểu diễn bit cho tham số xuống 8 bit.

A screenshot of a computer

Description automatically generated

Hình 3.11 Hàm tính số lượng tham số có thể huấn luyện được

Sau đó thiết lập cấu hình cho LoRA để chuẩn bị cho việc huấn luyện mô hình.

A screen shot of a computer program

Description automatically generated

Hình 3.12 Thiết lập cấu hình LoRA

### Huấn luyện mô hình

Thiết lập các siêu tham số tiến hành tinh chỉnh mô hình bằng phương pháp LoRA.

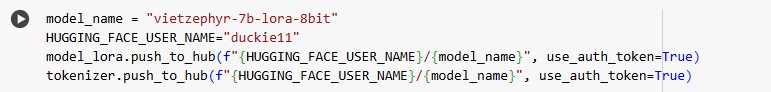
A screen shot of a computer code

Description automatically generated

Hình 3.13 Thiết lập các siêu tham số

### Triển khai mô hình

Sử dụng HuggingFace Hub, một nền tảng lưu trữ và chia sẻ mô hình, bộ dữ liệu và ứng dụng AI để lưu trữ mô hình và tokenizer đã tinh chỉnh lên.



Hình 3.14 Lưu trữ mô hình lên HuggingFace Hub

## Xây dựng front-end

React là một thư viện JavaScript mã nguồn mở, được phát triển bởi Meta (trước đây là Facebook), dùng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. React giúp việc phát triển các ứng dụng web trở nên nhanh chóng và hiệu quả nhờ vào khả năng tái sử dụng các thành phần giao diện, tính linh hoạt và hiệu suất cao.

Trong đồ án này, React được chọn làm công nghệ front-end chính để xây dựng giao diện người dùng cho chatbot. Việc sử dụng React giúp phân tách giao diện thành các thành phần nhỏ, dễ quản lý và bảo trì. Ngoài ra, React còn hỗ trợ cập nhật giao diện nhanh chóng nhờ vào cơ chế Virtual DOM, giúp tối ưu hiệu suất cho các ứng dụng có tính tương tác cao như chatbot.

A screenshot of a chat

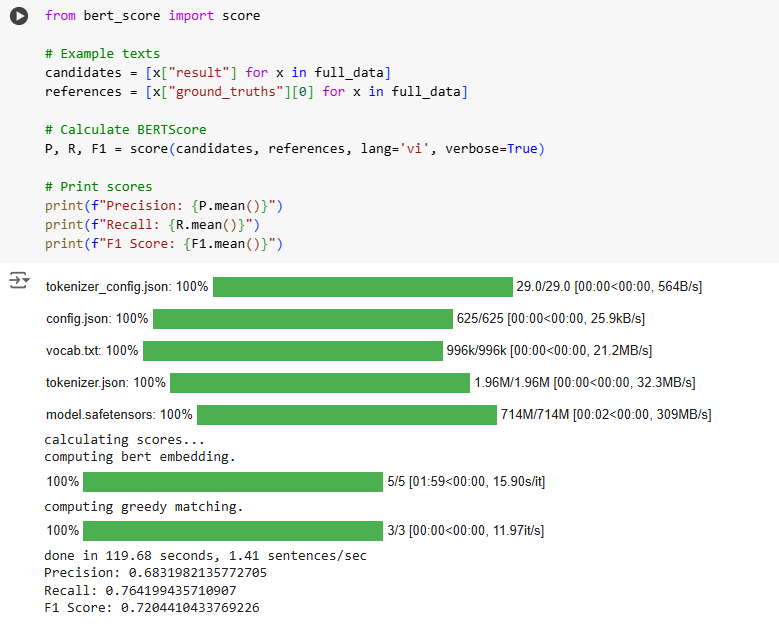
Description automatically generated

Hình 3.15 Giao diện hệ thống

# THỰC NGHIỆM VÀ ĐÁNH GIÁ

## Phương pháp đánh giá

Sử dụng bert\_score là một metric được sử dụng để đánh giá chất lượng các văn bản được tạo ra bởi các mô hình ngôn ngữ, dựa trên mức độ tương đồng ngữ nghĩa giữa văn bản dự đoán và văn bản tham chiếu. Qua đó dùng chúng để đánh giá các kết quả của mô hình sau khi được tinh chỉnh. Dữ liệu đánh giá được tạo bằng tay gồm hơn 100 cặp instructions và inputs đại diện cho câu hỏi và câu trả lời về các thông tin liên quan đến trường Đại học công nghiệp Hà Nội.



Hình 4.1 Các metric đánh giá mô hình

Hình trên cho ta thấy qua các thông số bert\_score ta có điểm Precision khoảng 0.683, Recall khoảng 0.764 và F1 Score khoảng 0.721 là những con số có thể chấp nhận được. Có thể thấy rằng mô hình đang cho các đầu ra khá giống với bộ dữ liệu đã được chuẩn bị để đánh giá.

## Kết quả thực nghiệm

Dưới đây là các sơ đồ phân bố của các thông số đánh giá.

* Precision:

A green graph with white text

Description automatically generated

Hình 4.2 Sơ đồ phân bố của Precision

* Recall:

A graph of a graph

Description automatically generated with medium confidence

Hình 4.3 Sơ đồ phân bố của Recall

* F1 Score:

A graph with blue lines

Description automatically generated

Hình 4.4 Sơ đồ phân bố F1 Score

## Đánh giá kết quả

Trong ngữ cảnh bài toán Chatbot cho trường học, các câu trả lời cho các câu hỏi cho học sinh hoặc phụ huynh cần đầy đủ, chính xác và nhanh chóng. Những thông tin sai hoặc bỏ sót có thể gây hiểu lầm.

Qua các thông số đánh giá, mô hình có điểm Recall khá tốt khoảng 0.764 cho thấy chatbot đã đáp ứng tốt yêu cầu không bỏ qua các thông tin quan trọng. Điều này rất phù hợp với bài toán trường học khi người dùng cần nhiều thông tin đa dạng. Cùng với đó F1-score cũng khá tốt khoảng 0.721 cho thấy chatbot có hiệu suất cân đối, đáp ứng phần lớn các nhu cầu người dùng. Tuy vậy, Precision của mô hình vẫn chưa cao khoảng 0.683 cho thấy đôi khi chatbot vẫn trả lời không chính xác hoặc cung cấp thông tin không phù hợp.

Có thể thấy rằng, mô hình vẫn cần cải thiện thêm để có thể có thể cải thiện độ chính xác trong các câu trả lời. Bằng việc tinh chỉnh thêm mô hình, thêm các câu trả lời chính xác và các câu hỏi phổ biến, thêm vào đó là kết hợp xử lý ngữ cảnh tốt hơn để giúp chatbot có thể hiểu rõ ý nghĩa câu hỏi trước khi trả lời.

# KẾT LUẬN

Trong quá trình nghiên cứu đề tài, tôi đã khảo sát nhu cầu cũng như thực trạng và các giải pháp đang được áp để triển khai một hệ thống chatbot phục vụ cho việc hỏi đáp bằng việc sử dụng các mô hình ngôn ngữ lớn đã được tiền huấn luyện, sau đó sẽ tinh chỉnh lại để phù hợp với nhiệm vụ hỏi đáp các thông tin liên quan đến Trường Đại học Công nghiệp Hà Nội.

Báo cáo trình bày đầy đủ về các kiến thức liên quan đến học sâu cũng như các kiến trúc mô hình ngôn ngữ lớn như Transformer cũng như quy trình huấn luyện các mô hình ngôn ngữ lớn.

Báo cáo cũng nêu rõ các kỹ thuật tinh chỉnh như LoRA, QLoRA, DoRA,.. giúp tinh chỉnh đạt hiệu quả cao trong việc tối ưu bộ nhớ giảm độ phức tạp tính toán. Cùng với đó sự kết hợp của RAG có thể thấy ra mô hình chatbot được đưa ra có thể trả lời được các câu hỏi về trường Đại học Công Nghiệp cũng như các kiến thức mà có thể truy xuất trong bộ dữ liệu Wikipedia.

Tuy nhiên do hạn chế về vấn đề thời gian cũng như tiềm lực kinh tế, bài báo cáo này chỉ hoàn thành dựa trên mức độ thực nghiệm. Mô hình được sử dụng ở báo cáo này là các mô hình nhỏ, và các dữ liệu dùng cho RAG chủ yếu là do xây dựng dựa trên các thông tin công khai trên trang điện tử của Trường Đại học Công nghiệp Hà Nội cùng với đó là dữ liệu trên Wikipedia. Nên các câu trả lời của chúng vẫn chưa thực sự tốt. Hướng đi cho đề tài này là việc thử nghiệm thêm các mô hình ngôn ngữ lớn hơn cùng việc tạo bộ dữ liệu đặc trưng lớn hơn cho việc huấn luyện cũng như truy xuất khi dùng RAG. Cùng với đó là sử dụng thêm các phương pháp học tăng cường như PPO và DPO để nâng cao hiệu suất của mô hình, giúp cho chatbot có thể trả lời các câu hỏi tự động một cách hiệu quả và nhanh chóng.

TÀI LIỆU THAM KHẢO

[1] *The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities (Version 1.0)*. (n.d.). <https://arxiv.org/html/2408.13296v1>

[2] *Dive into Deep Learning — Dive into Deep Learning 1.0.3 documentation*. (n.d.). https://d2l.ai/

[3] Aishwaryanr. (n.d.). *awesome-generative-ai-guide/free\_courses/Applied\_LLMs\_Mastery\_2024 at main · aishwaryanr/awesome-generative-ai-guide*. GitHub. https://github.com/aishwaryanr/awesome-generative-ai-guide/tree/main/free\_courses/Applied\_LLMs\_Mastery\_2024

[4] Alammar, J., & Grootendorst, M. (2024). *Hands-On Large Language Models*. O'Reilly Media, Inc.

[5] Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A comprehensive review of architectures, variants, and applications. *Information*, *15*(9), 517. <https://doi.org/10.3390/info15090517>

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *arXiv (Cornell University)*, *30*, 5998–6008. https://arxiv.org/pdf/1706.03762v5

[7] Schmidt, R. M. (2019). Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1912.05911>

[8] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023, December 18). *Retrieval-Augmented Generation for Large Language Models: A survey*. arXiv.org. https://arxiv.org/abs/2312.10997

[9] Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., & McHardy, R. (2023, July 19). *Challenges and applications of large language models*. arXiv.org. https://arxiv.org/abs/2307.10169

[10] Sang T. Truong, Duc Q. Nguyen, Toan Nguyen, Dong D. Le, Nhi N. Truong, Tho Quan, Sanmi Koyejo (26, May, 2024). *Crossing Linguistic Horizons:* *Finetuning and Comprehensive Evaluation of Vietnamese Large Language Model*. https://arxiv.org/html/2403.02715v2#S2

[11] Muhammad Usman Hadi, Qasem Al-Tashi, Rizwan Qureshi, Abbas Shah, Amgaf Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Mohammed Ali Al-Garadi, Jia Wu, Seyedeli Mirijalili, and Mubarak Shah (August, 2024). *LLMs: A Comprehensive Survey of Applications, Challenges, Datasets, Limitations, and Future Prospects*. https://www.researchgate.net/publication/383058502\_Large\_Language\_Models\_A\_Comprehensive\_Survey\_of\_its\_Applications\_Challenges\_Limitations\_and\_Future\_Prospects