|  |  |
| --- | --- |
|  |  |
| **NGUYỄN 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 MẠNG TRANSFORMER VÀ PHÁT TRIỂN CHATBOT TÍCH HỢP RAG** |
|  |
| **KHOA HỌC MÁY TÍNH** | **CBHD : *TS. Trần Hùng Cường*** |
| **Sinh viên : Nguyễn Minh Đức**  **Mã số sinh viên : *2021607287*** |
|  |
| Hà Nội – Năm 2025 |
|  |

# **LỜI CAM ĐOAN**

Em xin cam đoan các kết quả trình bày trong đồ án là công trình nghiên cứu của bản thân trong thời gian học tập tại Trường Đại học Công nghiệp Hà Nội dưới sự hướng dẫn của giáo viên hướng dẫn. Các số liệu, kết quả trình bày trong đồ án là hoàn toàn trung thực. Các kết quả sử dụng tham khảo đều đã được trích dẫn đầy đủ và theo đúng quy định.

Em sẽ 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, em xin chịu hoàn toàn trách nhiệm trước cơ quan quản lý và hội đồng đánh giá.

Chân thành cảm ơn.

*Hà Nội, ngày 10 tháng 5 năm 2025*

Sinh viên thực hiện

**Nguyễn Minh Đức**

**LỜI CẢM ƠN**

Lời đầu tiên cho phép em gửi lời cảm ơn sâu sắc tới các thầy cô trong khoa Công nghệ thông tin - Trường Đại học Công Nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn cho em những kiến thức, những bài học quý báu và bổ ích. Đặc biệt em xin được bày tỏ sự tri ân và xin chân thành cảm ơn Tiến sĩ **Trần Hùng Cường** là người đã tin tưởng và cho phép em thực hiện đề tài: “Nghiên cứu mạng Transformer và phát triển chatbot tích hợp RAG” cũng như là người trực tiếp hướng dẫn, chỉ bảo em trong suốt quá trình, nghiên cứu và hoàn thành được đồ án.

Em cũng xin bày tỏ lòng biết ơn 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.

Trong quá trình nghiên cứu và thực hiện, do năng lực, kiến thức, trình độ bản thân em còn hạn hẹp nên không tránh khỏi những thiếu sót và em mong mỏi nhận được sự thông cảm và những góp ý từ quý thầy cô.

Em xin chân thành cảm ơn! *Hà Nội, ngày 10 tháng 5 năm 2025*

Sinh viên thực hiện

**Nguyễn Minh Đức**

**MỤC LỤC**

[**LỜI CAM ĐOAN** i](#_Toc198133345)

[**DANH MỤC TỪ VIẾT TẮT** v](#_Toc198133346)

[**DANH MỤC CÁC BẢNG** vi](#_Toc198133347)

[**DANH MỤC HÌNH ẢNH** vii](#_Toc198133348)

[**LỜI NÓI ĐẦU** 1](#_Toc198133349)

[**LÝ DO CHỌN ĐỀ TÀI** 1](#_Toc198133350)

[**MỤC TIÊU NGHIÊN CỨU** 1](#_Toc198133351)

[**ĐỐI TƯỢNG NGHIÊN CỨU** 2](#_Toc198133352)

[**PHẠM VI NGHIÊN CỨU** 2](#_Toc198133353)

[**KẾT QUẢ MONG MUỐN** 2](#_Toc198133354)

[**CẤU TRÚC BÁO CÁO** 2](#_Toc198133355)

[**CHƯƠNG I: CƠ SỞ LÝ THUYẾT** 3](#_Toc198133356)

[**1.1. Tổng quan về chatbot** 3](#_Toc198133357)

[**1.1.1. Khái niệm** 3](#_Toc198133358)

[**1.1.2. Lịch sử phát triển** 3](#_Toc198133359)

[**1.1.3. Phân loại** 4](#_Toc198133360)

[**1.1.4. Ứng dụng** 5](#_Toc198133361)

[**1.1.5. Thách thức trong việc triển khai chatbot** 5](#_Toc198133362)

[**1.2. Mô hình ngôn ngữ lớn** 6](#_Toc198133363)

[**1.2.1. Khái niệm mô hình ngôn ngữ lớn** 6](#_Toc198133364)

[**1.2.2. Lịch sử phát triển** 7](#_Toc198133365)

[**1.3. Mạng nơ ron** 9](#_Toc198133366)

[**1.4. Word Embedding** 12](#_Toc198133367)

[**1.5. Transformers** 15](#_Toc198133368)

[**1.5.1. Tổng quan** 15](#_Toc198133369)

[**1.5.2. Lịch sử hình thành** 15](#_Toc198133370)

[**1.5.3. Kiến trúc** 17](#_Toc198133371)

[**1.5.4. Quy trình hoạt động** 19](#_Toc198133372)

[**1.6. Một số mô hình nổi tiếng sử dụng kiến trúc transformer** 30](#_Toc198133373)

[**1.6.1. BERT - Bidirectional Encoder Representations from Transformers** 30](#_Toc198133374)

[**1.6.2. GPT - Generative Pre-training Transformers** 31](#_Toc198133375)

[**1.6.3. LLAMA - Large Language Model Meta AI** 32](#_Toc198133376)

[**1.7. Phương pháp đào tạo mô hình LLM** 33](#_Toc198133377)

[**1.7.1. Huấn luyện từ đầu (Training from Scratch)** 34](#_Toc198133378)

[**1.7.2. Tinh chỉnh mô hình (Fine-tunning)** 35](#_Toc198133379)

[**1.8. Retrieval Augmented Generation** 35](#_Toc198133380)

[**1.8.1. Tổng quan** 35](#_Toc198133381)

[**1.8.2. Quy trình hoạt động** 37](#_Toc198133382)

[**1.8.3. Ưu nhược điểm** 41](#_Toc198133383)

[**CHƯƠNG II: PHÁT TRIỂN CHATBOT** 42](#_Toc198133384)

[**2.1. Tổng quan ý tưởng hệ thống** 42](#_Toc198133385)

[**2.2. Chuẩn bị dữ liệu** 42](#_Toc198133386)

[**2.2.1. Thu thập dữ liệu** 42](#_Toc198133387)

[**2.2.2. Đẩy dữ liệu lên cơ sở dữ liệu vector** 43](#_Toc198133388)

[**2.3. Truy xuất dữ liệu và tạo câu trả lời** 46](#_Toc198133389)

[**2.3.1. Xây dựng công cụ tìm kiếm thông tin** 46](#_Toc198133390)

[**2.3.2. Tạo câu trả lời** 47](#_Toc198133391)

[**2.4. Xây dựng giao diện và các chức năng hệ thống với Django** 48](#_Toc198133392)

[**2.4.1. Lý do lựa chọn Django Framework** 48](#_Toc198133393)

[**2.4.2. Kiến trúc tổng quan hệ thống web** 48](#_Toc198133394)

[**2.4.3. Một số hình ảnh kết quả** 49](#_Toc198133395)

[**CHƯƠNG III: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN** 52](#_Toc198133396)

[**3.1. Kết quả đạt được** 52](#_Toc198133397)

[**3.2. Hạn chế** 54](#_Toc198133398)

[**3.2. Hướng phát triển** 54](#_Toc198133399)

[**KẾT LUẬN** 55](#_Toc198133400)

# **DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| LLM/LLMs | Large Language Model | Mô hình ngôn ngữ lớn |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| RAG | Retrieval-Augmented Generation | Tạo tăng cường truy xuất |

# **DANH MỤC CÁC BẢNG**

[Bảng 1.1: Ví dụ về word embedding 13](#_Toc198131708)

[Bảng 1.2: Minh họa dữ liệu với one-hot encoding 13](#_Toc198131709)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1: Minh họa mạng nơ-ron 7](#_Toc198133407)

[Hình 1.2: Lược sử các mô hình từ 1920 đến nay 8](#_Toc198133408)

[Hình 1.3: Minh họa nơ ron 10](#_Toc198133409)

[Hình 1.4: Mô hình mạng nơ ron 10](#_Toc198133410)

[Hình 1.5: Minh họa mạng nơ ron kết nối đầy đủ 12](#_Toc198133411)

[Hình 1.6: Minh họa kiến trúc transformer 17](#_Toc198133412)

[Hình 1.7: : Mạng transformer gồm 6 lớp xếp tầng 18](#_Toc198133413)

[Hình 1.8: Minh họa tokenization 20](#_Toc198133414)

[Hình 1.9: Embedding token 21](#_Toc198133415)

[Hình 1.10: Công thức tính vị trí 22](#_Toc198133416)

[Hình 1.11: Đầu vào mô hình Transformer 22](#_Toc198133417)

[Hình 1.12: Khởi tạo và tính 3 vectơ Q, K, V 23](#_Toc198133418)

[Hình 1.13: Tính mức độ liên quan 24](#_Toc198133419)

[Hình 1.14: Tính trọng số Attention 24](#_Toc198133420)

[Hình 1.15: Kết quả đầu ra Self-Attention 25](#_Toc198133421)

[Hình 1.16: Minh họa multi-head attention 26](#_Toc198133422)

[Hình 1.17: Tổng hợp thông tin từ các heads 26](#_Toc198133423)

[Hình 1.18: Residual và Normalization 27](#_Toc198133424)

[Hình 1.19: Thành phần chính trong khối decoder 28](#_Toc198133425)

[Hình 1.20: Quá trình từ encoder sang decoder 29](#_Toc198133426)

[Hình 1.21: Minh họa kiến trúc mô hình BERT 30](#_Toc198133427)

[Hình 1.22: Minh họa kiến trúc mô hình GPT 32](#_Toc198133428)

[Hình 1.23: Minh họa kiến trúc mô hình LLAMA 33](#_Toc198133429)

[Hình 1.24: Pipeline RAG 37](#_Toc198133430)

[Hình 1.25: Chuẩn bị dữ liệu và đẩy lên database 38](#_Toc198133431)

[Hình 1.26: Truy vấn dữ liệu và tạo câu trả lời 40](#_Toc198133432)

[Hình 2.1: Giao diện của cơ sở dữ liệu milvus 44](#_Toc198133433)

[Hình 2.3: Sử dụng mô hình embedding để tạo vector 45](#_Toc198133434)

[Hình 2.4: Lưu dữ liệu ở cả dạng text và vector 45](#_Toc198133435)

[Hình 2.5: Tạo công cụ tìm kiếm tài liệu 47](#_Toc198133436)

[Hình 2.6: Trang đăng nhập 49](#_Toc198133437)

[Hình 2.7: Trang đăng ký tài khoản 49](#_Toc198133438)

[Hình 2.8: Trang tải tài liệu lên cơ sở dữ liệu vector 50](#_Toc198133439)

[Hình 2.9: Trang quản lý tài liệu đã tải lên 50](#_Toc198133440)

[Hình 2.10: Giao diện chat với trợ lý ảo HaOai 51](#_Toc198133441)

[Hình 2.11: Giao diện trang chủ 51](#_Toc198133442)

[Hình 3.1: Một số hình ảnh phản hồi của chatbot 53](#_Toc198133443)

# **LỜI NÓI ĐẦU**

## **LÝ DO CHỌN ĐỀ TÀI**

Với thời đại phát triển như hiện nay AI hay trí tuệ nhân tạo đang ngày càng đóng vai trò quan trọng trong nhiều lĩnh vực của con người từ cả đời sống, học tập lẫn công việc.

Trong đó các chatbot có thể giao tiếp với con người đang nổi bật hơn. Thể hiện ở việc các công ty công nghệ lớn trên toàn cầu đang đua nhau để phát triển các chatbot hiện đại như: GPT - OpenAI, Gemini - Google, Bing Chat,…

Việc phát triển và ứng dụng chatbot không chỉ mang lại lợi ích kinh tế mà còn góp phần nâng cao chất lượng cuộc sống. Không như con người cần nghỉ ngơi chatbot có thể hoạt động 24/7, hỗ trợ phản hồi các câu hỏi liên tục không phụ thuộc vào giờ làm việc. Bên cạnh đó nó có thể xử lý hàng nghìn yêu cầu cùng một lúc ở nhiều lĩnh vực như thương mại điện tử, giáo dục, học tập, y tế,…

Chính vì vậy em một sinh viên chuyên ngành Khoa học máy tính đã quyết định chọn đề tài: “Nghiên cứu mạng Transformer và phát triển chatbot tích hợp RAG” để có thể nắm bắt được công nghệ mới và phát triển kỹ năng bản thân.

## **MỤC TIÊU NGHIÊN CỨU**

Mục tiêu của đề tài là tập trung phát triển chatbot tích hợp công nghệ RAG giúp tra cứu, hỏi đáp thông tin nhanh hơn, dễ dàng hơn, liên tục mà không cần tới sự tham gia của con người.

## **ĐỐI TƯỢNG NGHIÊN CỨU**

Mô hình: Transformer

Thuật toán**:** Self-Attention, Multi-Head Attention, Positional Encoding.

Ngôn ngữ lập trình: Python.

Lý thuyết: Xử lý ngôn ngữ tự nhiên (NLP), học sâu (Deep Learning), tinh chỉnh mô hình (Fine-Tuning), truy xuất và tổng hợp thông tin (Retrieval-Augmented Generation - RAG).

Ứng dụng: Phát triển chatbot tích hợp RAG nhằm nâng cao khả năng trả lời câu hỏi dựa trên tài liệu.

## **PHẠM VI NGHIÊN CỨU**

Thời gian: Thực hiện trong khoảng thời gian từ 17/03/2025 đến 17/05/2025.

Lĩnh vực: Trí tuệ nhân tạo, Xử lý ngôn ngữ tự nhiên.

## **KẾT QUẢ MONG MUỐN**

Hiểu rõ cấu trúc, nguyên lý và cách thức hoạt động của mạng Transformer.

Xây dựng được chatbot cơ bản có thể thực hiện trả lời các câu hỏi.

Tích hợp được công nghệ RAG vào chatbot.

Đánh giá được hiệu quả của chatbot qua các tiêu chí, chỉ số.

## **CẤU TRÚC BÁO CÁO**

Chương 1: Cơ sở lý thuyết.

Chương 2: Phát triển chatbot.

Chương 3: Kết quả và hướng phát triển.

# **CHƯƠNG I: CƠ SỞ LÝ THUYẾT**

Chương này sẽ đi vào khám phá về các khái niệm như chatbot và Transformers, trình bày về quá trình huấn luyện và đặc trưng của các mô hình ngôn ngữ lớn (Large Language Model - LLMs).

## **1.1. Tổng quan về chatbot**

### **1.1.1. Khái niệm**

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.

### **1.1.2. Lịch sử phát triển**

Chatbot xuất hiện từ thập niên 1960. Dưới sự phát triển của Joseph Weizenbaum - nhà khoa học máy tính tại MIT, tiền thân của Chatbot được ra đời với cái tên ELIZA và đây là chương trình đầu tiên trên thế giới cho phép con người và máy tính có thể nói chuyện với nhau một cách tự nhiên nhất.

Các công ty lớn như Facebook, Google, Amazon, Microsoft suốt nhiều năm qua đã rót rất nhiều tiền cho việc phát triển và nâng cấp hệ thống Chatbot tự động của mình. Khi người dùng truy cập vào website của các doanh nghiệp này, họ sẽ nói chuyện trực tiếp với hệ thống Chatbot thông minh và có cảm giác như đó là người thật việc thật.

Vào năm 2022 với sự ra đời của ChatGPT của OpenAI được xem là bước ngoặt, làm thay đổi toàn diện lĩnh vực này. Nó đã đưa chatbot từ công cụ hỗ trợ đơn thuần thành trợ lý thông minh, gây ra sự thay đổi trong cách con người học tập – làm việc. Thúc đẩy việc nghiên cứu và áp dụng chatbot AI vào mọi lĩnh vực.

### **1.1.3. Phân loại**

#### **1.1.3.1. Chatbot dựa trên quy tắc, kịch bản (Rule-based chatbots)**

Đây là loại sẽ sử dụng các quy tắc được lập trình sẵn để phản hồi cho người dùng. Đồng thời, nó sẽ không thể tự động học và phát triển theo thời gian cũng như chỉ đáp ứng được các câu hỏi đơn giản và có giới hạn. Độ chi tiết, thông minh của Chatbot dựa trên các điều kiện thiết lập sẵn sẽ phụ thuộc rất lớn vào cơ sở dữ liệu mà đội ngũ phát triển nạp cho nó.

Ưu điểm:

* Dễ xây dựng.
* Phù hợp với các câu hỏi đơn giản, lặp lại.

Nhược điểm:

* Quá đơn giản không hiểu được nội dung câu hỏi của người dùng.
* Không có sự linh hoạt, khó trả lời các câu hỏi nằm ngoài kịch bản.

Ví dụ: chatbot đặt hàng, tra cứu thông tin sản phẩm.

#### **1.1.3.2. Chatbot dựa trên trí tuệ nhân tạo (Chatbot AI)**

Trái ngược hoàn toàn với loại Chatbot trên. Nó sử dụng các thuật toán học máy để tự động học và phát triển theo thời gian, đồng thời nó có khả năng xử lý thông tin phức tạp hơn và có thể đáp ứng nhiều câu hỏi khác nhau của người dùng. Chính vì được tích hợp trí tuệ nhân tạo nên loại Chatbot này gần như suy nghĩ và phản hồi yêu cầu của khách hàng giống như một người thực thụ.

Ưu điểm:

* Linh hoạt, có thể xử lý tốt các câu hỏi phức tạp.
* Câu trả lời tự nhiên không máy móc.

Nhược điểm:

* Chi phí cho việc xây dựng, phát triển và vận hành khá cao.
* Đòi hỏi dữ liệu để huấn luyện lớn và phải được đảm bảo chất lượng.
* Ví dụ: chatGPT, Gemini, …

### **1.1.4. Ứng dụng**

Một số ứng dụng của chatbot trong thực tiễn đời sống là:

* Dịch vụ khách hàng: hỗ trợ tư vấn, giải đáp các câu hỏi và các sự cố cơ bản,…
* Thương mại điện tử: hỗ trợ tư vấn mua sắm, tra cứu giá sản phẩm,…
* Y tế: giải đáp một số triệu chứng, chuẩn đoán bệnh.
* Giáo dục: giải thích, ôn luyện kiến thức bài tập mới.

Nói chung hiện nay chatbot đã mang lại rất nhiều lợi ích cho con người ở hầu hết mọi lĩnh vực.

### **1.1.5. Thách thức trong việc triển khai chatbot**

Bên cạnh một số lợi ích thì cũng tồn tại một số khó khăn trong việc triển khai thực tế:

* + **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.

## **1.2. Mô hình ngôn ngữ lớn**

### **1.2.1. 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.

### **1.2.2. Lịch sử phát triển**

Việc tạo ra các Mô hình ngôn ngữ lớn không diễn ra trong một sớm một chiều. Đáng chú ý là khái niệm đầu tiên về các mô hình ngôn ngữ bắt đầu với các hệ thống dựa trên quy tắc được gọi là Xử lý ngôn ngữ tự nhiên. Các hệ thống này tuân theo các quy tắc được xác định trước để đưa ra quyết định và suy ra kết luận dựa trên đầu vào văn bản.

Tiến bộ ban đầu lớn nhất là mạng nơ-ron (Neural Networks), được biết đến khi lần đầu tiên được giới thiệu vào năm 1943 lấy cảm hứng từ các nơ-ron trong chức năng não người, bởi nhà toán học Warren McCulloch. Mạng nơ-ron thậm chí còn có trước thuật ngữ “trí tuệ nhân tạo” khoảng 12 năm. Mạng nơ-ron trong mỗi lớp được tổ chức theo một cách cụ thể, trong đó mỗi nút giữ một trọng số xác định tầm quan trọng của nó trong mạng. Cuối cùng, mạng nơ-ron đã mở ra những cánh cửa bị đóng kín, tạo nên nền tảng mà AI từ đó được xây dựng lên cho đến ngày nay.

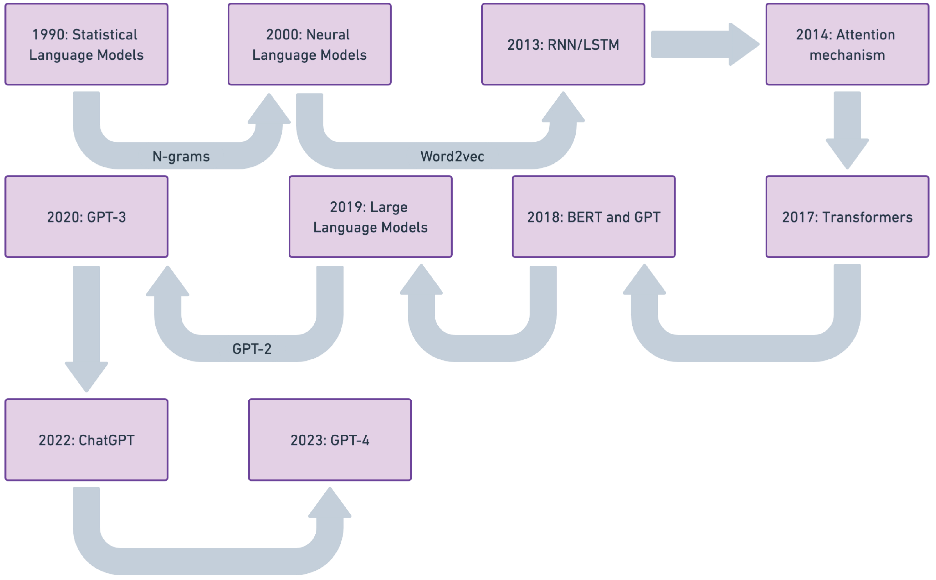
A diagram of a network

AI-generated content may be incorrect.

Hình 1.1: Minh họa mạng nơ-ron

Tuy vậy các hệ thống NLP chủ yếu dựa vào các quy tắc thủ công, sau đó chuyển sang các phương pháp thống kê như mô hình n-gram, HMM, và học máy truyền thống. Bước ngoặt lớn xảy ra khi mạng nơ-ron sâu (deep learning) được tái khám phá và phát triển mạnh mẽ từ năm 2012, cùng với sự ra đời của kỹ thuật từ nhúng (word embeddings) như Word2Vec và GloVe. Những nền tảng này giúp các mô hình hiểu rõ hơn về ngữ nghĩa và ngữ cảnh trong văn bản.

Cột mốc quan trọng nhất là sự ra đời của kiến trúc Transformer năm 2017, mở đường cho các LLM hiện đại như BERT, GPT, T5, và các mô hình mã nguồn mở như LLaMA hay Mistral. Những mô hình này có khả năng học từ lượng dữ liệu khổng lồ và thực hiện nhiều nhiệm vụ ngôn ngữ một cách linh hoạt mà không cần huấn luyện lại từ đầu. Từ 2020 trở đi, LLM trở thành công nghệ trung tâm trong nhiều ứng dụng AI thực tế như chatbot, tìm kiếm thông minh, hỗ trợ lập trình, và trợ lý ảo, đánh dấu một bước tiến quan trọng trong quá trình đưa AI vào đời sống hàng ngày.



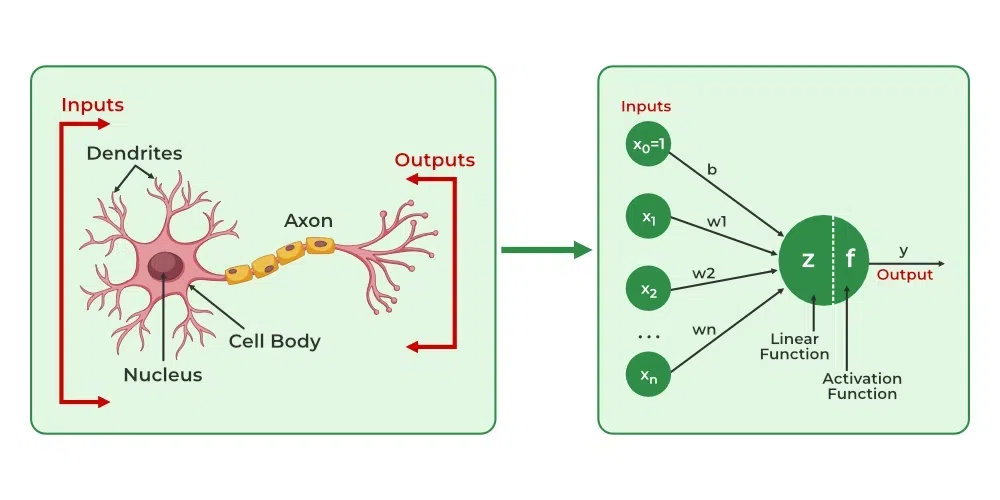
Hình 1.2: Lược sử các mô hình từ 1920 đến nay

## **1.3. Mạng nơ ron**

Mạng nơ-ron nhân tạo là một nhánh của học máy (Machine learning) dựa trên sự hiểu biết về hoạt động của não người. Một ANN được tạo bởi nhiều nơron cùng với nhiều kết nối, được tổ chức thành các lớp. Các nơ-ron của một lớp sẽ kết nối với các nơ-ron của một hoặc một số lớp tiếp theo [1].

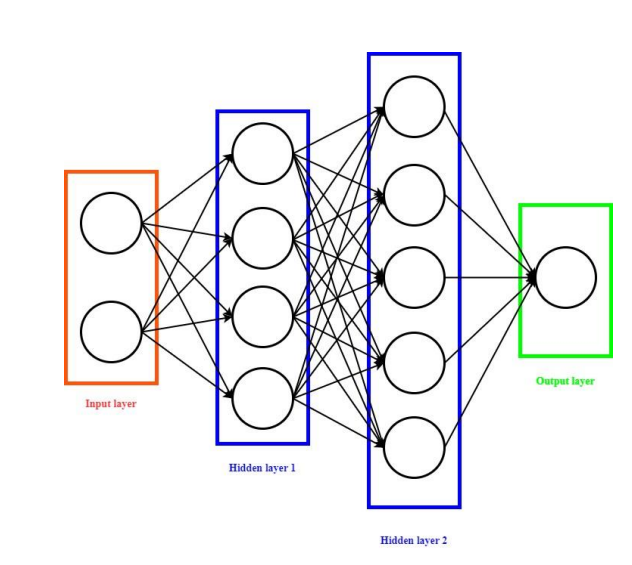
Một số thành phần chính:

* Nơ ron: Mỗi nơ-ron trong mạng tương tự như một tế bào thần kinh trong não người. Nó nhận dữ liệu đầu vào (input), xử lý chúng bằng một ngưỡng và hàm kích hoạt, rồi truyền tín hiệu đến các nơ-ron khác.
* Kết nối: Các kết nối giữa các nơ-ron giúp dữ liệu truyền từ lớp này sang lớp khác. Mỗi kết nối có một trọng số (weight) để điều chỉnh mức độ ảnh hưởng của kết nối đó.
* Trọng số (Weight) và độ lệch (Bias): Là các tham số điều chỉnh mức độ ảnh hưởng của đầu vào đối với đầu ra của nơ-ron. Trọng số cao có nghĩa là đầu vào sẽ có ảnh hưởng lớn hơn đối với kết quả của nơ-ron.
* Hàm lan truyền: Các hàm lan truyền giúp dữ liệu di chuyển qua các lớp của mạng nơ-ron, từ lớp đầu vào đến lớp đầu ra. Quá trình này có thể chia thành lan truyền xuôi(Forward Propagation) và lan truyền ngược (Backward Propagation).
* Quy tắc học: Đây là quá trình điều chỉnh trọng số và độ chệch để mạng học từ dữ liệu. Gradient Descent là thuật toán học phổ biến nhất, giúp tìm ra trọng số và độ chệch tối ưu bằng cách giảm thiểu sai số (loss) qua các bước học.



Hình 1.3: Minh họa nơ ron

Với mỗi một mạng nơ-ron thường bao gồm 3 kiểu lớp: Lớp đầu vào (Input Layer), các lớp ẩn (Hidden Layer) và lớp đầu ra (Output Layer).



Hình 1.4: Mô hình mạng nơ ron

Lớp đầu vào (Input Layer): đây là lớp đầu tiên trong mạng nơ-ron, có nhiệm vụ tiếp nhận dữ liệu đầu vào và truyền thông tin đến các lớp kế tiếp. Mỗi nơ-ron trong lớp này tương ứng với một đặc trưng (feature) của dữ liệu, do đó số lượng nơ-ron bằng với số chiều của dữ liệu đầu vào..

Các lớp ẩn (Hidden Layers): Là các lớp trung gian giữa lớp đầu vào và lớp đầu ra. Mỗi lớp ẩn bao gồm nhiều nơ-ron nhân tạo, được kết nối với các lớp liền trước và sau. Số lượng lớp ẩn và số lượng nơ-ron trong mỗi lớp phụ thuộc vào độ phức tạp của bài toán. Bài toán càng phức tạp thì càng cần nhiều lớp ẩn để mô hình có thể học được các đặc trưng sâu và trừu tượng từ dữ liệu. Với sự hỗ trợ của phần cứng hiện đại, ngày nay ta có thể huấn luyện các mạng sâu với hàng trăm lớp ẩn.

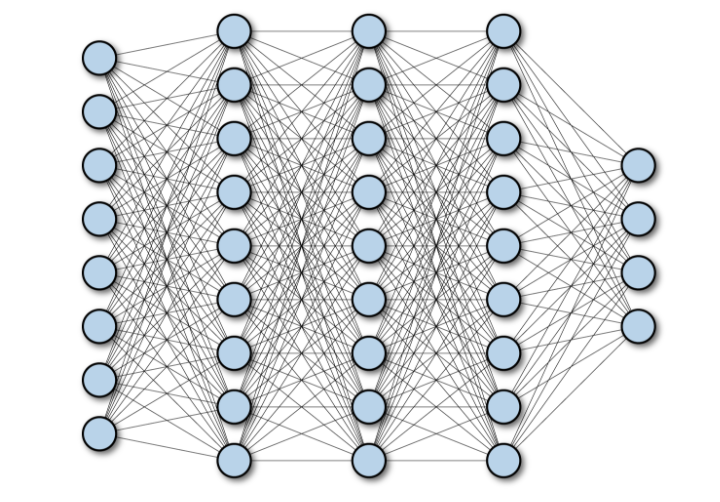
Lớp đầu ra (Output Layer): Là lớp cuối cùng trong mạng, chịu trách nhiệm tổng hợp thông tin từ lớp ẩn cuối và tạo ra kết quả đầu ra. Số lượng nơ-ron trong lớp đầu ra phụ thuộc vào loại bài toán:

* + Phân loại nhị phân: 1 nơ-ron với hàm kích hoạt sigmoid.
  + Phân loại đa lớp: Số nơ-ron bằng số lớp, thường dùng hàm softmax.
  + Hồi quy: 1 nơ-ron với hàm kích hoạt tuyến tính (hoặc không có).

Hàm kích hoạt tại lớp đầu ra được lựa chọn sao cho phù hợp với mục tiêu của bài toán.

Mạng nơ-ron kết nối đầy đủ (fully connected neural network), còn được gọi là mạng nơ-ron truyền thẳng (feedforward neural network), là một loại kiến trúc mạng nơ-ron cơ bản trong học sâu. Trong mạng nơ-ron này, mỗi nơ-ron trong một lớp được kết nối với tất cả các nơ-ron trong lớp liền trước và lớp liền sau nó [2].

Mặc dù đơn giản về cấu trúc, mạng nơ-ron kết nối đầy đủ vẫn cho thấy hiệu quả cao trong nhiều bài toán, đặc biệt là các bài toán phân loại hoặc hồi quy với dữ liệu có cấu trúc dạng bảng (structured data). Đây cũng là nền tảng của các mô hình học sâu phức tạp hơn.



Hình 1.5: Minh họa mạng nơ ron kết nối đầy đủ

Mạng nơ-ron kết nối đầy đủ bao gồm ít nhất một lớp đầu vào, một hoặc nhiều lớp ẩn, và một lớp đầu ra. Trong mỗi lớp, mỗi nơ-ron được kết nối với tất cả các nơ-ron trong lớp liền trước và lớp liền sau nó bằng các trọng số và độ lệch.

## **1.4. Word Embedding**

Như chúng ta đã biết máy tính không có khả năng hiểu được dữ liệu ở dạng text mà trong các mô hình ngôn ngữ lớn thì dữ liệu hầu như là dạng tài liệu văn bản rồi nên phải chuyển về dạng số để có thể xử lý và tính toán.

Khái niệm: là một không gian vector dùng để biểu diễn dữ liệu có khả năng miêu tả được mối liên hệ, sự tương đồng về mặt ngữ nghĩa, nội dung của dữ liệu. *Không gian này bao gồm nhiều chiều và các từ trong không gian đó mà có cùng văn cảnh hoặc ngữ nghĩa sẽ có vị trí gần nhau****.***

Ví dụ ở bảng sau ta sẽ thấy “vua” và “nữ hoàng” ở gần nhau vì có ngữ nghĩa gần tương tự:

|  |  |
| --- | --- |
| Từ | Vector embedding |
| vua | [0.5, 0.2, 0.1] |
| nữ hoàng | [0.52, 0.18, 0.12] |
| ghế | [-0.4, 0.9, 0.7] |

Bảng 1.1: Ví dụ về word embedding

Để hiểu rõ lợi ích của word embedding, hãy so sánh với cách biểu diễn truyền thống là one-hot encoding - vốn thường được dùng trong các bài toán phân loại hoặc multi-label.

Với one-hot encoding dữ liệu sẽ được biểu diễn dưới dạng sau:

|  |  |  |
| --- | --- | --- |
| Document | Index | One-hot encoding |
| a | 1 | [1, 0, 0, ...., 0](9999 số 0) |
| b | 2 | [0, 1, 0, ...., 0] |
| c | 3 | [0, 0, 1, ...., 0] |
| … | … | … |
| Cha | 999 | [0, 0, 0, ..., 1, 0] |
| Ba | 1000 | [0, 0, 0, ...., 0, 1] |

Bảng 1.2: Minh họa dữ liệu với one-hot encoding

Qua đấy ta có thể thấy mặc dù dùng one-hot encoding khá đơn giản và dễ hiểu nhưng tồn tại nhiều nhược điểm khá nghiêm trọng trong bài toán NLP, dưới đây là 3 vấn đề nổi bật:

* **Chi phí tính toán lớn:** nếu tập dữ liệu có 10.000 từ, mỗi từ sẽ được biểu diễn bằng vector có 10.000 chiều (gồm chủ yếu là số 0). Như vậy với tập dữ liệu hàng triệu từ, độ dài vector quá lớn gây tốn bộ nhớ và tính toán chậm.
* **Mang ít giá trị thông tin:** Vector chỉ có đúng 1 phần tử là 1, còn lại là 0 → không mang ngữ nghĩa. Không phản ánh được mối liên hệ giữa các từ (ví dụ: "cha" và "ba" hoàn toàn khác nhau trong one-hot, dù cùng ngữ nghĩa).
* **Độ khái quát yếu**: Các từ đồng nghĩa như “cha”, “ba”, “dượng” sẽ được biểu diễn hoàn toàn khác nhau, không thể hiện được rằng chúng có ý nghĩa giống nhau. Điều này gây khó khăn cho mô hình trong việc học mối quan hệ giữa các từ hiếm và từ phổ biến.

Trong khi đó với word embedding độ dài vector chỉ từ 100 – 300 chiều và biểu diễn được nghữ nghĩa khi các từ gần nghĩa sẽ có vector gần nhau và tăng khả năng tổng quát hóa. Do đó trong các mô hình ngôn ngữ lớn hiện nay embedding hay word embedding là bước cốt lõi giúp máy tính có thể hiểu được văn bản ở mức độ sâu hơn.

Một số phương pháp phổ biến để tạo word embedding hiện nay:

* Word2Vec: Sử dụng mạng nơ-ron để học vector biểu diễn từ dựa trên ngữ cảnh (CBOW hoặc Skip-gram).
* GloVe: Dựa trên ma trận tần suất từ và thống kê toàn cục để xây dựng vector.
* FastText: Mở rộng Word2Vec bằng cách phân tích cả các ký tự con (subword), giúp hiểu từ lạ, từ hiếm.
* BERT/ELMo: Tạo embedding theo ngữ cảnh, nghĩa là cùng một từ trong các câu khác nhau có thể có embedding khác nhau.

Tóm lại embedding đóng vai trò là cầu nối giữa dữ liệu văn bản và mô hình học máy. Nhờ vào đó, các mô hình có thể hiểu được ngữ nghĩa thay vì chỉ xử lý các ký tự một cách máy móc. Đây là một thành phần không thể thiếu trong bất kỳ mô hình ngôn ngữ hiện đại nào, đặc biệt là trong thời đại của các mô hình ngôn ngữ lớn.

## **1.5. Transformers**

### **1.5.1. Tổng quan**

**Transformer** là một kiến trúc mô hình học sâu dựa trên cơ chế self-attention, cho phép mô hình này hiểu được mối quan hệ giữa các từ trong một câu mà không cần đến kiến trúc tuần tự truyền thống như RNN (Recurrent Neural Networks) hay LSTM (Long Short-Term Memory). Transformer có khả năng xử lý toàn bộ câu cùng một lúc, điều này giúp tăng tốc độ huấn luyện và cải thiện hiệu quả xử lý.

### **1.5.2. Lịch sử hình thành**

Năm 2017, Google công bố bài báo “Attention Is All You Need” thông tin về Transformer như tạo ra bước ngoặt mới trong lĩnh vực NLP – Natural language processing. Có thể coi đây là bước đột phá đóng góp lớn cho sự ra đời của các chatbot tiên tiến như BERT(2018), GPT(2018), CHATGPT(2022).

Trước khi Transformer ra đời, hầu như các tác vụ xử lý ngôn ngữ tự nhiên, đặc biệt trong mảng Machine Translation (dịch máy) đều sử dụng kiến trúc Recurrent Neural Networks (RNNs) bao gồm Long short-term memory (LSTM) và Gated Recurrent Units (GRUs). Tuy đạt được 1 số thành công nhất định nhưng chúng vẫn còn tồn tại một số nhược điểm như:

* Xử lý các chuỗi dài: Do hiện tượng Gradient Vanishing – khi số lượng các từ ngày càng tăng thì số lượng units cũng ngày càng lớn, gradient giảm dần ở các units cuối dẫn đến mất thông tin về sự phụ thuộc xa giữa các units. Mặc dù LSTM và GRU đã được thiết kế để giải quyết vấn đề trên nhưng chúng vẫn gặp khó khăn khi học các phụ thuộc dài.
* Tốc độ huấn luyện chậm: do bản chất xử lý tuần tự - mỗi bước thời gian đều phụ thuộc vào bước trước đó nên không thể tận dụng được hiệu quả tuyệt vời của các GPU hay TPU hiện đại dẫn tới tốc độ và hiệu suất huấn luyện không được tối ưu.

Chính vì các bất cập trên nên transformer đã được ra đời nhằm giải quyết, khắc phục chúng:

* Xử lý song song: Transformer không xử lý đầu vào một cách tuần tự nhưng RNNs mà thay vào đó là xử lý theo từng khối. Điều này giúp việc huấn luyện mô hình được thực hiện song song giúp giảm đáng kể thời gian training.
* Self-Attention: đây là cơ chế chính giúp Transformer xác định được mối quan hệ của các từ trong một câu thông qua ma trận Attention bất kể khoảng cách của chúng trong câu từ đó giải quyết được vấn đề phụ thuộc dài hạn trong RNNs.

Do các ưu điểm nổi bật ở trên Transformer đã được coi là tiêu chuẩn mới trong lĩnh vực NLP, góp phần phát triển cho những mô hình chatbot hiện đại ngày nay BERT, GPT-3. Mở ra con đường mới cho các mô hình AI mạnh mẽ, hiệu quả.

### **1.5.3. Kiến trúc**

Transformers là mô hình bao gồm 2 thành phần chính: mã hóa (Encoder)

và giải mã (Decoder).

A diagram of a process

AI-generated content may be incorrect.

Hình 1.6: Minh họa kiến trúc transformer

Encoder là một ngăn xếp gồm 6 lớp riêng biệt xếp chồng lên nhau, số lớp ở đây có thể tùy chỉnh phụ thuộc vào mục đích từng bài toán. Mỗi lớp bao gồm 2 lớp con, đầu tiên là lớp xử lý chú ý đa đầu (Multi-head Attention) và lớp con thứ hai đơn giản là một mạng nơ-ron truyền thẳng (Feed-forward). Kết nối phần dư (Residuals Connection) bao quanh hai lớp con này tiếp theo đó là bước chuẩn hóa lớp (Layer Normalization) giúp mô hình huấn luyện hiệu quả hơn và tránh mất mát thông tin [3].

Decoder (bộ giải mã) trong kiến trúc Transformer là một ngăn xếp gồm 6 lớp giống với khối encoder, với mỗi lớp được thiết kế để sinh ra đầu ra tuần tự theo ngữ cảnh đã học từ đầu vào. Mỗi lớp trong Decoder bao gồm ba thành phần chính: đầu tiên là Masked Multi-Head Attention, cho phép mô hình tập trung vào các từ trước đó trong chuỗi đầu ra và ngăn không cho nhìn thấy các từ tương lai trong quá trình huấn luyện. Tiếp theo là Multi-Head Attention với giúp khai thác thông tin từ chuỗi đầu vào đã được mã hóa để tạo ra ngữ cảnh phù hợp; cuối cùng là một mạng nơ-ron truyền thẳng (Feed-Forward Neural Network), được áp dụng độc lập tại từng vị trí trong chuỗi. Tương tự như Encoder, mỗi lớp trong Decoder đều được tích hợp kết nối tắt (Residual Connection) và chuẩn hóa lớp (Layer Normalization) để cải thiện độ ổn định và hiệu quả huấn luyện của mô hình.

A diagram of a computer

AI-generated content may be incorrect.

Hình 1.7: : Mạng transformer gồm 6 lớp xếp tầng

### **1.5.4. Quy trình hoạt động**

#### **1.5.4.1. Input Embedding**

Do máy tính không thể xử lý dữ liệu ở dạng chữ được nên ta phải chuyển đổi input đầu vào sang dạng số. Gồm 2 bước:

**a. Tokenization:**

Đây là một trong những bước quan trọng nhất trong quá trình tiền xử lý văn bản của ở cả các kỹ thuật NLP truyền thống hay học sâu nâng cao.

Quá trình Tokenization sẽ tách một cụm từ, câu, đoạn văn hoặc nhiều tài liệu văn bản thành các đơn vị nhỏ hơn. Mỗi đơn vị được gọi là Tokens. Và cách chia sẽ tùy thuộc vào thuật toán áp dụng.  Các tokens được sử dụng để tạo ra từ vựng trong kho từ điển, tức là một tập hợp các từ và ký tự mà mô hình sẽ làm việc với.

Hiện nay có 3 kỹ thuật, thuật toán được áp dụng:

* Word-based tokenization: đây là phương pháp đơn giản nhất đó chính là chia theo khoảng trắng hoặc dấu câu giữa các từ. Có thể được thực hiện dễ dàng bằng cách sử dụng phương thức split () của RegEx hoặc Python,... Hạn chế của kỹ thuật này là nó dẫn đến một kho từ điển khổng lồ và một lượng từ vựng lớn, khiến mô hình cồng kềnh hơn và đòi hỏi nhiều tài nguyên tính toán hơn.
* Character-based tokenization: Mã hóa dựa trên ký tự chia văn bản thô thành các ký tự riêng lẻ. Logic đằng sau là một ngôn ngữ có nhiều từ khác nhau nhưng có một số ký tự cố định. Điều này dẫn đến một lượng từ vựng rất nhỏ. Do đó, mã hóa dựa trên ký tự sẽ sử dụng ít token hơn so với mã hóa dựa trên từ. Tuy kỹ thuật này giúp giảm kích thước từ vựng nhưng lại làm tăng độ dài chuỗi trong mã hóa dựa trên ký tự. Mỗi từ được chia thành từng ký tự và do đó, chuỗi mã hóa dài hơn nhiều so với văn bản thô ban đầu.
* Subword-based tokenization: tách theo tiền tố và hậu tố, là một phương pháp tối ưu hơn Word-based Tokenization khi xử lý các từ hiếm hoặc từ mới. Phương pháp này không chia từ thành từng đơn vị riêng lẻ mà chia thành các phần nhỏ hơn, giúp mô hình hiểu được cả từ phổ biến lẫn từ chưa từng gặp trước đó.

Sau khi đã chia input thành các token ta sẽ ánh xạ chúng sang các chỉ số ID từ từ điển đã tạo của mô hình.

A computer screen shot of a program

AI-generated content may be incorrect.

Hình 1.8: Minh họa tokenization

**b. Embedding Layer:**

Sau khi dữ liệu văn bản được chuyển đổi thành các chỉ số ID thông qua quá trình tokenization, bước tiếp theo là ánh xạ chúng sang một không gian vector để có thể sử dụng trong các mô hình học sâu.

Mỗi token được ánh xạ thành một vector có số chiều cố định **(embedding dimension)**. Các giá trị trong vector này ban đầu được khởi tạo ngẫu nhiên và sau đó được cập nhật thông qua quá trình huấn luyện.

Ta có thể huấn luyện lại từ đầu hoặc sử dụng các embedding có sẵn như Word2Vec, Glove – những vectơ đã được huấn luyện sẵn trên dữ liệu lớn và có thể dùng làm tham số ban đầu.

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 1.9: Embedding token

#### **1.5.4.2. Positional Encoding**

Do không có cơ chế xử lý tuần tự từng từ một như RNN nên để có thể phân biệt được vị trí của từng từ tránh trường hợp hiểu sai ngữ cảnh. Nên Transfomer đã sử dụng một hàm sin/cos để tạo ra một vectơ biểu diễn vị trí.

A close-up of a mathematical equation

AI-generated content may be incorrect.

Hình 1.10: Công thức tính vị trí

Sau đó ta sẽ cộng với vectơ embedding ở bước trên để thành đầu vào.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1.11: Đầu vào mô hình Transformer

#### **1.5.4.3. Encoder**

Encoder gồm nhiều layer giống nhau thường là 6 layer và mỗi layer gồm 2 thành phần chính: Multi-Head Attention và Feed Forward.

**a. Multi-Head Attention**

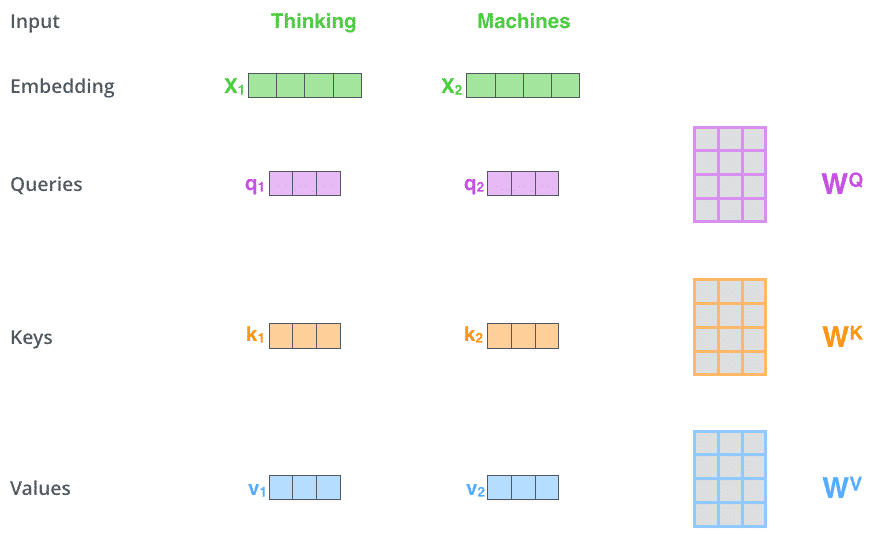
Trong một câu mỗi từ sẽ có thể liên quan đến các từ khác trong câu. Thành phần này sẽ giúp mô hình xác định mối quan hệ đó bằng cơ chế mới trên Transformer đó chính là: Self-Attention.

Đây là một cơ chế giúp mô hình Transformer tính toán mối quan hệ giữa một từ và các từ khác trong câu, giúp nó hiểu được từ nào quan trọng hơn trong một ngữ cảnh cụ thể.

Khác với mô hình RNN/LSTM xử lý tuần tự từng từ một, Self-Attention có thể xem xét toàn bộ câu cùng một lúc, giúp mô hình học được các mối quan hệ xa hơn giữa các từ.

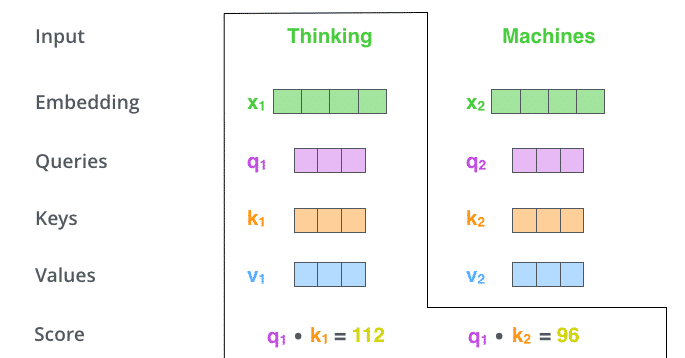
Self-Attention hoạt động qua 4 bước chính:

* Bước 1: Khởi tạo ngẫu nhiên 3 ma trận Wq, Wk, Wv và sẽ được thay đổi trong quá trình huấn luyện. Sau đó Ta nhân X -đầu vào với ba ma trận trọng số Wq, Wk, Wv **​** để thu được các vectơ Query (Q), Key (K), và Value (V).



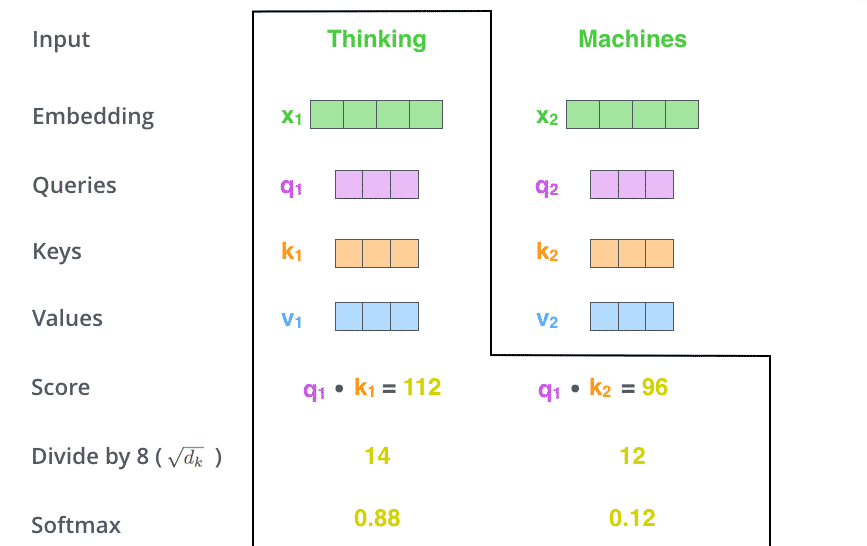
Hình 1.12: Khởi tạo và tính 3 vectơ Q, K, V

* Bước 2: Chúng ta tính điểm liên quan giữa các từ bằng cách nhân Q với . Cho biết mức độ liên quan giữa một từ với tất cả các từ khác trong câu.



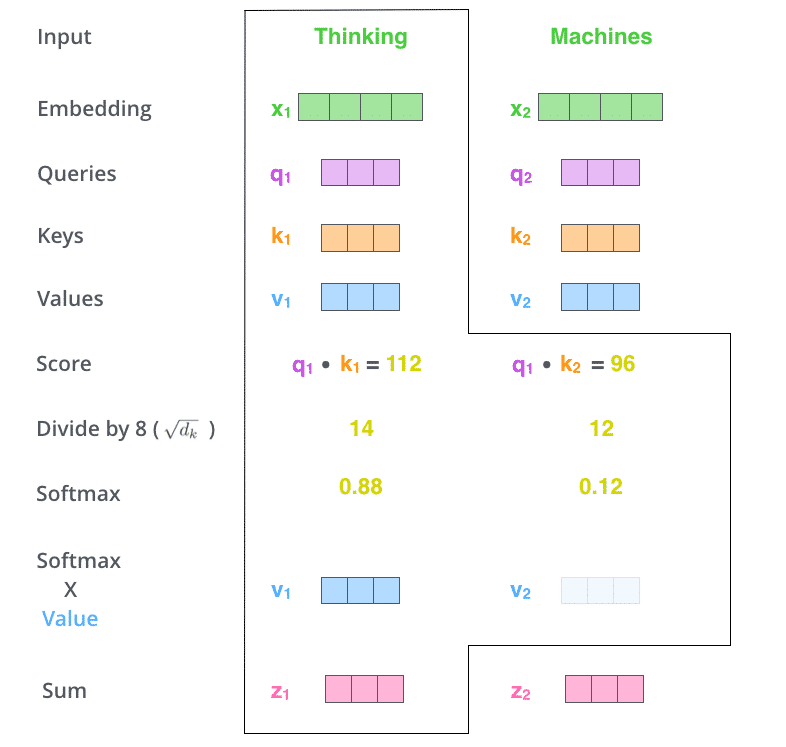
Hình 1.13: Tính mức độ liên quan

* Bước 3: Chia cho căn bậc 2 số chiều vectơ k để tránh các giá trị quá lớn gây mất ổn định rồi truyền vào hàm softmax để tạo trọng số Attention.



Hình 1.14: Tính trọng số Attention

* Bước 4: Nhân vectơ V với trọng số attention rồi cộng các vectơ thu lại được. Đây chính là đầu ra của lớp self-attention.



Hình 1.15: Kết quả đầu ra Self-Attention

Từ kết quả đầu ra trên giúp mô hình tập trung vào các từ quan trọng thay vì xử lý tất cả các từ như nhau. Tránh mất thông tin quan trọng, giúp mô hình nắm bắt được các mối quan hệ xa trong câu.

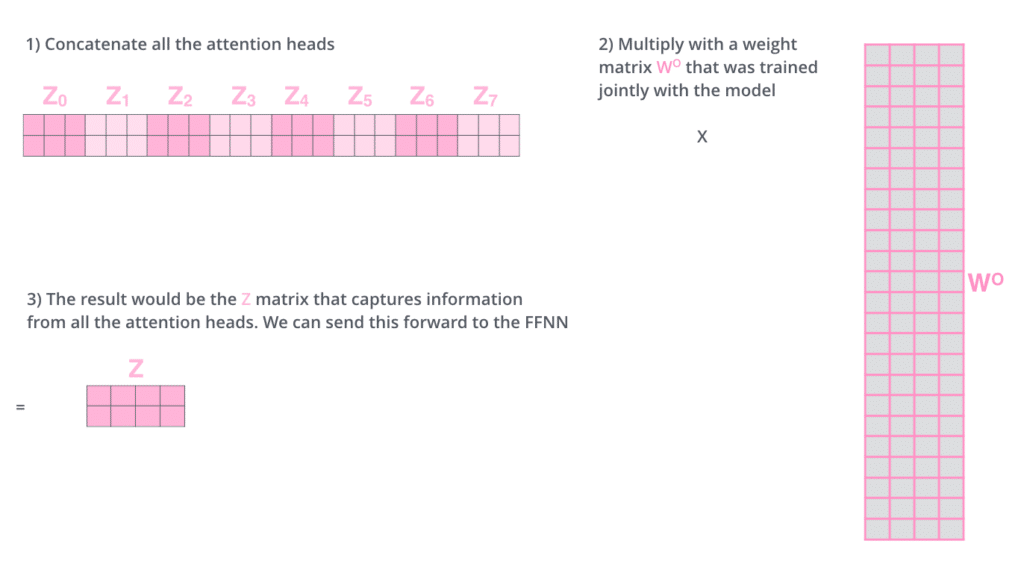
Vậy muilti-head attention là thay vì chỉ có bộ Wq, Wk, Wv ta sẽ khởi tạo n bộ và mỗi bộ sẽ cho ra một kết quả khác nhau. Từ đó học được mối quan hệ khác nhau giữa các từ.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1.16: Minh họa multi-head attention

Nếu tính attention như trên với n head ta sẽ được n ma trận Z khác nhau. Mạng truyền thẳng không phù hợp để nhận vào 8 ma trận, thay vào đó nó cần 1 ma trận duy nhất (mỗi từ một véc tơ). Do đó, ta cần biến đổi 8 ma trận về 1 ma trận duy nhất bằng cách nối tất cả các attention heads lại với nhau sau đó nhân với ma trận trọng số .

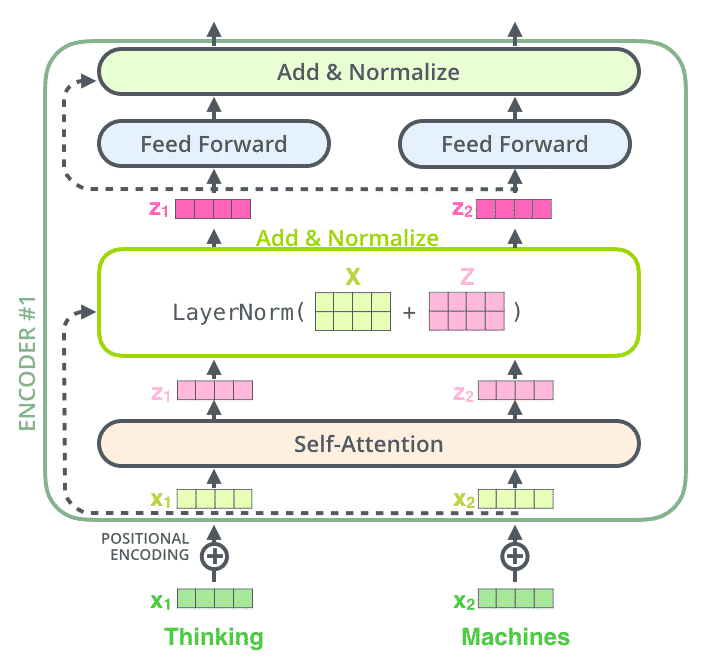


Hình 1.17: Tổng hợp thông tin từ các heads

**b. Feed Forward**

Trước khi nhận đầu ra của multihead-attention, Transformer sẽ thực hiện 2 bước:

* Residual Connection: cộng với input đầu vào ban đầu điều này giúp đảm bảo được thông tin ban đầu.
* Sau đó kết quả sẽ được đưa qua Layer Normalization để giúp mô hình ổn định hơn.



Hình 1.18: Residual và Normalization

Sau khi được Normalize, các vectors z được đưa qua mạng fully connected gồm hai lớp Dense với một hàm kích hoạt phi tuyến ở giữa. Giúp tăng khả năng biểu diễn của mô hình từ đó học được các mối quan hệ phức tạp giữa các từ trong câu.

#### **1.5.4.4. Decoder**

Decoder cũng được cấu trúc gồm nhiều lớp (layers) xếp chồng lên nhau. Mỗi lớp bao gồm ba thành phần chính, trong đó hai thành phần tương tự như ở encoder: Multi-Head Attention và Feed Forward. Thành phần duy nhất khác biệt trong decoder là Masked Multi-Head Attention.

*A diagram of a diagram

AI-generated content may be incorrect.*

Hình 1.19: Thành phần chính trong khối decoder

**a. Masked Multi-Head Attention**

Là một biến thể của Multi-Head Attention trong Decoder của Transformer. Nó được sử dụng để đảm bảo rằng tại mỗi thời điểm, mô hình chỉ được nhìn thấy các từ trước đó, chứ không thể thấy từ tương lai.

Cách hoạt động: hoạt động giống như Multi-Head Attention bình thường, nhưng sau khi tính attention thì sẽ cộng thêm ma trận mask chứa giá trị vô cực (-∞) cho các từ tương lai từ đó bắt mô hình phải dự đoán các từ tiếp theo dựa trên các từ đã biết.

Kết nối giữa Encoder và Decoder:

* Lấy đầu ra từ Encoder: Output của Encoder là một ma trận có dạng (seq\_len, d\_model), chứa thông tin đã được mã hóa từ câu nguồn.
* Sử dụng đầu ra của Encoder làm Key (K) và Value (V) cho Multi-Head Attention trong Decoder, ta nhân với hai ma trận trọng số Wk, Wv.
* Tính Query (Q) từ phần Decoder rồi kết hợp với Key, Value để tính attention. Giúp tính toán sự liên hệ của từng từ trong output với từng từ trong input.

Cuối cùng, các vector được đẩy vào lớp Linear (là 1 mạng Fully Connected) theo sau bới Softmax để cho ra xác suất của từ tiếp theo.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1.20: Quá trình từ encoder sang decoder

## **1.6. Một số mô hình nổi tiếng sử dụng kiến trúc transformer**

### **1.6.1. BERT - Bidirectional Encoder Representations from Transformers**

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình học sâu được phát triển bởi Google AI vào năm 2018. Dựa vào kiến trúc của khối Encoder và đặc biệt nhấn mạnh vào việc học ngữ cảnh hai chiều (bidirectional), giúp mô hình hiểu ngữ nghĩa từ một cách tốt hơn.

BERT là mô hình được xây dựng và huấn luyện bởi Google trên kho văn bản lớn gồm Toronto Book Corpus và Wikipedia. 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: trả lời câu hỏi và suy luận ngôn ngữ.

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.

A diagram of a block diagram

AI-generated content may be incorrect.

Hình 1.21: Minh họa kiến trúc mô hình BERT

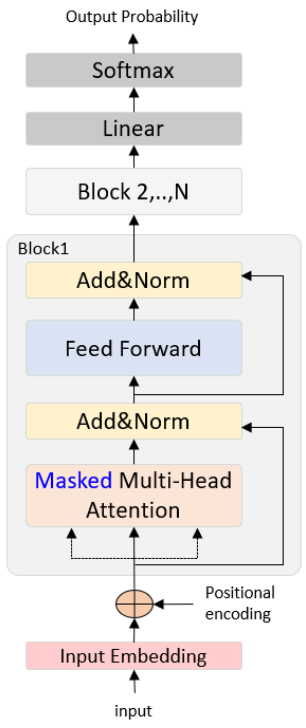
### **1.6.2. GPT - Generative Pre-training Transformers**

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.

Với nhiều ưu điểm vượt trội như: khả năng sinh văn bản tự nhiên, tạo ra nội dung mượt mà và có ý nghĩa nhờ cơ chế attention và GPT rất linh hoạt trong nhiều tác vụ NLP khác nhau và 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.

Hiện đang được ứng dụng trong nhiều lĩnh vực: trợ lý ảo, học tập, 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

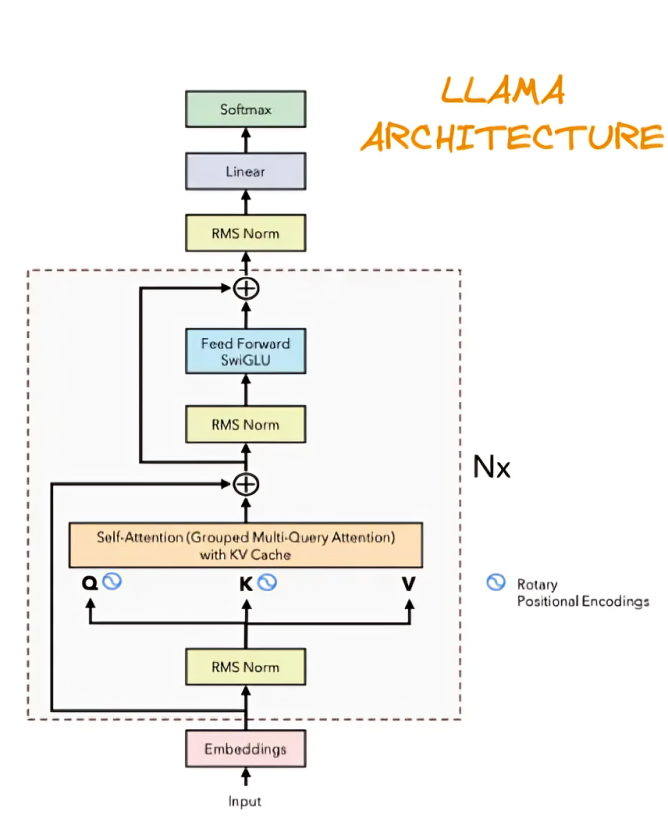


Hình 1.22: Minh họa kiến trúc mô hình GPT

### **1.6.3. LLAMA - Large Language Model Meta AI**

**LLaMA (Meta AI Large Language Model)** là mô hình ngôn ngữ lớn nền tảng hiện đại được Meta phát hành công khai, thiết kế để giúp các nhà nghiên cứu thúc đẩy công việc trong lĩnh vực AI. "Các mô hình nhỏ hơn, hiệu suất cao hơn như LLaMA cho phép những người khác trong cộng đồng nghiên cứu không có quyền truy cập vào lượng lớn cơ sở hạ tầng để nghiên cứu các mô hình này, tiếp tục dân chủ hóa quyền truy cập trong lĩnh vực quan trọng và thay đổi nhanh chóng này" [4]

Hiện phát hành ở nhiều kích cỡ tham số khác nhau, bao gồm 7B, 13B, 33B và 65B.



Hình 1.23: Minh họa kiến trúc mô hình LLAMA

## **1.7. Phương pháp đào tạo mô hình LLM**

Hiện nay có thể kể ra có 2 phương pháp chính để tiến hành đào tạo một mô hình cho một bài toán, lĩnh vực nào đó là:

* Huấn luyện từ đầu (Training from Scratch)
* Tinh chỉnh mô hình có sẵn (Fine-tuning)

### **1.7.1. Huấn luyện từ đầu (Training from Scratch)**

|  |  |
| --- | --- |
| Mô tả | - Dùng một tập dữ liệu lớn (hàng trăm GB đến vài TB) và huấn luyện mô hình hoàn toàn mới từ đầu.  - Không sử dụng trọng số (weights) từ các mô hình đã tồn tại. |
| Ưu điểm | - Toàn quyền kiểm soát cấu trúc mô hình, dữ liệu, và mục tiêu.  - Phù hợp cho ngôn ngữ đặc thù hoặc lĩnh vực chuyên biệt (nếu đủ dữ liệu). |
| Nhược điểm | - Cần hàng triệu USD tiền GPU, dữ liệu, thời gian  - Yêu cầu kỹ năng chuyên sâu về kiến trúc mạng, tokenizer, phân phối mô hình trên nhiều GPU.  - Khó khăn trong thu thập và xử lý dữ liệu chất lượng cao. |

Chỉ dùng khi là một tổ chức với quy mô lớn như: Google, OpenAI, VinAI và muốn tạo lại một mô hình từ đầu. Phương pháp này thường không phổ biến do chi phí khá cao, chưa nói đến việc có bộ dữ liệu tốt không thì riêng ví dụ với LlaMa-65B có 65 tỷ tham số, việc huấn luyện sẽ sử dụng card A100 GPU. Nếu sử dụng 2048 card A100 GPU để huấn luyện 1.4T token thì sẽ mất 21 ngày. Giá thành cho việc thuê A100 là 3.5$ (USD) cho mỗi giờ. Như vậy chi phí cho một lần huấn luyện sẽ là: $3.5/hr \*2048 \* 24 \* 21 = $3.6M.

### **1.7.2. Tinh chỉnh mô hình (Fine-tunning)**

|  |  |
| --- | --- |
| Mô tả | - Sử dụng một mô hình LLM đã được huấn luyện trước (pretrained) như LLaMA, Mistral, Falcon, GPT-J,...  - Tiếp tục huấn luyện mô hình này trên một tập dữ liệu cụ thể. |
| Ưu điểm | - Tốn ít tài nguyên hơn rất nhiều so với huấn luyện từ đầu.  - Có thể chuyên biệt hóa mô hình cho lĩnh vực như pháp luật, y tế, giáo dục...  - Hỗ trợ nhiều kỹ thuật như LoRA, QLoRA, PEFT giúp giảm tài nguyên cần thiết. |
| Nhược điểm | - Bị giới hạn bởi kiến trúc và năng lực của mô hình gốc.  - Khó khăn trong thu thập và xử lý dữ liệu chất lượng cao. |

Với các này thì chi phí sẽ đỡ tốn kém hơn so với huấn luyện lại từ đầu do sử dụng lại các mô hình đã được đào tạo sẵn. Nhưng khó khăn nằm ở tạo được tập dữ liệu có chất lượng cao cho bài toán, lĩnh vực đang cần giải quyết.

## **1.8.** **Retrieval Augmented Generation**

### **1.8.1. Tổng quan**

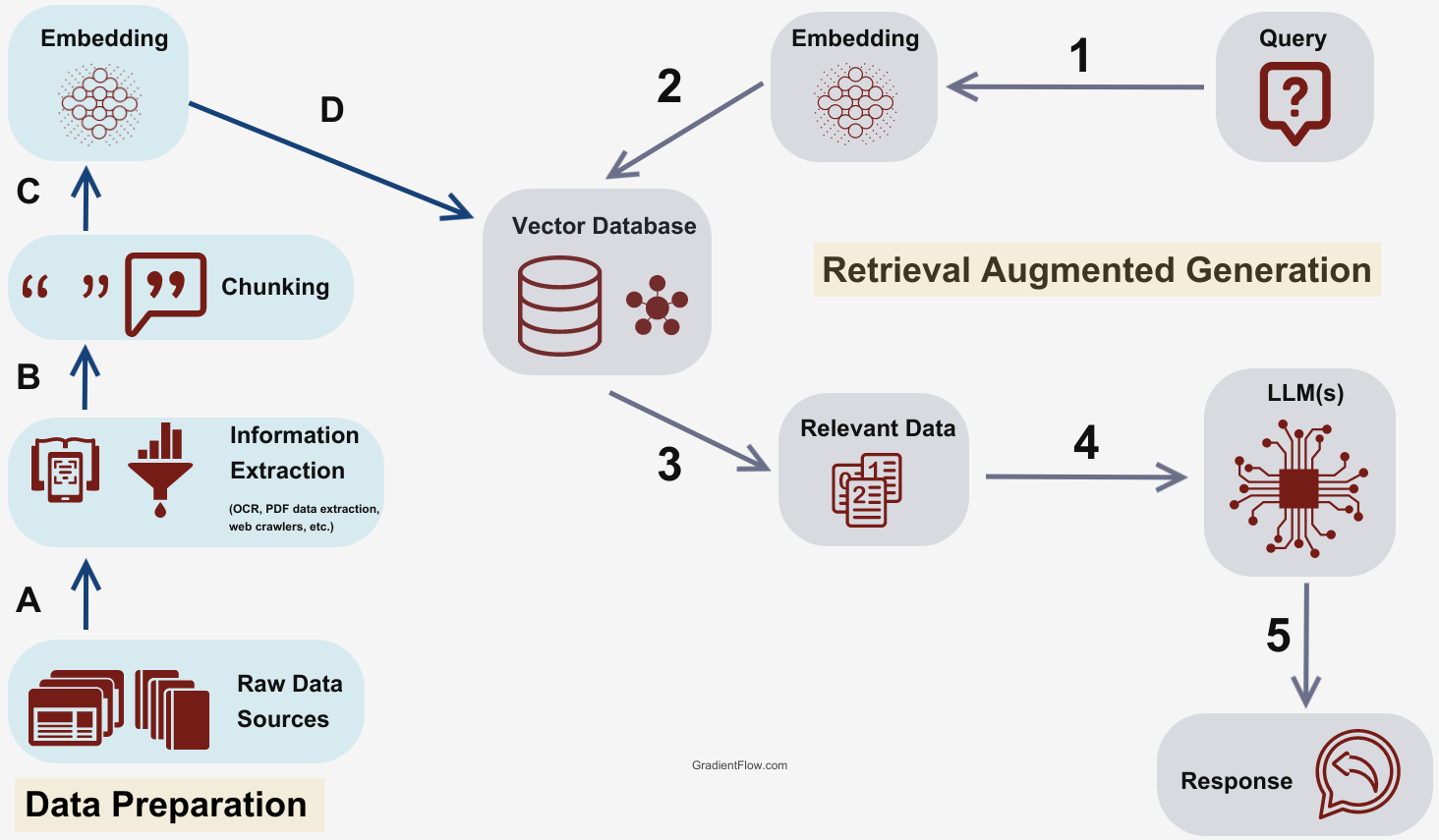
Hiện nay, các mô hình ngôn ngữ lớn (LLMs) sau khi được huấn luyện trên các tập dữ liệu quy mô lớn có khả năng sinh văn bản và trả lời câu hỏi một cách tự nhiên. Tuy nhiên, đối với các câu hỏi vượt ra ngoài phạm vi dữ liệu đã được huấn luyện, các mô hình này thường gặp phải hiện tượng **ảo giác thông tin** - đưa ra các câu trả lời có vẻ hợp lý nhưng thực chất không chính xác hoặc không tồn tại trong thực tế.

Khi các tổ chức, doanh nghiệp hoặc cá nhân có nhu cầu xây dựng một mô hình ngôn ngữ chuyên biệt để trả lời câu hỏi liên quan đến kiến thức nội bộ, một giải pháp phổ biến là tinh chỉnh (fine-tuning) mô hình dựa trên tập dữ liệu riêng, bao gồm các cặp câu hỏi - trả lời. Phương pháp này có thể mang lại kết quả chính xác và phù hợp hơn với ngữ cảnh cụ thể của doanh nghiệp. Tuy nhiên, fine-tuning đòi hỏi chi phí phần cứng lớn, thời gian huấn luyện dài, và đặc biệt là khó khăn trong việc cập nhật khi dữ liệu nội bộ thay đổi liên tục. Đây là rào cản lớn đối với các doanh nghiệp vừa và nhỏ.

Để khắc phục những hạn chế trên, **Retrieval-Augmented Generation (RAG)** ra đời như một giải pháp hiệu quả. RAG cho phép mô hình ngôn ngữ lớn kết hợp với kho dữ liệu bên ngoài, thường được tổ chức dưới dạng vector (vector store). Khi nhận được câu hỏi từ người dùng, hệ thống sẽ truy xuất thông tin liên quan từ kho dữ liệu này, sau đó tổng hợp cùng truy vấn ban đầu dưới dạng prompt để cung cấp ngữ cảnh cho mô hình sinh câu trả lời. Phương pháp này giúp giữ nguyên mô hình gốc, tránh việc tinh chỉnh lại toàn bộ mô hình mỗi khi dữ liệu thay đổi, đồng thời đảm bảo câu trả lời cập nhật và chính xác hơn theo thời gian.

Nguyên lý hoạt động của RAG: Khi nhận câu hỏi, hệ thống truy xuất thông tin liên quan từ kho dữ liệu bên ngoài, sau đó kết hợp thông tin này với câu hỏi thành một prompt ngữ cảnh. Mô hình ngôn ngữ lớn (LLM) sử dụng prompt để sinh câu trả lời chính xác hơn, giúp tận dụng dữ liệu bên ngoài mà không cần tinh chỉnh lại mô hình.

### **1.8.2. Quy trình hoạt động**



Hình 1.24: Pipeline RAG

Để hệ thống RAG hoạt động hiệu quả, toàn bộ quy trình được chia thành hai thành phần chính, tương ứng với hai giai đoạn chính trong pipeline. Mỗi thành phần đóng vai trò quan trọng trong việc đảm bảo mô hình có thể truy xuất thông tin chính xác và sinh câu trả lời phù hợp với ngữ cảnh. Cụ thể, quá trình bao gồm: Xây dựng cơ sở dữ liệu vector - nơi dữ liệu được xử lý và mã hóa thành dạng vector để phục vụ truy vấn và Truy xuất và tạo câu trả lời giai đoạn mà hệ thống tìm kiếm thông tin liên quan dựa trên câu hỏi và sử dụng mô hình ngôn ngữ để tạo câu trả lời hoàn chỉnh. Hai thành phần này phối hợp chặt chẽ để tận dụng tối đa dữ liệu ngữ cảnh và giảm thiểu hiện tượng ảo giác của mô hình ngôn ngữ lớn.

#### **1.8.2.1. Xây dựng cơ sở dữ liệu vector**

Bước này sẽ thực hiện chuyển đổi toàn bộ dữ liệu thành các vector và lưu trữ chúng vào một cơ sở dữ liệu vector. Trước đó sẽ có các bước chia nhỏ toàn bộ dữ liệu thành các đoạn nhỏ rồi sử dụng mô hình nhúng để embedding các đoạn thành các vector và cuối cùng lưu trữ các vector vào cơ sở dữ liệu vector.

A diagram of data indexing

AI-generated content may be incorrect.

Hình 1.25: Chuẩn bị dữ liệu và đẩy lên database

1. **Thu thập và chuẩn hóa dữ liệu**

Dữ liệu đầu vào có thể đến từ nhiều nguồn như website, tài liệu PDF, Word, hệ thống quản lý tài liệu nội bộ... Tất cả dữ liệu này cần được thu thập và đưa về một định dạng chuẩn (thường là dạng văn bản thuần – plain text) để dễ xử lý trong các bước tiếp theo.

1. **Chia nhỏ văn bản (Chunking)**

Văn bản thu thập được thường dài và chứa nhiều nội dung không liên quan đến nhau. Để tối ưu hóa khả năng tìm kiếm và phù hợp với giới hạn của mô hình ngôn ngữ, văn bản được chia nhỏ thành các đoạn (chunks) có độ dài hợp lý, ví dụ khoảng 100–300 từ mỗi đoạn, đảm bảo mỗi đoạn vẫn giữ được ý nghĩa độc lập.

1. **Chuyển văn bản thành vector (Embedding)**

Mỗi đoạn văn bản sau khi được chia nhỏ sẽ được đưa qua một mô hình nhúng (embedding model) để chuyển đổi thành một vector số học có chiều cố định (ví dụ 384, 768 hoặc 1024 chiều). Một số mô hình embedding phổ biến là:

* Sentence-BERT (SBERT)
* MiniLM
* Instructor-XL
* OpenAI Ada Embedding (nếu dùng API)

Mỗi đoạn văn bản sau khi được chia nhỏ sẽ được đưa qua một mô hình nhúng (embedding model) để chuyển đổi thành một vector số học có chiều cố định (ví dụ 384, 768 hoặc 1024 chiều). Một số mô hình embedding phổ biến là:

1. **Lưu trữ vector vào cơ sở dữ liệu vector**

Các vector sau khi được sinh ra sẽ được lưu trữ vào một cơ sở dữ liệu vector để phục vụ việc tìm kiếm nhanh sau này. Mỗi vector sẽ đi kèm với metadata như:

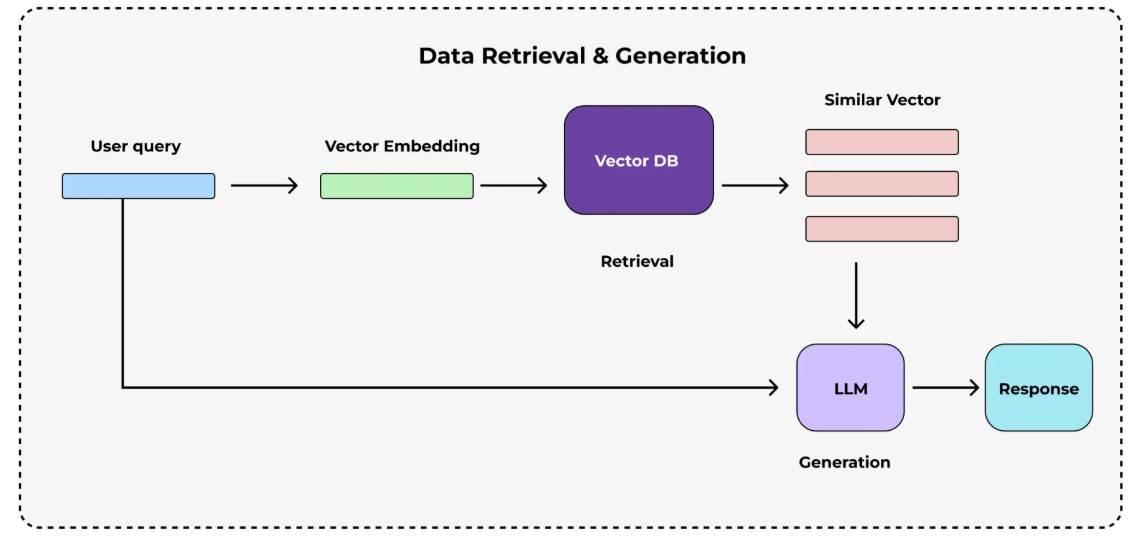
* Nội dung gốc của đoạn văn bản.
* Thông tin nguồn (tên tài liệu, đường dẫn, ngày cập nhật...).
* ID hoặc mã định danh phục vụ truy xuất.

Một số cơ sở dữ liệu vector phổ biến được sử dụng:

* Milvus: hiệu suất cao, mã nguồn mở, dễ tích hợp.
* FAISS: Facebook AI Similarity Search – nhẹ, hiệu quả, chạy local.
* Weaviate, Pinecone, Qdrant: dễ tích hợp API, có nhiều tính năng nâng cao.

#### **1.8.2.2. Truy xuất và tạo câu trả lời**

Sau khi đã xây dựng xong cơ sở dữ liệu vector, hệ thống sẽ thực hiện quy trình truy xuất thông tin và tạo câu trả lời mỗi khi người dùng gửi một truy vấn. Quá trình này bao gồm các bước sau:



Hình 1.26: Truy vấn dữ liệu và tạo câu trả lời

**a. Nhúng truy vấn người dùng (Query Embedding)**

Khi người dùng đặt câu hỏi, truy vấn đó sẽ được xử lý và chuyển đổi thành một vector embedding bằng cùng một mô hình nhúng (embedding model) đã sử dụng trong quá trình xây dựng cơ sở dữ liệu vector. Điều này đảm bảo rằng truy vấn và các đoạn văn bản được biểu diễn trong cùng một không gian vector, giúp việc so sánh và tìm kiếm trở nên chính xác hơn.

**b. Tìm kiếm các đoạn văn bản liên quan (Retrieval)**

Vector truy vấn sẽ được sử dụng để tìm các đoạn văn bản gần nhất trong cơ sở dữ liệu vector bằng cách đo **độ tương đồng cosine** hoặc các phương pháp tính khoảng cách khác. Hệ thống sẽ trả về **k đoạn văn bản gần nhất** (top-k), thường là 3–5 đoạn, được coi là ngữ cảnh phù hợp nhất với câu hỏi của người dùng.

**c. Tạo prompt ngữ cảnh (Contextual Prompt Construction)**

Các đoạn văn bản truy xuất được sẽ được kết hợp với câu hỏi người dùng thành một prompt hoàn chỉnh.

Prompt này sẽ đóng vai trò là đầu vào cho mô hình ngôn ngữ lớn (LLM).

**d. Sinh câu trả lời (Answer Generation)**

Mô hình ngôn ngữ lớn (như GPT, LLaMA, Mistral, Claude...) sẽ nhận prompt ngữ cảnh để sinh ra câu trả lời dựa trên các thông tin đã được cung cấp. Vì prompt đã chứa ngữ cảnh cụ thể từ nguồn dữ liệu nên mô hình sẽ trả lời chính xác và phù hợp hơn, đồng thời giảm đáng kể hiện tượng "ảo giác" (hallucination).

### **1.8.3. Ưu nhược điểm**

RAG cho phép LLMs khai thác dữ liệu bên ngoài mà không cần huấn luyện lại mô hình, giúp tiết kiệm thời gian và tài nguyên, đồng thời vẫn hiệu quả ngay cả khi dữ liệu gán nhãn còn hạn chế. Tuy nhiên, hiệu suất của RAG phụ thuộc nhiều vào độ chính xác của mô hình truy xuất và chất lượng kho tri thức. RAG đặc biệt phù hợp cho các ứng dụng như trợ lý ảo, chatbot – những hệ thống cần truy xuất thông tin theo thời gian thực để trả lời câu hỏi một cách chính xác và mạch lạc.

# **CHƯƠNG II: PHÁT TRIỂN CHATBOT**

Chương này tập trung vào việc phát triển một hệ thống Chatbot tích hợp kỹ thuật Retrieval-Augmented Generation (RAG) cho Trường Đại học Công nghiệp Hà Nội, sử dụng các mô hình ngôn ngữ lớn (LLMs). Hệ thống sử dụng cơ sở dữ liệu vector kết hợp với RAG để tăng độ chính xác khi truy xuất thông tin. Quá trình xây dựng được trình bày chi tiết từ giai đoạn thu thập và xử lý dữ liệu, thiết kế kiến trúc hệ thống.

## **2.1. Tổng quan ý tưởng hệ thống**

Hệ thống chatbot thông minh phục vụ Trường Đại học Công nghiệp Hà Nội hướng tới việc tự động hóa các tác vụ hỗ trợ tra cứu thông tin như: giới thiệu trường, thông tin tuyển sinh, các chương trình đào tạo, dự án tuyển sinh và các thông tin liên quan khác. Thay vì tinh chỉnh toàn bộ mô hình ngôn ngữ lớn (LLM) – vốn đòi hỏi chi phí và tài nguyên tính toán cao, hệ thống tập trung áp dụng kỹ thuật Retrieval-Augmented Generation (RAG) nhằm tận dụng sức mạnh của mô hình ngôn ngữ đã được huấn luyện trước kết hợp với truy xuất thông tin từ kho dữ liệu nội bộ.

Trọng tâm của hệ thống là tích hợp LLM với cơ sở dữ liệu vector, cho phép chatbot hiểu và phản hồi chính xác các truy vấn tiếng Việt dựa trên nguồn dữ liệu được cập nhật liên tục. Qua đó, chatbot đóng vai trò như một trợ lý số thông minh, hỗ trợ hiệu quả cho bộ phận tư vấn và nâng cao trải nghiệm người dùng trong môi trường giáo dục đại học.

## **2.2. Chuẩn bị dữ liệu**

### **2.2.1. Thu thập dữ liệu**

Địa chỉ website: <https://www.haui.edu.vn/vn>

Các bước thực hiện:

* Duyệt và phân tích cấu trúc của website, bao gồm các trang chính và chuyên mục như: *Giới thiệu, Đào tạo, Khoa học công nghệ, Hợp tác quốc tế, Tuyển sinh...*
* Thu thập và lưu trữ nội dung từ các bài viết, tin tức, thông báo liên quan đến: lịch sử, quy mô đào tạo, hội trường trường và thông tin các khoa,…
* Lưu trữ dữ liệu đã thu thập vào các file văn bản (.txt) để phục vụ xử lý và phân tích.

Kết quả thu được các file text (.txt):

* Tổng quan về trường: lịch sử, đội ngũ cán bộ - giảng viên, danh hiệu thi đua, cơ sở vật chất.
* Cơ cấu tổ chức: khoa, các tổ chức đoàn thể, ban giám hiệu, hội đồng trường.
* Chiến lược phát triển.
* Thông tin về một số khoa: khoa điện tử, công nghệ thông tin, điện.
* Thông tin đào tạo: đại học và sau đại học.

### **2.2.2. Đẩy dữ liệu lên cơ sở dữ liệu vector**

Hiện nay, có nhiều cơ sở dữ liệu vector phổ biến được sử dụng trong các ứng dụng trí tuệ nhân tạo và tìm kiếm ngữ nghĩa, bao gồm: Milvus, FAISS (Facebook AI Similarity Search), Pinecone, Weaviate, Qdrant, Vespa, và Redis (Vector Search). Các hệ thống này cho phép lưu trữ và tìm kiếm các vector có độ chiều cao, thường được dùng trong các bài toán như tìm kiếm văn bản, hình ảnh, âm thanh dựa trên ngữ nghĩa hoặc đặc trưng trích xuất bởi mô hình học sâu. Trong số đó, Milvus nổi bật là một giải pháp mã nguồn mở mạnh mẽ, được phát triển bởi Zilliz, hỗ trợ tìm kiếm hàng tỷ vector với tốc độ cao và khả năng mở rộng tốt.

Một số ưu điểm đáng chú ý như: hiệu năng cao, hỗ trợ nhiều thuật toán tìm kiếm gần đúng (ANN) như IVF, HNSW, khả năng mở rộng theo chiều ngang, và tích hợp tốt với các công cụ AI như LangChain, LlamaIndex, HuggingFace. Milvus cũng cho phép gắn metadata cho vector và thực hiện truy vấn có điều kiện tương tự SQL, giúp quản lý dữ liệu hiệu quả. Giao diện lập trình thân thiện, hỗ trợ đa ngôn ngữ (Python, Java, Go, REST API), cùng khả năng triển khai linh hoạt trên Docker hoặc Kubernetes khiến Milvus trở thành lựa chọn lý tưởng cho các hệ thống như chatbot tích hợp RAG, tìm kiếm ngữ nghĩa hoặc hệ thống đề xuất nội dung.

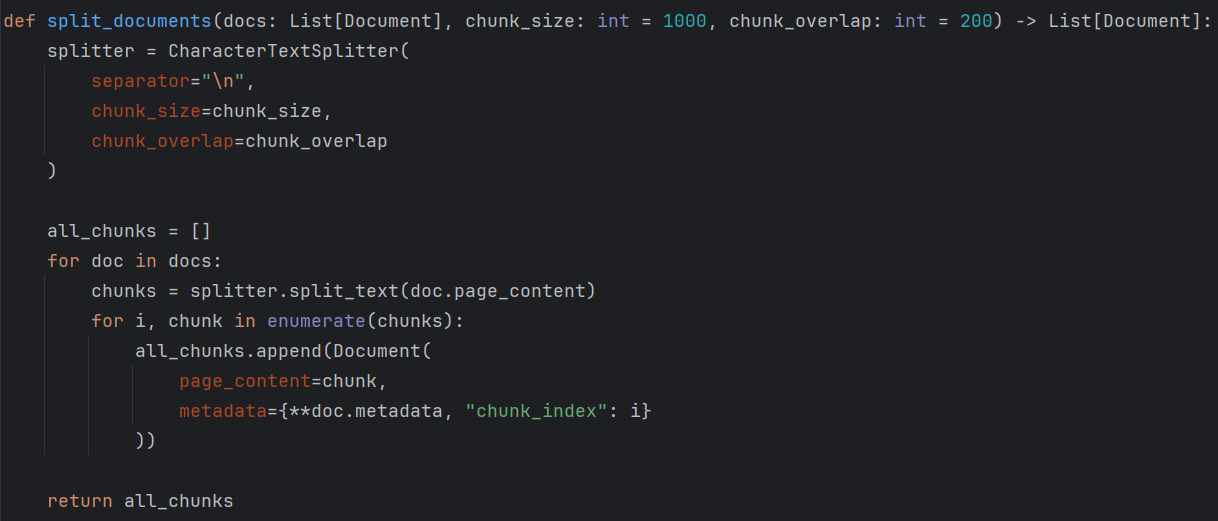
A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.1: Giao diện của cơ sở dữ liệu milvus

Các bước để đẩy dữ liệu lên milvus:

Bước 1: Sau khi có dữ liệu ở dạng text (.txt), ta sẽ tiến hành chia văn bảnthành các đoạn nhỏ (chunk) để phù hợp với quá trình tạo vector. Chia đoạn dựa theo số lượng từ 1000 từ mỗi đoạn.



*Hình 2.2: Tiến hành chia tài liệu thành các đoạn (chunk) nhỏ*

Bước 2: Sau khi chia đoạn, mỗi đoạn văn bản sẽ được chuyển đổi thành vector thông qua mô hình sinh embedding.

A screen shot of a computer

AI-generated content may be incorrect.

Hình 2.3: Sử dụng mô hình embedding để tạo vector

Bước 3: Kết nối và đẩy dữ liệu lên milvus

A computer screen with colorful text

AI-generated content may be incorrect.

Hình 2.4: Lưu dữ liệu ở cả dạng text và vector

## **2.3. Truy xuất dữ liệu và tạo câu trả lời**

Sau khi đã xây dựng và lưu trữ dữ liệu trong vector database Milvus, bước tiếp theo là truy xuất dữ liệu phù hợp để phục vụ việc tạo ra câu trả lời cho người dùng. Quá trình này bao gồm hai thành phần chính:

* Xây dựng công cụ tìm kiếm thông tin (retriever) từ Milvus kết hợp với BM25.
* Tối ưu lại prompt để tạo đầu vào cho LLM.
* Sử dụng mô hình ngôn ngữ lớn (LLM) như LLaMA chạy local qua Ollama.

### **2.3.1. Xây dựng công cụ tìm kiếm thông tin**

Đây là bước có thể nói là quan trọng nhất trong quy trình hoạt động của hệ thống. Ở đây sẽ là nơi tăng cường thông tin về câu hỏi của người dùng cho LLM.

Các bước thực hiện:

* Bước 1: Tạo retriever tìm kiếm theo vector và trả về 4 tài liệu có độ tương đồng lớn nhất.
* • Bước 2: Từ 4 tài liệu có độ tương đồng cao nhất, tiếp tục khởi tạo một retriever thứ hai sử dụng thuật toán BM25 nhằm tận dụng khả năng tìm kiếm theo từ khóa truyền thống. BM25 sẽ rà soát lại các tài liệu này để đánh giá mức độ liên quan dựa trên tần suất xuất hiện của các từ trong câu hỏi. Điều này giúp làm nổi bật những tài liệu chứa các từ khóa trực tiếp mà truy vấn yêu cầu.
* Bước 3: Kết hợp cả hai phương pháp truy xuất trên thành một EnsembleRetriever, với trọng số ưu tiên cao hơn cho truy vấn vector (ví dụ: 70% vector + 30% BM25). Điều này đảm bảo rằng hệ thống vừa khai thác được ngữ nghĩa sâu sắc của câu hỏi, vừa không bỏ sót những từ khóa quan trọng. Kết quả cuối cùng là một danh sách tài liệu được sắp xếp theo mức độ liên quan tổng hợp, dùng làm “context” đầu vào cho mô hình ngôn ngữ lớn (LLM) sinh câu trả lời.

A computer screen with many colorful text

AI-generated content may be incorrect.

Hình 2.5: Tạo công cụ tìm kiếm tài liệu

Nhờ việc kết hợp linh hoạt giữa tìm kiếm ngữ nghĩa và từ khóa này, hệ thống có thể cung cấp những câu trả lời vừa chính xác về mặt nội dung, vừa phù hợp với ngữ cảnh của người dùng. Đây chính là bước "retrieval" quan trọng trong mô hình RAG (Retrieval-Augmented Generation).

### **2.3.2. Tạo câu trả lời**

Sau khi đã truy xuất được các tài liệu liên quan thông qua bước tìm kiếm kết hợp vector và BM25, hệ thống sẽ sử dụng mô hình ngôn ngữ lớn (LLM) để sinh câu trả lời cho người dùng. Sau quá trình khảo sát em quyết định chọn mô hình LLaMA3.1:8B được chạy local thông qua nền tảng Ollama. Việc sử dụng LLM cài đặt cục bộ giúp giảm đáng kể chi phí vận hành so với các giải pháp đám mây như OpenAI hoặc Azure, đồng thời đảm bảo dữ liệu không bị gửi ra ngoài, tăng cường tính bảo mật.

Quá trình tạo câu trả lời diễn ra như sau:

* Bước 1: Các tài liệu đã được truy xuất từ bước trước được đưa vào làm "context" cho mô hình.
* Bước 2: Hệ thống khởi tạo một prompt template chứa phần hướng dẫn cố định cho mô hình và phần nội dung của người dùng (question).
* Bước 3: Prompt này cùng với context sẽ được truyền vào mô hình LLaMA3.1:8B để sinh ra câu trả lời bằng tiếng Việt phù hợp, chính xác và bám sát vào dữ liệu nội bộ đã cung cấp.

## **2.4. Xây dựng giao diện và các chức năng hệ thống với Django**

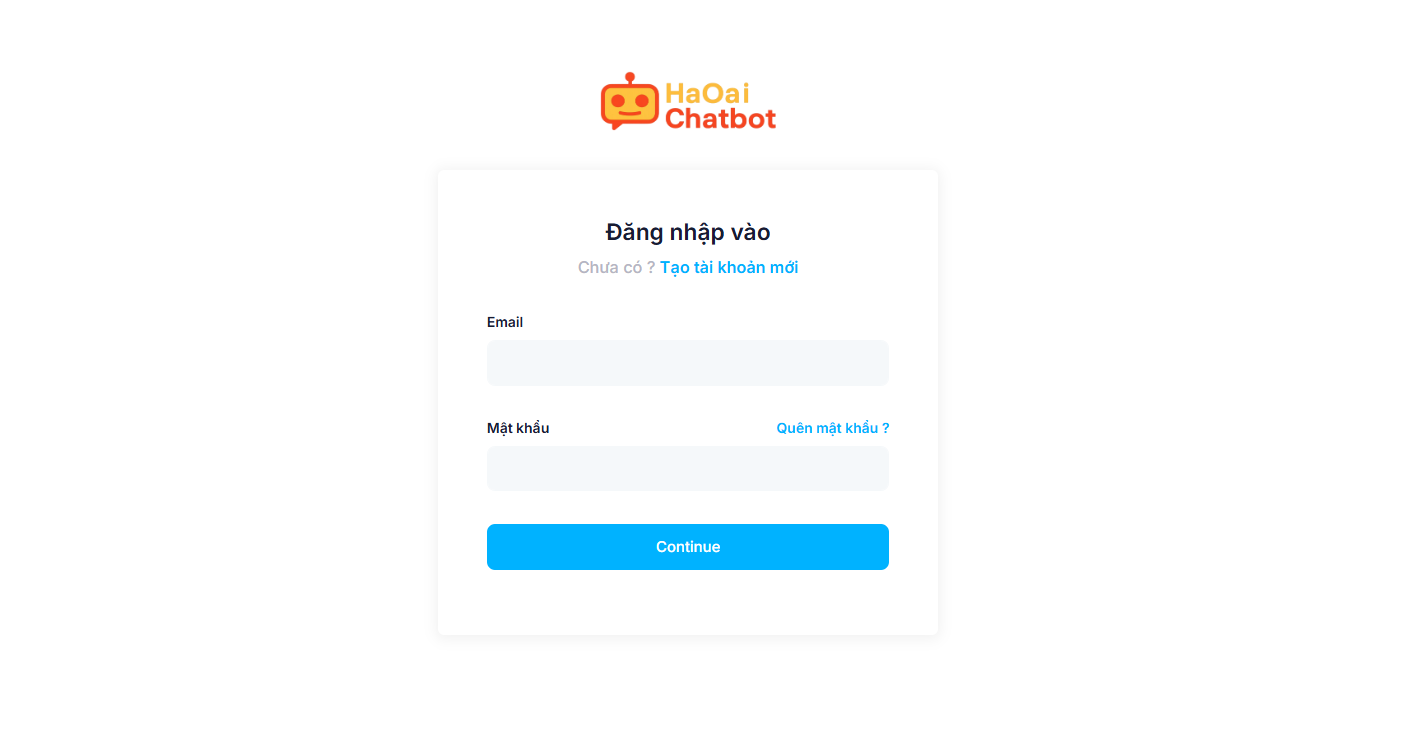
### **2.4.1. Lý do lựa chọn Django Framework**

* Django là một framework Python mạnh mẽ, dễ mở rộng và được hỗ trợ cộng đồng lớn.
* Có sẵn hệ thống quản lý người dùng (login/logout), session, CSRF bảo mật.
* Tích hợp dễ dàng với các API xử lý phía backend như truy vấn LLM, kết nối Redis, RabbitMQ, Milvus...
* Phù hợp cho việc xây dựng hệ thống chatbot có nhiều chức năng web frontend.

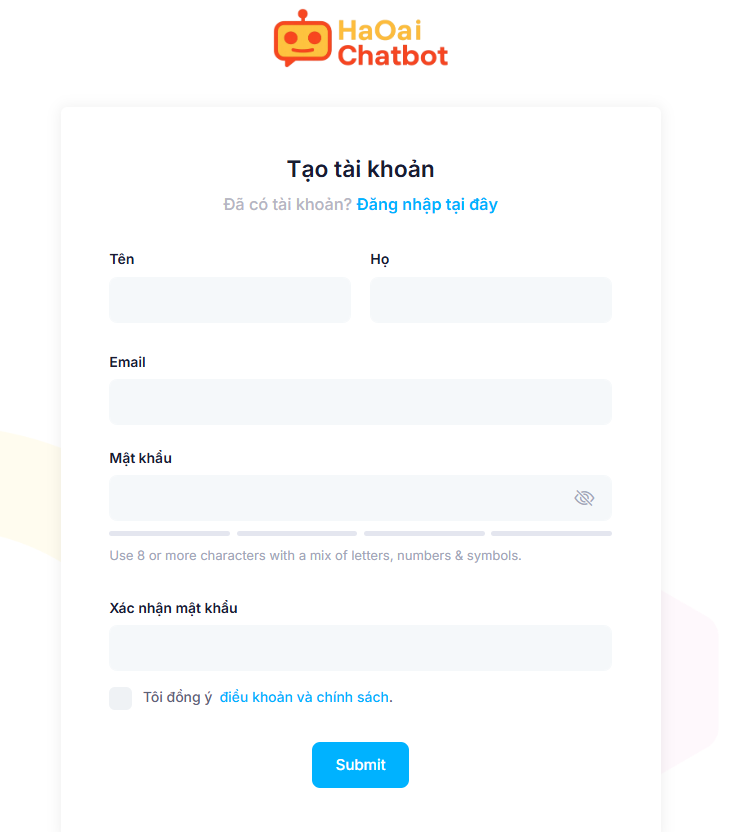
### **2.4.2. Kiến trúc tổng quan hệ thống web**

* Frontend: Sử dụng Django templates kết hợp với JavaScript để hiển thị giao diện trò chuyện động.
* Backend: Django xử lý routing, xác thực người dùng, giao tiếp với hệ thống truy xuất thông tin và LLM để sinh câu trả lời.
* Cơ sở dữ liệu: Postgres để lưu người dùng, tài liệu đã tải lên,..

### **2.4.3. Một số hình ảnh kết quả**



Hình 2.6: Trang đăng nhập



Hình 2.7: Trang đăng ký tài khoản

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.8: Trang tải tài liệu lên cơ sở dữ liệu vector

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.9: Trang quản lý tài liệu đã tải lên

A white rectangular object with a black border

AI-generated content may be incorrect.

Hình 2.10: Giao diện chat với trợ lý ảo HaOai

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.11: Giao diện trang chủ

# **CHƯƠNG III: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN**

## **3.1. Kết quả đạt được**

Sau quá trình nghiên cứu và triển khai, hệ thống chatbot tích hợp RAG đã đạt được một số kết quả cụ thể như sau:

1. Hoạt động ổn định trên môi trường local

* Giao diện web được xây dựng bằng Django Framework.
* Các chức năng chính hoạt động đầy đủ: đăng nhập, đăng xuất, giao diện trò chuyện, tải và quản lý tài liệu.

1. Hoàn thiện pipeline RAG

* Thu thập và xử lý dữ liệu.
* Sử dụng Milvus làm cơ sở dữ liệu vector để lưu trữ embedding.
* Triển khai retriever kết hợp giữa tìm kiếm vector và BM25 (EnsembleRetriever).
* Tích hợp mô hình LLaMA 3.1 8B chạy qua Ollama để sinh câu trả lời.

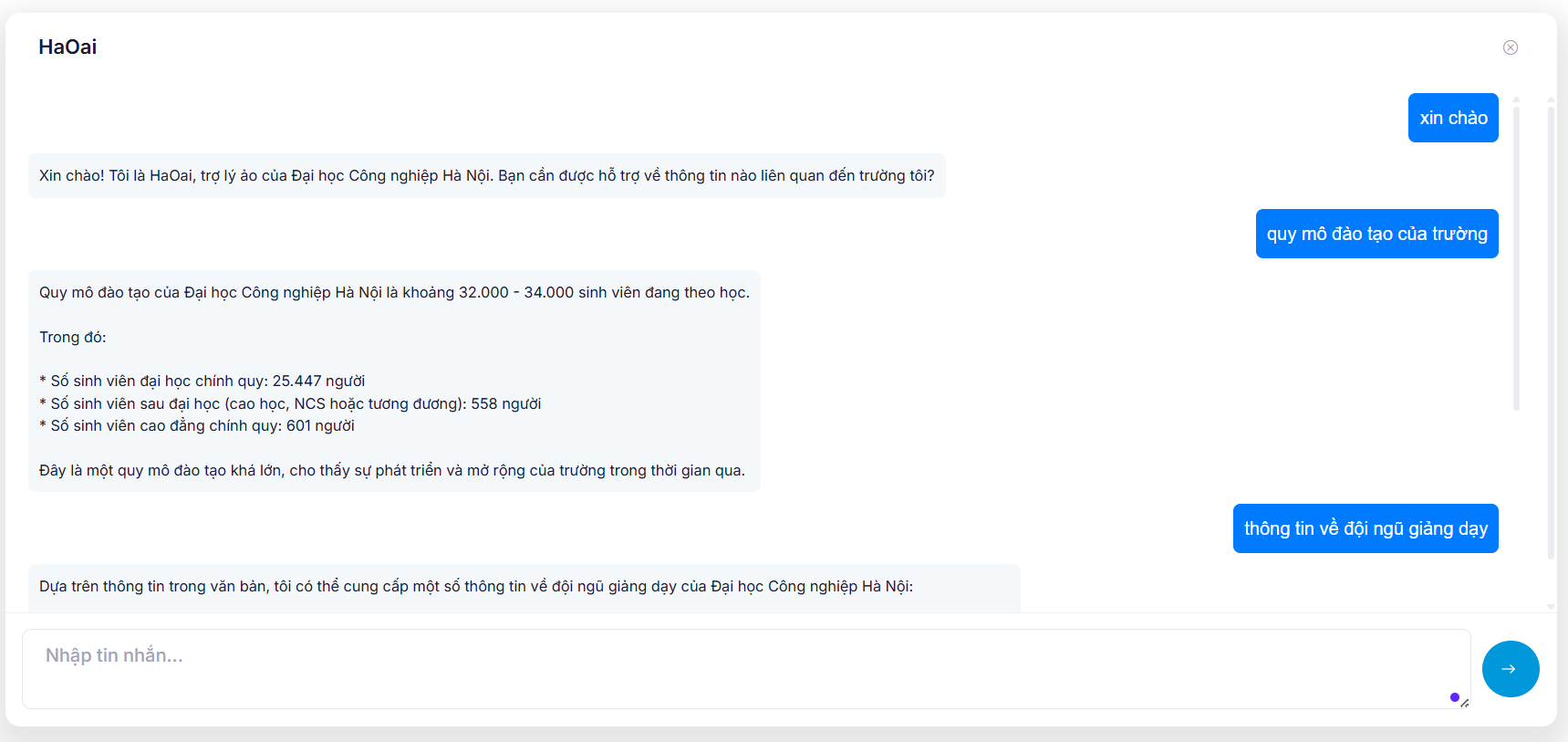
1. Tối ưu chi phí và đảm bảo tính riêng tư

* Không phụ thuộc vào API dịch vụ AI bên ngoài (OpenAI, Anthropic…), toàn bộ pipeline chạy local.
* Có thể mở rộng hoặc triển khai nội bộ tại các tổ chức giáo dục hoặc doanh nghiệp.

1. Trả lời chính xác theo ngữ cảnh câu hỏi

* Hệ thống trả lời có căn cứ dựa trên dữ liệu đã lưu trữ, hạn chế "hallucination".
* Hữu ích trong các tình huống như tư vấn tuyển sinh, giới thiệu trường, chương trình đào tạo.

1. Hình ảnh về phản hồi của chatbot:



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.1: Một số hình ảnh phản hồi của chatbot

## **3.2. Hạn chế**

* Hiệu suất: Khi dữ liệu lớn hơn, thời gian truy xuất và trả lời có thể tăng đáng kể.
* Khả năng cập nhật dữ liệu động: Chưa hỗ trợ đồng bộ dữ liệu mới theo thời gian thực.
* Khả năng mở rộng: Chưa triển khai cơ chế multi-user hoặc phân quyền truy cập.

## **3.2. Hướng phát triển**

Trong tương lai, hệ thống chatbot tích hợp RAG có thể được mở rộng và cải tiến theo các hướng sau:

1. Nâng cao chất lượng câu trả lời

* Fine-tuning thêm mô hình LLaMA hoặc các mô hình phù hợp với dữ liệu giáo dục trong nước.
* Áp dụng re-ranking để chọn lọc tài liệu trước khi trả lời.

1. Tích hợp dữ liệu thời gian thực

* Thiết kế cơ chế tự động crawl nội dung mới từ website và đồng bộ vào vector database.
* Kết hợp chatbot với hệ thống quản lý học tập (LMS) để truy xuất tài liệu khóa học, lịch thi, thông báo.

1. Phát triển hệ thống đa nền tảng

* Xây dựng API để tích hợp chatbot với Zalo, Messenger, Telegram...

1. Đảm bảo bảo mật và phân quyền truy cập

* Thêm cơ chế phân quyền truy vấn dữ liệu theo vai trò (sinh viên, cán bộ, giảng viên...).
* Mã hóa dữ liệu và log truy cập để bảo vệ thông tin nội bộ.

# **KẾT LUẬN**

Trong bối cảnh chuyển đổi số và ứng dụng trí tuệ nhân tạo ngày càng phổ biến trong giáo dục, việc xây dựng một hệ thống chatbot thông minh có khả năng trả lời các câu hỏi dựa trên dữ liệu nội bộ của tổ chức là một hướng đi thiết thực và giàu tiềm năng. Đề tài *"Phát triển chatbot tích hợp RAG (Retrieval-Augmented Generation)"* đã tiếp cận hướng nghiên cứu này thông qua việc ứng dụng mô hình truy xuất kết hợp (RAG), kết nối giữa cơ sở dữ liệu vector (Milvus) và mô hình ngôn ngữ lớn (LLM) như LLaMA.

Hệ thống đã được thiết kế và triển khai với các thành phần rõ ràng: thu thập và xử lý dữ liệu từ website và văn bản chính thức, xây dựng vector database, phát triển mô-đun truy xuất thông tin kết hợp giữa tìm kiếm vector và BM25, đồng thời tích hợp LLM chạy local qua Ollama để tối ưu chi phí. Toàn bộ quá trình được tổ chức trong một ứng dụng web sử dụng Django framework với đầy đủ các chức năng như đăng nhập, trò chuyện, lưu lịch sử, và giao diện thân thiện với người dùng.

Kết quả thực nghiệm cho thấy hệ thống có khả năng trả lời các câu hỏi một cách chính xác, linh hoạt và thân thiện, phù hợp với mục tiêu tư vấn tuyển sinh, cung cấp thông tin học tập – nghiên cứu trong môi trường đại học. Trong tương lai, hệ thống có thể được mở rộng thêm các tính năng như học liên tục từ dữ liệu mới, tích hợp nhiều nguồn dữ liệu khác nhau, cải thiện trải nghiệm người dùng và hỗ trợ đa ngôn ngữ để đáp ứng nhu cầu ngày càng cao của người sử dụng.

**TÀI LIỆU THAM KHẢO**

1. Nguyễn Thanh Tuấn (2019), Deep learning cơ bản, NTTUAN8.COM.
2. Charu C Aggarwal (2018), "Neural networks and deep learning",

Springer. 10(978), tr. 3.

1. Antonio Gulli Amita Kapoor, Sujit pal (2022), Deep learning with

tensorflow and keras, Packt Publishing Ltd.

1. Meta (2023). LLaMA: Meta AI’s Open Language Model.
2. **Hugging Face (2024)**. *Transformers Documentation*.

https://huggingface.co/docs/transformers/index

1. Vaswani, A., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems. <https://arxiv.org/abs/1706.03762>