# TÓM TẮT

Nội dung báo cáo được tổ chức bao gồm phần mở đầu và 03 chương chính với nội dung cụ thể trong mỗi phần như sau:

Phần mở đầu sẽ trình bày bối cảnh của đề tài, lý do lựa chọn đề tài này, nói rõ về những khó khăn hiện nay trong việc học tập bộ môn Lịch sử ở Việt Nam và nguyên nhân sử dụng mô hình ngôn ngữ lớn (LLM) làm nền tảng cho giải pháp. Ngoài ra, trong phần này cũng sẽ nêu rõ các phạm vi của đề tài cũng như mục tiêu dự kiến để triển khai các nội dung chi tiết.

Chương 1 – Cơ sở lý thuyết sẽ trình bày các cơ sở lý thuyết về Học máy (ML), Học sâu (DL), Xử lý ngôn ngữ tự nhiên (NLP), mô hình Transformer và mô hình ngôn ngữ lớn (LLM). Ngoài ra, trong chương này cũng sẽ trình bày các công nghệ Trí tuệ nhân tạo (AI) và các công nghệ hỗ trợ khác được sử dụng trong hệ thống, từ đó làm rõ các cơ sở khoa học và những tiềm năng để ứng dụng trong hệ thống chatbot hỗ trợ học Lịch sử Việt Nam.

Chương 2 – Phương pháp đề xuất xây dựng chatbot sẽ trình bày về phương pháp nghiên cứu, quá trình gồm các công đoạn như thu thập và xử lý dữ liệu, xây dựng các thành phần, thực hiện embedding dữ liệu, thiết kế và xây dựng thuật toán để truy xuất thông tin và thiết kế prompt; tất cả đều nhằm mục đích là tối ưu hiệu quả của chatbot.

Chương 3 – Triển khai hệ thống chatbot hỗ trợ học Lịch sử Việt Nam sẽ tập trung vào mô tả về quá trình triển khai hệ thống chatbot từ đầu tới cuối, gồm có xây dựng back-end, tích hợp các chức năng của hệ thống chatbot như trả lời câu hỏi, ra đề thi. Hiệu quả của hệ thống sẽ được đánh giá qua các thông số thu được sau khi thực nghiệm.

Phần cuối sẽ tổng kết lại những kết quả đã thu được, những đánh giá về hiệu quả của hệ thống và các đề xuất để cải thiện và phát triển hệ thống này trong tương lai.

# LỜI NÓI ĐẦU

Trong thời đại ngày nay, cùng với sự phát triển nhanh chóng của công nghệ, trí tuệ nhân tạo (AI) đã và đang thể hiện vai trò ngày càng chủ chốt trong rất nhiều lĩnh vực của cuộc sống. Nhờ có AI mà công việc xây dựng các hệ thống thông minh, có khả năng tự động hóa cao đã và đang trở nên khả thi hơn bao giờ hết. Từ sự phát triển của AI, các mô hình ngôn ngữ lớn (LLM) đã ra đời và đang ngày một được cải tiến. Giờ đây, các mô hình LLM không chỉ có thể xử lý các quy luật phức tạp trong ngôn ngữ tự nhiên mà còn trở thành nền tảng cho các chatbot hiện đại, với năng lực giao tiếp rất tiệm cận với mức độ của con người và có thể dễ dàng được tùy chỉnh cho phù hợp với các mục đích sử dụng đa dạng. Trong bối cảnh này, ý tưởng về một chatbot hỗ trợ học Lịch sử Việt Nam trở thành một giải pháp khả thi và rất có tiềm năng để hỗ trợ trong giáo dục nói chung và trong giáo dục bộ môn Lịch sử Việt Nam nói riêng, nâng cao trải nghiệm giáo dục và giảm thiểu các chi phí sẽ phải dành ra cho các công việc mang tính lặp đi lặp lại.

Tại Việt Nam, sau những biến động lớn nhỏ trên môi trường quốc tế trong thời gian gần đây, các vấn đề có liên quan đến lịch sử Việt Nam đang trở nên nổi bật và ngày càng thu hút được nhiều sự quan tâm của cả trong nước cũng như bạn bè thế giới. Cùng với đó, công tác giáo dục bộ môn lịch sử Việt Nam đang được tập trung cải thiện và phát triển. Tuy nhiên, phương pháp giáo dục lịch sử hiện tại ở Việt Nam, với nghe giảng, đọc tài liệu và ghi chép làm trọng, vẫn còn mang nặng tính truyền thống và có phần khô khan, khó tiếp cận. Ngoài ra, với khối lượng dữ liệu lớn, phương pháp giáo dục này khiến người học phải mất rất nhiều thời gian và chi phí đi kèm đề có thể tiếp thu, ghi nhớ các điểm kiến thức và liên kết chúng với nhau. Vì vậy, sử dụng một chatbot hỗ trợ học Lịch sử Việt Nam dựa trên nền tảng của LLM không những góp phần cải thiện phương pháp giáo dục hiện có, mà còn có thể xem là một bước tiến lớn trong áp dụng tự động hóa và ứng dụng các công nghệ tiên tiến trong giáo dục nói chung và giáo dục lịch sử nói riêng. Chatbot hỗ trợ học Lịch sử Việt Nam, ngoài khả năng trả lời các câu hỏi về chủ đề lịch sử Việt Nam một cách chính xác, mà còn có thể đóng vai trò là một người bạn hay một người giáo viên, giúp nâng cao trải nghiệm trong giáo dục và học tập lịch sử Việt Nam.

Đề tài này hướng tới mục tiêu là xây dựng một chatbot hỗ trợ học tập Lịch sử Việt Nam, được phát triển dựa trên nền tảng là các LLM hiện đại. Thông qua khả năng xử lý ngôn ngữ tự nhiên mạnh mẽ, chatbot sẽ giúp người học tiếp cận với các kiến thức lịch sử Việt Nam một cách linh hoạt và có tính tương tác cao hơn so với phương pháp truyền thống hiện có. Hệ thống được thiết kế và xây dựng để trả lời các câu hỏi liên quan đến lịch sử Việt Nam theo ngữ cảnh, đồng thời có thể đóng vai trò ra đề thi để người học ôn tập lại kiến thức. Ngoài ra, hệ thống còn được thiết kế để có thể tích hợp các chức năng mở rộng khác trong tương lai, như là trao đổi bằng giọng nói, tìm kiếm tài liệu liên quan. Báo cáo này sẽ trình bày toàn bộ quá trình nghiên cứu, phân tích và triển khai hệ thống chatbot này từ những bước đầu tiên cho đến khi hoàn thiện, qua đây góp phần để khẳng định vai trò của AI trong đổi mới phương thức dạy và học bộ môn Lịch sử Việt Nam.

# LỜI CAM ĐOAN

Em xin cam đoan rằng tất cả các kết quả và thông tin trình bày trong đề tài này là thành quả công việc của em dưới dự hướng dẫn của giảng viên TS. Nguyễn Năng Hùng Vân. Em đã tuân thủ mọi nguyên tắc và quy định về nghiên cứu khoa học và đạo đức trong việc thực hiện đồ àn tốt nghiệp này.

Tất cả các nguồn tham khảo và các công trình nghiên cứu trước đây của người khác đã được trích dẫn và trình bày một cách rõ ràng, chi tiết trong danh mục tài liệu tham khảo của đồ án. Em đã tuân thủ nguyên tắc không sao chép, không vi phạm bản quyền, và đã tuân thủ đúng quy định về việc ghi rõ nguồn gốc thông tin từ nguồn khác.

Các kết quả đánh giá và thống kê được trình bày trong khóa luận này đều là kết quả của những thực nghiệm đã được em tự thực hiện, và không có sự can thiệp từ bất kỳ nguồn nào khác. Em xin cam đoan tính trung thực và chính xác trong việc trình bày kết quả và thông tin trong khóa luận này.

**Sinh viên thực hiện**

Lê Việt Trung

# LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành và sâu sắc đến TS. Nguyễn Năng Hùng Vân – giảng viên hướng dẫn trong suốt quá trình thực hiện đồ án tốt nghiệp, thầy đã tận tâm chỉ bảo và định hướng cho em trong quá trình hoàn thiện đồ án. Những ý kiến đóng góp quý báu và sự động viên của thầy đã giúp em vượt qua nhiều khó khăn và hoàn thành tốt nhiệm vụ được giao.

Em cũng xin chân thành cảm ơn Ban giám hiệu nhà trường, quý thầy cô trong khoa Công nghệ thông tin và toàn thể giảng viên của Trường Đại học Bách khoa Đà Nẵng, những người đã tạo điều kiện thuận lợi và cung cấp kiến thức nền tảng để em có thể thực hiện đồ án tốt nghiệp cách tốt nhất.

Cuối cùng, em xin dành lời cảm ơn đặc biệt đến gia đình, những người luôn bên cạnh, động viên và tạo mọi điều kiện tốt nhất để em hoàn thành chặng đường học tập và thực hiện đồ án này.

Dù đã rất cố gắng, nhưng chắc chắn đồ án này vẫn còn những thiếu sót nhất định, em rất mong nhận được sự đóng góp và chỉ dẫn từ quý thầy cô để có thể cải thiện và phát triển hơn trong tương lai.

MỤC LỤC

[TÓM TẮT I](#_Toc198742299)

[LỜI NÓI ĐẦU II](#_Toc198742300)

[LỜI CAM ĐOAN III](#_Toc198742301)

[LỜI CẢM ƠN IV](#_Toc198742302)

[MỤC LỤC V](#_Toc198742303)

[DANH SÁCH CÁC HÌNH ẢNH X](#_Toc198742304)

[DANH SÁCH CÁC KÝ HIỆU VIẾT TẮT XII](#_Toc198742305)

[MỞ ĐẦU 1](#_Toc198742306)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 6](#_Toc198742307)

[1.1. Giới thiệu đề tài 6](#_Toc198742308)

[1.1.1. Tổng quan đề tài 6](#_Toc198742309)

[1.1.2. Nội dung nghiên cứu 6](#_Toc198742310)

[1.2. Lược sử về Xử lý ngôn ngữ tự nhiên 7](#_Toc198742311)

[1.3. Tổng quan về Học máy 7](#_Toc198742312)

[1.3.1. Giới thiệu về Học máy 7](#_Toc198742313)

[1.3.2. Phân loại Học máy 8](#_Toc198742314)

[1.3.3. Ứng dụng của ML trong NLP 10](#_Toc198742315)

[1.4. Tổng quan về Học sâu 11](#_Toc198742316)

[1.4.1. Giới thiệu về Học sâu 11](#_Toc198742317)

[1.4.2. Ứng dụng của Học sâu trong NLP 11](#_Toc198742318)

[1.5. Tổng quan về Mạng neural nhân tạo (ANN) 12](#_Toc198742319)

[1.5.1. Giới thiệu về ANN 12](#_Toc198742320)

[1.5.2. Các thành phần cấu trúc trong ANN 12](#_Toc198742321)

[1.5.3. Hiệu quả và hạn chế của ANN trong các tác vụ NLP 19](#_Toc198742322)

[1.6. Tổng quan về Mạng neural hồi quy (RNN) 20](#_Toc198742323)

[1.6.1. Giới thiệu về RNN 20](#_Toc198742324)

[1.6.2. Cấu trúc của RNN 20](#_Toc198742325)

[1.6.3. Hiệu quả của RNN trong các tác vụ NLP 21](#_Toc198742326)

[1.7. Tổng quan về Bộ nhớ ngắn hạn dài (LSTM) 22](#_Toc198742327)

[1.7.1. Giới thiệu về LSTM 22](#_Toc198742328)

[1.7.2. Các thành phần cấu trúc của LSTM 22](#_Toc198742329)

[1.7.3. So sánh hiệu quả của LSTM với RNN trong các tác vụ NLP 23](#_Toc198742330)

[1.8. Tổng quan về Embedding văn bản 25](#_Toc198742331)

[1.8.1. Giới thiệu về Embedding văn bản 25](#_Toc198742332)

[1.8.2. Lợi ích của Embedding văn bản 25](#_Toc198742333)

[1.8.3. Các cách biểu diễn Word embedding 25](#_Toc198742334)

[1.9. Tổng quan về mô hình Transformer 26](#_Toc198742335)

[1.9.1. Giới thiệu về Transformer 26](#_Toc198742336)

[1.9.2. Kiến trúc mô hình Transformer 26](#_Toc198742337)

[1.9.3. Ứng dụng của Transformer trong NLP 33](#_Toc198742338)

[1.9.4. Các biến thế của Transformer 33](#_Toc198742339)

[1.10. Tổng quan về Mô hình ngôn ngữ lớn 34](#_Toc198742340)

[1.10.1. Giới thiệu về LLM 34](#_Toc198742341)

[1.10.2. Kiến trúc tổng quan của LLM 34](#_Toc198742342)

[1.10.3. Tổng quan về Tokenization 35](#_Toc198742343)

[1.10.4. Quy trình để huấn luyện một LLM 37](#_Toc198742344)

[1.10.5. Quy trình sinh văn bản của LLM (text generation) 38](#_Toc198742345)

[1.10.6. Streaming trong LLM 39](#_Toc198742346)

[1.10.7. Các kỹ thuật NLP trong LLM 40](#_Toc198742347)

[1.10.8. Ứng dụng LLM để xây dựng chatbot 48](#_Toc198742348)

[1.11. Các công cụ hỗ trợ 48](#_Toc198742349)

[1.11.1. Ngôn ngữ lập trình Python 48](#_Toc198742350)

[1.11.2. Các Framework sử dụng 49](#_Toc198742351)

[CHƯƠNG 2. PHƯƠNG PHÁP ĐỀ XUẤT XÂY DỰNG CHATBOT 51](#_Toc198742352)

[2.1. Mô tả và phân tích bài toán 51](#_Toc198742353)

[2.2. Phương pháp đề xuất 52](#_Toc198742354)

[2.3. Thiết kế hệ thống thông tin 54](#_Toc198742355)

[2.3.1. Chức năng hỏi đáp người dùng 55](#_Toc198742356)

[2.3.2. Biểu đồ use case chức năng hỏi đáp người dùng 56](#_Toc198742357)

[2.3.3. Biểu đồ tuần tự chức năng hỏi đáp người dùng 56](#_Toc198742358)

[2.3.4. Cơ sở dữ liệu người dùng 57](#_Toc198742359)

[2.4. Thu thập và tạo dữ liệu 58](#_Toc198742360)

[2.5. Tiền xử lý và phân đoạn dữ liệu 60](#_Toc198742361)

[2.6. Đánh giá các mô hình Embedding đề xuất 62](#_Toc198742362)

[2.7. Phương pháp đánh giá hiệu quả retrieval dữ liệu 66](#_Toc198742363)

[2.8. Các mô hình LLM được sử dụng 68](#_Toc198742364)

[2.9. Prompt engineering trong hệ thống chatbot 69](#_Toc198742365)

[2.10. Tác nhân (Agents) 71](#_Toc198742366)

[2.10.1. Tác nhân hỏi đáp (Q&A Agent) 71](#_Toc198742367)

[2.10.2. Tác nhân ngoài phạm vi (Out-of-Domain Agent) 72](#_Toc198742368)

[2.10.3. Quản lý tác nhân (Agent Management) 73](#_Toc198742369)

[2.11. Cá nhân hóa người dùng 73](#_Toc198742370)

[CHƯƠNG 3. TRIỂN KHAI HỆ THỐNG CHATBOT TUYỂN SINH 75](#_Toc198742371)

[3.1. Triển khai embedding dữ liệu và truy vấn người dùng 75](#_Toc198742372)

[3.2. Triển khai lưu trữ Vector embedding vào Vector database 75](#_Toc198742373)

[3.3. Retrieval cho truy vấn người dùng 77](#_Toc198742374)

[3.4. Triển khai agent cho hệ thống chatbot 78](#_Toc198742375)

[3.4.1. Xây dựng cấu trúc Prompt 78](#_Toc198742376)

[3.4.2. Triển khai prompt hỏi đáp 79](#_Toc198742377)

[3.4.3. Triển khai prompt điều hướng người dùng 80](#_Toc198742378)

[3.4.4. Trình quản lý agent và function calling 82](#_Toc198742379)

[3.5. Triển khai trình quản lý bộ nhớ chatbot 83](#_Toc198742380)

[3.6. Cơ sở dữ liệu người dùng 85](#_Toc198742381)

[3.7. Triển khai hệ thống chatbot hoàn chỉnh 85](#_Toc198742382)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 90](#_Toc198742383)

[CÁC TÀI LIỆU THAM KHẢO I](#_Toc198742384)

**DANH SÁCH CÁC BẢNG BIỂU**

[Bảng 3. Bảng mô tả cơ sở dữ liệu người dùng trong hệ thống chatbot. 52](#_Toc198676678)

[Bảng 4. Minh hoạ một số mẫu dữ liệu được thu thập và tạo. 54](#_Toc198676679)

[Bảng 5. So sánh và đánh giá hiệu suất của các mô hình embedding đề xuất. 59](#_Toc198676680)

[Bảng 6. Bảng so sánh hiệu suất các phương pháp retrieval. 63](#_Toc198676681)

# DANH SÁCH CÁC HÌNH ẢNH

[Hình 1: Sơ đồ cấu trúc của một ANN đơn giản 12](#_Toc198663788)

[Hình 2: Minh họa ma trận trọng số cho một lớp của ANN 14](#_Toc198663789)

[Hình 3: Cấu trúc của một mạng neural hồi quy (RNN) 20](#_Toc198663790)

[Hình 4: Cấu trúc của bước thứ t trong mô hình LSTM 22](#_Toc198663791)

[Hình 5: Mô tả cơ chế self-attention 26](#_Toc198663792)

[Hình 6: Minh họa cấu trúc của mỗi head trong multi-head attention 28](#_Toc198663793)

[Hình 7: Cách định hình lại ma trận Z 28](#_Toc198663794)

[Hình 8: Minh họa một kết nối dư thừa 29](#_Toc198663795)

[*Hình 9: Mạng nơ-ron thường (bên trái) và mạng nơ-ron có kết nối dư (bên phải)* 30](#_Toc198663796)

[*Hình 10: Mã hóa vị trí từ embedding* 31](#_Toc198663797)

[Hình 17. Minh họa xử lý đa tác vụ của một LLM. 33](#_Toc198663798)

[Hình 18. Quá trình huấn luyện một mô hình LLM. 33](#_Toc198663799)

[Hình 19. Mô hình luồng hoạt động của kỹ thuật RAG. 38](#_Toc198663800)

[Hình 20. Minh hoạt hệ thống tác nhân (Agent) tự động được hỗ trợ bởi LLM [86]. 40](#_Toc198663801)

[Hình 21. Mô họa việc các câu lệnh kiến thức được tạo ra [90]. 41](#_Toc198663802)

[Hình 22. Tổng quan hệ thống chatbot sử dụng LLM. 42](#_Toc198663803)

[Hình 23. Tổng quan bài toán xây dựng hệ thống chatbot hỏi đáp tuyển sinh. 45](#_Toc198663804)

[Hình 24. Phương pháp đề xuất xây dựng chatbot hỏi đáp tuyển sinh sử dụng LLM. 46](#_Toc198663805)

[Hình 25. Biểu đồ chức năng hỏi đáp người dùng. 49](#_Toc198663806)

[Hình 26. Biểu đồ use case chức năng hỏi đáp tuyển sinh. 50](#_Toc198663807)

[Hình 27. Biểu đồ tuần tự chức năng hỏi đáp người dùng. 51](#_Toc198663808)

[Hình 28. Tạo dữ liệu tuyển sinh bằng cách thu thập và tạo kịch bản. 52](#_Toc198663809)

[Hình 29. Quy trình tiền xử lý và phân đoạn bộ dữ liệu. 55](#_Toc198663810)

[Hình 30. Tổng quan quy trình đánh giá hiệu suất các mô hình embedding. 56](#_Toc198663811)

[Hình 31. Quy trình đánh giá hiệu suất retrieval dữ liệu. 61](#_Toc198663812)

[Hình 32. Cấu trúc prompt trong hệ thống chatbot hỏi đáp tuyển sinh. 64](#_Toc198663813)

[Hình 33. Mô tả luồng hoạt động của Q&A agent. 65](#_Toc198663814)

[Hình 34. Mô tả luồng hoạt động của OOD agent. 66](#_Toc198663815)

[Hình 35. Quy trình thu thập thông tin và cá nhân hóa người dùng. 68](#_Toc198663816)

[Hình 36. Quá trình triển khai embedding bộ dữ liệu sử dụng mô hình embedding. 69](#_Toc198663817)

[Hình 37. Quá trình lưu trữ vector embedding vào vector database. 70](#_Toc198663818)

[Hình 38. Các bước thực hiện retrieval cho truy vấn người dùng. 71](#_Toc198663819)

[Hình 39. Cách một prompt được hoàn chỉnh và sử dụng trong hệ thống chatbot. 74](#_Toc198663820)

[Hình 40. Prompt điều hướng người dùng trong hệ thống chatbot. 75](#_Toc198663821)

[Hình 41. Minh họa chức năng của agent management. 76](#_Toc198663822)

[Hình 42. Mô hình kiến trúc hệ thống MemGPT [118]. 77](#_Toc198663823)

[Hình 43. Dữ liệu người dùng được thu thập và sử dụng cho mục đích cá nhân hóa. 79](#_Toc198663824)

[Hình 44. Minh họa bắt đầu trò chuyện sau skhi triển khai chatbot hoàn chỉnh. 80](#_Toc198663825)

[Hình 45. Minh họa kết quả triển khai hệ thống hoàn chỉnh. 80](#_Toc198663826)

[Hình 46. . Minh họa kết quả triển khai hệ thống hoàn chỉnh.s 81](#_Toc198663827)

[Hình 47. Minh họa kết quả triển khai hệ thống hoàn chỉnh. 81](#_Toc198663828)

[Hình 48. Minh họa kết quả triển khai hệ thống hoàn chỉnh. 82](#_Toc198663829)

[Hình 49. Minh họa kết quả triển khai hệ thống hoàn chỉnh. 83](#_Toc198663830)

# DANH SÁCH CÁC KÝ HIỆU VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Chữ viết tắt** | **Diễn giải** | **Ý nghĩa** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| ANN | Artificial Neural Network | Mạng Nơ-ron nhân tạo |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |
| DL | Deep learning | Học sâu |
| FFNN | Feed Forward Neural Network | Mạng nơ-ron truyền thẳng |
| LLM | Large Language Model | Mô hình ngôn ngữ lớn |
| LSTM | Long Short-Term Memory | Bộ nhớ dài-ngắn hạn |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| ML | Machine learning | Học máy |
| SL | Self-supervised learning | Học tự giám sát |
| RAG | Retrieval-Augmented Generation | Tạo sinh tăng cường truy xuất |
| RNN | Recurrent Neural Network | Mạng nơ-ron hồi tiếp |

# MỞ ĐẦU

1. **Bối cảnh đề tài**

Trong thời gian gần đây, sự phát triển mạnh mẽ của AI đã có những tác động sâu sắc đến đời sống nói chung và lĩnh vực giáo dục nói riêng. Hơn nữa, sự ra đời của các LLM với khả năng xử lý ngôn ngữ tự nhiên nổi bật đã mở ra những cánh cửa mới cho các hệ thống có tính tự động hóa cao, một trong số đó là các hệ thống chatbot. Những công nghệ mới này có thể xử lý một khối lượng dữ liệu đồ sộ, học hỏi các cấu trúc dữ liệu phức tạp, và tự động hóa các cuộc đối thoại một cách rất tự nhiên, mang lại trải nghiệm thân thiện và năng suất hơn cho người dùng.

Tại Việt Nam, việc dạy và học bộ môn Lịch sử vẫn gặp nhiều khó khăn. Phương pháp truyền thống là nghe giảng, đọc tài liệu và ghi nhớ thì đòi hỏi rất nhiều thời gian, các chi phí về mặt nhân sự và triển khai; chưa kể đến việc phương pháp này đối với người học thì có phần khô khan vì mang tính lặp đi lặp lại, và cũng khó để có thể triển khai trên quy mô lớn một cách nhanh chóng. Trong bối cảnh này, việc áp dụng các công nghệ tự động hóa tiên tiến như chatbot vào công tác dạy và học trở thành một ý tưởng rất tiềm năng để tối ưu hóa công tác giáo dục nói chung và bộ môn Lịch sử Việt Nam nói riêng.

Với mong muốn được góp phần phát triển nền giáo dục nước nhà, em đã đề xuất ý tưởng về việc xây dựng một chatbot hỏi đáp hỗ trợ học Lịch sử Việt Nam dựa trên nền tảng là LLM. Chatbot này sẽ đóng vai trò là một công cụ hỗ trợ tự động, có thể trả lời các câu hỏi về các chủ đề khác nhau trong phạm trù Lịch sử Việt Nam từ phía người dùng, từ các triều đại, các nhận vật nổi bật cho đến những cuộc kháng chiến và đời sống nhân dân trong các thời kỳ lịch sử. Với khả năng xử lý ngôn ngữ tự nhiên đã được công nhận của LLM, chatbot sẽ mang lại một trải nghiệm học tập mới mẻ, tự nhiên và thân thiện, đồng thời đóng vai trò bổ trợ cho phương pháp giáo dục truyền thống, giúp nó trở nên hiện đại hơn và hiệu quả hơn.

Do đó, bối cảnh của đề tài này không chỉ là nhu cầu cấp thiết ứng dụng các cải tiến về khoa học - công nghệ vào giáo dục, mà còn là mong muốn tạo ra một bước tiến trong việc dạy và học bộ môn Lịch sử Việt Nam, góp phần vào hiện đại hóa giáo dục và đáp ứng tốt hơn thị hiếu của xã hội hiện đại.

1. **Lý do chọn đề tài**

Lý do chọn đề tài “Xây dựng hệ thống chatbot hỗ trợ học Lịch sử Việt Nam” xuất phát từ nhu cầu cấp thiết phải đổi mới, ứng dụng các thành tựu công nghệ tiên tiến để cải thiện công tác giáo dục. Ngày nay, sự phát triển nhanh chóng của AI và LLM đã khiến cho việc sử dụng chúng để tối ưu hóa các phương pháp giảng dạy hiện tại trở nên khả thi và có nhiều hiệu quả hơn bao giờ hết. Việc ứng dụng AI giúp cho giáo dục, một công việc vẫn còn mang nặng tính truyền thống ở Việt Nam, bắt kịp với xu hướng của thời đại; tiết kiệm được các nguồn tài nguyên và nhân lực, đồng thời nâng cao trải nghiệm của người dùng nhờ vào sự tự nhiên trong tương tác và khả năng cá nhân hóa cao.

Đặc biệt, phạm trù Lịch sử là phạm trù có tính đặc thù và có vai trò cực kỳ quan trọng trong giáo dục. Lịch sử vận động liên tục theo dòng chảy của thời gian, mỗi một sự kiện lịch sử tuy chỉ diễn ra một lần, nhưng lại có những liên kết đến nhiều đối tượng thuộc đa dạng các phạm trù cuộc sống. Ngoài ra, các sự kiện lịch sử có quan hệ nhân – quả chặt chẽ, nên lịch sử sẽ giúp người học tiếp thu những bài học từ quá khứ, từ đó mà hình thành nên lòng tự tôn dân tộc, ý thức hệ và định hướng hành động cho hiện tại và tương lại. Tuy nhiên, với phương pháp học tập và giảng dạy truyền thống chú trọng vào nghe giảng, đọc tài liệu và ghi nhớ, thì chuyện đáp ứng được nhu cầu thông tin từ một số lượng lớn người dùng một cách nhanh chóng và chính xác là một thách thức không nhỏ. Chính vì vậy, việc xây dựng một chatbot hỗ trợ học tập Lịch sử Việt Nam vừa giúp tiết kiệm những chi phí về tài nguyên và nhân lực, vừa đảm bảo được tính chính xác và tốc độ trong cung cấp các dữ liệu lịch sử.

Mặt khác, lý do chọn đề tài này còn đến từ những tiềm năng mà các LLM có thể mang lại. Với khả năng xử lý ngôn ngữ tự nhiên một cách linh hoạt và tự nhiên như con người, các LLM có thể giúp cho chatbot đối ứng được với các câu hỏi từ đơn giản đến phức tạp, với thông tin chính xác và liên tục cập nhật. Hơn nữa, ứng dụng LLM vào chatbot hỗ trợ học tập còn là một bước tiến lớn trong giáo dục phổ thông, góp phần đẩy mạnh hơn nữa sự phát triển của lĩnh vực này trong tương lai.

Tóm lại, đề tài này vừa đáp ứng được nhu cầu thực tiễn phải phát triển phương pháp giáo dục bộ môn Lịch sử Việt Nam, vừa là một cơ hội để nghiên cứu và áp dụng một trong những công nghệ tiên tiến nhất hiện nay – LLM, từ đó góp phần nâng cao trải nghiệm học tập và giúp hoạt động giảng dạy, học tập trở nên hiệu quả hơn.

1. **Đối tượng nghiên cứu**

Đối tượng nghiên cứu của đề tài này xoay quanh việc nghiên cứu các LLM để phục vụ cho việc xây dựng một chatbot hỗ trợ học Lịch sử Việt Nam, cụ thể là việc ứng dụng các LLM như Llama, Qwen, GPT hoặc các LLM có khả năng xử lý ngôn ngữ tự nhiên mạnh mẽ khác vào công việc hỏi đáp kiến thức lịch sử. Các đối tượng nghiên cứu thì bao gồm các yếu tố về kỹ thuật và thuật toán của LLM, phương pháp nạp dữ liệu, phương pháp để tối ưu hóa mô hình và khả năng tương tác với người dùng trong ngữ cảnh học tập lịch sử của mô hình.

Ngoài ra, nghiên cứu cũng sử dụng các phương pháp đánh giá về hiệu năng hoạt động của hệ thống chatbot, bao gồm các dữ liệu về độ chính xác của câu trả lời, khả năng hiểu ngữ cảnh, mức độ hài lòng của người dùng, và khả năng mở rộng của chatbot để đáp ứng nhu cầu người dùng thực tế.

1. **Phạm vi nghiên cứu**

Việc xây dựng một chatbot hỗ trợ học tập Lịch sử Việt Nam là một công việc không dễ dàng với nhiều những khó khăn và thách thức. Do đó, em quyết định phát triển đề tài này trong một phạm vi giới hạn, cụ thể gồm những phạm vi nghiên cứu sau đây:

Phạm vi ứng dụng: Chatbot được xây dựng với mục đích chính là hỗ trợ học tập Lịch sử Việt Nam, vì thế mà chatbot sẽ tập trung vào các thông tin trong phạm trù lịch sử Việt Nam như là các triều đại, các nhân vật nổi bật, các thời kỳ trong lịch sử, các cuộc kháng chiến và khởi nghĩa, các sự kiện trọng đại trong lịch sử của Việt Nam. Chatbot cần đảm bảo trả lời chính xác các câu hỏi phổ biến, đồng thời phải có khả năng cập nhật các thông tin mới.

Phạm vi công nghệ: Đề tài sẽ tập trung vào các LLM có năng lực xử lý ngôn ngữ tự nhiên tốt, giúp chatbot có thể hiểu và phản hồi các câu hỏi từ người dùng một cách tự nhiên và chính xác. Ngoài ra, các nền tảng và thư viện hỗ trợ xây dựng hệ thống chatbot như Chainlit hay các API cung cấp các dịch vụ ngôn ngữ có sẵn cũng sẽ được nghiên cứu và ứng dụng để tối ưu hóa hiệu suất của chatbot.

Phạm vi người dùng: Chatbot hướng đến nhóm người dùng bao gồm học sinh, sinh viên có nhu cầu tìm hiểu thông tin về Trường Đại học Đông Á, phụ huynh học sinh và các cá nhân quan tâm đến việc nhập học tại trường. Đối tượng người dùng có thể truy cập chatbot qua các kênh truyền thông chính thức của nhà trường, giúp đảm bảo tính thuận tiện và dễ dàng cho người dùng khi tìm kiếm thông tin tuyển sinh. Với các đối tượng và phạm vi nghiên cứu rõ ràng, đề tài không chỉ tập trung vào việc phát triển chatbot tuyển sinh mà còn mở ra cơ hội nâng cao trải nghiệm tư vấn giáo dục thông qua ứng dụng AI.

Phạm vi người dùng: Đề tài sẽ hướng đến nhóm người dùng chính là học sính, sinh viên và những người có nhu cầu tìm hiểu các thông tin về lịch sử Việt Nam. Các đối tượng người dùng có thể truy cập đến hệ thống chatbot thông qua một giao diện người dùng (UI), giúp việc tra cứu thông tin và tương tác với hệ thống chatbot trở nên dễ dàng và thuận tiện hơn cho người dùng.

Với các đối tượng và phạm vi nghiên cứu rõ ràng, đề tài không chỉ tập trung vào xây dựng và phát triển một hệ thống chatbot hỗ trợ học tập Lịch sử Việt Nam, mà còn mở ra cánh cửa cho người dùng để tiếp cận và trải nghiệm một phương pháp giáo dục có ứng dụng AI.

1. **Nội dung nghiên cứu**

Đề tài này sẽ tập trung vào các nội dung nghiên cứu sau đây:

Đầu tiên, đề tài sẽ tập trung vào tìm hiểu và đánh giá các LLM và các mô hình embedding, mục tiêu là sau khi phân tích được các ưu nhược điểm, mô hình phù hợp nhất với đề tài sẽ được lựa chọn để áp dụng vào hệ thống chatbot. Đồng thời, các kỹ thuật tinh chỉnh mô hình nhằm đáp ứng các yêu cầu đặc thù của đề tài cũng sẽ được nghiên cứu.

Tiếp theo, đề tài sẽ tập trung vào xây dựng, triển khai hệ thống chatbot thông qua một thiết kế kiến trúc tổng thể, gồm có mô hình ngôn ngữ. Một pipeline dữ liệu từ quá trình thu thập dữ liệu, tiền xử lý dữ liệu, nạp dữ liệu vào LLM, sử dụng các thư viện và tích hợp các API dịch vụ ngôn ngữ nhằm tối ưu hóa hiệu suất sẽ được xây dựng.

Kế đó, các tiêu chí đánh giá như độ chính xác, thời gian phản hồi và khả năng hiểu ngữ cảnh cũng sẽ được thiết lập để đảm bảo chất lượng của hệ thống chatbot. Sau đó, các kịch bản hỏi đáp thực tế sẽ được tiến hành thử nghiệm để đánh giá hiệu năng của chatbot, từ đó thực hiện các bước tối ưu hóa để đảm bảo tính ổn định và độ chính xác cho hệ thống chatbot.

Cuối cùng, hệ thống chatbot sẽ được triển khai lên UI để dễ dàng tiếp cận với người dùng. Sau khi triển khai, đề tài cũng sẽ đánh giá về khả năng mở rộng các tích năng nâng cao cho chatbot trong tương lai như tự động cập nhật dữ liệu lịch sử, đối thoại thời gian thực hay các dịch vụ hỗ trợ học tập khác.

Nội dung nghiên cứu như trên nhằm đảm bảo rằng hệ thống chatbot được xây dựng một cách bài bản và chỉn chu, đáp ứng tốt nhất nhu cầu của người dùng và góp phần nâng cao hiệu quả trong học tập và giảng dạy bộ môn Lịch sử Việt Nam.

1. **Cấu trúc bài báo cáo**

Báo cáo này được chia thành ba chương, tập trung vào việc xây dựng một hệ thống chatbot hỗ trợ học tập môn Lịch sử Việt Nam dựa trên nền tảng là LLM. Cấu trúc của báo cáo được thiết kế với mong muốn dẫn dắt người đọc từ cơ sở lý thuyết đến phương pháp triển khai và kết quả thực nghiệm, nhằm đảm bảo sự mạch lạc và tính ứng dụng của nghiên cứu.

Phần mở đầu giới thiệu bối cảnh và động lực thực hiện đề tài, nêu bật những thách thức trong việc đổi mới phương thức giáo dục môn Lịch sử Việt Nam và lý do lựa chọn LLM làm nền tảng. Phần này làm rõ mục tiêu nghiên cứu và kỳ vọng đóng góp của đề tài, qua đó tạo tiền đề để tiến đến những nội dung chi tiết.

Chương 1 cung cấp cơ sở lý thuyết, tập trung vào các công nghệ và phương pháp liên quan như Học sâu, Xử lý ngôn ngữ tự nhiên, mô hình Transformer và các đặc điểm nổi bật của LLM, các công cụ và kỹ thuật được sử dụng. Việc trình bày lý thuyết nhằm mục đích không chỉ giúp làm rõ cơ sở khoa học của nghiên cứu mà còn phác họa được sự phù hợp và tiềm năng ứng dụng của các công nghệ này trong bối cảnh xây dựng hệ thống chatbot hỗ trợ học tập.

Chương 2 tập trung vào phương pháp nghiên cứu và triển khai hệ thống. Nội dung của chương này bao gồm các bước như thu thập dữ liệu, xử lý dữ liệu, thiết kế và tích hợp các thành phần như embedding dữ liệu, thuật toán truy xuất thông tin, xây dựng prompt. Phương pháp được xây dựng nhằm tối ưu hóa hiệu quả của hệ thống chatbot trong việc cung cấp thông tin và tương tác với người dùng, đồng thời đảm bảo tính linh hoạt để mở rộng cho các bài toán tương tự.

Chương 3 trình bày quá trình triển khai hệ thống chatbot hỗ trợ học Lịch sử Việt Nam. Phần này sẽ mô tả việc chuyển đổi lý thuyết và phương pháp đề xuất thành sản phẩm thực tế, bao gồm xây dựng back-end xử lý dữ liệu và tích hợp các chức năng cốt lõi như hỏi đáp và điều hướng người dùng. Hiệu quả của chatbot sẽ được đánh giá thông qua các thử nghiệm thực tế để minh chứng cho khả năng hiện thực hóa của đề tài này.

Cuối cùng, phần kết luận và hướng phát triển sẽ tóm tắt những kết quả đạt được, đánh giá hiệu quả hệ thống và đề xuất những cải tiến nhằm nâng cao chất lượng và khả năng mở rộng của hệ thống chatbot trong tương lai. Báo cáo hướng đến mục tiêu không chỉ giải quyết vấn đề trong giáo dục môn Lịch sử Việt Nam mà còn góp phần ứng dụng các công nghệ tiên tiến trong việc nâng cao hiệu quả của các hệ thống hỗ trợ thông minh.

# CƠ SỞ LÝ THUYẾT

## Giới thiệu đề tài

### Tổng quan đề tài

Từ khi ra đời cho đến nay, Trí tuệ nhân tạo (Artificial Intelligence – AI) với tốc độ phát triển nhanh chóng đang ngày càng thể hiện vai trò quan trọng trong nhiều lĩnh vực của xã hội, bao gồm khoa học – công nghệ, công nghiệp và thậm chí là cuộc sống hàng ngày của chúng ta [1]. Đặc biệt, sự xuất hiện của các mô hình ngôn ngữ lớn (Large Language Model – LLM), với khả năng hiểu và xử lý ngôn ngữ tự nhiên đáng kinh ngạc, đã mở ra nhiều cơ hội để tự động hóa các công việc phức tạp, điển hình như các hệ thống hội thoại tự động, hay còn được gọi là chatbot [2].

Trong bối cảnh ấy, việc sử dụng AI vào giáo dục, cụ thể là học tập môn Lịch sử Việt Nam, cho thấy khả năng tối ưu hóa quá trình tra cứu và ôn tập thông tin cho học viên, giảm tải khối lượng công việc cho giáo viên và nâng cao trải nghiệm của người dùng. Vì thế, đề tài “Xây dựng hệ thống chatbot hỗ trợ học Lịch sử Việt Nam” dựa trên LLM đã ra đời với mục đích là xây dựng được một hệ thống chatbot có khả năng trả lời nhanh chóng và chính xác các câu hỏi phổ biến liên quan đến lịch sử Việt Nam.

Hệ thống này không những góp phần vào tự động hóa công tác trả lời những câu hỏi thường gặp về các nhân vật, các đối tượng và các sự kiện trong dòng chảy của Lịch sử Việt Nam, mà những thông tin mới và chính xác có thể được cập nhật liên tục [3]. Điều này rất có ý nghĩa, khi mà phương thức giáo dục truyền thống đang tỏ rõ những yếu điểm về khả năng cập nhật thông tin, khả năng mở rộng và độ hiệu quả trong tiếp thu kiến thức của người học [4]; chưa kể đến việc những thông tin sai sự thật đang tràn lan với tốc độ chóng mặt vì sự phát triển của mạng xã hội. Việc ứng dụng LLM vào hệ thống chatbot sẽ làm cho hệ thống ấy trở nên thông minh, có khả năng hiểu, xử lý và phản hồi các loại câu hỏi, yêu cầu đa dạng từ người dùng, từ đó đáp ứng tốt những yêu cầu đối với giáo dục bộ môn Lịch sử [5].

### Nội dung nghiên cứu

Nội dung nghiên cứu của đề tài này bao gồm việc tìm hiểu và đánh giá các LLM và các mô hình embedding để phân tích các ưu, nhược điểm của từng mô hình để chọn ra các mô hình tối ưu với đề tài. Đề tài cũng tập trung vào thiết kế kiến trúc của hệ thống chatbot và xây dựng pipeline dữ liệu để phục vụ cho quá trình nạp dữ liệu và thử nghiệm hệ thống.

Ngoài ra, đề tài còn nghiên cứu việc đánh giá hiệu suất của chatbot thông qua các tiêu chí như độ chính xác và thời gian phản hồi. Cùng với đó, đề tài sẽ thử nghiệm hệ thống chatbot qua các kịch bản hỏi đáp thực tế, sau đó tối ưu hóa hệ thống để mong muốn đáp ứng những nhu cầu từ người dùng một cách tốt nhất.

Cuối cùng, đề tài sẽ trình bày việc triển khai hệ thống chatbot lên một giao diện người dùng và xây dựng các tài liệu hướng dẫn để đảm bảo rằng hệ thống sẽ được vận hành một cách hiệu quả và có thể mở rộng các tính năng mới một cách dễ dàng trong tương lai.

## Lược sử về Xử lý ngôn ngữ tự nhiên

Ngôn ngữ vốn là sản phẩm của sự tổng hòa của việc truyền tải những suy nghĩ, thông tin, ý tưởng cùng với cảm xúc, tính ẩn dụ và tính không hoàn hảo. Rất khó để xem ngôn ngữ là kết quả của các công thức toán học phức tạp mà hoạt động một cách đồng thời, vì vậy mà việc tạo ra các quy tắc logic để máy tính có thể hiểu và xử lý được ngôn ngữ tự nhiên cũng là không hề dễ dàng [6].

Sự phát triển của NLP bắt đầu với các phương pháp xử lý dựa trên các quy luật và tri thức ngôn ngữ được con người định nghĩa. Năm 1950, Alan Turing đã kiểm tra xem máy tính có thể suy nghĩ về ngôn ngữ giống con người hay không. Năm 1957, Chomsky công bố syntactic structure (cấu trúc cú pháp), đánh dấu bước đầu tiên của NLP trong lĩnh vực dịch máy. Năm 1969, Rogere Schank giới thiệu conceptual dependency theory (lý thuyết về sự phụ thuộc khái niệm) trong việc hiểu ngôn ngữ tự nhiên và được sử dụng rộng rãi. Đầu thập niên 1970, ATN (Augmented Transition Network – Mạng chuyển tiếp tăng cường) ra đời để biểu diễn đầu vào của ngôn ngữ tự nhiên, trở thành nền tảng cho các hệ thống hiểu ngôn ngữ tự nhiên cùng thời. Đến những năm 1980, những bước nền tảng và cốt lõi như hình thái hóa (morphology), ngữ nghĩa hóa (semantic), tham chiếu (reference) được tập trung nghiên cứu [6][7].

Sự ra đời của các thuật toán Học máy (ML) cho NLP vào cuối thập niên 1980 đã đánh dấu một bước tiến lớn của NLP, mở ra thời kỳ của NLP bằng phương thức thống kê [6]. Rất nhiều các thành công nổi bật đã đến trong lĩnh vực dịch máy [7]. Các phương thức sử dụng dữ liệu hiệu quả tiếp tục là lĩnh vực được nghiên cứu và phát triển, cùng với đó là sự xuất hiện của các thuật toán huấn luyện không giám sát, có giám sát, bán giám sát và một số phương pháp khác.[7].

Từ năm 2015, phương thức thống kê dần được thay thế bởi phương pháp tiếp cận bằng mạng neural, sử dụng semantic network (mạng ngữ nghĩa) và word embedding để nắm bắt các đặc điểm ngữ nghĩa của từ. Dịch máy bằng mạng neural, dựa trên các phương thức biến đổi sequence-to-sequence, dần thay thế các phương thức dịch máy bằng thống kê [7].

## Tổng quan về Học máy

### Giới thiệu về Học máy

Học máy hay máy học (Machine Learning – ML) là một lĩnh vực của AI liên quan đến việc nghiên cứu và xây dựng các kỹ thuật cho phép các hệ thống “học” tự động từ dữ liệu để giải quyết những vấn đề cụ thể [8]. Các thuật toán ML sẽ xây dựng một mô hình, học tập từ dữ liệu mẫu, còn gọi là dữ liệu huấn luyện, để đưa ra dự đoán cho dữ liệu nạp vào mà không cần được lập trình chi tiết về quy tắc, cách thức đưa ra dự đoán [9].

ML có liên quan lớn đến thống kê vì cả hai đều nghiên cứu việc phân tích dữ liệu, nhưng khác với thống kê, ML tập trung vào sự phức tạp của thuật toán trong việc thực thi các tính toán [10]. Ngoài ra, ML cũng có một phần là nghiên cứu về sự phát triển của các giải thuật suy luận xấp xỉ mà có thể tính toán được [10].

### Phân loại Học máy

Các loại thuật toán ML được phân loại dựa theo kết quả mong muốn của thuật toán. Các loại thuật toán ML thường dùng bao gồm:

#### Học có giám sát

Học có giám sát (Supervised Learning) là công nghệ phổ biến nhất trong các bài toán phân loại, khi mà mục tiêu của thuật toán này thường là khiến cho máy tính học được một hệ thống phân loại mà chúng ta đã tạo ra [11].

Quá trình học trong một mô hình ML đơn giản được chia thành hai giai đoạn: huấn luyện (training) và kiểm thử (testing). Trong giai đoạn training, các mẫu dữ liệu trong dữ liệu huấn luyện được xem như dữ liệu nạp vào của mô hình. Trong giai đoạn testing, mô hình đang huấn luyện sẽ sử dụng bộ máy xử lý của nó để tạo ra dự đoán cho dữ liệu kiểm thử hoặc dữ liệu thực tế. Dữ liệu đã gắn nhãn (tagged data), đầu ra của mô hình, là cái quyết định dự đoán cuối cùng hoặc dữ liệu được phân loại như thế nào [11].

Về mặt toán học, mục tiêu của ML là tìm ra một hàm:

|  |  |
| --- | --- |
|  | (1.1) |

với X là không gian đầu vào hay không gian đặc trưng, Y là không gian đầu ra hay không gian của các nhãn [12]. Thuật toán ML sử dụng các mẫu dữ liệu trong không gian dữ liệu huấn luyện

|  |  |
| --- | --- |
|  | (1.2) |

với là vector đặc trưng, là nhãn của [12]. Để tính toán mức độ học tập của thuật toán đối với dữ liệu huấn luyện, một hàm mất mát

|  |  |
| --- | --- |
|  | (1.3) |

được định nghĩa. Đối với mỗi cặp thì sự mất mát của giá trị dự đoán là [12].

#### Học không giám sát

Học không giám sát (Unsupervised Learning) là một nhánh của ML mà trong đó, các thuật toán sẽ tự khám phá cấu trúc từ dữ liệu không có nhãn. Học không giám sát tập trung chủ yếu vào nhận dạng và phân nhóm các mẫu trong dữ liệu mà không cần biết trước bất kỳ nhãn hay quy tắc nào [13].

Mục tiêu của học không giám sát là tìm cách tối ưu hóa một hàm mục tiêu , sao cho mô hình có thể biểu diễn được cấu trúc ẩn trong dữ liệu huấn luyện không có nhãn. Quá trình này thường liên quan đến việc tối ưu một hàm mất mát để khám phá các đặc trưng quan trọng hoặc cấu trúc của dữ liệu [13].

Từ dữ liệu huấn luyện , trong đó mỗi là một điểm dữ liệu trong không gian có số chiều là , ta sẽ xây dựng một hàm mục tiêu

|  |  |
| --- | --- |
|  | (1.4) |

với để biến đổi các dữ liệu của , sao cho có thể biểu diễn được các đặc trưng quan trọng của dữ liệu [14]. Hàm mục tiêu này có thể được chọn sao cho phản ánh được tốt nhất các mối quan hệ, các mẫu hoặc cấu trúc ẩn trong dữ liệu. Để tìm được các biểu diễn tối ưu, ta sẽ tối thiểu hóa một hàm mất mát tổng quát [14]

|  |  |
| --- | --- |
|  | (1.5) |

Về cơ bản, cùng với học có giám sát, học không giám sát là những thuật toán ML phổ biến nhất, được sử dụng trong nhiều tác vụ từ đơn giản đến phức tạp và đã chứng minh được hiệu quả của chúng [11].

#### Học bán giám sát

Học bán giám sát (Semi-supervised learning) là một lớp khác trong các thuật toán ML khi sử dụng cả dữ liệu đã gán nhãn và dữ liệu không gán nhãn, với tỉ trọng dữ liệu huấn luyện điển hình là một lượng nhỏ dữ liệu đã gán nhãn và một lượng lớn dữ liệu không gán nhãn [15], [16]. Có thể xem học bán giám sát đứng giữa học có giám sát và học không giám sát [15].

Nhiều nhà nghiên cứu nhận thấy một sự cải thiện đáng kể về độ chính xác khi sử dụng dữ liệu không gán nhãn với một lượng nhỏ dữ liệu đã gán nhãn. Điều này có giá trị thực tiễn rất lớn, vì trên thực tế, chi phí của quy trình gán nhãn cho dữ liệu khiến chuyện tập dữ liệu được gán nhãn hoàn toàn thì trở nên bất khả thi, trong khi dữ liệu không gán nhãn thường tương đối rẻ tiền [15], [16].

Học bán giám sát sẽ xử lý một tập có các mẫu dữ liệu phân phối đồng nhất và độc lập với các nhãn tương ứng và mẫu dữ liệu không có nhãn [15]. Các phương pháp phổ biến được dùng để tận dụng dữ liệu không nhãn gồm có:

Mô hình sinh (Generative Methods): với ý tưởng chính là giả định dữ liệu của mỗi lớp nhãn của dữ liệu được sinh ra từ một phân phối xác suất cụ thể, phương pháp này sẽ sử dụng định lý Bayes để suy ra , từ đó gán nhãn cho dữ liệu không có nhãn [17].

Phân tách vùng mật độ thấp (Low-density Separation): phương pháp này giả định rằng ranh giới phân loại nên nằm ở vùng dữ liệu có mật độ thấp, tức là các mẫu dữ liệu ở gần nhau thường có cùng nhãn, và các lớp nhãn nên được tách biệt bởi “vùng trống” trong không gian dữ liệu [18].

Điều chuẩn theo đồ thị (Laplacian Regularization): ý tưởng chính của phương thức này là mô hình hóa dữ liệu như một đồ thị, trong đó các mẫu dữ liệu là đỉnh, còn mức độ tương đồng giữa chúng là cạnh, rồi sử dụng công thức của hàm mất mát một thành phần

|  |  |
| --- | --- |
|  | (1.6) |

với là trọng số của cạnh nối và , là đầu ra dự đoán của mô hình tại điểm [19].

#### Học tăng cường

Học tăng cường (Reinforcement learning) là một lĩnh vực con của ML nhằm nghiên cứu cách thức một agent (thực thể có khả năng tương tác và ra quyết định) trong một môi trường thì nên chọn thực hiện các hành động nào để cực đại hóa một phần thưởng nào đó về lâu dài. Các thuật toán học tăng cường cố gắng tìm kiếm một chiến lược ánh xạ các trạng thái của môi trường thành các hành động mà agent nên chọn trong các trạng thái đó [20].

Xét một cách hình thức, mô hình của thuật toán học tăng cường gồm có [21]

* : tập hợp các trạng thái của môi trường
* : tập hợp các hành động có thể thực hiện
* : tập hợp các con số đại diện cho phần thưởng mà agent nhận được sau mỗi hành động.

Tại mỗi thời điểm , agent thấy được trạng thái và các hành động có thể thực hiện . Nó chọn một hành động và nhận được một trạng thái mới và phần thưởng . Dựa trên các tương tác này, agent phải phát triển một chiến lược để cực đại hóa tổng phần thưởng [21]. Cách tiếp cận này dẫn đến hai bước sau đây [21]:

* Với mỗi hành động có thể, lấy mẫu kết quả thu được khi thực hiện hành động đó
* Chọn chiến lược có kết quả trả về lớn nhất

Học tăng cường được ứng dụng vào nhiều lĩnh vực khác nhau như AI trong game, robot học kỹ năng, xe tự hành, tối ưu hóa hệ thống mạng, tài chính – đầu tư,… [22]

### Ứng dụng của ML trong NLP

ML đã góp phần quan trọng vào sự phát triển của các ứng dụng NLP. Với khả năng học tự động và có phương thức thống kê hiệu quả hơn, ML đã giúp tự động hóa rất nhiều các tác vụ phức tạp về phân loại văn bản, phân tích cảm xúc, gán nhãn thực thể và trích xuất thông tin [23]; đồng thời cải thiện độ chính xác của nhiều hệ thống NLP so với các phương pháp dựa trên thống kê đơn giản của thế hệ trước [24]. Một trong những thuật toán ML nổi tiếng là Naïve Bayes đã được sử dụng cho bài toán phân loại email thành thư rác và không phải thư rác, giúp đơn giản hóa mô hình, đem lại tốc độ tính toán nhanh chóng mà vẫn đảm bảo hiệu quả cao [25].

Tuy nhiên, dù các mô hình ML truyền thống đã đạt được những kết quả khá tốt, chúng vẫn phụ thuộc vào việc phải trích chọn đặc trưng một cách thủ công, thứ đã hạn chế khả năng mở rộng đi không ít. Mặt khác, việc trích chọn đặc trưng thủ công yêu cầu trình độ chuyên môn cao, do đó có ảnh hưởng lớn đến độ chính xác của những mô hình này [26]. Những hạn chế này đã thúc đẩy sự phát triển của các phương pháp học sâu, nơi mà các công việc thủ công bị loại bỏ hoàn toàn [27].

## Tổng quan về Học sâu

Trong nhiều năm, NLP dựa vào ML truyền thống như CRF, SVM hay HMM, tuy có đóng góp đáng kể nhưng vẫn chưa đủ để tạo ra những bước ngoặt lớn. Phải đến khi Học sâu (Deep Learning – DL) ra đời với khả năng tự động học các đặc trưng và khả năng xử lý ngữ cảnh hiệu quả hơn, NLP mới thực sự bắt đầu bước vào kỷ nguyên phát triển mạnh mẽ [28].

### Giới thiệu về Học sâu

Học sâu (Deep Learning) là một nhánh con của ML, tập trung vào việc sử dụng các mạng neural nhiều lớp (multi-layered neural networks) để thực hiện các công việc như phân loại (classification), hồi quy (regression) và học biểu diễn (representation learning) [29]. Lấy cảm hứng từ bộ môn thần kinh học, DL mô phỏng việc xếp chồng các neural nhân tạo thành các lớp, sau đó huấn luyện chúng để xử lý dữ liệu; đây cũng là lý do hình thành nên cái tên hiện tại của lĩnh vực này [29].

Một vài những kiến trúc mạng DL phổ biến ngày nay gồm có mạng neural hồi tiếp (RNNs), mạng neural tích chập (CNNs), mạng đối kháng sinh (GANs) và Transformer. Các kiến trúc này đã được ứng dụng vào nhiều lĩnh vực như thị giác máy tính, nhận dạng giọng nói, NLP,… với kết quả thu được tương đương hoặc trong một số trường hợp còn vượt qua con người [30].

### Ứng dụng của Học sâu trong NLP

Một trong những mô hình DL đầu tiên được sử dụng trong lĩnh vực NLP là mạng neural truyền thẳng (FNN). Chúng được phát triển khả năng học biểu diễn liên tục của word embedding và khả năng khai thác các mối quan hệ giữa các từ trong một không gian ngữ nghĩa, thay vì chỉ xem các từ như những thực thể rời rạc như những phương pháp truyền thống [31].

Sự xuất hiện của các mô hình học sâu, mà đặc biệt là mạng neural nhân tạo (ANN), với khả năng trích chọn đặc trưng tự động và có thể học được những mối liên hệ phức tạp giữa các từ trong cùng ngữ cảnh, đã làm thay đổi căn bản cách để xử lý ngôn ngữ [28]. Giờ đây, các mô hình DL đã có thể mô hình hóa các mối quan hệ phức tạp giữa từ với từ, điều mà các mô hình truyền thống khó mà làm được [31]. Từ đó, các mô hình DL tiên tiến hơn, với xuất phát là ANN, đã xuất hiện và được sử dụng rộng rãi trong các tác vụ phân loại văn bản, sinh văn bản và dịch máy [32].

## Tổng quan về Mạng neural nhân tạo (ANN)

### Giới thiệu về ANN

Mạng neural nhân tạo (Artificial Neural Network – ANN) hay mạng neural (NN) là một mô hình tính toán được lấy cảm hứng từ cấu trúc và chức năng của mạng neural sinh học [33]. Mô hình này bao gồm các neural nhân tạo kết nối với nhau một cách tương đối đơn giản và lỏng lẻo, nhằm mô phỏng lại kiến trúc của bộ não sinh học. Các mô hình ANN mô phỏng các neural sinh học một cách chi tiết hơn đang được nghiên cứu gần đây và đã cho thấy sự cải thiện đáng kể về mặt hiệu suất [34].

### Các thành phần cấu trúc trong ANN

Cấu trúc của một mô hình ANN đơn giản được minh họa ở hình 1 dưới đây:

A diagram of a complex structure

Description automatically generated

Hình 1: Sơ đồ cấu trúc của một ANN đơn giản

Cấu trúc của mạng ANN bao gồm những thành phần sau [35]:

* Lớp đầu vào có neural, mỗi neural tương ứng với một đặc trưng trong dữ liệu đầu vào.
* Tầng ẩn có lớp, mỗi lớp có số lượng neural khác nhau, mỗi neural trong lớp ẩn có một trọng số () và hàm kích hoạt được kết nối với các neural ở lớp trước đó . Mỗi neural trong lớp ẩn được cộng thêm một tham số bias (sai số) .
* Lớp đầu ra có neural, với mỗi neural () đại diện cho một đầu ra.
* Các neural trong mạng được liên kết với nhau thông qua các trọng số biểu thị sự quan trọng của một đặc trưng cho liên kết đó và thường được biểu diễn dưới dạng ma trận.

#### Lớp đầu vào

Giả sử ta có tập dữ liệu chứa mẫu dữ liệu. Mỗi mẫu dữ liệu lại có đặc trưng (hay chiều dữ liệu), và từng đặc trưng của được biểu diễn dưới dạng một phần từ của vector . Mỗi neural của input layer được đại diện bởi một nút tròn trong Hình 1, và tương ứng với một phần tử của vector của mẫu dữ liệu . Tại input layer, không có bất kỳ một tính toán nào được thực hiện [36].

#### Kết nối

Các kết nối giữa các neural trong một ANN là rất quan trọng đối với quá trình học. Mỗi neural trong một lớp lại được kết nối với mọi neural trong lớp liền kề. Mỗi kết nối đều có một giá trị trọng số để thể hiện mức độ quan trọng của kết nối ấy. Bằng cách lặp đi lặp lại tập dữ liệu huấn luyện trên mô hình, ta có thể đào tạo và tối ưu hóa các giá trị trọng số này, nhằm giúp cho mạng đạt được hiệu suất tổng thể mà ta mong muốn [37].

#### Độ lệch (bias)

Tham số bias của một ANN giúp điều chỉnh đầu ra của các neural, cho phép mạng có khả năng mô hình học được các mối quan hệ phi tuyến tính, đồng thời giúp đẩy nhanh quá trình tối ưu hóa mô hình. Trong một ANN, bias của từng lớp được tính riêng biệt, mỗi lớp neural sẽ có một vector bias riêng, và kích thước của nó phụ thuộc vào số lượng neural của lớp đó [38]. Với một ANN có hidden layer, lớp thứ có neural, thì bias của lớp thứ này có dạng

|  |  |
| --- | --- |
|  | (1.7) |

với là tham số bias của neural thứ [38].

#### Hàm kích hoạt

Hàm kích hoạt (activation function) đóng vai trò quan trọng trong việc đào tạo và hiệu suất của ANN. Chúng cung cấp các thuộc tính phi tuyến tính cần thiết cho bất kỳ ANN nào [39]. Activation function của một neural trong [ANN](https://en.wikipedia.org/wiki/Artificial_neural_network) sẽ tính toán đầu ra của neural dựa trên các đầu vào riêng lẻ và trọng số của chúng [39].

Sau mỗi phép biến đổi tuyến tính trong từng neural

|  |  |
| --- | --- |
|  | (1.8) |

Thì các hàm kích hoạt sẽ biến đầu ra tuyến tính thành một giá trị phi tuyến tính :

|  |  |
| --- | --- |
|  | (1.9) |

Nhờ đó, toàn bộ ANN sẽ có khả năng học và biểu diễn các quan hệ phi tuyến tính.

Một hàm kích hoạt bão hòa nếu

|  |  |
| --- | --- |
|  | (1.10) |

Và là không bão hòa nếu

|  |  |
| --- | --- |
|  | (1.11) |

Có nhiều hàm kích hoạt khác nhau, thường thì các hàm kích hoạt không bão hòa sẽ tốt hơn các hàm kích hoạt bão hòa vì chúng ít có khả năng gặp phải vấn đề biến mất độ dốc (gradient vanishing) [40], [41].

#### Trọng số

Trọng số (weight) là tham số thể hiện mức độ quan trọng của kết nối. Giá trị của trọng số của một kết nối càng cao thì neural ở phía đầu vào đóng vai trò càng lớn .

Giả sử ta đang xem xét lớp bất kỳ trong tầng ẩn của mô hình và có neural, cùng với lớp có neural, thì ma trận trọng số của lớp có thể được minh họa bằng hình dưới đây

A diagram of a number of equations

Description automatically generated with medium confidence

Hình 2: Minh họa ma trận trọng số cho một lớp của ANN

Trọng số đóng vai trò “học” của ANN, những tham số này có thể được điều chỉnh qua quá trình huấn luyện để tối ưu hóa đầu ra và hiệu suất học của mô hình. Thông qua lan truyền ngược (backpropagation) và các thuật toán tối ưu hóa như gradient descent, các trọng số này sẽ được cập nhật để giảm thiểu sai số dự đoán trên tập huấn luyện [42].

#### Lớp ẩn

Các lớp ẩn (hidden layers) trong một ANN đóng vai trò rất quan trọng trong việc xử lý và trích xuất các đặc trưng của dữ liệu đầu vào. Dữ liệu sau khi được nhận vào từ input layer, nó sẽ được truyền qua các hidden layers, nơi mà các phép toán phi tuyến tính sẽ được các neural thực thi để chuyển đổi dữ liệu [43].

Giả sử ta có một mạng ANN có lớp ẩn, lớp thứ có nơ-ron ( là chỉ số của lớp, ). Ta định nghĩa:

* Ma trận trọng số giữa lớp và lớp là có kích thước .
* Mỗi phần tử của ma trận trọng số biểu diễn trọng số từ nơ-ron thứ của lớp đến nơ-ron thứ của lớp .
* Quy ước lớp ẩn là lớp đầu vào.

Quá trình tính toán trong các hidden layer diễn ra như sau:

* Tính toán giữa input layer và hidden layer đầu tiên: Mỗi neural của hidden layer đầu tiên sẽ nhận dữ liệu từ D neural của input layer. Tổng các trọng số đầu vào của neural thứ k trong hidden layer đầu tiên có thể được biểu diễn dưới dạng [44]

|  |  |
| --- | --- |
|  | (1.12) |

với:

* : giá trị của neural thứ từ input layer
* : trọng số của kết nối giữa neural thứ từ input layer và neural thứ tại hidden layer đầu tiên
* : tham số bias của neural thứ tại hidden layer đầu tiên
* : tổng các trong số đầu vào của neural thứ tại hidden layer đầu tiên

rồi sau đó đi qua hàm kích hoạt để trở thành đầu ra của neural thứ của hidden layer đầu tiên

|  |  |
| --- | --- |
|  | (1.13) |

* Tính toán giữa hai hidden layer: Tương tự như trên, mỗi neural của hidden layer thứ sẽ nhận dữ liệu từ neural của hidden layer thứ . Tổng các trọng số đầu vào của neural thứ trong hidden layer thứ có thể được biểu diễn dưới dạng [44]

|  |  |
| --- | --- |
|  | (1.14) |

với:

* : tổng trọng số của neural thứ tại hidden layer thứ
* : trọng số của kết nối giữa neural thứ tại hidden layer thứ và neural thứ tại hidden layer thứ
* : tham số bias của neural thứ tại hidden layer thứ
* : tổng các trong số đầu vào của neural thứ tại hidden layer thứ

sau đó đi qua hàm kích hoạt để trở thành đầu ra của neural thứ của hidden layer thứ

|  |  |
| --- | --- |
|  | (1.15) |

#### Lớp đầu ra

Lớp đầu ra (output layer) là lớp cuối cùng trong ANN, chịu trách nhiệm trả về kết quả của mô hình. Số lượng neural trong lớp đầu ra thì phụ thuộc vào bài toán cụ thể. Đối với các bài toán hồi quy, số lượng neural của output layer là 1 nhằm biểu diễn giá trị liên tục. Trong khi với các bài toán phân loại, output layer thường có từ 2 neural trở lên, mỗi đầu ra tương ứng với một nhãn của tập dữ liệu [45].

Đầu vào của neural thứ () trong output layer là tổng có trọng số của các giá trị đầu ra từ hidden layer cuối cùng

|  |  |
| --- | --- |
|  | (1.16) |

trong đó:

* : tổng trọng số của neural thứ tại hidden layer cuối cùng
* : trọng số của kết nối giữa neural thứ tại hidden layer cuối cùng và neural thứ tại output layer
* : tham số bias của neural thứ tại output layer
* : tổng các trong số đầu vào của neural thứ tại output layer

Cuối cùng, sau khi đi qua một hàm kích hoạt nữa, ta được kết quả đầu ra là giá trị dự đó cho neural thứ của output layer như sau

|  |  |
| --- | --- |
|  | (1.17) |

#### Hàm mất mát

Hàm mất mát (loss function) đo lường mức độ sai lệch giữa giá trị dự đoán của mạng () và giá trị thực tế (). Mục tiêu của mạng là tối thiểu hóa hàm mất mát thông qua quá trình huấn luyện, bằng cách điều chỉnh các trọng số và tham số bias trong mạng.

Xin được nhắc lại:

* + Tập dữ liệu huấn luyện gồm mẫu dữ liệu với .
  + : Nhãn thực tế của mẫu .
  + : Giá trị dự đoán của ANN với đầu vào .

Hàm mất mát cơ bản của mô hình có dạng [38]:

|  |  |
| --- | --- |
|  | (1.18) |

trong đó, là hàm mất mát cho từng mẫu dữ liệu. Tùy vào bài toán cần giải quyết mà *ℓ* sẽ có một dạng cụ thể [38].

#### Hàm tối ưu

Hàm tối ưu (optimize function) trong ANN là một mục tiêu cần tối thiểu hóa hoặc tối đa hóa, và thường có liên quan trực tiếp đến hàm mất mát. Hàm này sẽ hướng dẫn quá trình học của mô hình, bằng cách cập nhật trọng số để giảm giá trị mất mát [38].

Biểu diễn tổng quát:

|  |  |
| --- | --- |
|  | (1.19) |

trong đó:

* : Tập trọng số của mạng.
* : Tập tham số bias.
* : Hàm mất mát cho mẫu dữ liệu .

#### Lan truyền xuôi

Lan truyền xuôi (forward propagation) là một quá trình diễn ra trong một ANN. Các dữ liệu đầu vào được truyền qua các lớp của mạng để tạo ra đầu ra. Quá trình này bao gồm các bước sau [38]:

* Tại input layer: Dữ liệu đầu vào được đưa vào input layer của mạng.
* Tại tầng ẩn: Dữ liệu đầu vào sẽ được xử lý thông qua một hoặc nhiều hidden layer có trong tầng ẩn. Mỗi neural trong mỗi hidden layer đều sẽ nhận đầu vào từ lớp trước, áp dụng hàm kích hoạt cho tổng trọng số của các đầu vào này và chuyển kết quả sang lớp tiếp theo.
* Tại output layer: Dữ liệu đã xử lý sẽ di chuyển qua output layer và tạo ra đầu ra cuối cùng của mạng. Thường thì output layer sẽ áp dụng một hàm kích hoạt lên dữ liệu trước khi đưa nó sang đầu ra. Giá trị thu được tại đầu ra của mạng, hoặc là kết quả dự đoán, hoặc là phân loại cho dữ liệu đầu vào.

Thuật ngữ lan truyền thuận được dùng như một cách chung để miêu tả quá trình tính toán trọng số đầu ra cho mạng. Các bước tính toán cho mỗi lớp trong quá trình lan truyền thuận đã được miêu tả chi tiết ở trên [38].

#### Lan truyền ngược

Lan truyền ngược (backward propagation) là quá trình tính toán gradient của hàm mất mát đối với các trọng số và tham số bias trong mạng nhằm giảm sai số [38]. Dựa vào các gradient này, các trọng số được cập nhật thông qua thuật toán tối ưu. Trong quá trình huấn luyện, sau khi thực hiện lan truyền thuận để tính đầu ra của mạng, các đầu ra được dự đoán này sẽ được so sánh với các giá trị thực tế (nhãn) của mẫu dữ liệu để tính toán mất mát. Sau đó, từ lớp đầu ra này tính gradient của hàm mất mát đối với từng trọng số và tham số bias thông qua đạo hàm theo chuỗi (chain rule) [38]. Quá trình này diễn ra từ lớp cuối ngược về lớp đầu tiên, cụ thể như sau:

* Tại lớp đầu ra: Gradient của hàm mất mát đối với đầu vào của lớp đầu ra được tính như sau:

|  |  |
| --- | --- |
|  | (1.20) |

trong đó:

* + ​: Gradient của hàm mất mát với đầu vào ​ tại output layer.
  + ​: Đầu ra từ hàm kích hoạt tại output layer.
  + ​: Nhãn thực tế.
* Lan truyền về lớp thứ : Gradient của hàm mất mát đối với đầu vào :

|  |  |
| --- | --- |
|  | (1.21) |

trong đó:

* + ​: Gradient của hàm mất mát đối với đầu vào ​ của neural thứ tại lớp thứ .
  + ​: Trọng số kết nối từ neural thứ tại lớp thứ đến neural thứ tại lớp thứ .
  + ​: Gradient tại neural thứ ở lớp thứ .
  + : Đạo hàm của hàm kích hoạt tại ​.
* Gradient của trọng số và bias:
  + Gradient của trọng số : Gradient của hàm mất mát đối với ma trận trọng số tại lớp được biểu diễn tổng quát:

|  |  |
| --- | --- |
|  | (1.22) |

trong đó:

* + : Ma trận trọng số kết nối từ lớp thứ đến lớp thứ .
  + : Vector gradient của lớp , thể hiện mức độ thay đổi của hàm mất mát theo đầu vào của hàm kích hoạt tại lớp .
  + : Vector đầu ra của các neural trong lớp .
* Gradient của bias : Gradient của hàm mất mát đối với vector tham số bias tại lớp :

|  |  |
| --- | --- |
|  | (1.23) |

với .

Gradient của trọng số là tích ngoài của vector gradient và vector đầu ra của lớp trước. Điều này cho phép tính toán toàn bộ các trọng số cùng lúc bằng cách sử dụng phép nhân ma trận.

* Cập nhật trọng số: Thuật toán tối ưu được dùng để điều chỉnh trọng số và tham số bias nhằm giảm sai số. Ma trận trọng số cho lớp sẽ được cập nhật toàn bộ như sau:

|  |  |
| --- | --- |
|  | (1.24) |

trong đó:

* + : Ma trận trọng số của lớp với kích thước ​.
  + ​: Ma trận gradient của hàm mất mát theo ma trận trọng số .
  + : Tốc độ học (learning rate).

Lan truyền ngược chính là cốt lõi trong việc huấn luyện các ANN, giúp tối ưu hóa các trọng số của mạng bằng cách giảm thiểu sai số giữa đầu ra dự đoán và giá trị thực tế. Đây cũng là nền tảng giúp các mạng neural sâu xử lý các bài toán phức tạp như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên và dự đoán dữ liệu [38], [46].

### Hiệu quả và hạn chế của ANN trong các tác vụ NLP

#### Hiệu quả

Với sự xuất hiện của các hàm kích hoạt, cấu trúc của ANN giúp cho mô hình có khả năng học được các mối liên hệ phi tuyến tính tốt hơn nhiều so với những mô hình tuyến tính truyền thống như SVM, Naïve Bayes [47]. Điều này cho phép ANN có khả năng xử lý những mối quan hệ phức tạp trong ngôn ngữ tự nhiên [48].

Ngoài ra, các chi tiết trong cấu trúc của ANN như số lớp, số neural của mỗi lớp, loại hàm kích hoạt,… đều có thể tùy chỉnh, giúp cho ANN giải quyết được nhiều loại bài toán NLP khác nhau như phân loại văn bản, nhận diện cảm xúc và nhận dạng thực thể có tên [49]. Mặt khác, ANN có thể kết hợp tốt với các vector word embedding để cải thiện hơn nữa chất lượng đầu vào [50].

#### Hạn chế

Từ cấu trúc của ANN, dễ thấy rằng các mô hình này có nhiều hạn chế khi thực hiện các tác vụ NLP [51]. Đầu tiên, vì không có cơ chế nhớ hoặc lưu trữ trạng thái, ANN không thể ghi nhớ các thông tin ở xa nhau nên không nắm bắt được ngữ cảnh, đặc biệt là trong các đoạn văn dài [52]. Tiếp theo, ANN xem dữ liệu đầu vào là dữ liệu tĩnh với kích thước cố định, nên đối với các bài toán có độ dài của dữ liệu đầu vào không cố định thì chúng không thể xử lý hiệu quả [53]. Mặt khác, do không có khả năng mô hình hóa các mối quan hệ giữa các từ ở cách xa nhau, nên ANN không thể đáp ứng được các bài toán NLP đòi hỏi sự hiểu sâu về ngữ nghĩa, như là hỏi đáp, tóm tắt văn bản hay sinh văn bản [54]. Chính vì những hạn chế đó, các mô hình kế thừa từ ANN như RNN và LSTM đã ra đời, dần thay thế mô hình này trong nhiều tác vụ NLP hiện đại [55].

## Tổng quan về Mạng neural hồi quy (RNN)

### Giới thiệu về RNN

Mạng neural hồi quy (Recurrent Neural Network – RNN) là một phân lớp của ANN, được thiết kế để có thể xử lý chuỗi dữ liệu tuần tự, chẳng hạn như văn bản, giọng nói hoặc chuỗi dữ liệu theo thời gian, nơi mà thứ tự của các phần tử trong chuỗi có vai trò quan trọng [56]. Trong mạng neural truyền thằng (Feedforward Neural Network – FNN) – dạng truyền thống của ANN, mỗi dữ liệu đầu vào sẽ được xử lý một cách độc lập[57]. Nhưng RNN thì sử dụng các kết nối hồi quy, trong đó đầu ra của một neural tại một bước trong chuỗi sẽ được truyền sang như là đầu vào cho neural ở bước kế tiếp[58]. Điều này cho RNN khả năng nắm bắt các mối quan hệ phụ thuộc theo thời gian, hoặc các cấu trúc trong mẫu dữ liệu [59].

### Cấu trúc của RNN

Cấu trúc của một RNN cơ bản [60] sẽ được minh họa bằng hình 3 sau đây:

A computer screen shot of a black background

AI-generated content may be incorrect.

Hình 3: Cấu trúc của một mạng neural hồi quy (RNN)

Giải thích các minh họa trong hình 3:

* : Vector chứa giá trị của đầu vào tại bước thứ của chuỗi
* : Vector chứa giá trị của trạng thái ẩn tại bước thứ của chuỗi
* : Đầu ra tại bước thứ của chuỗi
* : Ma trận trọng số của kết nối từ đầu vào đến trạng thái ẩn
* : Ma trận trọng số của kết nối từ trạng thái ẩn đến trạng thái ẩn
* : Ma trận trọng số của kết nối từ trạng thái ẩn đến đầu ra
* : Tham số bias ứng với trạng thái ẩn tại bước thứ của chuỗi.
* : Tham số bias ứng với đầu ra tại bước thứ của chuỗi.

Nhìn chung, mô hình RNN sẽ hoạt động theo thứ tự lần lượt, từng bước của chuỗi [60]. Tại mỗi bước thứ trong chuỗi, RNN sẽ:

* Nhận và từ bước thứ
* Tính toán : và sẽ được dùng để tính như trong công thức sau:

|  |  |
| --- | --- |
|  | (1.25) |

Trong công thức trên, hàm được sử dụng làm hàm kích hoạt phi tuyến để tránh hiện tượng bùng nổ gradient (exploding gradient), khi giá trị của sẽ luôn được giữ ở trong khoảng từ đến .

* Tính toán : thường được tính toán theo công thức sau:

|  |  |
| --- | --- |
|  | (1.26) |

Với hàm kích hoạt sẽ được lựa chọn tùy theo yêu cầu của bài toán.

Trong quá trình lan truyền ngược, các ma trận trọng số , và , cùng với các tham số bias và sẽ được cập nhật, nhằm tối ưu hóa hiệu quả của RNN trên tập dữ liệu huấn luyện [61].

### Hiệu quả của RNN trong các tác vụ NLP

So với FNN, RNN đã có những cải tiến quan trọng để mang lại hiệu quả cao hơn trong các tác vụ NLP [62]. Với cấu trúc dạng chuỗi của mình – tương tự như cấu trúc của văn bản, RNN có khả năng duy trì được ngữ cảnh sau khi dữ liệu trải qua nhiều bước trong chuỗi [63]. Đồng thời, các trạng thái ẩn đã giúp RNN có khả năng ghi nhớ trạng thái tại các bước trước đó trong chuỗi [63], thứ mà FNN không thể làm được. Nhiều tác vụ NLP đã từng ghi nhận các kết quả tích cực khi sử dụng RNN, như là dịch máy, sinh văn bản và mô hình hóa ngôn ngữ [62].

Tuy nhiên, RNN sau khi ra đời thì cũng đã sớm bộc lộ các vấn đề của mình. Đáng chú ý nhất là hiện tượng gradient biến mất (vanishing gradient), xảy ra khi sai số tại đầu ra của chuỗi được truyền ngược về đầu vào của chuỗi qua rất nhiều bước trung gian [64]. Vì các hàm kích hoạt như được sử dụng nhiều lần, khiến gradient khi đến được các bước đầu tiên của chuỗi đã trở nên rất nhỏ, và làm cho trọng số tại các bước đầu tiên của RNN không được cập nhật [64]. Đồng thời, cũng chính vì phải trải qua rất nhiều bước trung gian nên RNN gặp khó khăn khi phải nhớ những thông tin ở quá xa nhau, thứ tương đối phổ biến trong ngôn ngữ tự nhiên [58]. Mặt khác, cấu trúc dạng chuỗi của RNN không cho phép chúng xử lý dữ liệu một cách song song, làm cho các quá trình huấn luyện hay xử lý các chuỗi dữ liệu dài diễn ra chậm chạp [65]. Những vấn đề trên đã dẫn đến sự ra đời của các mô hình cải tiến tiếp theo như LSTM và GRU[66].

## Tổng quan về Bộ nhớ ngắn hạn dài (LSTM)

### Giới thiệu về LSTM

Bộ nhớ ngắn hạn dài (Long short-term memory – LSTM) là một dạng của RNN được thiết kế để khắc phục được hiện tượng vanishing gradient – vấn đề chính của các mô hình RNN truyền thống [67]. Một trong những điểm mới của LSTM là chúng có một bộ nhớ ngắn hạn đi kèm với RNN, và bộ nhớ này có thể được duy trì qua rất nhiều bước của chuỗi. Điểm cải tiến này khiến cho LSTM có khả năng xử lý các thông tin quan trọng mà không cần phải chú ý đến khoảng cách giữa các thông tin ấy đã đem lại lợi thế cho chúng trong các tác vụ NLP so với những mô hình RNN truyền thống [67].

### Các thành phần cấu trúc của LSTM

Về tổng thể, cấu trúc của LSTM giống như RNN, vẫn là một chuỗi các bước (hay neural). Tuy nhiên, cấu trúc của từng bước thì đã có sự thay đổi, hình 4 dưới đây sẽ phần nào minh họa điều ấy:

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 4: Cấu trúc của bước thứ t trong mô hình LSTM

Các ký hiệu cho các thành phần cấu trúc của mỗi bước trong LSTM [67]:

* : Giá trị của đầu vào tại bước thứ
* : Giá trị của trạng thái ẩn tại bước thứ , đóng vai trò vừa là bộ nhớ ngắn hạn cho bước thứ , vừa là đầu ra tại bước thứ
* : Giá trị của trạng thái ô nhớ (cell state) tại bước thứ , đóng vai trò là bộ nhớ dài hạn của chuỗi
* : Cổng quên (forget gate), có vai trò quyết định xem những phần nào của thì nên bị quên đi
* : Cổng đầu vào (input gate), có vai trò quyết định xem những thông tin mới nào từ và sẽ được thêm vào
* : Cổng đầu ra (output gate), có vai trò quyết định xem những thông tin nào từ và sẽ được chuyển thành đầu ra tại bước thứ , tức
* : Những thông tin từ và được xem là có ích để thêm vào
* : Ma trận trọng số cho kết nối giữa đầu vào và bước thứ
* : Ma trận trọng số cho kết nối giữa bước thứ và bước thứ
* : Tham số bias, xuất hiện ở ba cổng , , và vector

Quá trình lan truyền xuôi trên LSTM tại bước thứ sẽ trải qua các công đoạn như sau [67]:

* Nhận từ đầu vào và từ bước thứ trước đó
* Tính forget gate để chọn ra những thông tin từ mà nên bị quên đi:

|  |  |
| --- | --- |
|  | (1.27) |

* Tính input gate để chọn ra những thông tin mới ở trong và mà nên được thêm vào :

|  |  |
| --- | --- |
|  | (1.28) |

* Tính output gate để chọn ra những thông tin nào từ nên được sử dụng để tính đầu ra :

|  |  |
| --- | --- |
|  | (1.29) |

* Tính để tìm kiếm những thông tin mới mà có tiềm năng để đưa vào :

|  |  |
| --- | --- |
|  | (1.30) |

* Cập nhật cell state , bằng cách kết hợp các thông tin cũ được chọn bởi và các thông tin mới có tiềm năng được lọc bởi :

|  |  |
| --- | --- |
|  | (1.31) |

* Tính toán đầu ra bằng cách sử dụng để lọc thông tin từ :

|  |  |
| --- | --- |
|  | (1.32) |

Với cấu trúc như trên, dễ thấy rằng khi lan truyền ngược, các sai số thu được ở đầu ra của mạng sẽ được lưu trữ và duy trì trong cell state, nhờ vậy mà nó sẽ luôn được trả về cho từng cổng trong mỗi bước của LSTM, cho đến khi các cổng này học được cách tối ưu những sai số ấy [68].

### So sánh hiệu quả của LSTM với RNN trong các tác vụ NLP

LSTM, với sự xuất hiện của cell state trong kiến trúc so với RNN được trình bày ở trên, đã chứng tỏ được khả năng nhớ các thông tin lâu hơn vượt trội hơn so với RNN [69], đồng thời còn có thể học được các phụ thuộc của thông tin ở khoảng cách xa hơn so với RNN và cho ra hiệu quả cao hơn trong các tác vụ NLP cần khả năng lưu trữ ngữ cảnh tốt, ví dụ như gán nhãn cho chuỗi hay sinh văn bản [70]. Mặt khác, sự xuất hiện của các forget gate, input gate và output gate đã giúp LSTM tránh được hiện tượng “vanishing gradient” – một trong các vấn đề nghiêm trọng nhất của RNN [66], [67]. Dưới đây là bảng thể hiện chi tiết sự khác biệt về hiệu quả trong các tác vụ NLP của LSTM so với RNN [66]:

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **RNN** | **LSTM** |
| Khả năng ghi nhớ thông tin lâu dài | Thường quên các thông tin quan trọng sau một khoảng cách xa | Có thể ghi nhớ thông tin lâu dài |
| Hiện tượng “vanishing gradient” | Thường xảy ra | Được khắc phục nhờ vào cải tiến trong kiến trúc |
| Độ chính xác trong các tác vụ NLP cần ngữ cảnh dài | Thấp, do không thể ghi nhớ thông tin một cách lâu dài | Cao hơn rõ rệt so với RNN |
| Tốc độ huấn luyện | Nhanh hơn vì có cấu trúc đơn giản hơn | Chậm hơn vì trong cấu trúc có thêm nhiều thành phần |
| Yêu cầu về tài nguyên tính toán | Ít hơn vì có cấu trúc đơn giản hơn | Nhiều hơn vì có số lượng tham số nhiều hơn |
| Hiệu quả nhìn chung trong các tác vụ NLP | Giới hạn trong các bài toán xử lý nội dung ngắn hoặc không phức tạp về mặt ngữ cảnh | Được sử dụng rộng rãi cho các tác vụ của NLP, trước khi Transformer xuất hiện |

Bảng 1: Bảng so sánh sự khác nhau về hiệu quả trong các tác vụ NLP giữa RNN và LSTM

Tuy nhiên, LSTM vẫn tồn tại một số hạn chế nhất định. Khả năng xử lý dữ liệu một cách song song vẫn không được cải thiện, do cấu trúc tổng quan dạng chuỗi từ RNN được giữ nguyên, nên những khó khăn trong việc huấn luyện và xử lý dữ liệu vẫn còn đó [65]. Cũng vì kiến trúc còn mang tính tuần tự, nên dù đã làm tốt hơn rất nhiều so với RNN ở khả năng lưu trữ ngữ cảnh, nhưng LSTM vẫn gặp khó khi xử lý các phụ thuộc của những thông tin ở khoảng cách rất xa, khi mà vẫn sẽ có mất mát về gradient và thông tin trong cell state sau khi trải qua quá nhiều bước của chuỗi [67]. Mặt khác, sự xuất hiện của các cổng trong từng bước của cấu trúc đã làm cho LSTM tốn tài nguyên và khó mở rộng hơn RNN [71]. Chính những điểm hạn chế trên đã dẫn đến sự ra đời của Transformer – một bước nhảy vọt mới trong sự phát triển của NLP [72].

## Tổng quan về Embedding văn bản

### Giới thiệu về Embedding văn bản

Embedding văn bản là một kỹ thuật trong NLP, nhằm mục đích biểu diễn lại các chuỗi của từ, như một câu văn hay đoạn văn, thành các vector số trong một không gian ngữ nghĩa nhiều chiều, sao cho các đặc trưng về ngữ nghĩa của từ và ngữ cảnh của chuỗi được giữ lại nhiều nhất có thể [73].

Máy tính không thể hiểu được ngôn ngữ tự nhiên ở dạng văn bản. Vì vậy, cần thiết phải chuyển đổi từ ngữ sang một dạng số học, để các mô hình ML, DL có thể xử lý văn bản. Với kỹ thuật embedding, các văn bản giờ đây đã có thể được biểu diễn sang một cấu trúc dạng số học, thứ sẽ có vai trò là dữ liệu đầu vào cho các mô hình để thực thi những tác vụ NLP [74].

### Lợi ích của Embedding văn bản

Các kỹ thuật embedding văn bản đã cho thấy vai trò hỗ trợ quan trọng đối với các tác vụ NLP [75]. Chúng có thể biến đổi kiểu dữ liệu từ văn bản thành các vector số, thứ có thể được dùng làm đầu vào hoặc dữ liệu huấn luyện của các mô hình ML [76]. Một số kỹ thuật embedding văn bản có khả năng giữ lại được những thông tin về ngữ nghĩa và mối liên hệ giữa các từ, giúp tăng hiệu quả cho nhiều tác vụ NLP khác nhau như phân loại văn bản, tóm tắt hay dịch máy [77].

### Các cách biểu diễn Word embedding

Biểu diễn word embedding là quá trình ánh xạ để biểu diễn từng từ trong văn bản đầu vào sang một vector số thực, mỗi vector này chứa các thông tin về ngữ nghĩa, cú pháp của từ và ngữ cảnh của từ trong văn bản đầu vào [75]. Dựa vào cách mà vector số thực được sinh ra, các kỹ thuật embedding có thể được chia thành hai loại chính: embedding tĩnh và embedding theo ngữ cảnh[78]

#### Embedding tĩnh

Các kỹ thuật embedding tĩnh là thế hệ đầu tiên của những kỹ thuật biểu diễn word embedding trong NLP [75]. Về nguyên lý hoạt động, các kỹ thuật này sẽ gán cho mỗi từ một vector số cố định, bất kể ngữ cảnh mà từ đó xuất hiện [79]. Với nguyên lý đơn giản như vậy, các kỹ thuật embedding tĩnh rất dễ để huấn luyện, đồng thời có chi phí tính toán thấp; những điều này phù hợp với các tác vụ NLP liên quan đến phân loại, như là phân loại văn bản và gán nhãn cho văn bản [80]. Tuy nhiên, cũng vì sự đơn giản trong nguyên lý mà các kỹ thuật này không thể phân biệt được sự khác nhau về ngữ nghĩa và cú pháp của một từ trong các ngữ cảnh khác nhau, khiến chúng kém hiệu quả trong các tác vụ NLP yêu cầu sự hiểu sâu về ngữ cảnh, như dịch máy, tóm tắt văn bản hay trả lời câu hỏi [81].

#### Embedding theo ngữ cảnh

Các kỹ thuật embedding theo ngữ cảnh là thế hệ tiếp theo của những kỹ thuật biểu diễn word embedding trong NLP, ra đời nhằm khắc phục những hạn chế của các kỹ thuật embedding tĩnh trong việc phân biệt ngữ nghĩa của từ dựa trên ngữ cảnh [82]. Về nguyên lý hoạt động, các kỹ thuật này sẽ sử dụng thêm các thông tin về ngữ cảnh trên câu đầu vào để sinh ra vector số cho từ [83]. Nguyên lý này cho phép các kỹ thuật embedding theo ngữ cảnh có thể nắm bắt các phụ thuộc về ngữ nghĩa, cú pháp của từ tùy theo ngữ cảnh một cách hiệu quả hơn, vì thế nên các kỹ thuật embedding theo ngữ cảnh đang ngày càng trở thành lựa chọn ưu tiên trong các hệ thống NLP hiện đại [84]. Tuy nhiên, kỹ thuật này có phần phức tạp nên không phù hợp với các bài toán phân loại đơn giản; mặt khác, chúng cũng yêu cầu tài nguyên tính toán lớn hơn và khó huấn luyện hơn [85].

## Tổng quan về mô hình Transformer

### Giới thiệu về Transformer

Mô hình Transformer được giới thiệu bởi Vaswani và cộng sự vào năm 2017 với những cải tiến vượt bậc về hiệu suất trong nhiều tác vụ NLP như dịch máy, phân loại văn bản và sinh văn bản [86]. Khác với các mô hình truyền thống dựa trên hồi quy (như RNN), Transformer sử dụng “cơ chế chú ý” (attention), cho phép nó xử lý các dữ liệu tuần tự một cách hiệu quả và linh hoạt hơn. Đồng thời, nhờ có cơ chế này mà Transformer có thể xử lý dữ liệu một cách song song, giúp gia tăng đáng kể tốc độ huấn luyện và cải thiện khả năng biểu diễn ngữ nghĩa [86]. Ngày nay, Transformer đã trở thành nền tảng cho nhiều mô hình ngôn ngữ lớn, giúp cải thiện khả năng hiểu ngữ nghĩa và sinh ngôn ngữ tự nhiên [86].

### Kiến trúc mô hình Transformer

#### Tổng quan

Kiến trúc của Transformer bao gồm hai phần chính: bộ mã hóa (encoder) và bộ giải mã (decoder), bên trong mỗi phần này lại bao gồm nhiều lớp xếp chồng lên nhau [86]. Trong khi encoder chịu trách nhiệm tạo ra các vector số thực có ý nghĩa ngữ nghĩa, còn gọi là các biểu diễn liên tục (continuous representation) từ đầu vào, thì decoder có trách nhiệm là tạo ra đầu ra mong muốn từ các biểu diễn của encoder. Để đơn giản hóa thì, sau khi nhận được đầu vào là một chuỗi , encoder sẽ ánh xạ X thành một chuỗi của các biểu diễn . Sau đó, decoder sẽ sử dụng chuỗi để lần lượt sinh ra từng phần tử cho chuỗi đầu ra [86]. Với mỗi phần tử được decoder sinh ra, nó sẽ được decoder sử dụng lại như một phần đầu vào để sinh ra các phần tử tiếp theo, hay có thể nói là mô hình sẽ “tự động hồi quy” [86].

Nguyên lý hoạt động của Transformer chủ yếu dựa vào cơ chế attention, giúp nó xác định được sự phụ thuộc của các từ trong câu đối với nhau. Cơ chế này đã cho Transformer khả năng chú ý đến các từ khác trong câu khi tạo ra các biểu diễn mới, từ đó cải thiện được độ chính xác khi nắm bắt những mối quan hệ về ngữ nghĩa và ngữ pháp. Đồng thời, cơ chế này cho phép mô hình khả năng xử lý các chuỗi, các câu dài mà không gặp phải những vấn đề liên quan đến độ dài của chuỗi như RNN hay LSTM.

#### Encoder

Bộ mã hóa (encoder) được tạo thành bằng cách xếp chồng lớp giống hệt nhau. Mỗi lớp sẽ có hai lớp con (sub-layer) [86]:

* Lớp con đầu tiên là một lớp cơ chế tự chú ý đa đầu (multi-head self-attention), giúp mô hình học được nhiều loại quan hệ khác nhau trong chuỗi đầu vào
* Lớp con thứ hai là một mạng truyền thẳng kết nối đầy đủ theo từng vị trí (position-wise feed-forward network), giúp mô hình biến đổi đặc trưng sau khi đã hiểu được ngữ cảnh của chuỗi đầu vào.

Dữ liệu đầu vào sau khi đi qua các kết nối dư thừa (residual connection) của mỗi lớp con sẽ tiếp tục đi qua lớp chuẩn hóa (normalization layer). Nói cách khác, mỗi lớp con sẽ có đầu ra là , với là hàm thực hiện bởi chính lớp con đó [86].

Để tạo điều kiện thuận lợi cho những kết nối này, tất cả các lớp con trong mô hình cũng như các lớp embedding (embedding layers) sẽ tạo ra các đầu ra có kích thước cố định, thường là 512 chiều [86].

#### Decoder

Giống như encoder, decoder cũng bao gồm một chuỗi các lớp (thường là ) giống hệt nhau được xếp chồng lên nhau, mỗi lớp cũng có hai lớp con tương tự. Tuy nhiên, mỗi lớp con của decoder có thêm một lớp phụ đứng đầu tiên, đó là lớp tự chú ý đa đầu có mặt nạ (masked multi-head self-attention), có nhiệm vụ che giấu thông tin, không cho mô hình biết trước kết quả của từ tiếp theo trong chuỗi mà chưa thực hiện dự đoán. Lớp phụ này khi kết hợp với thực tế là các embedding đầu ra bị lệch đi một vị trí (từ đầu tiên của chuỗi luôn được cấp cho mô hình), sẽ đảm bảo rằng các dự đoán của mô hình cho vị trí chỉ có thể phụ thuộc vào các đầu ra đã biết ở các vị trí phía trước . Tức là đảm bảo rằng trong quá trình dự đoán từ tiếp theo, mô hình chỉ có thể “thấy” các từ trước đó, chứ không thấy được các từ đứng sau [86].

#### Attention

Cơ chế chú ý (attention) là thành phần trung tâm của Transformer, cho phép mô hình tập trung vào các phần khác nhau của đầu vào khi nó tạo ra đầu ra. Cơ chế này hoạt động bằng cách tính toán một ma trận attention score để biết mức độ quan trọng của từng từ trong đầu vào đối với từ hiện tại mà mô hình đang xử lý. Ma trận này sẽ được chuyển đổi thành xác suất thông qua hàm softmax, từ đó tạo ra một trọng số cho mỗi từ. Kết quả là mô hình có thể tạo ra các biểu diễn mà nhấn mạnh các từ quan trọng hơn trong các ngữ cảnh cụ thể [87].

Trong các tác vụ NLP, bước đầu tiên sẽ là chuyển đổi từng từ trong câu đầu vào thành một embedding vector số thông qua các thuật toán embedding, mỗi từ sau khi đi qua thuật toán embedding sẽ trở thành một vector có kích thước [87]. Công việc này xảy ra tại lớp đầu tiên của encoder, còn với các lớp encoder phía trên, đầu vào của chúng sẽ là một vector số - kết quả của lớp encoder bên dưới, thay vì là một từ. Câu đầu vào, giờ đây, trở thành một chuỗi các vector có kích thước đồng nhất. Độ dài của chuỗi vector này là một siêu tham số và tương ứng với độ dài của câu đầu vào dài nhất trong tập huấn luyện [87].

A screenshot of a computer

Description automatically generated

Hình 5: Mô tả cơ chế self-attention

Sau khi thực hiện embedding, bước tiếp theo trong encoder sẽ là tính toán cơ chế tự chú ý. Mỗi vector trong chuỗi vector vừa thu được phía trên, sau khi được nhân với ba ma trận trọng số mà mô hình có được sau quá trình huấn luyện, sẽ biến đổi thành ba vector mới [87]:

* Vector truy vấn : biểu diễn mục tiêu mà từ thứ đang tìm kiếm

|  |  |
| --- | --- |
|  | (1.33) |

* Vector khóa : biểu diễn những thuộc tính mà từ thứ đang có

|  |  |
| --- | --- |
|  | (1.34) |

* Vector giá trị : thể hiện mức độ quan trọng của tất cả các từ trong câu đối với từ thứ , kể cả chính từ đó.

|  |  |
| --- | --- |
|  | (1.35) |

Ba vector này có kích thước nhỏ hơn các embedding vector, chúng kết hợp với nhau để tạo ra một phép chiếu truy vấn, khóa và giá trị của mỗi từ trong câu đầu vào [87].

Sau khi đã tính toán cơ chế tự chú ý, mô hình sẽ tính toán chỉ số score – chỉ số ám chỉ mức độ liên quan của đối với từng từ của câu đầu vào. Giả sử ta đang xem xét từ thứ , thì score của từ này được xác định bằng tích vô hướng của vector truy vấn và vector khóa .

Tiếp đó, ta chia chỉ số score cho (ta lấy căn bậc 2 của số chiều của và , lúc này là , để làm ổn định các gradient; trên thực tế, con số này có thể tùy chỉnh được) rồi truyền kết quả sang cho phép toán softmax. Phép toán này sẽ chuẩn hóa tất cả các score, sao cho tất cả đều là số dương và tổng của chúng bằng [87], với công thức như sau:

|  |  |
| --- | --- |
|  | (1.35) |

Trong đó:

* là giá trị đầu vào thứ .
* là tổng giá trị đầu vào.

Kết quả của hàm softmax thể hiện mức độ phù hợp của mỗi từ trong câu đầu vào đối với từ ở vị trí , từ phù hợp nhất sẽ có điểm softmax cao nhất [87].

Bước tiếp theo là nhân mỗi vector giá trị với kết quả softmax. Ở bước này, chúng ta cần giữ nguyên giá trị của từ hoặc các từ quan trọng và loại bỏ các từ không liên quan, nên ta sẽ nhân các giá trị ấy với các số nhỏ (learning rate = chẳng hạn) [87].

Bước cuối cùng là cộng các vector giá trị có trọng số để tạo ra đầu ra của lớp self-attention tại vị trí xem xét. Kết thúc quá trình tính self-attention, vector giá trị là một vector mà chúng ta có thể gửi đến mạng neural truyền thẳng, nhưng trong quá trình triển khai thực tế, phép tính này thường được thực hiện dưới dạng ma trận [87].

#### Multi-head attention

Multi-head attention là một kỹ thuật mở rộng của cơ chế chú ý, cho phép mô hình học nhiều mặt khác nhau của dữ liệu cùng một lúc. Thay vì chỉ sử dụng một cơ chế chú ý duy nhất, multi-head attention tạo ra nhiều đầu chú ý (head attention), mỗi head có thể tập trung vào các phần khác nhau của đầu vào. Kết quả từ các head này, sau đó sẽ được kết hợp lại để giúp mô hình có cái nhìn toàn diện hơn về ngữ cảnh và cải thiện khả năng hiểu ngôn ngữ tự nhiên [88]. Việc sử dụng multi-head attention cải thiện hiệu suất của lớp chú ý theo hai cách [87]:

* Tăng khả năng tập trung vào các vị trí khác nhau của mô hình: Khi một từ chứa thông tin của tất cả mã hóa khác, trong đó có cả chính nó, sẽ có khả năng nó bị chi phối bởi điều này. Trong thực tế, nếu chúng ta đang dịch một câu như: “Con vật không băng qua đường vì nó quá mệt”, việc chứa thông tin của các mã hóa khác sẽ rất có ích khi biết từ “nó” đang ám chỉ từ nào [87].
* Cung cấp cho lớp chú ý nhiều “không gian con biểu diễn” (representation subspaces): Với multi-head attention, mô hình không chỉ có một mà là nhiều bộ ma trận trọng số truy vấn, khóa và giá trị (Transformer sử dụng tám multi-head attention, do đó chúng tạo ra tám bộ ma trận , , cho mỗi bộ encoder và decoder). Mỗi bộ trong số bộ này được khởi tạo ngẫu nhiên. Sau khi đào tạo, mỗi bộ được sử dụng để chiếu các embedding input (hoặc các vector từ bộ encoder và decoder trước đó) vào các không gian con biểu diễn khác nhau [87].

A chart of different colored squares

Description automatically generated

Hình 6: Minh họa cấu trúc của mỗi head trong multi-head attention

Với multi-head attention, các ma trận trọng số , , được duy trì riêng biệt cho mỗi head, từ đó dẫn đến mỗi , , đều khác nhau. Như đã đề cập ở trên, X được nhân với các ma trận , , để tạo ra các ma trận , , . Nếu cùng thực hiện một phép tính attention tám lần khác nhau với các ma trận trọng số khác nhau. Chúng ta sẽ có tám ma trận khác nhau [87].

Tuy nhiên, trước khi đưa vào mạng truyền thẳng, ta cần đưa các ma trận về thành một vector duy nhất (cần lưu ý rằng với mỗi là một từ) bằng cách nối các ma trận này với nhau, và sau đó nhân với một ma trận trọng số bổ sung [87].

A diagram of a test

Description automatically generated with medium confidence

Hình 7: Cách định hình lại ma trận Z

Ma trận được thêm vào để kết hợp các thông tin từ các heads và giảm kích thước của vector về ban đầu. Ma trận này là một tham số của mô hình, tức là nó được khởi tạo ngẫu nhiên ban đầu và sau đó sẽ được huấn luyện cùng với các tham số khác trong quá trình tối ưu hóa. Nó không phải là một ma trận cố định hay được tính toán từ đầu vào mà sẽ tự động điều chỉnh để tối ưu hóa kết quả của mô hình [87].

#### Kết nối dư thừa

Kết nối dư thừa (Residual Connection) là một kỹ thuật được sử dụng trong mạng neural sâu để tạo điều kiện thuận lợi cho việc đào tạo các kiến ​​trúc rất sâu [89]. Kỹ thuật này được giới thiệu trong ResNet bởi Kaiming He và cộng sự vào năm 2015, dựa trên một khái niệm được gọi là Học dư thừa (Residual learning). Ý tưởng chính của khái niệm này là cho phép đầu vào của một lớp bỏ qua các hoạt động của lớp đó và được thêm trực tiếp vào đầu ra của lớp đó. Các hoạt động có thể là bất cứ thứ gì như biến đổi tuyến tính, biến đổi phi tuyến tính, chuẩn hóa, bỏ qua, gộp, v.v. [90]

A diagram of a layer

Description automatically generated

Hình 8: Minh họa một kết nối dư thừa

Kết nối dư thừa hoạt động như sau: Giả sử ta có một mạng neural một (hoặc nhiều) lớp và mục tiêu là tìm một hàm để biểu diễn một lớp mạng nhất định, trong trường hợp này là hàm mà chúng ta muốn mô hình hoá. Với Học dư thừa, thay vì trực tiếp huấn luyện mạng để mô phỏng hàm , chúng ta sẽ huấn luyện để mạng mô phỏng một hàm . Phần dư thừa này được định nghĩa sao cho khi cộng với , chúng có thể khớp với . Hay nói cách khác .

A diagram of a layer

Description automatically generated

*Hình 9: Mạng nơ-ron thường (bên trái) và mạng nơ-ron có kết nối dư (bên phải)*

Ưu điểm của Học dư thừa trong Transformers là tạo điều kiện cho việc truyền tín hiệu theo cả hai hướng lan truyền thuận (forward propagation) và lan truyền ngược (backward propagation), đồng thời giải quyết vấn đề “vanishing gradient” khi mà các mạng phức tạp với các lớp sâu hơn như Transformer sẽ dễ gặp phải vấn đề này, và kết nối dư thừa sẽ giúp khôi phục gradient bằng cách thực hiện ánh xạ [90]

#### Lan truyền thuận

Như đã đề cập trong các phần trước, làn tuyền thuận là quá trình tính toán của mạng DL để cho kết quả đầu ra mô hình. Điều này vẫn tương tự đối với mô hình Transformer, tuy nhiên, mối quan hệ cho quá trình lan truyền ngược có thể được mô tả bằng do tồn tại kết nối dư thừa, nên , sẽ là đầu vào cho một tập hợp các lớp khác trong mạng, nhưng để đơn giản hóa, chúng ta chỉ xem xét một đơn vị kết nối dư cho trường hợp này [87].

#### Lan truyền ngược

Đây là nơi kết nối dư giải quyết vấn đề “vanishing gradient”. Để tính toán gradient, chúng ta xem xét riêng đạo hàm riêng của hàm mất mát đối với đầu vào . Sử dụng công thức trên, phương trình lan truyền ngược sẽ là

|  |  |
| --- | --- |
|  | (1.36) |

#### Lớp chuẩn hóa

Lớp chuẩn hóa (Normalization layer) sẽ chuẩn hóa đầu ra của những bước tính toán trước đó trên các features. Trong quá trình chuẩn hóa lớp, mỗi mục trong lô được chuẩn hóa trong phạm vi từ đến . Trong mô hình transformer, nó có thể là embedding vectơ có 512 chiều của một từ. Chúng ta tính toán giá trị trung bình và phương sai của các mục này và thay thế từng giá trị trong ma trận bằng công thức:

|  |  |
| --- | --- |
|  | (1.37) |

Trong quá trình đào tạo mô hình Transformer, các đầu ra do các neural trong một lớp được tạo ra sau khi áp dụng hàm kích hoạt cho tổng trọng số của các đầu vào, và được gọi là các kích hoạt. Sự phân bố của các kích hoạt này thay đổi theo thời gian do các tham số mạng thay đổi. Chuẩn hóa làm giảm sự thay đổi này bằng cách duy trì sự phân bố ổn định của các kích hoạt, giúp quá trình đào tạo trở nên trơn tru [87].

#### Masked Multi Head Attention

Masked Multi Head Attention là multi head attention, có chức năng dùng để encode các từ câu đích trong quá trình dịch. Lúc cài đặt cần phải che đi các từ ở tương lai chưa được mô hình dịch đến, bằng cách nhân với một vector chứa các giá trị 0,1. Trong decoder còn có một multi head attention khác có chức năng chú ý các từ ở mô hình encoder, layer này nhận vector key và value từ mô hình encoder, và output từ layer phía dưới. Do muốn so sánh sự tương quan giữa từ đang được dịch với các từ nguồn [91].

#### Mã hóa vị trí

Word embeddings đã giúp Transformer biểu diễn ngữ nghĩa của một từ, tuy nhiên cùng một từ ở vị trí khác nhau của câu lại mang ý nghĩa khác nhau. Do đó, Transformers có thêm một phần mã hóa vị trí (positional encoding) để đưa thêm thông tin về vị trí của một từ [91]. Mã hóa vị trí của từ thứ được tính như sau:

|  |  |
| --- | --- |
|  | (1.38) |
|  | (1.39) |

Trong đó là vị trí của từ trong câu, là giá trị phần tử thứ trong embeddings có độ dài . Sau đó cộng vector và Embedding vector để thêm thông tin về vị trí của từ vào embedding của nó.

A diagram of a mathematical equation

Description automatically generated

*Hình 10: Mã hóa vị trí từ embedding*

### Ứng dụng của Transformer trong NLP

Transformer là một mô hình hiện đại, tạo ra nhiều bước tiến lớn trong lĩnh vực NLP. Với cơ chế tự chú ý, Transformer đã có khả năng nắm bắt những phụ thuộc giữa các từ trong câu mà không quan tâm đến khoảng cách giữa chúng, giúp mô hình này có thể hiểu ngữ cảnh tốt hơn nhiều so với các mô hình dựa trên kiến trúc RNN [92]. Ngoài ra, việc có thể xử lý các dữ liệu trong chuỗi một cách song song đã cho phép Transformer giảm được thời gian huấn luyện và xử lý dữ liệu, đồng thời nâng cao khả năng mở rộng. Transformer đã trở thành nền tảng cho rất nhiều các mô hình ngôn ngữ tiên tiến, điển hình là các mô hình ngôn ngữ lớn [93].

### Các biến thế của Transformer

Từ khi ra đời, kiến trúc Transformer đã được mở rộng và cải tiến thành nhiều biến thể để phù hợp hơn với từng loại bài toán trong xử lý ngôn ngữ tự nhiên cũng như các lĩnh vực khác [94]. Một trong những biến thể nổi bật là BERT (Bidirectional Encoder Representations from Transformers), sử dụng kiến trúc encoder và huấn luyện theo hướng hai chiều, giúp mô hình hiểu ngữ cảnh tốt hơn trong các tác vụ như phân loại, trích xuất thông tin hay hỏi đáp [94]. Trong khi đó, GPT (Generative Pre-trained Transformer) tập trung vào kiến trúc decoder và huấn luyện theo hướng từ trái sang phải, phù hợp cho các tác vụ sinh văn bản [94]. Ngoài ra, còn có RoBERTa – một phiên bản cải tiến của BERT với cách huấn luyện tối ưu hơn, T5 (Text-to-Text Transfer Transformer) – mô hình chuyển đổi mọi bài toán NLP thành bài toán sinh văn bản, và DistilBERT – phiên bản nhẹ hơn của BERT nhằm giảm chi phí tính toán mà vẫn giữ được hiệu quả [94]. Những biến thể này thể hiện khả năng tùy biến linh hoạt của Transformer và liên tục thúc đẩy giới hạn của các hệ thống AI về ngôn ngữ.

## Tổng quan về Mô hình ngôn ngữ lớn

### Giới thiệu về LLM

Mô hình ngôn ngữ lớn (Large Language Model - LLM) là một loại chương trình [AI](https://www.cloudflare.com/learning/ai/what-is-artificial-intelligence/) chuyên về ngôn ngữ, được thiết kế để xử lý và tạo ra các văn bản bằng ngôn ngữ tự nhiên. Các mô hình LLM được huấn luyện trên [các tập dữ liệu khổng lồ](https://www.cloudflare.com/learning/ai/big-data/), do đó có tên là “lớn”. LLM hoạt động dựa trên nền tảng là các kỹ thuật và mô hình ML, cụ thể là mô hình Transformer – một mô hình có hiệu quả tốt khi thực hiện các tác vụ NLP [95].

Nói theo cách đơn giản hơn, LLM là một chương trình máy tính đã được huấn luyện bằng các tập dữ liệu rất lớn để có thể nhận dạng và diễn giải ngôn ngữ tự nhiên thành các loại dữ liệu phức tạp khác [95]. Chúng sử dụng phương pháp DL để phân tích và hiểu được cách mà các ký tự, các từ, các câu hoạt động và liên hệ với nhau trong một văn bản, từ đó nhận ra sự khác biệt giữa các ý nghĩa, ngữ cảnh hoặc mục đích biểu đạt của các ký tự, các từ, các câu trong một văn bản ngôn ngữ tự nhiên mà không cần sự can thiệp trực tiếp của con người [95]. Sau quá trình huấn luyện ban đầu, các LLM sẽ được huấn luyện thêm để tinh chỉnh cho phù hợp với các nhiệm vụ cụ thể mà nó cần phải hoàn thành, chẳng hạn như dịch văn bản, tóm tắt hoặc diễn giải một đoạn văn, hay là tạo ra câu trả lời cho các truy vấn bằng ngôn ngữ tự nhiên [95].

### Kiến trúc tổng quan của LLM

Trong LLM hay trong bất cứ mô hình ML nào, kiến trúc đóng một vai trò rất quan trọng. Kiến trúc khác nhau sẽ khiến các mô hình xử lý mối quan hệ giữa các đơn vị dữ liệu theo những cách khác nhau, ảnh hưởng đến hiệu suất và hiệu quả trong quá trình xử lý dữ liệu của mô hình [96].

Các LLM hiện đại thường được thiết kế dựa trên kiến trúc của mô hình Transformer, một kiến trúc vốn đã có khả năng xử lý chuỗi dữ liệu hiệu quả hơn nhiều so với các mô hình trước đó nhờ vào cơ chế tự chú ý. Tuy nhiên, tùy vào các đặc điểm của tác vụ cần xử lý mà kiến trúc Transformer trong LLM có thể được tùy biến cho phù hợp, từ số lượng các encoder/decoder, có sử dụng mã hóa vị trí hay không,… nhằm tối ưu hóa hiệu quả của LLM cho từng mục đích sử dụng [84].

Có ba dạng kiến trúc của Transformer đang phổ biến trong các LLM hiện đại [96]:

* Encoder – decoder: Kiến trúc này gồm có một encoding component chứa các encoder, được xếp lên trên một decoding component chứa các decoder. Nhờ vậy, kiến trúc này cho ra hiệu quả vượt trội trong các tác vụ NLP liên quan đến việc sinh ra một văn bản mới sau khi hiểu được văn bản cũ, như là dịch máy hay tóm tắt văn bản.
* Encoder – only: Kiến trúc này chỉ gồm một encoding component chứa các encoder, có khả năng chuyển đổi các văn bản đầu vào thành các biểu diễn ngữ cảnh tốt hơn mà không cần phải sinh ra một đầu ra. Kiến trúc này tập trung vào việc phân tích dữ liệu đầu vào hơn là tạo ra một đầu ra dài.
* Decoder – only: Kiến trúc này chỉ có decoding component với nhiều lớp decoder. Vì không có các lớp encoder nên kiến trúc này không thể xử lý toàn bộ văn bản đầu vào được, bù lại, chúng lại có khả năng sinh ra từ tiếp theo tốt hơn các kiến trúc khác.

### Tổng quan về Tokenization

#### Giới thiệu về Tokenization

Tokenization là bước tiền xử lý đầu tiên trong pipeline của một LLM, có vai trò chuyển đổi một văn bản trong ngôn ngữ tự nhiên thành các đơn vị rời rạc (có thể là từ, subword hay ký tự) được gọi là token, sau đó ánh xạ chúng thành một chỉ số trong từ điển của mô hình. Các chỉ số này sẽ được biến đổi thành các vector số tại lớp embedding, để trở thành đầu vào cho các lớp tiếp theo của LLM [97].

LLM về cơ bản là một chương trình máy tính, nó không thể hiểu được văn bản ở dạng thô. Do đó, cần có một bước xử lý trung gian để chuyển đổi văn bản từ dạng thô sang các cấu trúc số học, đó chính là nhiệm vụ của Tokenization [98].

#### Các chiến lược xử lý văn bản thành các token

Trên thực tế, vì cấu trúc linh hoạt của ngôn ngữ tự nhiên và các yêu cầu đa dạng của các tác vụ NLP khác nhau, dẫn đến sự ra đời cách tiếp cận với tokenization khác nhau. Mỗi cách tiếp cận đều có phương pháp xử lý tokenzation riêng, nhưng nhìn chung thì chúng sẽ thuộc một trong ba chiến lược: mã hóa theo từ, mã hóa theo bán từ và mã hóa theo ký tự [99].

* Mã hóa theo từ: Chiến lược này sẽ xem mỗi từ trong văn bản đầu vào là một token. Lấy ví dụ về văn bản , sau khi sử dụng mã hóa theo từ, ta thu được token và token
* Mã hóa theo bán từ: Chiến lược này sẽ chia từ ra thành các bán từ, mỗi bán từ sẽ được xem là một token. Ví dụ về văn bản , sau khi sử dụng mã hóa theo bán từ, ta có thể thu được các token , , và
* Mã hóa theo ký tự: Chiến lược này sẽ tách văn bản thành các ký tự - đơn vị nhỏ nhất của ngôn ngữ tự nhiên. Ví dụ, văn bản “máy” sau khi trải qua sự mã hóa theo ký tự, ta sẽ thu được các token , và

Vì có cách xử lý vấn đề tokenization khác nhau, nên mỗi chiến lược tokenization trên sẽ có những lợi thế riêng [99]. Dưới đây sẽ là bảng biểu diễn chi tiết sự khác biệt giữa các chiến lược tokenization này.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí đánh giá** | **Mã hóa theo từ** | **Mã hóa theo bán từ** | **Mã hóa theo ký tự** |
| Mức độ tính toán | Nhẹ | Trung bình | Nặng |
| Khả năng giữ ngữ nghĩa của từ | Tốt | Tương đối tốt | Kém |
| Khả năng linh hoạt với từ mới | Không thể | Có thể | Vô cùng linh hoạt với từ mới |

Bảng 2: So sánh các chiến lược tokenization

#### Tác động của chiến lược tokenization đối với hiệu suất của LLM

Tokenization, vì có nhiệm vụ chuyển đổi ngôn ngữ tự nhiên từ dạng thô sang dạng số học mà máy tính có thể xử lý được, nên nó có vai trò tối quan trọng, có ảnh hưởng lớn đến hiệu suất của LLM về cả độ chính xác và chi phí tính toán [98].

Với chiến lược mã hóa theo từ, mỗi token sẽ mang trọn vẹn các thông tin ngữ nghĩa của một từ, giúp mô hình học được nhanh hơn. Tuy nhiên, độ tổng quát hóa của mô hình sẽ giảm xuống, vì chiến lược này không có cách nào để mã hóa các từ không nằm trong từ điển của nó [98].

Với chiến lược mã hóa theo bán từ, các từ sẽ được chia nhỏ thành các bán từ phổ biến hơn. Vì vậy, chiến lược này có thể xử lý từ mới tốt hơn, tăng khả năng tổng quát hóa và thu nhỏ được kích thước từ điển; đồng thời, các thông tin ngữ nghĩa của một từ cũng được duy trì, nên tốc độ học của mô hình vẫn được đảm bảo. Hiện nay, hầu hết các LLM hiện đại đều chọn dùng chiến thuật mã hóa theo bán từ để xử lý tokenization [98].

Với chiến lược mã hóa theo ký tự, mỗi từ được chia nhỏ thành các ký tự - đơn vị nhỏ nhất trong ngôn ngữ tự nhiên. Điều này giúp cho LLM có độ linh hoạt cực cao và hoàn toàn không gặp vấn đề khi xử lý từ mới hay lỗi chính tả. Đánh đổi cho lợi thế này là chi phí huấn luyện tăng vọt, vì chuỗi đầu vào trở nên rất dài, và bị mất các thông tin ngữ cảnh ở mức trừu tượng cao, do từ đã bị xé thành các phần tử quá nhỏ [98].

### Quy trình để huấn luyện một LLM

A diagram of a work flow

Description automatically generated

Hình 11: Quá trình huấn luyện một mô hình LLM.

Quy trình để huấn luyện một LLM có thể được mô tả ngắn gọn như trong hình 11. Về cơ bản, các bước để huấn luyện một LLM tương tự với việc huấn luyện các mô hình ML khác, nhưng thêm vào đó, các kỹ thuật phức tạp và các phương pháp DL cũng sẽ được sử dụng để huấn luyện các LLM này [95].

Bước đầu tiên (1) là chuẩn bị dữ liệu huấn luyện: Các dữ liệu văn bản sẽ được lấy từ nhiều nguồn khác nhau. Vì chất lượng của dữ liệu huấn luyện có ảnh hưởng rất lớn đến kết quả huấn luyện của LLM, nên chất lượng của dữ liệu huấn luyện này thường được kiểm soát rất chặt chẽ [95].

Bước thứ hai (2) là xây dựng kiến trúc mô hình: Tuy được xây dựng dựa trên kiến trúc của Transformer, nhưng kiến trúc của các LLM cần phải được điều chỉnh để có thể tạo ra một mô hình có khả năng xử lý ngôn ngữ mạnh mẽ [95].

Bước thứ ba (3) là huấn luyện mô hình: Các LLM sẽ được huấn luyện để có thể học được các thông tin về mối liên hệ giữa các từ, sự thay đổi ngữ nghĩa của từ trong các ngữ cảnh khác nhau và cấu trúc của câu văn, một cách tổng quan [95].

Bước thứ tư (4) là tinh chỉnh mô hình: Tùy theo nhiệm vụ cần được hoàn thành, các LLM sẽ trải qua một quá trình huấn luyện thứ hai, thường được gọi là fine-tuning, để điều chỉnh cách mà mô hình sẽ xử lý ngôn ngữ, sao cho phù hợp và hiệu quả với nhiệm vụ nhất [95].

Bước thứ năm (5) là thực hiện đánh giá mô hình bằng các phương pháp đánh giá chuyên biệt dành cho LLM: Sau quá trình tinh chỉnh, các LLM sẽ trải qua các bài đánh giá khác nhau, chẳng hạn như kiểm tra khả năng hiểu ngữ cảnh, khả năng trả lời câu hỏi, khả năng tạo văn bản hoặc khả năng suy luận. Các bài đánh giá này giúp xác định mức độ chính xác, tính mạch lạc và khả năng tổng quát hóa của mô hình trên nhiều loại nhiệm vụ ngôn ngữ khác nhau [95].

Bước thứ sáu (6) là triển khai mô hình: Các LLM, sau khi trải qua một hoặc nhiều lần giai đoạn huấn luyện từ bước 1 đến bước 5, sẽ được triển khai lên môi trường thực tế [95].

#### Huấn luyện trước

Trong huấn luyện một LLM, hai giai đoạn chủ yếu là huấn luyện trước (pre-training) và tinh chỉnh mô hình (fine-tuning). Trong giai đoạn pre-training, các mô hình Transformer được huấn luyện trên dữ liệu thô, một tập hợp rất nhiều văn bản. Quá trình này được thực hiện bằng các kỹ thuật học không giám sát (unsupervised learning) – một hình thức huấn luyện để mô hình có khả năng dán nhãn cho dữ liệu mà không cần có sự can thiệp của con người, với mục tiêu là giúp mô hình hiểu được các mẫu thống kê của ngôn ngữ tự nhiên. Để có thể đạt được độ chính xác cao hơn trên kiến trúc của Transformer, người ta đã tăng số lượng tham số của mô hình và tăng kích thước dữ liệu, thứ sẽ làm cho mô hình trở nên lớn hơn, từ đó có khả năng hiểu các mẫu thống kê của ngôn ngữ tự nhiên tốt hơn. Do vậy, hầu hết các LLM tiên tiến hiện nay đều có quy mô hàng tỷ tham số và được huấn luyện trên các tập dữ liệu khổng lồ, cũng vì thế mà quá trình pre-training thường kéo dài, đòi hỏi tài nguyên tính toán cao và chi phí rất lớn [96].

#### Tinh chỉnh mô hình

Quá trình pre-training đã cho phép kiến trúc Transformer có được những hiểu biết cơ bản về ngôn ngữ tự nhiên, nhưng là không đủ để có thể thực hiện các tác vụ NLP cụ thể với độ chính xác cao. Để tránh tiêu tốn thêm thời gian và chi phí huấn luyện, Transformer tận dụng các kỹ thuật học chuyển giao để tách giai đoạn pre-training khỏi giai đoạn tinh chỉnh (fine-tuning), cho phép các nhà phát triển chọn các mô hình đã trải qua giai đoạn pre-training rồi tinh chỉnh chúng dựa trên các tập dữ liệu huấn luyện hẹp hơn, cụ thể hơn theo nhiệm vụ cần xử lý. Trong nhiều trường hợp, quá trình tinh chỉnh được thực hiện với sự hỗ trợ của người đánh giá, sử dụng kỹ thuật reinforcement learning từ phản hồi của con người [96].

Nhìn chung, quá trình huấn luyện hai giai đoạn đã cho phép điều chỉnh LLM tùy theo các tác vụ hạ nguồn, hay nói cách khác, tính năng này đã biến LLM thành mô hình nền tảng của vô số ứng dụng NLP, và cũng là cơ sở để có thể xây dựng một chatbot hỗ trợ học lịch sử Việt Nam [96].

### Quy trình sinh văn bản của LLM (text generation)

#### Cơ chế sinh văn bản của LLM

Sau khi trải qua quá trình huấn luyện trước, các LLM sẽ có khả năng sinh văn bản, bằng cách dự đoán token tiếp theo dựa trên những token nó nhận được. Tại mỗi bước, LLM sẽ tạo ra một phân phối xác suất trên một từ điển cố định tương ứng với xác suất mà một token trong từ điển là token tiếp theo của chuỗi token đầu vào, sau đó mô hình sẽ chọn một token đầu ra thông qua các decoder của nó. Cách chọn token sẽ phụ thuộc vào decoder, tùy vào việc mô hình muốn chọn token có xác suất cao nhất, hay sẽ chọn ngẫu nhiên một trong các token có xác suất cao nhất. Quá trình này sẽ lặp đi lặp lại, cho đến khi văn bản gặp token kết thúc chuỗi, hoặc khi văn bản đã đạt độ dài tối đa [98].

#### Các phương pháp chọn token tiếp theo

Các chiến lược chọn token tiếp theo (chiến lược sampling) có vai trò quan trọng trong quá trình sinh văn bản của LLM [100]. Sampling sẽ quyết định chọn token nào làm token tiếp theo cho chuỗi đầu vào, nên có tác động trực tiếp đến chất lượng đầu ra của mô hình. Một số chiến thuật sampling phổ biến hiện nay có thể kể ra như chọn tham lam (greedy sampling), chọn top-k (top-k sampling), chọn hạt nhân (nucleus sampling hay top-p sampling) và chọn theo độ sắc nét (temperature sampling) [101].

* Greedy sampling: Chiến lược này sẽ luôn chọn token có xác suất cao nhất làm token cho đầu ra. Vì thế, chiến lược này có độ ổn định cao, nhưng lại thiếu sáng tạo vì token đầu ra sẽ dễ lặp lại [101].
* Top-k sampling: Chiến lược này sẽ chọn ngẫu nhiên một token từ token có xác suất cao nhất. Do đó, chiến lược này cho ra token đầu ra đa dạng hơn, đồng thời kiểm soát tốt độ ngẫu nhiên của đầu ra. Tuy nhiên, nếu nhỏ quá thì chiến lược này không khác gì với greedy sampling, còn nếu lớn quá thì chiến lược có thể sẽ chọn một token không liên quan lắm về mặt ngữ nghĩa đối với chuỗi token trước đó, gây lạc đề [101].
* Top-p sampling: Chiến lược này sẽ chọn ra một nhóm có ít token nhất mà tổng xác suất của chúng vượt qua một ngưỡng . Dễ thấy, chiến lược này sẽ giảm thiểu khả năng chọn nhầm một token có xác suất thấp, giúp mô hình giảm khả năng bị lạc đề. Tuy vậy, chiến lược này phụ thuộc vào , nên ngưỡng này phải được tinh chỉnh rất kỹ để đảm bảo chất lượng của văn bản được sinh ra [101].
* Temperature sampling: Chiến lược này xây dựng một công thức điều chỉnh lại giá trị của phân phối xác suất với một hệ số . Tùy thuộc vào mà mô hình sẽ chuyển đổi cách chọn token đầu ra, hoặc là giống với greedy sampling, hoặc là giống với top-k sampling. Sự linh hoạt này cho phép mô hình được điều chỉnh văn phong của đầu ra, hoặc là cứng nhắc hơn, hoặc là linh hoạt hơn [101].

### Streaming trong LLM

#### Giới thiệu về streaming trong LLM

Streaming trong LLM là một cơ chế cho phép mô hình trả kết quả một cách tuần tự theo từng token hoặc cụm token, thay vì phải sinh xong văn bản mới trả kết quả. Nhờ tận dụng cơ chế sinh văn bản của LLM, sự xuất hiện của nó đã làm cải thiện đáng kể trải nghiệm của người dùng [102].

#### Cách hoạt động của streaming

Như đã nói, cách hoạt động của streaming dựa trên cơ chế sinh văn bản của LLM. Ngay khi LLM chọn được token tiếp theo cho chuỗi văn bản đầu vào, nó sẽ trả luôn token ấy ra ngoài ngay lập tức, trước khi lặp lại quá trình chọn token kế tiếp. Cụ thể thì, cơ chế sinh đầu ra của LLM có streaming có thể được mô tả như sau[102]

1. LLM tiếp nhận một văn bản làm đầu vào
2. LLM thực hiện các công tác tính toán, xử lý và chọn ra token tiếp theo cho đầu vào.
3. Lúc này, cơ chế streaming sẽ thông qua một dòng dữ liệu liên tục (stream data pipe) để đưa đến người dùng hoặc ứng dụng ở cuối stream.
4. LLM đưa token chọn được vào chuỗi của văn bản đầu vào, sau đó quay lại bước 1 để lặp lại quá trình này, cho đến khi văn bản đạt độ dài tối đa hoặc gặp token kết thúc.

Để cơ chế streaming hoạt động hiệu quả, LLM phải có các kỹ thuật dựng stream data pipe như WebSocket hoặc gRPC streaming đi kèm, đồng thời phải có khả năng lưu giữ trạng thái qua mỗi lần lặp [102].

#### Hiệu quả của cơ chế streaming

Sự xuất hiện của cơ chế streaming có ý nghĩa đặc biệt lớn đối với các ứng dụng cần khả năng xử lý theo thời gian thực như chatbot hoặc trợ lý ảo [102]. Nó giúp quá trình phản hồi của LLM, giờ đây, trở nên tự nhiên hơn, đồng thời làm giảm khoảng thời gian mà người dùng phải chờ đợi sự phản hồi, nâng cao trải nghiệm sử dụng của người dùng [103]. Trong bối cảnh mà LLM đang được triển khai ngày càng rộng rãi trong các hệ thống đối thoại, streaming đã trở thành một cơ chế thiết yếu để cải thiện hiệu suất và khả năng ứng dụng thực tiễn của LLM [103].

### Các kỹ thuật NLP trong LLM

#### Định nghĩa Token

Trong Xử lý Ngôn ngữ Tự nhiên, một token là đơn vị cơ bản của văn bản được xử lý bởi các thuật toán. Mỗi token có thể là một từ, một ký tự, hoặc một chuỗi ký tự có ý nghĩa trong ngữ cảnh cụ thể. Ví dụ câu: *"Tôi yêu lập trình."*:

* Token hóa theo từ: ["Tôi", "yêu", "lập", "trình", "."]
* Token hóa theo ký tự: ["T", "ô", "i", " ", "y", "ê", "u", " ", "l", "ậ", "p", " ", "t", "r", "ì", "n", "h", "."]
* Token hóa theo subword: ["Tôi", "yêu", "lập", "##trình"]

Token không thể xử lý trực tiếp bởi máy tính, nên cần được biểu diễn bằng các dạng **số hóa**. Có nhiều cách biểu diễn token trong NLP, một vài cách phổ biến nhất gồm:

* One-Hot Encoding: Gán mỗi token một vector nhị phân.
* Index Mapping: Gán mỗi token một số nguyên duy nhất.
* Embedding: Sử dụng thuật toán embedding để tạo vector biểu diễn ngữ nghĩa.

#### Kỹ thuật Chunking

Chunking là một kỹ thuật trong xử lý ngôn ngữ tự nhiên và phân tích văn bản. Nó giúp phân tích văn bản lớn thành các phân đoạn hoặc khối nhỏ, dễ quản lý, giúp xử lý và phân tích khối lượng dữ liệu lớn dễ dàng hơn. Kỹ thuật chunking thường được sử dụng trong nhiều ứng dụng khác nhau, chẳng hạn như tóm tắt nội dung, đánh giá tình cảm hoặc trích xuất thông tin chính. Trong mọi trường hợp, nó đóng vai trò quan trọng trong việc nâng cao hiệu suất của ứng dụng, giúp dễ dàng xử lý, phân tích hoặc đưa vào mô hình. Việc này đặc biệt quan trọng với các mô hình ngôn ngữ lớn (LLM) vì chúng có giới hạn về số lượng token có thể xử lý cùng lúc (context length) [104], [105]. Các kỹ thuật chunking phổ biến bao gồm:

* Theo Độ Dài Cố Định: Chia văn bản thành các đoạn có số lượng token cố định, ví dụ mỗi đoạn có 8000 token.

Giả sử ta có một văn bản với token, ta chia thành đoạn:

​ là token thứ , với là số token mỗi chunk. Mỗi chunk ​ là:

với .

* Theo Cấu Trúc Ngữ Nghĩa: Chia văn bản theo đoạn văn, câu, hoặc theo các mục logic (chương, tiêu đề).
* Sliding Window (Cửa Sổ Trượt): Dùng một cửa sổ cố định (ví dụ 8000 token) và trượt qua văn bản, với một khoảng trùng lặp (overlap) để không mất ngữ cảnh.

Với , khung token và khoảng trượt (overlap). Chunk thứ :

với chạy đến khi .

Ví dụ văn bản: "*Học máy là một lĩnh vực khoa học dữ liệu phát triển nhanh chóng. LLM có khả năng xử lý ngôn ngữ tự nhiên mạnh mẽ.*" Token hóa: [*Học, máy, là, một, lĩnh, vực, khoa, học, ...*].

* Chia Theo Độ Dài Cố Định (L = 5):
  + Chunk 1: [Học, máy, là, một, lĩnh]
  + Chunk 2: [vực, khoa, học, dữ, liệu]
  + Chunk 3: [phát, triển, nhanh, chóng, LLM]
  + Chunk 4: [có, khả, năng, xử, lý]
  + Chunk 5: [ngôn, ngữ, tự, nhiên, mạnh]
  + Chunk 6: [mẽ].
* Sliding Window (L = 5, O = 2):
  + Chunk 1: [Học, máy, là, một, lĩnh]
  + Chunk 2: [một, lĩnh, vực, khoa, học]
  + Chunk 3: [vực, khoa, học, dữ, liệu]
  + ...

Lý do cần phải chunking cho dữ liệu vì hầu hết các mô hình LLM bị giới hạn ngữ cảnh, các LLM chỉ hỗ trợ số lượng token nhất định. Nếu văn bản quá dài, chunking giúp chia nhỏ văn bản để không vượt quá giới hạn này. Việc chunking được thiết kế để giữ cho các đoạn văn bản vẫn có ý nghĩa riêng lẻ, giúp bảo toàn ngữ cảnh. Ngoài ra, chunking giúp lưu trữ và tìm kiếm dễ dàng hơn, tăng hiệu quả tìm kiếm khi dùng embedding hoặc vector database [104], [105].

#### Kỹ thuật Embedding

Embedding là kỹ thuật chuyển đổi dữ liệu (thường là văn bản, hình ảnh, hoặc âm thanh) thành vector số học (tập hợp các số thực) để mô hình học máy hoặc thuật toán có thể xử lý. Ý tưởng chính chính là embeddings cố gắng biểu diễn ngữ nghĩa hoặc đặc trưng của dữ liệu trong không gian vector [106].

Giả sử chúng ta có một tập dữ liệu văn bản , trong đó ​ là một đoạn văn bản hoặc câu. Embedding là ánh xạ từ văn bản sang một không gian vector : , với , trong đó ​ là vector biểu diễn embedding của văn bản , và là số chiều của không gian vector (embedding dimension). Các embeddings thường dùng khoảng cách Euclidean hoặc cosine để đo độ tương tự giữa hai vector ​ và [106]:

* Khoảng cách Euclidean:
* Cosine Similarity:

Trong đó:

* + ​ là tích vô hướng (dot product) của hai vector.
  + là norm (độ dài) của các vector, được tính bằng:

Embedding sẽ tạo ra một không gian vector sao cho các điểm gần nhau có tính chất tương tự. Với ngữ nghĩa [106]:

Trong đó:

* + là các vector embedding tương ứng của các văn bản ​.
  + là khoảng cách Euclidean giữa hai vector.

Các điểm vector gần nhau trong không gian embedding thường có ý nghĩa tương tự nhau trong ngữ cảnh dữ liệu. Phương pháp này đặc biệt hữu ích với những đặc trưng hạng mục có số phần tử lớn ở đó phương pháp chủ yếu để biểu diễn mỗi giá trị thường là một vector dạng one-hot. Một cách lý tưởng, các giá trị có ý nghĩa tương tự nhau nằm gần nhau trong không gian embedding [106].

#### Kỹ thuật RAG

Tăng cường truy xuất tạo sinh (Retrieval Augmented Generation – RAG) là một kỹ thuật giúp nâng cao khả năng của mô hình LLM kết hợp với tri thức bên ngoài (external knowledge). Trong sự ra đời của các mô hình LLM rất hay gặp vấn đề ảo giác, trong từ điển chuyên ngành gọi là hallucination, khi đó các mô hình sinh ra những câu văn nghe rất trôi chảy và hợp lý, nhưng trong thực tế nó lại không chính xác. Chính vì đó là nguồn cảm hứng cho việc kỹ thuật RAG ra đời và như là một phương pháp để giải quyết vấn đã nêu [107].

Giả sử cơ sở dữ liệu có tập tài liệu , Một truy vấn được biểu diễn dưới dạng vector embedding , mỗi tài liệu ​ có embedding . Bài toán Retrieval là tìm tập sao cho:

Trong đó:

* : Hàm đo độ tương đồng giữa truy vấn và tài liệu được tính theo công thức khoảng cách Euclidean hoặc sự tương đồng Cosine Similarity.
* : Tập hợp các tài liệu phù hợp nhất.

Trong các mô hình ngôn ngữ truyền thống, phản hồi được tạo ra chỉ dựa trên các mẫu và thông tin đã học trước trong giai đoạn đào tạo. Tuy nhiên, các mô hình này vốn bị giới hạn bởi dữ liệu mà chúng được đào tạo, thường dẫn đến các phản hồi có thể thiếu chiều sâu hoặc kiến ​​thức cụ thể. RAG giải quyết hạn chế này bằng cách kéo dữ liệu bên ngoài khi cần trong quá trình tạo [108]. Về cơ bản, đây là một quy trình gồm hai phần bao gồm một thành phần truy xuất (retrieval) và một thành phần tạo (generation). Cách thức hoạt động của RAG được mô tả trong hình 17:

A diagram of a algorithm

Description automatically generated

Hình 19. Mô hình luồng hoạt động của kỹ thuật RAG.

* Thành phần Retrieval: Có chức năng tìm các tài liệu hoặc thông tin có liên quan có thể giúp trả lời câu hỏi truy vấn. Nó lấy truy vấn đầu vào và tìm kiếm trong cơ sở dữ liệu để thu thập thông tin có thể hữu ích cho việc tạo phản hồi.
* Thành phần Generator: Trình tạo là mô hình LLM tạo ra văn bản đầu ra cuối cùng. Nó lấy truy vấn đầu vào và ngữ cảnh được trình thu thập để tạo ra phản hồi mạch lạc và có liên quan. Trình tạo không hoạt động riêng lẻ, nó sử dụng ngữ cảnh do retrieval cung cấp để đưa ra phản hồi, đảm bảo rằng đầu ra không chỉ hợp lý mà còn giàu chi tiết và chính xác.

Cụ thể từng bước cách hoạt động của RAG như sau [108]:

* Xử lý truy vấn (Query processing): Tất cả bắt đầu bằng một truy vấn. Đây có thể là một câu hỏi, một lời nhắc hoặc bất kỳ đầu vào nào mà chúng ta muốn mô hình LLM phản hồi.
* Mô hình embedding (Embedding model): Sau đó truy vấn được chuyển đến mô hình embedding. Mô hình này chuyển đổi truy vấn thành vector, là biểu diễn số có thể được hệ thống hiểu và xử lý.
* Truy xuất Cơ sở dữ liệu vector: Vectơ truy vấn được sử dụng để tìm kiếm trong cơ sở dữ liệu vector database. Cơ sở dữ liệu này chứa các vector được tính toán trước của các ngữ cảnh tiềm năng mà mô hình có thể sử dụng để tạo phản hồi. Hệ thống truy xuất các ngữ cảnh có liên quan nhất dựa trên mức độ khớp chặt chẽ của các vector của chúng với vector truy vấn.
* Truy xuất bối cảnh (Retrieved Contexts): Các bối cảnh đã được truy xuất sau đó được chuyển đến mô hình LLM. Các bối cảnh này chứa thông tin mà LLM sử dụng để tạo ra phản hồi có hiểu biết và chính xác.
* Tạo phản hồi LLM: LLM tính đến cả truy vấn gốc và ngữ cảnh được truy xuất để tạo ra phản hồi toàn diện và có liên quan. Nó tổng hợp thông tin từ ngữ cảnh để đảm bảo rằng phản hồi không chỉ dựa trên kiến ​​thức đã có từ trước mà còn được bổ sung các chi tiết cụ thể từ dữ liệu được truy xuất.
* Phản hồi cuối cùng: Cuối cùng, LLM đưa ra phản hồi, phản hồi này được thông báo bởi dữ liệu bên ngoài thu thập được trong quá trình này, giúp phản hồi chính xác và chi tiết hơn.

Việc kết hợp RAG với mô hình ngôn ngữ LLM có khả năng tra cứu thông tin trực tiếp, như thể AI “đọc sách” trước khi trả lời. Điều này giúp phản hồi chính xác, sâu sắc và bám sát thực tế hơn, điều này đặc biệt hữu ích khi chúng ta muốn sử dụng dữ liệu chuyên biệt của chúng ta để có thể huấn luyện chatbot, đánh dấu một bước tiến lớn trong việc hiểu và tạo ngữ cảnh [108].

#### Thuật toán sắp xếp mức độ ưu tiên Reciprocal Rank Fusion

Trong quá trình truy xuất thông tin, nếu trình truy xuất không thể truy xuất các tài liệu có liên quan từ trình truy xuất, thì độ chính xác sẽ thấp và khả năng xảy ra ảo giác sẽ tăng lên [109]. Chính vì thế cần có một thuật toán để tính toán mức độ chính xác cho thông tin truy vấn được, trong dự án này sử dụng phương pháp Hợp nhất thứ hạng qua lại (Reciprocal Rank Fusion - RRF), đây là một kỹ thuật phổ biến trong việc kết hợp và sắp xếp lại các kết quả từ nhiều bộ máy tìm kiếm hoặc mô hình truy xuất thông tin khác nhau, đặc biệt là trong các hệ thống RAG. Phương pháp này giúp cải thiện độ chính xác và tính đa dạng của kết quả [109], [110]. Công thức toán biểu diễn của FFR như sau:

Trong đó:

* : Tập hợp các nguồn hoặc mô hình.
* : Vị trí xếp hạng của tài liệu trong nguồn .
* : Một hằng số để đảm bảo các tài liệu ở thứ hạng cao không bị ảnh hưởng quá nhiều (thường ).

Các bước hoạt động của RFF trong hệ thống RAG thường tuân theo các bước sau [109]:

* Truy vấn của người dùng (User Query): Quy trình bắt đầu khi người dùng nhập câu hỏi hoặc truy vấn.
* Nhiều bộ thu thập (Multiple Retrievers): Truy vấn được gửi đến nhiều bộ thu thập. Đây có thể là các mô hình thu thập khác nhau (ví dụ: dày đặc, thưa thớt, lai).
* Xếp hạng riêng lẻ (Individual Rankings): Mỗi bộ thu thập tạo ra thứ hạng riêng của các tài liệu có liên quan.
* Hợp nhất RRF (RRF Fusion): Thứ hạng từ tất cả các bộ thu thập được kết hợp bằng công thức RRF.
* Xếp hạng cuối cùng (Final Ranking): Một thứ hạng thống nhất được tạo ra dựa trên điểm RRF.
* Tạo (Generation): Mô hình tạo sử dụng các tài liệu được xếp hạng cao nhất để tạo ra câu trả lời cuối cùng.

Thuật toán FFR có ưu điểm đơn giản và hiệu quả mà lại dễ dàng triển khai. Trong hệ thống RAG, RRF giúp sắp xếp lại các kết quả một cách khách quan trong việc sắp xếp lại thứ hạng của các kết quả, giúp cải thiện chất lượng tổng thể của top-k kết quả bằng cách kết hợp ưu điểm từ nhiều nguồn, phù hợp với các bài toán yêu cầu tính đơn giản, hiệu quả và khả năng mở rộng [109], [110].

#### Tác nhân mô hình ngôn ngữ lớn (LLM Agents)

Các mô hình LLM có khả năng hiểu ngữ cảnh và sinh văn bản tốt, tuy nhiên để một LLM có khả năng suy nghĩ và trả lời chính xác điều này phức tạp hơn rất nhiều và cần có một quy trình cụ thể để hướng dẫn LLM có thể trả lời chính xác câu hỏi mong muốn, đó chính là các tác nhân LLM (LLM agents).

Mặc dù LLM thông thường đã cho thấy khả năng xử lý ngôn ngữ tự nhiên ấn tượng và cung cấp câu trả lời dựa trên dữ liệu đào tạo, chúng vẫn còn hạn chế trong việc thực hiện các tác vụ phức tạp và yêu cầu quản lý quy trình đa bước. LLM agents không chỉ hiểu và xử lý ngôn ngữ mà còn có khả năng lên kế hoạch, ra quyết định, và thực hiện các hành động cụ thể dựa trên ngữ cảnh và mục tiêu đề ra. Điều này cho phép LLM agents hoạt động một cách linh hoạt và hiệu quả hơn trong các tình huống thực tế, giúp tối ưu hóa quy trình làm việc và nâng cao trải nghiệm hiệu suất [111], [112]. Hình 18 là minh hoạ cách một tác nhân LLM tương tác với các thành phần khác.

A diagram of a company

Description automatically generated

Hình 20. Minh hoạt hệ thống tác nhân (Agent) tự động được hỗ trợ bởi LLM [113].

Một LLM agent hoạt động thông qua một quy trình có cấu trúc từ khi người dùng nhập thông tin đầu vào cho đến khi nhiệm vụ được hoàn thành. Đầu tiên, người dùng nhập một yêu cầu hoặc câu hỏi vào hệ thống. LLM agent sau đó sẽ phân tích và hiểu ngữ cảnh của yêu cầu này, sử dụng khả năng diễn giải của LLM để giải mã ý định của người dùng. Tiếp theo, LLM agent xây dựng một kế hoạch hành động, chia nhiệm vụ lớn thành các bước nhỏ hơn, có thể quản lý được [111].

Trong quá trình thực hiện, LLM agent có khả năng tương tác với các hệ thống khác (Ví dụ: Web search, Call API, v.v.) hoặc thu thập thêm dữ liệu cần thiết để hoàn thành nhiệm vụ. Nó thực hiện từng bước theo kế hoạch đã định. Cuối cùng, sau khi hoàn thành tất cả các bước, LLM agent tổng hợp kết quả và cung cấp câu trả lời hoặc thực hiện hành động cụ thể mà người dùng yêu cầu, đảm bảo rằng nhiệm vụ được hoàn thành một cách chính xác và hiệu quả [111].

#### Kỹ thuật Prompt

Mặc dù các mô hình LLM đã trở thành chuẩn mực mới trong lĩnh vực NLP, nhưng chúng đã được chứng minh là đôi khi tạo ra các câu trả lời không chính xác hoặc mơ hồ. Tuy nhiên, khả năng của nó có thể được giảm bớt thông qua một kỹ thuật gọi là kỹ thuật nhắc nhở (Prompt) [114].

Kỹ thuật Prompt có thể được định nghĩa là tạo ra đầu ra mong muốn từ LLM bằng cách đưa ra một bộ hướng dẫn rõ ràng về đầu ra mong muốn. Kỹ thuật Prompt giúp hiểu rõ hơn về khả năng và hạn chế của LLM. Phương pháp này không chỉ là thiết kế và phát triển các lời nhắc. Nó bao gồm nhiều kỹ năng và kỹ thuật cần thiết để tương tác, phát triển và kiểm tra LLM. Một lời nhắc đơn giản với một số hướng dẫn có thể giúp LLM có được đầu ra, tuy nhiên, để có được kết quả chất lượng tốt lời nhắc phải được xây dựng tốt. Một lời nhắc có thể chứa các hướng dẫn như câu hỏi và các chi tiết khác cung cấp nhiều ngữ cảnh hơn như đưa ra gợi ý về câu trả lời [114], [115], [116]. Hình 19 mô tả cách một lời nhắc hoạt động:



Hình 21. Mô họa việc các câu lệnh kiến thức được tạo ra [117].

Đầu tiên câu hỏi của người dùng nhập vào sẽ được mô hình LLM thực hiện phân tích các yêu cầu cụ thể để có được yêu cầu tri thức. Sau khi xác định được yêu cầu của câu hỏi, mô hình sẽ dựa vào các yêu cầu tri thức được học trước đó kết hợp với những yêu cầu tri thức từ câu hỏi đã được phân tích để trả lời câu hỏi [117]. Các lời nhắc này chính là nền tảng để LLM xây dựng câu hỏi và mở rộng cuộc trò chuyện [116].

### Ứng dụng LLM để xây dựng chatbot

Các mô hình LLM ngày nay đã được chứng minh là có hiệu quả vượt bậc trong các tác vụ NLP so với trước đây [118], [119], và do đó việc ứng dụng LLM vào nhiều lĩnh vực khác nhau đang ngày càng trở nên phổ biến [119]. Chính vì thế việc áp dụng một mô hình LLM phù hợp để xây dựng chatbot tuyển sinh là điều cần thiết và mang tính thực tiễn cao, giúp tối ưu hóa quy trình tuyển sinh và chăm sóc đối tượng khách hàng tiềm năng. Việc lựa chọn mô hình LLM phù hợp và huấn luyện trên lượng dữ liệu tuyển sinh chuyên biệt của Trường Đại học Đông Á sẽ tăng hiệu quả trả lời và cung cấp thông tin chính xác hơn. Ngoài ra tâm lý của giới trẻ ngày nay đề cao sự riêng tư, do đó việc tương tác với một chatbot sẽ không ảnh hưởng đến sự riêng tư của người dùng mà còn tạo tâm lý thoải moái nếu người dùng có nhiều câu hỏi và muốn tìm hiểu nhiều vấn đề. Hình 20 dưới đây là sơ đồ tổng quan cho việc sử dụng LLM để xây dựng chatbot.

A diagram of a model

Description automatically generated

Hình 22. Tổng quan hệ thống chatbot sử dụng LLM.

## Các công cụ hỗ trợ

### Ngôn ngữ lập trình Python

Python là một ngôn ngữ lập trình bậc cao, đa năng, được Guido van Rossum phát triển vào năm 1991. Ngôn ngữ này nổi bật nhờ cú pháp rõ ràng, tập trung vào tính hiệu quả và khả năng ứng dụng thực tế trong nhiều lĩnh vực [120], [121]. Python được sử dụng rộng rãi trong:

* Phát triển web: Cung cấp các framework như Django, Flask, hỗ trợ xây dựng ứng dụng từ cơ bản đến phức tạp.
* Khoa học dữ liệu và trí tuệ nhân tạo: Với các thư viện như NumPy, Pandas, TensorFlow, Python là công cụ quan trọng để phân tích dữ liệu và xây dựng hệ thống AI.
* Tự động hóa: Thích hợp để viết script xử lý các tác vụ lặp lại, từ quản lý tệp đến kiểm thử phần mềm.
* Hệ thống nhúng và IoT: Thường được dùng trên Raspberry Pi để phát triển các giải pháp IoT hoặc các hệ thống nhúng đơn giản.

Điểm mạnh của Python là khả năng tương thích đa nền tảng, hệ sinh thái thư viện phong phú và cộng đồng phát triển mạnh mẽ, giúp đáp ứng tốt các nhu cầu từ lập trình cơ bản đến ứng dụng chuyên sâu [120], [121]. Với sự phong phú đa dạng các thư viện hỗ trợ trong lĩnh vực AI và NLP, đây là lý do nghiên cứu này lựa chọn ngôn ngữ Python để thực hiện xây dựng chatbot.

### Các Framework sử dụng

#### Bộ công cụ thiết lập giao diện chatbot Chainlit

Bộ công cụ xây dựng website Chainlit là một framework mã nguồn mở được thiết kế để hỗ trợ các nhà phát triển trong việc xây dựng, thử nghiệm và triển khai giao diện người dùng (UI) tương tác cho các ứng dụng sử dụng LLM. Chainlit giúp đơn giản hóa quy trình phát triển, cung cấp các công cụ trực quan để tương tác với mô hình và nhanh chóng tạo các nguyên mẫu ứng dụng AI có giao diện thân thiện. Chainlit đặc biệt phù hợp cho việc phát triển chatbot, trợ lý ảo, và các ứng dụng hội thoại phức tạp, cho phép nhà phát triển dễ dàng theo dõi, kiểm tra logic, và tối ưu hóa phản hồi của mô hình AI [122], [123]. Việc sử dụng Chainlit trong dự án này không chỉ giúp tiết kiệm thời gian và giảm độ phức tạp trong việc xây dựng ứng dụng chatbot mà còn cung cấp một môi trường mạnh mẽ để kiểm tra và tối ưu hóa các mô hình ngôn ngữ lớn. Nhờ các tính năng trực quan và tích hợp tốt, Chainlit là một công cụ lý tưởng để phát triển các ứng dụng NLP hiện đại.

#### Bộ công cụ lưu trữ dữ liệu vector database Milvus

Milvus là công cụ mạnh mẽ dành cho các ứng dụng AI hiện đại yêu cầu xử lý dữ liệu vector lớn, với khả năng tìm kiếm và lưu trữ hiệu suất cao. Với tính năng dễ sử dụng, khả năng mở rộng, và tích hợp mạnh mẽ với các framework AI, Milvus giúp đơn giản hóa quy trình xây dựng các hệ thống thông minh như tìm kiếm ngữ nghĩa, phân loại và lưu trữ dữ liệu [124], [125]. Đối với việc xây dựng một hệ thống chatbot cần một cơ sở dữ liệu để lưu trữ lịch sử chatbot người dùng, việc lựa chọn Milvus trong dự án này là lựa chọn tối ưu khi được ứng dụng rộng rãi trong các hệ thống chatbot khác [125].

#### Công cụ quản trị cơ sở dữ liệu MongoDB

MongoDB [126] là một hệ quản trị cơ sở dữ liệu phi cấu trúc (Non Structured Query Languag - NoSQL) phổ biến, sử dụng cấu trúc dữ liệu dạng text thay vì bảng như các cơ sở dữ liệu quan hệ. Đặc điểm chính nổi bật của công cụ này nằm ở chỗ:

* Lưu trữ linh hoạt: Dữ liệu được lưu dưới dạng JSON (hoặc BSON), dễ dàng thay đổi cấu trúc.
* Khả năng mở rộng: Hỗ trợ phân mảnh (sharding) để xử lý dữ liệu lớn.
* Tốc độ cao: Tối ưu hóa cho truy vấn nhanh và xử lý dữ liệu phi cấu trúc.
* Dễ tích hợp: Thích hợp cho các ứng dụng web, mobile và big data nhờ tính linh hoạt và đa dạng API.

Với những ưu điểm trên, MongoDB thường được sử dụng trong các dự án cần lưu trữ dữ liệu phi cấu trúc hoặc bán cấu trúc, và phù hợp cho các dự án chatbot có quy mô vừa hoặc nhỏ.

#### Hệ thống bộ nhớ chatbot MemGPT

Mặc dù các mô hình LLM ngày nay đã có nhiều phát triển và cải tiến đáng kể, nhưng chúng vấn bị hạn chế bởi bộ nhớ ngắn hạn do cấu trúc của nó, và cần một hệ thống bộ nhớ mạnh mẽ để có thể ghi nhớ các ký ức và nhận thức được tình huống hiện tại. Hệ thống này phải có khả năng ghi lại và truy xuất các dấu vết bộ nhớ bằng ngữ cảnh, cho phép các tác nhân AI xử ly thông tin và đưa ra quyết định sáng suốt [127]. Một nhóm các nhà nghiên cứu tại Berkeley đã tạo ra memGPT, hoạt động như một công cụ quản lý bộ nhớ cho các mô hình ngôn ngữ lớn. Điều này cho phép truy xuất bộ nhớ dài hạn và khả năng viết cũng như bỏ qua giới hạn đầu vào của cửa sổ ngữ cảnh [128]. MemGPT, được giới thiệu trong bài báo “MemGPT: Hướng tới LLM như Hệ điều hành” [129], giúp các mô hình ngôn ngữ lớn (LLM) xử lý các cuộc hội thoại dài hơn bằng cách quản lý khéo léo các tầng bộ nhớ khác nhau. Nó biết khi nào cần lưu trữ thông tin quan trọng và truy xuất thông tin đó sau trong khi trò chuyện. Điều này giúp các mô hình AI có thể có các cuộc hội thoại kéo dài, cải thiện đáng kể tính hữu ích của chúng [130]. MemGPT được thiết kế để nâng cao hiệu suất của các Mô hình ngôn ngữ lớn (LLM) bằng cách giới thiệu một chương trình quản lý bộ nhớ tiên tiến hơn, giúp khắc phục những thách thức do các cửa sổ ngữ cảnh cố định đặt ra. Dưới đây là một số tính năng chính của MemGPT [131]:

* Quản lý bộ nhớ (Memory Management): MemGPT kết hợp hệ thống bộ nhớ theo tầng vào bộ xử lý LLM ngữ cảnh cố định, cấp cho nó khả năng quản lý bộ nhớ của riêng nó. Bằng cách xử lý thông minh các tầng bộ nhớ khác nhau, nó mở rộng ngữ cảnh có sẵn trong cửa sổ ngữ cảnh giới hạn của LLM, giải quyết vấn đề về cửa sổ ngữ cảnh bị hạn chế thường gặp trong các mô hình ngôn ngữ lớn.
* Quản lý ngữ cảnh ảo (Virtual Context Management): MemGPT giới thiệu một phương pháp được gọi là quản lý ngữ cảnh ảo. Đây là một tính năng chính hỗ trợ quản lý cửa sổ ngữ cảnh của LLM.
* Lấy cảm hứng từ hệ điều hành (Operating System-Inspired): Kiến trúc của MemGPT lấy cảm hứng từ các hệ điều hành truyền thống, đặc biệt là hệ thống bộ nhớ phân cấp giúp dữ liệu di chuyển giữa bộ nhớ nhanh và chậm dễ dàng hơn. Cách tiếp cận này cho phép quản lý tài nguyên bộ nhớ hiệu quả, tương tự như cách hệ điều hành cung cấp ảo giác về tài nguyên bộ nhớ lớn cho các ứng dụng thông qua phân trang bộ nhớ ảo.
* Xử lý gián đoạn (Interruption Handling): MemGPT sử dụng các ngắt để quản lý luồng điều khiển giữa nó và người dùng, đảm bảo tương tác mượt mà và quản lý bộ nhớ hiệu quả trong quá trình hoạt động.
* Bối cảnh đàm thoại mở rộng (Extended Conversational Context): Thông qua quản lý bộ nhớ hiệu quả, MemGPT tạo điều kiện cho bối cảnh đàm thoại mở rộng, cho phép tương tác dài hơn và mạch lạc hơn, vượt qua những hạn chế do cửa sổ ngữ cảnh có độ dài cố định áp đặt.

Về bản chất, MemGPT là một bước tiến đáng kể trong việc sử dụng Mô hình ngôn ngữ lớn, tạo ra con đường cho các tương tác hiệu quả và mở rộng hơn, giống với diễn ngôn của con người bằng cách quản lý tài nguyên bộ nhớ một cách thông minh [127].

# PHƯƠNG PHÁP ĐỀ XUẤT XÂY DỰNG CHATBOT

## Mô tả và phân tích bài toán

Mục tiêu của bài toán là xây dựng một hệ thống chatbot hỏi đáp tuyển sinh bằng cách sử dụng các công nghệ trí tuệ nhân tạo. Hệ thống chatbot phải có khả năng trả lời một cách tự động, liên tục và tự nhiên như con người để trả lời các câu hỏi tuyển sinh trong phạm vi tuyển sinh trường Đại học Đông Á nhằm hỗ trợ kịp thời nhu cầu tìm hiểu về chương trình học, ngành đào tạo và các chính sách học bổng… cho người dùng đang tìm hiểu về trường. Đối với các yêu cầu hoặc câu hỏi nằm ngoài phạm vi của người dùng, chatbot phải biết gửi lời xin lỗi và nhắc nhở người dùng những phạm vi mà chatbot có thể hỗ trợ, đồng thời có thể điều hướng hoặc cung cấp các thông tin cán bộ giảng viên trực thuộc để có thể giải đáp các yêu cầu nằm ngoài phạm vi của chatbot. Hình 21 là sơ đồ tổng quan bài toán xây dựng hệ thống chatbot hỏi đáp tuyển sinh.

A diagram of a chatbot

Description automatically generated

Hình 23. Tổng quan bài toán xây dựng hệ thống chatbot hỏi đáp tuyển sinh.

Đầu vào của bài toán này là trả lời các câu hỏi liên quan đến lĩnh vực tuyển sinh tại trường đại học Đông Á do người dùng nhập vào. Đầu ra của bài toán là câu trả lời của người dùng dạng văn bản, cung cấp đầy đủ thông tin chính xác cho câu trả lời, ngoài ra chatbot cũng gợi ý người dùng các câu hỏi hoặc thông tin liên quan đến câu hỏi trước mà người dùng có thể quan tâm, từ đó có thể tiếp tục dẫn dắt người dùng tìm hiểu sâu hơn vào các vấn đề mà người dùng quan tâm.

## Phương pháp đề xuất

Phương pháp đề xuất xây dựng chatbot tuyển sinh sử dụng mô hình ngôn ngữ lớn (LLM) được thiết kế nhằm tối ưu hóa quy trình tư vấn tuyển sinh, đảm bảo cung cấp thông tin nhanh chóng, chính xác và thân thiện với người dùng. Hình 20 là sơ đồ tổng quan phương pháp đề xuất xây dựng chatbot sử dụng LLM cho dự án này. Quy trình bao gồm từ việc thu thập và xử lý dữ liệu đến vận hành chatbot, sử dụng các tác nhân thông minh và đánh giá hiệu quả.

Hệ thống được chia thành 2 giai đoạn chính, cụ thể giải đoạn thứ nhất chịu trách nhiệm tiền xử lý và chuẩn bị hệ thống, các bước bao gồm (1, 2, 3, 4, 5.1, 5.2). Tại giai đoạn này chúng tôi tiến hành thử nghiệm và đánh giá các thành phần quan trọng ảnh hưởng đến hiệu suất, bao gồm đánh giá hiệu quả embedding, chất lượng trả lời câu hỏi của mô hình LLM, và thử nghiệm và đánh giá các thuật toán retrieval và độ chính xác. Giai đoạn thứ 2 (bao gồm các bước còn lại) chúng tôi tiến hành triển khai hệ thống chatbot theo quy trình thiết kế theo phương pháp đề xuất.



Hình 24. Phương pháp đề xuất xây dựng chatbot hỏi đáp tuyển sinh sử dụng LLM.

Trong sơ đồ hình 22, mỗi thành phần đóng vai trò cụ thể trong việc xử lý truy vấn và cải thiện trải nghiệm người dùng. Cụ thể mỗi bước chịu trách nhiệm và xử lý công việc như sau:

* (1): Bước đầu tiên trong phương pháp đề xuất của chúng tôi là thu thập và tạo kịch bản dữ liệu. Dữ liệu tuyển sinh được thu thập trên website chính thức của Trường Đại học Đông Á, đồng thời chúng tôi cũng tạo ra các kịch bản câu hỏi có thể xảy ra dựa trên tình huồng tuyển sinh thực thực tế được lấy từ các trang facebook tuyển sinh của trường năm 2024..
* (2): Bước thứ hai là thực tiện tiền xử lý dữ liệu, trong bước này thực hiện làm sạch dữ liệu, chuẩn hóa dữ liệu và gắn nhãn cho dữ liệu tùy theo từng danh mục hoặc pham vi chủ đề câu hỏi. Đồng thời để tăng thêm tính chính xác cho câu trả lời của chatbot sau này, chúng tôi bổ sung sung các trường hợp từ viết tắt phổ biến,
* (3): Bước thứ ba là thực hiện phân đoạn dữ liệu, các mẫu có độ dài vượt quá khả năng xử lý của các mô hình LLM sẽ được cắt ngắn và sắp xếp lại để có thể dễ dàng xử lý và phân tích. Việc chia nhỏ giúp cải thiện tìm kiếm thông tin và giảm tải cho các mô hình xử lý dữ liệu lớn.
* (4): Bước thứ tư sau khi hoàn thành tất cả các quy trình chuẩn hóa, dữ liệu sẽ được đưa vào mô hình embedding thể thực hiện embedding dữ liệu thành các vector. Việc embedding dữ liệu mục đích là để mô có thể tính toán và truy xuất ngược lại sau này. Ngoài ra, dữ liệu được chuyển đổi thành vector thông qua mô hình embedding này sẽ giúp biểu diễn ý nghĩa ngữ nghĩa của dữ liệu. Mô hình embedding đảm bảo rằng các đoạn dữ liệu tương tự sẽ gần nhau trong không gian vector.
* (5.1): Sau đó lưu trữ các vector embedded này vào một vector dabase. Điều này giúp tối ưu hóa quá trình truy xuất thông tin trong các bước sau truy xuất sau này. Ngoài ra, sau khi thực hiện embedding và lưu vector embedding và cơ sở dữ liệu, cũng cần thực hiện đánh giá kết quả retrieval bằng các phương pháp retrieval (5.2), điều này giúp đưa ra cái nhìn tổng quan về chất lượng dữ liệu, hiệu quả embedding và tốc độ truy xuất của vector databse. Đồng thời chắc chắn rằng các truy vấn nhập vào có thể tìm được mẫu trả lời đúng một cách hiệu quả.
* (6): Trong quá trình giao tiếp giữa người dùng và hệ thống chatbot, đầu vào của người dùng, đầu vào cũng phải được embedding để hệ thống có thể xử lý. Lúc này câu truy vấn đã được embedding này sẽ được dùng để truy xuất thông tin từ cơ sở dữ liệu vector database (7). Hệ thống RAG tìm kiếm thông tin phù hợp từ vector database bằng cách so sánh vector truy vấn của người dùng với các vector trong cơ sở dữ liệu (8). Từ đó các kết quả tương tự nhất sẽ được lấy ra (9) cho mục đích tạo câu trả lời.
* (10.1): Tiếp theo là thực hiện tạo ra các prompt tối ưu để định hướng đầu vào cho mô hình ngôn ngữ lớn (LLM). Các lời nhắc này được thiết kế sao cho LLM hiểu rõ yêu cầu của ngữ cảnh. Cùng với đó là việc thiết kế và tạo một hệ thống quản lý các agent. Agent giúp đảm bảo rằng mỗi yêu cầu của người dùng được chuyển đến công cụ xử lý phù hợp. Điều này bao gồm việc định tuyến yêu cầu tới mô hình LLM. Trên thực tế, sau thi hệ thông RAG thực hiện truy xuất thông tin từ câu hỏi người dùng, các thông tin sẽ được truyền qua trình tạo prompt (10.1) và trình quản lý agent (10.2). Thông tin prompt cũng được trình quản lý agent sử dụng để xử lý và quyết định sẽ dùng công cụ nào để gọi hàm (11).
* (12): Khi đã có đầy đủ thông tin, trình quản lý tác nhân sẽ thực hiện gọi tác nhân (agent call) để phân luồng truy vấn đến tác nhân phù hợp.
* (13.1, 13.2, 13.3): Đối với agent hỏi đáp, prompt sẽ được đưa vào mô hình LLM để thực hiện trả lời phản hồi dựa trên các thông tin tìm kiếm được, điều này bao gồm các thông tin lịch sử trò chuyện được tóm tắt trước đó. Sau đó câu trả lời của mô hình sẽ được đưa đến trình hiển thị phản hồi.
* (14.1, 14.2, 14.3, 14.4): Nếu agent ngoài phạm vi trả lời của chatbot, prompt tương ứng sẽ được đưa vào LLM để sinh ra phản hồi phù hợp, sau đó điều hướng người dùng tới cán bộ tuyển sinh nếu người dùng đồng ý. Tất cả những dữ liệu phản hồi này được đưa tới trình hiển thị phản hồi.
* (15, 16): Cuối cùng các đầu ra của các tác nhân này sẽ được truyền tới một bộ phân tích cú pháp để thực hiện phản hồi người dùng cuối cùng (15). Trong suốt quá trình trò chuyện, lịch sử sẽ được ghi lại để cải thiện phản hồi của chatbot và giữ lại được ngữ cảnh và tình huống đang giao tiếp. Đồng thời dữ liệu quan trọng được ghi vào cơ sở dữ liệu người dùng cho các cuộc trò chuyện sau này.
* (17,1 17.2): Trong toàn bộ quá trình xử lý hỏi đáp của hệ thống chatbot, các thông tin người dùng nhập vào sẽ được thu thập tự động bằng cách sử dụng LLM để trích xuất, dựa trên các yêu cầu truy vấn này, hệ thống sẽ biết được các thông tin mong muốn của người dùng (về ngành học, quan tâm tới học phí, các chính sách học bổng…), nhu cầu tìm hiểu về các lĩnh vực quan tâm khác. Những thông tin này sẽ được đưa vào LLM để hỗ trợ cho quá trình phản hồi tới người dùng, khi chatbot hiểu các yêu cầu và lĩnh vực mà người dùng quan tâm, chatbot sẽ trả lời các câu hỏi và gợi ý tìm hiểu các chủ đề xoay quanh các khía cạnh mà người dùng qua tâm. Từ đó giúp duy trì cuộc hội thoại lâu dài hơn và tăng thêm sự chủ động trong việc tìm hiểu nhiều hơn về trường cho người dùng. Các thông tin này sẽ được lưu trữ trong cơ sở dữ liệu MongoDB.

Trên đây là chi tiết toàn bộ quy trình trong hệ thống chatbot tuyển sinh. Trong quá trình trên đã ứng dụng các phương pháp hiện đại trong bối cảnh hiện tại. Đây là một quy trình chặt chẽ và có tính hệ thống cao. Bằng cách kết hợp công nghệ hiện đại và quy trình rõ ràng, phương pháp này giúp chatbot hoạt động hiệu quả, đáp ứng tốt nhu cầu người dùng và dễ dàng mở rộng trong tương lai và tích hợp với các hệ thống khác.

## Thiết kế hệ thống thông tin

Trong phần này, chúng tôi sẽ trình bày thiết kế hệ thống thông tin cho ứng dụng chatbot hỗ trợ hỏi đáp tuyển sinh. Đồ án tập trung vào việc phát triển một hệ thống chatbot có khả năng trả lời chính xác các câu hỏi liên quan đến tuyển sinh, bằng cách ứng dụng công nghệ trí tuệ nhân tạo và phương pháp thiết kế hiện đại. Mục tiêu là tạo ra một chatbot thông minh, không chỉ cung cấp thông tin đúng và kịp thời, mà còn mang lại trải nghiệm tương tác tự nhiên cho người dùng dựa trên dữ liệu đã được thu thập và tổ chức. Hệ thống được thiết kế với chức năng cốt lõi là hỏi đáp, nhưng để đạt được hiệu quả cao nhất, quy trình xây dựng đòi hỏi sự chuẩn bị kỹ lưỡng, thử nghiệm chặt chẽ, và đánh giá toàn diện. Dù chỉ đảm nhận một chức năng chính, hệ thống này là kết quả của việc tích hợp những nghiên cứu chuyên sâu và các công nghệ tiên tiến nhất hiện nay, nhằm đáp ứng nhu cầu thực tiễn trong lĩnh vực giáo dục.

### Chức năng hỏi đáp người dùng

Trọng tâm của chức năng hỏi đáp người dùng là cho phép người dùng đặt câu hỏi liên quan đến tuyển sinh và hệ thống chatbot sẽ dựa vào các thông tin được cung cấp để trả lời câu hỏi cho người dùng. Khi bắt đầu mỗi phiên trò chuyện, hệ thống sẽ yêu cầu người dùng nhập vào các thông tin cá nhân và lưu vào cơ sở dữ liệu MongoDB. Hình 23 dưới đây là sơ đồ mô tả chức năng hỏi đáp của người dùng.

A diagram of a flowchart

Description automatically generated

Hình 25. Biểu đồ chức năng hỏi đáp người dùng.

Đầu tiên khi bắt đầu một cuộc trò chuyện mới, hệ thống sẽ yêu cầu người dùng nhập vào thông tin cá nhân như họ tên, số điện thoại và địa chỉ email. Đầu vào này là một văn bản. Hệ thống sẽ kiểm tra những thông tin này bằng cách sử dụng LLM để kiếm tra người dùng đã nhập tên và các thông tin khác hay chưa. Nếu vẫn chưa thì chatbot sẽ tiếp tục gửi phản hồi lời mời nhập thông tin cá nhân để có thể bắt đầu đoạn chat. Nếu người dùng nhập vào thông tin thì những thông tin này sẽ được hệ thống trích xuất một cách tự động và đưa vào cơ sở dữ liệu MongoDB. Lúc này người dùng có thể nhập bất kỳ câu hỏi nào để hệ thống chatbot xử lý và phản hồi. Vòng lặp hỏi và đáp của người dùng và chatbot sẽ kết thúc nếu người dùng yêu cầu hoặc thoát hệ thống. Các lịch sử trò chuyện sẽ không được lưu vào hệ thống mà sẽ bị xóa đi sau khi phiên trò chuyện kết thúc.

### Biểu đồ use case chức năng hỏi đáp người dùng

Biểu đồ use case chức năng hỏi đáp (hình 24) gồm có các thành phần:

* Tác nhân: Ở đây là người dùng.
* Mô tả: Thực hiện hỏi đáp tư vấn.
* Điều kiện: Người dùng đã nhập thông tin cá nhân vào hệ thống chatbot.
* Luồng xử lý: Khi người dùng nhập câu hỏi cần tư vấn vào chatbox, hệ thống sẽ xem xét yêu cầu của câu hỏi/yêu cầu của người dùng có nằm trong phạm vi hỏi đáp của chatbot hay không. Nếu có thì sẽ trả lời câu hỏi theo dữ liệu được truy vấn từ vector database. Nếu không thì hệ thống chatbot sẽ xin lỗi người dùng và nhắc nhở người dùng rằng câu hỏi mới được nhập và không nằm trong phạm vi xử lý của chatbot.

A diagram of a flowchart

Description automatically generated

Hình 26. Biểu đồ use case chức năng hỏi đáp tuyển sinh.

### Biểu đồ tuần tự chức năng hỏi đáp người dùng

Biểu đồ tuần tự chức năng hỏi đáp người dùng biểu diễn tuần tự quá trình xử lý và phản hồi của hệ thống chatbot và sự tương tác của người dùng. Hệ thống gồm có 3 thành phần chính là người dùng, trình quản lý agent chatbot và các hàm xử lý, tool hành động. Hình 25 là biểu đồ tuần tự mô tả chức năng hỏi đáp người dùng.

Đầu tiên khi bắt đầu một cuộc trò chuyện mới, trình quản lý agent sẽ truyền đến người dùng và yêu cầu nhập vào thông tin cá nhân của người dùng (1) bao gồm họ và tên, số điện thoại và địa chỉ email. Người dùng nhập các thông tin vào hệ thống (2) sẽ được hệ thống kiểm tra độ phù hợp và luu vào hệ thống cơ sở dữ liệu người dùng. Sau đó chatbot sẽ thông báo với người dùng là đã ghi nhớ thông tin và bắt đầu cuộc trò chuyện (3).

Bắt đầu từ bước (4) là quá trình hoạt động chủ yếu của hệ thống chatbot, tại đây hệ thống hoạt động như một vòng lặp để duy trì cuộc hội thoại. Người dùng sẽ nhập vào các câu hỏi liên quan đến tư vấn tuyển sinh. Các câu hỏi này sẽ được trình quản lý agent nhận diện và điều hành (5), bằng cách sử dụng các công cụ và thành phần xử lý, kết quả cuối cùng nhận được là một câu phản hồi cho câu truy vấn đầu vào (6), lúc này hệ thống chatbot app sẽ hiển thị cho người dùng. Nếu người dùng tiếp tục nhập vào câu hỏi, hệ thống sẽ tiếp tục xử lý theo quy trình như bắt đầu từ bước (4).



Hình 27. Biểu đồ tuần tự chức năng hỏi đáp người dùng.

### Cơ sở dữ liệu người dùng

Trong hệ thống chatbot tuyển sinh, chúng tôi xây dựng một cơ sở dữ liệu đơn giản để lưu thông tin các người dùng hệ thống chatbot. Dưới đây là bảng mô tả bảng cơ sở dữ liệu người dùng (bảng 3):

Bảng 3. Bảng mô tả cơ sở dữ liệu người dùng trong hệ thống chatbot.

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| ID | String | ID của người dùng |
| NAME | String | Họ và tên người dùng |
| SDT | String | Số điện thoại người dùng |
| EMAIL | String | Địa chỉ email của người dùng |
| USER DESIRE | String | Lưu trữ thông tin thu thập từ người dùng |

## Thu thập và tạo dữ liệu

Dữ liệu tuyển sinh qua các năm thường thay đổi tùy theo cơ chế của từng trường. Việc thu thập dữ liệu chính xác là rất quan trọng, giúp mô hình phân tích đưa ra câu trả lời chính xác hơn thay vì chỉ mang tính chung chung. Trong nghiên cứu này, chúng tôi tập trung thu thập dữ liệu tuyển sinh của Trường Đại học Đông Á qua các năm, bao gồm thông tin chi tiết từng ngành học. Đồng thời thực hiện tạo dữ liệu kịch bản cho những câu hỏi mà sinh viên có thể sẽ quan tâm.

A diagram of a cylinder

Description automatically generated

Hình 28. Tạo dữ liệu tuyển sinh bằng cách thu thập và tạo kịch bản.

Cụ thể, đối với việc thu thập dữ liệu, chúng tôi nhắm đến dữ liệu bao gồm chỉ tiêu tuyển sinh, điểm chuẩn, tổ hợp xét tuyển, thông tin khối ngành xét tuyển và các phương thức xét tuyển như học bạ, điểm thi tốt nghiệp THPT, hoặc đánh giá năng lực. Ngoài ra những thông tin mà phụ huynh và học sinh thường quan tâm là chính sách học bổng và học phí, chúng tôi thu thập thông tin chính sách học bổng các năm, các thông tin và thông báo cũng được xem xét để xây dựng dữ liệu tham khảo đối chiếu. Ngoài ra, chương trình đào tạo ngành học và cơ hội việc làm cũng được quan tâm lớn. Chúng tôi thu thập các module chương trình đào tạo, nêu rõ các môn học cho mỗi ngành qua từng module, đồng thời thu thập những thông tin tổng quan về ngành để sinh viên có thể nắm rõ về ngành cụ thể mà sinh viên có sự quan tâm. Việc thu thập thông tin được lấy từ trang website chính thức của Trường Đại học Đông Á [132] để đảm bảo tính chính xác nhất. Những thông tin này không chỉ giúp chatbot trả lời đúng trọng tâm câu hỏi mong muốn mà còn đáp ứng vấn đề thời gian.

Ngoài việc thu thập dữ liệu trên trang website chính thức của trường, chúng tôi nhận thấy rằng sinh viên có xu hướng tìm hiểu các vấn đề liên quan đến việc nạp hồ sơ, cách đóng học phí, nhu cầu tìm phòng trọ cũng như mức sống tại thành phố Đà Nẵng. Vì thế chúng tôi quyết định sẽ tạo thêm dữ liệu kịch bản thiết thực nhất trong vai trò là một sinh viên mới muốn tìm hiểu về ngành học và trường. Chúng tôi xem xét lựa chọn và thu thập các câu hỏi của sinh viên trong nhóm Facebook tuyển sinh chính thức của Trường Đại học Đông Á năm 2024 [133] cùng với hệ thống website của trường tại [132] để tổng hợp dữ liệu bao quát nhất.

Nhìn chung trong cả hai trường hợp, quy tắc thu thập dữ theo một khuôn mẫu cố định bao gồm chủ đề quan tâm – nội dung bao quát hoặc câu hỏi - trả lời. Đối với các mẫu có số lượng từ lớn hoặc quá dài, chúng tôi thực hiện chia nhỏ dữ liệu theo từng đoạn theo phương pháp chunking. Sau khi thu thập và tạo dữ liệu, dữ liệu của chúng tôi có 736 mẫu cho tất cả các ngành học và các lĩnh vực dữ liệu liên quan đến tuyển sinh. Bảng 4 dưới đây là minh họa một vài ví dụ về mẫu dữ liệu được thu thập:

Bảng 4. Minh hoạ một số mẫu dữ liệu được thu thập và tạo.

|  |  |
| --- | --- |
| **Câu hỏi** | **Nội dung phản hồi thu thập được** |
| Khi học ngành thiết kế thời trang, sinh viên có cơ hội tham gia các dự án thực tế trong quá trình học không? | Có, nhà trường thường xuyên tổ chức các đồ án thực tế như thiết kế BST (bộ sưu tập) theo chủ đề và trình diễn thời trang. Những dự án này giúp sinh viên trải nghiệm từ việc lên ý tưởng, thiết kế, đến trình diễn sản phẩm. Ngoài ra, sinh viên còn có cơ hội tham gia các sự kiện thời trang của trường hoặc các cuộc thi thiết kế quốc gia và quốc tế, giúp phát triển kỹ năng và nâng cao hồ sơ cá nhân. |
| Vị trí việc làm của ngành Công nghệ thực phẩm? | Là ngành học khá phổ biến nhưng ít ai biết rằng cơ hội việc làm đối với ngành học này rất cao. Ứng dụng của ngành đa dạng và có tính thiết yếu trong cuộc sống. Sinh viên ngay khi ra trường có thể đảm nhận nhiều vị trí sau: - Kỹ sư chế biến, đảm bảo chất lượng, kiểm tra chất lượng sản phẩm, an toàn thực phẩm.  - Quản lý sản xuất, khởi nghiệp trong lĩnh vực thực phẩm, nghiên cứu R&D, nhân viên kiểm định chất lượng, nhân viên kiểm soát chất lượng nguyên liệu.  - Chuyên viên nghiên cứu và phát triển sản phẩm, chuyên viên dinh dưỡng, kỹ thuật viên sản xuất, nhân viên phòng thí nghiệm…  - Và nhiều công việc khác tại nhà máy, công ty thực phẩm như: công ty bia rượu, nước giải khát, công ty chế biến sữa và các thực phẩm từ sữa, công ty chế biến gia vị… |
| Học ngành Quản trị khách sạn là học về cái gì? | Quản trị khách sạn là ngành học dịch vụ, đào tạo chuyên sâu về các hình thức giải trí, nghỉ dưỡng, vui chơi, dịch vụ ăn uống. Nói một cách dễ hiểu, ngành Quản trị khách sạn quản lý, tổ chức, điều phối các hoạt động của khách sạn sao cho hiệu quả và hợp lý nhất. Bên cạnh đó là cách quản lý nhân sự về cả khối lượng công việc cần thực hiện đến yếu tố con người để đảm bảo hiệu quả cho doanh nghiệp, tập đoàn. |
| Học bạ THPT với giấy chứng nhận TN mình công chứng ở đâu ạ? | Em có thể phôtô học bạ và giấy chứng nhận tốt nghiệp đến trường THPT em đã học (nơi cấp) hoặc đến Phường/Xã (Bộ phận Tư pháp), các văn phòng công chứng để chứng thực bạn nhé! |
| Hồ sơ xét tuyển gồm những thủ tục nào? | Hồ sơ xét tuyển đại học Đông Á bao gồm: 1. Đối với thí sinh xét tuyển theo học bạ THPT - Đơn đăng ký xét tuyển (theo mẫu của nhà trường),  - Học bạ THPT/THPT(GDTX) (bản sao có chứng thực); - Bằng tốt nghiệp THPT/THPT(GDTX) (bản sao có chứng thực) hoặc chứng nhận tạm thời tốt nghiệp THPT - Lệ phí xét tuyển: 30.000 đồng. 2. Đối với thí sinh xét tuyển theo kết quả thi THPT QG - Giấy chứng nhận kết quả thi THPT Quốc gia (bản sao có chứng thực); - Giấy chứng nhận tạm thời tốt nghiệp THPT (bản sao có chứng thực); - Lệ phí xét tuyển: 30.000 đồng. |

## Tiền xử lý và phân đoạn dữ liệu

Sau khi hoàn thành thu thập dữ liệu tuyển sinh từ các nguồn tin cậy từ [132], [133], cần kiểm tra các mẫu dữ liệu để đảm bảo tính chính xác và đúng đắn. Hình 27 là các nhiệm vụ được thực hiện trong phần này.

A diagram of a diagram

Description automatically generated

Hình 29. Quy trình tiền xử lý và phân đoạn bộ dữ liệu.

Đầu tiên chúng tôi thực hiện loại bỏ các thông tin có thể gây nhiễu hoặc không cần thiết, làm sạch các lỗi chính tả hoặc các ký tự đặc biệt. Để đảm bảo chatbot có khả năng hiểu tốt hơn về ngữ cảnh, chúng tôi tiến hành xử lý các trường hợp có chữ viết tắt. Nắm bắt xu hướng học sinh thường có hay có thói quen viết tắt các từ khóa thông dụng, chính vì thế chúng tôi thực hiện tái tạo thêm mẫu dữ liệu với các câu trả lời có chữ viết tắt để tăng thêm hiệu quả trả lời của chatbot. Cụ thể chúng tôi tái tạo các câu trả lời với các từ khóa thông dụng như “Công nghệ thông tin” thành các từ khóa “CNTT” hoặc “cntt”, một ví dụ khác như từ khóa “Quản trị kinh doanh” sẽ được tái tạo thêm các mẫu như là “QTKD” hoặc “qtkd” và các từ khóa các ngành các nữa. Sau đó đối với mỗi mẫu dữ liệu, chúng tôi thực hiện kiểm tra lại văn phong câu trả lời và hiệu chỉnh để câu trả lời trôi chảy tự nhiên hơn. Cuối cùng là thực hiện gắn nhãn dữ liệu có cùng chủ đề với nhau, các câu hỏi có tính liên quan này sẽ được đề xuất trong suốt quá trình chatbot, điều này làm tăng kích thích người dùng tìm hiểu sâu hơn về một chủ đề người dùng quan tâm.

Sau khi làm sạch dữ liệu, chúng tôi tiến hành kiểm tra và xử lý các mẫu dữ liệu có độ dài vượt quá giới hạn mà các mô hình embedding và LLM có thể xử lý. Để khắc phục vấn đề này, các mẫu trả lời được phân đoạn lại nhằm đảm bảo phù hợp với kích thước yêu cầu, đồng thời hiệu chỉnh văn phong để dữ liệu trở nên mạch lạc và tự nhiên hơn. Đối với các câu trả lời dài và phức tạp, chúng tôi phân chia thành các đoạn ngắn hơn nhưng vẫn đảm bảo tính liên kết, đầy đủ ý nghĩa và giữ nguyên bối cảnh cần thiết.

Trong bước tiếp theo, dữ liệu đã qua xử lý được chuyển đổi thành định dạng embedding thông qua các mô hình tiên tiến. Đây là giai đoạn quan trọng nhằm mã hóa ý nghĩa ngữ nghĩa của dữ liệu, tạo điều kiện cho việc truy xuất thông tin hiệu quả và đáp ứng nhanh chóng. Việc xây dựng embedding không chỉ tối ưu hóa dung lượng lưu trữ mà còn cải thiện độ chính xác khi chatbot thực hiện tìm kiếm và đưa ra câu trả lời.

## Đánh giá các mô hình Embedding đề xuất

Hình 28 dưới đây mô tả quá trình đánh giá và so sánh hiệu suất của các mô hình embedding.

A diagram of a model embedding

Description automatically generated

Hình 30. Tổng quan quy trình đánh giá hiệu suất các mô hình embedding.

Việc so sánh nhiều mô hình embedding giúp mang lại góc nhìn toàn diện và tối ưu hóa việc lựa chọn mô hình phù hợp. Đầu tiên, việc đưa ra nhiều mô hình và đánh giá giúp hiểu rõ các ưu điểm, hạn chế, cũng như hiệu suất thực tế của từng mô hình trên các tiêu chí cụ thể như tốc độ, độ chính xác, và tài nguyên yêu cầu. Điều này đặc biệt quan trọng khi ứng dụng các mô hình trong bối cảnh cụ thể như hệ thống chatbot, nơi hiệu suất thực tế có thể thay đổi tùy thuộc vào dữ liệu hoặc nhu cầu. Hơn thế nữa, so sánh nhiều mô hình mang lại lợi ích về mặt kinh tế và chiến lược giúp chọn được giải pháp tối ưu về chi phí và hiệu năng, đồng thời giảm thiểu rủi ro khi triển khai​ [134], [135].

Dưới đây là 4 mô hình được cân nhắc lựa chọn chọn cho nhiệm vụ embedding dữ liệu, bao gồm:

* Mô hình *Sentence Transformers - all-MiniLM-L6-v2*: Được phát triển bởi nhóm nghiên cứu Sentence Transformers, dẫn đầu bởi Nils Reimers, trong cộng đồng Hugging Face. Được công bố vào năm 2020-2021 như một phần của bộ mô hình sử dụng MiniLM của Microsoft. Mô hình này được tối ưu hóa để mã hóa văn bản thành vector 384 chiều, hỗ trợ các tác vụ như tìm kiếm ngữ nghĩa (semantic search), cụm văn bản (clustering) và đo độ tương tự (sentence similarity) [136], [137]. Nó nổi bật vì sự cân bằng giữa hiệu suất và tốc độ xử lý. Dữ liệu huấn luyện trên hơn 1 tỷ cặp câu sử dụng kỹ thuật học tương phản (contrastive learning), kết hợp dữ liệu từ nhiều nguồn như Reddit, WikiAnswers, và Stack Exchange [137]. Mô hình này miễn phí và mã nguồn mở, đây là một mô hình nhẹ và rất phổ biến trong các ứng dụng tìm kiếm ngữ nghĩa và embedding văn bản
* Mô hình *Alibaba-NLP/gte-multilingual-base*: Ra đời bởi Alibaba NLP Team. Mô hình này được công bố vào khoảng năm 2022. Mục tiêu của nó là hướng tới việc xử lý ngôn ngữ đa ngữ (multilingual), mô hình này hỗ trợ embedding cho hơn 100 ngôn ngữ. Đây là một giải pháp mạnh mẽ cho các hệ thống toàn cầu, yêu cầu hỗ trợ nhiều ngôn ngữ trong các tác vụ như tìm kiếm đa ngữ hoặc dịch thuật. Ưu điểm tích hợp dễ dàng vào các pipeline đa ngữ, với hiệu suất cao trên nhiều ngữ cảnh [138]. Mô hình này miễn phí và mã nguồn mở, được cung cấp bởi nhóm Alibaba NLP. Mô hình này có thể tải về từ Hugging Face [139].
* Mô hình *BAAI/bge-m3*: Ra đời bởi Beijing Academy of Artificial Intelligence (BAAI), Trung Quốc. Được giới thiệu trong năm 2021. Mục tiêu là được thiết kế đặc biệt cho các tác vụ tìm kiếm văn bản và hiểu ngữ nghĩa sâu hơn trong văn bản lớn. Mô hình này hướng tới cải thiện độ chính xác của các tác vụ embedding thông qua việc sử dụng kỹ thuật học sâu (deep learning) với kiến trúc hiện đại. Dữ liệu huấn luyện được tận dụng kho dữ liệu phong phú từ các nguồn Trung Quốc và quốc tế, giúp hỗ trợ tốt cả ứng dụng đơn ngữ và đa ngữ. Mô hình này miễn phí do Beijing Academy of Artificial Intelligence phát hành trên nền tảng Hugging Face. Nó được sử dụng rộng rãi trong các tác vụ tìm kiếm và xử lý ngữ nghĩa [140], [141]​.
* Mô hình *nomic-ai/nomic-embed-text-v1.5*: Được hát hành vào năm 2024, mô hình này được đào tạo trình embedding bằng cách sử dụng một pipline đào tạo nhiều giai đoạn. Bắt đầu từ mô hình BERT ngữ cảnh dàitrên lượng dữ liệu lớn được tạo từ các cặp văn bản có liên quan yếu, các cặp tiêu đề - nội dung. Trong giai đoạn tinh chỉnh, các tập dữ liệu được gắn nhãn chất lượng cao hơn như các truy vấn tìm kiếm và trả lời từ các tìm kiếm web được tận dụng. Mô hình này hiện đang mở cho mục đích nghiên cứu và có thể tham khảo từ Hugging Face [142], [143].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bảng 5. So sánh và đánh giá hiệu suất của các mô hình embedding đề xuất. | **4Trung bình thời gian embed câu hỏi** | 0,005 | 0,010 | 0,021 | 0,012 |
| **Trung bình độ dài câu hỏi** | 14,681 | 166,484 | 9,734 | 14,681 |
| **Trung bình thời gian embed câu** | 0,013 | 0,056 | 0,024 | 0,014 |
| **Trung bình độ dài câu** | 389,149 | 1.744,195 | 373,787 | 398,149 |
| **Không gian GPU** | 0,109 | 1,466 | 2,725 | 0,656 |
| **Số lượng tham số** | 22.713.216 | 305.368.320 | 567.754.762 | 136.731.648 |
| **Độ dài bối cảnh** | 256 | 8.192 | 8.192 | 8.192 |
| **Số chiều** | 384 | 768 | 1.024 | 1.024 |
| **Top 10** | 0,468 | 0,872 | 0,851 | 0,574 |
| **Top 5** | 0,351 | 0,840 | 0,798 | 0,436 |
| **Top 3** | 0,234 | 0,755 | 0,787 | 0,351 |
| **Top 1** | 0,096 | 0,500 | 0,521 | 0,213 |
| **Tên mô hình** | Sentence-transformer/all-MiniLM-L6-v2 | Alibaba-NLP/gte-multilingual-base | BAAI/bge-m3 | nomic-ai/nomic-embed-text-v1.5 |

Trong bảng 3 so sánh hiệu suất tăng cường truy xuất tạo sinh (RAG) cho một số mô hình LLM nổi bật miễn phí. Trong đó, chỉ số Top (Top-K Accuracy) biểu thị tỷ lệ phần trăm các câu trả lời đúng xuất thành công xuất hiện trong danh sách kết quả đứng đầu (tương ứng với 1, 3, 5 hoặc 10 kết quả đầu tiên). Các giá trị này đánh giá khả năng truy xuất chính xác của mô hình. Mô hình có giá trị cao hơn có nghĩa là tốt hơn trong việc xếp hạng các kết quả phù hợp với câu hỏi truy vấn [134], [135].

Số chiều (Dimensions) là kích thước của vector embedding được tạo bởi mô hình. Ví dụ: 384, 768, hoặc 1.024 chiều. Vector có số chiều cao hơn thường nắm bắt được nhiều đặc trưng ngữ nghĩa hơn, nhưng cũng đòi hỏi nhiều bộ nhớ và tính toán hơn khi thực hiện so sánh vector [134], [135].

Độ dài bối cảnh (Context Length) là số token tối đa mà mô hình có thể xử lý trong một lần (context window). Mô hình có độ dài bối cảnh cao hơn có thể xử lý các câu dài hoặc đoạn văn lớn hơn, phù hợp cho các ứng dụng như tìm kiếm hoặc phân tích văn bản dài [134], [135].

Số lượng tham số (Parameters) là tổng số tham số trong mô hình, thường biểu thị kích thước của mô hình. Mô hình lớn hơn (số tham số cao) thường mạnh mẽ hơn trong việc nắm bắt các mối quan hệ ngữ nghĩa phức tạp, nhưng tốn nhiều tài nguyên tính toán hơn [134], [135].

Không gian GPU là lượng bố nhớ GPU chiếm dụng khi triển khai truy vấn.

Trung bình độ dài câu biểu thị độ dài trung bình của các câu hoặc đoạn văn được embed. Thường được tính bằng số token hoặc ký tự. Độ dài lớn hơn có thể yêu cầu mô hình thực hiện nhiều bước tính toán hơn, ảnh hưởng đến tốc độ và hiệu suất [134], [135].

Trung bình thời gian embed câu (Seconds) là thời gian trung bình để mã hóa một đoạn văn bản (hoặc câu) thành vector embedding. Giá trị này quan trọng trong các ứng dụng xử lý dữ liệu lớn, vì nó ảnh hưởng đến hiệu suất tổng thể khi xử lý hàng loạt dữ liệu [134], [135].

Trung bình độ dài câu hỏi biểu thị độ dài trung bình của các câu hỏi (số lượng token hoặc ký tự) được sử dụng để kiểm tra mô hình. Độ dài này ảnh hưởng đến tốc độ xử lý vì câu hỏi dài hơn yêu cầu nhiều tính toán hơn [134], [135].

Trung bình thời gian embedding câu hỏi (Seconds) là thời gian trung bình (tính bằng giây) để mã hóa một câu hỏi thành vector embedding. Mô hình có giá trị thấp hơn sẽ nhanh hơn trong việc xử lý các truy vấn, phù hợp cho các ứng dụng thời gian thực hoặc yêu cầu độ trễ thấp [134], [135].

Mỗi mô hình embedding đều có những thông số xử lý tùy thuộc vào từng dự án cụ thể và ưu nhược điểm riêng. Sau khi phân tích và so sánh các chỉ số hiệu suất trong bảng 3, chúng tôi quyết định chọn mô hình *nomic-ai/nomic-embed-text-v1.5* làm mô hình embedding chính vì những lý do sau:

* Hiệu suất Top-K: Mô hình *nomic-ai* cho thấy khả năng truy xuất chính xác vượt trội với giá trị Top-K Accuracy cao trong các mẫu thử nghiệm. Điều này chứng tỏ Nomic có hiệu quả trong việc xếp hạng các kết quả liên quan, phù hợp cho việc tăng cường truy xuất tạo sinh (RAG).
* Số chiều vector tối ưu: Mô hình cung cấp số chiều embedding cân bằng, vừa đủ lớn để nắm bắt đặc trưng ngữ nghĩa quan trọng nhưng không quá lớn để gây ra chi phí tính toán vượt mức. Điều này giúp giảm thiểu gánh nặng bộ nhớ và thời gian xử lý mà vẫn đảm bảo chất lượng embedding.
* Thời gian embedding: *nomic-ai* có thời gian trung bình để embedding câu và câu hỏi thấp hơn so với nhiều mô hình khác. Điều này rất quan trọng đối với các ứng dụng yêu cầu tốc độ xử lý nhanh, như chatbot thời gian thực hoặc hệ thống hỏi đáp.
* Khả năng xử lý bối cảnh dài: Với độ dài bối cảnh cao, cho phép tối đa khoảng 8,000 tokens, *nomic-ai* phù hợp để xử lý các đoạn văn bản dài và câu hỏi phức tạp, nâng cao khả năng phục vụ trong các trường hợp truy vấn ngữ nghĩa sâu.
* Tối ưu hóa tài nguyên GPU: Mô hình Nomic có mức chiếm dụng bộ nhớ GPU tối ưu hơn so với các mô hình lớn khác. Điều này giúp dễ dàng triển khai trên các hệ thống có tài nguyên phần cứng hạn chế mà không ảnh hưởng đến hiệu năng.
* Độ trễ thấp và hiệu quả xử lý hàng loạt: Khả năng mã hóa nhanh của *nomic-ai* giúp xử lý dữ liệu lớn trong thời gian ngắn, tối ưu hóa cho các kịch bản yêu cầu tốc độ như phân tích dữ liệu hàng loạt hoặc cung cấp phản hồi người dùng tức thì.
* Miễn phí thử nghiệm: Mô hình *nomic-ai/nomic-embed-text-v1.5* cung cấp lượng tokens thử nghiệm, điều này phù hợp để lựa chọn xây dựng và thử nghiệm cho đề tài chatbot có quy mô nhỏ hoặc vừa phải.

## Phương pháp đánh giá hiệu quả retrieval dữ liệu

Sau khi thực hiện embedding dữ liệu và lưu vào vector databse, các mô hình retrieval được sử dụng để đánh giá hiệu quả embedding, chất lượng dữ liệu và độ chính xác retrieval dữ liệu trước khi đưa vào mô hình LLM. Một phương pháp retrieval tốt cung cấp câu trả lời tốt. Trong dự án này chúng tôi sử dụng 3 phương pháp retrieval bao gồm retrieval theo vector, retrieval theo keyword và retireval theo hybrid. Trong hình 29 là mô tả quá trình thực hiện đánh giá hiệu suất retrieval.

A diagram of a model

Description automatically generated

Hình 31. Quy trình đánh giá hiệu suất retrieval dữ liệu.

Với Retrieval theo Vector, sử dụng các vector embedding để đại diện cho các đối tượng thông tin (như tài liệu, câu hỏi, hoặc câu trả lời). Đây là một kỹ thuật phổ biến trong các hệ thống tìm kiếm thông tin hiện đại, đặc biệt khi kết hợp với các mô hình học sâu. Bên cạnh đó, keyword retrieval là phương pháp truy xuất thông tin truyền thống hơn, nơi các truy vấn của người dùng được so khớp với các từ khóa trong cơ sở dữ liệu. Phương pháp này thường sử dụng các kỹ thuật tìm kiếm chuỗi, như tìm kiếm từ khóa chính xác hoặc biến thể từ khóa trong văn bản. Ngoài ra, phương pháp hybrid retrieval kết hợp cả retrieval theo vector và retrieval theo keyword để kết hợp những ưu điểm của cả hai phương pháp, từ đó cải thiện chất lượng và hiệu quả của việc tìm kiếm.

Theo đó chúng tôi sử dụng các chỉ số đánh giá hiệu quả retrieval bao gồm: chỉ số P@3, Recall@3, và MRR@3, đây là các chỉ số đánh giá phổ biến trong các hệ thống retrieval thông tin, đặc biệt là trong các hệ thống trả lời câu hỏi như chatbot. Mỗi chỉ số này giúp đo lường một khía cạnh khác nhau của hiệu suất tìm kiếm, đặc biệt là trong việc so sánh kết quả trả về và so khớp với các tài liệu hay câu trả lời đúng.

* Chỉ số P@3 (Precision at 3): Precision at 3 (P@3) đo lường độ chính xác của kết quả tìm kiếm dựa trên top 3 kết quả đầu tiên trả về từ hệ thống.

P@3 giúp đo lường khả năng của hệ thống trong việc trả về các kết quả chính xác ngay từ đầu (3 kết quả đầu tiên). Một giá trị P@3 cao cho thấy hệ thống có xu hướng đưa ra các kết quả có liên quan cao ngay lập tức.

* Chỉ số Recall@3: Recall at 3 (Recall@3) đo lường khả năng của hệ thống trong việc lấy lại tất cả các tài liệu hoặc câu trả lời đúng từ bộ dữ liệu, trong số 3 kết quả đầu tiên.

Recall@3 đo lường khả năng của hệ thống trong việc không bỏ sót các câu trả lời đúng trong số 3 kết quả đầu tiên. Một giá trị Recall cao nghĩa là hệ thống có khả năng tìm thấy phần lớn các kết quả đúng trong số ít các kết quả đầu tiên.

* Chỉ số MRR@3 (Mean Reciprocal Rank at 3): Mean Reciprocal Rank at 3 (MRR@3) đo lường mức độ tốc độ mà hệ thống trả về câu trả lời đúng trong top 3 kết quả đầu tiên. Đây là một chỉ số đánh giá xếp hạng cho biết vị trí của câu trả lời đúng trong danh sách trả về.

MRR@3 tập trung vào vị trí của câu trả lời đúng, với giá trị càng cao khi câu trả lời đúng xuất hiện càng sớm trong danh sách trả về. MRR đánh giá khả năng của hệ thống trong việc đưa câu trả lời đúng lên vị trí cao trong danh sách kết quả.

Dưới đây là bảng so sánh các chỉ số đánh giá retrieval của các phương pháp trên:

Bảng 6. Bảng so sánh hiệu suất các phương pháp retrieval.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **P@3** | **Recall@3** | **MRR@3** |
| VECTOR | 0,173 | 0,52 | 0,363 |
| KEYWORD | 0,197 | 0,59 | 0,477 |
| HYBRID | 0,213 | 0,64 | 0,44 |

## Các mô hình LLM được sử dụng

Sự phát triển mạnh mẽ của các mô hình LLM trong thời gian gần đây trong cuộc đua AI đã tạo nên sự đa dạng mô hình LLM, khi ngày càng có nhiều mô hình được ra mắt với khả năng không ngừng cải tiến về độ chính xác, tốc độ và khả năng xử lý ngữ cảnh [144], [145]. Các LLM hiện đại không chỉ tập trung vào việc sinh văn bản mà còn mở rộng ứng dụng vào các lĩnh vực như tìm kiếm ngữ nghĩa (semantic search), lập luận logic, và hỗ trợ đa ngôn ngữ, đáp ứng nhu cầu ngày càng tăng trong nghiên cứu và công nghiệp [144], [145]. Các mô hình LLM mạnh mẽ hầu hết được phát triển cho mục đích công nghiệp, tuy nhiên vẫn có những mô hình LLM được phát hành miễn phí dành cho các nhà phát triển thử nghiệm và nghiên cứu [144], [145], [146]. Dưới đây là một vài mô hình LLM mở được cân nhắc sử dụng trong báo cáo này:

#### Mô hình LLM Llama 3

Llama 3 là một mô hình ngôn ngữ lớn (LLM) mới của Meta, tiếp nối thành công của Llama 2. Nó được thiết kế với cấu trúc Transformer tối ưu và có sẵn dưới dạng hai kích thước: 8 tỷ và 70 tỷ tham số. Llama 3 được huấn luyện trên tập dữ liệu lớn gấp bảy lần so với Llama 2, bao gồm hơn 15 nghìn tỷ token, giúp tăng khả năng xử lý ngôn ngữ và giảm tỉ lệ từ chối câu trả lời sai. Mô hình này hỗ trợ ngữ cảnh dài tới 128.000 token, vượt trội so với các mô hình trước đó. Trong các thử nghiệm, Llama 3 có hiệu suất tốt hơn nhiều đối thủ như Gemini 1.5 và Claude trên các chỉ số hiệu năng quan trọng. Mô hình này cũng hỗ trợ các ứng dụng sử dụng LangChain và LlamaIndex để tích hợp vào các hệ thống chatbot hoặc AI khác

#### Mô hình LLM Gemini 1.5

Gemini 1.5 là phiên bản nâng cấp của dòng Gemini do Google phát triển. Mô hình này tập trung vào khả năng sinh ngữ cảnh chi tiết và hỗ trợ đa ngôn ngữ. Với các cải tiến về huấn luyện và xử lý token, Gemini 1.5 đã tăng độ chính xác và tốc độ xử lý trong các tác vụ như dịch thuật và hội thoại. Dù không đạt được mức hiệu suất của Llama 3 trên một số chỉ số, Gemini 1.5 vẫn là lựa chọn mạnh mẽ cho các ứng dụng AI tập trung vào xử lý hội thoại tự nhiên và ngôn ngữ toàn cầu​

#### Mô hình LLM Grok

Grok là mô hình AI do X phát triển, được tích hợp vào nền tảng mạng xã hội X để hỗ trợ viết bài đăng và tương tác với người dùng. Đây là một LLM hướng đến khả năng tạo nội dung ngắn gọn, súc tích, với trọng tâm là tối ưu trải nghiệm người dùng trong các tác vụ hội thoại đơn giản và viết nội dung truyền thông. Mặc dù Grok không được thiết kế để xử lý các tác vụ phức tạp như các mô hình lớn khác, nó vẫn đóng vai trò quan trọng trong các ứng dụng hỗ trợ người dùng trực tiếp.

Việc lựa chọn nhiều LLM để thực hiện xây dựng chatbot là vì để tận dụng tối đa những gì miễn phí mà các mô hình này có như tài nguyên tính toán, lượng truy vấn giới hạn của mỗi mô hình, ngoài ra khi thực hiện một nhiệm vụ được yêu cầu từ người dùng, việc phân chia các nhiệm vụ cho các mô hình này thực hiện cùng lúc sẽ làm giảm thời gian trong quá trình xử lý, do đó tăng hiệu suất trả lời và giảm độ trễ phản hồi của chatbot.

## Prompt engineering trong hệ thống chatbot

Prompt Engineering (lời nhắc) đóng vai trò cốt lõi trong việc giao tiếp với mô hình LLM. Giúp định hướng mô hình xử lý thông tin chính xác, hiểu ngữ cảnh của câu hỏi từ người dùng và tạo ra câu trả lời phù hợp với kỳ vọng. Việc xây dựng prompt tốt giúp tăng độ chính xác, đảm bảo mô hình LLM hiểu được mục tiêu câu hỏi. Giảm sai lệch, tránh việc mô hình đưa ra các phản hồi không liên quan hoặc không chính xác. Tối ưu hóa trải nghiệm, đưa ra câu trả lời đầy đủ, dễ hiểu, và ngắn gọn. Dưới đây là quy trình triển khai prompt cho hệ thống. Cấu trúc của một prompt trong hệ thống chatbot hỏi đáp tuyển sinh này bao gồm các thành phần sau (hình 30):

A screenshot of a phone

Description automatically generated

Hình 32. Cấu trúc prompt trong hệ thống chatbot hỏi đáp tuyển sinh.

Trong đó:

* Vai trò: Xác định vai trò hoặc vị trí mà hệ thống AI cần đóng. Mục tiêu là định hướng hệ thống vào một cách tư duy hoặc phong cách phản hồi cụ thể.
* Bối cảnh: Cung cấp thông tin cần thiết để AI hiểu tình huống hoặc nội dung liên quan.
* Câu hỏi: Câu hỏi hoặc yêu cầu chính mà hệ thống giải quyết.
* Các ví dụ: Là các bản hướng dẫn, cung cấp các ví dụ minh họa để hệ thống có thể hiểu rõ kỳ vọng về đầu ra.
* Hướng dẫn: Đưa ra hướng dẫn chi tiết về cách hệ thống nên trả lời
* Lưu ý: Các lưu ý quan trọng giúp chatbot có thể tránh các điều cần tránh trả lời và các điều mà chatbot nên làm, ví dụ như prompt yêu cầu chatbot hạn chế đưa ra các từ toxic.

Prompt được cấu trúc thành một chuỗi văn bản đầy đủ bao gồm các thành phần chính bao gồm ngữ cảnh chứa nội dung liên quan từ cơ sở dữ liệu (ví dụ: thông tin truy vấn, dữ liệu lịch sử của người dùng) và câu hỏi của người dùng.

Việc xây dựng prompt diễn ra tại bước (9.1), sau khi thông tin được trích xuất từ cơ sở dữ liệu vector (8). Quy trình cụ thể bao gồm:

Thu thập thông tin từ truy vấn:

* Dữ liệu từ người dùng được gửi qua bước trích xuất thông tin (7, 8) để lấy ra các đoạn thông tin phù hợp từ cơ sở dữ liệu vector.
* Các đoạn thông tin này đóng vai trò cung cấp ngữ cảnh cho prompt.

Prompt sau khi được định dạng sẽ được gửi đến mô hình LLM để tạo phản hồi. Tạo điều kiện để LLM xử lý các truy vấn có liên quan trực tiếp đến dữ liệu được lưu trữ. Ngoài ra, tùy chỉnh lời nhắc theo loại tác nhân để phản hồi phù hợp để tối ưu hóa chất lượng câu trả lời, giảm thiểu các lỗi như câu trả lời quá ngắn, lan man, hoặc không chính xác.

## Tác nhân (Agents)

Tác nhân (agent) là các thành phần trung gian, hoạt động độc lập hoặc phối hợp với LLM để giải quyết các tác vụ cụ thể. Trong hệ thống này, các tác nhân giúp chia nhỏ công việc của chatbot thành các nhiệm vụ, đảm bảo chatbot có thể xử lý đa dạng các yêu cầu từ người dùng, tăng khả năng mở rộng của hệ thống bằng cách dễ dàng thêm mới các tác nhân với chức năng đặc thù. Hệ thống chatbot tuyển sinh triển khai ba loại tác nhân chính như sau: Tác nhân hỏi đáp, tác nhân điều hướng người dùng cho các truy vấn ngoài phạm vi.

### Tác nhân hỏi đáp (Q&A Agent)

Hình 31 dưới đây mô tả luồng hoạt động của một Q&A agent từ đầu đến cuối.

A diagram of a company

Description automatically generated

Hình 33. Mô tả luồng hoạt động của Q&A agent.

Question & Answering Agent (Tác nhân Hỏi & Đáp – Q&A agent) là thành phần quan trọng nhất trong hệ thống, chịu trách nhiệm xử lý và trả lời hầu hết các câu hỏi của người dùng. Phạm vi câu hỏi được giới hạn trong dữ liệu tuyển sinh đã được thu thập và lưu trữ trong cơ sở dữ liệu vector, đảm bảo các câu trả lời luôn bám sát thông tin đã được xác minh.

* Mô tả chức năng của Q&A agent: Trả lời các câu hỏi thường gặp và những câu hỏi liên quan đến thông tin tuyển sinh trong cơ sở dữ liệu vector. Đảm bảo phản hồi chính xác và nhanh chóng bằng cách tận dụng cơ sở dữ liệu vector và sức mạnh xử lý ngôn ngữ tự nhiên của mô hình LLM. Cung cấp câu trả lời trực tiếp, ngắn gọn nhưng đầy đủ, nhằm tối ưu trải nghiệm người dùng.
* Quy trình hoạt động: Khi hệ thống phát hiện truy vấn của người dùng nằm trong phạm vi thông tin đã được lưu trữ, Q&A agent sẽ được agent manager điều phối. Cơ chế nhận diện dựa trên việc so khớp ngữ nghĩa của truy vấn với nội dung trong cơ sở dữ liệu vector. Hệ thống tìm kiếm và lấy ngữ cảnh liên quan từ cơ sở dữ liệu vector. Thông tin này sau đó được chuyển tiếp đến mô hình LLM thông qua cơ chế lời nhắc (prompting) để hỗ trợ tạo câu trả lời chính xác và phù hợp
* Tạo phản hồi: Sau khi xử lý thông tin, mô hình LLM sẽ đưa ra phản hồi trực tiếp dựa trên dữ liệu ngữ cảnh đã được cung cấp. Câu trả lời cuối cùng được hệ thống gửi lại cho người dùng thông qua giao diện tương tác.

### Tác nhân ngoài phạm vi (Out-of-Domain Agent)

Tác nhân trả lời câu hỏi ngoài phạm vi được thiết kế nhằm đảm bảo trải nghiệm người dùng không bị gián đoạn ngay cả khi chatbot không thể trực tiếp trả lời câu hỏi. Nó hoạt động như một cầu nối, hỗ trợ người dùng tiếp cận thông tin đúng cách hoặc kết nối với nguồn hỗ trợ phù hợp. Hình 26 dưới đây mô tả luồng hoạt động của một out\_of\_domain agent (OOD agent).

A diagram of a process

Description automatically generated

Hình 34. Mô tả luồng hoạt động của OOD agent.

* Mô tả: Xử lý các yêu cầu không thuộc phạm vi kiến thức hiện tại của chatbot. Khi truy vấn không phù hợp, hệ thống ngay chuyển quyền xử lý cho tác nhân này. Trong trường hợp cần thiết, tác nhân này sẽ chuyển yêu cầu đến các cán bộ hoặc nhân viên có chuyên môn để đảm bảo người dùng nhận được hỗ trợ đầy đủ.
* Quy trình hoạt động: Dựa trên cơ chế nhận diện ngữ nghĩa, tác nhân xác định rằng truy vấn của người dùng không khớp với dữ liệu hiện có trong cơ sở dữ liệu vector. Hệ thống sẽ tự động yêu cầu người dùng cung cấp thêm thông tin để làm rõ truy vấn, hoặc điều hướng tới cán bộ tư vấn trực tiếp. Những phản hồi này được LLM đưa ra thông qua cơ chế prompt.
* Tạo phản hồi: Hệ thống sẽ nhận phản hồi từ mô hình LLM và cung cấp cho agent manager để hiển thị đầu ra cho người dùng.

### Quản lý tác nhân (Agent Management)

Vai trò: Điều phối các tác nhân trong hệ thống để đảm bảo chọn đúng tác nhân dựa trên loại truy vấn. Quản lý giao tiếp giữa tác nhân, cơ sở dữ liệu và mô hình LLM.

Quy trình:

* Phân loại truy vấn sau khi người dùng nhập câu hỏi có nằm trong phạm vi trả lời của chatbot hay không.
* Kích hoạt tác nhân phù hợp thông qua Agent Call (11, 12.x).
* Gửi thông tin trả về từ tác nhân đến bước xử lý tiếp theo.

Thông qua việc xây dựng lời nhắc và tác nhân cho hệ thống chatbot, sự phối hợp giữa kỹ thuật tạo lời nhắc và các tác nhân giúp hệ thống chatbot trở nên mạnh mẽ, linh hoạt và dễ mở rộng. Trong đó Prompt Engineering tối ưu hóa tương tác giữa người dùng và LLM. Các tác nhân đảm bảo hệ thống có thể xử lý nhiều loại nhiệm vụ khác nhau, từ việc hỏi đáp đơn giản đến điều hướng và xử lý các yêu cầu phức tạp. Việc triển khai kỹ thuật này không chỉ nâng cao độ chính xác mà còn tăng cường trải nghiệm người dùng và khả năng ứng dụng của hệ thống chatbot trong các lĩnh vực khác nhau.

## Cá nhân hóa người dùng

Một trong những thành phần quan trọng nhất trong hệ thống chatbot là làm thế nào để giúp chatbot hiểu người dùng, từ đó chatbot sẽ đưa ra những câu trả lấy người dùng làm trung tâm. Điều này tạo nên trải nghiệm tương tác thân thiện, đồng thời mang lại nhiều lợi ích hơn trong việc tư vấn tuyển sinh, như:

* Tăng cường trải nghiệm người dùng: Khi chatbot hiểu rõ nhu cầu, sở thích, và ngữ cảnh của từng cá nhân, câu trả lời sẽ trở nên phù hợp hơn. Điều này giúp người dùng cảm thấy được quan tâm và hài lòng khi sử dụng hệ thống.
* Tăng khả năng tương tác lâu dài: Chatbot được cá nhân hóa có khả năng tạo ra các cuộc hội thoại thú vị và gần gũi, từ đó khuyến khích người dùng quay lại và tương tác nhiều hơn. Điều này đặc biệt quan trọng đối với các sinh viên tiềm năng có nhu cầu tìm hiểu về trường.
* Đưa ra các câu hỏi gợi ý cá nhân hóa: Đối với các nhiệm vụ tư vấn tuyển sinh, việc chatbot được cá nhân hóa cho người dùng có thể gợi ý các câu hỏi, lĩnh vực phù hợp dựa trên sở thích và nhu cầu tìm hiểu, từ đó tăng khả năng tin tưởng và tạo thiện cảm với người dùng.

A diagram of a flowchart

Description automatically generated

Hình 35. Quy trình thu thập thông tin và cá nhân hóa người dùng.

Hình 33 mô tả quy trình thu thập thông tin truy vấn đầu vào và thực hiện cá nhân hóa người dùng dựa trên những thông tin thu thạp được. Đầu tiên người dùng sẽ nhập vào các thông tin muốn tìm hiểu.Các thông tin này sẽ được trích xuất thông minh để nhận diện các chủ đề, lĩnh vực hoặc khía cạnh mà người dùng quan tâm và sau đó lưu vào cơ sở dữ liệu MongoDB, đồng thời đưa tới LLM, kết hợp với các thông tin truy xuất cho mục đích trả lời, từ đó chatbot sẽ đưa ra câu trả lời được cá nhân hóa cho người dùng hiện tại.

# TRIỂN KHAI HỆ THỐNG CHATBOT TUYỂN SINH

## Triển khai embedding dữ liệu và truy vấn người dùng

Sau khi dữ liệu được tiền xử lý và chuẩn hóa, hệ thống sử dụng mô hình embedding để chuyển đổi dữ liệu này thành các vector số, giúp biểu diễn thông tin theo dạng có thể tính toán. Đồng thời, câu truy vấn của người dùng cũng được chuyển đổi thành vector tương tự bằng cách sử dụng cùng mô hình embedding, giúp dễ dàng so sánh và tìm kiếm thông tin liên quan trong cơ sở dữ liệu. Hình 34 là quá trình triển khai embedding dữ liệu và câu truy vấn người dùng sử dụng mô hình embedding *nomic-ai/nomic-embed-text-v1.5*:

A diagram of a question

Description automatically generated with medium confidence

Hình 36. Quá trình triển khai embedding bộ dữ liệu sử dụng mô hình embedding.

Đầu tiên dữ liệu được chuẩn hóa sẽ được xử lý bằng chương trình Python, sau khi định dạng phù hợp thì đưa vào mô hình embedding, tại đây mô hình sẽ tự động thực hiện embedding từ văn bản sang svector embedding. Các vector embedding này sẽ được sử dụng để đưa vào lưu trữ trong Milvus.

## Triển khai lưu trữ Vector embedding vào Vector database

Vector embeddings được sinh ra từ dữ liệu câu hỏi và câu trả lời bằng mô hình embedding, sau đó được tổ chức và lưu trữ trong một vector database sử dụng Milvus, với ưu điểm dễ sử dụng và khả năng quản lý dữ liệu hiệu quả, là lựa chọn phù hợp cho dự án nhỏ và trung bình như chatbot tuyển sinh. Bước lưu trữ này thường được thực hiện sau khi đã embedding dữ liệu từ mô hình, nhằm chuẩn bị cho các hoạt động tìm kiếm thông tin hoặc truy vấn tương tự (vector search). Đây là nền tảng để chatbot có thể tìm kiếm và phản hồi chính xác hơn. Các bước thực hiện lưu vector vào database Milvus thông thường bao gồm các bước như sau:

* Cài đặt các thư viện cần thiết.
* Khởi chạy Milvus trên máy tính cục bộ bằng Docker:
* Sử dụng pymilvus để kết nối tới Milvus:
* Chuẩn bị dữ liệu embeddings: Sau khi đã thực hiện embedding từ mô hình, chuẩn bị dữ liệu theo định dạng phù hợp:
* Lưu embeddings vào Milvus: Sử dụng Milvus client để gửi dữ liệu vector vào collection.

Hình 35 là quá trình thực hiện xử lý và lưu vector embedding vào vector database bằng Milvus.

A diagram of a software development process

Description automatically generated

Hình 37. Quá trình lưu trữ vector embedding vào vector database.

Tại bước này, chúng tôi định dạng metadata có cấu trúc như sau:

*# Định dạng metadata để lưu vào vector database*

metadata = [

    {

        "id": 1,

        "question": "Khi nào bắt đầu tuyển sinh 2025?",

        "answer": "Tháng 6 năm 2025",

*# Embeddings được tạo từ mô hình embedding*

        "embedding": [0.1, 0.2, ...]

    },

    {

        "id": 2,

        "question": "Chỉ tiêu tuyển sinh năm 2025?",

        "answer": "10,000 sinh viên",

        "embedding": [0.4, 0.5, ...]

    },

]

Id định danh cho mẫu dữ liệu, số lượng mẫu trong data được chuẩn hóa tương ứng với số Id trong vector database, meta bao gồm câu hỏi, câu trả lời và vector embedding đã được chuẩn hóa trước đó.

## Retrieval cho truy vấn người dùng

Việc retrieval cho một truy vấn người dùng được nhập vào trong hệ thống chatbot được thực hiện sau khi embedding dữ liệu thành vector và lưu trữ trong vector database Milvus. Hệ thống sẽ tìm kiếm dựa trên vector để liên kết các câu hỏi của người dùng với các thông tin đã được lưu trữ. Phương pháp kết hợp embedding với tìm kiếm vector giúp nâng cao khả năng tìm kiếm theo ngữ cảnh, cho phép chatbot truy xuất các kết quả sát nghĩa nhất với câu truy vấn, đồng thời tối ưu hóa thời gian phản hồi. Trong phần này, chúng tôi trình bày cách thực hiện truy vấn từ cơ sở dữ liệu Milvus bằng cách sử dụng embedding của câu truy vấn, sau đó bằng cơ chế tìm kiếm vector của Milvus để trả về các câu trả lời phù hợp nhất. Hình 36 dưới đây là quy trình thực hiện truy xuất dữ liệu từ đầu vào của người dùng, bao gồm embedding truy vấn từ vector database Milvus.

A diagram of a process

Description automatically generated

Hình 38. Các bước thực hiện retrieval cho truy vấn người dùng.

Đầu tiên, người dùng nhập vào một câu truy vấn, câu hỏi sẽ được embedding bằng mô hình nomic-ai/nomic-embed-text-v1.5 embedding (1). Sau đó chương trình sẽ dùng vector embediding này (2), bằng thuật toán retrieval để tìm kiếm dữ liệu liên quan trong vector database (3). Sau đó trả về các kết quả truy vấn được (4) theo chỉ định, trong dự án của này chúng tôi chỉ định trả về 5 kết quả liên quan nhất trong vector database (5). Các kết quả này bao gồm một điểm số score biểu thị mức độ liên quan nhất được sắp xếp theo thứ tự liên quan nhất. Các kết quả này sẽ được dùng để tạo câu trả lời cho sau này.

## Triển khai agent cho hệ thống chatbot

Agent đóng vai trò quan trọng trong hệ thống chatbot hỏi đáp tuyển sinh. Nó giúp hệ thống có khả năng thực hiện các hành động nhất định, như giao tiếp với người dùng, xử lý thông tin, truy vấn dữ liệu, và phản hồi. Một agent có thể sử dụng prompt để giao tiếp với mô hình AI. Agent sẽ chuẩn bị prompt bao gồm các câu hỏi, thông điệp hoặc yêu cầu và gửi nó đến mô hình LLM để nhận được phản hồi. Trong trường hợp này, prompt là thành phần mà agent sử dụng để tương tác với mô hình. Trong phần này chúng tôi sẽ trình bày các thành phần của agent cho hệ thống chatbot hỏi đáp và cách triển khai cho từng thành phần của agent.

### Xây dựng cấu trúc Prompt

Prompt được xây dựng dựa trên thiết kế trong hình 30 của phương pháp đề xuất. Đây là thành phần quan trọng giúp định hướng cách thức mà LLM sẽ trả lời câu hỏi của người dùng, đặc biệt chất lượng phản hồi của chatbot phụ thuộc phần lớn vào prompt được xây dựng. Khi các kết quả tìm kiếm từ cơ sở dữ liệu được truy xuất dựa trên truy vấn của người dùng đã được thu thập và xếp hạng, chúng sẽ được sử dụng để tạo ra prompt cho mô hình. Prompt có thể được xây dựng bằng cách kết hợp câu hỏi người dùng và các thông tin từ kết quả tìm kiếm, nhằm cung cấp ngữ cảnh rõ ràng và chính xác hơn cho mô hình LLM. Thông qua việc sử dụng các kết quả tìm kiếm này, hệ thống có thể cung cấp câu trả lời phù hợp hơn, dựa trên kiến thức có sẵn từ cơ sở dữ liệu. Quy trình xây dựng prompt cho hệ thống chatbot tuyển sinh như sau:

* Nhắc nhở cho LLM biết nó là ai và vai trò của nó hiện tại, điều này giúp chatbot có thể xưng hô với người dùng dựa theo tên được cung cấp. Ví dụ như cho chatbot biết rằng nó là UDAChat, nhiệm vụ của nó là sử dụng các thông tin quan trọng để cung cấp cho người dùng…
* Cung cấp một tóm tắt bối cảnh dành cho chatbot bao gồm cung cấp thông tin bối cảnh hiện tại, cung cấp lời nhắc rằng chatbot sẽ làm những nhiệm vụ mà chatbot có thể làm. Ví dụ như nhiệm vụ của chatbot là sẽ trả lời các câu hỏi tư vấn tuyển sinh, các chủ đề bao gồm…
* Cung cấp cho chatbot câu hỏi mà người dùng cần tư vấn do người dùng nhập.
* Nhận dạng kết quả từ retrieval và rerank: Sau khi thực hiện truy xuất retrieval và có kết quả, cần lọc ra các văn bản có liên quan nhất từ cơ sở dữ liệu hoặc vector database. Các kết quả này là các câu trả lời từ cơ sở dữ liệu mà chatbot có thể sử dụng để cung cấp thông tin cho người dùng. Các thông tin này sẽ được đưa vào prompt kết hợp với câu hỏi do người dùng nhập vào.
* Cung cấp cho chatbot những hướng dẫn cần thiết và các lưu ý để chatbot có thể trả lời một cách hiệu quả và tốt nhất.

### Triển khai prompt hỏi đáp

Prompt hỏi đáp có nhiệm vụ cung cấp cho LLM thông tin để trả lời câu hỏi tư vấn tuyển sinh. Dựa trên cấu trúc prompt được đề xuất trong dự án này. Prompt hỏi đáp cũng sẽ bao gồm các thành phần cơ bản của prompt. Dưới đây là một prompt cho tác nhân hỏi đáp hệ thống chatbot tuyển sinh:

Bạn là UDAChat.

{human\_preference}. Sử dụng những thông tin quan trọng này để tăng tương tác cũng như mang đến thông tin cần thiết và hữu ích cho người dùng.

Nhiệm vụ của bạn là trả lời các câu hỏi về thông tin tuyển sinh và thông tin chung của trường Đại học Đông Á như ngành học, học phí, chính sách học bổng, môi trường học tập, hồ sơ đăng kí xét tuyển, giấy tờ hồ sơ cần thiết, điểm chuẩn xét học bạ, điểm chuẩn xét tuyển thi...

Số điện thoại hotline: *\*0236.351.9929\**

<Câu hỏi:>

<Thông tin:>

Ví dụ:

Các thông tin bạn sẽ phản hồi về tuyển sinh và về trường Đại học Đông Á

Lưu ý:

- Bạn nên xưng hô là `mình` cho tất cả phản hồi của bạn.

- IMPORTANT: Your answer should be in **\*\*Vietnamese\*\***.

- Hãy chọn lọc cẩn thận những *\*nội dung được trích xuất\** để trả lời yêu cầu của người dùng một các phù hợp.

Trong đoạn prompt trên, chúng tôi tạo một lời nhắc cho biết vai trò và nhiệm vụ, đồng thời cung cấp thông tin số điện thoại của ban tuyển sinh (lấy số máy của Trường Đại học Đông Á làm ví dụ). Tiếp đó, prompt sẽ được cung cấp câu hỏi của người dùng và các thông tin truy xuất được. Cuồi cùng là cung cấp những ví dụ và các nhắc nhở lưu ý để LLM có thể trả lời tốt hơn. Prompt cho nhiệm vụ hỏi đáp này có thể tùy chỉnh, lợi ích của việc tạo prompt tùy chỉnh có thể dễ dàng linh hoạt nâng cấp, cải thiện và sửa đổi sau này.

Quy trình sử dụng prompt hỏi đáp câu hỏi tuyển sinh được xử lý bởi hầu hết các thành phần trong hệ thống. Hình 37 dưới đây là cách mà prompt được sử dụng cho nhiệm vụ hỏi đáp.

A diagram of a process

Description automatically generated

Hình 39. Cách một prompt được hoàn chỉnh và sử dụng trong hệ thống chatbot.

Trong đó prompt template là một đoạn prompt được tạo từ trước, sau đó prompt này được cung cấp thêm thông tin nhờ kết quả truy xuất từ cơ sở dữ liệu để tạo thành prompt hoàn chỉnh. Lúc này prompt hoàn chỉnh sẽ được đưa vào LLM để sinh ra câu hỏi. Câu trả lời sẽ được hệ thống chatbot xử lý và hiển thị cho người dùng.

### Triển khai prompt điều hướng người dùng

Quy trình triển khai prompt nhắc nhở người dùng yêu cầu nằm ngoài phạm vi và thực hiện điều hướng người dùng tương tự với quy trình triển khai prompt trả lời câu hỏi tuyển sinh, đó là dựa theo cấu trúc cơ bản prompt được đê cập trong phương pháp đề xuất. Điểm khác là prompt điều hướng người dùng cần phải cho chatbot biết xin lỗi người dùng vì câu hỏi không nằm trong phạm vi của chatbot. Ngoài ra trong prompt điều hướng người dùng sẽ không bao gồm các thông tin được truy xuất từ cơ sở dữ liệu và cách sử dụng prompt điều hướng người dùng cũng sẽ khác với prompt trả lời câu hỏi tuyển sinh. Dưới đây là prompt điều hướng người dùng khi quy vấn ngoài phạm vi của chatbot tuyển sinh:

Bạn là UDAChat.

{human\_preference} một trợ lý ảo chuyên hỗ trợ thông tin tuyển sinh và thông tin chung về trường Đại học Đông Á.

Nhiệm vụ chính của bạn bao gồm: Trả lời các câu hỏi về thông tin tuyển sinh: ngành học, học phí, môi trường học tập, hồ sơ đăng ký xét tuyển, giấy tờ cần thiết, điểm chuẩn xét học bạ, điểm chuẩn xét tuyển thi, v.v.

Tuy nhiên người dùng đã nhập vào một thông tin không liên quan đến tuyển sinh, bạn cần nhắc người dùng rằng bạn chỉ hỗ trợ thông tin tuyển sinh. Lịch sự giải thích rằng bạn không thể hỗ trợ chính xác câu hỏi này.

Gợi ý liên hệ với đội ngũ chuyên trách hoặc cung cấp nguồn thông tin phù hợp.

Đảm bảo giữ tương tác thân thiện và hỗ trợ người dùng tiếp cận thông tin cần thiết.

- Số điện thoại hotline: *\*0236.351.9929\**

Lưu ý:

  - Bạn nên xưng hô là `mình` trong tất cả phản hồi của bạn.

  - IMPORTANT: Your answer should be in **\*\*Vietnamese\*\***.

  - Các thông tin nằm trong ngữ cảnh trò chuyện bạn hãy phản hồi một các tự nhiên và phù hợp nhất.

  - *Không trả lời và xin lỗi một cách lịch sự là bạn không biết* đối với các câu hỏi nằm ngoài hoàn toàn phạm vi tuyển sinh *năm 2024*, thông tin về Trường Đại học Đông Á hoặc các câu hỏi ở các lĩnh vực khác như thời tiết, môn học, thể thao, giải trí, v.v. Bạn chỉ có thể trả lời các thông tin về trường và tuyển sinh của Trường Đại học Đông Á.

  - *Không trả lời* các thông tin *có thể liên quan đến giáo dục và tuyển sinh của trường* nhưng không phải của *năm 2024* (Ví dụ: năm ngoái(2023), năm tới(2025), đề thi trung học phổ thông quốc gia 2025). Trong trường hợp này hãy xin lỗi và điều hướng người dùng sang cán bộ tuyển sinh.(Thời gian hiện tại: năm 2024)

Hình 38 dưới đây là minh họa cách mà prompt điều hướng người dùng được sử dụng trong hệ thống chatbot.

A diagram of a person's process

Description automatically generated

Hình 40. Prompt điều hướng người dùng trong hệ thống chatbot.

Đầu tiên (1), việc đặt giám định phát hiện truy vấn ngoài phạm vi là điều cần thiết, tránh cho việc chatbot có thể trả lời các thông tin không chính xác hoặc không chính thống. Điều này đảm bảo rằng chatbot chỉ trả lời duy nhất về các thông tin liên quan đến tuyển sinh của trường dựa trên thông tin đã được thu thập mà thôi. Nếu phát hiện ra truy vấn của người dùng (2) không nằm trong phạm vi, hệ thống sẽ trực tiếp sử dụng prompt này vào LLM (3) để tạo ra phản hồi (4) và nhắc nhở người dùng rằng chatbot không hỗ trợ phạm vi câu hỏi của người dùng.

### Trình quản lý agent và function calling

Để chatbot có thể hoạt động hiệu quả từ bước người dùng nhập truy vấn tới tới bước đưa ra phản hồi hoàn chỉnh, cần thiết lập một trình quản lý agent (agent management) để điều hướng các chức năng và giải quyết nhiệm vụ trong chatbot. Cụ thể tại đây sẽ nhận đầu vào là truy vấn của người dùng cùng với danh sách các hàm có thể gọi, đi kèm với đó là thông tin các tham số cần được truyền vào hàm, một mô tả ngắn về chức năng và nhiệm vụ của mỗi hàm. Những thông tin này sẽ được đưa vào mô hình LLM, sau đó LLM sẽ phân tích ngữ nghĩa của câu hỏi và lựa chọn hàm phù hợp, sau đó trả về một phản hồi ở định dạng JSON, trong đó bao gồm:

* Tên của hàm cần gọi (function\_calling): Đây là tên hàm mà LLM xác định là phù hợp nhất để xử lý câu hỏi.
* Các tham số cần thiết (parameters): Một đối tượng JSON chứa dữ liệu đầu vào cần thiết cho hàm đó.

Bằng cách này, hệ thống có thể tự động điều hướng đến đúng chức năng (hàm) dựa trên yêu cầu của người dùng. Đồng thời giảm độ phức tạp của logic xử lý trong Manager Agent, vì trách nhiệm chọn hàm được giao cho LLM. Hình 39 là minh họa cách hoạt động của một agent management.

A diagram of a company

Description automatically generated

Hình 41. Minh họa chức năng của agent management.

## Triển khai trình quản lý bộ nhớ chatbot

Trình quản lý bộ nhớ chatbot MemGPT là một giải pháp tiên tiến trong việc xây dựng chatbot, đặc biệt phù hợp cho các ứng dụng đòi hỏi quản lý bối cảnh dài hạn và phản hồi chính xác. Nhờ khả năng tóm tắt và lưu trữ thông tin, MemGPT giúp chatbot duy trì ngữ cảnh hội thoại trong giới hạn token và ghi nhớ thông tin quan trọng từ lịch sử tương tác trước. Điều này tạo nên trải nghiệm cá nhân hóa, hiệu quả hơn trong việc giải quyết các truy vấn phức tạp như tư vấn tuyển sinh. Lợi ích lớn nhất của MemGPT là kết hợp giữa tính chính xác, tính liên tục của ngữ cảnh và khả năng thích ứng với người dùng, giúp cải thiện cả hiệu suất và sự hài lòng trong giao tiếp.

Trong hệ thống chatbot này, chúng tôi sử dụng mô hình kiến trúc của MemGPT để làm bộ nhớ lưu trữ ngữ cảnh cho hệ thống chatbot trong một phiên trò chuyện của người dùng. Kiến trúc của MemGPT được thể hiện trong hình 40, bao gồm các thành phần và miêu tả chức năng như sau:

A diagram of a workflow

Description automatically generated

Hình 42. Mô hình kiến trúc hệ thống MemGPT [147].

Kiến trúc hệ thống của MemGPT được sử dụng lại bao gồm đầy đủ các thành phần trong hệ thống. Gồm các thành phần chính như sau [147]:

* LLM Finite Context Window: Đây là vùng làm việc chính của LLM, giới hạn bởi số lượng token tối đa (ví dụ: 8k token). Bao gồm hai phần chính:
  + Prompt Tokens: Các thông tin đầu vào để LLM thực hiện suy luận.
  + Completion Tokens: Đầu ra của LLM sau khi xử lý.
* System Instructions và MemGPT System Prompt (Các hướng dẫn hệ thống và MemGPT System Prompt), trong đó:
  + System Instructions: Các chỉ dẫn cốt lõi được hệ thống định nghĩa trước, không thay đổi, nhằm đảm bảo LLM hoạt động theo một cách thống nhất.
  + MemGPT System Prompt: Một vùng dữ liệu chỉ đọc, cung cấp ngữ cảnh nền tảng và các hướng dẫn cơ bản để hệ thống vận hành.
* Working Context và FIFO Queue:
  + Working Context: Là vùng lưu trữ tạm thời trong LLM để xử lý thông tin ngắn hạn. Có thể đọc-ghi, cho phép thêm thông tin mới thông qua các hàm xử lý (Functions).
  + FIFO Queue (First-In-First-Out): Một hàng đợi đơn giản, nơi thông tin được thêm vào theo thứ tự và dữ liệu cũ nhất sẽ bị đẩy ra khi đạt giới hạn. Dùng để quản lý luồng thông tin trong ngữ cảnh của LLM, giảm nguy cơ quá tải.
* Output Buffer: Là khu vực lưu trữ tạm thời kết quả (Completion Tokens) trước khi trả về cho người dùng hoặc các thành phần khác. Đảm bảo đầu ra được định dạng đúng và sẵn sàng sử dụng.
* Archival Storage (Lưu trữ dài hạn): Mục đích là để lưu trữ dữ liệu quan trọng để truy xuất trong tương lai. Ghi và truy vấn thông tin thông qua các Function Executor, cho phép lưu giữ thông tin không phụ thuộc vào ngữ cảnh hiện tại.
* Recall Storage (Lưu trữ ngắn hạn/trung hạn): Dùng để giữ thông tin cần truy cập nhanh hoặc thường xuyên trong quá trình xử lý. Kết nối chặt chẽ với Queue Manager để đảm bảo dữ liệu ngắn hạn được xử lý hiệu quả.
* Function Executor: Là thành phần trung gian chịu trách nhiệm ghi dữ liệu vào Archival Storage, truy xuất dữ liệu cần thiết từ các bộ lưu trữ, chuyển dữ liệu qua lại giữa các thành phần như FIFO Queue và Recall Storage. Những chức năng này giúp mở rộng khả năng ghi nhớ và xử lý của hệ thống.
* Queue Manager có chức năng: Điều phối hoạt động của FIFO Queue và Recall Storage, giúp đảm bảo thông tin trong FIFO Queue không bị quá tải, đồng thời sắp xếp và quản lý truy xuất dữ liệu trong Recall Storage và hỗ trợ chuyển giao dữ liệu giữa các vùng lưu trữ và LLM.

Cách nó hoạt động và tương tác với hệ thống chatbot như sau [147]:

* Khi nhận yêu cầu từ người dùng, Prompt Tokens được nạp vào Working Context. Tại đây nếu ngữ cảnh quá dài, MemGPT sẽ lưu trữ các thành phần ngữ cảnh cũ hơn vào Archived Storage, đồng thời đẩy ngữ cảnh ít quan trọng hơn vào FIFO Queue để giải phóng dung lượng xử lý.
* Trong quá trình xử lý (Processing Stage), MemGPT sẽ tự động kích hoạt Function Executor. Khi thông tin cần thiết không còn nằm trong ngữ cảnh hiện tại (Working Context), Function Executor sẽ truy xuất thông tin từ Recall Storage hoặc Archival Storage, đồng thời ghi thông tin cần thiết trở lại ngữ cảnh làm việc. Dữ liệu không còn quan trọng trong FIFO Queue được xóa hoặc lưu dài hạn vào Archival Storage.
* Khi trả lời người dùng (Output Stage), Kết quả (Completion Tokens) sau khi được xử lý xong sẽ được lưu vào Output Buffer, dữ liệu bổ sung từ Archival Storage hoặc Recall Storage được ghi nhớ cho các lần truy vấn sau này và trả lời cuối cùng được gửi đến người dùng qua giao diện chatbot.

Những chức năng của MemGPT giúp tối ưu hóa bối cảnh hội thoại, giữ thông tin quan trọng trong phạm vi token cho phép từ hệ thống. Nhờ vào khả năng tóm tắt và cắt giảm thông minh, MemGPT không chỉ giúp duy trì thông tin quan trọng mà còn giảm thiểu tình trạng quá tải bộ nhớ, đảm bảo rằng hệ thống luôn hoạt động hiệu quả. Điều này giúp chatbot duy trì sự liên tục trong cuộc trò chuyện, ngay cả khi số lượng tin nhắn hoặc thông tin người dùng ngày càng tăng, từ đó cung cấp các phản hồi chính xác và phù hợp hơn trong từng tình huống.

## Cơ sở dữ liệu người dùng

Trước khi bắt đầu một đoạn chat, yêu cầu người dùng nhập vào họ tên, số điện thoại để có thể bắt đầu đoạn chat. Sau khi nhập dữ liệu này thì được lưu vào MongoDB. Bằng cách này, hệ thống có thể quản lý thông tin người dùng một cách hiệu quả, đảm bảo rằng mỗi phiên trò chuyện đều được liên kết với một hồ sơ cụ thể. Việc lưu trữ thông tin trong MongoDB giúp dễ dàng truy xuất và phân tích dữ liệu người dùng, hỗ trợ các chiến lược chăm sóc khách hàng và tối ưu hóa trải nghiệm. Hình 41 minh hoạ cách thông tin người dùng được thu thập cho mục đích cá nhân hóa trải nghiệm hệ thống.

A diagram of a flowchart

Description automatically generated

Hình 43. Dữ liệu người dùng được thu thập và sử dụng cho mục đích cá nhân hóa.

Trong suốt quá trình diễn ra hành động hỏi đáp giữa người dùng và hệ thống, các thông tin của người dùng liên tục được thu thập và gửi tới hệ thống xử lý chatbot, nhằm cung cấp thêm thông tin và nhận diện được các khía cạnh mà người dùng quan tâm. Điều này giúp chatbot trả lời thân thiện, và xoay quanh vấn đề của người dùng và làm tăng thêm tính hiểu người dùng.

## Triển khai hệ thống chatbot hoàn chỉnh

Trong dự án này, chúng tôi sử dụng Framework Chainlit để tận dụng những công cụ và tính năng mạnh mẽ để xây dựng giao diện người dùng (UI). Với mục tiêu tối ưu hóa trải nghiệm người dùng, Chainlit cung cấp các thành phần giao diện, các tùy chọn tuỳ chỉnh giao diện, và khả năng tích hợp dễ dàng với các nền tảng khác. Việc có thể triển khai một cách nhanh chóng bằng công cụ giúp tiết kiệm thời gian xây dựng, hơn thế nữa có thể dánh nhiều thời gian hơn cho việc thiết kế, triển khai và tối ưu hóa hệ thống chatbot một cách hiệu quả hơn. Các hình ảnh dưới đây là kết quả minh hoạ triển khai hệ thống chatbot hỏi đáp tuyển sinh trường đại học Đông Á sử dụng LLM.

A screenshot of a computer

Description automatically generated

Hình 44. Minh họa bắt đầu trò chuyện sau skhi triển khai chatbot hoàn chỉnh.

A screenshot of a computer

Description automatically generated

A text on a white background

Description automatically generated

Hình 45. Minh họa kết quả triển khai hệ thống hoàn chỉnh.

A screenshot of a chat

Description automatically generated

Hình 46. . Minh họa kết quả triển khai hệ thống hoàn chỉnh.s

A screenshot of a chat

Description automatically generated

Hình 47. Minh họa kết quả triển khai hệ thống hoàn chỉnh.

A screenshot of a computer

Description automatically generatedA screenshot of a chat

Description automatically generated

Hình 48. Minh họa kết quả triển khai hệ thống hoàn chỉnh.

A screenshot of a chat

Description automatically generated

Hình 49. Minh họa kết quả triển khai hệ thống hoàn chỉnh.

Như vậy, đồ án xây dựng chatbot tuyển sinh trong phạm vi Trường Đại học Đông Á sử dụng LLM đã được hoàn thành. Trong báo cáo này chúng tôi đã trình bày về cơ sở lý thuyết liên quan, từ những kiến thức nền móng cho tới những chủ đề phức tạp xoay quanh vấn đề xây dựng hệ thống chatbot. Cụ thể trong chương 1, chúng tôi đã đi từ những phương pháp học máy, các mô hình học sâu là cảm hứng cho những mô hình LLM. Sau đó chúng tôi cũng đề cập đến các phần mềm và công cụ hỗ trợ, đồng thời nêu lên lợi ích và cách triển khai cho những bộ công cụ hỗ trợ ấy. Trong chương 2, chúng tôi đã nêu rõ phương pháp đề xuất của chúng tôi. Bằng cách đưa ra một chiến lược tối ưu ngay từ lúc bắt đầu, chúng tôi đã triển khai thành công các thành phần quan trọng cần được xây dựng để đáp ứng cho đề tài của chúng tôi. Cuối cùng, trong chương 3 chúng tôi đã đi từ những bước đầu cho đến bước hoàn thành mà chúng tôi đã thực hiện trong quá trình xây dựng đồ án. Cụ thể chúng tôi đã nêu rõ cách dữ liệu cần được xử lý trước khi đưa vào hệ thống chatbot, cách vận dụng các công cụ để trích xuất thông tin và điều khiển chúng hoạt động, đi cùng với đó chúng tôi đã nêu trọng tâm cách triển khai các thành phần của hệ thống chatbot, đồng thời miêu tả cách nó hoạt động cùng với chức năng đi kèm. Để xem thêm kết quả của hệ thống chatbot, chúng tôi trình bày các thử nghiệm của chúng tôi trong phần phụ lục để có thể tham khảo thêm.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

#### Kết luận chung

Đồ án *"Xây dựng chatbot tuyển sinh trường Đại học Đông Á sử dụng LLM"* đã giải quyết thành công bài toán xây dựng một hệ thống hỗ trợ tuyển sinh thông minh, tận dụng các tiến bộ trong công nghệ xử lý ngôn ngữ tự nhiên và học máy. Với tầm quan trọng ngày càng tăng của chuyển đổi số trong giáo dục, việc triển khai chatbot không chỉ giúp tự động hóa quy trình tư vấn tuyển sinh mà còn nâng cao trải nghiệm người dùng thông qua phản hồi nhanh chóng và chính xác.

Đồ án đã không chỉ xây dựng một hệ thống chatbot hoàn chỉnh mà còn triển khai một nền tảng lý thuyết vững chắc để hỗ trợ các khía cạnh kỹ thuật phức tạp. Các công nghệ như lưu trữ cơ sở dữ liệu vector, mô hình embedding, quản lý agents, và kỹ thuật truy xuất thông tin đều được tích hợp và vận hành mượt mà trong hệ thống. Hệ thống không chỉ đáp ứng được yêu cầu ban đầu mà còn chứng minh tiềm năng ứng dụng thực tiễn cao, mở ra hướng đi mới trong việc tự động hóa quản lý giáo dục tại trường Đại học Đông Á.

Quan trọng hơn, đồ án đã đạt được mục tiêu chính: thiết kế một chatbot không chỉ cung cấp thông tin tuyển sinh mà còn hỗ trợ giải đáp câu hỏi tuyển sinh, các vấn đề ngoài phạm vi và quản lý dữ liệu người dùng. Các thành quả này khẳng định tính hiệu quả và khả năng ứng dụng rộng rãi của hệ thống, góp phần cải thiện chất lượng dịch vụ của nhà trường.

#### Các kết quả đạt được

Với sự triển khai toàn diện, đồ án đã đạt được nhiều kết quả nổi bật. Về mặt lý thuyết, đồ án đã nghiên cứu và áp dụng các công nghệ hiện đại như mô hình Transformer, LLM, và kỹ thuật Prompt Engineering, đồng thời xây dựng nền tảng lý thuyết chặt chẽ về xử lý ngôn ngữ tự nhiên và học máy. Những nền tảng này không chỉ giúp phát triển hệ thống chatbot mà còn cung cấp một khung tham chiếu cho các nghiên cứu tương lai.

Về thực tiễn, chatbot đã được thiết kế và triển khai với nhiều tính năng quan trọng. Hệ thống sử dụng cơ sở dữ liệu vector để lưu trữ và truy xuất thông tin nhanh chóng, tích hợp các thuật toán tiên tiến để tăng độ chính xác của kết quả. Các tác nhân như Q&A Agent, và Out-of-Domain Agent đã hoạt động hiệu quả, mang đến trải nghiệm mượt mà cho người dùng. Đặc biệt, hệ thống quản lý lời nhắc (Prompt Engineering) và trình quản lý tác nhân giúp chatbot có khả năng phân luồng câu hỏi và xử lý linh hoạt các tình huống khác nhau.

Hệ thống chatbot đã được đánh giá hiệu quả thông qua các chỉ số chính như độ chính xác của phản hồi, thời gian truy vấn, và tính ổn định. Các tính năng như đặt lịch hẹn tư vấn hay quản lý dữ liệu người dùng đều hoạt động chính xác, phù hợp với nhu cầu thực tế của bộ phận tuyển sinh.

#### Phương hướng phát triển tiếp theo

Mặc dù đạt được những thành quả đáng kể, hệ thống chatbot vẫn còn nhiều tiềm năng phát triển nhằm nâng cao hiệu quả và mở rộng phạm vi ứng dụng. Một định hướng quan trọng là mở rộng khả năng hỗ trợ của chatbot sang các ngành học hoặc bộ phận khác trong nhà trường. Điều này có thể bao gồm việc tư vấn học thuật, hỗ trợ thủ tục hành chính, hoặc cung cấp thông tin chi tiết cho từng ngành học và phòng ban, giúp chatbot trở thành công cụ toàn diện phục vụ cho cả sinh viên và cán bộ nhà trường.

Hệ thống cũng cần được tối ưu hóa về hiệu suất để đảm bảo khả năng xử lý đồng thời cho nhiều người dùng. Việc áp dụng các thuật toán retrieval và rerank tiên tiến hơn có thể giúp cải thiện tốc độ phản hồi và độ chính xác của kết quả. Ngoài ra, tích hợp chatbot vào các nền tảng phổ biến như mạng xã hội, ứng dụng di động hoặc hệ thống nội bộ sẽ tăng cường khả năng tiếp cận người dùng, tạo ra trải nghiệm nhất quán trên nhiều môi trường.

Một yếu tố quan trọng khác là cải thiện khả năng tự động học hỏi từ dữ liệu mới. Điều này sẽ giúp chatbot thích nghi với những thay đổi trong thông tin tuyển sinh hoặc quy trình của trường mà không cần can thiệp thủ công. Việc tích hợp công cụ phân tích dữ liệu người dùng để dự đoán xu hướng hoặc đưa ra gợi ý thông minh cũng sẽ gia tăng giá trị của hệ thống.

Ngoài ra, để đảm bảo hệ thống hoạt động an toàn và bền vững, cần tiếp tục nghiên cứu và áp dụng các biện pháp bảo mật dữ liệu cá nhân, đặc biệt khi triển khai trên các môi trường công cộng. Những định hướng này không chỉ giúp hoàn thiện hệ thống hiện tại mà còn tạo tiền đề cho việc mở rộng nghiên cứu và ứng dụng trong các lĩnh vực khác, góp phần vào sự phát triển toàn diện của nhà trường.

# CÁC TÀI LIỆU THAM KHẢO

[1] Y. Xu *et al.*, “Artificial intelligence: A powerful paradigm for scientific research,” *The Innovation*, vol. 2, no. 4, Nov. 2021, doi: 10.1016/j.xinn.2021.100179.

[2] J. Chen *et al.*, “When large language models meet personalization: perspectives of challenges and opportunities,” *World Wide Web*, vol. 27, no. 4, p. 42, 2024, doi: 10.1007/s11280-024-01276-1.

[3] S. Mazumder, N. Ma, and B. Liu, “Towards a continuous knowledge learning engine for chatbots,” *arXiv preprint arXiv:1802.06024*, 2018.

[4] V. Boumová, “Traditional vs. modern teaching methods: Advantages and disadvantages of each,” Masarykova univerzita, Filozofická fakulta, 2008.

[5] A. T. Neumann, Y. Yin, S. Sowe, S. Decker, and M. Jarke, “An LLM-Driven Chatbot in Higher Education for Databases and Information Systems,” *IEEE Transactions on Education*, vol. 68, no. 1, pp. 103–116, 2025, doi: 10.1109/TE.2024.3467912.

[6] P. Johri, S. K. Khatri, A. T. Al-Taani, M. Sabharwal, S. Suvanov, and A. Kumar, “Natural Language Processing: History, Evolution, Application, and Future Work,” in *Proceedings of 3rd International Conference on Computing Informatics and Networks*, A. Abraham, O. Castillo, and D. Virmani, Eds., Singapore: Springer Singapore, 2021, pp. 365–375.

[7] Wikipedia, “History of natural language processing,” 2025.

[8] A. Panesar, *Machine learning and AI for healthcare*, vol. 10. Springer, 2019.

[9] A. Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, “Machine learning algorithm validation with a limited sample size,” *PLoS One*, vol. 14, no. 11, p. e0224365, 2019.

[10] H. Ij, “Statistics versus machine learning,” *Nat Methods*, vol. 15, no. 4, p. 233, 2018.

[11] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons. b*, vol. 4, no. 51–62, p. 56, 2017.

[12] Wikipedia, “Supervised learning — Section: How supervised learning algorithms work,” 2025.

[13] Z. Ghahramani, “Unsupervised learning,” in *Summer school on machine learning*, Springer, 2003, pp. 72–112.

[14] I. Goodfellow, “Deep learning,” 2016, *MIT press*.

[15] Wikipedia, “Weak supervision,” 2025.

[16] Y. Zhang and X. Wang, “Semi-Supervised Learning with Graph Neural Networks,” *arXiv preprint arXiv:2209.13777*, 2022.

[17] K. Choi, A. Grover, T. Singh, R. Shu, and S. Ermon, “Fair generative modeling via weak supervision,” in *International Conference on Machine Learning*, 2020, pp. 1887–1898.

[18] W. H. Tok, A. Bahree, and S. Filipi, *Practical Weak Supervision*. “ O’Reilly Media, Inc.,” 2021.

[19] V. Cabannes, L. Pillaud-Vivien, F. Bach, and A. Rudi, “Overcoming the curse of dimensionality with Laplacian regularization in semi-supervised learning,” *Adv Neural Inf Process Syst*, vol. 34, pp. 30439–30451, 2021.

[20] Y. Li, “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*, 2017.

[21] F. Somenzi and A. Trivedi, “Reinforcement learning and formal requirements,” in *Numerical Software Verification: 12th International Workshop, NSV 2019, New York City, NY, USA, July 13–14, 2019, Proceedings 12*, 2019, pp. 26–41.

[22] M. Naeem, S. T. H. Rizvi, and A. Coronato, “A gentle introduction to reinforcement learning and its application in different fields,” *IEEE access*, vol. 8, pp. 209320–209344, 2020.

[23] E. Boiy and M.-F. Moens, “A machine learning approach to sentiment analysis in multilingual web texts,” *Inf Retr Boston*, vol. 12, pp. 526–558, 2009.

[24] P. Shetty, R. Udhayakumar, A. Patil, M. Manwal, P. S. Vadar, and others, “Application of natural language processing (NLP) in machine learning,” in *2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE)*, 2023, pp. 949–957.

[25] E. H. Tusher, M. A. Ismail, M. A. Rahman, A. H. Alenezi, and M. Uddin, “Email spam: A comprehensive review of optimize detection methods, challenges, and open research problems,” *IEEE Access*, 2024.

[26] L. E. Lwakatare, A. Raj, I. Crnkovic, J. Bosch, and H. H. Olsson, “Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions,” *Inf Softw Technol*, vol. 127, p. 106368, 2020.

[27] S. Pouyanfar *et al.*, “A survey on deep learning: Algorithms, techniques, and applications,” *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[28] W. Khan, A. Daud, K. Khan, S. Muhammad, and R. Haq, “Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends,” *Natural Language Processing Journal*, vol. 4, p. 100026, 2023.

[29] I. H. Sarker, “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions,” *SN Comput Sci*, vol. 2, no. 6, pp. 1–20, 2021.

[30] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures,” *IEEE access*, vol. 7, pp. 53040–53065, 2019.

[31] F. Incitti, F. Urli, and L. Snidaro, “Beyond word embeddings: A survey,” *Information Fusion*, vol. 89, pp. 418–436, 2023.

[32] M. Zulqarnain and M. Saqlain, “Text readability evaluation in higher education using CNNs,” *Journal of industrial intelligence*, vol. 1, no. 3, pp. 184–193, 2023.

[33] C. C. Aggarwal and others, *Neural networks and deep learning*, vol. 10, no. 978. Springer, 2018.

[34] J. Tang *et al.*, “Bridging biological and artificial neural networks with emerging neuromorphic devices: fundamentals, progress, and challenges,” *Advanced materials*, vol. 31, no. 49, p. 1902761, 2019.

[35] H. Pothina and K. V Nagaraja, “Artificial neural network and math behind it,” in *Smart Trends in Computing and Communications: Proceedings of SmartCom 2022*, Springer, 2022, pp. 205–221.

[36] S. Haykin, *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[37] A. D. Dongare, R. R. Kharde, A. D. Kachare, and others, “Introduction to artificial neural network,” *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 1, pp. 189–194, 2012.

[38] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1, no. 2. MIT press Cambridge, 2016.

[39] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, “A review of activation function for artificial neural network,” in *2020 IEEE 18th world symposium on applied machine intelligence and informatics (SAMI)*, 2020, pp. 281–286.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Adv Neural Inf Process Syst*, vol. 25, 2012.

[41] T. Babs, “The mathematics of neural networks,” *Medium. https://medium. com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05 (accessed Oct. 6, 2024)*, 2018.

[42] S. Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomputing*, vol. 5, no. 4–5, pp. 185–196, 1993.

[43] J. Mao and A. K. Jain, “Artificial neural networks for feature extraction and multivariate data projection,” *IEEE Trans Neural Netw*, vol. 6, no. 2, pp. 296–317, 1995.

[44] L. F. Guilhoto, “An overview of artificial neural networks for mathematicians,” *Univ. Chicago*, 2018.

[45] J. R. M. Smits, W. J. Melssen, L. M. C. Buydens, and G. Kateman, “Using artificial neural networks for solving chemical problems: Part I. Multi-layer feed-forward networks,” *Chemometrics and Intelligent Laboratory Systems*, vol. 22, no. 2, pp. 165–189, 1994.

[46] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[47] M. Khashei, A. Z. Hamadani, and M. Bijari, “A novel hybrid classification model of artificial neural networks and multiple linear regression models,” *Expert Syst Appl*, vol. 39, no. 3, pp. 2606–2620, 2012.

[48] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[49] N. F. Mohammed and N. Omar, “Arabic named entity recognition using artificial neural network,” *Journal of Computer Science*, vol. 8, no. 8, p. 1285, 2012.

[50] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, “Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification,” *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[51] Y. Goldberg, *Neural network methods in natural language processing*. Morgan & Claypool Publishers, 2017.

[52] S. Zhang, C. Liu, H. Jiang, S. Wei, L. Dai, and Y. Hu, “Feedforward sequential memory networks: A new structure to learn long-term dependency,” *arXiv preprint arXiv:1512.08301*, 2015.

[53] S. Zhang, H. Jiang, M. Xu, J. Hou, and L. Dai, “A fixed-size encoding method for variable-length sequences with its application to neural network language models,” *arXiv preprint arXiv:1505.01504*, 2015.

[54] Y. Yao and A. Koller, “Structural generalization is hard for sequence-to-sequence models,” *arXiv preprint arXiv:2210.13050*, 2022.

[55] D. M. Nemeskey, “Natural language processing methods for language modeling,” 2020.

[56] I. D. Mienye, T. G. Swart, and G. Obaido, “Recurrent neural networks: A comprehensive review of architectures, variants, and applications,” *Information*, vol. 15, no. 9, p. 517, 2024.

[57] T. L. Fine, *Feedforward neural network methodology*. Springer Science & Business Media, 2006.

[58] S. Grossberg, “Recurrent neural networks,” *Scholarpedia*, vol. 8, no. 2, p. 1888, 2013.

[59] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent advances in recurrent neural networks,” *arXiv preprint arXiv:1801.01078*, 2017.

[60] A. L. Caterini, D. E. Chang, A. L. Caterini, and D. E. Chang, “Recurrent neural networks,” *Deep neural networks in a mathematical framework*, pp. 59–79, 2018.

[61] A. Alqushaibi, S. J. Abdulkadir, H. M. Rais, and Q. Al-Tashi, “A review of weight optimization techniques in recurrent neural networks,” in *2020 international conference on computational intelligence (ICCI)*, 2020, pp. 196–201.

[62] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding.,” in *Interspeech*, 2013, pp. 2524–2528.

[63] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” *arXiv preprint arXiv:1506.00019*, 2015.

[64] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[65] U. Lotrič and A. Dobnikar, “Parallel implementations of recurrent neural network learning,” in *International Conference on Adaptive and Natural Computing Algorithms*, 2009, pp. 99–108.

[66] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, “A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU,” *arXiv preprint arXiv:2305.17473*, 2023.

[67] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[68] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, and others, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” 2001, *A field guide to dynamical recurrent neural networks. IEEE Press In*.

[69] P. S. Muhuri, P. Chatterjee, X. Yuan, K. Roy, and A. Esterline, “Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks,” *Information*, vol. 11, no. 5, p. 243, 2020.

[70] J. Zhao *et al.*, “Do RNN and LSTM have long memory?,” in *International Conference on Machine Learning*, 2020, pp. 11365–11375.

[71] H. H. Björnsson and J. Kaldal, “Exploration and Evaluation of RNN Models on Low-Resource Embedded Devices for Human Activity Recognition,” 2023.

[72] M. Beck *et al.*, “xlstm: Extended long short-term memory,” *arXiv preprint arXiv:2405.04517*, 2024.

[73] P. L. Rodriguez and A. Spirling, “Word embeddings: What works, what doesn’t, and how to tell the difference for applied research,” *J Polit*, vol. 84, no. 1, pp. 101–115, 2022.

[74] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad, “A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-Art Word Representation Language Models,” *arXiv preprint arXiv:2010.15036*, 2020.

[75] F. Almeida and G. Xexéo, “Word Embeddings: A Survey,” *arXiv preprint arXiv:1901.09069*, 2019.

[76] Milvus, “How do embedding models convert text into vectors?,” *Milvus AI Quick Reference*, 2023, [Online]. Available: https://milvus.io/ai-quick-reference/how-do-embedding-models-convert-text-into-vectors

[77] P. Worth Jr, “Word Embeddings and Semantic Spaces in Natural Language Processing,” *Scientific Research Publishing*, 2023, [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=122466

[78] P. Gupta and M. Jaggi, “Obtaining Better Static Word Embeddings Using Contextual Embedding Models,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 5241–5253. [Online]. Available: https://aclanthology.org/2021.acl-long.408/

[79] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *arXiv preprint arXiv:1301.3781*, 2013.

[80] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[81] K. Ethayarajh, “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings,” *arXiv preprint arXiv:1909.00512*, 2019.

[82] Q. Liu, M. J. Kusner, and P. Blunsom, “A Survey on Contextual Embeddings,” *arXiv preprint arXiv:2003.07278*, 2020.

[83] K. Ethayarajh, “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings,” *arXiv preprint arXiv:1909.00512*, 2019.

[84] M. E. Peters *et al.*, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 2227–2237.

[85] S. Arora, A. May, J. Zhang, and C. Ré, “Contextual Embeddings: When Are They Worth It?,” *arXiv preprint arXiv:2005.09117*, 2020.

[86] A. Vaswani *et al.*, “Attention is all you need,” *Adv Neural Inf Process Syst*, vol. 30, 2017.

[87] J. Alammar, “The illustrated transformer,” *The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time*, vol. 27, pp. 1–2, 2018.

[88] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” *arXiv preprint arXiv:1904.09675*, 2019.

[89] S. Jastrzebski, D. Arpit, N. Ballas, V. Verma, T. Che, and Y. Bengio, “Residual connections encourage iterative inference,” *arXiv preprint arXiv:1710.04773*, 2017.

[90] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.

[91] T. H. Việt and N. T. Hiền, “MÔ HÌNH TRANSFORMERS VÀ ỨNG DỤNG TRONG XỬ LÝ NGÔN NGỮ TỰ NHIÊN”.

[92] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, “Disan: Directional self-attention network for rnn/cnn-free language understanding,” in *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[93] D. Myers *et al.*, “Foundation and large language models: fundamentals, challenges, opportunities, and social impacts,” *Cluster Comput*, vol. 27, no. 1, pp. 1–26, 2024.

[94] N. Patwardhan, S. Marrone, and C. Sansone, “Transformers in the real world: A survey on nlp applications,” *Information*, vol. 14, no. 4, p. 242, 2023.

[95] Cloudflare, “Mô hình ngôn ngữ lớn (LLM) là gì?,” 2024.

[96] J. C. Luna, “What is an LLM? A Guide on Large Language Models and How They Work,” 2024. [Online]. Available: https://www.datacamp.com/blog/what-is-an-llm-a-guide-on-large-language-models

[97] J. Jia *et al.*, “From Principles to Applications: A Comprehensive Survey of Discrete Tokenizers in Generation, Comprehension, Recommendation, and Information Retrieval,” *arXiv preprint arXiv:2502.12448*, 2025.

[98] C. W. Schmidt *et al.*, “Tokenization Is More Than Compression,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 678–702. doi: 10.18653/v1/2024.emnlp-main.40.

[99] S. J. Mielke *et al.*, “Between Words and Characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP,” 2021. [Online]. Available: https://arxiv.org/abs/2112.10508

[100] X. Liang *et al.*, “Controllable text generation for large language models: A survey,” *arXiv preprint arXiv:2408.12599*, 2024.

[101] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, “The Curious Case of Neural Text Degeneration,” in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://arxiv.org/pdf/1904.09751

[102] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, “Efficient Streaming Language Models with Attention Sinks,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. [Online]. Available: https://arxiv.org/abs/2309.17453

[103] J. Liu, J.-W. Chung, Z. Wu, F. Lai, M. Lee, and M. Chowdhury, “Andes: Defining and Enhancing Quality-of-Experience in LLM-Based Text Streaming Services,” in *Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC ’24)*, 2024. [Online]. Available: https://arxiv.org/abs/2404.16283

[104] Nithyashree, “What is Chunking in Natural Language processing?,” 2021.

[105] D. Raikar, “How Chunking Strategies Work: Paragraph, Sentence and Smart Techniques,” 2024.

[106] T. ML, “Embedding,” 2024.

[107] P. Phan, “ChatGPT Series 5: Tìm hiểu về Retrieval Augmented Generation (RAG),” 2023.

[108] S. Ahmed, “What is Retrieval-Augmented Generation(RAG) in LLM and How it works?,” 2024.

[109] D. Shah, “Reciprocal Rank Fusion (RRF) explained in 4 mins — How to score results form multiple retrieval methods in RAG,” 2024.

[110] Z. Rackauckas, “RAG-Fusion: a New Take on Retrieval-Augmented Generation,” 2024. doi: https://doi.org/10.5121/ijnlc.2024.13103.

[111] T. Q. Huy, “LLM agent: Tổng quan về LLM agent,” 2024.

[112] T. Palaniyappan, “How LLM Agent works?,” 2024.

[113] L. Weng, “LLM Powered Autonomous Agents,” 2023.

[114] R. Graph, “Prompt Engineering,” 2024.

[115] L. Phan, “Tất tần tật những kĩ thuật Prompt Engineering hữu ích nhất cho chatGPT,” 2024.

[116] S. Ahmed, “LLM Prompt Engineering for Beginners: What It Is and How to Get Started,” 2024.

[117] J. Liu *et al.*, “Generated Knowledge Prompting for Commonsense Reasoning,” *CoRR*, vol. abs/2110.08387, 2021, [Online]. Available: https://arxiv.org/abs/2110.08387

[118] J. C. Luna, “What is an LLM? A Guide on Large Language Models and How They Work,” 2024.

[119] M. Gothankar, “What is LLM & How to Build Your Own Large Language Models?,” 2024.

[120] M. T. School, “Ưu điểm vượt trội của lập trình trí tuệ nhân tạo python?,” 2024.

[121] Wikipedia, “Python (ngôn ngữ lập trình),” 2024.

[122] C. Team, “Chainlit,” 2024.

[123] T. Rasul, “Building a Chatbot Application with Chainlit and LangChain,” 2024.

[124] Milvus, “Milvus,” 2024.

[125] Milvus, “introducing-milvus-lite-start-building-a-genai-application-in-seconds,” 2024.

[126] MongoDB, “MongoDB,” 2024.

[127] M. Narang, “MemGPT with Real-life Example: Bridging the Gap Between AI and OS,” 2024.

[128] M. Raspuzzi, “what is memGPT?,” 2023.

[129] C. Packer *et al.*, “MemGPT: Towards LLMs as Operating Systems,” 2024. [Online]. Available: https://arxiv.org/abs/2310.08560

[130] M. L. Expert, “MemGPT - Unlimited Context (Memory) for LLMs,” 2024.

[131] L. Teixeira, “MemGPT: Unlimited Memory without Token Constraints for Generative AI Platforms, like GPT-4, LaMDA, PaLM, LLAMA, CLAUDE, and others,” 2023.

[132] donga, “ĐẠI HỌC ĐÔNG Á,” 2024.

[133] T. sinh Đại học Đông Á, “Tuyển sinh Đại học Đông Á Official 2024,” 2024.

[134] R. Khawaja, “Demystifying Embeddings 101: The Foundation of Large Language Models,” 2023.

[135] novita.ai, “What is LLM Embeddings: All You Need To Know,” 2024.

[136] Sbert, “Pretrained Models: Sentence Transformers,” 2024.

[137] Deepinfra, “sentence-transformers/all-MiniLM-L6-v2,” 2024.

[138] Sbert, “Pretrained Models: Multilingual Models,” 2024.

[139] H. Face, “Alibaba-NLP/gte-multilingual-base,” 2024.

[140] huggingface, “BAAI/bge-m3,” 2024. [Online]. Available: https://huggingface.co/BAAI/bge-m3

[141] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, “BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation,” 2024. [Online]. Available: https://arxiv.org/abs/2402.03216

[142] Z. Nussbaum, J. X. Morris, B. Duderstadt, and A. Mulyar, “Nomic Embed: Training a Reproducible Long Context Text Embedder,” 2024.

[143] H. Face, “nomic-ai/nomic-embed-text-v1.5,” 2024.

[144] A. Trivedi, “Top 10 Open-Source LLMs for 2025 and Their Uses,” 2024.

[145] J. Sumrak, “7 LLM use cases and applications in 2024,” 2024.

[146] J. Fell, “Stanford just released its annual AI Index report. Here’s what it reveals,” 2024.

[147] MemGPT, “MemGPT,” 2024.

--HẾT--