**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP

**Nghiên cứu mô hình Vistral-7B-Chat và áp dụng vào xây dựng ứng dụng Chatbot hỗ trợ môn học dựa trên mô hình, hỗ trợ đa ngôn ngữ cho tài liệu bài giảng khoa Công nghệ thông tin**

|  |  |  |
| --- | --- | --- |
| Ngành | : | Công nghệ thông tin |
| Niên khóa | : | 2021-2025 |
| Lớp | : | DH21DTC |
| Sinh viên thực hiện |  |  |
| Võ Nguyễn Nhật Khương | : | 21130407 |
| Lê Phi Long | : | 21130436 |

TP.HỒ CHÍ MINH, ngày …. tháng …. năm 2025

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM**

**CÔNG NGHỆ THÔNG TIN**



LUẬN VĂN TỐT NGHIỆP

**Nghiên cứu mô hình Vistral-7B-Chat và áp dụng vào xây dựng ứng dụng Chatbot hỗ trợ môn học dựa trên mô hình, hỗ trợ đa ngôn ngữ cho tài liệu bài giảng khoa Công nghệ thông tin**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **Sinh viên thực hiện** |
| **Nguyễn Đức Công Song** | **Võ Nguyễn Nhật Khương** |
|  | **Lê Phi Long** |

TP.HỒ CHÍ MINH, ngày …. tháng …. năm 2025

# LỜI CẢM ƠN

# CÔNG TRÌNH HOÀN TẤT TẠI TRƯỜNG ĐẠI HỌC NÔNG LÂM THÀNH PHỐ HỒ CHÍ MINH

Cán bộ hướng dẫn: Thạc sĩ Nguyễn Đức Công Song

Cán bộ phản biện: Thạc sĩ Lê Phi Hùng

Luận văn cử nhân được bảo vệ tại HỘI ĐỒNG CHẤM LUẬN VĂN CỬ NHÂN TRƯỜNG ĐẠI HỌC NÔNG LÂM TP HCM ngày tháng năm

NHIỆM VỤ LUẬN VĂN CỬ NHÂN

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐH NÔNG LÂM TPHCM**

**CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc Lập-Tự Do-Hạnh Phúc**

Họ tên sinh viên: **VÕ NGUYỄN NHẬT KHƯƠNG**

Ngày tháng năm sinh: 21/05/2003

# Chuyên ngành: Công nghệ thông tin

Ngành : Công nghệ thông tin

Phái: **Nam**

Nơi sinh: Bình Thuận

Họ tên sinh viên: **LÊ PHI LONG**

Ngày tháng năm sinh: 19/02/2003

Chuyên ngành: Công nghệ thông tin

Ngành : Công nghệ thông tin

Phái: **Nam**

Nơi sinh: Đồng Hới

## TÊN ĐỀ TÀI : Nghiên cứu mô hình Vistral-7B-Chat và áp dụng vào xây dựng ứng dụng Chatbot hỗ trợ môn học dựa trên mô hình, hỗ trợ đa ngôn ngữ cho tài liệu bài giảng khoa Công nghệ thông tin

## NHIỆM VỤ VÀ NỘI DUNG :

## NGÀY GIAO NHIỆM VỤ :

## NGÀY HOÀN THÀNH NHIỆM VỤ :

## HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN :

Ngày / /

CÁN BỘ HƯỚNG DẪN

(Ký và ghi rõ họ tên)

Ngày / /

KHOA CNTT

(Ký và ghi rõ họ tên)

Ngày / /

CÁN BỘ PHẢN BIỆN

(Ký và ghi rõ họ tên)

# DANH MỤC CÁC BẢNG

[Hình 1 : Quá trình phát triển của mô hình LLMs 1](#_Toc205156362)

[Hình 2 : biểu đồ thể hiện mức độ quan tâm của cộng đồng đối với LLMs 2](#_Toc205156363)

[Hình 3 : Bảng đánh giá các mô hình LLMs 10](#_Toc205156364)

[Hình 4 : code cho phần trích xuất dữ liệu PDF 36](#_Toc205156365)

[Hình 5: code cho phần phân đoạn tài liệu 38](#_Toc205156366)

[Hình 6 : code cho phần cơ chế truy xuất 41](#_Toc205156367)

[Hình 7: code tạo tập dữ liệu vector có cấu trúc 43](#_Toc205156368)

# DANH SÁCH CHỮ VIẾT TẮT

**AI** – Artificial Intelligence

**BERT** – Bidirectional Encoder Representations from Transformers

**CNNs** – Convolutional Neural Networks

**CPU** – Central Processing Unit

**CRFs** – Conditional Random Fields

**CSV** – Comma-Separated Values

**DNNs** – Deep Neural Networks

**GB** – Gigabyte

**GloVe** – Global Vectors for Word Representation

**GPT** – Generative Pre-trained Transformer

**GPU** – Graphics Processing Unit

**GRU** – Gated Recurrent Unit

**HMMs** – Hidden Markov Models

**LLM** – Large Language Model

**LSTM** – Long Short-Term Memory

**NLP** – Natural Language Processing

**RAG** – Retrieval Augmented Generation

**NLTK** – Natural Language Toolkit

**PDF** – Portable Document Format

**RAM** – Random Access Memory

**STEM** – Science, Technology, Engineering, and Mathematics

**TF-IDF** – Term Frequency–Inverse Document Frequency

# MỤC LỤC

[LỜI CẢM ƠN iii](#_Toc205040627)

[CÔNG TRÌNH HOÀN TẤT TẠI TRƯỜNG ĐẠI HỌC NÔNG LÂM THÀNH PHỐ HỒ CHÍ MINH iv](#_Toc205040628)

[Cán bộ hướng dẫn: Thạc sĩ Nguyễn Đức Công Song iv](#_Toc205040629)

[Cán bộ phản biện: Thạc sĩ Lê Phi Hùng iv](#_Toc205040630)

[NHIỆM VỤ LUẬN VĂN CỬ NHÂN v](#_Toc205040631)

[Chuyên ngành: Công nghệ thông tin](https://d.docs.live.net/58b321c27f7eb267/Ta%CC%80i%20li%C3%AA%CC%A3u/Document/docx/KLTN_THANG_8.docx#_Toc205040632) v

[I TÊN ĐỀ TÀI : Nghiên cứu mô hình Vistral-7B-Chat và áp dụng vào xây dựng ứng dụng Chatbot hỗ trợ môn học dựa trên mô hình, hỗ trợ đa ngôn ngữ cho tài liệu bài giảng khoa Công nghệ thông tin v](#_Toc205040633)

[II NHIỆM VỤ VÀ NỘI DUNG : v](#_Toc205040634)

[III NGÀY GIAO NHIỆM VỤ : v](#_Toc205040635)

[IV NGÀY HOÀN THÀNH NHIỆM VỤ : v](#_Toc205040636)

[V HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN : v](#_Toc205040637)

[DANH MỤC CÁC BẢNG vi](#_Toc205040638)

[DANH SÁCH CHỮ VIẾT TẮT vii](#_Toc205040639)

[MỤC LỤC ix](#_Toc205040640)

[MỞ ĐẦU 1](#_Toc205040641)

[I Lý do chọn đề tài : 1](#_Toc205040642)

[II Các công trình nghiên cứu liên quan : 2](#_Toc205040643)

[III Mục tiêu của đề tài : 2](#_Toc205040644)

[IV Nội dung và phạm vi nghiên cứu : 3](#_Toc205040645)

[V Sản phẩm của đề tài : 3](#_Toc205040646)

[CƠ SỞ LÝ THUYẾT 4](#_Toc205040647)

[I Sơ lược về ngôn ngữ tự nhiên : 4](#_Toc205040648)

[II Các ứng dụng xử lý ngôn ngữ tự nhiên trong thực tiễn : 4](#_Toc205040649)

[III Các mô hình học sâu xử lý ngôn ngữ tự nhiên : 5](#_Toc205040650)

[IV Vấn đề của các mô hình xử lý ngôn ngữ tự nhiên truyền thống : 6](#_Toc205040651)

[PHƯƠNG PHÁP LUẬN 7](#_Toc205040652)

[I Đánh giá mô hình ngôn ngữ lớn : 7](#_Toc205040653)

[1.1 Cấu hình thiết bị : 7](#_Toc205040654)

[1.2 Tập dữ liệu VMLU : 7](#_Toc205040655)

[1.3 Phương pháp đánh giá : 8](#_Toc205040656)

[1.4 Kết quả đánh giá : 10](#_Toc205040657)

[1.5 Phân tích kết quả : 11](#_Toc205040658)

[II Định hướng cải thiện : 11](#_Toc205040659)

[III Chuẩn bị bộ dữ liệu huấn luyện mô hình : 11](#_Toc205040660)

[3.1 Trích xuất văn bản từ file PDF : 12](#_Toc205040661)

[3.2 Xác định từ khóa : 14](#_Toc205040662)

[3.3 Tiền xử lý dữ liệu : 15](#_Toc205040663)

[3.4 Loại bỏ dòng trống : 16](#_Toc205040664)

[3.5 Loại bỏ dòng in hoa toàn bộ : 17](#_Toc205040665)

[3.6 Phân tách câu : 19](#_Toc205040666)

[3.7 Chuẩn hóa văn bản : 20](#_Toc205040667)

[3.8 Loại bỏ stop words : 22](#_Toc205040668)

[3.9 Lọc từ tần suất thấp : 23](#_Toc205040669)

[3.10 Lọc câu đưa ra theo phân loại : 26](#_Toc205040670)

[3.11 Lọc và sắp xếp dữ liệu : 28](#_Toc205040671)

[3.12 Kiểm tra và sửa lỗi chính tả : 30](#_Toc205040672)

[3.13 Tổng hợp và lưu trữ dữ liệu : 32](#_Toc205040673)

[3.14 Kết quả : 35](#_Toc205040674)

[IV Tinh chỉnh mô hình : 36](#_Toc205040675)

[V Tích hợp RAG vào mô hình : 36](#_Toc205040676)

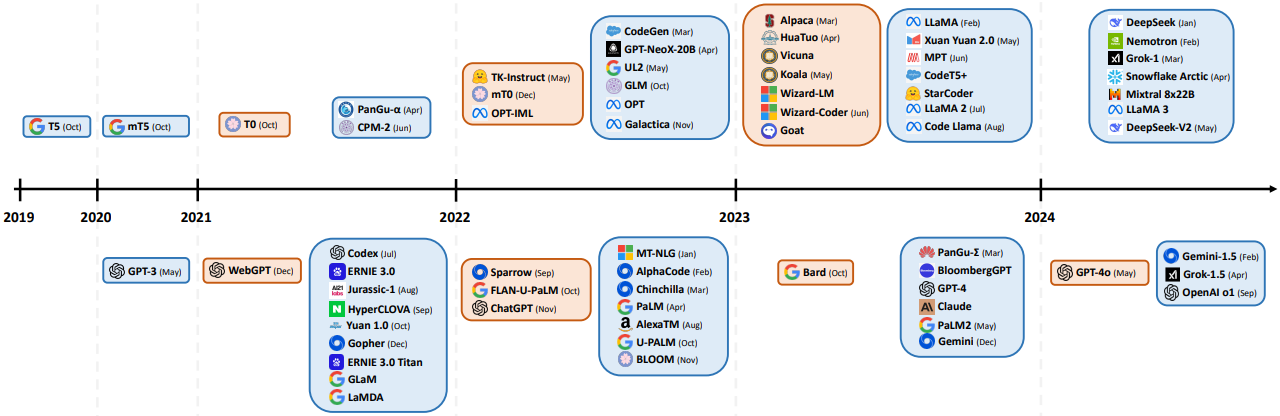
[TÀI LIỆU THAM KHẢO 36](#_Toc205040677)

[PHỤ LỤC 36](#_Toc205040678)

# MỞ ĐẦU

## Lý do chọn đề tài :

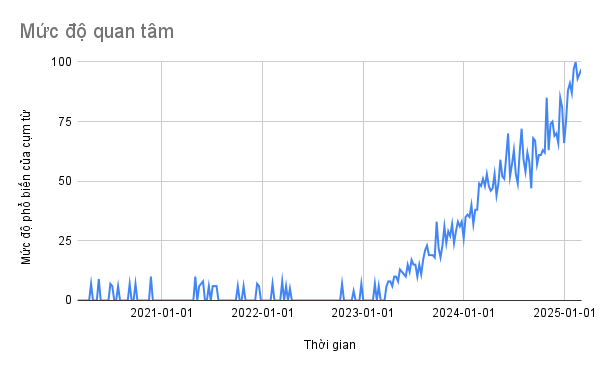
Dựa trên những diễn biến gần đây trong lĩnh vực trí tuệ nhân tạo, sự trỗi dậy mạnh mẽ của các mô hình ngôn ngữ tạo sinh đã thu hút sự quan tâm đặc biệt của cộng đồng nghiên cứu và công nghệ. Sự bùng nổ này được đánh dấu bằng sự ra đời và phát triển vượt bậc của hàng loạt các mô hình ngôn ngữ lớn (LLMs) trong những năm gần đây.



Hình : Quá trình phát triển của mô hình LLMs

Biểu đồ trên (Error: Reference source not found) minh họa một cách trực quan sự gia tăng đáng kể về số lượng và độ phức tạp của các LLMs theo thời gian, cho thấy một xu hướng phát triển nhanh chóng và mạnh mẽ trong lĩnh vực này.

Để làm rõ hơn về mức độ quan tâm của cộng đồng đối với chủ đề này, chúng tôi tham khảo dữ liệu từ Google Trends với từ khóa "llm" trong khoảng thời gian từ tháng 3 năm 2020 đến tháng 2 năm 2025. Kết quả được thể hiện qua biểu đồ sau:



Hình : biểu đồ thể hiện mức độ quan tâm của cộng đồng đối với LLMs

Lưu ý: Các giá trị trên biểu đồ thể hiện mức độ quan tâm tương đối đến hoạt động.

tìm kiếm liên quan đến từ khóa "llm" so với thời điểm và khu vực đạt mức cao nhất (giá trị 100). Giá trị 50 cho thấy mức độ quan tâm bằng một nửa so với đỉnh điểm, và giá trị 0 chỉ ra rằng không có đủ dữ liệu cho từ khóa này.

## Các công trình nghiên cứu liên quan :

Sự bùng nổ của các Mô hình Ngôn ngữ lớn (LLMs) đã tạo ra những bước tiến vượt bậc trong lĩnh vực Xử lý ngôn ngữ tự nhiên (NLP) với khả năng giải quyết nhiều tác vụ phức tạp từ tạo sinh văn bản đến trả lời các câu hỏi từ người dùng. Với việc mô hình Mistral 7B đã đánh bại mô hình Llama 2 13B trên nhiều điểm chuẩn nhờ áp dụng các kỹ thuật tiên tiến như Grouped-Query Attention (GQA) và Sliding Window Attention (SWA), giúp tối ưu hóa tốc độ suy luận và khả năng xử lý chuỗi dài.

Tuy nhiên, các kết quả trên chỉ khả quan với ngôn ngữ chủ yếu là tiếng Anh, còn với tiếng Việt thì vẫn tồn tại một khoảng trống đáng kể do hạn chế về các bộ dự liệu chuẩn cho tiếng Việt. Để giải quyết vấn đề này, Vi-Mistral-X (ViStral) đã được giới thiệu như một mô hình ngôn ngữ lớn tiên tiến dành riêng cho tiếng Việt. Dựa trên kiến trúc của Mistral, ViStral sử dụng phương pháp continual pre-training trên bộ dữ liệu tiếng Việt phong phú, giúp mô hình nắm bắt sâu sắc các sắc thái ngôn ngữ và tạo ra văn bản tiếng Việt chính xác, đặc biệt trong Vietnamese Multitask Language Understanding (VMLU).

Dựa trên nền tảng từ các nghiên cứu trên, đề tài của nhóm tôi tập trung vào việc áp dụng mô hình Vistral-7B-Chat, kết hợp với kỹ thuật Retrieval-Augmented Generation (RAG) và phương pháp tinh chỉnh hiệu quả tham số LoRA, để xây dựng ứng dụng chatbot hỗ trợ môn học, giải quyết nhu cầu đặc thù của tài liệu giảng dạy khoa công nghệ thông tin.

## Mục tiêu của đề tài :

1. Tiến hành nghiên cứu và so sánh hiệu quả của các mô hình LLM phổ biến, bao gồm vinai/PhoGPT-4B-Chat, Llama2-7B, vietnamese-Llama2-7B-120GB và đặc biệt là Vistral-7B-Chat với bộ dữ liệu ViLLM-Eval.
2. Xây dựng và triển khai ứng dụng web demo:
   1. Phát triển một ứng dụng web demo cho phép người dùng tương tác với chatbot.
   2. Ứng dụng sẽ được xây dựng dựa trên mô hình Vistral-7B-Chat đã được tinh chỉnh (finetune) để tối ưu hóa khả năng hỗ trợ học tập.
   3. Chatbot sẽ có khả năng trả lời các câu hỏi và cung cấp thông tin liên quan đến tài liệu bài giảng Data Warehouse, Cấu trúc dữ liệu và giải thuật của khoa Công nghệ thông tin, sách Java 8 in Action: Lambdas, streams, and functional-style programming.
   4. Ứng dụng có thể hỗ trợ đa ngôn ngữ cho tài liệu bài giảng.

## Nội dung và phạm vi nghiên cứu :

* Nội dung nghiên cứu:
  + Nghiên cứu và triển khai, tinh chỉnh mô hình Vistral-7B-Chat
  + Nghiên cứu và ứng dụng RAG vào mô hình LLM
  + Nghiên cứu framework FastAPI, Hugging Face
* Phạm vị nghiên cứu:
  + Tập trung vào các tài liệu phổ biến như PDF, DOCX, TXT
  + Hỗ trợ đọc, hiểu đầu vào của mô hình ngôn ngữ với tiếng Việt và tiếng Anh
  + Ứng dụng chạy trên nền tảng web, theo mô hình Client-Server

## Sản phẩm của đề tài :

Bộ tài liệu nghiên cứu và ứng dụng mô hình Vistral-7B-Chat

Kết quả so sánh giữa các mô hình: Vistral-7B-Chat, vinai/PhoGPT-4B-Chat, Llama2-7B, Vietnamese-Llama2-7B-120GB

Ứng dụng demo Chatbot trên nền tảng web hỗ trợ mô học dựa trên tài liệu bài giảng của khoa Công nghệ thông tin, với khả năng hỗ trợ đa ngôn ngữ, cho phép người dùng nhập câu hỏi bằng một ngôn ngữ và nhận câu trả lời bằng ngôn ngữ mong muốn

# CƠ SỞ LÝ THUYẾT

## Sơ lược về ngôn ngữ tự nhiên :

Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực thuộc ngành trí tuệ nhân tạo (Artificial Intelligence - AI), tập trung vào khả năng giúp máy tính hiểu, diễn giải và tạo ra ngôn ngữ của con người một cách thông minh. Mục tiêu cốt lõi của NLP là thu hẹp khoảng cách giao tiếp giữa con người và máy móc, cho phép tương tác tự nhiên và hiệu quả hơn. NLP bao gồm một tập hợp các kỹ thuật và phương pháp, từ phân tích cú pháp, ngữ nghĩa đến hiểu biết ngữ cảnh và sinh ngôn ngữ. Sự phát triển của NLP đóng vai trò then chốt trong việc xây dựng các hệ thống thông minh có khả năng làm việc với dữ liệu ngôn ngữ phong phú trong thế giới thực.

## Các ứng dụng xử lý ngôn ngữ tự nhiên trong thực tiễn :

NLP đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, mang lại những tác động to lớn đến đời sống và công việc. Một số ứng dụng tiêu biểu bao gồm:

* Dịch máy (Machine Translation): Cho phép chuyển đổi văn bản hoặc lời nói từ ngôn ngữ này sang ngôn ngữ khác một cách tự động, hỗ trợ giao tiếp và trao đổi thông tin trên toàn cầu.
* Trợ lý ảo (Virtual Assistants/Chatbots): Các hệ thống có khả năng tương tác với người dùng thông qua ngôn ngữ tự nhiên để cung cấp thông tin, giải đáp thắc mắc hoặc thực hiện các tác vụ đơn giản.
* Phân tích tình cảm (Sentiment Analysis): Xác định cảm xúc, thái độ hoặc ý kiến mà người viết hoặc người nói thể hiện thông qua văn bản hoặc lời nói. Ứng dụng trong việc đánh giá phản hồi của khách hàng, theo dõi dư luận xã hội.
* Trích xuất thông tin (Information Extraction): Tự động nhận diện và trích xuất các thông tin quan trọng từ văn bản, giúp tổng hợp và phân tích dữ liệu hiệu quả.
* Tìm kiếm thông tin (Information Retrieval): Cải thiện khả năng tìm kiếm thông tin trên internet và các kho dữ liệu lớn bằng cách hiểu được ý định và ngữ cảnh của người dùng.
* Tóm tắt văn bản (Text Summarization): Tạo ra các bản tóm tắt ngắn gọn nhưng vẫn giữ được nội dung chính của các văn bản dài.
* Nhận dạng giọng nói (Speech Recognition) và Tổng hợp giọng nói (Text-to-Speech): Cho phép máy tính chuyển đổi giữa ngôn ngữ nói và ngôn ngữ viết, tạo ra các giao diện tương tác bằng giọng nói.

## Các mô hình học sâu xử lý ngôn ngữ tự nhiên :

Sự phát triển của học sâu (Deep Learning) đã mang lại những bước tiến vượt bậc trong lĩnh vực NLP. Các mô hình học sâu có khả năng tự động học các biểu diễn phức tạp của ngôn ngữ từ dữ liệu lớn, vượt trội hơn so với các phương pháp truyền thống dựa trên các đặc trưng được thiết kế thủ công. Một số kiến trúc mạng nơ-ron sâu (Deep Neural Networks - DNNs) phổ biến trong NLP bao gồm:

* Mạng nơ-ron hồi quy (Recurrent Neural Networks - RNNs): Đặc biệt phù hợp với việc xử lý các chuỗi dữ liệu tuần tự như ngôn ngữ, nơi mà thông tin ở các bước thời gian trước có thể ảnh hưởng đến các bước sau. Các biến thể của RNN như Mạng nơ-ron hồi quy dài-ngắn hạn (Long Short-Term Memory - LSTM) và Đơn vị hồi quy theo cổng (Gated Recurrent Unit - GRU) đã giải quyết vấn đề suy giảm gradient và cho phép mô hình học được các phụ thuộc dài hạn trong văn bản.
* Mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs): Ban đầu được phát triển cho thị giác máy tính, CNNs cũng được áp dụng thành công trong NLP, đặc biệt là trong các tác vụ như phân loại văn bản và trích xuất đặc trưng cục bộ.
* Kiến trúc Transformer: Đây là một đột phá lớn trong NLP, đặc biệt với cơ chế tự chú ý (Self-Attention), cho phép mô hình tập trung vào các phần quan trọng nhất của chuỗi đầu vào khi đưa ra dự đoán. Các mô hình dựa trên Transformer như BERT, GPT và các biến thể của chúng đã đạt được những kết quả ấn tượng trong nhiều tác vụ NLP và là nền tảng cho sự phát triển của các mô hình ngôn ngữ lớn (LLMs).

## Vấn đề của các mô hình xử lý ngôn ngữ tự nhiên truyền thống :

Trước sự trỗi dậy của học sâu, các mô hình NLP truyền thống thường dựa trên các phương pháp thống kê và học máy cổ điển, kết hợp với việc thiết kế thủ công các đặc trưng ngôn ngữ (hand-crafted features). Mặc dù có những thành công nhất định, các mô hình này gặp phải một số hạn chế đáng kể:

* Phụ thuộc vào đặc trưng được thiết kế thủ công: Việc trích xuất các đặc trưng ngôn ngữ phù hợp (ví dụ: TF-IDF, word embeddings tĩnh như Word2Vec, GloVe) đòi hỏi kiến thức chuyên môn sâu về ngôn ngữ học và tốn nhiều thời gian. Các đặc trưng này thường không thể nắm bắt được hết sự phức tạp và đa dạng của ngôn ngữ.
* Khó khăn trong việc xử lý các phụ thuộc dài hạn: Các mô hình như Hidden Markov Models (HMMs) và Conditional Random Fields (CRFs) gặp khó khăn trong việc mô hình hóa các mối quan hệ giữa các từ hoặc cụm từ cách xa nhau trong câu hoặc đoạn văn.
* Hạn chế trong việc hiểu ngữ cảnh: Các mô hình truyền thống thường xử lý từ và cụm từ một cách độc lập hoặc trong một phạm vi ngữ cảnh hạn chế, dẫn đến việc khó khăn trong việc hiểu được ý nghĩa thực sự của ngôn ngữ, đặc biệt là trong các trường hợp có sự mơ hồ hoặc cần suy luận.
* Khả năng khái quát hóa hạn chế: Các mô hình này thường hoạt động tốt trên tập dữ liệu huấn luyện cụ thể nhưng có thể gặp khó khăn khi được áp dụng vào các miền dữ liệu khác hoặc các tác vụ mới.

Sự ra đời của các mô hình học sâu đã giải quyết được nhiều vấn đề này, mang lại khả năng học biểu diễn tự động, xử lý hiệu quả các phụ thuộc dài hạn và hiểu ngữ cảnh tốt hơn, từ đó mở ra một kỷ nguyên mới cho lĩnh vực xử lý ngôn ngữ tự nhiên.

# PHƯƠNG PHÁP LUẬN

## Phát biểu bài toán:

Bài toán cốt lõi của nghiên cứu này là xây dựng một ứng dụng chatbot thông minh và hiệu quả nhằm hỗ trợ sinh viên trong quá trình học tập các môn chuyên ngành thuộc Khoa Công nghệ thông tin. Hiện tại, sinh viên thường gặp khó khăn trong việc nhanh chóng tìm kiếm, tổng hợp thông tin và giải đáp thắc mắc từ khối lượng lớn tài liệu bài giảng và sách tham khảo. Các công cụ tìm kiếm truyền thống có thể chưa đủ linh hoạt và không thể cung cấp câu trả lời ngữ cảnh theo dạng hội thoại. Hơn nữa, với đặc thù của kiến thức chuyên ngành và nhu cầu hỗ trợ đa ngôn ngữ, việc triển khai một giải pháp chatbot sử dụng các mô hình ngôn ngữ lớn (LLM) phù hợp cho tiếng Việt là cần thiết.

Để giải quyết vấn đề này, chúng tôi tập trung vào việc nghiên cứu và áp dụng mô hình ViSTRAL-7B-Chat – một LLM được tối ưu hóa đặc biệt cho tiếng Việt – làm nền tảng cho chatbot. Bài toán đặt ra là làm thế nào để tích hợp hiệu quả kiến thức chuyên ngành từ tài liệu bài giảng của Khoa Công nghệ thông tin vào mô hình, vượt qua hạn chế về bộ nhớ ngữ cảnh và khả năng "ảo giác" của LLM. Chúng tôi đề xuất sử dụng phương pháp Retrieval-Augmented Generation (RAG) để truy xuất thông tin chính xác từ bộ dữ liệu tài liệu đã được vector hóa, đảm bảo phản hồi của chatbot luôn dựa trên nguồn đáng tin cậy. Đồng thời, nghiên cứu cũng giải quyết việc tinh chỉnh mô hình (fine-tune) ViSTRAL-7B-Chat bằng phương pháp LoRA trên bộ dữ liệu chuyên ngành tự tạo, nhằm nâng cao khả năng hiểu và tạo sinh văn bản liên quan trực tiếp đến các môn học. Cuối cùng, bài toán bao gồm việc phát triển một hệ thống chatbot hoàn chỉnh với kiến trúc backend FastAPI và frontend Next.js, cung cấp giao diện thân thiện và hỗ trợ đa ngôn ngữ cho tài liệu, mang lại một công cụ học tập tương tác và hiệu quả cho sinh viên.

## Đánh giá và so sánh các mô hình ngôn ngữ lớn :

### Cấu hình thiết bị :

Do yêu cầu về tài nguyên tính toán lớn để thực hiện đánh giá các mô hình ngôn ngữ lớn (LLM), nhóm chúng tôi đã sử dụng dịch vụ điện toán đám mây được cung cấp bởi Kaggle. Môi trường này cung cấp cấu hình phần cứng mạnh mẽ, đáp ứng tốt nhu cầu chạy các thí nghiệm phức tạp.

Cấu hình chi tiết của môi trường đánh giá trên Kaggle được ghi nhận như sau:

|  |  |  |
| --- | --- | --- |
| Hệ điều hành | | Linux |
| Phiên bản Python | | 3.10.12 |
| CPU | Kiến trúc | x86\_64 |
| Số nhân (Cores) | 4 |
| GPU | Tên thiết bị | Tesla P100-PCIE-16GB |
| Bộ nhớ (VRAM) | 16GB |
| RAM | Tổng dung lượng | 33.66 GB |
| Khả dụng | 32.24 GB |
| Dung lượng lưu trữ | /dev/loop1 | Tổng 20.96 GB, khả dụng 20.94 GB |
| /dev/sda1 | Tổng 130.80 GB, khả dụng 125.86 GB |
| /dev/mapper/snap | Tổng 8.66 TB, khả dụng 2.03 TB |

Bảng 1 : Cấu hình chi tiết của môi trường đánh giá LLMs

### Tập dữ liệu VMLU :

Vietnamese Multitask Language Understanding (VMLU) là một bộ dữ liệu tiếng Việt được thiết kế để đánh giá khả năng hiểu ngôn ngữ đa nhiệm của các mô hình máy học. Bộ dữ liệu này là kết quả hợp tác nghiên cứu giữa Zalo AI và Viện Khoa học & Công nghệ Tiên Tiến Nhật Bản (JAIST).

VMLU bao gồm 10.880 câu hỏi trắc nghiệm, trải rộng trên 58 chủ đề riêng biệt. Đa phần, mỗi chủ đề chứa khoảng 200 câu hỏi, được phân loại thành bốn lĩnh vực chính: STEM, Khoa học Xã hội (Social Sciences), Nhân văn (Humanities) và các lĩnh vực khác (other). Các câu hỏi trong VMLU có độ khó đa dạng, từ trình độ cơ bản đến chuyên sâu, nhằm thách thức cả kiến thức chung và khả năng giải quyết vấn đề phức tạp của mô hình.

Dữ liệu trong bộ VMLU được thu thập từ các kỳ thi chất lượng cao do các tổ chức giáo dục uy tín tổ chức, bao gồm các cấp học từ tiểu học đến đại học. Ngoài ra, bộ dữ liệu còn sử dụng một phần dữ liệu từ các kỳ thi tốt nghiệp Trung học Phổ thông, được tổ chức và giám sát bởi Bộ Giáo dục và Đào tạo. Điều này đảm bảo tính đa dạng và độ tin cậy của bộ dữ liệu.

Bộ dữ liệu VMLU được chia thành bốn thư mục nhỏ, mỗi thư mục tập trung vào một nhiệm vụ cụ thể:

* Vi-MQA (Vietnamese Multiple-choice Question Answering): Tập dữ liệu cho nhiệm vụ trả lời câu hỏi trắc nghiệm tiếng Việt.
* Vi-Dialog (Vietnamese Dialogue Dataset): Tập dữ liệu đối thoại tiếng Việt.
* Vi-Drop (Vietnamese Discrete Reasoning Over Paragraphs): Tập dữ liệu cho nhiệm vụ lý luận rời rạc trên các đoạn văn tiếng Việt.
* Vi-SQuAD (Vietnamese Stanford Question Answering Dataset): Tập dữ liệu trả lời câu hỏi theo định dạng Stanford Question Answering Dataset, được chuyển ngữ sang tiếng Việt.

### Phương pháp đánh giá :

Trong khuôn khổ bài luận này, nhóm nghiên cứu đã lựa chọn thư mục Vi-MQA để tiến hành đánh giá và so sánh hiệu suất của các mô hình ngôn ngữ lớn. Cụ thể, chúng tôi sẽ sử dụng tập tin **valid.jsonl** trong thư mục này.

Mỗi dòng trong tập tin **valid.jsonl** chứa thông tin về một câu hỏi, bao gồm các trường sau:

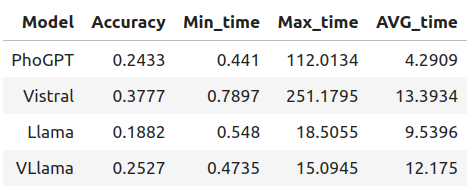
* **id**: Mã định danh của câu hỏi.
* **question**: Nội dung câu hỏi.
* **choices**: Danh sách các lựa chọn đáp án.
* **answer**: Đáp án đúng cho câu hỏi.

{"id": "28-0007", "question": "Hoạt động nào sau đây của ngân hàng Trung Ương sẽ làm tăng cơ sở tiền tệ", "choices": ["A. Bán ngoại tệ trên thị trường ngoại hối", "B. Cho các ngân hàng thương mại vay", "C. Hạ tỷ lệ dự trữ bắt buộc đối với các ngân hàng thương mại", "D. Tăng lãi suất chiết khấu"], "answer": "C"}

Quy trình đánh giá các mô hình ngôn ngữ lớn được thực hiện theo các bước sau:

1. Lặp qua danh sách mô hình: Tiến hành đánh giá lần lượt từng mô hình trong danh sách các mô hình được so sánh.
2. Đánh giá trên toàn bộ tập dữ liệu: Với mỗi mô hình, thực hiện đánh giá trên toàn bộ các câu hỏi có trong tập tin **valid.jsonl**.
3. Sử dụng mẫu lệnh phù hợp: Do mỗi mô hình có thể yêu cầu định dạng mẫu lệnh (prompt template) khác nhau, chúng tôi sẽ có sự điều chỉnh nhỏ để đảm bảo tính tương thích và khách quan của quá trình kiểm thử.
4. Mô hình đưa ra dự đoán: Mỗi mô hình sẽ nhận câu hỏi và danh sách các đáp án. Nhiệm vụ của mô hình là phân tích câu hỏi và chọn ra đáp án đúng từ các lựa chọn được cung cấp.
5. So sánh với đáp án chuẩn: Kết quả dự đoán của mô hình sẽ được so sánh với đáp án chính xác (ground truth) được cung cấp trong dữ liệu.
6. Ghi nhận kết quả: Nếu dự đoán của mô hình trùng khớp với đáp án chuẩn, câu hỏi đó sẽ được đánh dấu là trả lời đúng trong file kết quả (định dạng .csv).
7. Tính toán các chỉ số đánh giá: Dựa trên số lượng câu trả lời đúng, chúng tôi sẽ tính toán độ chính xác (accuracy) của từng mô hình. Bên cạnh đó, chúng tôi cũng sẽ đo lường và phân tích thời gian mà mỗi mô hình cần để đưa ra câu trả lời (thời gian tối thiểu, tối đa và trung bình).

### Kết quả đánh giá :



Hình : Bảng đánh giá các mô hình LLMs

***Chú thích :***

Các số được làm tròn 4 giá trị sau chữ số thập phân

Để tiện theo dõi và tham khảo trong bài luận, chúng tôi sử dụng các ký hiệu viết tắt sau:

* Các mô hình ngôn ngữ lớn (LLMs):
  + PhoGPT: Đại diện cho mô hình vinai/PhoGPT-4B-Chat.
  + Vistral: Đại diện cho mô hình Viet-Mistral/Vistral-7B-Chat.
  + Llama: Đại diện cho mô hình meta-llama/Llama-2-7b-hf.
  + VLlama: Đại diện cho mô hình bkai-foundation-models/vietnamese-llama2-7b-120GB.
* Các chỉ số thời gian:
  + Min\_time: Khoảng thời gian ngắn nhất mà một mô hình mất để tạo ra câu trả lời cho một câu hỏi.
  + Max\_time: Khoảng thời gian dài nhất mà một mô hình cần để tạo ra câu trả lời cho một câu hỏi.
  + AVG\_time: Thời gian trung bình mà một mô hình cần để tạo ra câu trả lời trên toàn bộ tập dữ liệu đánh giá.

### Phân tích kết quả :

Từ bảng kết quả, có thể thấy mô hình **Vistral** đạt độ chính xác cao nhất với tỷ lệ **0.3777**. Tuy nhiên, mô hình này cũng có thời gian phản hồi trung bình (**AVG\_time**) đáng kể là **13.3934 giây**. Đây là một khoảng thời gian tương đối dài, có thể gây ảnh hưởng đến trải nghiệm người dùng nếu ứng dụng đòi hỏi phản hồi nhanh chóng.

### Định hướng cải thiện :

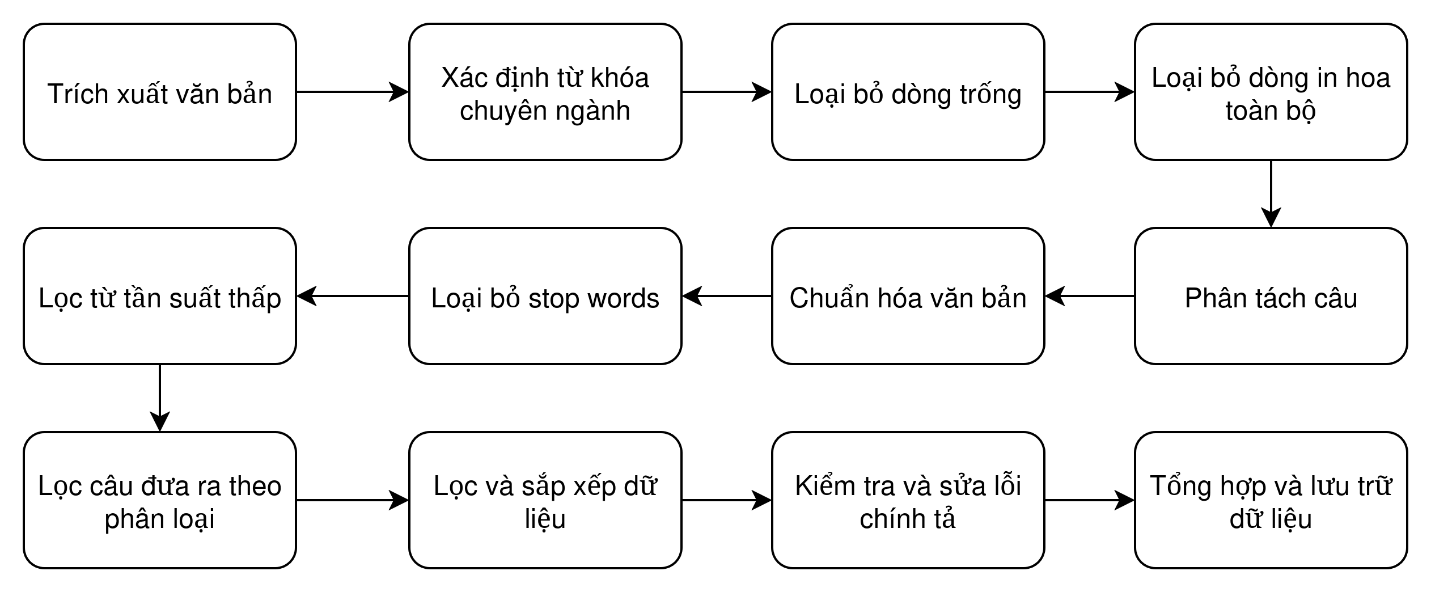
Nhận thấy sự đánh đổi giữa độ chính xác và thời gian phản hồi ở mô hình Vistral, nhóm nghiên cứu dự định sẽ thực hiện quá trình tinh chỉnh (fine-tune) mô hình này. Mục tiêu của việc tinh chỉnh là tối ưu hóa mô hình để vừa duy trì được độ chính xác cao, vừa giảm thiểu thời gian phản hồi xuống mức chấp nhận được đối với người dùng.

## Chuẩn bị bộ dữ liệu huấn luyện mô hình :

Trong giai đoạn chuẩn bị bộ dữ liệu huấn luyện, mục tiêu chính của chúng tôi là xây dựng một corpus[[1]](#footnote-1) chất lượng cao, đóng vai trò nền tảng cho quá trình tinh chỉnh (finetuning[[2]](#footnote-2)) mô hình ngôn ngữ lớn (LLM) đã được huấn luyện sơ bộ (pre-trained[[3]](#footnote-3)) trên thư viện Hugging Face. Nguồn dữ liệu ban đầu được trích xuất từ các tài liệu chuyên ngành, cụ thể là cuốn sách Java 8 in Action và Data-warehousing-fundamentals-for-it-professionals Second Edition.

Việc lựa chọn dữ liệu từ các nguồn sách này nhằm mục đích trang bị cho mô hình một lượng kiến thức tổng quát và đa dạng. Chúng tôi kỳ vọng rằng, thông qua việc tiếp thu tri thức nền tảng từ các tài liệu này, mô hình sẽ có khả năng xử lý tốt hơn các truy vấn đa dạng từ sinh viên, thay vì chỉ tập trung vào các mẫu câu hỏi cụ thể, dễ dẫn đến hiện tượng quá khớp (overfitting[[4]](#footnote-4)).

Do đó, chiến lược của chúng tôi là ưu tiên huấn luyện mô hình trên tập dữ liệu tổng quát ban đầu này. Sau đó, dựa trên nền tảng kiến thức đã được усвоено, chúng tôi sẽ tiến hành tinh chỉnh (finetune) mô hình bằng cách sử dụng dữ liệu đặc thù được trích xuất từ cơ sở dữ liệu của trường. Cách tiếp cận này nhằm mục đích dung hòa giữa khả năng hiểu biết rộng rãi và khả năng cung cấp thông tin chi tiết, chính xác cho người dùng.

Chi tiết quá trình tiền xử lý dữ liệu và tạo ra bộ dữ liệu huấn luyện được mô tả qua các bước dưới đây:  


### Trích xuất văn bản từ file PDF :

Phương pháp được đề xuất để trích xuất nội dung văn bản từ tài liệu định dạng PDF được hiện thực hóa thông qua hàm *extract\_text\_from\_pdf*. Hàm này được thiết kế để linh hoạt trong việc lựa chọn nội dung cần trích xuất bằng cách cho phép người dùng chỉ định số lượng trang cần bỏ qua ở đầu và cuối tài liệu, đồng thời giới hạn số lượng ký tự được trích xuất.

Đầu vào (Input):

Hàm *extract\_text\_from\_pdf* nhận các tham số sau:

* pdf\_path (kiểu dữ liệu: str): Đường dẫn tuyệt đối hoặc tương đối đến tệp tin PDF cần trích xuất nội dung.
* skip\_start (kiểu dữ liệu: int, giá trị mặc định: 0): Số lượng trang ở đầu tài liệu PDF cần bỏ qua trước khi bắt đầu trích xuất văn bản. Tham số này hữu ích trong trường hợp tài liệu có các trang giới thiệu, mục lục không cần thiết cho quá trình huấn luyện mô hình.
* skip\_end (kiểu dữ liệu: int, giá trị mặc định: 0): Số lượng trang ở cuối tài liệu PDF cần bỏ qua. Tham số này thường được sử dụng để loại trừ các trang phụ lục, tài liệu tham khảo.
* max\_chars (kiểu dữ liệu: float, giá trị mặc định: -1): Tham số này quy định số lượng ký tự tối đa được trích xuất từ tài liệu. Các tùy chọn cho tham số này bao gồm:
  + -1: Trích xuất toàn bộ nội dung văn bản từ các trang đã chọn.
  + 0 < max\_chars < 1: Trích xuất một tỷ lệ phần trăm tương ứng của tổng số ký tự có trong các trang đã chọn. Ví dụ, 0.5 sẽ trích xuất 50% số ký tự.
  + max\_chars >= 1: Trích xuất chính xác số lượng ký tự được chỉ định.

Đầu ra (Output):

Hàm trả về một chuỗi (str) chứa nội dung văn bản được trích xuất từ các trang của tệp PDF, sau khi đã bỏ qua các trang đầu và cuối theo yêu cầu, và giới hạn số lượng ký tự (nếu được chỉ định). Các trang được phân tách bằng ký tự xuống dòng (\n).

Quá trình thực hiện:

Quá trình trích xuất văn bản được thực hiện theo các bước sau:

* Mở tệp PDF: Sử dụng thư viện fitz (PyMuPDF) để mở tệp PDF được chỉ định bởi tham số pdf\_path.
* Xác định tổng số trang: Lấy tổng số trang có trong tài liệu PDF.
* Kiểm tra số lượng trang bỏ qua: Đảm bảo rằng tổng số trang cần bỏ qua ở đầu (skip\_start) và cuối (skip\_end) không vượt quá tổng số trang của tài liệu. Nếu điều này xảy ra, hàm sẽ trả về lỗi ValueError.
* Duyệt qua các trang hợp lệ: Sử dụng vòng lặp để duyệt qua các trang của tài liệu, bắt đầu từ trang có chỉ số skip\_start và kết thúc ở trang có chỉ số total\_pages - skip\_end - 1.
* Trích xuất văn bản từ từng trang: Với mỗi trang trong phạm vi duyệt, sử dụng phương thức get\_text() của đối tượng trang để lấy nội dung văn bản. Nội dung văn bản từ mỗi trang được thêm vào biến extracted\_text, kèm theo ký tự xuống dòng để phân tách nội dung giữa các trang.
* Xử lý giới hạn số lượng ký tự: Sau khi trích xuất toàn bộ văn bản từ các trang đã chọn, hàm tiến hành kiểm tra và xử lý tham số max\_chars:
* Nếu max\_chars là -1 hoặc lớn hơn hoặc bằng tổng số ký tự đã trích xuất, hàm sẽ trả về toàn bộ nội dung extracted\_text.
* Nếu max\_chars nằm trong khoảng (0, 1), giá trị này được hiểu là tỷ lệ phần trăm. Hàm sẽ tính toán số lượng ký tự tương ứng và chỉ trả về phần đầu của chuỗi extracted\_text với độ dài đã tính toán.
* Nếu max\_chars lớn hơn hoặc bằng 1, giá trị này được hiểu là số lượng ký tự cụ thể. Hàm sẽ trả về phần đầu của chuỗi extracted\_text với độ dài bằng max\_chars (sau khi đã được chuyển đổi sang kiểu số nguyên).
* Trả về văn bản đã trích xuất: Hàm trả về chuỗi extracted\_text chứa nội dung văn bản đã được trích xuất và xử lý theo các tham số đầu vào.

Phương pháp này cung cấp một cách tiếp cận hiệu quả và linh hoạt để thu thập dữ liệu văn bản từ các tài liệu PDF, cho phép tùy chỉnh phạm vi và dung lượng dữ liệu trích xuất, phù hợp với yêu cầu của các tác vụ huấn luyện mô hình ngôn ngữ lớn.

### Xác định từ khóa :

Trong quá trình chuẩn bị dữ liệu huấn luyện, một bước quan trọng là xác định và tập hợp các từ khóa chuyên ngành liên quan đến lĩnh vực kiến thức của tài liệu nguồn. Tập hợp các từ khóa này được lưu trữ trong một cấu trúc dữ liệu dạng mảng, ký hiệu là DOMAIN\_KEYWORDS. Mục đích chính của việc xây dựng tập hợp này là nhằm bảo toàn tính chính xác và ngữ nghĩa của các thuật ngữ chuyên môn trong quá trình tiền xử lý văn bản, đặc biệt là trong giai đoạn kiểm tra và sửa lỗi chính tả.

Nguồn gốc của các từ khóa trong DOMAIN\_KEYWORDS được tổng hợp từ nhiều nguồn khác nhau để đảm bảo tính đầy đủ và bao quát:

* Bảng chú thích thuật ngữ của sách: Các thuật ngữ quan trọng được định nghĩa và liệt kê trong phần chú thích thuật ngữ (glossary) của các cuốn sách Java 8 in Action và Data-warehousing-fundamentals-for-it-professionals Second Edition đã được thu thập và đưa vào mảng.
* Kiến thức nền tảng: Các thuật ngữ và khái niệm cơ bản trong lĩnh vực công nghệ thông tin và khoa học dữ liệu, được tích lũy từ kiến thức chuyên môn, cũng được bổ sung vào tập hợp từ khóa.
* Tham khảo từ chatbot GPT: Để mở rộng phạm vi bao phủ của các từ khóa, một số thuật ngữ liên quan đã được tham khảo và đề xuất từ mô hình ngôn ngữ lớn GPT, đảm bảo tính cập nhật và toàn diện của tập hợp.

Mục tiêu của việc duy trì danh sách DOMAIN\_KEYWORDS là để ngăn chặn việc mô hình kiểm tra và sửa lỗi chính tả thực hiện các thay đổi không chính xác đối với các thuật ngữ chuyên ngành. Việc sửa đổi như vậy có thể dẫn đến sự sai lệch về ý nghĩa và làm giảm chất lượng của dữ liệu huấn luyện.

Trong quy trình tiền xử lý, trước khi áp dụng bất kỳ thuật toán sửa lỗi chính tả nào, hệ thống sẽ thực hiện so sánh các từ trong văn bản với các phần tử trong mảng DOMAIN\_KEYWORDS. Nếu một từ trùng khớp với một từ khóa trong danh sách, nó sẽ được loại trừ khỏi quá trình sửa lỗi chính tả. Ngược lại, chỉ những từ không nằm trong danh sách DOMAIN\_KEYWORDS mới được xem xét để thực hiện các điều chỉnh chính tả, đảm bảo rằng các thuật ngữ chuyên môn được giữ nguyên vẹn.

### Tiền xử lý dữ liệu :

Tiếp theo giai đoạn trích xuất văn bản thô từ tài liệu PDF, bước quan trọng kế tiếp trong quy trình chuẩn bị dữ liệu là tiến hành các hoạt động tiền xử lý dữ liệu. Giai đoạn này được quản lý bởi phương thức preprocess\_text, đóng vai trò là trung tâm điều phối, gọi đến các phương thức chuyên biệt hơn để thực hiện từng tác vụ tiền xử lý cụ thể. Để tối ưu hóa hiệu suất và quản lý tài nguyên hiệu quả, đặc biệt với các bộ dữ liệu lớn, phương thức preprocess\_text được thiết kế để xử lý dữ liệu theo từng lô (batch) có kích thước xác định. Cách tiếp cận này không chỉ giúp giảm thiểu mức tiêu thụ bộ nhớ mà còn tạo điều kiện thuận lợi cho việc áp dụng các kỹ thuật xử lý song song, từ đó tăng tốc đáng kể quá trình tiền xử lý. Mục tiêu xuyên suốt của giai đoạn này vẫn là tinh chỉnh, chuẩn hóa dữ liệu văn bản, loại bỏ các yếu tố nhiễu và định dạng không phù hợp, nhằm đảm bảo chất lượng cao nhất cho bộ dữ liệu huấn luyện (dataset) cuối cùng, từ đó tối ưu hóa hiệu suất và độ chính xác của mô hình ngôn ngữ lớn trong các giai đoạn huấn luyện và ứng dụng tiếp theo.

### Loại bỏ dòng trống :

Một trong những bước quan trọng trong quá trình tiền xử lý dữ liệu văn bản là loại bỏ các dòng trống hoặc các dòng chỉ chứa ký tự khoảng trắng. Những dòng này không mang lại thông tin hữu ích cho quá trình huấn luyện mô hình và có thể gây nhiễu loạn cho quá trình xử lý tiếp theo. Để thực hiện tác vụ này, chúng tôi sử dụng hàm remove\_empty\_lines.

Đầu vào (Input):

Hàm remove\_empty\_lines nhận một tham số duy nhất:

* lines (kiểu dữ liệu: List[str]): Một danh sách các chuỗi, trong đó mỗi chuỗi đại diện cho một dòng văn bản được trích xuất từ tài liệu.

Đầu ra (Output):

Hàm trả về một danh sách các chuỗi (List[str]) đã được lọc. Danh sách này chỉ chứa các dòng văn bản không phải là dòng trống hoặc không chỉ chứa các ký tự khoảng trắng.

Quá trình thực hiện:

Hàm remove\_empty\_lines thực hiện việc lọc các dòng trống theo quy trình sau:

* Duyệt qua danh sách dòng: Hàm sử dụng một biểu thức comprehension danh sách (list comprehension) để duyệt qua từng chuỗi line trong danh sách đầu vào lines.
* Kiểm tra tính không trống: Đối với mỗi dòng, phương thức strip() được gọi để loại bỏ các khoảng trắng ở đầu và cuối chuỗi. Nếu sau khi loại bỏ khoảng trắng, chuỗi vẫn còn chứa ít nhất một ký tự khác khoảng trắng, điều đó có nghĩa là dòng đó không phải là dòng trống.
* Tạo danh sách mới: Chỉ những dòng thỏa mãn điều kiện không trống (sau khi đã loại bỏ khoảng trắng) mới được thêm vào danh sách kết quả.
* Trả về danh sách đã lọc: Hàm trả về danh sách mới chứa các dòng văn bản đã được lọc, loại bỏ các dòng trống hoặc chỉ chứa khoảng trắng.

Việc loại bỏ các dòng trống giúp làm sạch dữ liệu, giảm kích thước của bộ dữ liệu và tập trung vào các thông tin có giá trị, từ đó cải thiện hiệu quả của quá trình huấn luyện mô hình ngôn ngữ.

### Loại bỏ dòng in hoa toàn bộ :

Một bước tiền xử lý quan trọng khác là loại bỏ các dòng văn bản mà tất cả các ký tự (ngoại trừ dấu câu và khoảng trắng) đều là chữ in hoa. Thông thường, các dòng này thường là tiêu đề, đề mục hoặc các phần tử trang trí khác, chứa ít thông tin có giá trị cho việc huấn luyện mô hình ngôn ngữ. Ví dụ, trong đoạn văn sau:

*“ACKNOWLEDGMENTS*

*I wish to acknowledge my indebtedness and to express my gratitude to the authors listed in the reference section at the end of the book. Their insights and observations have helped me every topic adequately.”*

Dòng "ACKNOWLEDGMENTS" chủ yếu mang tính hình thức và không đóng góp đáng kể vào việc cung cấp ngữ cảnh hoặc thông tin chi tiết cho mô hình học. Do đó, việc loại bỏ các dòng này giúp tập trung dữ liệu huấn luyện vào nội dung chính. Hàm remove\_all\_uppercase\_lines được sử dụng để thực hiện tác vụ này.

Đầu vào (Input):

Hàm remove\_all\_uppercase\_lines nhận một tham số:

* lines (kiểu dữ liệu: List[str]): Một danh sách các chuỗi, mỗi chuỗi đại diện cho một dòng văn bản.

Đầu ra (Output):

Hàm trả về một danh sách các chuỗi (List[str]) đã được lọc, trong đó các dòng mà tất cả các ký tự (sau khi loại bỏ dấu câu và khoảng trắng) đều là chữ in hoa đã bị loại bỏ.

Quá trình thực hiện:

* Hàm remove\_all\_uppercase\_lines thực hiện việc lọc các dòng in hoa toàn bộ theo các bước sau:
* Định nghĩa hàm kiểm tra is\_all\_upper: Một hàm nội bộ is\_all\_upper được định nghĩa để kiểm tra xem một dòng văn bản có phải là dòng in hoa toàn bộ hay không.
* Loại bỏ dấu câu và khoảng trắng: Trong hàm is\_all\_upper, biểu thức chính quy (regex) r'[{} \t\n\r]'.format(re.escape(string.punctuation)) được sử dụng để loại bỏ tất cả các dấu câu, khoảng trắng, tab, ký tự xuống dòng và ký tự carriage return khỏi dòng văn bản. Hàm re.escape được sử dụng để đảm bảo các ký tự dấu câu được xử lý đúng cách trong biểu thức chính quy.
* Kiểm tra chữ in hoa: Sau khi đã loại bỏ các ký tự không phải chữ cái, hàm kiểm tra xem chuỗi kết quả có rỗng hay không (để tránh trường hợp dòng chỉ chứa dấu câu và khoảng trắng) và sau đó sử dụng phương thức isupper() để xác định xem tất cả các ký tự còn lại trong chuỗi có phải là chữ in hoa hay không.
* Lọc danh sách dòng: Hàm chính remove\_all\_uppercase\_lines sử dụng một biểu thức comprehension danh sách để duyệt qua từng dòng trong danh sách đầu vào lines. Đối với mỗi dòng, hàm is\_all\_upper được gọi. Nếu hàm này trả về False (nghĩa là dòng không phải là dòng in hoa toàn bộ), dòng đó sẽ được giữ lại trong danh sách kết quả.
* Trả về danh sách đã lọc: Hàm trả về danh sách mới chứa các dòng văn bản đã được lọc, loại bỏ các dòng in hoa toàn bộ.

Bằng cách loại bỏ các dòng chữ in hoa toàn bộ, chúng ta có thể tập trung vào nội dung chính của văn bản, cải thiện chất lượng dữ liệu huấn luyện và giảm thiểu sự ảnh hưởng của các yếu tố không mang tính thông tin.

### Phân tách câu :

Sau khi đã thực hiện các bước làm sạch dữ liệu ở mức độ dòng, bước tiếp theo là phân tách văn bản thành các đơn vị nhỏ hơn, cụ thể là các câu. Việc phân tách câu (sentence tokenization) là một bước quan trọng trong quá trình chuẩn bị dữ liệu cho các tác vụ xử lý ngôn ngữ tự nhiên, vì nó giúp mô hình hiểu được cấu trúc ngữ pháp và ngữ nghĩa của văn bản ở cấp độ câu.

Phương pháp tiếp cận đơn giản là sử dụng dấu chấm câu (".") làm dấu hiệu kết thúc câu có thể không hiệu quả trong nhiều trường hợp, đặc biệt là đối với văn bản tiếng Anh. Ví dụ, các từ viết tắt như "Mr.", "Mrs.", "Dr." chứa dấu chấm nhưng không đánh dấu sự kết thúc của một câu. Để giải quyết vấn đề này, chúng tôi sử dụng hàm tokenize\_sentences, tận dụng sức mạnh của thư viện Natural Language Toolkit (NLTK). Cụ thể, hàm này sử dụng hàm sent\_tokenize của NLTK, một công cụ mạnh mẽ được thiết kế để xác định ranh giới câu một cách chính xác, ngay cả trong những trường hợp phức tạp.

Đầu vào (Input):

Hàm tokenize\_sentences nhận một tham số:

* text (kiểu dữ liệu: str): Chuỗi văn bản đã được làm sạch ở các bước trước đó.

Đầu ra (Output):

Hàm trả về một danh sách các chuỗi (List[str]), trong đó mỗi chuỗi là một câu được phân tách từ văn bản đầu vào.

Quá trình thực hiện:

Hàm tokenize\_sentences thực hiện việc phân tách câu theo quy trình sau:

* Sử dụng nltk.sent\_tokenize: Hàm này trực tiếp gọi hàm sent\_tokenize từ thư viện NLTK.
* Phân tích cú pháp: Hàm sent\_tokenize sử dụng các quy tắc và mô hình ngôn ngữ được huấn luyện để phân tích cấu trúc của văn bản và xác định vị trí kết thúc của mỗi câu. Nó có khả năng xử lý các trường hợp phức tạp như dấu chấm trong từ viết tắt, dấu chấm hỏi, dấu chấm than và các ký tự kết thúc câu khác một cách chính xác.
* Trả về danh sách câu: Kết quả của quá trình phân tích là một danh sách các chuỗi, mỗi chuỗi đại diện cho một câu hoàn chỉnh được trích xuất từ văn bản đầu vào.

Việc sử dụng thư viện NLTK giúp đảm bảo rằng quá trình phân tách câu được thực hiện một cách chính xác và đáng tin cậy, tạo nền tảng vững chắc cho các bước xử lý ngôn ngữ tự nhiên tiếp theo trong quy trình xây dựng bộ dữ liệu huấn luyện.

### Chuẩn hóa văn bản :

Sau quá trình phân tách câu, bước tiếp theo trong tiền xử lý dữ liệu là chuẩn hóa văn bản. Mục đích của bước này là loại bỏ các ký tự không cần thiết như dấu câu, ký tự đặc biệt ngoài bảng mã ASCII cơ bản và các biểu tượng cảm xúc (emojis), nhằm làm sạch văn bản và giảm sự phức tạp không cần thiết cho mô hình ngôn ngữ. Ví dụ, xét đoạn văn bản sau:

*“This chapter covers*

* *Why Java is changing again*
* *Changing computing background: multicore and processing large datasets (big data)*
* *Pressure to evolve: new architectures favor functional style over imperative*
* *Introducing core new features of Java 8: lambdas, streams, default methods”*

Trong đoạn văn trên, các ký tự như dấu chấm câu, dấu hai chấm và đặc biệt là ký tự "·" không trực tiếp đóng góp vào ý nghĩa ngữ cảnh của các câu và có thể được loại bỏ để đơn giản hóa dữ liệu. Hàm remove\_punctuation được sử dụng để thực hiện quá trình chuẩn hóa này.

Đầu vào (Input):

Hàm remove\_punctuation nhận một tham số:

* text (kiểu dữ liệu: str): Chuỗi văn bản đầu vào cần được chuẩn hóa.

Đầu ra (Output):

Hàm trả về một chuỗi (str) là phiên bản đã được làm sạch của văn bản đầu vào, trong đó tất cả dấu câu, ký tự đặc biệt ngoài bảng mã ASCII cơ bản và emojis đã bị loại bỏ.

Quá trình thực hiện:

Hàm remove\_punctuation thực hiện quá trình chuẩn hóa văn bản qua ba giai đoạn chính:

* Loại bỏ dấu câu tiêu chuẩn: Sử dụng phương thức str.maketrans để tạo một bảng ánh xạ, trong đó tất cả các ký tự dấu câu được định nghĩa trong string.punctuation sẽ được ánh xạ tới None (tức là bị xóa). Sau đó, phương thức translate được áp dụng để thực hiện việc loại bỏ các dấu câu này khỏi chuỗi văn bản.
* Loại bỏ ký tự đặc biệt ngoài bảng mã ASCII cơ bản: Biểu thức chính quy r'[^\x00-\x7F]+' được sử dụng để tìm kiếm và loại bỏ tất cả các ký tự không nằm trong phạm vi mã ASCII từ 0 đến 127. Phạm vi này bao gồm các ký tự đặc biệt như ký hiệu tiền tệ, ký tự toán học, các dấu gạch đầu dòng không chuẩn, v.v.
* Loại bỏ emojis: Một biểu thức chính quy phức tạp hơn được định nghĩa để nhận diện và loại bỏ các emojis dựa trên các khoảng mã Unicode đặc trưng của chúng. Các khoảng mã này bao gồm nhiều loại biểu tượng cảm xúc và hình vẽ phổ biến. Hàm re.compile được sử dụng để biên dịch biểu thức chính quy này để tăng hiệu suất, và sau đó phương thức sub được gọi để thay thế tất cả các emojis tìm thấy bằng một chuỗi rỗng.

Việc loại bỏ các ký tự không cần thiết này giúp tạo ra một tập dữ liệu văn bản sạch hơn, dễ xử lý hơn cho các bước tiếp theo trong quá trình huấn luyện mô hình ngôn ngữ, đồng thời có thể cải thiện hiệu suất của mô hình bằng cách giảm thiểu sự phân tán và tập trung vào các thành phần ngôn ngữ có ý nghĩa.

### Loại bỏ stop words :

Trong quá trình tiền xử lý ngôn ngữ tự nhiên, một bước thường được thực hiện là loại bỏ các "stop words". Stop words là các từ xuất hiện với tần suất cao trong văn bản nhưng lại ít đóng góp vào ý nghĩa ngữ nghĩa của câu. Chúng thường được coi là các từ chức năng hoặc từ nối, và việc giữ lại chúng có thể tạo ra nhiễu trong quá trình huấn luyện mô hình, làm tăng kích thước của bộ từ vựng mà không mang lại lợi ích tương ứng về mặt thông tin. Ví dụ điển hình của stop words trong tiếng Anh bao gồm "the", "is", "at", "on", "which", "in", "some", "many", và trong tiếng Việt là "cái", "các", "cả", "thì", "mà". Việc loại bỏ stop words giúp mô hình tập trung hơn vào các từ khóa thực sự mang ý nghĩa cho từng câu.

Để thực hiện tác vụ này, chúng tôi sử dụng hàm remove\_stopwords\_from\_sentence. Thư viện NLTK (Natural Language Toolkit) cung cấp một danh sách các stop words phổ biến cho nhiều ngôn ngữ, bao gồm cả tiếng Anh. Tuy nhiên, tùy thuộc vào đặc thù của bài toán và tập dữ liệu, danh sách này có thể được điều chỉnh bằng cách thêm hoặc bớt các từ cho phù hợp. Để sử dụng danh sách stop words của NLTK, cần tải xuống bộ dữ liệu tương ứng bằng lệnh nltk.download('stopwords') trước khi sử dụng.

Đầu vào (Input):

Hàm remove\_stopwords\_from\_sentence nhận một tham số:

* sentence (kiểu dữ liệu: str): Chuỗi văn bản đầu vào, đại diện cho một câu đã được phân tách.

Đầu ra (Output):

Hàm trả về một chuỗi (str) là câu đã được xử lý, trong đó các stop words đã bị loại bỏ.

Quá trình thực hiện:

Hàm remove\_stopwords\_from\_sentence thực hiện việc loại bỏ stop words theo các bước sau:

* Token hóa câu: Câu đầu vào được chia thành các từ riêng lẻ (tokens) bằng cách sử dụng hàm word\_tokenize (thường cũng được cung cấp bởi thư viện NLTK).
* Lọc các token: Một danh sách mới filtered\_tokens được tạo ra bằng cách duyệt qua từng token trong danh sách đã token hóa. Đối với mỗi token, nó được chuyển về dạng chữ thường (token.lower()) và so sánh với tập hợp các stop words (STOPWORDS). Nếu token không nằm trong tập hợp stop words, nó sẽ được giữ lại trong danh sách filtered\_tokens.
* Kết hợp các token đã lọc: Cuối cùng, các token còn lại trong danh sách filtered\_tokens được kết hợp lại thành một chuỗi duy nhất, với các từ được phân tách bằng dấu cách, và chuỗi này được trả về làm kết quả của hàm.

Bằng việc loại bỏ các stop words, chúng ta có thể giảm bớt sự nhiễu loạn trong dữ liệu văn bản, làm nổi bật các từ khóa quan trọng và có khả năng cải thiện hiệu suất của c

ác mô hình học máy trong các tác vụ như phân loại văn bản, trích xuất thông tin hoặc xây dựng các hệ thống hỏi đáp.

### Lọc từ tần suất thấp :

Bên cạnh việc loại bỏ các stop words, trong nhiều ứng dụng xử lý ngôn ngữ tự nhiên, đặc biệt là khi làm việc với các bộ dữ liệu lớn, việc loại bỏ các từ xuất hiện với tần suất thấp cũng là một kỹ thuật được cân nhắc. Các từ hiếm gặp có thể là các từ lỗi chính tả, các thực thể (entities) ít quan trọng trong ngữ cảnh chung, hoặc đơn giản là các từ không phổ biến. Việc loại bỏ chúng có thể giúp giảm kích thước của bộ từ vựng, tăng cường khả năng khái quát hóa của mô hình và giảm thiểu ảnh hưởng của các từ nhiễu. Tuy nhiên, cần lưu ý rằng việc loại bỏ các từ hiếm có thể dẫn đến mất mát thông tin ngữ nghĩa, đặc biệt là trong các bộ dữ liệu có kích thước trung bình hoặc nhỏ, nơi mỗi từ có thể mang một ý nghĩa đặc trưng.

Để thực hiện việc lọc các từ có tần suất xuất hiện thấp trong một lô (batch) các câu, chúng tôi sử dụng hàm remove\_rare\_words\_from\_batch. Hàm này cho phép xác định ngưỡng tần suất theo hai cách: một giá trị số nguyên tuyệt đối (loại bỏ các từ xuất hiện ít hơn ngưỡng này) hoặc một giá trị số thực trong khoảng (0, 1) biểu thị tần suất tương đối (loại bỏ các từ có tần suất xuất hiện dưới ngưỡng này so với tổng số từ).

Đầu vào (Input):

Hàm remove\_rare\_words\_from\_batch nhận hai tham số:

* sentences (kiểu dữ liệu: List[str]): Một danh sách các chuỗi, trong đó mỗi chuỗi đại diện cho một câu trong lô dữ liệu.
* threshold (kiểu dữ liệu: float | int): Ngưỡng tần suất để xác định các từ hiếm.
  + Nếu là số nguyên, các từ xuất hiện ít hơn threshold lần sẽ bị loại bỏ.
  + Nếu là số thực (lớn hơn 0 và nhỏ hơn 1), các từ có tần suất tương đối (số lần xuất hiện chia cho tổng số từ) nhỏ hơn threshold sẽ bị loại bỏ.

Đầu ra (Output):

Hàm trả về một danh sách các chuỗi (List[str]), là các câu trong lô dữ liệu sau khi các từ có tần suất xuất hiện thấp (dựa trên ngưỡng đã cho) đã bị loại bỏ.

Quá trình thực hiện:

Hàm remove\_rare\_words\_from\_batch thực hiện việc lọc các từ hiếm theo các bước sau:

* Token hóa toàn bộ lô câu: Tất cả các câu trong danh sách sentences được token hóa thành các từ riêng lẻ và chuyển về dạng chữ thường. Các token này được lưu trữ trong danh sách all\_tokens.
* Đếm tần suất xuất hiện của các từ: Sử dụng lớp Counter từ thư viện collections để đếm số lần xuất hiện của mỗi từ trong danh sách all\_tokens. Kết quả được lưu trữ trong biến counter.
* Xác định các từ hiếm: Dựa vào kiểu dữ liệu của tham số threshold, hàm sẽ xác định tập hợp các từ hiếm (rare\_words):
* Nếu threshold là số nguyên, tập hợp rare\_words sẽ chứa tất cả các từ có số lần xuất hiện nhỏ hơn threshold trong counter.
* Nếu threshold là số thực, tổng số từ trong lô dữ liệu được tính toán. Sau đó, tập hợp rare\_words sẽ chứa tất cả các từ có tần suất xuất hiện (số lần xuất hiện chia cho tổng số từ) nhỏ hơn threshold.
* Nếu threshold không thuộc một trong hai kiểu dữ liệu trên, tập hợp rare\_words sẽ là một tập hợp rỗng.
* Lọc từ hiếm khỏi từng câu: Một hàm nội bộ filter\_sentence được định nghĩa để xử lý từng câu riêng lẻ. Hàm này thực hiện các bước sau:
* Token hóa câu đầu vào.
* Duyệt qua các token và chỉ giữ lại những token (sau khi chuyển về chữ thường) không nằm trong tập hợp rare\_words.
* Kết hợp các token còn lại thành một chuỗi, cách nhau bằng dấu cách.
* Áp dụng bộ lọc cho toàn bộ lô câu: Hàm chính sử dụng một biểu thức comprehension danh sách để áp dụng hàm filter\_sentence cho từng câu trong danh sách sentences ban đầu.
* Trả về danh sách các câu đã lọc: Hàm trả về danh sách mới chứa các câu đã được lọc, trong đó các từ có tần suất xuất hiện thấp đã bị loại bỏ.

Việc lọc các từ tần suất thấp là một bước tiền xử lý hữu ích để giảm kích thước bộ từ vựng và tập trung vào các từ có ý nghĩa thống kê hơn trong bộ dữ liệu. Tuy nhiên, cần cân nhắc kỹ lưỡng ngưỡng tần suất để tránh loại bỏ các từ khóa quan trọng, đặc biệt trong các bộ dữ liệu nhỏ.

### Lọc câu đưa ra theo phân loại :

Trong một số trường hợp, việc lọc các câu dựa trên nội dung chủ đề của chúng có thể hữu ích để đảm bảo rằng bộ dữ liệu huấn luyện tập trung vào các thông tin liên quan. Phương pháp lọc này có thể được thực hiện bằng cách sử dụng mô hình phân loại không nhãn (zero-shot classification). Zero-shot classification cho phép phân loại văn bản vào các danh mục (labels) mà mô hình chưa từng được huấn luyện trực tiếp trên đó, dựa vào khả năng hiểu ngữ nghĩa và mối quan hệ giữa các khái niệm của các mô hình ngôn ngữ lớn đã được huấn luyện trước đó. Hàm filter\_with\_zero\_shot được thiết kế để thực hiện việc lọc câu dựa trên phương pháp này.

Đầu vào (Input):

Hàm filter\_with\_zero\_shot nhận các tham số sau:

* sentence (kiểu dữ liệu: str): Chuỗi văn bản đầu vào, là câu cần được đánh giá và lọc.
* classifier: Một đối tượng pipeline từ thư viện Hugging Face Transformers, đã được cấu hình cho tác vụ zero-shot classification. Đối tượng này đã được huấn luyện trước đó trên một tập dữ liệu lớn và có khả năng suy luận về mối quan hệ giữa văn bản và các nhãn (labels).
* candidate\_labels (kiểu dữ liệu: list[str]): Một danh sách các chuỗi, trong đó mỗi chuỗi đại diện cho một nhãn (chủ đề) tiềm năng mà câu có thể thuộc về.
* score\_threshold (kiểu dữ liệu: float): Một giá trị ngưỡng (trong khoảng từ 0 đến 1) quy định mức độ tin cậy tối thiểu mà mô hình cần gán cho ít nhất một trong các nhãn candidate\_labels để câu được coi là phù hợp và vượt qua bộ lọc.

Đầu ra (Output):

Hàm trả về một giá trị boolean (bool):

* True: Nếu câu được phân loại là có liên quan đến ít nhất một trong các nhãn candidate\_labels với điểm số tin cậy lớn hơn hoặc bằng score\_threshold.
* False: Nếu không có nhãn nào trong candidate\_labels được gán cho câu với điểm số tin cậy đạt đến ngưỡng score\_threshold.

Quá trình thực hiện:

Hàm filter\_with\_zero\_shot thực hiện việc lọc câu theo các bước sau:

* Thực hiện phân loại không nhãn: Câu đầu vào sentence được đưa vào mô hình classifier cùng với danh sách các nhãn tiềm năng candidate\_labels và tham số multi\_label được đặt là False (chỉ chọn nhãn có điểm số cao nhất). Mô hình sẽ dự đoán mức độ phù hợp của câu với từng nhãn trong danh sách.
* Lấy điểm số tin cậy cao nhất: Kết quả phân loại trả về một danh sách các điểm số tương ứng với mức độ tin cậy của mô hình đối với từng nhãn. Hàm này trích xuất điểm số cao nhất từ danh sách này.
* So sánh với ngưỡng: Điểm số tin cậy cao nhất được so sánh với giá trị ngưỡng score\_threshold.
* Trả về kết quả lọc: Nếu điểm số tin cậy cao nhất lớn hơn hoặc bằng score\_threshold, hàm trả về True, cho biết câu được coi là có liên quan đến một trong các chủ đề quan tâm và vượt qua bộ lọc. Ngược lại, nếu không có nhãn nào đạt được điểm số tin cậy đủ cao, hàm trả về False, và câu sẽ bị loại bỏ.

Phương pháp lọc này cho phép chúng ta tự động phân loại và giữ lại các câu có nội dung phù hợp với một tập các chủ đề được định nghĩa trước, mà không cần phải huấn luyện mô hình phân loại cụ thể cho các chủ đề này. Điều này đặc biệt hữu ích trong các trường hợp mà chúng ta muốn tập trung vào một khía cạnh cụ thể của dữ liệu văn bản.

### Lọc và sắp xếp dữ liệu :

Sau khi hoàn thành giai đoạn tiền xử lý dữ liệu, bước tiếp theo là lọc và sắp xếp các câu dựa trên mức độ liên quan của chúng đến chủ đề của khóa luận. Để thực hiện tác vụ này, chúng tôi sử dụng mô hình phân loại không nhãn (zero-shot classification) đã được huấn luyện trước đó, cụ thể là mô hình "MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli" từ thư viện Hugging Face Transformers. Mô hình này được lựa chọn nhờ khả năng hiểu ngữ nghĩa sâu sắc và khả năng đánh giá mức độ phù hợp giữa một đoạn văn bản và một tập các nhãn (labels) cho trước mà không cần được huấn luyện cụ thể trên các nhãn đó.

Hàm filter\_useful\_content được thiết kế để thực hiện việc lọc và sắp xếp này. Hàm nhận một danh sách các câu đã được tiền xử lý, áp dụng mô hình phân loại không nhãn để tính toán điểm số liên quan của mỗi câu đối với một tập các nhãn chủ đề được định nghĩa trước, và sau đó sắp xếp các câu theo điểm số này theo thứ tự giảm dần.

Đầu vào (Input):

Hàm filter\_useful\_content nhận các tham số sau:

* sentences (kiểu dữ liệu: List[str]): Một danh sách các chuỗi, trong đó mỗi chuỗi là một câu đã được tiền xử lý.
* pretrained\_model (kiểu dữ liệu: pipeline, giá trị mặc định: PIPELINE\_CLASSIFIER): Một đối tượng pipeline từ thư viện Hugging Face Transformers đã được khởi tạo với mô hình zero-shot classification. Mô hình mặc định được thiết lập để nhận diện nội dung kỹ thuật.
* candidate\_labels (kiểu dữ liệu: List[str], giá trị mặc định: ["technical", "programming", "IT"]): Một danh sách các chuỗi đại diện cho các nhãn chủ đề mà chúng tôi quan tâm, ví dụ như "technical" (kỹ thuật), "programming" (lập trình), "IT" (công nghệ thông tin). Mô hình sẽ đánh giá mức độ liên quan của mỗi câu đối với các nhãn này.
* batch\_size (kiểu dữ liệu: int, giá trị mặc định: 32): Kích thước lô (batch size) được sử dụng để xử lý các câu. Việc xử lý theo lô giúp tối ưu hóa hiệu suất, đặc biệt khi làm việc với số lượng lớn câu.

Đầu ra (Output):

Hàm trả về một danh sách các tuples (List[Tuple[float, str]]). Mỗi tuple chứa hai phần tử:

* Phần tử đầu tiên (kiểu dữ liệu: float) là điểm số liên quan cao nhất mà mô hình gán cho câu đó đối với bất kỳ nhãn nào trong candidate\_labels.
* Phần tử thứ hai (kiểu dữ liệu: str) là câu tương ứng.

Danh sách kết quả được sắp xếp theo thứ tự giảm dần của điểm số liên quan, nghĩa là các câu được coi là có mức độ liên quan cao nhất sẽ xuất hiện đầu tiên.

Quá trình thực hiện:

Hàm filter\_useful\_content thực hiện việc lọc và sắp xếp dữ liệu theo các bước sau:

* Kiểm tra đầu vào: Hàm kiểm tra xem danh sách câu đầu vào và danh sách nhãn có rỗng hay không. Nếu một trong hai danh sách rỗng, hàm sẽ trả về lỗi ValueError.
* Xử lý theo lô: Các câu đầu vào được chia thành các lô có kích thước batch\_size để xử lý hiệu quả hơn.
* Phân loại không nhãn: Với mỗi lô câu, mô hình pretrained\_model được áp dụng để thực hiện phân loại không nhãn dựa trên danh sách candidate\_labels. Tham số multi\_label được đặt là False, nghĩa là mô hình sẽ chọn nhãn có điểm số cao nhất cho mỗi câu.
* Lấy điểm số cao nhất: Đối với mỗi câu trong lô, hàm trích xuất điểm số tin cậy cao nhất từ kết quả phân loại.
* Lưu trữ kết quả: Điểm số cao nhất và câu tương ứng được lưu trữ dưới dạng một tuple trong danh sách filtered.
* Xử lý lỗi: Một khối try-except được sử dụng để bắt và xử lý các lỗi có thể xảy ra trong quá trình xử lý theo lô, đảm bảo tính ổn định của chương trình.
* Sắp xếp kết quả: Sau khi xử lý tất cả các lô, danh sách filtered chứa các tuple (điểm số, câu) được sắp xếp theo thứ tự giảm dần của điểm số, sử dụng phương thức sort với tham số reverse=True.
* Trả về danh sách đã lọc và sắp xếp: Hàm trả về danh sách các tuple đã được sắp xếp, cho phép chúng ta dễ dàng truy cập các câu có mức độ liên quan cao nhất đến chủ đề quan tâm.

Bằng cách sử dụng mô hình phân loại không nhãn và hàm filter\_useful\_content, chúng ta có thể tự động đánh giá và ưu tiên các phần nội dung quan trọng nhất từ văn bản đã trích xuất, giúp tập trung vào thông tin có giá trị cao cho các bước xử lý tiếp theo, chẳng hạn như tạo cặp câu hỏi-trả lời.

### Kiểm tra và sửa lỗi chính tả :

Sau khi các câu đã được lọc và sắp xếp theo mức độ liên quan, bước tiếp theo là tiến hành kiểm tra và sửa lỗi chính tả. Mục đích của giai đoạn này là nâng cao chất lượng của dữ liệu văn bản bằng cách khắc phục các lỗi chính tả có thể xuất hiện trong quá trình trích xuất hoặc tiền xử lý. Để thực hiện tác vụ này, chúng tôi sử dụng mô hình ngôn ngữ lớn đã được huấn luyện trước đó "google/flan-t5-small" từ thư viện Hugging Face Transformers. Mô hình này có khả năng sinh văn bản và được kỳ vọng sẽ thực hiện tốt việc sửa lỗi chính tả.

Một yếu tố quan trọng cần xem xét trong quá trình sửa lỗi chính tả là việc bảo toàn các thuật ngữ chuyên ngành, đặc biệt là các từ khóa liên quan đến lĩnh vực công nghệ thông tin và khoa học dữ liệu, được lưu trữ trong danh sách DOMAIN\_KEYWORDS. Việc mô hình sửa đổi các từ khóa này có thể dẫn đến sai lệch về ý nghĩa và gây ảnh hưởng tiêu cực đến quá trình huấn luyện mô hình hỏi đáp. Do đó, quy trình sửa lỗi chính tả được thiết kế để chỉ can thiệp vào các từ không nằm trong danh sách DOMAIN\_KEYWORDS.

Quá trình kiểm tra và sửa lỗi chính tả được thực hiện thông qua hai hàm chính: correct\_sentence và spell\_correction.

Hàm correct\_sentence:

Hàm này chịu trách nhiệm sửa lỗi chính tả cho một câu đơn lẻ, đồng thời bảo vệ các từ khóa chuyên ngành. Quá trình thực hiện như sau:

* Che giấu từ khóa chuyên ngành: Trước khi đưa câu vào mô hình sửa lỗi chính tả, các từ khóa có trong DOMAIN\_KEYWORDS sẽ được thay thế bằng các mã định danh tạm thời (placeholders). Việc này được thực hiện bằng cách sử dụng biểu thức chính quy để tìm kiếm và thay thế các từ khóa (không phân biệt chữ hoa chữ thường và đảm bảo khớp toàn bộ từ) bằng các token đặc biệt như "ITKEYWORD0", "ITKEYWORD1", v.v. Một bản đồ ánh xạ giữa các token này và từ khóa gốc được lưu lại.
* Thực hiện sửa lỗi chính tả: Câu đã được che giấu sẽ được đưa vào pipeline sửa lỗi chính tả (được khởi tạo với mô hình "google/flan-t5-small"). Pipeline này sẽ trả về câu đã được sửa lỗi.
* Khôi phục từ khóa chuyên ngành: Sau khi nhận được câu đã sửa lỗi, các mã định danh tạm thời sẽ được thay thế trở lại bằng các từ khóa chuyên ngành ban đầu, sử dụng bản đồ ánh xạ đã tạo ở bước Xác định từ khóa.

Hàm spell\_correction:

Hàm này được sử dụng để áp dụng quy trình sửa lỗi chính tả cho một danh sách các câu đã được lọc (đầu ra từ bước lọc và sắp xếp dữ liệu). Hàm này hỗ trợ xử lý song song để tăng tốc độ xử lý và cho phép giới hạn số lượng câu cần được sửa lỗi. Quá trình thực hiện như sau:

* Xác định số lượng câu cần xử lý: Dựa trên tham số limit, hàm xác định số lượng câu cần được sửa lỗi. Tham số này cho phép xử lý một phần nhất định của danh sách (theo tỷ lệ phần trăm hoặc số lượng cụ thể) hoặc toàn bộ danh sách.
* Chia danh sách câu: Danh sách các câu đã lọc được chia thành hai phần: phần cần được xử lý và phần còn lại (nếu có giới hạn được áp dụng).
* Xử lý song song: Sử dụng ThreadPoolExecutor để thực hiện việc sửa lỗi chính tả cho các câu trong phần cần xử lý một cách song song. Mỗi câu được truyền vào hàm correct\_sentence cùng với pipeline sửa lỗi chính tả và danh sách từ khóa chuyên ngành. Việc sử dụng thread pool giúp tránh việc khởi tạo lại mô hình nhiều lần, đặc biệt khi sử dụng các mô hình trên GPU.
* Kết hợp kết quả: Kết quả từ quá trình xử lý song song (danh sách các tuple chứa điểm số và câu đã sửa lỗi) được kết hợp với phần còn lại của danh sách câu ban đầu.

Thông qua việc sử dụng mô hình "google/flan-t5-small" kết hợp với cơ chế bảo vệ từ khóa chuyên ngành, chúng tôi đảm bảo rằng quá trình kiểm tra và sửa lỗi chính tả được thực hiện một cách hiệu quả, cải thiện độ chính xác của dữ liệu văn bản đồng thời giữ nguyên vẹn các thuật ngữ kỹ thuật quan trọng.

### Tổng hợp và lưu trữ dữ liệu :

Sau khi trải qua các bước tiền xử lý, lọc, sắp xếp và kiểm tra lỗi chính tả, giai đoạn cuối cùng trong quy trình chuẩn bị dữ liệu là tổng hợp các thông tin đã xử lý và lưu trữ chúng vào một định dạng cấu trúc, sẵn sàng cho việc huấn luyện mô hình chatbot hỏi đáp. Hàm generate\_questions\_and\_answers\_and\_save\_csv được thiết kế để thực hiện tác vụ này, bao gồm việc tự động tạo ra các cặp câu hỏi-trả lời từ các câu đã được làm sạch và lưu trữ chúng cùng với điểm số liên quan ban đầu vào một tệp định dạng CSV.

Hàm này sử dụng hai pipeline từ thư viện Hugging Face Transformers: một pipeline cho việc tạo câu hỏi từ một đoạn văn bản (ví dụ, sử dụng mô hình "allenai/t5-small-squad2-question-generation") và một pipeline cho việc trả lời câu hỏi dựa trên một ngữ cảnh cho trước (ví dụ, sử dụng mô hình "deepset/roberta-base-squad2"). Việc sử dụng các mô hình đã được huấn luyện trước này giúp tự động hóa quá trình tạo ra các cặp dữ liệu huấn luyện chất lượng cao.

Đầu vào (Input):

Hàm generate\_questions\_and\_answers\_and\_save\_csv nhận các tham số sau:

* records (kiểu dữ liệu: List[Tuple[float, str]]): Một danh sách các tuples, trong đó mỗi tuple chứa điểm số liên quan và câu đã được sửa lỗi chính tả.
* question\_generation\_pipeline (kiểu dữ liệu: pipeline, giá trị mặc định: PIPELINE\_QUESTION\_GENERATION - mô hình allenai/t5-small-squad2-question-generation): Một đối tượng pipeline từ thư viện Hugging Face Transformers đã được khởi tạo cho tác vụ tạo câu hỏi.
* question\_answering\_pipeline (kiểu dữ liệu: pipeline, giá trị mặc định: PIPELINE\_QUESTION\_ANSWERING - mô hình deepset/roberta-base-squad2): Một đối tượng pipeline từ thư viện Hugging Face Transformers đã được khởi tạo cho tác vụ trả lời câu hỏi.
* dataset\_path (kiểu dữ liệu: str, giá trị mặc định: DATASET\_PATH): Đường dẫn đến tệp CSV nơi bộ dữ liệu sẽ được lưu trữ.
* min\_question\_word\_count (kiểu dữ liệu: int, giá trị mặc định: 4): Số lượng từ tối thiểu mà một câu hỏi được tạo ra phải có để được coi là hợp lệ. Các câu hỏi quá ngắn có thể không đủ ngữ nghĩa cho việc huấn luyện.

Đầu ra (Output):

Hàm trả về một số nguyên (int) đại diện cho số lượng bản ghi (cặp câu hỏi-trả lời) đã được ghi thành công vào tệp CSV.

Quá trình thực hiện:

Hàm generate\_questions\_and\_answers\_and\_save\_csv thực hiện việc tổng hợp và lưu trữ dữ liệu theo các bước sau:

* Định nghĩa hàm xử lý bản ghi process\_record: Một hàm nội bộ process\_record được định nghĩa để xử lý từng bản ghi (score, sentence). Hàm này thực hiện các bước sau:
  + Tạo câu hỏi: Sử dụng question\_generation\_pipeline để tạo ra một hoặc nhiều câu hỏi có thể được trả lời bởi câu đầu vào. Một câu hỏi được chọn ngẫu nhiên từ các ứng viên để tăng tính đa dạng của bộ dữ liệu.
  + Kiểm tra độ dài câu hỏi: Câu hỏi được tạo ra sẽ được kiểm tra độ dài (số lượng từ). Nếu số lượng từ nhỏ hơn min\_question\_word\_count, câu hỏi đó sẽ bị bỏ qua.
  + Tạo câu trả lời: Sử dụng question\_answering\_pipeline để tìm câu trả lời cho câu hỏi đã tạo trong ngữ cảnh của câu gốc.
  + Trả về dữ liệu đã xử lý: Nếu cả câu hỏi và câu trả lời được tạo thành công, hàm sẽ trả về một tuple chứa điểm số, câu gốc, câu hỏi đã tạo và câu trả lời tương ứng. Nếu không, hàm trả về None.
* Xử lý song song các bản ghi: Sử dụng ThreadPoolExecutor để xử lý đồng thời nhiều bản ghi trong danh sách records. Điều này giúp tận dụng tối đa sức mạnh của bộ xử lý, đặc biệt là khi sử dụng các pipeline trên GPU, và giảm tổng thời gian xử lý.
* Lọc các kết quả không thành công: Các kết quả trả về là None (do lỗi hoặc không tạo được câu hỏi hợp lệ) sẽ bị loại bỏ.
* Gán ID cho các bản ghi hợp lệ: Một ID duy nhất (bắt đầu từ 0) được gán cho mỗi bản ghi hợp lệ để dễ dàng theo dõi và quản lý trong bộ dữ liệu.
* Ghi dữ liệu vào tệp CSV: Cuối cùng, dữ liệu đã được xử lý (bao gồm ID, điểm số, câu gốc, câu hỏi và câu trả lời) được ghi vào một tệp CSV tại đường dẫn dataset\_path. Tệp CSV sẽ có các cột tương ứng là "sentence\_id", "score", "sentence", "question" và "answer".
* Trả về số lượng bản ghi đã ghi: Hàm trả về số lượng bản ghi đã được ghi thành công vào tệp CSV.

Giai đoạn tổng hợp và lưu trữ dữ liệu này đánh dấu sự hoàn thành của quy trình chuẩn bị dữ liệu. Kết quả là một bộ dữ liệu có cấu trúc chứa các cặp câu hỏi-trả lời chất lượng cao, sẵn sàng được sử dụng để huấn luyện mô hình chatbot hỏi đáp cho sinh viên. Việc xử lý song song giúp tăng tốc quá trình tạo bộ dữ liệu, đặc biệt khi làm việc với lượng lớn văn bản.

### Kết quả :

Để chuẩn bị bộ dữ liệu cho quá trình huấn luyện mô hình, nhóm đã tiến hành một quy trình tiền xử lý và tổng hợp dữ liệu chi tiết, được thực hiện trong môi trường Jupyter Notebook trên nền tảng đám mây Kaggle (phiên bản "Save version 8"). Quy trình này được thiết kế dưới dạng một pipeline, trong đó đầu ra của mỗi bước xử lý đóng vai trò là đầu vào cho bước tiếp theo, đảm bảo một luồng dữ liệu liên tục và có hệ thống.

Sau khi hoàn thành các bước tiền xử lý ban đầu, danh sách các câu đã được sắp xếp theo thứ tự giảm dần dựa trên điểm số liên quan bằng cách sử dụng hàm sorted() của Python, cho phép tập trung vào các nội dung có mức độ phù hợp cao nhất. Tiếp theo, nhóm đã áp dụng quy trình kiểm tra và sửa lỗi chính tả cho 800 bản ghi có điểm số liên quan cao nhất trong danh sách này.

Cuối cùng, phương thức generate\_questions\_and\_answers\_and\_save\_csv() đã được sử dụng để tạo ra các cặp câu hỏi-trả lời từ các câu đã được xử lý và lưu trữ chúng vào tệp CSV. Trong lần chạy này, 800 bản ghi có mức độ liên quan cao nhất đã được đưa vào phương thức này. Tuy nhiên, trong quá trình tạo câu hỏi bằng mô hình allenai/t5-small-squad2-question-generation, một số trường hợp mô hình đã tạo ra các chuỗi rỗng hoặc các câu hỏi quá ngắn, không đạt yêu cầu về chất lượng. Do đó, các bản ghi này đã bị loại bỏ trong quá trình lưu trữ.

Kết quả cuối cùng của quá trình này là 775 bản ghi đã được lưu thành công vào tệp CSV, trong khi 25 bản ghi không đạt yêu cầu đã bị loại bỏ. Chi tiết về kết quả của lần chạy notebook này có thể được xem [tại đây](https://www.kaggle.com/code/khuonngvongnhat/prepare-dataset).

## Tinh chỉnh mô hình :

### Mục tiêu và cơ sở lý thuyết của tinh chỉnh mô hình

Việc tinh chỉnh mô hình ngôn ngữ lớn (LLM) là một giai đoạn thiết yếu nhằm nâng cao hiệu suất và khả năng thích ứng của mô hình với các tác vụ chuyên biệt. Mục tiêu chính của quá trình này là chuyển giao và chuyên biệt hóa kiến thức của mô hình nền tảng (pre-trained model) từ một miền dữ liệu tổng quát sang một miền cụ thể hơn. Đối với nghiên cứu này, việc tinh chỉnh ViSTRAL-7B-Chat hướng đến mục tiêu giúp chatbot của chúng tôi hiểu sâu sắc ngữ cảnh, thuật ngữ chuyên ngành và cung cấp phản hồi chính xác từ các tài liệu giảng dạy của Khoa Công nghệ thông tin. Điều này khắc phục hạn chế của các LLM tổng quát trong việc xử lý kiến thức chuyên biệt và giảm thiểu hiện tượng "ảo giác" (hallucination).

Trong bối cảnh phát triển LLM, có ba phương pháp chính để thích ứng mô hình với tác vụ mới bao gồm Few-shot Prompting, Fine-tuning và Retrieval-Augmented Generation (RAG).

Few-shot Prompting yêu cầu các ví dụ trong câu lệnh mà không sửa đổi tham số mô hình, phù hợp với các tác vụ đơn giản nhưng bị giới hạn bởi độ dài ngữ cảnh và độ chính xác.

RAG giúp mô hình truy xuất thông tin từ cơ sở tri thức bên ngoài để tạo sinh phản hồi, giảm thiểu "ảo giác" và cung cấp thông tin cập nhật mà không cần tinh chỉnh mô hình, tuy nhiên có thể gặp thách thức trong việc tổng hợp thông tin phức tạp.

Fine-tuning là quá trình huấn luyện tiếp tục một mô hình đã được huấn luyện trước trên một tập dữ liệu nhỏ và cụ thể. Phương pháp này cho phép mô hình học các mẫu và sắc thái của miền dữ liệu mục tiêu, dẫn đến hiệu suất cao hơn trong các tác vụ chuyên biệt. Về mặt kinh tế và tài nguyên, tinh chỉnh mô hình thường tiết kiệm thời gian và chi phí tính toán đáng kể so với việc tự xây dựng và huấn luyện một mô hình ngôn ngữ lớn hoàn toàn từ đầu, đồng thời tận dụng được các kiến thức rộng lớn mà mô hình nền tảng đã học được.

### Phương pháp Tinh chỉnh LoRA (Low-Rank Adaption)

Để tinh chỉnh mô hình Vistral-7B-Chat một các hiệu quả và tiết kiệm tài nguyên, chúng tôi đã áp dụng phương pháp Low-Rank Adaption (LoRA), một kỹ thuật tiên tiến thuộc nhóm Parameter-Efficient Fine-Tuning (PEFT)

LoRA hoạt động dựa trên nguyên lý phân rã hạng thấp (low-rank factorization) của ma trận trọng số. Cụ thể, thay vì huấn luyện lại toàn bộ ma trận trọng số ​ của một lớp mạng nơ-ron (vốn có kích thước rất lớn, ví dụ d×k), LoRA đóng băng ​​ và thêm vào một cặp ma trận hạng thấp A (d×r) và B (r×k), với r≪min(d,k). Sự thay đổi trọng số ΔW được biểu diễn dưới dạng tích của hai ma trận này: ΔW=BA. Trong quá trình tinh chỉnh, chỉ các tham số trong ma trận A và B này được huấn luyện, trong khi ​​ của mô hình gốc được giữ nguyên. Điều này giúp giảm đáng kể lượng tham số cần huấn luyện (từ hàng tỷ xuống còn vài triệu), nhưng vẫn đảm bảo khả năng học và thích ứng của mô hình với dữ liệu mới, từ đó duy trì được chất lượng của mô hình đã tinh chỉnh. Khi cần suy luận, các ma trận A và B có thể được hợp nhất vào ​​, giúp quá trình suy luận không phát sinh thêm chi phí đáng kể so với mô hình gốc.

Việc lựa chọn LoRA mang lại nhiều ưu điểm vượt trội cho nghiên cứu này:

* Hiệu quả bộ nhớ:
  + Chỉ các tham số của các adapter (ma trận A và B) được lưu trữ và cập nhật trong bộ nhớ GPU, trong khi trọng số của mô hình cơ sở vẫn được giữ nguyên và có thể được tải ở độ chính xác thấp hơn (ví dụ, 4-bit với QLoRA), cho phép tinh chỉnh các mô hình ngôn ngữ lớn trên các GPU thông thường như GPU tiêu dùng.
* Hiệu quả đào tạo:
  + LoRA cung cấp một cơ chế tinh chỉnh hiệu quả, dễ dàng tích hợp với các thư viện hiện có, đòi hỏi thiết lập tối thiểu.
  + Nó hỗ trợ các biến thể nâng cao như QLoRA (Quantized LoRA), giúp cải thiện hiệu quả bộ nhớ hơn nữa thông qua lượng tử hóa, mở rộng khả năng tinh chỉnh các mô hình cực lớn.
* Quản lý bộ điều hợp:
  + Các trọng số của bộ điều hợp có thể được lưu trữ độc lập với mô hình cơ sở, tạo điều kiện thuận lợi cho việc kiểm tra và chuyển đổi giữa các tác vụ khác nhau.
  + Hỗ trợ tính năng hợp nhất các bộ điều hợp trở lại mô hình cơ sở khi triển khai, đơn giản hóa quá trình sử dụng.

### Bộ dữ liệu huấn luyện

Để tinh chỉnh mô hình ViSTRAL-7B-Chat đạt hiệu quả cao trong việc hỗ trợ các môn học Công nghệ thông tin, việc xây dựng một bộ dữ liệu huấn luyện chuyên biệt và đa dạng là vô cùng quan trọng. Chúng tôi đã sử dụng kết hợp hai nguồn dữ liệu chính để tạo nên bộ dữ liệu cuối cùng.

Đầu tiên, một bộ dữ liệu khởi tạo (dataset.csv) đã được chuẩn bị thông qua quá trình trích xuất và tiền xử lý từ các tài liệu bài giảng và sách tham, chính là kết quả của bước [Chuẩn bị bộ dữ liệu huấn luyện mô hình](#_Chuẩn_bị_bộ). Bộ dữ liệu này bao gồm 636 mẫu, với các trường sentence\_id, score, sentence, question, và answer. Tuy nhiên, với số lượng mẫu hạn chế và yêu cầu về tính bao quát, chất lượng của bộ dữ liệu này đối với việc huấn luyện một mô hình ngôn ngữ lớn còn là một thách thức.

Do đó, để bổ sung và nâng cao chất lượng dữ liệu huấn luyện, chúng tôi đã tích hợp thêm bộ dữ liệu aiswariya24/DSA\_interview\_datasets từ nền tảng Hugging Face. Bộ dữ liệu công khai này cung cấp thêm 2973 mẫu từ tập huấn luyện và 1275 mẫu từ tập kiểm tra, tổng cộng 4248 mẫu, tập trung vào các câu hỏi và trả lời liên quan đến Cấu trúc dữ liệu và Giải thuật (DSA).

Hai bộ dữ liệu này sau đó đã được gộp lại và tiền xử lý đồng nhất để tạo thành một bộ dữ liệu huấn luyện duy nhất. Việc kết hợp này không chỉ tăng cường đáng kể số lượng mẫu dữ liệu (tổng cộng 4884 mẫu) mà còn cải thiện tính đa dạng và độ bao phủ của kiến thức, đảm bảo mô hình có thể học được nhiều ngữ cảnh và mẫu câu hỏi-trả lời hơn. Bộ dữ liệu tổng hợp này là nền tảng vững chắc cho quá trình tinh chỉnh LoRA, giúp ViSTRAL-7B-Chat thích nghi hiệu quả với miền tri thức chuyên ngành và nâng cao năng lực phản hồi của chatbot.

### Quy trình huấn luyện

Quy trình huấn luyện mô hình ViSTRAL-7B-Chat được triển khai nhằm tối ưu hóa khả năng của mô hình trên bộ dữ liệu chuyên ngành thông qua phương pháp LoRA. Quy trình huấn luyện mô hình được triển khai trên môi trường jupyter notebook của Kaggle kết hợp với các thư viện tiên tiến như transformer, peft, trl của Hugging Face.

Đầu tiên, mô hình ViSTRAL-7B-Chat và bộ mã hóa (tokenizer) tương ứng được tải từ Hugging Face Hub ở định dạng float16 để tối ưu hóa bộ nhớ. Một cấu hình LoRA (LoraConfig) được khởi tạo với các tham số cụ thể: r=16, lora\_alpha=32, lora\_dropout=0.05, bias="none", và target\_modules bao gồm các lớp q\_proj, v\_proj trong kiến trúc attention của mô hình để áp dụng LoRA. Sau đó, mô hình cơ sở được chuyển đổi thành mô hình PEFT bằng cách tích hợp các adapter LoRA này.

Các tham số huấn luyện được định nghĩa thông qua lớp TrainingArguments, bao gồm num\_train\_epochs=3 để đảm bảo mô hình có đủ thời gian học từ dữ liệu, per\_device\_train\_batch\_size=4 và gradient\_accumulation\_steps=2 để mô phỏng kích thước batch lớn hơn trong điều kiện bộ nhớ giới hạn. Tốc độ học learning\_rate được thiết lập là 2e-4, và trình tối ưu hóa AdamW cùng với cosine\_lr\_scheduler được sử dụng để điều chỉnh tốc độ học trong suốt quá trình. Để theo dõi tiến độ, logging\_steps và save\_steps được đặt ở 100 và 200 tương ứng.

Cuối cùng, một đối tượng SFTTrainer được khởi tạo, tích hợp mô hình đã cấu hình LoRA, bộ mã hóa, bộ dữ liệu huấn luyện đã chuẩn bị, và các tham số huấn luyện. SFTTrainer được chọn vì khả năng tối ưu hóa việc tinh chỉnh có giám sát các mô hình theo dạng hướng dẫn, tự động xử lý các tác vụ như định dạng dữ liệu và mặt nạ chú ý. Quá trình huấn luyện được khởi chạy bằng phương thức trainer.train(), cho phép mô hình điều chỉnh các trọng số của adapter LoRA và tối ưu hóa hiệu suất trên tác vụ hỏi đáp chuyên ngành. Mô hình tinh chỉnh cuối cùng sẽ được lưu trữ để phục vụ cho giai đoạn xây dựng và đánh giá ứng dụng chatbot.

## Tích hợp RAG vào mô hình :

Trong thời đại bùng nổ thông tin, các mô hình ngôn ngữ lớn (LLMs) như GPT, BERT hay Mistral tuy mạnh mẽ nhưng vẫn gặp hạn chế về khả năng truy xuất kiến thức thực tế hoặc dữ liệu cập nhật sau thời điểm huấn luyện. Để giải quyết vấn đề này, phương pháp **RAG (Retrieval-Augmented Generation)** được đề xuất như một cách tiếp cận hiệu quả, kết hợp giữa khả năng sinh ngôn ngữ và khả năng truy xuất thông tin từ cơ sở tri thức bên ngoài.

Trong phần thực nghiệm của đề tài, nhóm đã tiến hành xây dựng một hệ thống RAG đơn giản bao gồm các thành phần sau:

### Trích xuất dữ liệu từ file PDF :

Hàm ***extract\_text\_from\_pdf(filepath, start\_page=1, end\_page=None)*** được sử dụng để trích xuất nội dung văn bản từ một tài liệu PDF trong phạm vi các trang chỉ định. Hàm sử dụng thư viện PyMuPDF để mở tệp PDF và đọc nội dung từ trang ***start\_page*** đến ***end\_page***. Kết quả đầu ra là một chuỗi chứa toàn bộ văn bản được ghép nối từ các trang đã chọn, phục vụ cho các bước tiền xử lý và phân tích tiếp theo trong hệ thống.



Hình : code cho phần trích xuất dữ liệu PDF

### Xử lý dữ liệu đầu vào :

Hàm ***chunk\_text\_with\_context(text, chunk\_size=5, overlap=2)*** được xây dựng nhằm mục tiêu phân chia văn bản thành các đoạn nhỏ có kích thước cố định, đồng thời gắn kèm thông tin ngữ cảnh để đảm bảo tính liên kết và mạch lạc trong quá trình xử lý dữ liệu văn bản.

Quá trình thực thi của hàm được chia thành các bước như sau:

* **Bước 1: Tiền xử lý văn bản**  
  Văn bản đầu vào được chia thành từng dòng riêng lẻ dựa trên ký tự xuống dòng (\n). Các dòng rỗng sẽ bị loại bỏ để đảm bảo tính chính xác trong phân tích.
* **Bước 2: Xác định tiêu đề (heading)**  
  Tiêu đề được xác định dựa trên một số biểu hiện hình thức như:
  + Dòng bắt đầu bằng emoji.
  + Dòng kết thúc bằng dấu : hoặc ?.
  + Dòng được viết in hoa hoàn toàn.

Việc xác định tiêu đề được thực hiện thông qua hàm con is\_heading(), đóng vai trò làm điểm đánh dấu cho ngữ cảnh nội dung theo sau.

* **Bước 3: Ghi nhận ngữ cảnh hiện tại**  
  Khi gặp một dòng được xác định là tiêu đề, ngữ cảnh hiện hành (context\_stack) sẽ được cập nhật. Ngữ cảnh này sẽ được gắn kèm với các dòng nội dung sau đó cho đến khi gặp tiêu đề mới.
* **Bước 4: Ghi nhận nội dung có ngữ cảnh**  
  Các dòng không phải là tiêu đề sẽ được lưu vào bộ đệm buffer dưới dạng cặp giá trị gồm nội dung và ngữ cảnh tương ứng tại thời điểm đó.
* **Bước 5: Phân đoạn theo cửa sổ trượt**  
  Văn bản được chia thành từng đoạn (chunk) theo kích thước chunk\_size, với số dòng chồng lấn giữa các đoạn là overlap. Việc chồng lấn giúp đảm bảo không mất thông tin ở ranh giới giữa các đoạn. Mỗi đoạn văn bản được gắn ngữ cảnh dựa trên dòng đầu tiên trong đoạn.
* **Kết quả trả về**  
  Hàm trả về một danh sách các đoạn văn bản, mỗi đoạn bao gồm hai trường thông tin:
  + content: nội dung đoạn văn bản.
  + context: tiêu đề ngữ cảnh áp dụng cho đoạn đó.

Hàm này đóng vai trò quan trọng trong hệ thống tiền xử lý văn bản, giúp duy trì tính liên tục của mạch văn và cải thiện hiệu quả trong các tác vụ NLP như phân loại, tìm kiếm ngữ nghĩa và truy xuất thông tin theo ngữ cảnh.



Hình : code cho phần phân đoạn tài liệu

### Sinh vector embedding :

Hàm ***embed\_chunks(chunks, model)*** được sử dụng để chuyển đổi các đoạn văn bản đã được phân chia từ bước trước thành các vector số (vector biểu diễn ngữ nghĩa) thông qua mô hình embedding.

#### Mục tiêu :

Biến đổi từng đoạn văn bản thành một vector có khả năng biểu diễn ý nghĩa ngữ cảnh của đoạn đó trong không gian số, từ đó phục vụ cho việc tìm kiếm, phân loại hoặc truy vấn thông minh.

#### Phương pháp thực hiện :

* **Dữ liệu đầu vào**: danh sách các đoạn văn bản chunks, trong đó mỗi phần tử là một từ điển gồm hai trường: "content" (nội dung văn bản) và "context" (ngữ cảnh).
* **Mô hình sử dụng**: mô hình embedding có hỗ trợ phương thức encode(), ví dụ như SentenceTransformer.

#### Các bước xử lý :

1. **Tách nội dung**: lấy trường "content" từ từng đoạn để tạo danh sách các đoạn văn bản.
2. **Sinh vector**: truyền danh sách văn bản vào hàm model.encode() để tạo ra các vector tương ứng, với kết quả được chuyển sang định dạng numpy nhằm tối ưu hóa cho các bước xử lý số tiếp theo.

#### Kết quả đầu ra

Một mảng ***numpy.ndarray,*** trong đó mỗi hàng tương ứng với một đoạn văn bản, biểu diễn nội dung của đoạn đó trong không gian vector.

Hàm này là bước tiền đề cho việc xây dựng chỉ mục tìm kiếm (indexing) trong hệ thống truy vấn ngữ nghĩa.

### Cơ chế truy xuất (retrieval) :

Hàm retrieve\_top\_k(query, model, index, chunks, k=3) được sử dụng để truy xuất **k đoạn văn bản liên quan nhất** đến câu truy vấn đầu vào dựa trên độ tương đồng ngữ nghĩa trong không gian vector.

#### Mục tiêu :

Tìm ra các đoạn văn bản trong tập dữ liệu đã mã hóa (embedding) mà có ý nghĩa gần nhất với truy vấn (query), hỗ trợ cho các bài toán như **tìm kiếm ngữ nghĩa**, **trả lời câu hỏi**, và **hệ thống RAG (retrieval-augmented generation)**.

#### Phương pháp thực hiện:

* **Truy vấn đầu vào (query)**: là một chuỗi văn bản do người dùng nhập.
* **Mô hình embedding (model)**: dùng để chuyển truy vấn thành vector.
* **Chỉ mục tìm kiếm (index)**: thường là một cấu trúc dữ liệu như FAISS, cho phép tìm kiếm nhanh các vector gần nhau.
* **Tập dữ liệu gốc (chunks)**: danh sách các đoạn văn bản ban đầu dùng để tra cứu.

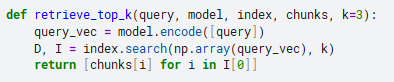
#### Các bước xử lý:

1. **Mã hóa truy vấn**: biến câu truy vấn thành vector bằng model.encode([query]).
2. **Tìm kiếm top-k**: sử dụng index.search() để tìm ra k vector gần nhất trong chỉ mục, trả về:
   * D: khoảng cách (không dùng trong hàm này, nhưng có thể dùng để đánh giá độ tương tự).
   * I: chỉ số (index) các đoạn phù hợp nhất.
3. **Truy xuất văn bản gốc**: dựa vào chỉ số I, trích xuất các đoạn tương ứng từ danh sách chunks.

#### Kết quả đầu ra:

Danh sách k đoạn văn bản phù hợp nhất với câu truy vấn, duy trì cả nội dung (content) và ngữ cảnh (context).

Hàm này đóng vai trò trung tâm trong pipeline của hệ thống tìm kiếm ngữ nghĩa, giúp liên kết truy vấn người dùng với thông tin có ý nghĩa trong tập dữ liệu.



Hình : code cho phần cơ chế truy xuất

### Xây dựng tập dữ liệu vector cho lưu trữ và truy vấn :

Hàm build\_vector\_dataset(chunk\_texts, contexts, embeddings, source, topic="general") được sử dụng để tạo ra một tập dữ liệu vector có cấu trúc, phục vụ cho việc lưu trữ, tìm kiếm và huấn luyện các mô hình học sâu có khả năng hiểu ngữ nghĩa.

#### Mục tiêu:

Kết hợp các thành phần: nội dung đoạn văn bản, ngữ cảnh, embedding vector và thông tin siêu dữ liệu (metadata) thành một tập dữ liệu có cấu trúc rõ ràng, phù hợp để lưu trữ vào cơ sở dữ liệu vector như FAISS, Weaviate, Qdrant hoặc Milvus.

#### Phương pháp thực hiện:

* **chunk\_texts**: danh sách các đoạn văn bản gốc.
* **contexts**: danh sách ngữ cảnh tương ứng với từng đoạn văn bản.
* **embeddings**: danh sách vector biểu diễn từng đoạn văn bản (dưới dạng numpy.ndarray).
* **source**: thông tin về nguồn gốc tài liệu (ví dụ: tên tệp PDF).
* **topic** (tùy chọn): nhãn chủ đề chung cho toàn bộ dữ liệu.

#### Các bước xử lý:

1. **Xác nhận độ dài khớp nhau**: hàm sử dụng assert để đảm bảo rằng ba danh sách đầu vào có cùng độ dài, tránh lỗi không đồng bộ dữ liệu.
2. **Khởi tạo danh sách dữ liệu**: duyệt qua từng phần tử để xây dựng một từ điển (dictionary) gồm:
   * id: mã định danh duy nhất (UUID).
   * content: nội dung đoạn văn bản.
   * context: tiêu đề hoặc ngữ cảnh liên quan.
   * topic: chủ đề tổng quát.
   * embedding: vector biểu diễn ngữ nghĩa (chuyển về dạng danh sách để dễ lưu trữ JSON).
   * source: tên hoặc đường dẫn của tài liệu nguồn.
3. **Trả về kết quả**: danh sách các entry dạng dictionary sẵn sàng cho lưu trữ hoặc indexing.

#### Kết quả đầu ra:

Danh sách các đối tượng JSON-like, mỗi đối tượng đại diện cho một đoạn văn bản có embedding và metadata đầy đủ. Tập dữ liệu này có thể được đưa vào các hệ thống truy vấn vector hoặc các pipeline xử lý ngữ nghĩa.

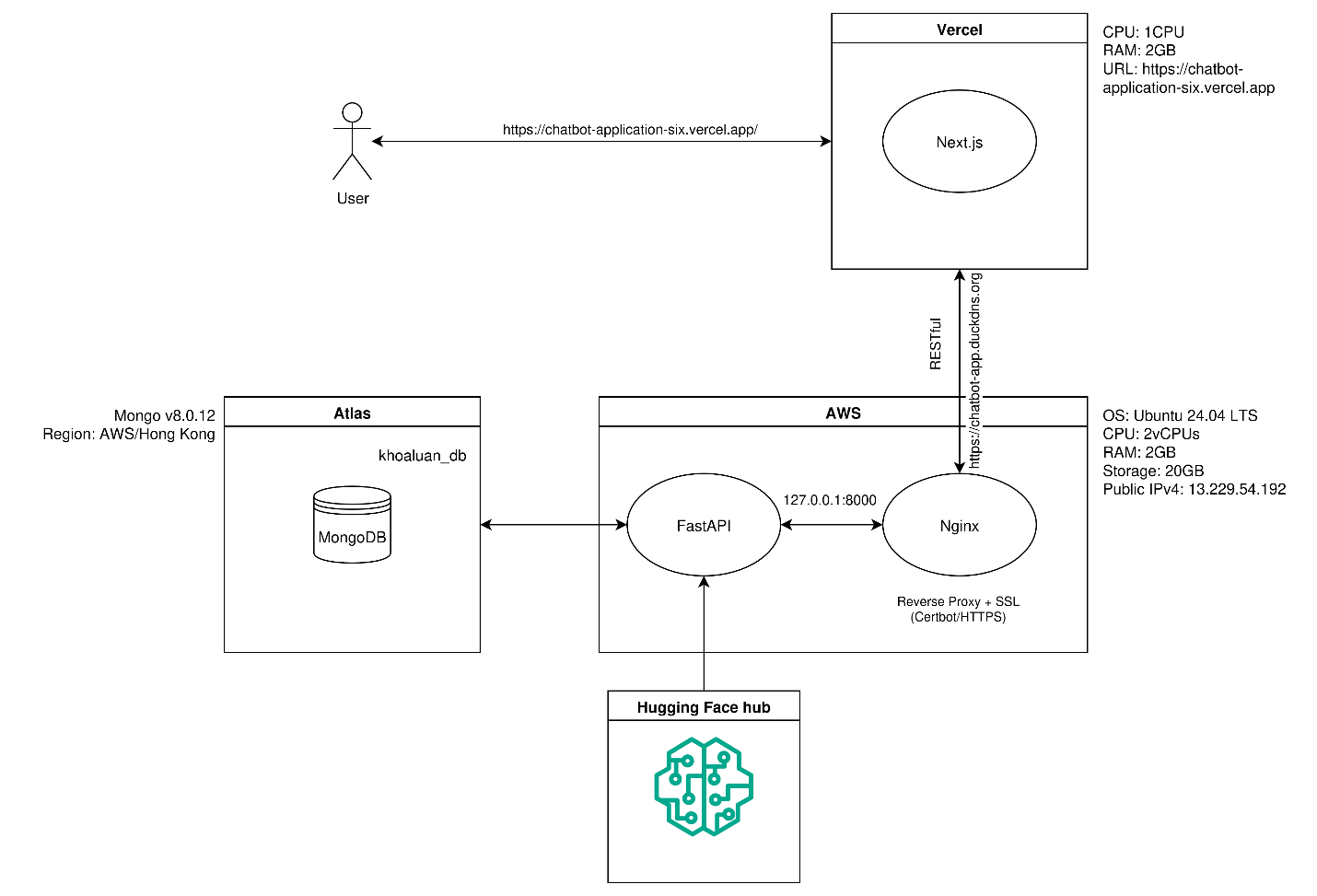
Hàm này đóng vai trò là bước kết nối giữa giai đoạn tiền xử lý văn bản và hệ thống lưu trữ, đảm bảo tính toàn vẹn dữ liệu và khả năng truy xuất hiệu quả trong các hệ thống NLP hiện đại.



Hình : code tạo tập dữ liệu vector có cấu trúc

# Triển khai thực hiện

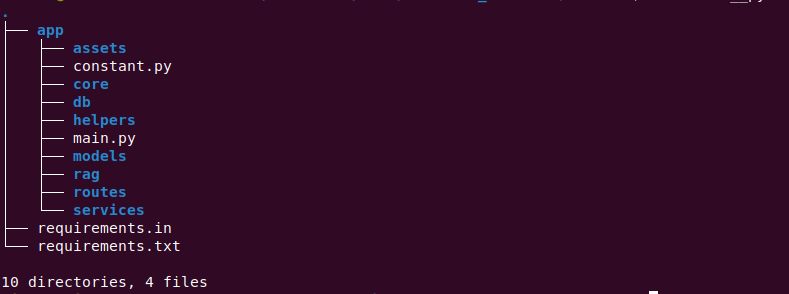
## Kiến trúc tổng quan hệ thống



## Phát triển Backend với FastAPI và MongoDB

Phần backend của chương trình được xây dựng để quản lý và xử lý các yêu cầu được gửi từ giao diện người dùng, điều phối quá trình xử lý ngôn ngữ tự nhiên và tương tác với cơ sở dữ liệu.

### FastAPI

FastAPI được chọn làm framework phát triển API backend nhờ khả năng xây dựng nhanh các API hiệu năng cao và dễ mở rộng. Nền tảng này tận dụng các tính năng của Python hiện đại như type hints (gợi ý kiểu dữ liệu) và tích hợp sẵn công cụ validation dữ liệu Pydantic, giúp đảm bảo tính đúng đắn của dữ liệu đầu vào và đầu ra của API. Đồng thời, FastAPI cũng tự động tạo tài liệu API tương tác theo chuẩn OpenAPI và Swagger UI, hỗ trợ mạnh mẽ cho quá trình phát triển và kiểm thử. Kiến trúc của backend được tổ chức theo từng module chức năng rõ ràng, bao gồm các endpoint cho người dùng (User), thông tin của người dùng (Profile), cuộc trò chuyện (Conversation), nội dung cuộc trò chuyện (Message),… Các API này được thiết kế theo chuẩn RESTful, cung cấp giao diện giao tiếp đồng nhất với frontend.  


### Tích hợp mô hình và RAG

Trong kiến trúc backend, FastAPI đóng vai trò trung tâm trong việc điều phối tương tác giữa yêu cầu của người dùng, hệ thống Retrieval-Augmented Generation (RAG) và mô hình ngôn ngữ lớn (LLM) đã được tinh chỉnh. Mục tiêu là đảm bảo mọi phản hồi của chatbot không chỉ mang tính hội thoại mà còn được căn cứ vào thông tin chính xác từ bộ dữ liệu tri thức đã được huấn luyện ở bước Tinh chỉnh mô hình trước đó.

Khi một yêu cầu câu hỏi được gửi tới endpoint của hệ thống, luồng xử lý được kích hoạt. Ban đầu, nội dung câu hỏi từ người dùng được chuyển đến module truy xuất (retriever) của hệ thống RAG. Module này sử dụng mô hình nhúng (embedding model), cụ thể là SentenceTransformer("all-MiniLM-L6-v2"), để chuyển đổi câu hỏi thành biểu diễn vector. Sau đó, một thuật toán tìm kiếm tương đồng (ví dụ: tìm kiếm láng giềng gần nhất trên chỉ mục FAISS đã được tải) được thực hiện để xác định và trích xuất các đoạn văn bản có ngữ cảnh liên quan nhất từ cơ sở dữ liệu tri thức đã được vector hóa.

Tiếp theo, các đoạn văn bản ngữ cảnh được truy xuất này cùng với câu hỏi gốc của người dùng được sử dụng để xây dựng một prompt đầu vào cho LLM. Quá trình tạo prompt này bao gồm việc phát hiện ngôn ngữ của câu hỏi (langdetect) và phân loại loại câu hỏi (how, why, definition, comparison, default) để chọn ra một hướng dẫn (instruction) phù hợp từ tập các template đã định nghĩa trước cho tiếng Việt hoặc tiếng Anh. Hướng dẫn này được nhúng vào prompt để định hướng phản hồi của mô hình.

Sau khi prompt được hoàn thiện, nó được gửi tới mô hình để thực hiện quá trình suy luận (inference). Mô hình sẽ tạo ra một câu trả lời dựa trên cả câu hỏi và ngữ cảnh được cung cấp. Câu trả lời này sau đó được thu nhận, và một bản ghi tin nhắn mới cho phản hồi của chatbot được lưu trữ vào cơ sở dữ liệu, đồng thời cập nhật thời gian chỉnh sửa gần nhất của cuộc hội thoại liên quan. Cuối cùng, phản hồi từ chatbot được gửi trả về phía frontend để hiển thị cho người dùng, hoàn thành chu trình tương tác.

### MongoDB

MongoDB được sử dụng làm cơ sở dữ liệu chính cho backend do tính linh hoạt của mô hình tài liệu (document-oriented database), phù hợp cho việc lưu trữ dữ liệu phi cấu trúc như lịch sử trò chuyện của người dùng. Mỗi phiên trò chuyện có thể được lưu trữ dưới dạng một tài liệu JSON, cho phép dễ dàng thêm các trường dữ liệu mới (ví dụ: thời gian, ID người dùng, nội dung câu hỏi/trả lời) mà không yêu cầu thay đổi lược đồ phức tạp. Việc này đơn giản hóa quá trình phát triển và cho phép khả năng mở rộng linh hoạt khi lượng dữ liệu và số lượng người dùng tăng lên trong tương lai. Backend tương tác với MongoDB để lưu trữ các đoạn hội thoại, giúp duy trì ngữ cảnh cho các cuộc trò chuyện kéo dài và cho phép người dùng xem lại lịch sử tương tác của họ.

## Phát triển Fronend với Next.js

Phần giao diện người dùng (frontend) của ứng dụng chatbot được xây dựng bằng Next.js phiên bản 15, sử dụng kiến trúc App Router. Mục tiêu là tạo ra một giao diện trực quan, phản hồi nhanh và tối ưu hóa trải nghiệm tương tác cho sinh viên.

### Lựa chọn và Kiến trúc Next.js

Chúng tôi đã chọn Next.js làm framework phát triển frontend do khả năng hỗ trợ Server-Side Rendering (SSR) và Static Site Generation (SSG) linh hoạt, cho phép tối ưu hóa hiệu suất tải trang và khả năng SEO của ứng dụng web. Đặc biệt, việc sử dụng App Router trong Next.js 15 mang lại một mô hình kiến trúc hiện đại, tập trung vào các thành phần React Server Components và Server Actions. Điều này cho phép phân tách rõ ràng logic phía máy chủ và phía máy khách, giúp cải thiện hiệu suất bằng cách giảm lượng JavaScript được gửi đến trình duyệt và tăng cường khả năng xử lý dữ liệu trực tiếp trên server, đồng thời đơn giản hóa việc quản lý trạng thái và tương tác với API backend.

### Tương tác với Backend

Frontend giao tiếp với backend FastAPI thông qua các yêu cầu API RESTful bất đồng bộ. Khi người dùng thực hiện gửi một yêu cầu thông qua các api được phía backend cung cấp, frontend sẽ chờ đợi phản hồi từ backend, sau đó cập nhật giao diện người dùng để hiển thị các thay đổi tương ứng với dữ liệu được phía backend xử lý. Việc quản lý trạng thái của ứng dụng được thực hiện thông qua các hook của React, đảm bảo giao diện luôn được đồng bộ với dữ liệu từ backend và mang lại trải nghiệm tương tác mượt mà, không bị gián đoạn.

## Quy trình triển khai và môi trường vận hành

Để đưa ứng dụng chatbot vào hoạt động và đảm bảo tính khả dụng, chúng tôi đã triển khai các thành phần frontend và backend trên các môi trường đám mây khác nhau, kết hợp các dịch vụ tối ưu cho từng mục đích.

### Triển khai Frontend

Giao diện người dùng của ứng dụng chatbot được triển khai trên nền tảng **Vercel** sử dụng gói miễn phí. Vercel là một nền tảng được tối ưu hóa cho các ứng dụng [Next.js](http://next.js), cung cấp khả năng triển khai nhanh chóng, tích hợp liên tục (CI/CD) tự động từ kho mã nguồn, và phân phối nội dung thông qua mạng lưới CDN toàn cầu. Điều này đảm bảo ứng dụng frontend có thời gian phản hồi nhanh và khả năng truy cập ổn định cho người dùng. Địa chỉ truy cập công khai của ứng dụng là [**https://chatbot-application-six.vercel.app**](https://chatbot-application-six.vercel.app).

### Triển khai Backend và Quản lý Cơ sở hạ tầng

Phần backend của hệ thống được triển khai trên một máy chủ ảo **Amazon EC2** của Amazon Web Services (AWS) với cấu hình Ubuntu 24.04 LTS, 2GB RAM và 20GB bộ nhớ trong. Địa chỉ IPv4 công khai của máy chủ là 13.229.54.192. Để đảm bảo bảo mật và khả năng kết nối HTTPS từ frontend (Vercel) đến backend, chúng tôi đã cấu hình máy chủ web **Nginx** làm reverse proxy.

Nginx được thiết lập để chuyển tiếp các yêu cầu từ tên miền [chatbot-app.duckdns.org](http://chatbot-app.duckdns.org) (cũng như từ địa chỉ IP công khai) tới ứng dụng FastAPI đang chạy nội bộ trên cổng 8000 (HTTP). Cấu hình Nginx bao gồm việc chuyển hướng mọi truy cập HTTP sang HTTPS và xử lý chứng chỉ SSL. Chứng chỉ SSL được ký bởi **Certbot** (sử dụng Let's Encrypt), đảm bảo giao tiếp mã hóa giữa client và server theo chuẩn bảo mật hiện hành. Điều này khắc phục vấn đề lỗi kết nối từ môi trường HTTPS của Vercel tới backend HTTP thuần túy.

### Cơ sở dữ liệu và Lưu trữ dữ liệu nhúng

Hệ thống sử dụng **MongoDB Atlas** với gói miễn phí để quản lý cơ sở dữ liệu phi quan hệ. Cluster KieranCluster và cơ sở dữ liệu khoaluan\_db được sử dụng để lưu trữ lịch sử trò chuyện và thông tin người dùng, cung cấp một giải pháp cơ sở dữ liệu linh hoạt và có khả năng mở rộng.

Đối với hệ thống RAG, các vector nhúng (embeddings) của tài liệu chuyên ngành cùng với chỉ mục FAISS được lưu trữ trực tiếp trên máy chủ backend EC2. Cụ thể, dữ liệu metadata được lưu trữ tại **backend/app/assets/all\_vector\_data.json** và chỉ mục FAISS tại **backend/app/assets/all\_vector\_index.faiss**. Điều này đảm bảo tốc độ truy xuất ngữ cảnh nhanh chóng, do dữ liệu nhúng được tải vào bộ nhớ khi ứng dụng backend khởi động thông qua mô hình Singleton, giảm thiểu độ trễ trong quá trình truy vấn RAG.

# TÀI LIỆU THAM KHẢO

# PHỤ LỤC

1. Tập hợp các văn bản hoặc lời nói có thể đọc được bằng máy, được sử dụng để đào tạo các hệ thống AI Xử lý ngôn ngữ tự nhiên [↑](#footnote-ref-1)
2. Quá trình điều chỉnh một mô hình đã được huấn luyện trước đó trên một tập dữ liệu nhỏ hơn, cụ thể hơn cho một nhiệm vụ nhất định. [↑](#footnote-ref-2)
3. Mô hình đã được huấn luyện ban đầu trên một lượng lớn dữ liệu tổng quát trước khi được tinh chỉnh cho một mục tiêu cụ thể. [↑](#footnote-ref-3)
4. Hiện tượng mô hình học dữ liệu huấn luyện quá kỹ, dẫn đến việc kém hiệu quả trên dữ liệu mới chưa từng thấy. [↑](#footnote-ref-4)