ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ TRI THỨC

BÁO CÁO ĐỒ ÁN CUỐI KỲ

| Đề tài: 101 |

HUẤN LUYỆN MÔ HÌNH NGÔN NGỮ LỚN ĐỂ TRẢ LỜI CÂU HỎI TRẮC NGHIỆM TRONG ĐỀ THI THPT QUỐC GIA

| Giảng viên hướng dẫn |

TS NGUYỄN HỒNG BẢO LONG

TS ĐINH ĐIỀN

ThS LƯƠNG AN VINH

| Sinh viên thực hiện đồ án |

22127006 NGUYỄN DUY ÂN

22127120 CAO NGUYỄN HUY HOÀNG

22127195 ĐỖ LÊ KHOA

MÔN HỌC: NHẬP MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

THÀNH PHỐ HỒ CHÍ MINH – 2024

MỤC LỤC

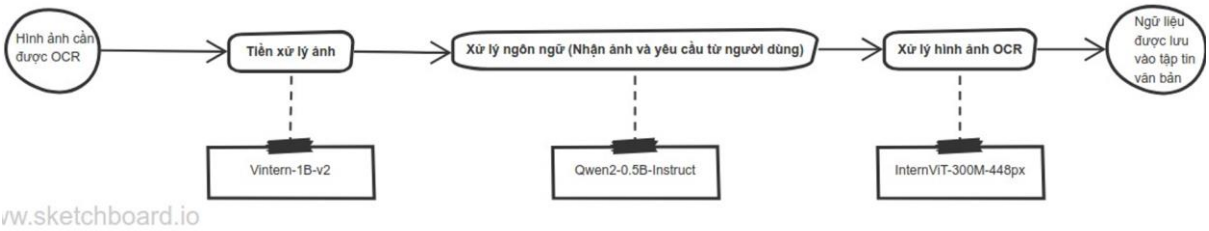
1. Quá trình OCR:

1.1. OCR bằng Tesseract-OCR:

- Tesseract thực hiện OCR bằng cách nhận dạng kí tự theo dòng và theo mẫu.
- Tesseract hỗ trợ nhiều định dạng ảnh đầu vào như JPEG, PNG, TIFF và định dạng văn bản đầu ra như: văn bản thuần túy, HTML, PDF, … [1]
- Người dùng cần phải tải Tesseract-OCR về thiết bị để có thể sử dụng.

1.2. OCR bằng Mô hình ngôn ngữ lớn (LLM):

- Mô hình ngôn ngữ lớn được dung để OCR: Vintern 1B v2.
- Mô hình Vintern 1B v2 đã được fine-tuned từ 2 mô hình:
  - Nhận dạng: Mô hình InternViT-300M-448px. [2]
  - Xử lý ngôn ngữ: Mô hình Qwen2-0.5B-Instruct. [3]
- Người dùng có thể sử dụng mô hình trên các nền tảng đám mây như Google Colab.



Hình 1: Sơ đồ quy trình OCR của mô hình Vintern 1B v2.

1.3. So sánh các phương pháp OCR:

	Tesseract	Vintern 1B v2
Độ chính xác	Độ chính xác cao, tuy nhiên có thể phải yêu cầu ảnh có chất lượng tốt.	Độ chính xác cao, tuy nhiên yêu cầu phần prompt từ người dùng phải tốt và hợp lý.
Bộ nhớ	Yêu cầu người dùng tải về và tải riêng bộ dữ liệu đã huấn luyện cho từng ngôn ngữ nếu cần.	Người dùng có thể sử dụng nền tảng đám mây như Google Colab để sử dụng mô hình mà không tiêu tốn dung lượng.
Thời gian	Nhanh hơn do xử lý trực tiếp trên ảnh.	Chậm hơn do phải xử lý prompt trước khi xử lý ảnh.
Cách xử lý	Lấy dữ liệu từ mô hình cục bộ.	Chuyển hóa ký tự từ prompt thành token.

2. Phương pháp Chunking Text:

- Chunking Text là kỹ thuật dùng để chia nhỏ thông tin thành các cụm nhỏ hơn để xử lý.
- Trong đồ án, chúng em đã tìm hiểu 2 phương pháp bao gồm Sematic Chunking và Recursive Chunk. Chúng em đã chọn Recursive Chunk để chia cắt văn bản.

2.1. Semantic Chunk:

- Thực hiện cắt văn bản OCR thành các đoạn văn bản có liên kết với nhau về mặt ý nghĩa.
- Cách thực hiện:
  - Sử dụng 1 mô hình ngôn ngữ như 1 từ điển.
  - Sử dụng 1 tập hợp chứa tất cả các câu từ tập tin chứa nội dung của trang vừa OCR (Các câu ban đầu cách nhau bởi dấu chấm ".").
  - Thực hiện cắt nội dung có ý nghĩa của trang thành từng đoạn nhỏ và lưu vào 1 danh sách.

2.2. Recursive Chunk:

- Thực hiện cắt văn bản OCR theo người dùng quy định.
- Cách thực hiện:
  - Người dùng quy định số lượng ký tự của 1 chunk.
  - Sử dụng 1 tập hợp chứa tất cả các câu từ tập tin chứa nội dung của trang vừa OCR (Các câu ban đầu cách nhau bởi dấu chấm ".").
  - Thực hiện cắt nội dung của trang thành từng đoạn đến khi đủ số lượng kí tự hoặc khi đủ 1 câu đầy đủ và lưu vào danh sách.

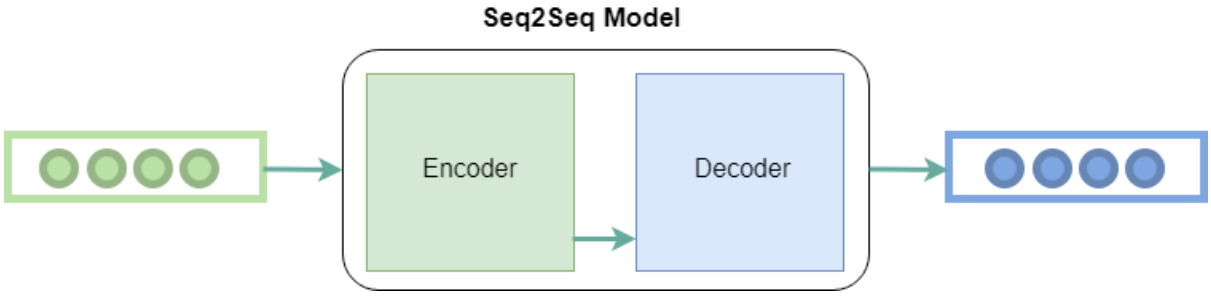
3. Mô hình tạo câu hỏi:

3.1. Mô hình tạo câu hỏi từ nội dung:

- Mô hình được sử dụng là: msmarco-vietnamese-mt5-base-v1.
- Đây là 1 mô hình ngôn ngữ Sequence-to-Sequence.
- Mô hình này có thể tạo 25 đến 40 truy vấn và đánh chỉ số cho các câu truy vấn trên. Các câu truy vấn sẽ được xếp hạng dựa trên độ liên quan với đoạn văn. Ngoài ra, mô hình còn đánh giá lại trọng số của từ để xem xét tính quan trọng của từ mặc dù từ đó ít xuất hiện trong câu. [4]
- Mô hình có thể được dùng để tạo ra dữ liệu huấn luyện để học mô hình nhúng.

3.2. Mô hình ngôn ngữ Sequence-to-Sequence (Seq2Seq):

- Đây là dạng kiến trúc học máy được dùng để xử lý các tác vụ có liên đến các chuỗi dữ liệu.
- Encoder và Decoder là hai thành phần chính của dạng mô hình sequence-to-sequence.
- Mô hình Seq2Seq nhận dữ liệu đầu vào là 1 chuỗi dữ liệu và truyền vào Encoder.
    - Encoder xử lý và lưu trữ thông tin quan trọng của chuỗi dữ liệu và lưu vào 1 trạng thái ẩn được gọi là 1 vector "ngữ cảnh". Vector sau đó được truyền vào Decoder.
    - Decoder sử dụng vector "ngữ cảnh" để có thể "hiểu" được chuỗi dữ liệu được truyền vào và tạo ra chuỗi dữ liệu đầu ra tương ứng bằng cách tạo sinh theo kiểu tự hồi quy. Tại mỗi bước, Decoder sử dụng các thành phần được tạo ra ở các bước trước, vector "ngữ cảnh" và dữ liệu đầu vào để dự đoán thành phần kế tiếp của dữ liệu đầu ra. Quá trình sẽ dừng lại khi dữ liệu đầu ra đã được hoàn thiện. [5]



Hình 2. Sơ đồ xử lý dữ liệu của mô hình Sequence-to-Sequence. [5]

3.3. Cấu trúc mô hình gốc:

- Mô hình msmarco-vietnamese-mt5-base-v1 được dựa trên mô hình mT5.
- T5 là mô hình encoder-decoder được huấn luyện trước (pretrained) trên một tập hợp đa tác vụ bao gồm nhiều tác vụ giám sát và không giám sát. Mỗi tác vụ được chuyển hóa thành định dạng text-to-text. [6]
- mT5 là mô hình đa ngôn ngữ của T5, được pretrained trên mC4 corpus, bao gồm 101 ngôn ngữ. [6]

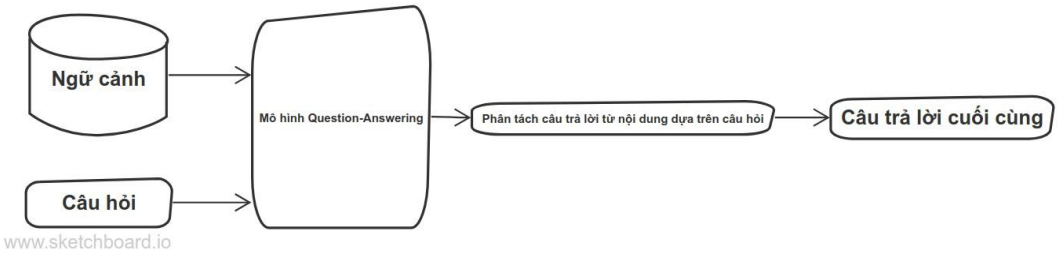
4. Mô hình tạo câu trả lời

4.1. Mô hình tạo câu trả lời từ nội dung và câu hỏi:

- Mô hình được sử dụng là xlm-roberta-large-vi-qa.
- Đây là 1 mô hình ngôn ngữ Question-Answering.
- Mô hình tokenize câu hỏi và nội dung để tạo ra câu trả lời.

4.2. Mô hình ngôn ngữ Question-Answering:

- Đây là mô hình dùng để tạo ra câu trả lời dựa trên câu hỏi và ngữ cảnh.
- Mô hình này có thể tách các cụm ý trả lời từ đoạn văn ngữ cảnh được cung cấp, tạo ra và diễn giải câu trả lời.
- Mô hình này phụ thuộc vào tập dữ liệu huấn luyện được huấn luyện và dạng vấn đề mà người dùng cần câu trả lời (có thể là tự luận, trắc nghiệm,…). [7]
- Mô hình Question-Answering cần phải hiểu được cấu trúc ngôn ngữ làm việc cùng, ý nghĩa về mặt câu từ của ngữ cảnh và câu hỏi và khả năng định vị các ý trả lời cho câu hỏi từ ngữ cảnh được trao. [7]



Hình 3. Sơ đồ xử lý dữ liệu của mô hình ngôn ngữ Question-Answering

4.3. Cấu trúc mô hình gốc:

- Mô hình xlm-roberta-large-vi-qa được dựa trên mô hình ngôn ngữ xlm-roberta-large của Meta.
- Đây là mô hình đa ngôn ngữ của RoBERTa được huấn luyện trên dữ liệu được lọc của CommonCrawl bao gồm 100 ngôn ngữ. [8]
- RoBERTa là mô hình cải thiện hơn từ mô hình BERT phát hành năm 2018 của Google. [9]
    - Điểm cải thiện của RoBERTa so với BERT trong kĩ thuật pretraining:
        - Mã hóa token động: Với mỗi epoch thì các token sẽ được mã hóa khác nhau.
        - Kết hợp câu: Các câu được kết hợp lại với nhau để đạt được 512 tokens.
        - Nhóm dữ liệu lớn hơn: Việc huấn luyện sử dụng nhóm dữ liệu lớn hơn nhiều so với BERT.

    - Byte-level BPE (Byte Pair Encoding) vocabulary: Sử dụng kĩ thuật BPE để làm việc trực tiếp với các giá trị byte thay vì kí tự, bỏ qua sự phụ thuộc vào mã hóa ký tự như UTF-8, UTF-16,…
- Mô hình này thường được dùng để fine-tune các downstream tasks.

5. Mô hình xác minh câu hỏi:

5.1. Mô hình xác minh câu hỏi:

- Mô hình được sử dụng là Meta-Llama-3-8B-Instruct.Q4_0 của Meta.
- Mô hình Llama 3 là một mô hình ngôn ngữ tự hồi quy sử dụng kiến trúc Transformers đã được tối ưu hóa. Mô hình có 2 kích thước là 8 tỷ và 70 tỷ tham số được pre-trained và instruction tuned. [10]
- Đầu vào của mô hình này chỉ nhận dữ liệu văn bản (text).
- Mô hình có thể xuất dữ liệu đầu ra là văn bản và code.

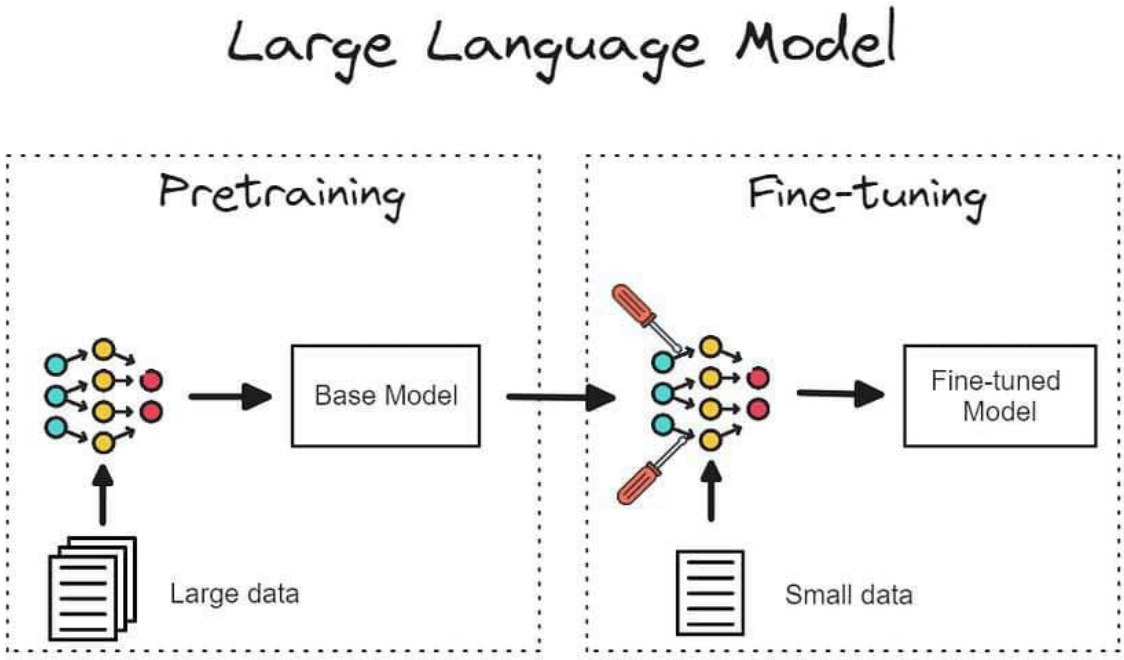
5.2. Kiến trúc Transformers:

- Đây là 1 dạng mô hình mạng nơ-ron (neural network) "học" ngữ cảnh của chuỗi dữ liệu đầu vào và tạo ra dữ liệu mới từ đó.
- Transformers là mô hình xử lý ngôn ngữ tự nhiên tân tiến và có thể được xem là phiên bản cải tiến hơn của kiến trúc Encoder-Decoder. Tuy nhiên, Encoder-Decoder phụ thuộc chủ yếu vào Recurrent Neural Networks (RNNs) để phân tách thông tin từ chuỗi dữ liệu, còn Transformers hoàn toàn không có đặc điểm này.
- Transformers được thiết kế để "hiểu" ngữ cảnh và ý nghĩa thông qua việc phân tích mối quan hệ giữa các thành phần khác nhau, và kiến trúc này phụ thuộc vào kĩ thuật toán học "Attention" để thực hiện việc này. [11]
    - Attention là cơ chế dùng để cải thiện hiệu suất của mô hình bằng cách tập trung vào các thông tin có liên quan. Cơ chế này cho phép mô hình đánh giá mức độ liên quan và quan trọng của các thành phần trong dữ liệu đầu vào.

Hình 4. Sơ đồ xử lý dữ liệu của kiến trúc Transformers. [11]

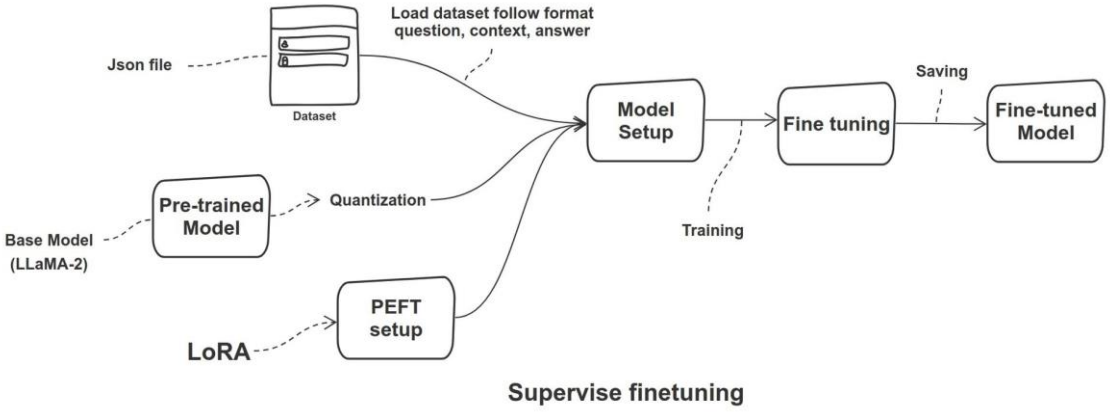
6. Fine-tuning:

- Fine-tuning là quá trình sử dụng 1 mô hình được huấn luyện từ trước và khiến mô hình đó "thích nghi" và thực hiện 1 hoặc 1 vài tác vụ cụ thể bằng cách huấn luyện với một tập dữ liệu nhỏ hơn, cụ thể và chuyên biệt hơn. Việc fine-tuning giúp cho mô hình có thể thực hiện 1 hoặc 1 vài tác vụ cụ thể với độ chính xác cao hơn mà không cần một tập dữ liệu khổng lồ hay tài nguyên máy tính. [12]



Hình 5. Quy trình fine-tuning của một mô hình ngôn ngữ lớn. [13]

- Fine-tuning giúp:
  - Hướng mô hình được huấn luyện đến thực hiện các tác vụ cụ thể một cách tối ưu hơn.
  - Dữ liệu đầu ra mong muốn có độ chính xác cao hơn.
  - Giảm thiểu dữ liệu đầu ra không liên quan đến câu hỏi.



Hình 6. Sơ đồ quá trình fine-tuning mô hình của nhóm.

6.1. Tiền xử lý bộ ngữ liệu để thành dữ liệu fine-tuning:

- Dữ liệu gốc của nhóm là các tập tin có định dạng là json. Tuy nhiên, dữ liệu gốc cần được biến đổi thành tập dữ liệu (dataset) để phù hợp cho quá trình fine-tune.
- Quá trình tiền xử lý bộ ngữ liệu:
    - Biến đổi tập tin json thành 1 dataset.
    - Thực hiện biến đổi dataset cho từng phần tử trong dataset:
        - Lấy phần câu hỏi, ngữ cảnh và câu trả lời từ tập tin json.
        - Kết hợp cả 3 để trở thành 1 chuỗi duy nhất gồm 3 phần câu hỏi, ngữ cảnh và câu trả lời.
    - Sau khi biến đổi, chỉ giữ lại cột có tên là "text", xóa tất cả các cột khác.

6.2. Thiết lập mô hình dùng để fine-tuning:

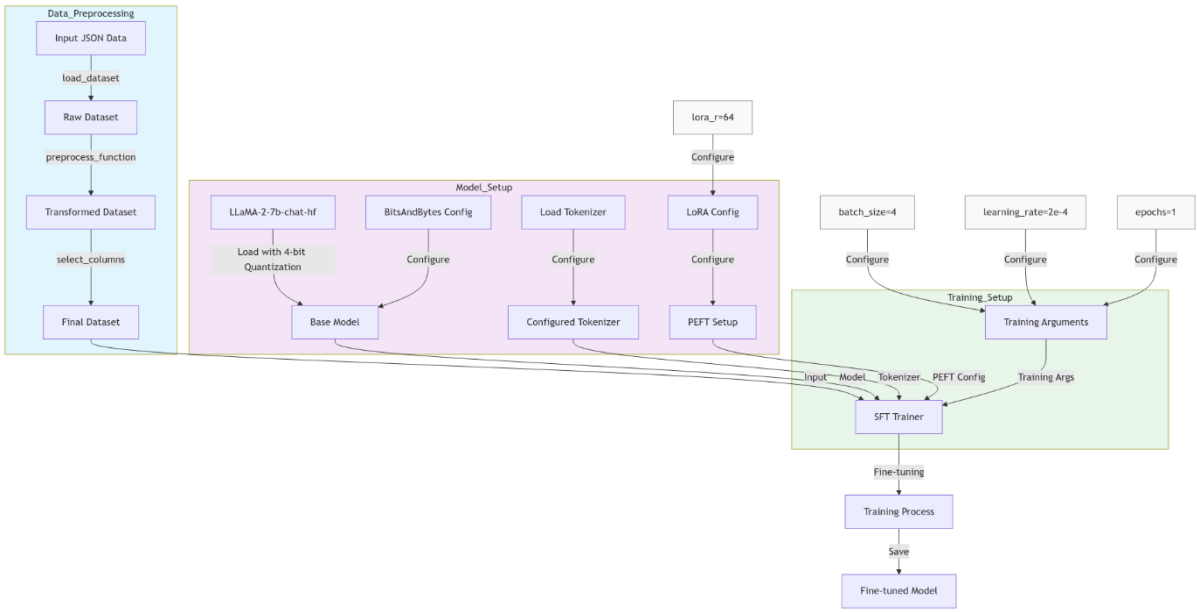
- Mô hình pretrained được sử dụng là LLaMA-2-7b-chat-hf.
    - Mô hình gốc LLaMA 2 là một mô hình ngôn ngữ tự hồi quy sử dụng kiến trúc Transformers đã được tối ưu hóa. Mô hình có 3 kích thước là 7, 13 và 70 tỷ tham số được pre-trained và instruction tuned. [14]
    - Đầu vào của mô hình này chỉ nhận dữ liệu văn bản (text).
    - Mô hình có thể xuất dữ liệu đầu ra là văn bản và code.
- Điểm khác nhau chính giữa LLaMA 2 và LLaMA 3:
    - LLaMA 2 được huấn luyện trên 2 tỷ tokens.
    - LLaMA 3 được huấn luyện trên 15 tỷ tokens.
- Mô hình trên kết hợp với kỹ thuật Lượng tử hóa (Quantization) bitsandbytes sẽ sinh ra mô hình gốc (Base model)
- Quantization là kĩ thuật làm giảm kích thước của mô hình bằng cách sử dụng kiểu dữ liệu có low-precision như số nguyên 4 bit hoặc 8 bit để biểu diễn các trọng số và hàm kích hoạt. Việc này hỗ trợ trong việc giảm chi phí bộ nhớ và tính toán, giúp thực hiện các phép nhân ma trận được thực hiện nhanh hơn với các phép tính chỉ gồm số nguyên. [15]
- Ngoài ra, Tokenizer và LoRA cần phải được tạo và tinh chỉnh để phù hợp với việc thực hiện fine-tuning.

- LoRA (Low-Rank Adaptation) là một phương pháp PEFT (Parameter-Efficient Fine-Tuning) dùng để phân rã một ma trận lớn thành 2 ma trận có hạng nhỏ hơn trong lớp attention. Phương pháp này giảm mạnh số lượng tham số cần được fine-tuned. [16]

6.3. Thiết lập các đối số (arguments) trong quá trình huấn luyện:

- Các đối số cần được đặt để quá trình huấn luyện được thực hiện.
- Một số đối số quan trọng như learning rate, batch size, epoch, optimizer,…
- Ngoài ra, các thiết lập của phần 6.1 và 6.2 cũng được thêm vào để thực hiện fine-tuning có giám sát (supervised fine-tuning).

6.4. Sơ đồ chi tiết quá trình thực hiện fine-tuning:



Hình 7. Sơ đồ chi tiết quá trình fine-tuning.

❖ So sánh mô hình trước và sau khi fine-tune:

- Trước khi fine-tune: Mô hình có thể trả lời không phải tiếng Việt và không đúng định dạng mong muốn.

```
Context: - Nền nhiệt độ thiên về khí hậu cận xích đạo, quanh năm nóng, nhiệt độ trung bình năm trên 25°C và không có tháng nào dưới 20°C.
Question: Nhiệt độ trung bình cận xích đạo?

Answer: The average temperature in the equatorial region is around 25°C (77°F) throughout the year, with no month dropping below 20°C (68°F
```

- Sau khi fine-tune: Mô hình trả lời bằng tiếng Việt, theo định dạng mong muốn và hợp lý hơn

```
Context: - Nền nhiệt độ thiên về khí hậu cận xích đạo, quanh năm nóng, nhiệt độ trung bình năm trên 25°C và không có tháng nào dưới 20°C.
Question: Nhiệt độ trung bình cận xích đạo?
Answer: 25°C.
```

7. Tài liệu tham khảo:

[1]. https://github.com/tesseract-ocr/tesseract/blob/main/README.md, 31/12/2024

[2]. https://huggingface.co/OpenGVLab/InternViT-300M-448px, 31/12/2024

[3]. https://huggingface.co/Qwen/Qwen2-0.5B-Instruct, 31/12/2024

[4]. https://huggingface.co/doc2query/msmarco-vietnamese-mt5-base-v1, 31/12/2024

[5]. https://www.geeksforgeeks.org/seq2seq-model-in-machine-learning/, 2/1/2025

[6]. https://huggingface.co/docs/transformers/en/model_doc/t5, 31/12/2024

[7]. https://www.digitalocean.com/community/tutorials/how-to-train-question-answering-machine-learning-models, 2/1/2025

[8]. https://huggingface.co/FacebookAI/xlm-roberta-large, 31/12/2024

[9]. https://huggingface.co/docs/transformers/en/model_doc/roberta, 2/1/2025

[10]. https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct, 31/12/2024

[11]. https://www.datacamp.com/tutorial/how-transformers-work, 2/1/2025

[12]. https://www.geeksforgeeks.org/fine-tuning-large-language-model-llm/, 3/1/2025

[13]. https://medium.com/@prasadmahamulkar/fine-tuning-phi-2-a-step-by-step-guide-e672e7f1d009, 3/1/2025

[14]. https://huggingface.co/NousResearch/Llama-2-7b-chat-hf, 3/1/2025

[15]. https://huggingface.co/docs/optimum/en/concept_guides/quantization, 3/1/2025

[16]. https://huggingface.co/docs/peft/main/en/package_reference/lora, 3/1/2025