Báo cáo tuần 2

1. Sử dụng Llama3 trên máy local để extract thông tin

Các bước thực hiện

* Bước 1: tải các công cụ để triển khai và quản lý các mô hình ngôn ngữ lớn(Ollama, LM-studio)

Trong bài báo cáo này em sẽ sửu dụng LM-studio để tạo local model và lấy api để trích xuất thông tin

* Bước 2: Tải và cài đặt các mô hình ngôn ngữ lớn

Trong báo cáo em sẽ tải và sử dụng Meta-Llama-3.1-8B-Instruct

* Bước 3: Viết code và sử đụng prompt engineering

<https://github.com/thangcn1943/extract_with_Llama3.git>

Giải thích:

Import các thư viện cần thiết

Gửi request để lấy api của local-model

A computer screen with text

Description automatically generated

Tạo form dạng json để model tạo sinh thông tin

A black screen with text

Description automatically generated

Viết system\_message sử dụng các kĩ thuật prompt engineering để model đưa ra kết quả theo ý muốn

A screenshot of a computer program

Description automatically generated

Chọn lọc các thông tin dưới dạng json bằng thư viện re

Chuyển đổi từ dạng json thành xlsx và lưu vào file excel

A computer screen shot of text

Description automatically generated

* Đánh giá:
* Kết quả cho ra khá chính xác
* Nhưng vì local-model chưa sử dụng GPU nên thời gian để tạo sinh ra các thông tin khá lâu, do đó cần phải cải thiện và fine tune model để có được tốc độ tốt hơn
* Phương án tiếp theo: tìm các thức fine tune model để được tốc độ nhanh hơn

1. Sử dụng Llama3 trên colab để có tốc độ nhanh hơn

Bước 1: truy cập vào hugging face, tạo tài khoản, điền các thông tin cần thiết và chấp nhận chính sách của Hugging Face  
A screenshot of a computer

Description automatically generated

Bước 2: chọn setting

Setting -> access token -> create new token

Chọn write token sau đó lưu và lấy token

Bước 3: viết mã trong colab sử dụng token của hugging face

Mã nguồn ví dụ: <https://colab.research.google.com/drive/1nyWS9cO84xJNKMxy7dTaghp_LlbtaO8s#scrollTo=EaIM7R04eDq8>

Giải thích:

Cũng tương tự như sử dụng trên máy local

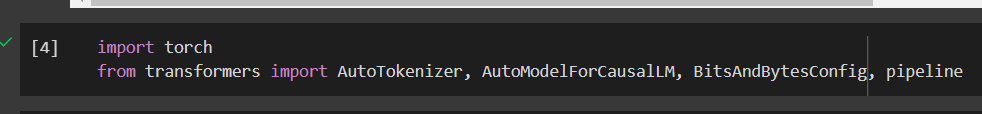
Vì colab hỗ trợ T4-GPU đến 15gb do đó tốc độ xử lí của các model sẽ nhanh hơn

Tải các thư viện cần sử dụng

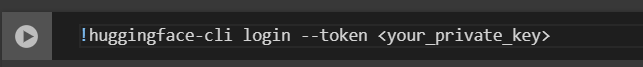
A screenshot of a computer screen

Description automatically generated

Import các module trong thư viện



Đây là cách để xác thực và thiết lập kết nối với tài khoản Hugging Face của bạn từ dòng lệnh.



Sử dụng quantization giúp giảm yêu cầu về bộ nhớ và cải thiện tốc độ tính toán, đồng thời vẫn duy trì chất lượng mô hình.A screen shot of a computer

Description automatically generated

* **BitsAndBytesConfig**: Đây là lớp cấu hình cho phép sử dụng các kỹ thuật quantization nhằm giảm yêu cầu về bộ nhớ và tăng tốc độ tính toán cho mô hình. Các tham số trong BitsAndBytesConfig giúp cấu hình việc nén mô hình xuống các bit nhỏ hơn mà vẫn giữ chất lượng tốt.
* **load\_in\_8bit=True**: Chỉ định rằng mô hình nên được tải với độ chính xác 8 bit, giúp giảm dung lượng bộ nhớ cần thiết để lưu trữ mô hình.
* **bnb\_4bit\_quant\_type="nf4"**: Chỉ định loại quantization 4 bit sử dụng loại nf4 (narrowing format 4-bit). Đây là kỹ thuật để nén mô hình xuống chỉ 4 bit cho mỗi trọng số.
* **bnb\_4bit\_use\_double\_quant=True**: Cho phép việc sử dụng double quantization (nén hai lần) để cải thiện chất lượng mô hình sau khi quantization.
* **bnb\_4bit\_compute\_dtype=torch.bfloat16**: Xác định kiểu dữ liệu tính toán (compute dtype) là bfloat16 từ thư viện PyTorch, giúp duy trì độ chính xác trong quá trình tính toán mặc dù mô hình đã được quantized.
* **tokenizer = AutoTokenizer.from\_pretrained(model\_id)**
* **AutoTokenizer.from\_pretrained(model\_id)**: Tải tokenizer tương ứng với mô hình từ Hugging Face. Tokenizer này sẽ phân tích và mã hóa văn bản đầu vào để mô hình có thể xử lý.
* **model = AutoModelForCausalLM.from\_pretrained(model\_id, quantization\_config=bnb\_config, device\_map="auto")**
* **AutoModelForCausalLM.from\_pretrained(model\_id, quantization\_config=bnb\_config, device\_map="auto")**: Tải mô hình từ Hugging Face với các cấu hình được chỉ định.
* **quantization\_config=bnb\_config**: Cung cấp cấu hình quantization đã được thiết lập để giảm dung lượng mô hình và cải thiện hiệu suất.
* **device\_map="auto"**: Tự động xác định và phân phối mô hình lên các thiết bị có sẵn (CPU hoặc GPU) để tối ưu hóa hiệu suất.

Bước 4: sử dụng prompt engineering để model đưa ra kết quả theo ý muốn

Kết quả: tốc độ đưa ra kết quả đã nhanh hơn nhưng vẫn cần cải thiện tốc độ để sử dụng vào mục đích tự động hóa

1. Phương án tiếp theo

Phương án tiếp theo để cải thiện hiệu suất của mô hình Llama3 là thực hiện các bước tinh chỉnh và tối ưu hóa mô hình. Đây là một quá trình quan trọng để làm cho mô hình chạy nhanh hơn và tiêu tốn ít tài nguyên hơn mà không làm giảm chất lượng của các dự đoán. Dưới đây là một kế hoạch chi tiết cho việc tinh chỉnh mô hình Llama3:

**1. Tinh chỉnh Mô Hình với Kỹ Thuật Quantization**

Quantization là một kỹ thuật giúp giảm kích thước mô hình và tăng tốc độ tính toán bằng cách giảm số lượng bit cần thiết để đại diện cho các trọng số của mô hình. Đối với mô hình Llama3, có thể sử dụng các kỹ thuật quantization sau:

* **Quantization 8-bit (8-bit Quantization)**: Chuyển các trọng số của mô hình từ dạng số thực 32-bit (float32) xuống dạng số nguyên 8-bit. Kỹ thuật này giảm dung lượng bộ nhớ và tăng tốc độ tính toán mà không làm giảm nhiều chất lượng của mô hình.
* **Quantization 4-bit (4-bit Quantization)**: Tiến xa hơn nữa với việc giảm số lượng bit xuống 4-bit. Mặc dù có thể ảnh hưởng đến chất lượng mô hình, nhưng kỹ thuật này cực kỳ hiệu quả trong việc giảm kích thước mô hình và cải thiện tốc độ tính toán.

**2. Sử Dụng Tinh Chỉnh Cấu Hình (Configuration Tuning)**

Tinh chỉnh cấu hình mô hình là một bước quan trọng để tối ưu hóa hiệu suất. Các cấu hình có thể điều chỉnh bao gồm:

* **Precision (Độ Chính Xác)**: Sử dụng các kiểu dữ liệu tính toán hiệu quả hơn, như bfloat16 hoặc float16, giúp tăng tốc tính toán trong khi giảm nhu cầu về bộ nhớ.
* **Layer Pruning (Cắt Tầng)**: Xóa bớt các tầng hoặc nơ-ron không quan trọng của mô hình để giảm số lượng tham số và tính toán cần thiết.

**3. Triển Khai và Tinh Chỉnh Thực Tế**

* **Fine-tuning (Tinh Chỉnh)**: Tinh chỉnh mô hình trên tập dữ liệu cụ thể của bạn để cải thiện khả năng dự đoán và phù hợp hơn với các yêu cầu của ứng dụng.
* **Model Compression (Nén Mô Hình)**: Sử dụng các kỹ thuật nén như quantization và pruning để giảm kích thước mô hình sau khi tinh chỉnh.