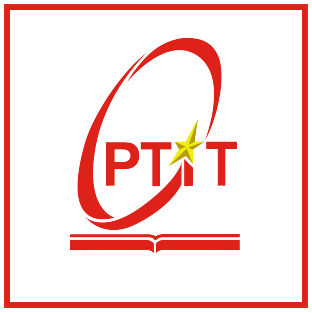
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

———————o0o———————



BÁO CÁO BTL

LLAMA3
CHO BÀI TOÁN
phân loại cảm xúc

Báo cáo viên:
NGUYỄN QUANG HUY

Hà Nội, 2024

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Lý thuyết

1.0.1 Bộ dữ liệu

1. **Phân tích cảm xúc trong tài chính và kinh tế:**

   - Phân tích cảm xúc trên thông tin tài chính và kinh tế rất quan trọng với các doanh nghiệp, bởi nó cung cấp thông tin về xu hướng thị trường, lòng tin của nhà đầu tư và hành vi của người tiêu dùng.
   - Phân tích cảm xúc giúp doanh nghiệp quản lý rủi ro, đặc biệt là những rủi ro về danh tiếng, cũng như đánh giá tiềm năng của các cơ hội đầu tư.

2. **Tìm dữ liệu thích hợp để tinh chỉnh mô hình:**

   - Trước khi đi vào các bước kỹ thuật tinh chỉnh mô hình ngôn ngữ lớn như Llama 3, cần tìm một bộ dữ liệu phù hợp để minh họa khả năng tinh chỉnh.
   - Trong lĩnh vực tài chính và kinh tế, dữ liệu được chú thích (annotated datasets) rất hiếm, phần lớn là tài sản độc quyền.

3. **Bộ dữ liệu FinancialPhraseBank:**

   - Để giải quyết vấn đề thiếu dữ liệu đào tạo, các nhà nghiên cứu từ Trường Kinh doanh Đại học Aalto đã giới thiệu vào năm 2014 một bộ dữ liệu với khoảng 5000 câu.
   - Bộ dữ liệu này được chú thích bởi 16 người có kiến thức chuyên môn về thị trường tài chính. Họ được hướng dẫn đánh giá các câu theo quan điểm của nhà đầu tư, dựa trên việc thông tin đó có tác động tích cực, tiêu cực hay trung lập đến giá cổ phiếu.
   - FinancialPhraseBank là một bộ sưu tập bao gồm các tiêu đề tin tức tài chính, được phân loại theo cảm xúc tiêu cực, trung lập hoặc tích cực từ góc nhìn của nhà đầu tư bán lẻ.

4. **Ứng dụng của bộ dữ liệu:**

   - Bộ dữ liệu FinancialPhraseBank này đã được sử dụng trong nhiều nghiên cứu để phân tích và hiểu rõ hơn về cảm xúc trong tin tức tài chính.
   - Nghiên cứu ban đầu của bộ dữ liệu này được công bố bởi Malo, P. và các cộng sự trong bài báo "Good debt or bad debt: Detecting semantic orientations in economic texts" vào năm 2014.

1.0.2 Các thư viện

1. **Thư viện cần cài đặt:**

   - `accelerate`: Đây là một thư viện hỗ trợ đào tạo phân tán của PyTorch, phát triển bởi HuggingFace. Thư viện này giúp huấn luyện mô hình trên nhiều GPU hoặc CPU cùng lúc (cấu hình phân tán), từ đó tăng tốc quá trình đào tạo khi sử dụng nhiều GPU. Tuy nhiên, trong ví dụ này, `accelerate` sẽ không được sử dụng.
   - `peft`: Đây là một thư viện Python của HuggingFace được thiết kế để điều chỉnh hiệu quả các mô hình ngôn ngữ đã được huấn luyện (PLMs) cho các ứng dụng sau khi đã huấn luyện mà không cần tinh chỉnh tất cả tham số của mô hình. `PEFT` chỉ tinh chỉnh một số lượng nhỏ tham số bổ sung, do đó giảm đáng kể chi phí tính toán và lưu trữ.

- `bitsandbytes`: Thư viện này được tạo bởi Tim Dettmers, là một lớp bao bọc nhẹ xung quanh các hàm tùy chỉnh CUDA, đặc biệt là các tối ưu hóa 8-bit, nhân ma trận (LLM.int8()) và các hàm lượng tử hóa. Nó cho phép chạy các mô hình được lưu trữ ở độ chính xác 4-bit, mặc dù trọng số được lưu trữ ở 4-bit, việc tính toán vẫn diễn ra ở độ chính xác 16-bit hoặc 32-bit (float16, bfloat16, float32, v.v.).
- `transformers`: Đây là một thư viện Python cho xử lý ngôn ngữ tự nhiên (NLP). Nó cung cấp một số mô hình đã được huấn luyện sẵn cho các tác vụ NLP như phân loại văn bản, trả lời câu hỏi và dịch thuật.
- `trl`: Đây là một thư viện đầy đủ các công cụ do HuggingFace phát triển để huấn luyện các mô hình ngôn ngữ transformer bằng phương pháp học tăng cường. Thư viện cung cấp các bước từ Huấn luyện Giám sát (SFT), Mô hình hóa Phần thưởng (RM) cho đến Tối ưu hóa Chính sách Tiệm cận (PPO).

1.0.3 Các bước thực hiện: tạo dataset

1. **Đọc dữ liệu đầu vào:**
   - Đọc dữ liệu từ tệp CSV có tên `all-data.csv`, trong đó có hai cột: `sentiment` (cảm xúc) và `text` (văn bản).
2. **Chia bộ dữ liệu:**
   - Bộ dữ liệu được chia thành hai phần: train set (bộ huấn luyện) và test set (bộ kiểm tra), mỗi phần có 300 mẫu.
   - Việc chia dữ liệu đảm bảo tính đại diện theo cảm xúc (phân chia có stratification), tức là mỗi bộ dữ liệu đều chứa đủ các mẫu có cảm xúc tích cực, trung tính và tiêu cực.
3. **Trộn dữ liệu huấn luyện:**
   - Dữ liệu huấn luyện được trộn lại theo thứ tự có thể tái lặp được, sử dụng tham số `random_state=10` để đảm bảo quá trình trộn có thể được lặp lại trong các lần chạy khác nhau.
4. **Chuyển đổi văn bản thành các gợi ý cho Llama:**
   - Văn bản từ bộ train và test được chuyển đổi thành các prompts (gợi ý) để sử dụng cho mô hình Llama. Trong các gợi ý thuộc bộ huấn luyện, câu trả lời mong đợi (cảm xúc) được bao gồm để mô hình học cách dự đoán.
5. **Chuẩn bị dữ liệu đánh giá:**
   - Các mẫu còn lại không nằm trong bộ huấn luyện hoặc kiểm tra sẽ được sử dụng làm dữ liệu đánh giá để báo cáo kết quả trong quá trình huấn luyện. Tuy nhiên, chúng sẽ không được sử dụng để dừng sớm quá trình huấn luyện.
   - Vì số lượng mẫu tiêu cực khá ít nên những mẫu này sẽ được lấy lại nhiều lần để tạo ra một mẫu đánh giá có tỷ lệ 50/50/50 (tương ứng giữa các cảm xúc tích cực, trung tính và tiêu cực).
6. **Sử dụng lớp của Hugging Face để bọc dữ liệu:**
   - Bộ dữ liệu huấn luyện (`train_data`), đánh giá (`eval_data`) và kiểm tra (`test_data`) được đóng gói bằng các lớp từ thư viện Hugging Face để tiện sử dụng trong quá trình tinh chỉnh mô hình.

Đoạn này mô tả các bước mà hàm đánh giá kết quả từ mô hình phân tích cảm xúc đã được tinh chỉnh sẽ thực hiện. Dưới đây là các bước cụ thể:

1. **Chuyển đổi nhãn cảm xúc thành dạng số:**
   - Nhãn cảm xúc sẽ được ánh xạ sang các giá trị số tương ứng:
     - 2 đại diện cho tích cực.
     - 1 đại diện cho trung tính.
     - 0 đại diện cho tiêu cực.
2. **Tính độ chính xác của mô hình trên bộ dữ liệu kiểm tra:**
   - Hàm sẽ tính toán tỷ lệ chính xác (accuracy) của mô hình khi dự đoán các cảm xúc trong bộ `test_data` (bộ dữ liệu kiểm tra). Độ chính xác cho biết tỷ lệ dự đoán đúng so với tổng số dự đoán.
3. **Tạo báo cáo độ chính xác cho từng nhãn cảm xúc:**
   - Báo cáo độ chính xác riêng biệt sẽ được tạo cho từng loại cảm xúc (tích cực, trung tính và tiêu cực) để xem mô hình hoạt động như thế nào với mỗi loại cảm xúc.
4. **Tạo báo cáo phân loại:**
   - Classification report là một báo cáo tổng hợp cung cấp các chỉ số quan trọng cho mô hình, chẳng hạn như precision (độ chính xác), recall (tỷ lệ phát hiện), và F1-score cho từng nhãn cảm xúc. Điều này giúp đánh giá chi tiết hơn về hiệu suất của mô hình.
5. **Tạo ma trận nhầm lẫn (confusion matrix):**
   - Confusion matrix là một bảng hiển thị số lượng dự đoán đúng và nhầm lẫn cho từng loại cảm xúc. Ma trận này giúp hiểu rõ mô hình đã phân loại đúng hay nhầm lẫn giữa các cảm xúc như thế nào, ví dụ: mô hình có thể nhầm lẫn giữa cảm xúc trung tính và tiêu cực.

Tóm lại, hàm này được sử dụng để đánh giá toàn diện về hiệu suất của mô hình phân tích cảm xúc sau khi đã tinh chỉnh, giúp phân tích độ chính xác, hiểu rõ hơn về khả năng phân loại của mô hình và các lỗi dự đoán.

1.0.4 Giải thích quy trình kiểm thử mô hình không tinh chỉnh và các bước chuẩn bị cho việc tải mô hình cùng với lượng tử hóa mô hình

Tải và lượng tử hóa mô hình:

1. **Tải mô hình:**
   - Mô hình ngôn ngữ Llama-3 (với 8 tỷ tham số, không có RLHF – Reinforcement Learning from Human Feedback) được tải từ Kaggle Models. Đây là mô hình tương thích với thư viện Hugging Face.
2. **Sử dụng kiểu dữ liệu float16 từ thư viện Torch:**
   - Kiểu dữ liệu float16 sẽ được dùng cho các phép tính toán trên mô hình, giúp giảm yêu cầu về bộ nhớ và tăng tốc độ tính toán.

3. **Tạo đối tượng BitsAndBytesConfig với các thiết lập sau:**
    - `load_in_4bit`: Mô hình sẽ tải trọng số dưới dạng 4-bit, giúp giảm kích thước bộ nhớ cần thiết để lưu trọng số.
    - `bnb_4bit_quant_type`: Sử dụng kiểu lượng tử hóa `nf4` (4-bit NormalFloat), là một kiểu dữ liệu mới tối ưu về mặt thông tin cho các trọng số phân phối chuẩn.
    - `bnb_4bit_compute_dtype`: Sử dụng kiểu dữ liệu float16 để tính toán.
    - `bnb_4bit_use_double_quant`: Không sử dụng lượng tử hóa kép, điều này giúp giảm mức sử dụng bộ nhớ trung bình vì lượng tử hóa cả các hằng số lượng tử hóa, tiết kiệm thêm 0.4 bit cho mỗi tham số.
4. **Tạo đối tượng AutoModelForCausalLM từ mô hình Llama-2 đã được huấn luyện trước:**
    - Sử dụng cấu hình `BitsAndBytesConfig` đã thiết lập để lượng tử hóa mô hình.
5. **Vô hiệu hóa tính năng caching cho mô hình:**
    - Điều này có thể nhằm tiết kiệm tài nguyên hệ thống hoặc phù hợp với việc kiểm thử mô hình.
6. **Đặt xác suất token tiền huấn luyện về 1:**
    - Điều này có thể liên quan đến việc điều chỉnh mô hình trong quá trình kiểm thử hoặc huấn luyện tiếp tục.

Tải và thiết lập Tokenizer:

1. **Tải tokenizer cho mô hình Llama-3:**
    - Tokenizer là thành phần chuyển đổi văn bản thành các token mà mô hình có thể xử lý.
2. **Thiết lập token lấp đầy là token EOS (End-of-Sequence):**
    - Token EOS thường được sử dụng để đánh dấu sự kết thúc của chuỗi văn bản, và trong trường hợp này, nó cũng được dùng để lấp đầy các chuỗi ngắn hơn nhằm chuẩn bị cho đầu vào mô hình.
3. **Thiết lập hướng lấp đầy (padding side) là "phải":**
    - Điều này có nghĩa là các chuỗi ngắn hơn sẽ được lấp đầy ở phía bên phải. Đây là cách Llama 2 thực hiện việc lấp đầy, và nó cần được làm chính xác để mô hình xử lý đầu vào đúng cách.

Như vậy, toàn bộ quá trình này nhằm chuẩn bị mô hình Llama-3 và tokenizer cho quá trình kiểm thử mà không tinh chỉnh (fine-tuning), giúp đánh giá hiệu suất mô hình trong trạng thái đã được huấn luyện trước, nhưng chưa được điều chỉnh cho tác vụ cụ thể như phân tích cảm xúc.

1.0.5 Giải thích cách thiết lập hàm dự đoán cảm xúc của tiêu đề tin tức bằng mô hình ngôn ngữ Llama-3

Hàm này nhận ba đối số:

1. **test:** Một DataFrame của Pandas chứa các tiêu đề tin tức cần dự đoán cảm xúc.
2. **model:** Mô hình ngôn ngữ Llama-3 đã được huấn luyện trước.
3. **tokenizer:** Bộ tokenizer dành cho mô hình Llama-3.

Cách thức hoạt động của hàm:

- Đối với mỗi tiêu đề tin tức trong DataFrame test:
  - Tạo một prompt cho mô hình ngôn ngữ, yêu cầu nó phân tích cảm xúc của tiêu đề và trả về nhãn cảm xúc tương ứng.
  - Sử dụng hàm `pipeline()` từ thư viện Hugging Face Transformers để tạo văn bản từ mô hình ngôn ngữ dựa trên prompt.
  - Trích xuất nhãn cảm xúc dự đoán từ văn bản được tạo ra.
  - Thêm nhãn cảm xúc dự đoán vào danh sách `y_pred`.
- Trả về danh sách `y_pred` chứa các nhãn cảm xúc dự đoán.

Chi tiết về pipeline():

- `pipeline()` là hàm tạo văn bản từ mô hình ngôn ngữ của thư viện Hugging Face Transformers.
  - Đối số `task` chỉ định nhiệm vụ là tạo văn bản (text generation).
  - Các đối số `model` và `tokenizer` chỉ định mô hình Llama-3 đã huấn luyện và bộ tokenizer tương ứng.
  - `max_new_tokens` chỉ định số lượng token mới tối đa sẽ được tạo ra.
  - `temperature` kiểm soát mức độ ngẫu nhiên của văn bản tạo ra.
    - Nhiệt độ thấp sẽ tạo ra văn bản có tính tiên đoán cao hơn, nhiệt độ cao sẽ tạo ra văn bản sáng tạo hơn và có thể bất ngờ hơn.

Điều kiện kiểm tra trong câu lệnh if:

- Câu lệnh if kiểm tra xem văn bản tạo ra có chứa từ `"positive"` hay không. Nếu có, nhãn cảm xúc dự đoán là `"positive"`.
- Nếu không, tiếp tục kiểm tra xem văn bản có chứa từ `"negative"` không. Nếu có, nhãn cảm xúc dự đoán là `"negative"`.
- Nếu không, kiểm tra xem văn bản có chứa từ `"neutral"` không. Nếu có, nhãn cảm xúc dự đoán là `"neutral"`.

Tóm lại, hàm này giúp sử dụng mô hình Llama-3 để dự đoán cảm xúc của các tiêu đề tin tức bằng cách phân tích văn bản tạo ra từ mô hình, sau đó xác định cảm xúc tương ứng dựa trên nội dung của văn bản đó.

1.0.6 Chuẩn bị cho quá trình Fine-tune mô hình

Trong phần này, chúng ta chuẩn bị mọi thứ để fine-tune mô hình bằng cách cấu hình và khởi tạo một Simple Fine-tuning Trainer (SFTTrainer) để huấn luyện mô hình ngôn ngữ lớn (LLM) sử dụng phương pháp Parameter-Efficient Fine-Tuning (PEFT). Phương pháp này tiết kiệm thời gian vì chỉ điều chỉnh một số lượng tham số nhất định so với toàn bộ kích thước của mô hình, từ đó giúp giảm đáng kể chi phí tính toán và lưu trữ.

Cấu hình PEFT:

- `PEFTConfig`:
  - Đối tượng `peft_config` chỉ định các tham số cho quá trình fine-tuning PEFT. Dưới đây là một số tham số quan trọng:
    - `lora_alpha`: Tốc độ học cho các ma trận cập nhật của LoRA.
    - `lora_dropout`: Xác suất dropout cho các ma trận cập nhật của LoRA.
    - `r`: Hạng của các ma trận cập nhật LoRA.
    - `bias`: Loại bias được sử dụng. Các giá trị có thể là `none`, `additive`, và `learned`.
    - `task_type`: Loại nhiệm vụ mà mô hình đang được huấn luyện, các giá trị có thể là `CAUSAL_LM` (mô hình ngôn ngữ nhân quả) và `MASKED_LM` (mô hình ngôn ngữ mặt nạ).

Các tham số `TrainingArguments`:

- `TrainingArguments`: Đối tượng `training_arguments` chỉ định các tham số cho quá trình huấn luyện mô hình. Một số tham số quan trọng:
  - `output_dir`: Thư mục để lưu nhật ký huấn luyện và các điểm kiểm tra.
  - `num_train_epochs`: Số lượng epoch để huấn luyện mô hình.
  - `per_device_train_batch_size`: Số lượng mẫu trong mỗi batch trên mỗi thiết bị.
  - `gradient_accumulation_steps`: Số batch cần tích lũy gradient trước khi cập nhật tham số của mô hình.
  - `optim`: Optimizer để sử dụng khi huấn luyện mô hình.
  - `save_steps`: Số bước sau đó sẽ lưu một điểm kiểm tra.
  - `logging_steps`: Số bước sau đó sẽ ghi nhật ký các số liệu huấn luyện.
  - `learning_rate`: Tốc độ học cho optimizer.
  - `weight_decay`: Tham số weight decay cho optimizer.
  - `fp16`: Sử dụng độ chính xác 16-bit floating-point hay không.
  - `bf16`: Sử dụng độ chính xác BFloat16 hay không.
  - `max_grad_norm`: Chuẩn gradient tối đa.
  - `max_steps`: Số bước tối đa để huấn luyện mô hình.
  - `warmup_ratio`: Tỉ lệ số bước huấn luyện được sử dụng để warm-up tốc độ học.
  - `group_by_length`: Có nhóm các mẫu huấn luyện theo độ dài hay không.
  - `lr_scheduler_type`: Loại bộ điều chỉnh tốc độ học.
  - `report_to`: Các công cụ để báo cáo các số liệu huấn luyện.
  - `evaluation_strategy`: Chiến lược đánh giá mô hình trong quá trình huấn luyện.

SFTTrainer:

- `SFTTrainer` là một lớp custom từ thư viện TRL, được sử dụng để huấn luyện các mô hình ngôn ngữ lớn (LLM) bằng phương pháp PEFT.
- Đối tượng `SFTTrainer` được khởi tạo với các đối số sau:
  - `model`: Mô hình cần huấn luyện.
  - `train_dataset`: Bộ dữ liệu huấn luyện.
  - `eval_dataset`: Bộ dữ liệu đánh giá.
  - `peft_config`: Cấu hình PEFT.
  - `dataset_text_field`: Tên của trường văn bản trong bộ dữ liệu.
  - `tokenizer`: Tokenizer được sử dụng.
  - `args`: Các tham số huấn luyện.
  - `packing`: Có gói các mẫu huấn luyện hay không.
  - `max_seq_length`: Độ dài chuỗi tối đa.

Sau khi khởi tạo đối tượng `SFTTrainer`, chúng ta có thể gọi phương thức `train()` để bắt đầu quá trình fine-tuning mô hình.
Tổng kết:
Phần này thiết lập các cấu hình cần thiết và khởi tạo `SFTTrainer` để huấn luyện mô hình ngôn ngữ lớn bằng cách sử dụng phương pháp PEFT, giúp giảm thời gian và chi phí trong quá trình fine-tuning, đồng thời giảm thiểu vấn đề "quên lãng thảm khốc"thường gặp khi điều chỉnh toàn bộ các tham số của mô hình lớn.

1.0.7 Kiểm thử mô hình

Phần kiểm thử ở đây tiến hành dự đoán các nhãn cảm xúc cho tập kiểm thử bằng hàm `predict()` và đánh giá hiệu năng của mô hình bằng hàm `evaluate()`. Kết quả dự kiến là độ chính xác tổng thể hơn 0.8, cùng với độ chính xác, độ nhạy (precision) và khả năng thu hồi (recall) cao cho các nhãn cảm xúc riêng lẻ. Dự đoán cho nhãn `neutral` (trung lập) vẫn có thể cải thiện, nhưng vẫn ấn tượng khi chỉ với một lượng dữ liệu nhỏ và fine-tuning, ta có thể đạt được kết quả tốt như vậy.
Kết quả:

```
y_pred = predict(test, model, tokenizer)
evaluate(y_true, y_pred)
```

- Độ chính xác tổng thể: 0.873
- Độ chính xác cho nhãn 0 (tích cực): 0.937
- Độ chính xác cho nhãn 1 (tiêu cực): 0.847
- Độ chính xác cho nhãn 2 (trung lập): 0.837

Báo cáo phân loại (Classification Report):

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       300
           1       0.80      0.85      0.82       300
           2       0.87      0.84      0.86       300

    accuracy                           0.87       900
   macro avg       0.88      0.87      0.87       900
weighted avg       0.88      0.87      0.87       900
```

Ma trận nhầm lẫn (Confusion Matrix):

```
[[281  18   1]
 [ 11 254  35]
 [  3  46 251]]
```

Tạo DataFrame cho đánh giá lỗi:

```
evaluation = pd.DataFrame({'text': X_test["text"],
                           'y_true': y_true,
                           'y_pred': y_pred},
                          )
evaluation.to_csv("test_predictions.csv", index=False)
```

File CSV chứa dữ liệu dự đoán và nhãn thực tế sẽ giúp hiểu rõ hơn về các lỗi mà mô hình fine-tuned gặp phải, đồng thời cung cấp thông tin để cải thiện các prompt trong tương lai.

2 Coding

2.0.1 Tiền xử lý dữ liệu cho phân tích cảm xúc

Đoạn code trên thực hiện các bước tiền xử lý dữ liệu cho một bài toán phân tích cảm xúc của các tiêu đề tin tức tài chính. Nó bao gồm việc tải dữ liệu, chia dữ liệu thành các tập huấn luyện, kiểm tra và đánh giá, và chuẩn bị các dữ liệu đầu vào cho mô hình ngôn ngữ. Sau đây là giải thích chi tiết từng phần:

1. **Đọc dữ liệu từ file CSV**:

```
filename = "../input/sentiment-analysis-for-financial-news/all-data.csv"

df = pd.read_csv(filename,
                 names=["sentiment", "text"],
                 encoding="utf-8", encoding_errors="replace")
```

- `filename`: Đường dẫn tới tệp CSV chứa dữ liệu phân tích cảm xúc. - $pd.read_csv$: Hàm đọc dữ liệu từ tệp CSV. - `names=["sentiment", "text"]`: Đặt tên cho các cột, gồm cột `sentiment` (nhãn cảm xúc) và `text` (nội dung tin tức). - `encoding="utf-8", encoding_errors="replace"`: Đọc file với mã hóa UTF-8 và thay thế các ký tự lỗi.

2. **Chia dữ liệu thành tập huấn luyện và kiểm tra theo nhãn cảm xúc**:

```
X_train = list()
X_test = list()
for sentiment in ["positive", "neutral", "negative"]:
    train, test  = train_test_split(df[df.sentiment==sentiment],
                                    train_size=300,
                                    test_size=300,
                                    random_state=42)
    X_train.append(train)
    X_test.append(test)
```

- `X_train` và `X_test`: Danh sách để lưu trữ các tập huấn luyện và kiểm tra cho mỗi nhãn cảm xúc. - Vòng lặp `for` thực hiện việc lọc các dòng có cảm xúc tương ứng (tích cực, trung lập, tiêu cực). - `train_test_split`: Chia dữ liệu cho mỗi nhãn cảm xúc thành 2 tập con: tập huấn luyện (`train_size=300`) và tập kiểm tra (`test_size=300`), với cùng một hạt giống ngẫu nhiên (`random_state=42`) để đảm bảo tái lập kết quả.

3. **Kết hợp và xáo trộn các dữ liệu huấn luyện và kiểm tra**:

```
X_train = pd.concat(X_train).sample(frac=1, random_state=10)
X_test = pd.concat(X_test)
```

- `pd.concat`: Ghép nối các tập huấn luyện và kiểm tra từ danh sách thành một DataFrame duy nhất. - `sample(frac=1, random_state=10)`: Xáo trộn các mẫu trong tập huấn luyện để tránh sự lệch lạc trong thứ tự dữ liệu.

4. **Tạo tập đánh giá từ các mẫu chưa được sử dụng trong tập huấn luyện và kiểm tra**:

```
eval_idx = [idx for idx in df.index if idx not in list(X_train.index) + list(X_te
X_eval = df[df.index.isin(eval_idx)]
X_eval = (X_eval
          .groupby('sentiment', group_keys=False)
          .apply(lambda x: x.sample(n=50, random_state=10, replace=True)))
X_train = X_train.reset_index(drop=True)
```

- `eval_idx`: Danh sách các chỉ số (index) của các mẫu không thuộc tập huấn luyện và kiểm tra. - `X_eval`: Tạo tập đánh giá (`eval`) bằng cách lấy các mẫu không thuộc các tập trước và sau đó lấy ngẫu nhiên 50 mẫu cho mỗi nhãn cảm xúc.

5. **Hàm tạo prompt để huấn luyện và kiểm tra mô hình**:

```
def generate_prompt(data_point):
    return f"""
            Analyze the sentiment of the news headline enclosed in square bracket
            determine if it is positive, neutral, or negative, and return the ans
            the corresponding sentiment label "positive" or "neutral" or "negativ

            [{data_point["text"]}] = {data_point["sentiment"]}
```

```
            """.strip()

    def generate_test_prompt(data_point):
        return f"""
                Analyze the sentiment of the news headline enclosed in square bracket
                determine if it is positive, neutral, or negative, and return the ans
                the corresponding sentiment label "positive" or "neutral" or "negativ

                [{data_point["text"]}] = """.strip()
```

- `generate_prompt`: Tạo prompt huấn luyện với tin tức và nhãn cảm xúc tương ứng (dùng trong tập huấn luyện và tập đánh giá). - `generate_test_prompt`: Tạo prompt kiểm tra mà không có nhãn cảm xúc (dùng trong tập kiểm tra).

6. **Áp dụng các hàm prompt và chuyển đổi thành DataFrame**:

```
X_train = pd.DataFrame(X_train.apply(generate_prompt, axis=1),
                       columns=["text"])
X_eval = pd.DataFrame(X_eval.apply(generate_prompt, axis=1),
                      columns=["text"])

y_true = X_test.sentiment
X_test = pd.DataFrame(X_test.apply(generate_test_prompt, axis=1), columns=["text"
```

- Tạo các DataFrame cho tập huấn luyện và tập đánh giá từ các mẫu đã tạo prompt. - `y_true`: Lấy nhãn thực tế của tập kiểm tra. - Tạo DataFrame `X_test` chứa các prompt kiểm tra không có nhãn.

7. **Chuyển dữ liệu huấn luyện thành định dạng Dataset**:

```
train_data = Dataset.from_pandas(X_train)
```

- `Dataset.from_pandas`: Chuyển đổi DataFrame `X_train` thành định dạng Dataset để sử dụng trong quá trình huấn luyện mô hình.

Tổng quan: Đoạn code này thực hiện các bước tiền xử lý cho bài toán phân loại cảm xúc, bao gồm việc chia dữ liệu, tạo prompt cho từng mẫu dữ liệu và chuẩn bị đầu vào cho quá trình huấn luyện mô hình ngôn ngữ.

2.0.2 Khởi Tạo Mô Hình và Tokenizer

```
model_name = "../input/llama-3/transformers/8b-chat-hf/1"

compute_dtype = getattr(torch, "float16")

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
```

```
    bnb_4bit_use_double_quant=False,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=compute_dtype,
)

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map=device,
    torch_dtype=compute_dtype,
    quantization_config=bnb_config,
)

model.config.use_cache = False
model.config.pretraining_tp = 1

max_seq_length = 512 #2048
tokenizer = AutoTokenizer.from_pretrained(model_name, max_seq_length=max_seq_le
tokenizer.pad_token_id = tokenizer.eos_token_id
```

Giải thích mã:

- `model_name`: Đường dẫn đến mô hình đã được huấn luyện sẵn.
- `compute_dtype`: Định dạng dữ liệu cho phép sử dụng số thực 16 bit thông qua hàm `getattr` để lấy kiểu dữ liệu từ thư viện `torch`.
- `bnb_config`: Cấu hình cho việc nén mô hình với `BitsAndBytesConfig`, bao gồm các tham số:
  - `load_in_4bit`: Tải mô hình ở chế độ 4 bit.
  - `bnb_4bit_use_double_quant`: Không sử dụng quantization kép.
  - `bnb_4bit_quant_type`: Loại quantization được sử dụng là `nf4`.
  - `bnb_4bit_compute_dtype`: Định dạng dữ liệu cho tính toán được xác định bởi `compute_dtype`.
- `model`: Tạo một mô hình ngôn ngữ với lớp `AutoModelForCausalLM` từ mô hình đã được huấn luyện sẵn, sử dụng các tham số sau:
  - `device_map`: Bản đồ thiết bị để xác định nơi mô hình sẽ chạy.
  - `torch_dtype`: Định dạng dữ liệu cho mô hình.
  - `quantization_config`: Cấu hình quantization cho mô hình.
- `model.config.use_cache`: Tắt tính năng lưu cache của mô hình.
- `model.config.pretraining_tp`: Thiết lập số lượng tp (tensor parallelism) trong huấn luyện mô hình.
- `max_seq_length`: Độ dài tối đa của chuỗi đầu vào, được thiết lập là 512.
- `tokenizer`: Khởi tạo tokenizer từ mô hình đã được huấn luyện sẵn, với độ dài tối đa của chuỗi đầu vào.
- `tokenizer.pad_token_id`: Thiết lập ID của token padding bằng với ID của token kết thúc chuỗi (eos token).

2.0.3 Dự Đoán Nhãn Cảm Xúc Với Mô Hình

```
def predict(test, model, tokenizer):
    y_pred = []
    for i in tqdm(range(len(X_test))):
        prompt = X_test.iloc[i]["text"]
        pipe = pipeline(task="text-generation",
                        model=model,
                        tokenizer=tokenizer,
                        max_new_tokens = 1,
                        temperature = 0.0,
                       )
        result = pipe(prompt)
        answer = result[0]['generated_text'].split("=")[-1]
        if "positive" in answer:
            y_pred.append("positive")
        elif "negative" in answer:
            y_pred.append("negative")
        elif "neutral" in answer:
            y_pred.append("neutral")
        else:
            y_pred.append("none")
    return y_pred
```

Giải thích mã:

1. **Khởi tạo danh sách để lưu kết quả dự đoán**

   ```
   y_pred = []
   ```

   - `y_pred`: Danh sách này sẽ lưu trữ các nhãn cảm xúc dự đoán cho từng mẫu trong tập kiểm tra.

2. **Vòng lặp để dự đoán cho từng mẫu trong tập kiểm tra**

   ```
   for i in tqdm(range(len(X_test))):
   ```

   - Sử dụng `tqdm` để hiển thị thanh tiến trình trong khi lặp qua từng chỉ số (index) của tập kiểm tra `X_test`.

3. **Lấy prompt (câu đầu vào) từ tập kiểm tra**

   ```
   prompt = X_test.iloc[i]["text"]
   ```

   - `prompt`: Lấy chuỗi văn bản từ dòng thứ `i` trong tập kiểm tra, đây là đầu vào mà mô hình sẽ phân tích.

4. **Khởi tạo pipeline cho tác vụ sinh văn bản**

```
pipe = pipeline(task="text-generation",
                model=model,
                tokenizer=tokenizer,
                max_new_tokens = 1,
                temperature = 0.0,
               )
```

- `pipeline`: Hàm này khởi tạo một pipeline cho tác vụ sinh văn bản, cụ thể là để phân tích cảm xúc trong văn bản.

  - `task="text-generation"`: Định nghĩa tác vụ là sinh văn bản.
  - `model=model`: Sử dụng mô hình đã được tải trước đó.
  - `tokenizer=tokenizer`: Sử dụng tokenizer đã được tải.
  - `max_new_tokens=1`: Giới hạn số lượng token mới được sinh ra là 1, nghĩa là mô hình chỉ tạo ra một token phản hồi.
  - `temperature=0.0`: Đặt nhiệt độ là 0, điều này có nghĩa là mô hình sẽ chọn câu trả lời có xác suất cao nhất (không có độ ngẫu nhiên).

5. **Tạo kết quả từ prompt**

```
result = pipe(prompt)
```

- `result`: Kết quả đầu ra từ pipeline sau khi truyền vào `prompt`.

6. **Lấy và xử lý câu trả lời từ kết quả sinh ra**

```
answer = result[0]['generated_text'].split("=")[-1]
```

- `answer`: Lấy văn bản sinh ra từ mô hình và tách nó theo dấu -"để lấy phần sau dấu -". Giả định rằng mô hình đã được huấn luyện để trả về kết quả theo định dạng như vậy.

7. **Xác định nhãn cảm xúc từ câu trả lời**

```
if "positive" in answer:
    y_pred.append("positive")
elif "negative" in answer:
    y_pred.append("negative")
elif "neutral" in answer:
    y_pred.append("neutral")
else:
    y_pred.append("none")
```

- Kiểm tra giá trị trong biến `answer`:

  - Nếu chứa `"positive"`, thêm `"positive"` vào danh sách `y_pred`.
  - Nếu chứa `"negative"`, thêm `"negative"` vào danh sách `y_pred`.

- Nếu chứa `"neutral"`, thêm `"neutral"` vào danh sách `y_pred`.
- Nếu không thuộc các nhãn trên, thêm `"none"` vào danh sách `y_pred`.

8. **Trả về danh sách dự đoán**

```
return y_pred
```

- Hàm `predict` trả về danh sách `y_pred`, chứa nhãn cảm xúc dự đoán cho từng mẫu trong tập kiểm tra.

Tóm tắt: Hàm `predict` được thiết kế để dự đoán nhãn cảm xúc cho các tiêu đề tin tức trong tập kiểm tra bằng cách sử dụng mô hình sinh văn bản. Nó tạo ra văn bản dự đoán từ mô hình cho từng đầu vào và xác định nhãn cảm xúc dựa trên kết quả đầu ra của mô hình. Hàm này rất hữu ích trong các tác vụ phân tích cảm xúc trong văn bản, đặc biệt là trong lĩnh vực tài chính hoặc tin tức.

2.0.4 Fine Tune mô hình

1. **Nhập khẩu các thư viện cần thiết**

```
from sklearn.metrics import (accuracy_score,
                             recall_score,
                             precision_score,
                             f1_score)
from transformers import EarlyStoppingCallback, IntervalStrategy
```

- Các hàm từ `sklearn.metrics` được sử dụng để tính toán độ chính xác (accuracy), độ nhạy (recall), độ chính xác dương tính (precision) và điểm F1. - `EarlyStoppingCallback` được sử dụng để dừng quá trình huấn luyện khi mô hình không cải thiện trong một số epoch liên tiếp.

2. **Định nghĩa hàm** `compute_metrics`

```
def compute_metrics(p):
    pred, labels = p
    pred = np.argmax(pred, axis=1)
    accuracy = accuracy_score(y_true=labels, y_pred=pred)
    recall = recall_score(y_true=labels, y_pred=pred)
    precision = precision_score(y_true=labels, y_pred=pred)
    f1 = f1_score(y_true=labels, y_pred=pred)
    return {"accuracy": accuracy, "precision": precision, "recall": recall, "f1":
```

- **Đầu vào**: `p` là một tuple chứa hai thành phần: `pred` (dự đoán của mô hình) và `labels` (nhãn thực tế). - `np.argmax(pred, axis=1)`: Lấy chỉ số của giá trị lớn nhất trong mỗi dự đoán (tức là chọn lớp dự đoán). - Tính toán và trả về các metric như độ chính xác, độ nhạy, độ chính xác dương tính và điểm F1 dưới dạng một từ điển.

3. **Cấu hình đường dẫn xuất dữ liệu**

```
output_dir="trained_weigths"
```

- `output_dir`: Đường dẫn nơi các trọng số đã được huấn luyện sẽ được lưu.

4. **Cấu hình PEFT với LoRA**

```
peft_config = LoraConfig(
    lora_alpha=16,
    lora_dropout=0,
    r=64,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj",
                    "gate_proj", "up_proj", "down_proj",],
)
```

- `LoraConfig`: Cấu hình cho việc sử dụng Low-Rank Adaptation (LoRA) để tối ưu hóa mô hình. - `lora_alpha`: Tham số điều chỉnh của LoRA. - `lora_dropout`: Tỉ lệ dropout cho LoRA. - `r`: Số lượng hạng trong việc phân tích hạng thấp. - `task_type`: Loại tác vụ, ở đây là `CAUSAL_LM` (Language Model với tác vụ nguyên nhân). - `target_modules`: Các module mục tiêu trong mô hình mà LoRA sẽ áp dụng.

5. **Cấu hình các tham số huấn luyện**

```
training_arguments = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=5,
    per_device_train_batch_size=1,
    gradient_accumulation_steps=8,
    gradient_checkpointing=True,
    optim="paged_adamw_32bit",
    save_steps=0,
    logging_steps=25,
    learning_rate=2e-4,
    weight_decay=0.001,
    fp16=True,
    bf16=False,
    max_grad_norm=0.3,
    max_steps=-1,
    warmup_ratio=0.03,
    group_by_length=False,
    lr_scheduler_type="cosine",
    report_to="tensorboard",
)
```

- `TrainingArguments`: Các tham số cấu hình cho quá trình huấn luyện. - `num_train_epochs`: Số lượng epoch huấn luyện. - `per_device_train_batch_size`: Kích thước batch cho mỗi thiết

bị trong quá trình huấn luyện. - `gradient_accumulation_steps`: Số bước gradient trước khi thực hiện cập nhật trọng số. - `gradient_checkpointing`: Sử dụng checkpointing để tiết kiệm bộ nhớ. - `optim`: Phương pháp tối ưu hóa sử dụng. - `logging_steps`: Số bước để ghi log. - `learning_rate`: Tốc độ học. - `weight_decay`: Hệ số suy giảm trọng số. - `fp16`: Sử dụng precision 16-bit. - `max_grad_norm`: Giới hạn độ lớn gradient. - `warmup_ratio`: Tỉ lệ warmup. - `lr_scheduler_type`: Loại bộ lập lịch tốc độ học. - `report_to`: Nơi báo cáo các metric (ở đây là TensorBoard).

6. **Khởi tạo trainer**

```
trainer = SFTTrainer(
    model=model,
    args=training_arguments,
    train_dataset=train_data,
    peft_config=peft_config,
    dataset_text_field="text",
    tokenizer=tokenizer,
    max_seq_length=max_seq_length,
    packing=False,
    dataset_kwargs={
        "add_special_tokens": False,
        "append_concat_token": False,
    },
)
```

- `SFTTrainer`: Khởi tạo một trainer cho mô hình. - `model`: Mô hình sẽ được huấn luyện. - `args`: Các tham số huấn luyện đã được cấu hình. - `train_dataset`: Tập dữ liệu huấn luyện. - `peft_config`: Cấu hình LoRA. - `dataset_text_field`: Trường văn bản trong tập dữ liệu. - `tokenizer`: Tokenizer sử dụng cho mô hình. - `max_seq_length`: Chiều dài tối đa của chuỗi đầu vào. - `packing`: Nếu được đặt là True, sẽ kết hợp các chuỗi ngắn hơn để sử dụng bộ nhớ hiệu quả hơn. - `dataset_kwargs`: Các tùy chọn bổ sung cho tập dữ liệu.

Tóm tắt: Đoạn mã thiết lập một quy trình huấn luyện cho mô hình ngôn ngữ với các cấu hình cụ thể. Nó bao gồm việc tính toán các chỉ số đánh giá, cấu hình PEFT với LoRA, định nghĩa các tham số huấn luyện, và khởi tạo một `trainer` để thực hiện quá trình huấn luyện. Các chỉ số như accuracy, precision, recall và F1-score sẽ giúp đánh giá hiệu suất của mô hình trong việc dự đoán nhãn cảm xúc cho dữ liệu đầu vào.

3 Tài liệu tham khảo

Tài liệu

[1] 2017. *DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction*. Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. In Proceedings of the 26th International Joint Conference on Artificial Intelligence. AAAI Press, pp. 1725–1731.