Phương pháp, Kỹ thuật và Mô hình sử dụng

Phương pháp

Sử dụng **LLM** để phân loại với các task khác nhau và các nhãn khác nhau, sau đó sử dụng kỹ thuật **Ensemble Voting** và cuối cùng là **Postprocessing**.

Submission 1

Train **3 mô hình dự đoán nhãn nhị phân (binary)**, thay vì dự đoán 3 nhãn cùng lúc. Việc này giúp mô hình có ít lựa chọn hơn → dễ đạt độ chính xác cao hơn:

- **no / yes** → Phân loại response có hay không có ảo giác so với context.
  → Kết quả: `predict_label_m1`
- **known / unknown** → Phân loại tất cả thông tin trong response có được suy ra từ context (known) hay có ít nhất 1 thông tin không được suy ra (unknown).
  → Kết quả: `predict_label_m2`
- **support / contradictory** → Phân loại tất cả thông tin trong response có được suy ra từ context (support) hay có ít nhất 1 thông tin bóp méo, sai lệch hoàn toàn (contradictory).
  → Kết quả: `predict_label_m3`

Submission 2

Train **1 mô hình dự đoán 3 nhãn**: `no`, `intrinsic`, `extrinsic`.
→ Kết quả: `predict_label_m4`

Submission 3 — Ensemble

Chi tiết phương pháp ensemble, vui lòng xem ở đường dẫn Source-code-github/README.md

Đội sử dụng logic ensemble sau để chốt kết quả `predict_label` cuối cùng (0.8400 F1):

```
def final_label(row):
    if row["predict_label_m1"] == "no" and row["predict_label_m2"] == "known" and row["predict_label_m3"] == "supported":
        return "no"
    if row["predict_label_m1"] == "yes" and row["predict_label_m2"] == "unknown" and row["predict_label_m3"] == "supported":
        return "intrinsic"
    if row["predict_label_m1"] == "yes" and row["predict_label_m2"] == "unknown" and row["predict_label_m3"] == "contradictory":
        return "extrinsic"
    return row["predict_label_m4"]
```

Kết quả của `predict_label` sẽ phụ thuộc vào việc 3 model trên submission 1 có đồng thuận với nhau hay không.
Nếu không thì sẽ phụ thuộc vào model từ submission 2.

Submission 4 — Compare & Postprocessing

Chi tiết vui lòng xem ở đường dẫn Source-code-github/all-in-one-inference.ipynb

- So sánh kết quả giữa Submission 2 và Submission 3 → tạo file `compare.csv`.
- Phân tích lỗi cho thấy mô hình ở Submission 3 thường dự đoán nhãn “extrinsic” thành “no” trong trường hợp văn bản reaffirm lại thông tin từ prompt nhưng không có trong context.
- Trong khi đó, Submission 2 lại gán đúng “extrinsic” cho các trường hợp này.

Quy tắc chỉnh sửa:
Chỉ trong `compare.csv` (khoảng 52 dòng), nếu Submission 2 dự đoán là `extrinsic` và Submission 3 dự đoán là `no` → chỉnh lại thành `extrinsic`.

→ Kết quả tăng nhẹ lên **0.8401 F1** (5/2000 dòng được thay đổi).

Kỹ thuật

Chuẩn bị Prompt

Trong các mô hình instruction-tuned, đầu vào gồm:

- **System prompt**: Định nghĩa vai trò, nhiệm vụ, cách ứng xử. Ở đây, system prompt quy định mô hình phải phân tích Context và Response, sau đó đưa ra nhãn cuối cùng.
- **User prompt**: Chứa context + response + yêu cầu phân loại. Đây là phần mô hình “thấy” để suy luận.
- **Assistant output**: Nhãn ground-truth từ dữ liệu huấn luyện.

Cách này giúp mô hình quen với định dạng hội thoại và dễ khớp với pipeline inference.

Lựa chọn Model

- Sử dụng **Qwen 3–4B** (dung lượng vừa phải, cân bằng hiệu năng và chi phí).

- Dùng **quantization 4-bit (nf4)** để giảm tải bộ nhớ GPU, huấn luyện trên phần cứng hạn chế.

Tiền xử lý dữ liệu (Preprocess)

- Chuẩn hóa dữ liệu về 3 trường: `context`, `response`, `label`.
- Tạo chuỗi hội thoại: system + user prompt → nhãn.
- Dùng chat template mô phỏng định dạng hội thoại.
- Che toàn bộ token input bằng `-100`, chỉ tính loss cho phần nhãn.
- Pad/truncate batch về độ dài chung → đồng bộ `input_ids`, `labels`, `attention_mask`.

Lý do không dùng prompt gốc:

- Prompt raw thường lỗi chính tả, format không chuẩn, gây nhiễu → khó hội tụ.
- Tốn GPU/VRAM vì prompt dài.
- Mô hình sinh output sai format, dài dòng.

Giải pháp:
Thiết kế input đơn giản, rõ ràng → mô hình học tốt hơn, tiết kiệm tài nguyên.

Hậu xử lý dữ liệu (Postprocess)

Sau khi mô hình sinh output:

1. Xoá tag đặc biệt (`<...>`).
2. Trim khoảng trắng.
3. Nếu output không khớp từ điển nhãn → fallback lấy từ cuối cùng hoặc map sang nhãn gần nhất.

Kỹ thuật huấn luyện

- Khi huấn luyện, ta chỉ tính loss ở phần nhãn assistant.
- **LoRA**: Fine-tuning tham số nhẹ trên attention modules (q, k, v, o, gate, up, down).
- **Quantization-aware**: Huấn luyện trên mô hình 4-bit.
- **Gradient checkpointing**: Giảm memory footprint.
- **Optimizer**: `paged_adamw_8bit` + scheduler cosine + warmup nhỏ.

Đánh giá

Pipeline đánh giá nhiều bước:

1. **Logits → Predictions**
    - Argmax trên logits.
    - Nếu logits là tuple → lấy phần tử đầu tiên.
2. **Xử lý chuỗi dự đoán**
    - Tìm vị trí bắt đầu nhãn thật trong `labels`.
    - Cắt chuỗi từ đó, bỏ `-100`, pad, token đặc biệt (EOS, BOS).
    - Dừng khi gặp `eos_token_id`.
3. **Decode → Text**
    - Token → nhãn text: `no`, `intrinsic`, `extrinsic`.
4. **Mapping Text → Label ID**
    - Chuẩn hóa nhãn dự đoán & thật thành `int_preds`, `int_labels`.

Metrics: Accuracy, Precision, Recall, F1. Kết thúc training, load model tốt nhất với F1 cao nhất.

Quản lý mô hình với Hugging Face

- Sử dụng `Hugging Face Trainer` với `push_to_hub=True` để commit mô hình và log lên Hub.
- Lưu trữ trên Hub giúp quản lý lịch sử huấn luyện, rollback, so sánh phiên bản.

Mô hình

Qwen/Qwen3-4B-Instruct-2507
Link: https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507