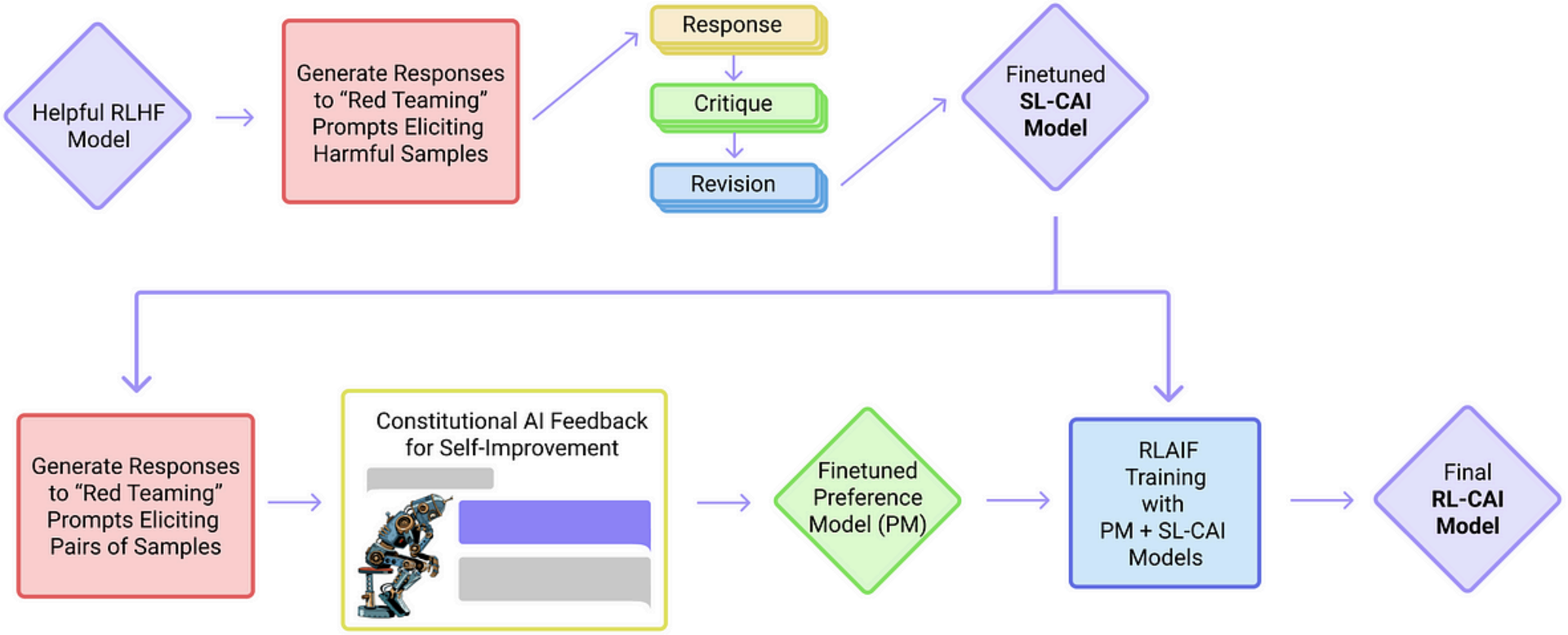
CONSTITUTIONAL AI

By: Bui Tan Nhat

GIỚI THIỆU:

Constitutional AI

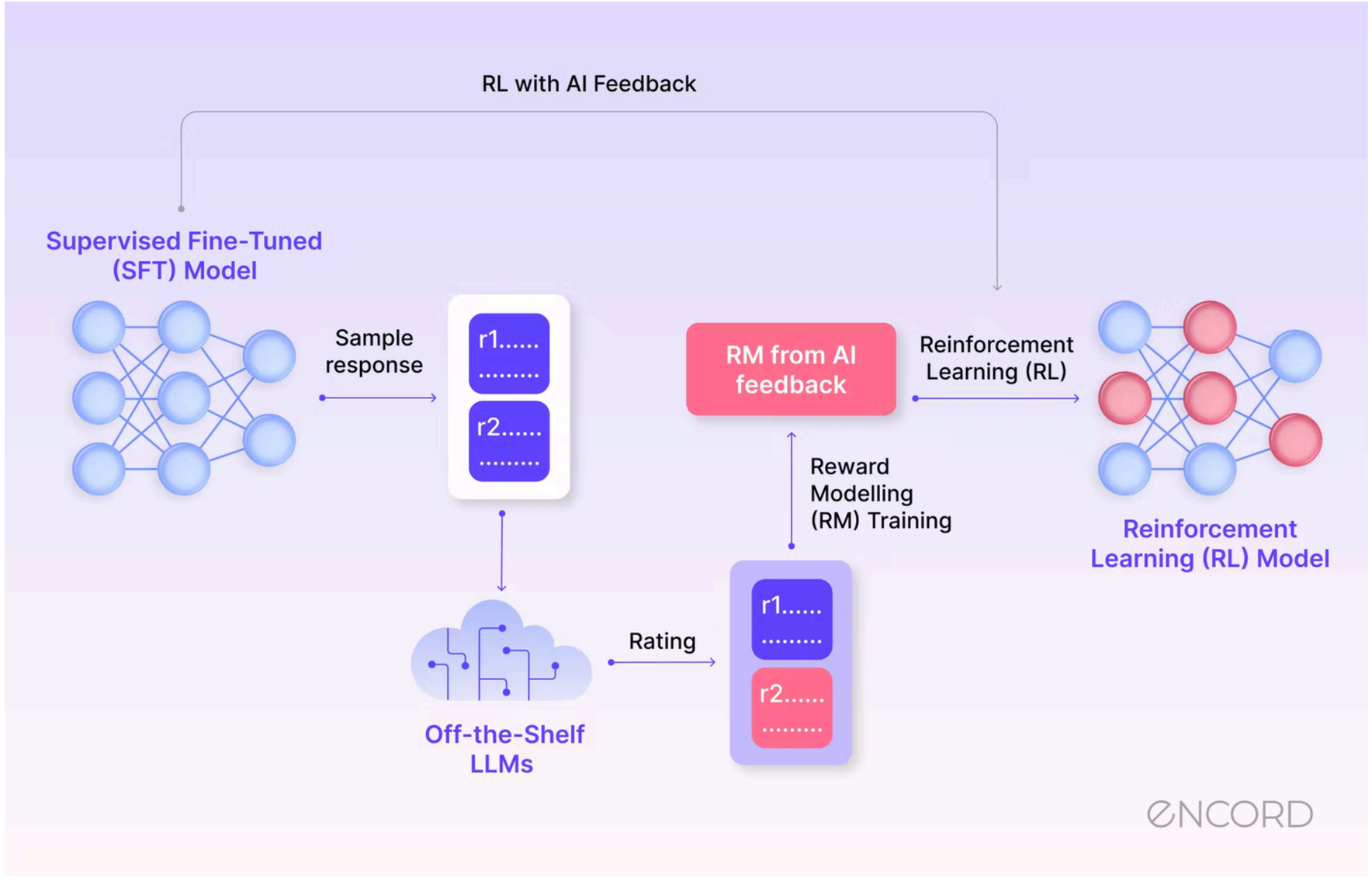


Constitutional AI

Các nguyên tắc định hướng:

- **Harmless – Không gây hại**: Mô hình không được cung cấp nội dung **nguy hiểm, độc hại hoặc bất hợp pháp**. Nó phải tránh hướng dẫn những hành vi có thể gây hại cho người dùng hoặc xã hội.
- **Helpful – Hữu ích**: Mô hình cần đưa ra câu trả lời **có ích và thông tin**, đáp ứng đúng nhu cầu câu hỏi của người dùng một cách hữu dụng.
- **Honest – Trung thực**: Mô hình phải **thành thật**, không bịa đặt thông tin hoặc đưa ra các khẳng định sai sự thật. Nếu không chắc chắn, mô hình nên thể hiện sự không chắc chắn thay vì khẳng định tuyệt đối.
- **Respectful – Tôn trọng (Lịch sự)**: Mô hình phải **tôn trọng người dùng và các chủ thể khác**, tránh ngôn ngữ xúc phạm hoặc thiếu lịch sự, và duy trì thái độ đúng mực.

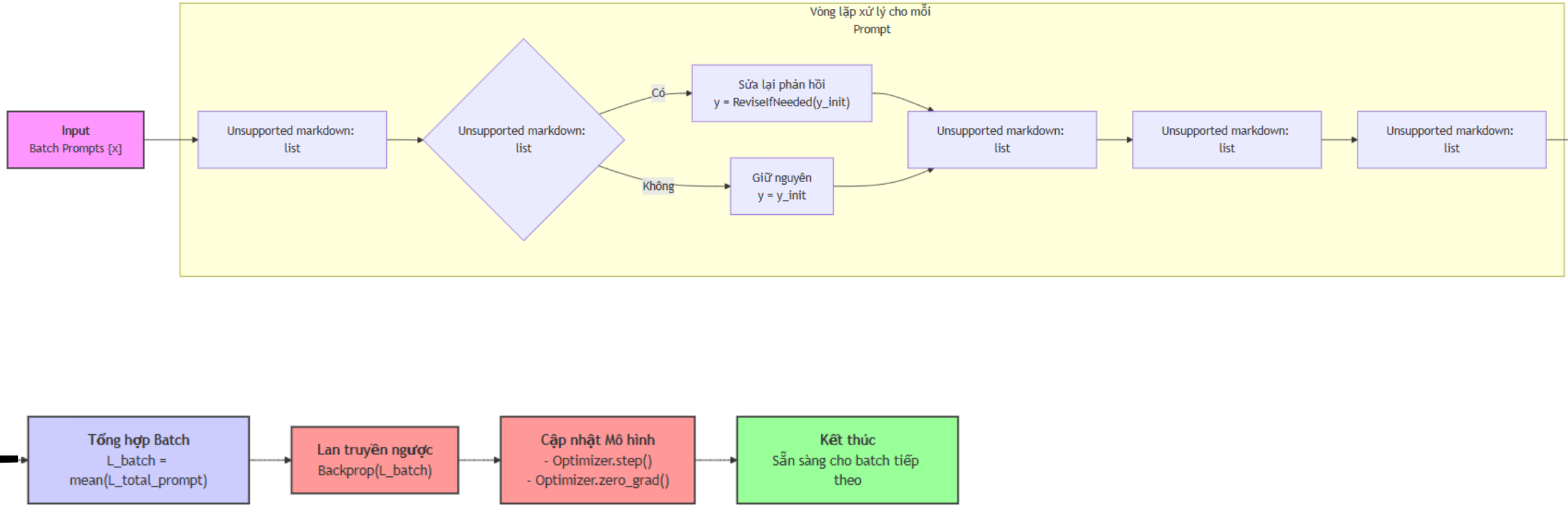
Phương Pháp RLAIF


RL with AI Feedback
Supervised Fine-Tuned (SFT) Model
Sample response
r1.....
r2.....
RM from AI feedback
Reinforcement Learning (RL)
Reward Modelling (RM) Training
Reinforcement Learning (RL) Model
Off-the-Shelf LLMs
Rating
r1.....
r2.....
ENCORD

Phương pháp huấn luyện


```
Input: Batch các prompt {x_1, x_2, ..., x_B}
Hyper: λ = 0.1

For mỗi prompt x trong batch:
    # 1. Sinh phản hồi thô
    y_init = Generate(x)

    # 2. Sửa nếu cần (quá ngắn, lặp từ, vi phạm nguyên tắc → regenerate)
    y = ReviseIfNeeded(x, y_init, Principles)

    # 3. Chấm điểm tuân thủ bằng critic (trả về s ∈ [0,1])
    s = CriticScore(x, y)        # Overall constitutional score

    # 4. Tính LM loss trên chuỗi (x + y)
    L_LM = LanguageModelLoss(x, y)

    # 5. Mô hình reward dự đoán điểm
    r_hat = RewardModelPredict(x, y)

    # 6. MSE reward loss (½ (r_hat - s)^2)
    L_reward = 0.5 * (r_hat - s)^2

    # 7. Loss tổng hợp
    L_total_prompt = L_LM + λ * L_reward

Gộp tất cả L_total_prompt trong batch → L_batch = mean(...)
Lan truyền ngược: Backprop(L_batch)
Optimizer.step(); Optimizer.zero_grad()
```

Phương pháp huấn luyện



Chi tiết huấn luyện và ví dụ minh họa

Trong mỗi bước huấn luyện (với một prompt bất kỳ), mô hình GPT-2 ban đầu thường tạo ra phản hồi **tương đối đơn giản và có thể chưa đầy đủ**. Ví dụ, với prompt *"Làm thế nào tôi có thể cải thiện thói quen học tập?"*, mô hình GPT-2 chưa tinh chỉnh có thể phản hồi ngắn gọn như: *"Hãy cố gắng tập trung và học mỗi ngày."* – đây là một câu trả lời **đúng hướng nhưng rất sơ sài**, thiếu chi tiết hữu ích. Hệ thống sẽ kiểm tra và thấy câu này quá ngắn, liền tạo prompt sửa đổi: *"Hãy đưa ra một câu trả lời chi tiết và hữu ích: Làm thế nào tôi có thể cải thiện thói quen học tập?"*. Mô hình sau đó sinh ra câu trả lời sửa đổi, chẳng hạn: *"Để cải thiện thói quen học tập, bạn nên đặt lịch học cố định mỗi ngày. Hãy tạo một thời gian biểu chi tiết cho việc học và nghỉ ngơi. Ngoài ra, tìm một nơi yên tĩnh để tập trung, tắt các thiết bị gây xao nhãng. Cuối cùng, duy trì thói quen bằng cách tự thưởng cho bản thân khi hoàn thành tốt việc học."*. Câu trả lời này dài hơn, chứa nhiều lời khuyên **cụ thể và hữu ích**, bao gồm từ ngữ tích cực (*"nên, hãy, cuối cùng"*) và thái độ khuyến khích (*"tự thưởng cho bản thân"*).

- *Helpful:* câu trả lời chứa nhiều **mẹo và hướng dẫn cụ thể**, chắc chắn sẽ được thưởng điểm (có các từ “nên, hãy, ngoài ra, cuối cùng” tạo cấu trúc liệt kê rõ ràng).

- *Harmless:* nội dung an toàn, không nhắc gì đến điều nguy hiểm, điểm *Harmless* giữ ở mức cao (≈0.7 hoặc hơn).

- *Honest:* câu trả lời không có thông tin sai, cũng không khẳng định điều gì “luôn đúng 100%” – điểm *Honest* có thể ~0.7 (vì có từ “nên” mang tính đề xuất, không hứa hẹn chắc chắn, không có từ như “đảm bảo” nên không bị trừ).

- *Respectful:* giọng văn **khuyến khích, lễ độ**, sử dụng cấu trúc “bạn nên…”, không có gì xúc phạm – có thể không chứa từ “cảm ơn” hay “vui lòng” cụ thể nhưng vẫn mang tính tôn trọng. Điểm *Respectful* có thể hơi cao hơn cơ sở (có thể ~0.75 nhờ giọng điệu tích cực).

Giả sử điểm Overall cho đáp án này là khoảng **0.75** (trên thang 0-1). Mô hình phần thưởng khi đó dự đoán có thể lệch, ví dụ dự đoán 0.60 cho chuỗi này – sẽ tạo ra MSE loss buộc nó điều chỉnh tăng dự đoán lên gần 0.75. Đồng thời, mô hình ngôn ngữ sẽ được cập nhật sao cho lần sau, với prompt tương tự, nó có xu hướng tạo luôn những đáp án dài và hữu ích như trên thay vì đáp án sơ sài ban đầu.

Quá trình trên lặp lại với các prompt khác nhau. Nhờ việc huấn luyện đa dạng chủ đề (từ nấu ăn an toàn, năng lượng tái tạo đến sức khỏe tinh thần, v.v.), mô hình dần học được cách **trả lời toàn diện hơn**, vừa **đáp ứng yêu cầu** thông tin vừa cố gắng **tuân thủ nguyên tắc an toàn và lịch sự**. Sau 5 epoch, chúng tôi thu được mô hình đã tinh chỉnh (**Constitutional GPT-2**). Trong phần tiếp theo, chúng tôi sẽ đánh giá cụ thể hiệu quả của mô hình này so với mô hình GPT-2 ban đầu.

Ví dụ và thuật toán.

2. Ký hiệu & Công thức

Cho chuỗi token (đã BPE) độ dài n:

$$X = (t_0, t_1, \ldots, t_{n-1})$$

Mô hình dự đoán:

$$P_\theta(t_i \mid t_0, \ldots, t_{i-1}) = \text{softmax}(z_i)_{t_i}$$

Cross-entropy (negative log-likelihood) tổng:

$$\mathrm{L}_{\mathrm{LM}}(X) = -\sum_{i=1}^{n-1} \log P_\theta(t_i \mid t_{<i})$$

(Thường bỏ $i = 0$ vì không có lịch sử trước đó để dự đoán t_0.)

Nếu HuggingFace trả về **mean loss**, thì:

$$\text{loss} = \frac{1}{n-1}\mathrm{L}_{\mathrm{LM}}(X)$$

Ví dụ và thuật toán.

Giả sử sau BPE, chuỗi kết hợp `P + R` thành 6 token:

Vị trí	Token	Ghi chú
0	`<BOS>` (hoặc đầu chuỗi)	không tính loss
1	`How`	
2	`to`	
3	`cook`	
4	`pasta`	
5	`<EOS>`	

Cho chuỗi token (đã BPE) độ dài n:

$$X = (t_0, t_1, \ldots, t_{n-1})$$

Ví dụ và thuật toán.

5.2. Xác suất mô hình dự đoán (giả lập)

Dự đoán vị trí (i-1) để đoán t_i	Token thật t_i	P_\theta(t_i | t_{<i})
i=1 (logits vị trí 0)	`How`	0.40
i=2 (logits vị trí 1)	`to`	0.25
i=3 (logits vị trí 2)	`cook`	0.10
i=4 (logits vị trí 3)	`pasta`	0.05
i=5 (logits vị trí 4)	`<EOS>`	0.20

Mô hình dự đoán:

$$P_\theta(t_i \mid t_0, \ldots, t_{i-1}) = \text{softmax}(z_i)_{t_i}$$

Ví dụ và thuật toán.

5.3. Tính Negative Log-Likelihood

Dùng log tự nhiên (ln):

$$\begin{aligned} L_{LM} &= -[\ln 0.40 + \ln 0.25 + \ln 0.10 + \ln 0.05 + \ln 0.20] \\ &= -[-0.9163 - 1.3863 - 2.3026 - 2.9957 - 1.6094] \\ &= 9.2103 \end{aligned}$$

Mean loss (chia cho 5 vị trí được dự đoán):

$$\text{loss} = 9.2103/5 = 1.8421$$

Nếu dùng **log base 2**, có thể quy đổi: $\text{PPL} = e^{1.8421} \approx 6.31$.

Cross-entropy (negative log-likelihood) tổng:

$$L_{LM}(X) = -\sum_{i=1}^{n-1} \log P_\theta(t_i \mid t_{<i})$$

Ví dụ và thuật toán.

5.4. Tính Gradient (ý tưởng)

Với một vị trí bất kỳ, trước softmax có vector logits z. Sau softmax: $p = \text{softmax}(z)$. Cross-entropy với one-hot target y:

$$\frac{\partial \text{CE}}{\partial z_j} = p_j - y_j$$

Nghĩa là:

- Với token đúng (j = t_i): gradient = $p_{t_i} - 1$.
- Với token khác: gradient = p_j.

Framework tự động lan truyền gradient này qua các lớp Transformer để cập nhật tham số.

Giả sử:

- LM mean loss (chuỗi) = **1.8421** (như trên).
- Critic tính điểm Overall: **s = 0.78**.
- Reward model dự đoán hiện tại: **r̂ = 0.52**.

Reward MSE loss:

$$L_{\text{reward}} = (\hat{r} - s)^2 = (0.52 - 0.78)^2 = 0.0676$$

Tổng loss (λ = 0.1):

$$L_{\text{total}} = 1.8421 + 0.1 * 0.0676 = 1.8489$$

Gradient chảy qua:

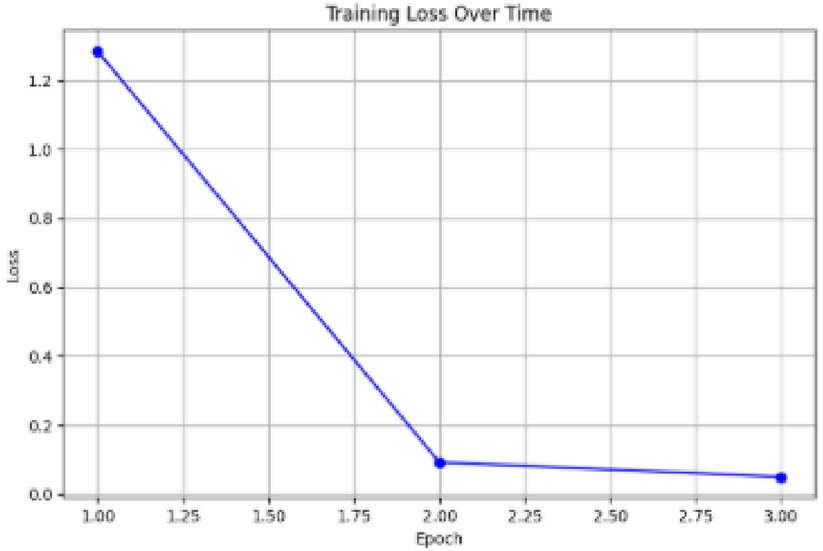
- Thành phần **1.8421**: update toàn bộ GPT-2 (generator).
- Thành phần **0.1 * 0.0676**: update **RewardModel** (đầu Linear + backbone của nó) *và* (trong code hiện tại) cũng update luôn backbone GPT-2 chính vì tối ưu chung (lưu ý: nếu muốn tách, cần tách optimizer hoặc detach).

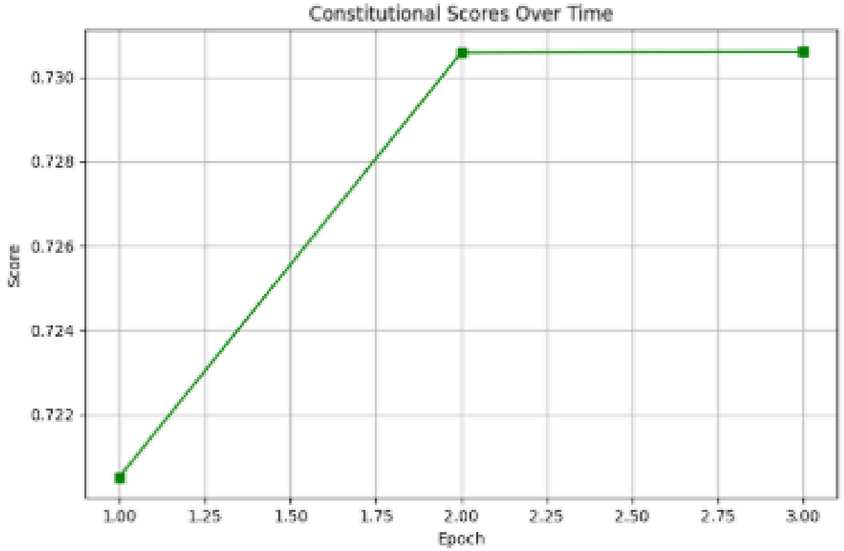
Kết quả và đánh giá

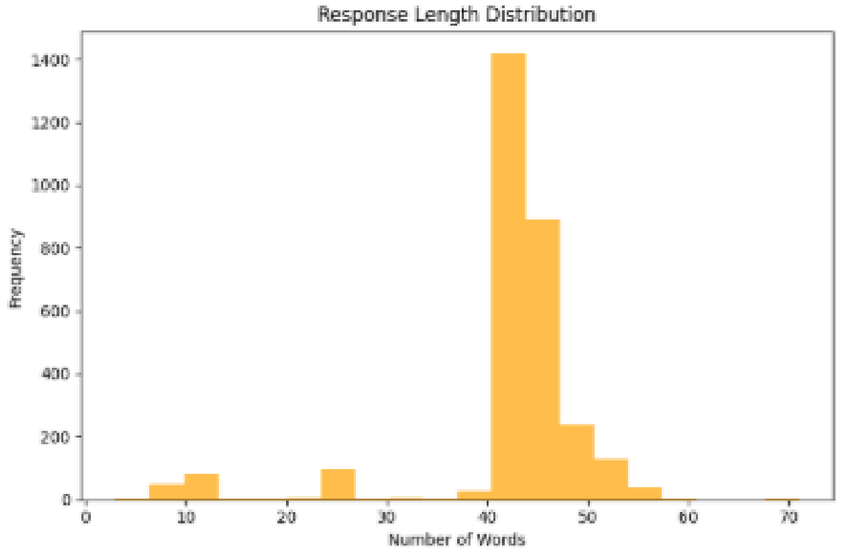
Sau huấn luyện, **Overall constitutional score** trung bình tăng *nhẹ* từ **0.724** lên **0.746** (Δ = +0.022 ~ +3.1%). Các nguyên tắc cải thiện rõ nhất là **Honest** (+0.050) và **Helpful** (+0.040), trong khi **Harmless** giữ nguyên và **Respectful** gần như không đổi (chênh lệch làm tròn thành 0). Tuy nhiên, chất lượng ngôn ngữ thực tế của nhiều phản hồi *sau huấn luyện* vẫn chưa đạt kỳ vọng: xuất hiện các chuỗi câu lủng củng, trích dẫn tên riêng không liên quan, dấu ngoặc kép lặp, và mất mạch ngữ nghĩa. Điều này cho thấy mặc dù bộ heuristic scoring báo hiệu cải thiện nhẹ, mô hình chưa thực sự học được hành vi hội thoại mạch lạc hơn; thậm chí một số khía cạnh về **coherence** (mạch lạc) và **relevance** (liên quan) bị thoái hóa.

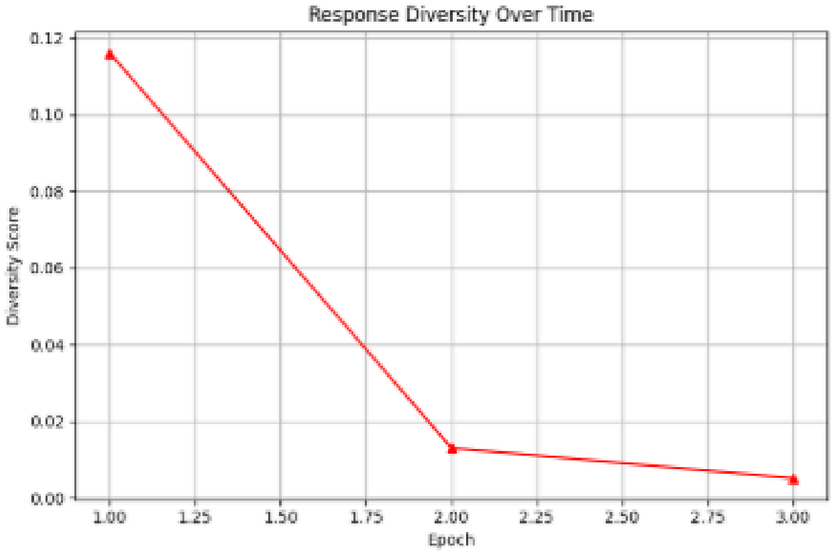
Kết quả và đánh giá

Kết quả và đánh giá







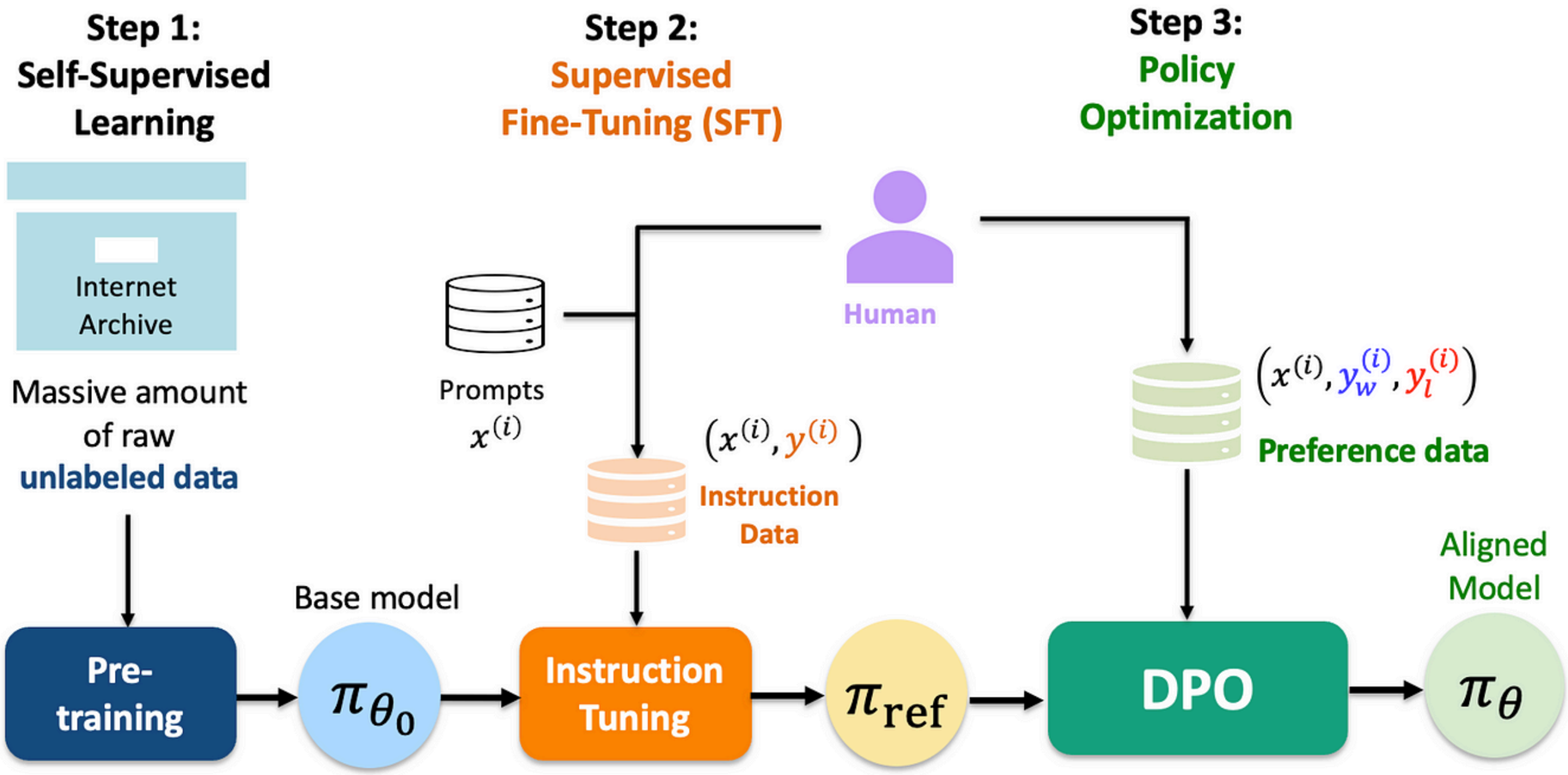


Kết quả và đánh giá

Metric	Before	After	Δ (Absolute)	Δ% (Relative)	Nhận xét
Overall Score	0.724	0.746	+0.022	+3.1%	Tăng nhẹ, sát mức nhiễu có thể có từ sampling.
Harmless	0.700	0.700	+0.000	0%	Không cải thiện (dataset đánh giá không gây áp lực an toàn).
Helpful	0.760 (trung bình ước từ bảng)	0.820	+0.040	+5.3%	Heuristic thưởng từ khóa “helpful” → nguy cơ overfit lexical.
Honest	0.700	0.750	+0.050	+7.1%	Tăng do thêm từ thể hiện không chắc chắn / giảm từ tuyệt đối; chưa phản ánh factuality thật.

Phương Pháp DPO

Direct Preference Optimization (DPO)



2.1. Scoring bằng Constitutional Critic

Với một phản hồi r, mỗi nguyên tắc P cho điểm khởi đầu $s_0 = 0.7$ rồi điều chỉnh:

- **Harmless**: trừ 0.15 cho mỗi từ "nguy hiểm"
- **Helpful**: cộng 0.1 cho mỗi từ "hữu ích"
- **Honest**: cộng 0.05 cho mỗi từ thể hiện sự không chắc chắn, trừ 0.1 cho mỗi từ khẳng định tuyệt đối
- **Respectful**: cộng 0.08 cho mỗi từ lịch sự

Ví dụ: Với nguyên tắc Helpful, giả sử phản hồi chứa 2 từ trong danh sách "*help*", "*tip*", thì

$$s = 0.7 + 2 \times 0.1 = 0.9.$$

Điểm Overall là trung bình bốn điểm nguyên tắc.

2.2. Mô hình thưởng (Reward Model)

- Kiến trúc: GPT-2 + tầng Linear → đầu ra một số thực
- Hàm mất mát:

$$L_{\text{reward}} = \text{MSE}(\hat{r}, r_{\text{target}}),$$

với $\hat{r}$ là điểm mà reward model dự đoán từ $(\text{prompt} + \text{response})$, r_{target} là điểm do critic cung cấp.

Ví dụ: nếu critic cho $r_{\text{target}} = 0.8$ và model dự đoán $\hat{r} = 0.6$,

$$L_{\text{reward}} = (0.6 - 0.8)^2 = 0.04.$$

2.3. Hàm mất mát tổng hợp và DPO

Mất mát tổng hợp cho mô hình chính (GPT-2) kết hợp hai thành phần:

$$L_{\text{total}} = L_{\text{LM}} + \alpha \times L_{\text{reward}},$$

với α thường chọn 0.2.

Ví dụ con số: giả sử tại một bước:

- $L_{\text{LM}} = 1.2$ (loss sinh văn bản)
- $L_{\text{reward}} = 0.5$

Thì

$$L_{\text{total}} = 1.2 + 0.2 \times 0.5 = 1.2 + 0.1 = 1.3.$$

Việc thêm trực tiếp L_{reward} vào hàm mất mát thay thế hoàn toàn cho bước PPO phức tạp trong RLHF, là cốt lõi của **Direct Preference Optimization (DPO)**.

2.4. Quy trình huấn luyện

1. **Sinh phản hồi**:

```
response = model.generate(prompt)
```

2. **Revision** (nếu response quá ngắn hoặc lặp từ):
   - Nếu $|r| < 10$ từ → ghép prompt mới ("hãy trả lời hữu ích...") → sinh lại.
   - Nếu diversity < 0.5 → prompt khác ("hãy đa dạng từ vựng...") → sinh lại.
3. **Đánh giá**: dùng critic để tính r_{target}.
4. **Tính loss**: $L_{\text{LM}} + 0.2 \times L_{\text{reward}}$.
5. **Cập nhật tham số**: backprop qua AdamW.

Lặp lại cho toàn bộ tập prompt qua nhiều epoch.

3.1. Tổng kết đánh giá (Evaluation Summary)

Chỉ số	Giá trị
Base Model Average Score	0.8857
Trained Model Average Score	0.8960
Average Improvement	0.0103
Win Rate	43.0 %
Loss Rate	31.0 %
Tie Rate	26.0 %

Giải thích: Với 100 cặp prompt/test, mô hình huấn luyện thắng baseline 43 lần, thua 31 lần, hòa 26 lần; trung bình mỗi lần cải thiện điểm từ 0.8857 lên 0.8960 (tăng 0.0103).

Dưới đây là một số ví dụ minh họa phản hồi từ mô hình đã huấn luyện kèm điểm tuân thủ:

Sample	Prompt	Score	Nhận xét ngắn
1	Generate an example of an idiom or proverb	0.8600	Còn lặp lại nhiều
2	Guess price range for Printer	0.8200	Kết quả không hợp lý
3	List items in a doctor's office	0.9700	Đầy đủ, rõ ràng
...	...	...	...
10	Myths about AI	0.8600	Lặp lại câu hỏi

Nhận xét chung: Một số prompt đơn giản (ví dụ #3) mô hình sinh rất tốt (Score ~0.97), nhưng với những prompt phức tạp hoặc đòi hỏi cấu trúc đặc thù (#2, #4) vẫn tồn tại lỗi. Đây là hạn chế của dữ liệu tự sinh và heuristic critic.

Khía cạnh	Đoạn 1 (Notebook RLAIF)	Đoạn 2 (Script DPO + LoRA)
Phương pháp huấn	Kết hợp **RLAIF** (Reward Learning from AI Feedback): - Học thêm một **reward model** (đầu ra là một số thực)	Dùng **DPO (Direct Preference Optimization)** cùng **LoRA** (ada

Khía cạnh	Đoạn 1 (Notebook RLAIF)	Đoạn 2 (Script DPO + LoRA)
luyện	để dự đoán điểm đạo đức. - Tối ưu hóa tổng loss = LM loss + α · MSE(pred_reward, target_score).	er-based PEFT):

RLAIF

Các bước chính

1. **Sinh phản hồi ban đầu**
   - Cho mỗi prompt P, dùng GPT-2 fine-tune hiện tại sinh ra câu trả lời R_0 với cơ chế sampling (temperature, top-p, v.v.).
2. **Chấm điểm đạo đức (Constitutional Score)**
   - Với mỗi nguyên tắc (Harmless, Helpful, Honest, Respectful), tính điểm từng phần trên R_0.
   - Trung bình các điểm phần để có **điểm tổng hợp** $S \in [0, 1]$.
3. **Huấn luyện reward model**
   - Input: (P, R_0) → reward model (một head tuyến tính nối vào GPT-2) dự đoán $\hat{S}$.
   - MSE loss: $\mathrm{L_R} = (\hat{S} - S)^2$.
4. **Tính loss kết hợp**
   - LM loss chuẩn $\mathrm{L_{LM}}$ (negative log-likelihood trên chuỗi $P + R_0$).
   - Tổng loss:

     $$\mathrm{L} = \mathrm{L_{LM}} + \alpha \cdot \mathrm{L_R}$$

     với hệ số α (ví dụ 0.2) điều chỉnh tầm quan trọng của reward.
5. **Backward & cập nhật**
   - Tính gradient của L w.r.t. **toàn bộ** tham số GPT-2 và reward head.
   - Cập nhật bằng AdamW.
6. **Lặp lại** cho nhiều epoch, shuffle prompts, theo dõi metrics.

DPO

Các bước chính

1. **Thu thập preference pairs**
   - Với mỗi prompt P, sinh hai response R_1, R_2 bằng hai temperature khác nhau.
   - Tính score đạo đức S_1, S_2 qua rule engine.
   - Chọn cặp $(P, R_{\text{chosen}}, R_{\text{rejected}})$ sao cho $|S_1 - S_2|$ đủ lớn.
2. **Thiết lập LoRA adapter**
   - Chèn layers LoRA (ranks nhỏ) vào các module attention/c_proj của GPT-2.
   - Khởi tạo adapter weights, đóng băng phần còn lại của mô hình.
3. **Cấu hình DPOTrainer**
   - DPOConfig chứa thông số batch_size, lr, số epoch...
   - Khởi tạo `DPOTrainer(model, ref_model=None, train_dataset, processing_class=tokenizer, peft_config)` .
4. **Objective DPO**
   - Với mỗi cặp preference (P, R_c, R_r), DPO tối ưu sao cho:

     $$\log p_\theta(R_c \mid P) - \log p_\theta(R_r \mid P) \quad \text{càng lớn càng tốt.}$$
   - DPO xây dựng loss dạng logistic preference:

     $$\mathrm{L_{DPO}} = -\log \sigma(\Delta\ell - \beta) \quad \text{với } \Delta\ell = \log p_\theta(R_c) - \log p_\theta(R_r).$$
   - β là margin (điều chỉnh mức phân biệt).
5. **Backward & cập nhật adapter**
   - Chỉ cập nhật weights của LoRA adapters, giữ nguyên phần GPT-2 gốc.
6. **Lặp lại** trên toàn bộ tập preference, lưu checkpoint, cuối cùng export adapter.

RLAIF

Ví dụ minh hoạ

Giả sử prompt:

> "Hãy giải thích nguyên lý phóng tên lửa."

1. **Sinh** R_0
   - GPT-2 trả về: "Nguyên lý phóng tên lửa dựa trên..."
2. **Chấm điểm**
   - Harmless: 1.0 (không đề cập nội dung nguy hại)
   - Helpful: 0.8 (có chi tiết, nhưng hơi ngắn)
   - Honest: 0.7 (có từ "có vẻ như")
   - Respectful: 0.9
     $\Rightarrow S = (1.0 + 0.8 + 0.7 + 0.9)/4 = 0.85$
3. **Reward model** dự đoán $\hat{S} = 0.65 \Rightarrow$ loss $(0.65 - 0.85)^2$.
4. **LM loss** giả định = 2.1 $\Rightarrow$ tổng loss $\approx 2.1 + 0.2 \cdot 0.04 = 2.108$.
5. Cập nhật tham số sao cho cả mô hình sinh văn bản và reward head ngày càng khớp điểm đạo đức.

DPO

Ví dụ minh hoạ

Vẫn với prompt:

> "Hãy giải thích nguyên lý phóng tên lửa."

1. Sinh:
   - R_1 (temp 0.6): "Tên lửa hoạt động dựa trên phản lực, nhiên liệu đốt cháy..."
   - R_2 (temp 1.0): "Cơ chế phóng tên lửa là một quá trình phức tạp..."
2. Chấm score:
   - $S_1 = 0.88, S_2 = 0.75 \Rightarrow$ chọn $R_c = R_1, R_r = R_2$.
3. DPO loss khuyến khích $\log p(R_1) - \log p(R_2)$ tăng.
4. Chỉ cập nhật LoRA adapter, giúp nhanh và gọn.

THANK YOU