Отчет

Чурилкин Артем
Project on GitHub

Level 1

В статье Ahmadian et al, Back to Basics: Revisiting REINFORCE Style Optimization for Learning from Human Feedback in LLMs предлагают отказаться от сложностей PPO и использовать более простые методы, такие как REINFORCE, который и был реализован в **Level1**. Потенциально мы тратим меньше вычислительных ресурсов при обучении LLM, а получаем результат лучше.

Основная идея метода:

- **Обучаем политику (LLM)**, максимизируя ожидаемую награду.
- **Используем Reward Model (RM)** для оценки сгенерированных текстов.
- **Применяем скользящее среднее награды (moving average)** в качестве *baseline*, чтобы уменьшить дисперсию градиента.

Этот метод **заменяет сложный PPO** (используемый в RLHF) на более простую и эффективную альтернативу.
Ниже приведен псевдокод, на основе которого и был реализован требуемый алгоритм:

Algorithm 1 REINFORCE with Baseline for Fine-Tuning LLMs

Initialize: SFT model, Reward Model, tokenizer
Set parameters: batch_size, learning_rate, α (for baseline)
$b \leftarrow 0.0$ ▷ Initialize baseline
for *step* $= 1$ to num_iterations **do**
    Sample batch_size prompts x_i from dataset
    Generate responses y_i using SFT model
    Compute reward $r_i = RM(y_i)$
    Update baseline: $b \leftarrow \alpha b + (1 - \alpha) \cdot \text{mean}(r_i)$
    Compute log probabilities $\log \pi_\theta(y_i|x_i)$
    Compute advantage: $A_i = r_i - b$
    Compute loss: $L = -\frac{1}{N} \sum_i A_i \cdot \log \pi_\theta(y_i|x_i)$
    Zero gradients
    Perform backpropagation
    Update model parameters
    if *step* $\mod 50 = 0$ **then**
        Print current loss, reward, baseline
    end if
end for

Работали с датасетом $esfrankel17/HelpSteer2_binarized$.
В качестве SFT модели выступает HuggingFaceTB/SmolLM2-135M-Instruct. Все используемые для подсчета метрик модели сохранены в архивах (.zip, .rar) на GitHub с использованием Git LFS.
Были подсчитаны оценка средней награды и KL-дивергения на validation наборе.

Level 2

Также была предпринята попытка обучить Reward Model, которая выдаёт не скалярную оценку, а распределение вероятности поверх дискретных оценок.

Мы можем поставить задачу так: пусть Reward Model для данного текста y выдаёт вектор

$$p(y) = \big(p_1(y), p_2(y), \ldots, p_{10}(y)\big)$$

где $p_i(y)$ — вероятность того, что текст получает оценку i (при этом $\sum_{i=1}^{10} p_i(y) = 1$ из соображений полной вероятности). Если у нас есть пара текстов для одного промпта x: выбранный y_w и отвергнутый y_l, то мы хотим, чтобы выбранный текст получал, как правило, более высокую оценку.

Рассмотрим такую идею: допустим, мы случайным образом берем оценки из распределений $p(y_w)$ и $p(y_l)$. В таком случае вероятность того, что оценка выбранного текста окажется больше, чем оценка отвергнутого, равна

$$p(y_w \succ y_l \mid x) = \sum_{i=1}^{10} \sum_{j=1}^{i-1} p_i(y_w)\, p_j(y_l).$$

То есть мы суммируем по всем парам оценок, где индекс выбранного текста i строго больше, чем индекс отвергнутого j. Этот способ кажется достаточно естественным для задания вероятности того, что выбранный текст «выигрывает» отвергнутый, поскольку не является чем-то "страшным"для реализации+понимания и при этом имеет за собой весьма понятную идею. Для максимизации этой вероятности("хороший"ответ получает оценки выше, чем "плохой") предлагается использовать следующую функцию потерь:

$$\mathcal{L} = -\log\left(\sum_{i=1}^{10} \sum_{j=1}^{i-1} p_i(y_w)\, p_j(y_l)\right).$$

Теперь немного обсудим логику, которая стоит за этой функцией потерь. Допустим, выбранный текст действительно лучше, тогда его распределение $p(y_w)$ должно быть «сдвинуто» в сторону высоких оценок, а $p(y_l)$ — в сторону низких. В таком случае сумма

$$\sum_{i>j} p_i(y_w)\, p_j(y_l)$$

будет близка к 1, а $-\log(\cdot) \longrightarrow 0$. Если же отвергнутый текст получает не меньшие оценки, чем выбранный, то сумма будет меньше, а функция потерь выше.

Таким образом, минимизация $\mathcal{L}$ соответствует максимизации вероятности того, что выбранный ответ предпочтительнее отвергнутого.

Анализ полученных результатов

Базовая (SFT) модель

Средняя награда: 0.652. Это значение служит исходной точкой и отражает качество ответов исходной модели согласно наградной функции RM.

RLHF модель, дообученная с использованием скалярной RM

Средняя награда: 0.705. По сравнению с базовой моделью наблюдается рост средней награды (с 0.652 до 0.705). Это говорит о том, что процесс RLHF с использованием скалярного сигнала награды сумел сместить генерацию модели в сторону ответов, которые Reward Model оценивает чуть лучше.

Как на это можно посмотреть: такой рост, пусть и не очень большой по абсолютной величине, но все же является положительным сигналом. Можно предположить, что модель стала генерировать ответы, более соответствующие критериям качества, заложенным в RM.

В таком случае важно, чтобы модель не «отошла» слишком сильно от исходной, что можно контролировать, например, измеряя KL-дивергенцию. Это позволит избежать деградации генеративных свойств или потери знаний, полученных при предобучении. В нашем случае **KL** = 0.0101 для Level 1, но число само по себе нельзя однозначно назвать хорошим или плохим без контекста. В некоторых

исследованиях устанавливают допустимые значения порядка 0.1, чтобы ограничить отклонение. В нашем случае это может означать, что модель сместилась намного меньше, чем хотелось бы. Есть основания полагать, что это не обязательно плохо, поскольку при этом наблюдается улучшение по наградной метрике.

Было бы здорово посчитать KL еще и для Level 2, но это не удалось в связи с техническими проблемами при работе с моделью(возможно она некорректно сохранилась в Google Colab, а перезаписывать возможности уже нет — закончились бесплатные compute unit'ы на нескольких аккаунтах), хотя такая цель была. Поэтому имеем только одну цифру, на основе которой было бы не совсем корректно делать какие-либо заявления о интегральной эффективности)

RLHF модель, дообученная с использованием вероятностной RM

Средняя награда: 0.339. Здесь ожидаемое значение награды оказалось существенно ниже, чем для базовой модели и RLHF с скалярной RM.
Попробуем объяснить такую разницу несколькими **факторами**.

Калибровка шкалы наград: Вероятностная RM выдаёт распределение вероятностей по оценкам (от 1 до 10), и ожидаемое значение вычисляется как взвешенная сумма оценок. Вероятно вышло, что RM обучилась таким образом, что распределения получаются «размазанными» или смещёнными вниз, ожидаемое значение может оказаться ниже, даже если качественные свойства ответов улучшились.

Возможно, функция потерь для вероятностной RM (которая максимизирует вероятность того, что выбранный ответ получает более высокую оценку, чем отвергнутый) даёт другой масштаб изменения сигнала, и RLHF процесс с текущими гиперпараметрами не смог эффективно использовать этот сигнал для улучшения генерации. Гиперпараметры были выбраны именно такими в связи с ограничениями в вычислительных мощностях и времени(напомним, что для обоих Level'ов они совпадают), поэтому не стоит исключать, что при подборе других значений результат мог измениться.

Еще есть мысль связанная с шумами, а именно — распределение оценок может быть более «шумным»/«размазанным», то есть RM склонна к высокой энтропии, что приводит к более низким ожидаемым значениям награды. В свою очередь, RLHF процесс, основанный на подобном подходе, может оказаться менее устойчивым, что может быть проверено сменой гиперпараметров(скорость обучения, коэффициенты baseline и т.д.) и изучением на другом датасете. Поэтому на случай, когда целью является использование вероятностной RM, нам может потребоваться масштабирование или перенормировка ожидаемого значения награды, чтобы привести его к сопоставимому диапазону со скалярным вариантом.

Итоговый вывод

Результаты показывают, что методика RLHF способна внести изменения в поведение модели, но качество этих изменений напрямую зависит от формы наградного сигнала. Скалярная RM дала положительный эффект, а вероятностная RM — негативный, что требует дополнительного исследования. Важно сравнить не только числовые метрики, но и провести качественную оценку сгенерированных ответов (например, парное сравнение или оценку экспертами), а также анализировать динамику KL-дивергенции, чтобы понять, насколько RLHF модель отклонилась от исходной SFT. Вероятностная RM требует дополнительной настройки и, возможно, переосмысления используемой функции награды. Это открывает перспективы для дальнейших исследований и экспериментов для оптимального дообучения генеративных моделей.

Что можно было бы сделать дальше

1. **Провести калибровку вероятностной RM:** Определить, соответствует ли масштаб ожидаемого значения оценок реальным предпочтениям. К примеру, можно попробовать использовать дополнительные методы калибровки (например, температурное масштабирование).

2. **Изменить гиперпараметры RLHF:** Как бы банально это не звучало, но ограничиваться только одним сетом параметров — не даст полноценной картины, тем более, когда мы выяснили, что ресурсов требуется значительно меньше, чем для PPO. Можно попробовать увеличить вес сигнала награды или изменить параметры baseline, чтобы RLHF модель лучше адаптировалась к вероятностному сигналу.

3. **Качественный анализ:** Провести экспертное сравнение сгенерированных ответов от всех трёх вариантов (SFT, RLHF со скалярной RM и RLHF с вероятностной RM) для более глубокого понимания, какие изменения произошли. Есть подозрения, что разметчики с таким справиться вполне могут, но тут уже возникает другой вопрос — а насколько им можно верить в таких случаях(будут ли их мнения достаточно валидными?) и как вообще обрабатывать их мнения... хотя это уже вопрос отдельного исследования

4. **Мониторинг других метрик:** Помимо средней награды, можно добавить измерения разнообразия, перплексности и тп. Если возможно, проводить A/B тестирование с участием реальных пользователей.