Universidad Nacional de Colombia
Facultad de Ciencias - Departamento de Matemáticas
Matemáticas del Aprendizaje de Máquina - 2024 I

Santiago Botero Daza — sboterod@unal.edu.co
David José Daza Jaimes — ddazaj@unal.edu.co
Mateo Sebastian Ortiz Higuera — mortizhi@unal.edu.co
David Leonardo Ortiz Uribe — dlortizu@unal.edu.co

Human Feedback for LLMs

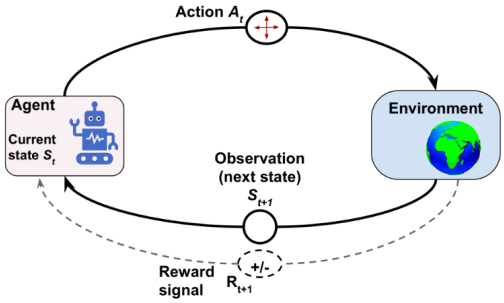
1. Introducción

En el presente reporte introduciremos el método de Reinforcement Learning for Human Feedback (RLHF) y el método Direct Reward Optimization en modelos de generación de lenguaje. Abordaremos los principales temas adyacentes a estos métodos, explicando desde el Modelo de Recompensa hasta la Optimización de la Política del modelo. Finalmente expondremos algunos resultados de experimentos realizados.

1.1 Sobre RL

Antes de hablar sobre RLHF, hagamos una pequeña introducción a lo que es Reinforcement Learning. El Reinforcement Learning es un subcampo del machine learning en el que una agente aprende a tomar decisiones en un entorno interactuando con él. El agente interactúa a través de acciones guiadas por una política y recibe una retroalimentación del entorno en forma de una recompensa y un nuevo estado. El objetivo del proceso de entrenamiento es optimizar la política de tal manera que se maximice el valor esperado de la recompensa acumulada a partir de cualquier estado inicial.

Figura 1: Esquema del método RL

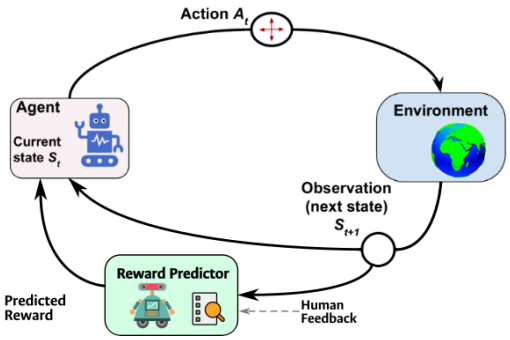


1.2 Sobre RLHF

Ahora bien, en algunas ocasiones, por la complejidad del problema, dichas recompensas pueden no ser suficientes para garantizar que el agente aprenda a tomar las mejores decisiones en un contexto muy específico. Pensemos por ejemplo en un modelo de generación de lenguaje, supongamos que por alguna razón sin justificación introducimos el siguiente prompt "`¿Cómo construir una bomba?`", efectivamente nuestro agente podría respondernos con una lista de pasos para construir una bomba. Sin embargo, lo ideal es que ante una pregunta de este estilo, el agente esté en capacidad de dar una respuesta más sensata.

El enfoque realizado en RLHF es el de entrenar un modelo de recompensa a partir de retroalimentación humana. Volviendo al ejemplo anterior, otra posible respuesta podría ser "`No deberías construir una bomba.`", está claro que esta nueva respuesta es bastante mejor en comparación a la anterior. El modelo de recompensa a partir de retroalimentación humana nos permitiría preferir la segunda respuesta respuesta sobre la primera y así ajustar mejor nuestro modelo inicial.

Figura 2: Esquema con la retroalimentación humana



2. Reward Model

El modelo de recompensa para modelos de lenguaje se realiza por comparación, y es que de esta forma se hace más fácil ajustar nuestro modelo inicial, tomemos por ejemplo ocurre en el ejemplo de la sección anterior.

Formalmente, dado un estado $s_t \in \mathcal{S}$ y una acción $a_t \in \mathcal{A}$ en un tiempo t, definimos una política como sigue $\pi : \mathcal{S} \longrightarrow \mathcal{A}$. Después de que el entorno produzca una señal de recompensa, supongamos que existe un humano que puede expresar preferencias entre segmentos de trayectoria, esto último podemos definirlo formalmente como una secuencia de observaciones y acciones

$$\varsigma = ((s_0, a_0), (s_1, a_1), \dots, (s_{n-1}, a_{n-1})) \in (\mathcal{S} \times \mathcal{A})^n$$

y escribimos $\varsigma^1 \succ \varsigma^2$ para denotar que dicho humano prefiere el segmento ς^1 sobre el segmento ς^2 de una trayectoria τ. De esta forma, podemos definir una función de recompensa como sigue $r : \Gamma^* \times (\mathcal{S} \times \mathcal{A})^n \longrightarrow \mathbb{R}$, donde Γ es un alfabeto y $x \in \Gamma^*$ un prompt o pregunta. El objetivo del modelo es el de producir trayectorias que son preferidas por el humano

mientras se realiza el menor número de consultas posibles para el humano. [4]

2.1 Ajustando el modelo

Dados dos segmentos tales que $\varsigma^1 \succ \varsigma^2$ dada una pregunta x, utilizando el modelo Bradley-Terry, tenemos que $P(\varsigma^1 \succ \varsigma^2, x)$ está dado como sigue

$$P(\varsigma^1 \succ \varsigma^2, x) = \frac{\exp(r_\phi(x, \varsigma^1)))}{\exp(r_\phi(x, \varsigma^1)) + \exp(r_\phi(x, \varsigma^2))}$$

por esto se puede deducir que la función para ajustar nuestro modelo está dada por

$$L(r) = \min_\phi -\mathbb{E}_{(x,\varsigma^1,\varsigma^2)} \log \sigma[r_\phi(x, \varsigma^1) - r_\phi(x, \varsigma^2)] \qquad (1)$$

donde σ representa la función sigmoide y ϕ los parámetros implícitos en la función r a maximizar.

3. Política

Como dijimos previamente, lo que buscamos con el Reinforcement learning es seleccionar la política que maximice la recompensa esperada, formalmente esto es:

$$\pi^* = \arg \ \max_\pi J(\pi)$$

Donde la recompensa esperada es la recompensa esperada sobre todas las posibles trayectorias, es decir, podemos verlo así

$$J(\pi) = \int_\tau P(\tau|\pi)R(\tau) = \mathbb{E}_{\tau\sim\pi}[R(\tau)].$$

Donde la probabilidad de la trayectoria es

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

y la recompensa es $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t$

4. Policy Gradient Optimization

Ahora supongamos que tenemos una política parametrizada por θ. Podemos usar Stochastic Gradient Descent para optimizar los parámetros, de tal manera que:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)\Big|_{\theta_k}$$

La expresión del gradiente de la política está dada por:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right]$$

A fines prácticos lo que calcularemos es una aproximación a partir de un conjunto de trayectorias, pero esto tiene un problema, ya que al tomar una muestra del total de trayectorias tendremos una varianza alta, por lo que para disminuirla se introduce un nuevo concepto llamado Advantage function teniendo una expresión de la forma:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_{t,i}|s_{t,i})\right) A^t(s,a)$$

Lo que hace la Advantage function es decirnos qué tan bueno es elegir una acción particular **a** en un estado **s** por encima de la expectativa promedio que obtenemos al elegir aleatoriamente una acción, aumentando la recompensa para dichas acciones.

5. Proximal Policy Optimization

Es un algoritmo utilizado para entrenar un agente en el que buscaremos optimizar la siguiente función:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

donde:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

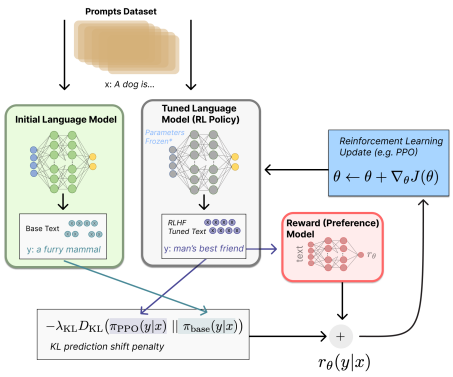
Aquí, $\pi_\theta(a_t \mid s_t)$ es la probabilidad de tomar la acción a_t en el estado s_t bajo la nueva política con parámetros θ, y $\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ es la probabilidad bajo la política anterior.

$\hat{A}_t$ es la ventaja estimada en el tiempo t.

ϵ es un hiperparámetro que determina cuán agresivo es el clipping, con un valor típico de 0.2.

El proceso consiste en recolectar trayectorias, calcular las ventajas y optimizar la función objetivo. Recolectamos trayectorias tanto de la política antigua como de la nueva con el fin de que el proceso no sea tan lento.

Figura 3: Esquema de todo el proceso. Tomado de `https://huggingface.co/blog/rlhf`



6. *Direct Preference Optimization (DPO)

Otro método alternativo es el llamado DPO, con el cual, a diferencia del método RLHF, donde se aprende una recompensa y luego optimiza a través de RL, buscamos optimizar

directamente el modelo de lenguaje utilizando una parametrización particular de la función de recompensa que permite extraer su política óptima en una forma cerrada, sin necesidad de un ciclo de entrenamiento RL. [8]

Recordemos que el objetivo del método RLHF es el de encontrar una política que maximice la recompensa, esto puede ser interpretado como sigue

$$\max_{\pi_\theta} \mathbb{E}[r_\phi(x,\varsigma)] - \beta \mathbb{D}_{\mathrm{KL}}[\pi_\theta(\varsigma|x)||\pi_{\mathrm{ref}}(\varsigma|x)]$$

La solución a dicho problema de optimización está dado por la siguiente ecuación

$$\pi_r(\varsigma|x) = \frac{1}{Z(x)}\pi_{\mathrm{ref}}(\varsigma|x)\exp\left(\frac{1}{\beta}r(x,\varsigma)\right) \tag{2}$$

donde $Z(x) = \sum_\varsigma \pi_{\mathrm{ref}}(\varsigma|x)\exp\left(\frac{1}{\beta}r(x,\varsigma)\right)$. Sin embargo, operar dicha ecuación es muy costoso computacionalmente hablando.

Suponiendo que tenemos la política óptima $\pi_r(\varsigma|x)$, por 2, la función de recompensa está dada por

$$r(x,\varsigma) = \beta \log \frac{\pi^*(\varsigma|x)}{\pi_{\mathrm{ref}}(\varsigma|x)} + \beta \log Z(x) \tag{3}$$

Finalmente, por 1 y por 3, la función para ajustar el modelo está dada por

$$L(\pi_\theta;\pi_{\mathrm{ref}}) = -\mathbb{E}\left[\log\sigma\left(\beta\frac{\pi_\theta(\varsigma^1|x)}{\pi_{\mathrm{ref}}(\varsigma^1|x)} - \beta\frac{\pi_\theta(\varsigma^2|x)}{\pi_{\mathrm{ref}}(\varsigma^2|x)}\right)\right]$$

Entonces, en lugar de optimizar la función de recompensa, estamos optimizando la política óptima (la cual depende la función de recompensa como se ve en 2).

7. Experimentos

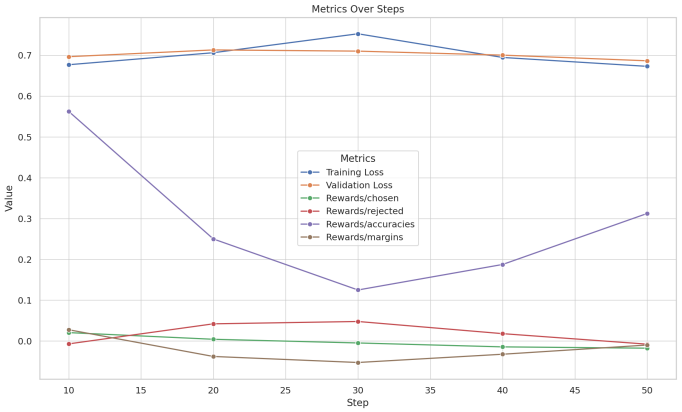
Con la intención de aprender a ajustar un modelo LLM preentrenado a preferencias humanas, encontramos un código en GitHub [7], extraído de YouTube [6], que fue ligeramente modificado en Google Colab [1]. Este código utiliza el algoritmo DPO para entrenar el LLM Hermes Mistral 2-GPTQ, obtenido de Hugging Face [10]. Este modelo, con 7 mil millones de parámetros, ya había sido preentrenado y afinado. Utilizamos un dataset en formato "prompt-chosen-rejected" para ajustar el modelo a preferencias humanas mediante DPO. Aunque se empleó un ultradataset [5] generado por otros LLMs, este podría ser reemplazado sin problema por uno generado por humanos.

Se empleó la técnica PEFT (parameter efficient fine tuning) para entrenar el modelo, ajustando solo una proporción de los parámetros y no todos, lo que hace el entrenamiento más eficiente y viable. Además, se redujo el consumo de memoria, permitiendo que el modelo fuera entrenado y usado para inferencia en Google Colaboratory, cuantizando y representando pesos y activaciones en 4 bits. Una vez entrenado, se hizo inferencia con el modelo y se comparó con el modelo original sin DPO.

7.1 Resultados

El entrenamiento con DPO reveló las siguientes métricas:

Figura 4: Métricas sobre pasos, basado en [1]



Se observa una ligera reducción en la función de pérdida y en la pérdida de validación para el paso 50. Sin embargo, otras métricas no mostraron cambios significativos en el modelo. El reward/chosen disminuyó cuando debería aumentar, el reward/rejected se mantuvo casi igual, aunque afortunadamente negativo, y reward/accuracies y reward/margin también disminuyeron, aunque deberían haber aumentado [2]. Sin embargo, estos cambios fueron insignificativos. Es posible que con otros hiperparámetros y más recursos de memoria para operaciones y representación de pesos y activaciones, se pueda obtener un mejor desempeño del modelo al ser entrenado.

Sorprendentemente, notamos un cambio cualitativo significativo en la respuesta generada por el modelo ante el prompt ejemplo propuesto [1], lo que sugiere que es posible observar mejoras cualitativas significativas en el modelo a pesar de cambios aparentemente insignificativos en las métricas que miden su entrenamiento.

Referencias

[1] Human_feedback_for_llm_dpo_project.ipynb. `https://acortar.link/ICNcWP`, 2024. Accessed: 2024-06-05.

[2] T. Andre. Fine-tune a mistral 7b model with direct preference optimization, 2023. Accessed: 2024-06-05.

[3] S. Chaudhari, P. Aggarwal, V. Murahari, T. Rajpurohit, A. Kalyan, K. Narasimhan, A. Deshpande, and B. C. da Silva. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. `https://api.semanticscholar.org/CorpusID:269137670`, 04 2024.

[4] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information*

Processing Systems, volume 30. Curran Associates, Inc., 2017.

[5] HuggingFaceH4. ultrafeedback_binarized, 2023. Accessed: 2024-06-05.

[6] A. Karpathy. Training language models with human feedback: An introduction. https://www.youtube.com/watch?v=crXjRYbvT1U, 2023. Accessed: 2024-06-05.

[7] V. Kumar. Nlp projects nhv. Accessed: 2024-06-05.

[8] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. `https://arxiv.org/abs/2305.18290`, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[10] TheBloke. Openhermes-2-mistral-7b-gptq, 2023. Accessed: 2024-06-05.

[11] L. Weng. Controllable neural text generation. *lilianweng.github.io*, Jan 2021.