

Human Feedback for LLMs

Santiago Botero Daza
David José Daza Jaimes
Mateo Sebastian Ortiz Higuera
David Leonardo Ortiz Uribe

Facultad de Ciencias-Departamento de Matemáticas
Universidad Nacional de Colombia
Matemáticas para el Aprendizaje de Máquinas

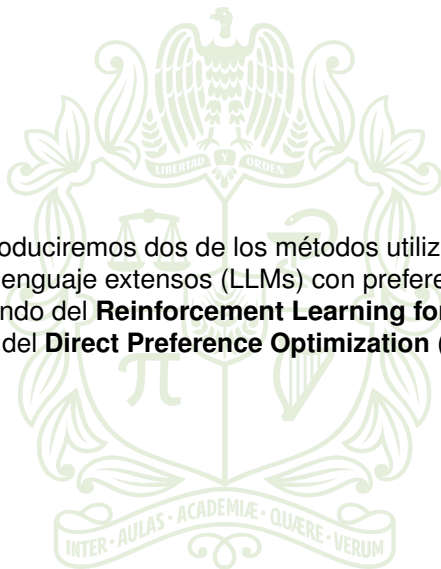
2024-I



MATEMÁTICAS

Introducción

En esta presentación introduciremos dos de los métodos utilizados para ajustar modelos de lenguaje extensos (LLMs) con preferencias humanas, estamos hablando del **Reinforcement Learning for Human Feedback** (RLHF) y del **Direct Preference Optimization** (DPO).

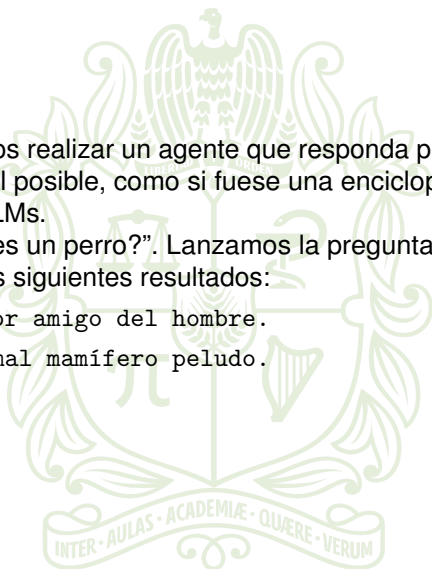


Introducción II

Supongamos que queremos realizar un agente que responda preguntas de la forma más formal posible, como si fuese una enciclopedia, entonces diseñamos un LLMs.

Recibe el prompt: “¿Qué es un perro?”. Lanzamos la pregunta dos veces y obtenemos los dos siguientes resultados:

- 1 El perro es el mejor amigo del hombre.
- 2 El perro es un animal mamífero peludo.



Introducción II

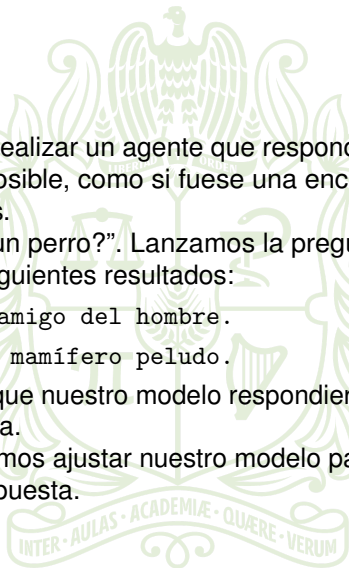
Supongamos que queremos realizar un agente que responda preguntas de la forma más formal posible, como si fuese una enciclopedia, entonces diseñamos un LLMs.

Recibe el prompt: “¿Qué es un perro?”. Lanzamos la pregunta dos veces y obtenemos los dos siguientes resultados:

- 1 El perro es el mejor amigo del hombre.
- 2 El perro es un animal mamífero peludo.

En este caso nos interesaría que nuestro modelo respondiera siempre como en la segunda respuesta.

Utilizando RLHF o DPO podemos ajustar nuestro modelo para que se premie más a la segunda respuesta.



Aprendizaje por Refuerzo (RL)

El aprendizaje por refuerzo (*reinforcement learning*) es un área del Machine Learning en donde se busca entrenar un agente a través de recompensas recibidas por realizar interacciones en un ambiente.

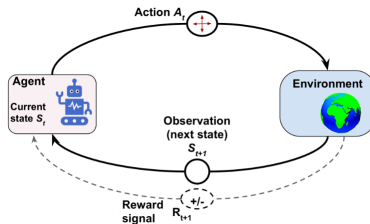


Figura: Esquema del método RL

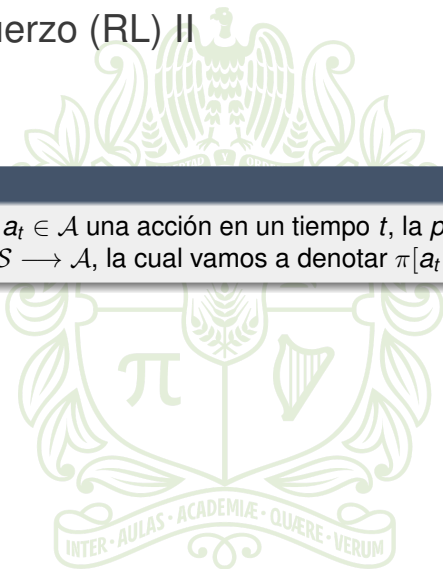


Aprendizaje por Refuerzo (RL) II

Formalmente tenemos

Definición

Dados $s_t \in \mathcal{S}$ un estado y $a_t \in \mathcal{A}$ una acción en un tiempo t , la *política* se define como sigue $\pi : \mathcal{S} \rightarrow \mathcal{A}$, la cual vamos a denotar $\pi[a_t|s_t]$.



Aprendizaje por Refuerzo (RL) II

Formalmente tenemos

Definición

Dados $s_t \in \mathcal{S}$ un estado y $a_t \in \mathcal{A}$ una acción en un tiempo t , la *política* se define como sigue $\pi : \mathcal{S} \rightarrow \mathcal{A}$, la cual vamos a denotar $\pi[a_t|s_t]$.

Definición

La secuencia de estados y acciones

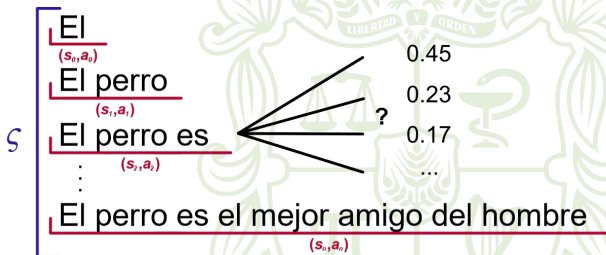
$$\varsigma = ((s_0, a_0), (s_1, a_1), (s_2, a_2), \dots, (s_{n-1}, a_{n-1})) \in (\mathcal{S} \times \mathcal{A})^n$$

es llamado *segmento* de una trayectoria τ .



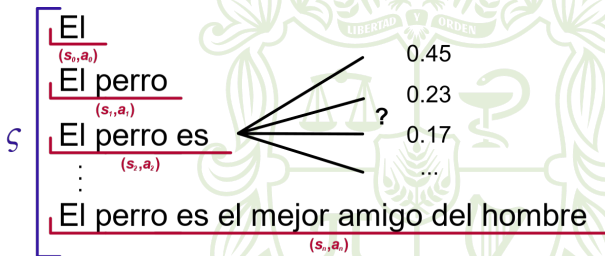
Ejemplo

En un LLMs, dado el prompt “¿Qué es un perro?”.



Ejemplo

En un LLMs, dado el prompt “¿Qué es un perro?”.



Nota

El objetivo del RL es optimizar ς seleccionando políticas que maximicen la recompensa.



Ajustando la Política

Esto que hemos visto es un *Proceso de Decisión Markoviano*, es decir, la probabilidad de la selección de una acción está dado por el estado inmediatamente anterior en la que se encontraba el agente.

Definición

Dado una distribución de probabilidad $P(r_{t+1}|s_t)$ sobre las posibles recompensas r_{t+1} que ocurren en el siguiente estado, recompensa de una trayectoria τ , esta dada por

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

en donde $\gamma \in (0, 1]$ es llamado factor de descuento.



Ajustando la Política II

Definición

La probabilidad de ocurrencia de una trayectoria τ dada una política parametrizada $\pi[a_t|s_t, \theta]$ se define como sigue

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

Con esto ya podemos describir formalmente nuestro problema de ajuste.

INTER · AULAS · ACADEMIÆ · QUÆRE · VERUM



MATEMÁTICAS

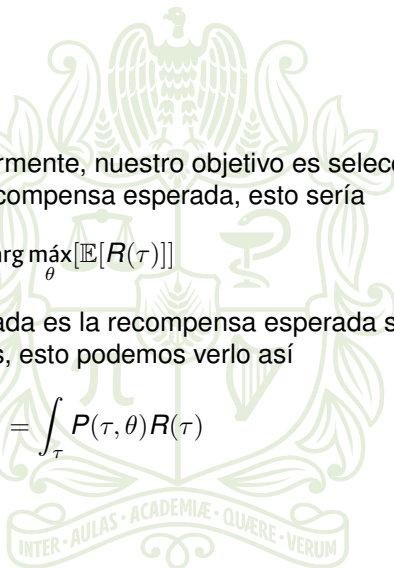
Ajustando la Política III

Como ya se ha dicho anteriormente, nuestro objetivo es seleccionar políticas que maximicen la recompensa esperada, esto sería

$$\theta = \arg \max_{\theta} [\mathbb{E}[R(\tau)]]$$

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

$$\mathbb{E}[R(\tau)] = \int_{\tau} P(\tau, \theta) R(\tau)$$



Policy Gradient Optimization

Como ya se ha dicho anteriormente, nuestro objetivo es seleccionar políticas que maximicen la recompensa esperada, esto sería

$$\theta = \arg \max_{\theta} [\mathbb{E}[R(\tau)]]$$

Utilizando el gradiente en descenso estocástico podemos ajustar los parámetros θ para maximizar la recompensa esperada sobre todas las posibles trayectorias, esto es

$$\begin{aligned}\theta_{k+1} &\leftarrow \theta_k + \alpha \nabla_{\theta} \mathbb{E}[R(\tau)] \\ &\leftarrow \theta_k + \alpha \nabla_{\theta} \int P(\tau, \theta) R(\tau) d\tau\end{aligned}$$



Policy Gradient Optimization II

$$\begin{aligned}\nabla_{\theta} \mathbb{E}[R(\tau)] &= \nabla_{\theta} \int P(\tau, \theta) R(\tau) d\tau \\ &= \int \nabla_{\theta} P(\tau, \theta) R(\tau) d\tau \\ &= \int P(\tau, \theta) \nabla_{\theta} \log P(\tau, \theta) R(\tau) d\tau \quad (\text{Regla de la cadena}) \\ &= \mathbb{E}[\nabla_{\theta} \log P(\tau, \theta) R(\tau)]\end{aligned}$$

Ahora bien, evaluemos $\nabla_{\theta} \log P(\tau, \theta)$. Recordemos

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



Policy Gradient Optimization III

$$\begin{aligned}\nabla_{\theta} \log P(\tau, \theta) &= \nabla_{\theta} \log P(s_0) + \sum_{t=0}^T \nabla_{\theta} \log P(s_{t+1} | s_t, a_t) + \nabla_{\theta} \log \pi[a_t | s_t, \theta] \\ &= \sum_{t=0}^T \nabla_{\theta} \log \pi[a_t | s_t, \theta]\end{aligned}$$

De esta forma tenemos

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



Policy Gradient Optimization III

$$\begin{aligned}\nabla_{\theta} \log P(\tau, \theta) &= \nabla_{\theta} \log P(s_0) + \sum_{t=0}^T \nabla_{\theta} \log P(s_{t+1} | s_t, a_t) + \nabla_{\theta} \log \pi[a_t | s_t, \theta] \\ &= \sum_{t=0}^T \nabla_{\theta} \log \pi[a_t | s_t, \theta]\end{aligned}$$

De esta forma tenemos

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

Nota

Sin embargo, realizar gradiente en descenso sobre todas las posibles trayectorias significa un gran costo computacional.

Policy Gradient Optimization IV

Nota

Sin embargo, realizar gradiente en descenso sobre todas las posibles trayectorias significa un gran costo computacional.

Una forma para solucionar esto es aproximando la ecuación tomando un conjunto \mathcal{D} de trayectorias. Así tenemos la siguiente fórmula

$$\nabla_{\theta} \mathbb{E}[R(\tau)] \approx \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \nabla_{\theta} \log \pi[\mathbf{a}_t | \mathbf{s}_t, \theta] R(\tau)$$



Policy Gradient Optimization IV

Nota

Sin embargo, realizar gradiente en descenso sobre todas las posibles trayectorias significa un gran costo computacional.

Una forma para solucionar esto es aproximando la ecuación tomando un conjunto \mathcal{D} de trayectorias. Así tenemos la siguiente fórmula

$$\nabla_{\theta} \mathbb{E}[R(\tau)] \approx \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \nabla_{\theta} \log \pi[a_t | s_t, \theta] R(\tau)$$

Advantage Function

Otro de los problemas generados con respecto a la muestra de trayectorias tiene que ver con la elevada varianza que podemos obtener para ajustar el modelo. Con esta función podemos predecir qué tan bueno es elegir una acción a en un estado s .

Aprendizaje por Refuerzo con Retroalimentación Humana (RLHF)

En este primer método entrenamos un modelo de recompensa el cual nos permite evaluar el nivel de preferencia de una respuesta del modelo inicial.

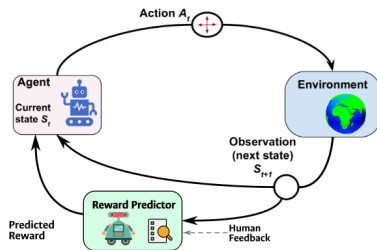


Figura: Esquema del método RLHF



Aprendizaje por Refuerzo con Retroalimentación Humana II

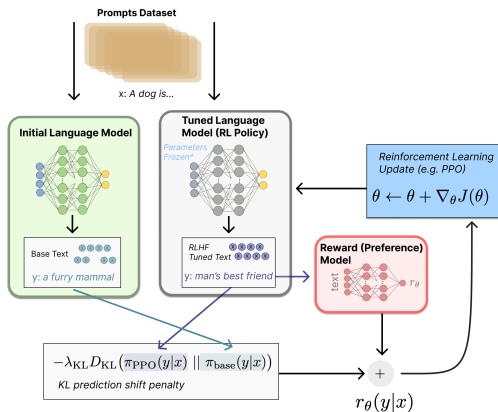


Figura: Esquema completo del método RLHF



Ajustando el Modelo

La forma más fácil de evaluar resultados es comparando pares. Formalmente tenemos

Definición

Dados dos segmentos ς^1 y ς^2 , diremos que un humano prefiere el primero sobre el segundo con la siguiente notación $\varsigma^1 \succ \varsigma^2$.

Así bien, podemos definir una función de recompensa

Definición

Dados \mathcal{S} , \mathcal{A} un conjunto de estados y un conjunto de acciones respectivamente, y Γ un alfabeto donde $x \in \Gamma^*$ es una pregunta o un prompt, la función

$$r : \Gamma^* \times (\mathcal{S} \times \mathcal{A})^n \longrightarrow \mathbb{R}$$

es llamada una función de recompensa.



Ajustando el Modelo II

Ahora bien, el modelo probabilístico de Bradley-Terry nos permite saber la probabilidad de que un humano prefiera un segmento sobre otro. Tendríamos lo siguiente

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

Con lo anterior podemos deducir la función para ajustar el modelo utilizando máxima verosimilitud logarítmica (log-likelihood).



Ajustando el Modelo III

$$\begin{aligned}
 P(\varsigma^1 \succ \varsigma^2, x) &= \frac{\exp(r_\phi(x, \varsigma^1))}{\exp(r_\phi(x, \varsigma^1)) + \exp(r_\phi(x, \varsigma^2))} \\
 &= \frac{\frac{\exp r_\phi(x, \varsigma^1)}{\exp r_\phi(x, \varsigma^1)}}{\frac{\exp r_\phi(x, \varsigma^1) + \exp r_\phi(x, \varsigma^2)}{\exp r_\phi(x, \varsigma^1)}} \\
 &= \frac{1}{1 + \frac{\exp r_\phi(x, \varsigma^2)}{\exp r_\phi(x, \varsigma^1)}} \\
 &= \frac{1}{1 + \exp(r_\phi(x, \varsigma^2) - r_\phi(x, \varsigma^1))} \\
 &= \sigma(r_\phi(x, \varsigma^1) - r_\phi(x, \varsigma^2))
 \end{aligned}$$

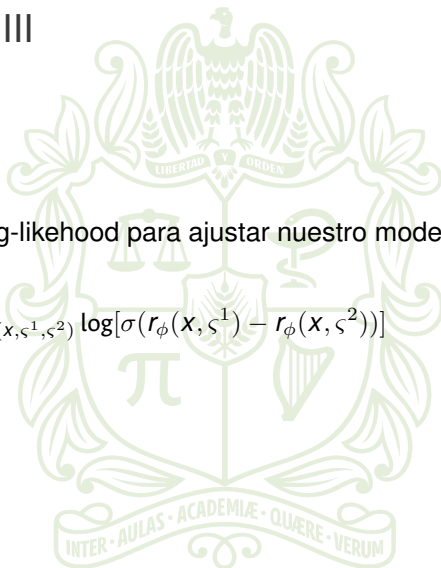
donde σ es la función sigmoide.



Ajustando el Modelo III

Finalmente, utilizamos log-likelihood para ajustar nuestro modelo, así tenemos

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

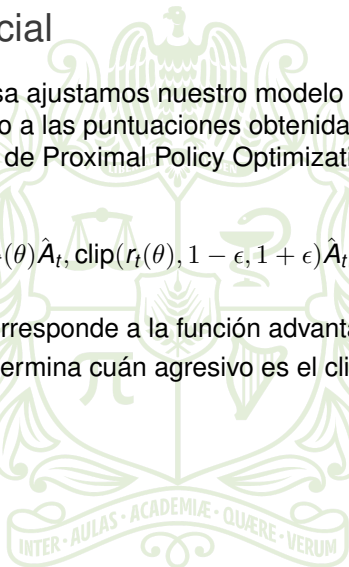


Ajustando el Modelo Inicial

Con el modelo de recompensa ajustamos nuestro modelo inicial (o incluso uno nuevo) de acuerdo a las puntuaciones obtenidas. Podemos utilizar el algoritmo de Proximal Policy Optimization (PPO) para ello.

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

Donde $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, \hat{A}_t corresponde a la función advantage [13], ϵ es un hiperparámetro que determina cuán agresivo es el clipping.



Ajustando el Modelo Inicial

Con el modelo de recompensa ajustamos nuestro modelo inicial (o incluso uno nuevo) de acuerdo a las puntuaciones obtenidas. Podemos utilizar el algoritmo de Proximal Policy Optimization (PPO) para ello.

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

Donde $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, \hat{A}_t corresponde a la función advantage [13], ϵ es un hiperparámetro que determina cuán agresivo es el clipping.

Nota

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.



Divergencia KL

Nota

El algoritmo de PPO podría verse como una versión de la función de Divergencia KL, la cual es una métrica para dos distribuciones de probabilidad.

Sin embargo, en nuestro contexto de RL, también puede ser vista como regularizador para penalizar desviaciones grandes entre la política actual y políticas anteriores.

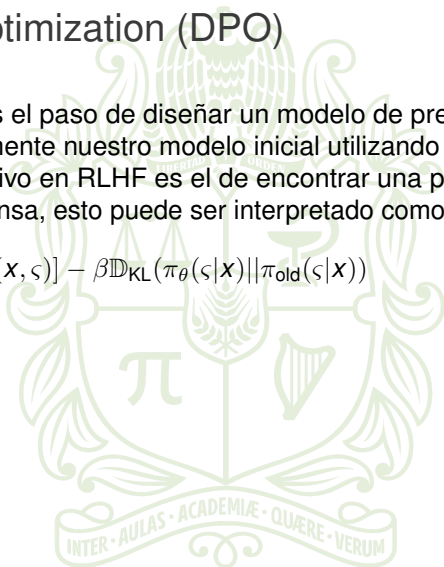
$$L_{\text{regularizada}} = L_{\text{original}} - \beta \mathbb{D}_{KL}(\pi_{\theta_{\text{old}}} || \pi_{\theta})$$



Direct Preference Optimization (DPO)

En este método omitimos el paso de diseñar un modelo de preferencias y ajustamos directamente nuestro modelo inicial utilizando RL. Recordemos que el objetivo en RLHF es el de encontrar una política que maximice la recompensa, esto puede ser interpretado como sigue

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



Direct Preference Optimization (DPO)

En este método omitimos el paso de diseñar un modelo de preferencias y ajustamos directamente nuestro modelo inicial utilizando RL. Recordemos que el objetivo en RLHF es el de encontrar una política que maximice la recompensa, esto puede ser interpretado como sigue

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

La solución a dicha ecuación es presentada en [1], la cual viene dada como sigue

$$\pi_r(\varsigma|x) = \frac{1}{Z(x)} \pi_{\text{old}}(\varsigma|x) \exp\left(\frac{1}{\beta} r(x, \varsigma)\right) \quad (1)$$

donde $Z(x) = \sum_{\varsigma} \pi_{\text{old}}(\varsigma|x) \exp\left(\frac{1}{\beta} r(x, \varsigma)\right)$.



Direct Preference Optimization (DPO) II

Nota

Calcular la función $Z(x)$ de la ecuación 1, es muy costoso computacionalmente.

Si suponemos que tenemos la política óptima π_r , podemos deducir una nueva función

$$\pi_r(\varsigma|x) = \frac{1}{Z(x)} \pi_{\text{old}}(\varsigma|x) \exp\left(\frac{1}{\beta} r(x, \varsigma)\right)$$

$$Z(x) \pi_r(\varsigma|x) = \pi_{\text{old}}(\varsigma|x) \exp\left(\frac{1}{\beta} r(x, \varsigma)\right)$$

$$\frac{Z(x) \pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)} = \exp\left(\frac{1}{\beta} r(x, \varsigma)\right)$$

$$\log\left(\frac{Z(x) \pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)}\right) = \frac{1}{\beta} r(x, \varsigma)$$



Direct Preference Optimization (DPO) III

$$r(x, \varsigma) = \beta \left(\log Z(x) + \log \frac{\pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)} \right)$$

$$r(x, \varsigma) = \beta \log \frac{\pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)} + \beta \log Z(x)$$



Direct Preference Optimization (DPO) III

$$r(x, \varsigma) = \beta \left(\log Z(x) + \log \frac{\pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)} \right)$$

$$r(x, \varsigma) = \beta \log \frac{\pi_r(\varsigma|x)}{\pi_{\text{old}}(\varsigma|x)} + \beta \log Z(x)$$

Nota

Recordemos el modelo Bradley-Terry de la deducción para ajustar el modelo de recompensa.



Ajustando el Modelo

Del modelo probabilístico de Bradley-Terry y por la observación anterior tenemos

$$\begin{aligned} L_{DPO}(\pi_{\theta}; \pi_{\text{old}}) &= \max_{\theta} -\mathbb{E}_{(x, \zeta^1, \zeta^2)} \log[\sigma(r_{\theta}(x, \zeta^1) - r_{\theta}(x, \zeta^2))] \\ &= \max_{\theta} -\mathbb{E}_{(x, \zeta^1, \zeta^2)} \log \left[\sigma \left(\beta \log \frac{\pi_r(\zeta^1|x)}{\pi_{\text{old}}(\zeta^1|x)} - \beta \log \frac{\pi_r(\zeta^2|x)}{\pi_{\text{old}}(\zeta^2|x)} \right) \right] \end{aligned}$$



Introducción al experimento

- **Código Fuente:** GitHub, modificado en Google Colab
- **Algoritmo:** DPO
- **Modelo:** Hermes Mistral 2-GPTQ (7B parámetros) de Hugging Face
- **Dataset:** "Prompt-Chosen-Rejected", generado por otros LLMs
- **Técnica:** PEFT (Parameter Efficient Fine Tuning)
 - Ajuste de solo una proporción de parámetros
 - Reducción de consumo de memoria (cuantización a 4 bytes)
- **Resultados:**
 - Entrenamiento eficiente y viable en Google Colaboratory
 - Comparación de rendimiento con el modelo original



Introducción al código

■ Principales bibliotecas utilizadas:

- “torch” y “transformers” para manipulación de modelos
- “peft” para ajuste eficiente de parámetros
- “trl” para entrenar con DPO
- “bitsandbytes”, “optimum”, “auto-gpt” para cuantización del modelo

■ Requerimientos:

- GPU T4 de Colab
- Cuenta y autenticación en Hugging Face

■ Proceso del Código:

- 1 Extraer el modelo “OpenHermes-2-Mistral-7B-GPTQ” y el dataset “ultrafeedback_binarized” de Hugging Face
- 2 Crear una copia del modelo como referencia para el algoritmo DPO
- 3 Adaptar los modelos a PEFT, ajustar los argumentos de entrenamiento y entrenar con DPO
- 4 Mostrar métricas como pérdida de entrenamiento y validación
- 5 Hacer una inferencia con el modelo entrenado con DPO y compararla con la del modelo original (sin DPO)



Métricas de Entrenamiento del Modelo

Bajo las siguientes métricas obtenidas del entrenamiento del modelo, se puede evaluar el desempeño cuantitativo del modelo:

- Train loss
- Validation loss
- Reward/chosen
- Reward/rejected
- Reward/margin
- Reward/accuracy

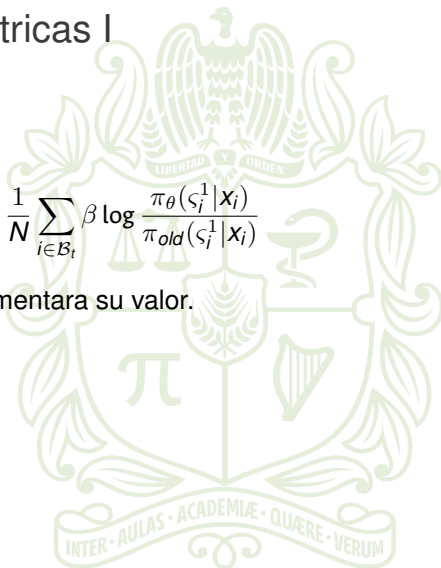


Aclaración de las métricas I

■ Reward/chosen:

$$\frac{1}{N} \sum_{i \in \mathcal{B}_t} \beta \log \frac{\pi_{\theta}(\varsigma_i^1 | \mathbf{x}_i)}{\pi_{old}(\varsigma_i^1 | \mathbf{x}_i)}$$

Se esperaría que aumentara su valor.



Aclaración de las métricas I

■ Reward/chosen:

$$\frac{1}{N} \sum_{i \in \mathcal{B}_t} \beta \log \frac{\pi_{\theta}(\zeta_i^1 | x_i)}{\pi_{old}(\zeta_i^1 | x_i)}$$

Se esperaría que aumentara su valor.

■ Reward/rejected:

$$\frac{1}{N} \sum_{i \in \mathcal{B}_t} \beta \log \frac{\pi_{\theta}(\zeta_i^2 | x_i)}{\pi_{old}(\zeta_i^2 | x_i)}$$

Se esperaría que disminuyera su valor.

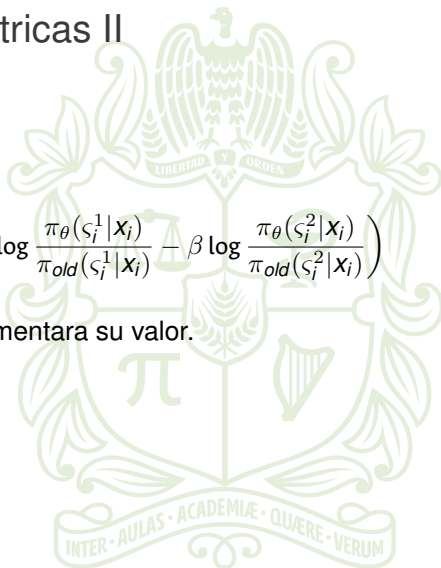


Aclaración de las métricas II

■ Reward/margin:

$$\frac{1}{N} \sum_{i \in \mathcal{B}_t} \left(\beta \log \frac{\pi_{\theta}(\varsigma_i^1 | \mathbf{x}_i)}{\pi_{old}(\varsigma_i^1 | \mathbf{x}_i)} - \beta \log \frac{\pi_{\theta}(\varsigma_i^2 | \mathbf{x}_i)}{\pi_{old}(\varsigma_i^2 | \mathbf{x}_i)} \right)$$

Se esperaría que aumentara su valor.



Aclaración de las métricas II

■ Reward/margin:

$$\frac{1}{N} \sum_{i \in \mathcal{B}_t} \left(\beta \log \frac{\pi_{\theta}(\varsigma_i^1 | \mathbf{x}_i)}{\pi_{old}(\varsigma_i^1 | \mathbf{x}_i)} - \beta \log \frac{\pi_{\theta}(\varsigma_i^2 | \mathbf{x}_i)}{\pi_{old}(\varsigma_i^2 | \mathbf{x}_i)} \right)$$

Se esperaría que aumentara su valor.

■ Reward/accuracy:

- Promedio de veces que reward/chosen es mayor que reward/rejected.
- Se esperaría que aumentara su valor.



Análisis de las métricas obtenidas

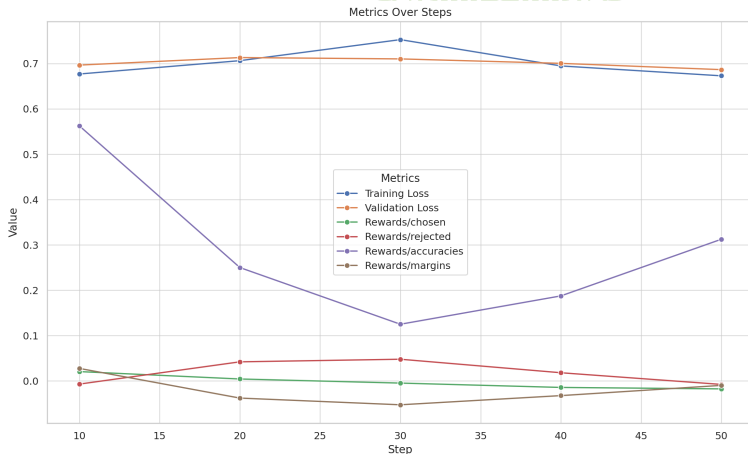


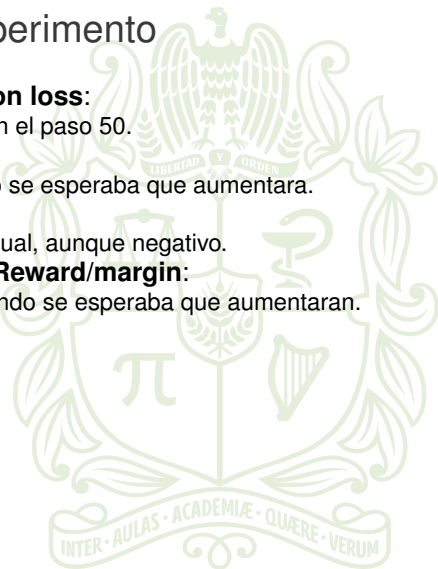
Figura: Métricas derivadas del entrenamiento del modelo



Conclusiones del Experimento

Métricas Cuantitativas:

- **Train loss y Validation loss:**
 - Ligera reducción en el paso 50.
- **Reward/chosen:**
 - Disminuyó, cuando se esperaba que aumentara.
- **Reward/rejected:**
 - Se mantuvo casi igual, aunque negativo.
- **Reward/accuracy y Reward/margin:**
 - Disminuyeron, cuando se esperaba que aumentarán.



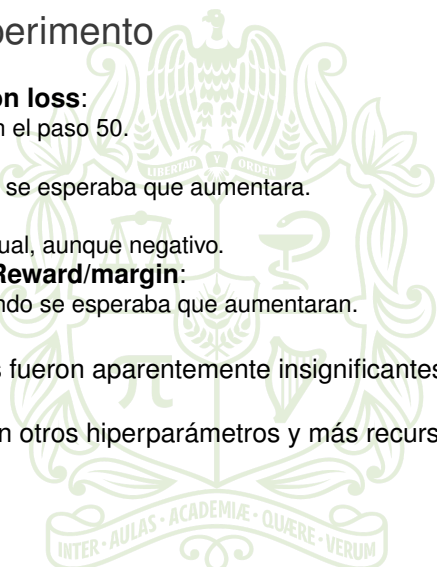
Conclusiones del Experimento

Métricas Cuantitativas:

- **Train loss y Validation loss:**
 - Ligera reducción en el paso 50.
- **Reward/chosen:**
 - Disminuyó, cuando se esperaba que aumentara.
- **Reward/rejected:**
 - Se mantuvo casi igual, aunque negativo.
- **Reward/accuracy y Reward/margin:**
 - Disminuyeron, cuando se esperaba que aumentarían.

Observaciones:

- Cambios cuantitativos fueron aparentemente insignificantes o contraproducentes.
- Es posible mejorar con otros hiperparámetros y más recursos de memoria.



Conclusiones del Experimento

Métricas Cuantitativas:

- **Train loss y Validation loss:**
 - Ligera reducción en el paso 50.
- **Reward/chosen:**
 - Disminuyó, cuando se esperaba que aumentara.
- **Reward/rejected:**
 - Se mantuvo casi igual, aunque negativo.
- **Reward/accuracy y Reward/margin:**
 - Disminuyeron, cuando se esperaba que aumentarían.

Observaciones:

- Cambios cuantitativos fueron aparentemente insignificantes o contraproducentes.
- Es posible mejorar con otros hiperparámetros y más recursos de memoria.

Conclusión Cualitativa:

- Cambio cualitativo significativo en la respuesta generada por el modelo entrenado con DPO.
- Sugiere posibles mejoras cualitativas importantes a pesar de cambios mínimos o insignificativos en métricas cuantitativas.

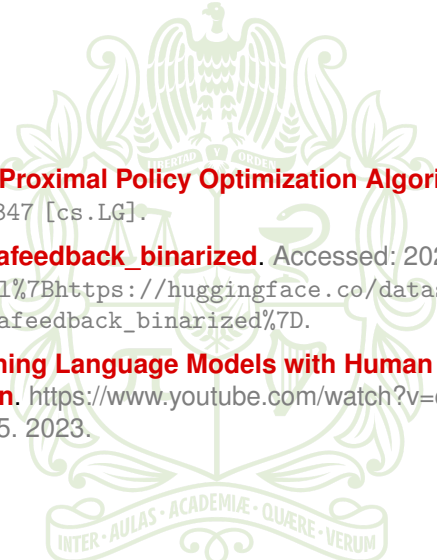


Referencias I

- [1] Rafael Rafailov et al. **Direct Preference Optimization: Your Language Model is Secretly a Reward Model**. <https://arxiv.org/abs/2305.18290>. 2023. arXiv: 2305.18290 [cs.LG].
- [2] Lilian Weng. «Controllable Neural Text Generation.». En: **lilian-weng.github.io** (ene. de 2021). URL: <https://lilianweng.github.io/posts/2021-01-02-controllable-text-generation/>.
- [3] Shreyas Chaudhari et al. **RLHF Deciphered: A Critical Analysis of Reinforcement Learning from Human Feedback for LLMs**. <https://api.semanticscholar.org/CorpusID:269137670>. Abr. de 2024.
- [4] Paul F Christiano et al. «Deep Reinforcement Learning from Human Preferences». En: **Advances in Neural Information Processing Systems**. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.



Referencias II

- 
- [5] John Schulman et al. **Proximal Policy Optimization Algorithms**. 2017. arXiv: 1707.06347 [cs.LG].
- [6] HuggingFaceH4. **ultrafeedback_binarized**. Accessed: 2024-06-05. 2023. URL: %5Curl%7Bhttps://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized%7D.
- [7] Andrej Karpathy. **Training Language Models with Human Feedback: An Introduction**. <https://www.youtube.com/watch?v=crXjRYbvT1U>. Accessed: 2024-06-05. 2023.

