IIC3670 Procesamiento de Lenguaje Natural

https://github.com/marcelomendoza/IIC3670

Generación condicional con atención

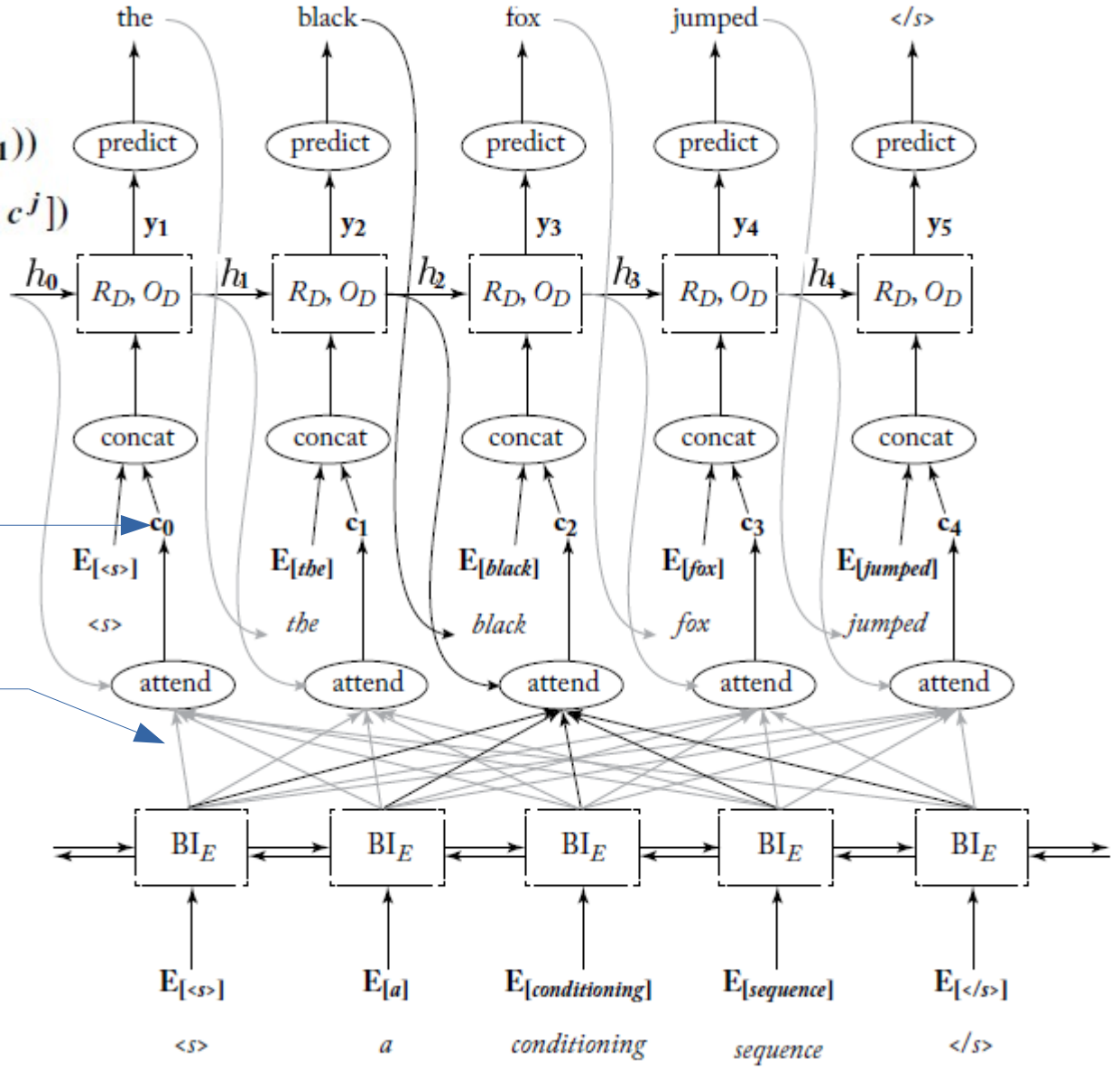
$$p(t_{j+1} = k \mid \hat{t}_{1:j}, x_{1:n}) = f(O_{\text{dec}}(h_{j+1}))$$

$$h_{j+1} = R_{\text{dec}}(h_j, [\hat{t}_j; c^j])$$

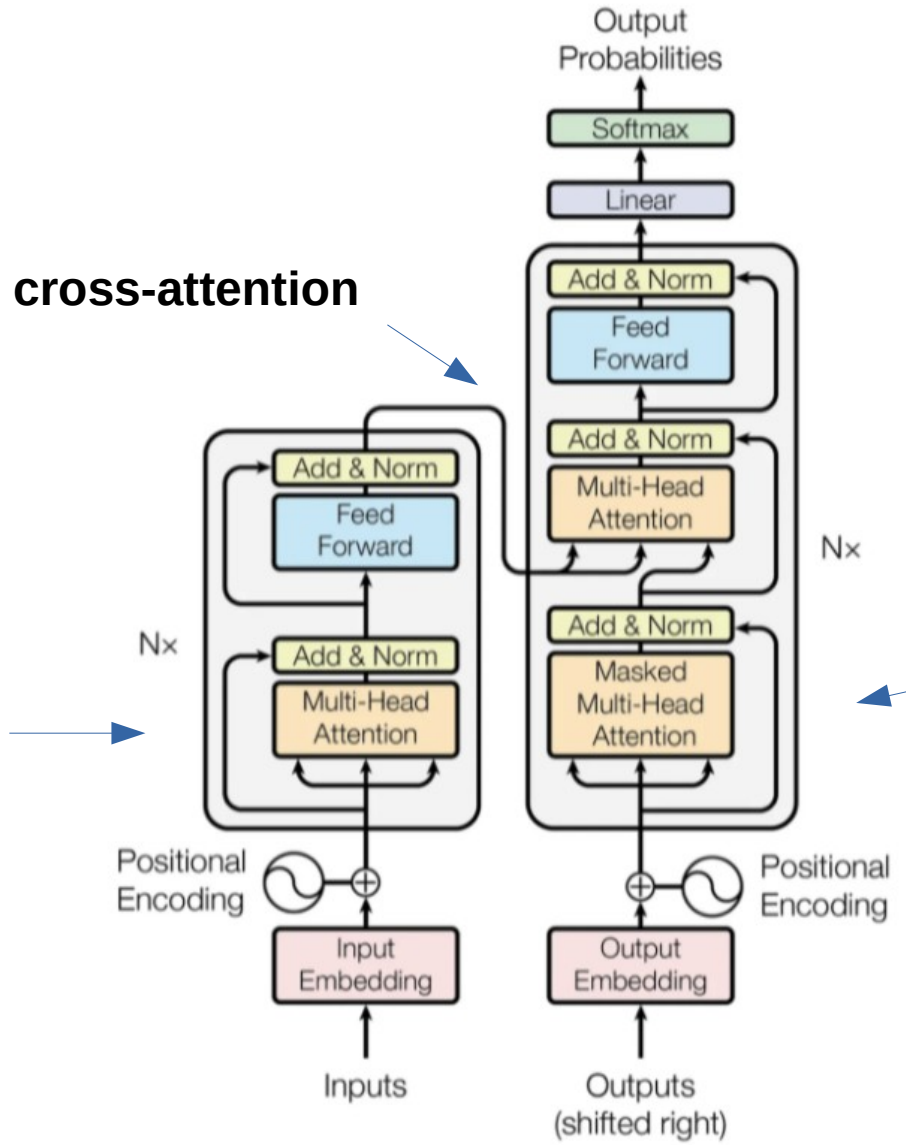
$$c^j = \sum_{i=1}^{n} \alpha^j_{[i]} \cdot c_i$$

$$\bar{\alpha}^j_{[i]} = \text{MLP}^{\text{att}}(h_j; c_i])$$

$$c_{1:n} = \text{biRNN}^{\star}_{\text{enc}}(x_{1:n})$$



Generación condicional con atención (transformer seq2seq)



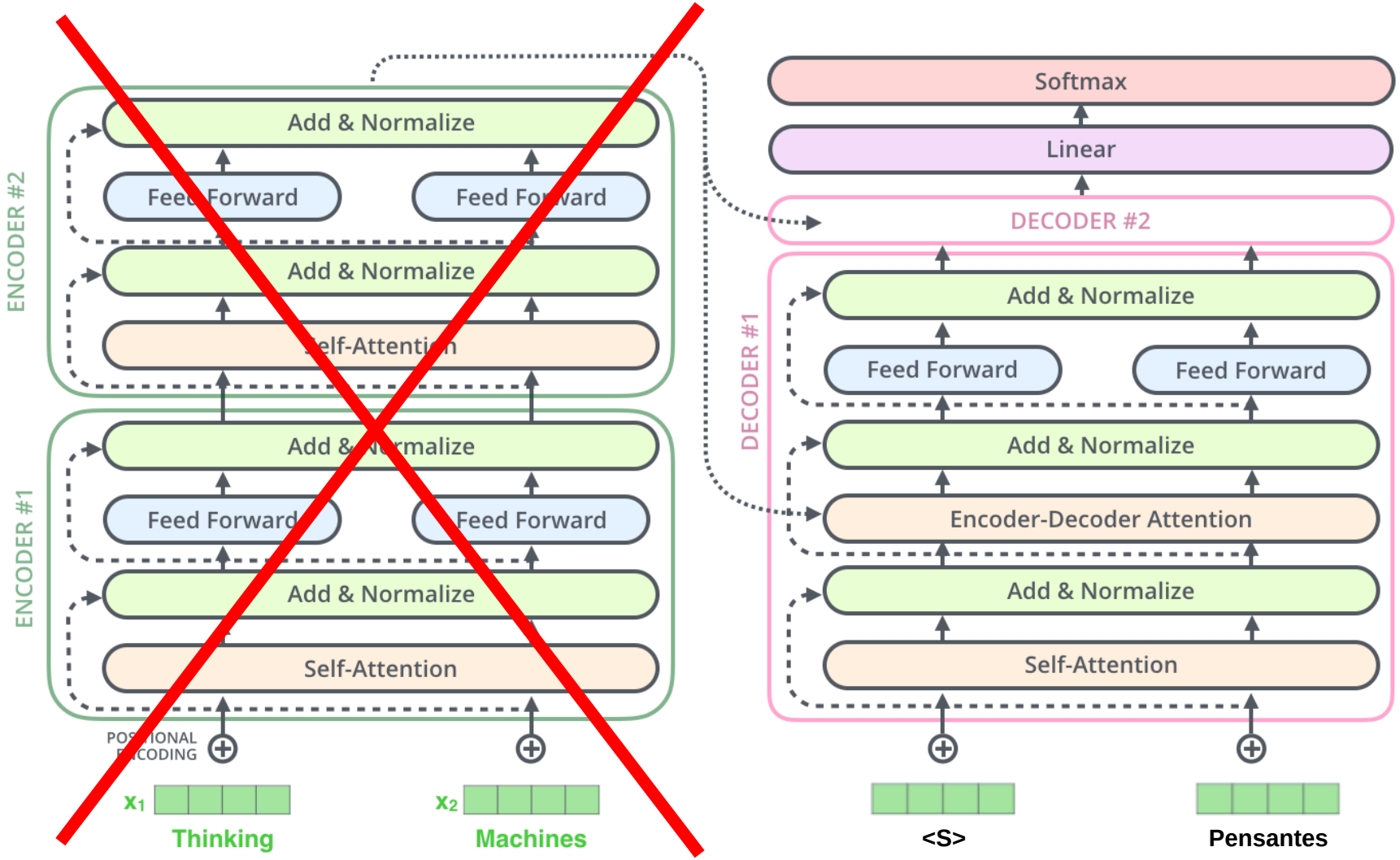
cross-attention

global attention

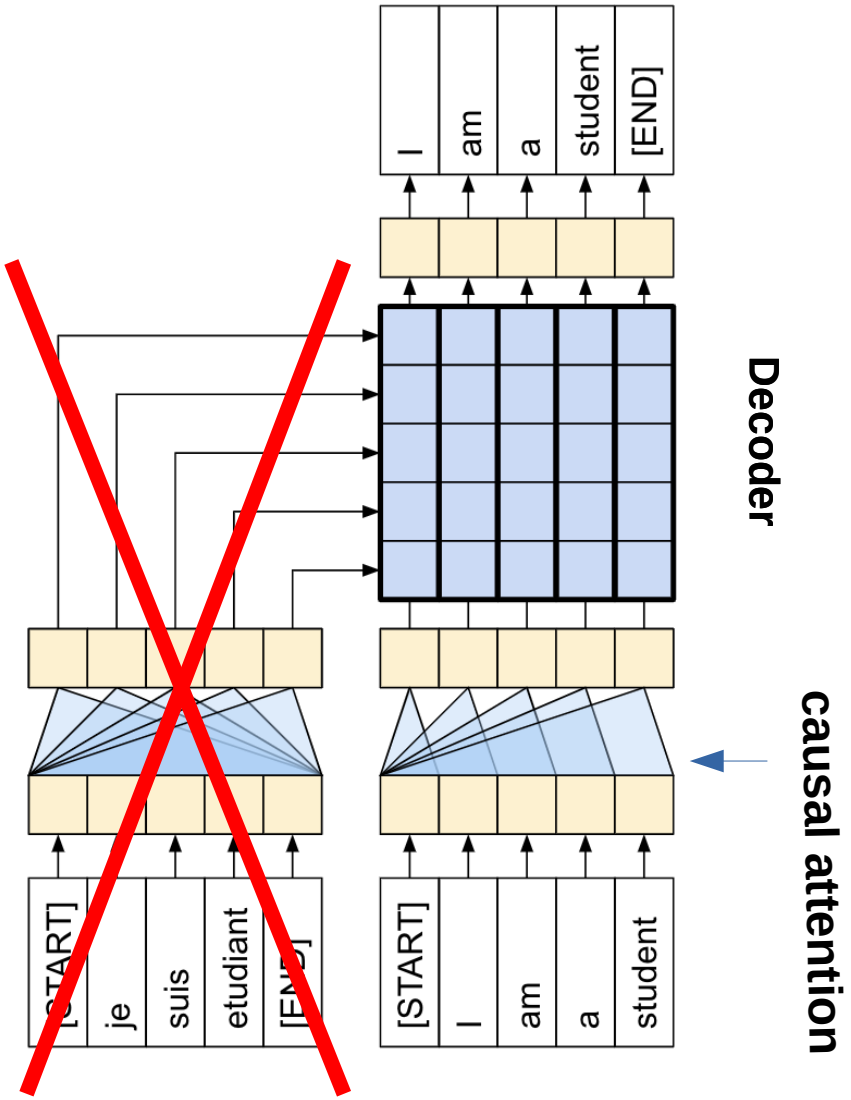
causal attention

- GENERATIVE PRETRAINED TRANSFORMERS -

Se elimina el encoder!!!



Se elimina el encoder!!!



Transformer decoder

Seq2seq

- Cada secuencia $(m^1, ..., m^n) \mapsto (y^1, ..., y^\eta)$

se transforma en:

Símbolo separador

$$(w^1, ..., w^{n+\eta+1}) = (m^1, ..., m^n, \delta, y^1, ..., y^\eta)$$

- Luego, el **decoder** resuelve la siguiente tarea:

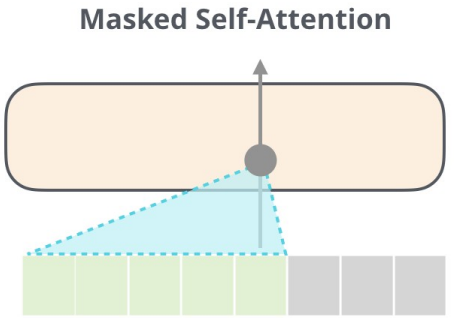
secuencias largas

$$p(w^1, ..., w^{n+\eta}) = \prod_{j=1}^{n+\eta} p(w^i | w^1, ..., w^{j-1})$$

Transformer decoder

Las oraciones van a ser más largas

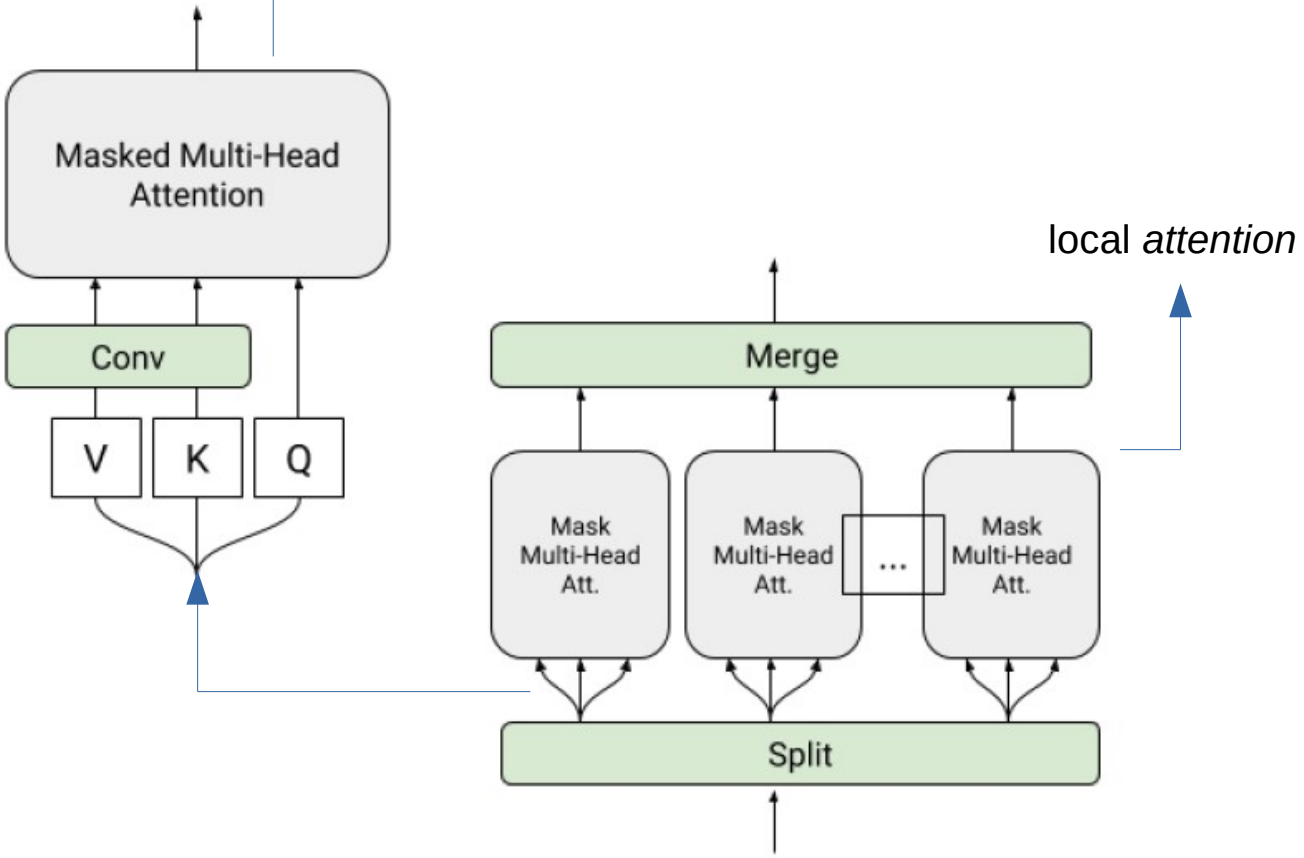
Modelo **abstractivo**



Causal LM

Memory-compressed attention

local *attention*



Reduce las V y K usando kernels de 3x3

Divide la oración en *batches* del mismo tamaño

Transformer decoder

LLM

Mecanismo auto-regresivo

tokens

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$$U = (u_{-k}, \ldots, u_{-1})$$ **contexto de u**

Token embedding matrix

$$h_0 = UW_e + W_p$$ Position embedding matrix

$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

#_layers

task $$P(u) = \texttt{softmax}(h_n W_e^T)$$

Idea propuesta en:

GENERATING WIKIPEDIA BY SUMMARIZING LONG SEQUENCES
Peter J. Liu, Mohammad Saleh,
Etienne Poty, Ben Goodrich, Ryan Sepassi, Łukasz Kaiser, Noam Shazeer
ICLR 2018

GPT 1

Transformer decoder + supervised fine tuning

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Transformer decoder

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

Supervised fine-tuning

label

Última capa del transformer decoder

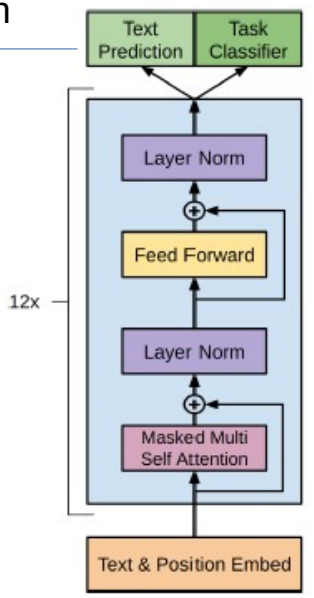
en LM *y* es un token

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

La pérdida combinada es:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

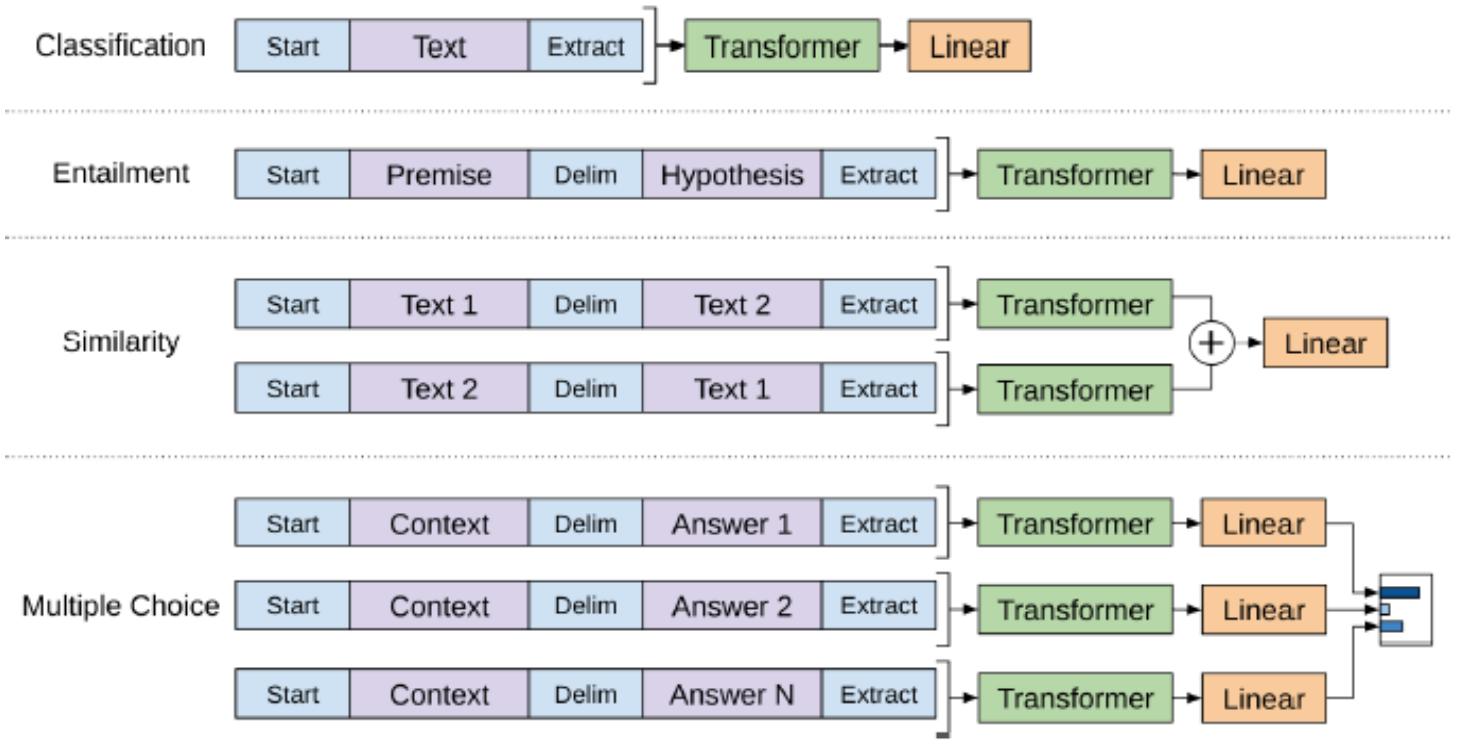
GPT 1 resuelve ambas tareas simultáneamente (LM y specific task)



GPT 1

Supervised fine-tuning

La secuencia de entrada depende de la tarea



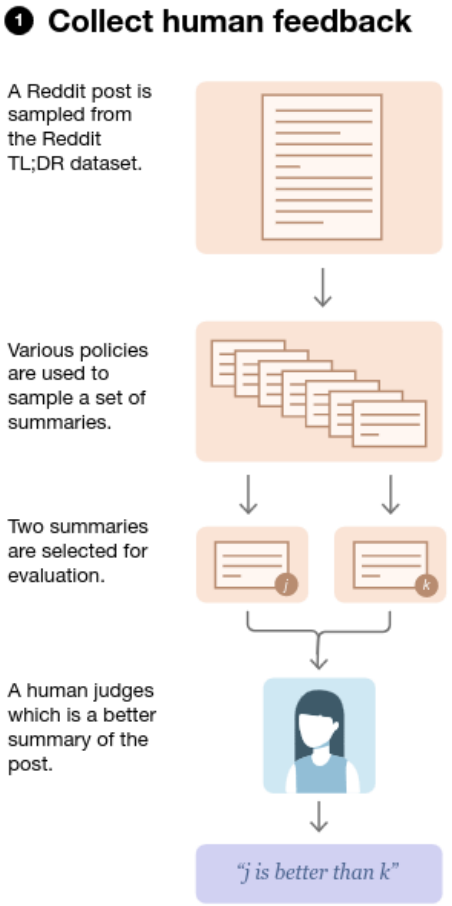
GPT 1

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. Improving language understanding by generative pre-training, 2018.

- INSTRUCT GPT -

Human in the loop

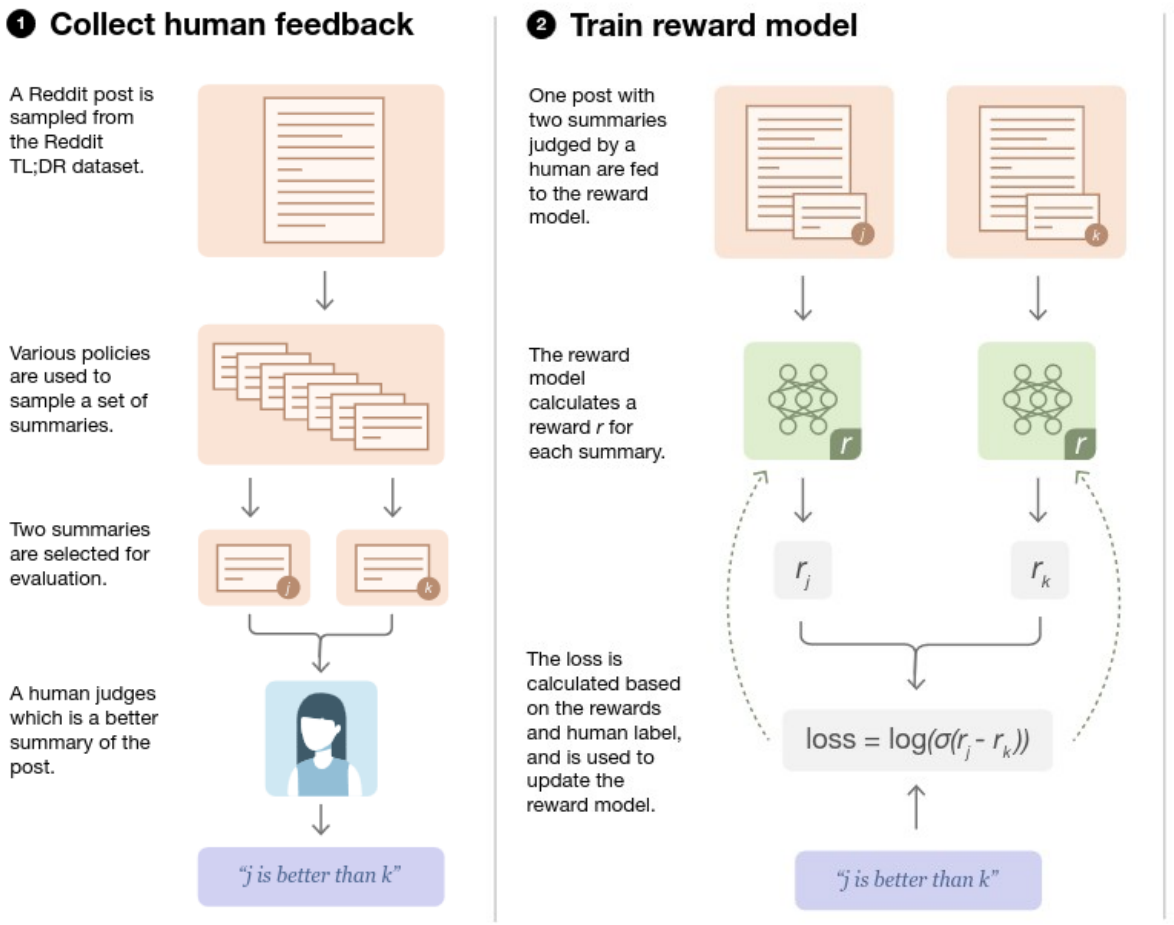
Inicialmente este tema se abordó para construcción de resúmenes



Stiennon et al. Learning to summarize from human feedback, NeurIPS 2020

Human in the loop

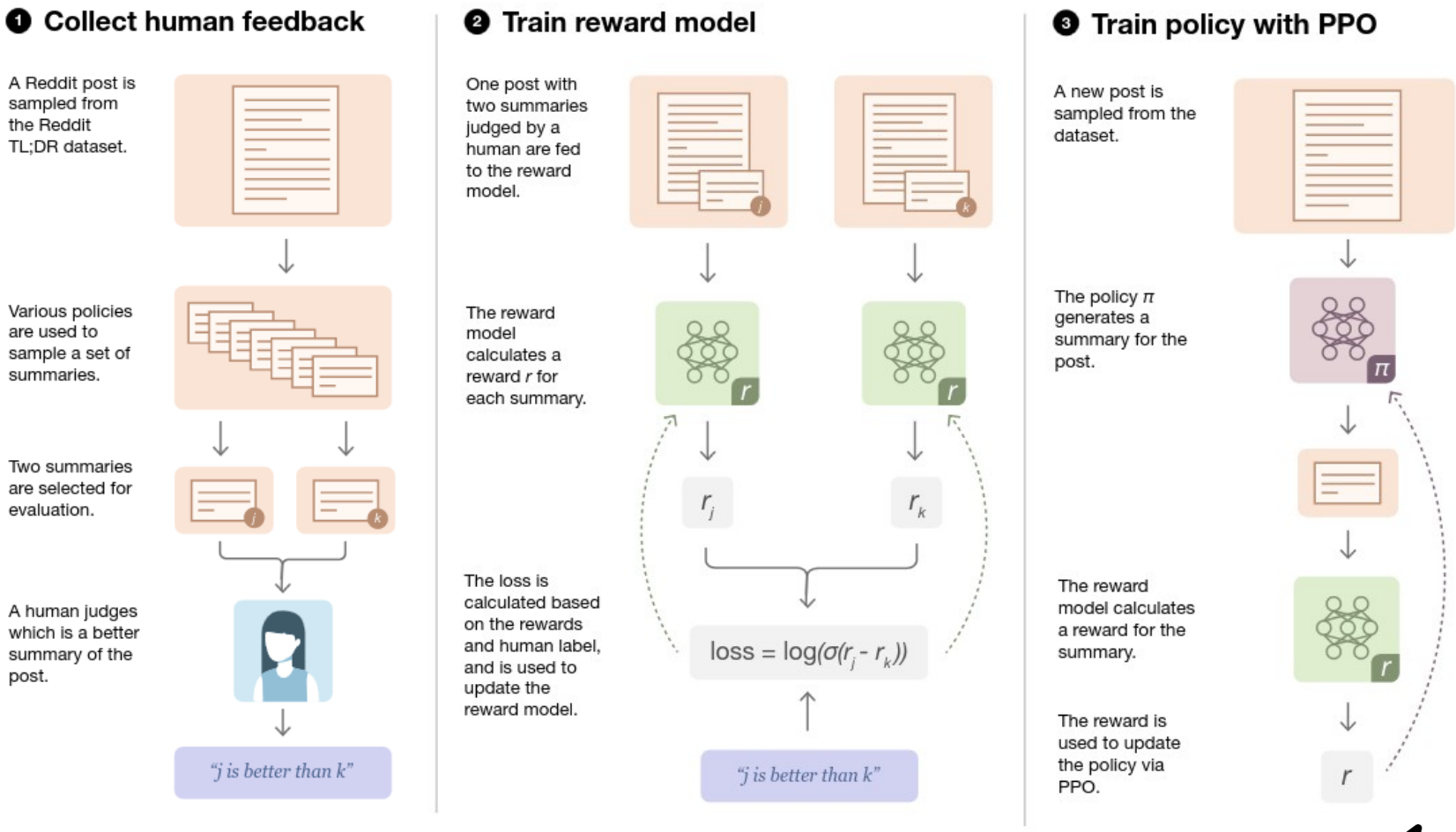
Inicialmente este tema se abordó para construcción de resúmenes



Stiennon et al. Learning to summarize from human feedback, NeurIPS 2020

Human in the loop

Inicialmente este tema se abordó para construcción de resúmenes



Stiennon et al. Learning to summarize from human feedback, NeurIPS 2020

Reward Model (RM)

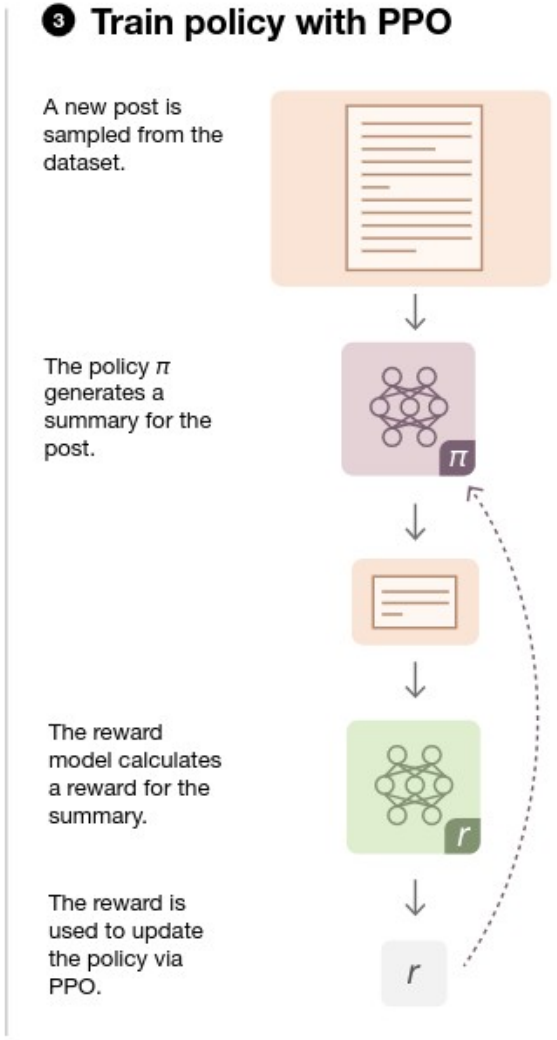
- El RM es un modelo lineal que entrega un escalar (regresión).
- Se entrena el modelo para predecir a partir de resúmenes cuál es mejor:

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}[\log(\sigma(r_\theta(x, y_i) - r_\theta(x, y_{1-i})))]$$

resumen preferido $y \in \{y_0, y_1\}$

- Una vez que termina el entrenamiento, se normaliza el RM para que el reward tenga un score medio en 0.

Proximal Policy Optimization (PPO)



- La salida del RM es un reward para el sistema.

- Se penaliza la divergencia entre el modelo mejorado (RL) y el original SFT para evitar que las salidas de los modelo sean muy distintas a las vistas por el RM durante el entrenamiento.

- El full reward queda dado por:

$$R(x, y) = r_\theta(x, y) - \beta \log[\pi_\phi^{\text{RL}}(y|x)/\pi^{\text{SFT}}(y|x)]$$