FLAR
FONDO LATINOAMERICANO DE RESERVAS

Modelos de lenguaje de gran tamaño

BCRP – CEFA 2025

Objetivos

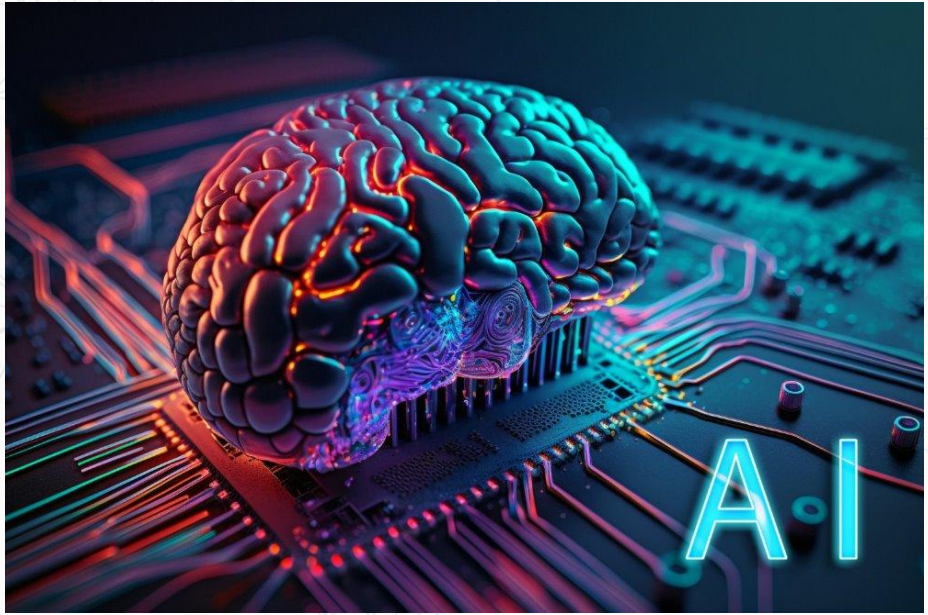
OBJETIVO:
Introducir los modelos de lenguaje de gran tamaño como modelos de Aprendizaje de Máquinas.

OBJETIVOS ESPECÍFICOS:

1. Introducir conceptos de Deep Learning
2. Introducir los Modelos De Lenguaje de Gran Tamaño.
3. Prompt Engineering.

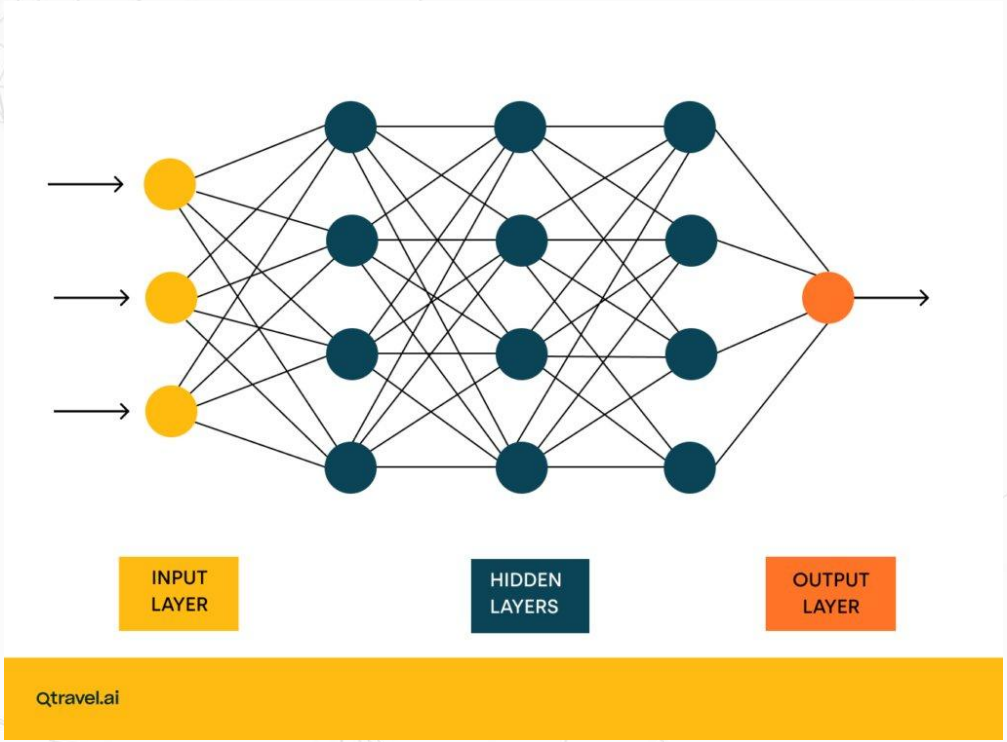
Definición de AI

AI se la puede definir como la capacidad de un programa o máquina para pensar, aprender y tomar acciones sin que explícitamente se le den esas instrucciones. En esta medida los modelos de Inteligencia Artificial pueden ser pensados como computadores que ejecutan tareas de forma autónoma como la ingesta y análisis de grandes volúmenes de información, o para reconocer patrones en los datos (NVIDIA)

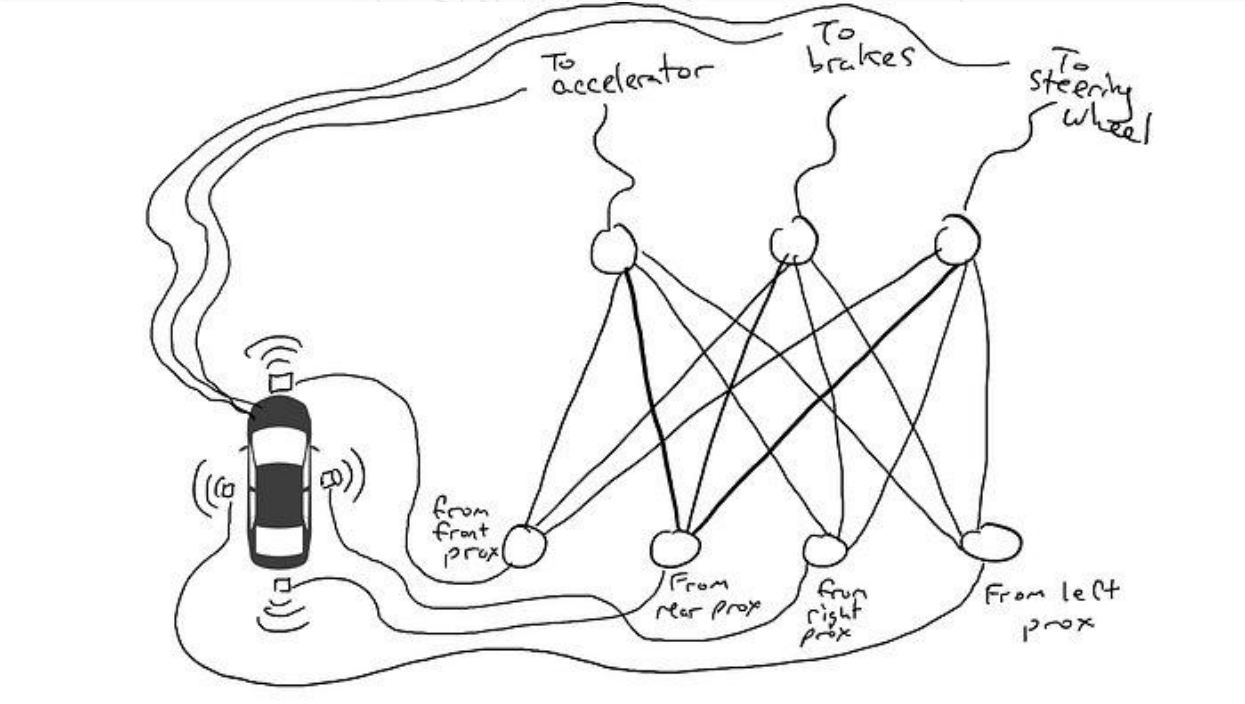


Definición de Deep Learning

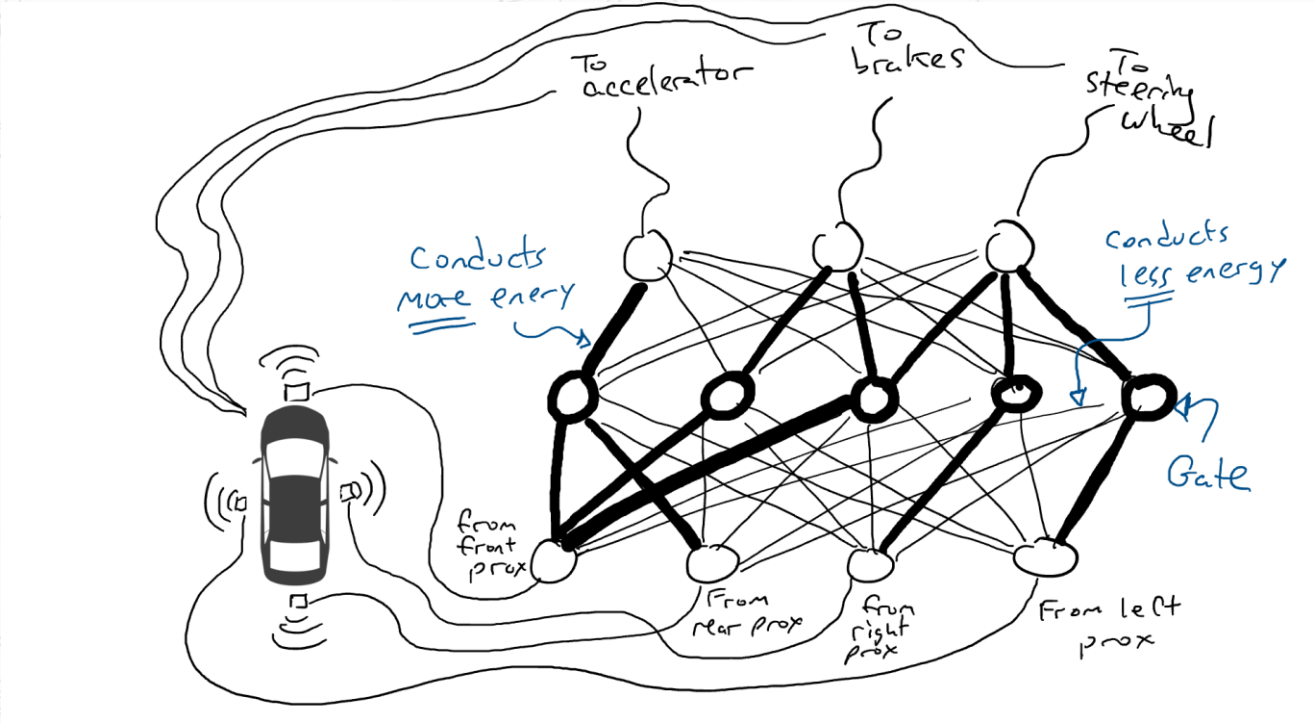
Deep Learning lo podemos entender como una rama de los modelos de Aprendizaje de Máquinas (ML), que se centra en el uso de modelos de redes neuronales para responder a problemas de 'Regresión', 'Clasificación', 'Clusterización' o 'Identificación de patrones'.



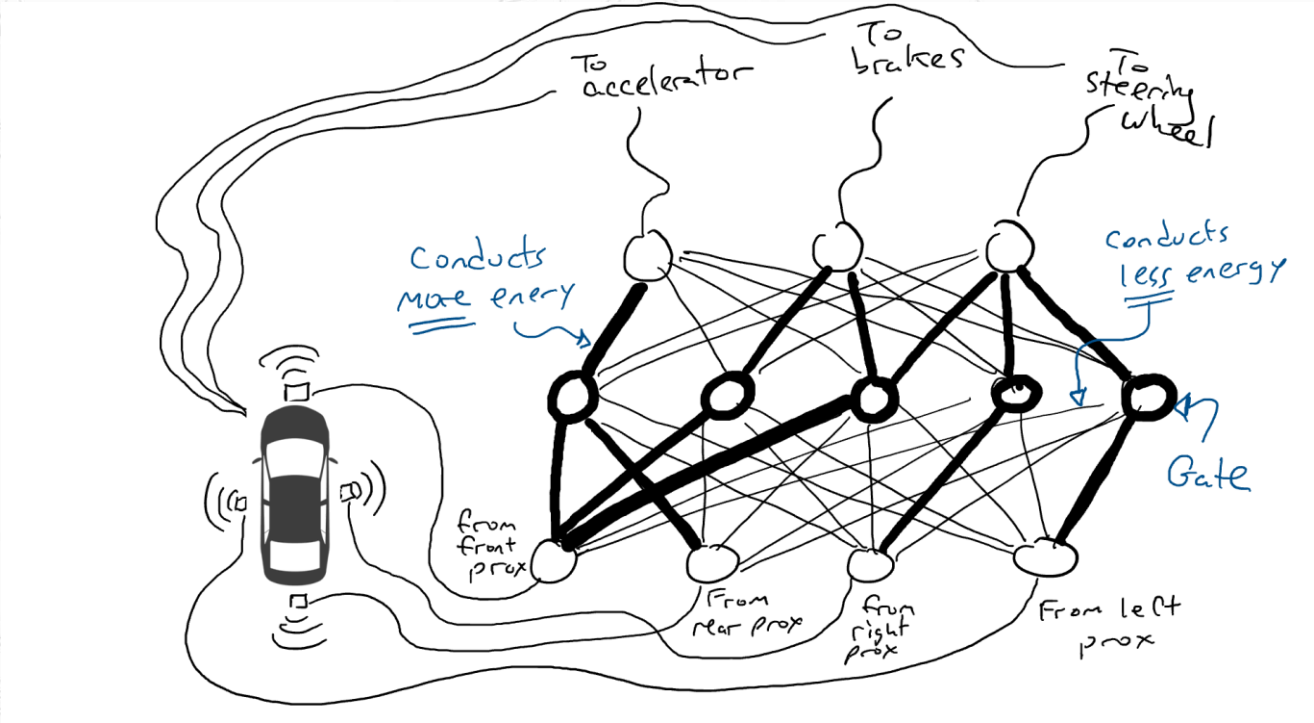
Ejemplo Modelos de DL



Ejemplo Modelos de DL



Ejemplo Modelos de DL



Modelos de Deep Learning

Generalidades del Deep Learning, los modelos de redes neuronales ya se habían empezado a desarrollar desde 1980 no obstante su uso no estaba difundido por temas de (1) costos computacionales y la (2) dificultad para entrenarlos correctamente respecto a otros métodos.

No obstante, para problemas que empezaron a ser altamente no lineales y que adicionalmente estaban desplegados sobre una gran cantidad de datos, empezaron a tener nuevamente fuerza.

Empezaron a surgir entonces dos grandes grupos de modelos de Deep Learning. Modelos de redes convolucionales (CNN) y modelos de redes recurrentes (RNNs).

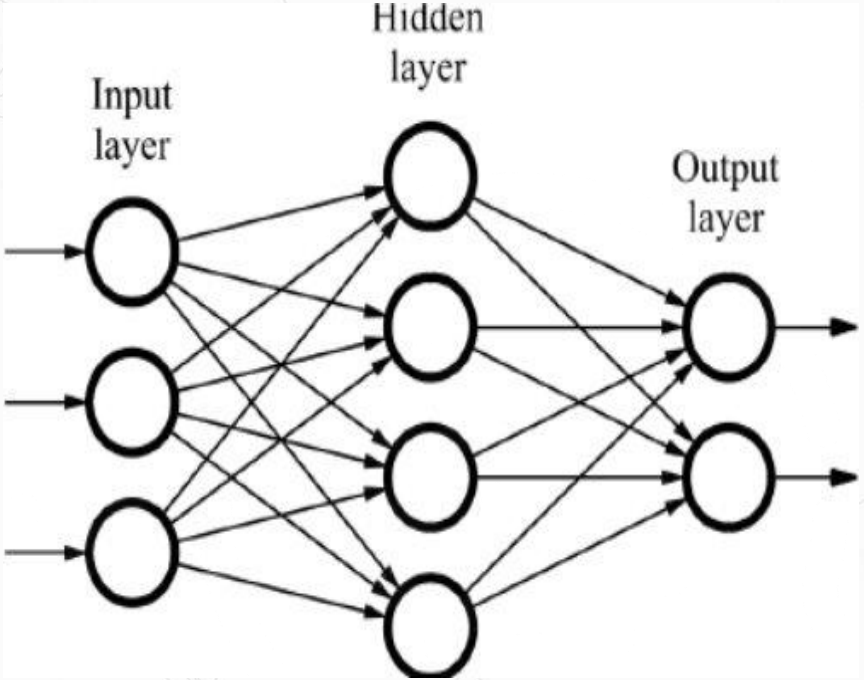
Modelos de Deep Learning

- Red neuronal simple:

Una red neuronal simple nos permite entonces relacionar una series de inputs con una serie de outputs, por medio de K unidades, esto lo podemos representar con la siguiente ecuación:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

- La función es entonces un componente clave del modelo entendiendo que este el que debe determinar la no-linealidad del problema.



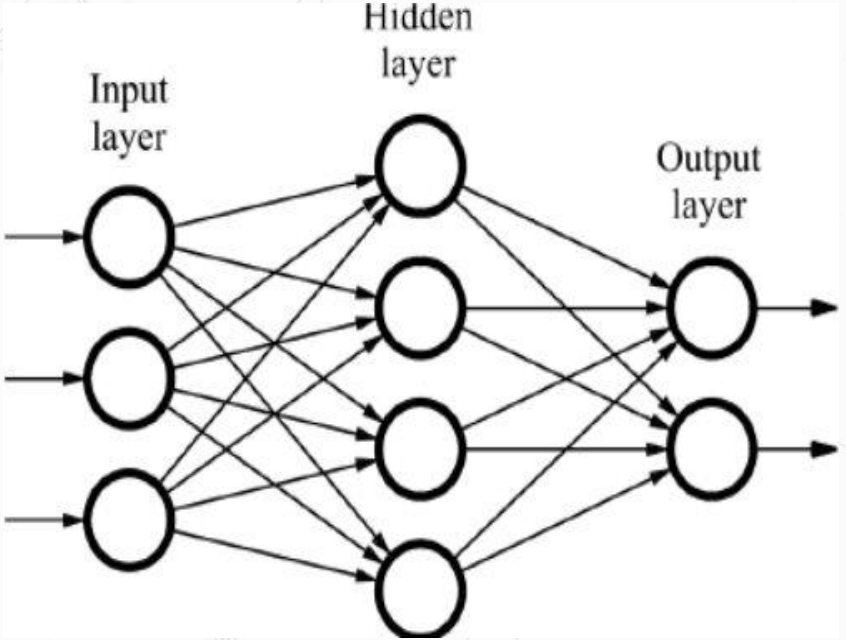
Modelos de Deep Learning

- Red neuronal simple:

Una red neuronal simple nos permite entonces relacionar una series de inputs con una serie de outputs, por medio de K unidades, esto lo podemos representar con la siguiente ecuación:

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)$$

- La función es entonces un componente clave del modelo entendiendo que este el que debe determinar la no-linealidad del problema.



Modelos de Deep Learning

A continuación, mostramos ejemplos posibles para la función de activación h_k.

Función Sigmoide:

$$f(x) = \frac{e^z}{1 + e^z}$$

Función ReLU:

$$g(x) = (z)_+ = \begin{cases} 0 & if \quad z < 0 \\ z & otherwise \end{cases}$$

Modelos de Deep Learning

•Un ejemplo de red neuronal simple se puede considerar el modelo de la regresión logística. En el caso de una regresión logística queremos resolver un problema de clasificación lo que generalmente implica traducir en una de dos categorías (1 si hará default 0 si no hará default):

•Este tipo de modelos asigna una probabilidad a cada 'output' y con base a dicha probabilidad se puede delimitar el escenario entre default y no default.

$$P(y = 1|\beta, x) = \hat{y}$$
$$P(y = 0|\beta, x) = 1 - \hat{y}$$

Modelos de Regresión Logística 👩‍💻

El cuál se puede generalizar como:

$$P(y = 1|X) = G(X\beta) = p(X)$$

Es decir lo queremos es obtener una transformación del modelo de regresión lineal que nos ayude a obtener una probabilidad asociado a cada resultado. De manera más específica se asumía que había una variable latente $y^* = X\beta + \epsilon$, si esta variable latente es mayor a cero, entonces la variable de respuesta va a tener un valor de 1 formalmente.

$$P(y = 1|X) = P(e > -X\beta|X) = 1 - G(-X\beta) = G(X\beta)$$

Esta transformación particularmente para los modelos Logit era la transformación sigmoide.

$$G(z) = \frac{\exp(z)}{1 + \exp(z)}$$

Modelos de Regresión Logística 🤷‍♀️

Algunos elementos del modelo de regresión logística son los siguientes:

$$P(y = 1|X) = G(X\beta) = p(X)$$

Es decir, lo queremos es obtener una transformación del modelo de regresión lineal que nos ayude a obtener una probabilidad asociado a cada resultado.
De manera más específica, se asumía que había una variable latente $y^* = X\beta + \epsilon$, si esta variable latente es mayor a cero, entonces la variable de respuesta va a tener un valor de 1 formalmente.

$$P(y = 1|X) = P(e > -X\beta|X) = 1 - G(-X\beta) = G(X\beta)$$

Esta transformación particularmente para los modelos Logit era la función sigmoide.

$$G(z) = \frac{\exp(X\beta)}{1 + \exp(X\beta)}$$

Bajo esta modelación el término del error de la variable latente $y^* = X\beta + \epsilon$ queda definido como una distribución logística estándar.

Modelos de Regresión Logística 👩‍💻

Estimación de Máxima Verosimilitud de la función de respuesta binaria. Queremos entonces maximizar la probabilidad de que cuando las variables sean igual a 1, la probabilidad sea lo más alta posible, formalmente esto significa:

$$para\ las\ yi = 1: \prod_{i:y_i=1} p(x_i)$$

$$para\ las\ yi = 0: \prod_{j:y_j=0} (1 - p(x_i))$$

Esto se puede expresar como el producto de ambas funciones:

$$L(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} (1 - p(x_i))$$

Podemos reescribir la ecuación con una sola productoria:

$$L(\beta) = \prod_{i} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

Obtenemos el log-likelihood de dicha función para facilitar la optimización:

$$l(\beta) = \sum_{i} y_i \log\big(p(x_i)\big) + (1 - y_i)\log(1 - p(x_i))$$

Se halla entonces el parámetro $\hat{\beta} = argmax(l(\beta))$

Modelos de Regresión Logística 🤦‍♀️

Este modelo de regresión logística se puede generalizar para que no corresponda con únicamente una categoría sino con varias. De esta manera afirmamos que vamos a tener que vamos a definir un modelo para que la variable tome uno de los siguientes valores j=0,…, J.

$$P(y = j|x) = \frac{\exp(x\beta_j)}{1 + \sum_{h=1}^{j} \exp(x\beta_h)}$$

Lo cuál es fácilmente generalizable:

$$P(y = 0|x) = \frac{1}{1 + \sum_{h=1}^{j} \exp(x\beta_h)}$$

Modelos de Regresión Logística 👩‍💻

Así mismo para esta función le podemos estimar su función de máximo verosimilitud.

$$l_i(\beta) = \sum_{j=0}^{J} 1[y_i = j]\log(p_j(X_i))$$

Donde 1 esta definido como el indicador de la función logística:

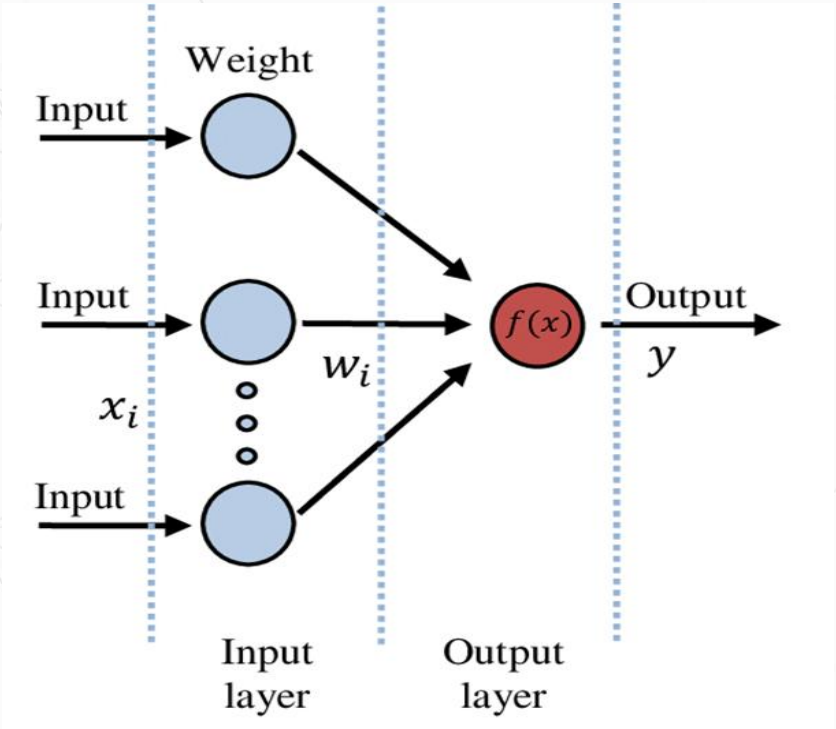
$$1[y_i = j] - \begin{cases} 1\ si\ y_i = j \\ 0\ si\ y_i \neq j \end{cases}$$

Ejemplo red neuronal simple

Podemos entonces entender el modelo logit como una red neuronal de una sola unidad. Esta red neuronal la podríamos escribir como:

$$f(X) = \beta_0 + \beta_1 g(w_{k0} + \sum_{j=1}^{P} w_{kj} X_j)$$

Donde existen P variables.



Ejemplo red neuronal simple

Mientras que los parámetros betas son , $\beta_0 + \sum K_{k=1}\beta_k h_k$,

$$f(X) = \beta_0 + \beta_1 \sigma \left(w_{k0} + \sum_{j=1}^{P} w_{kj} X_j \right)$$

Donde estamos fijando los parámetros $\beta_0 \; y \; \beta_1$ en 0 y 1 respectivamente. Y la matriz de W pasarían a ser los betas que estimamos con la función logística.

$$f(X) = \sigma \left(w_{k0} + \sum_{k=1}^{K} w_{kj} X_j \right)$$

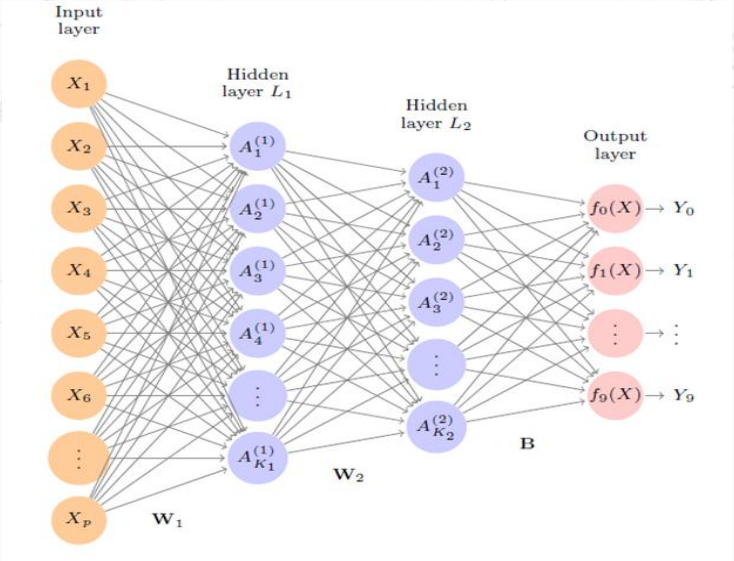
Para hacer más clara la notación con $\beta_0 = 0$ y $B_1 = 1$:

$$f(X) = \sigma \left(w_{k0} + \sum_{k=1}^{K} w_{kj} X_j \right)$$

$$f(X) = \sigma \left(\beta + \sum_{k=1}^{K} \beta_{ki} X_i \right)$$

Redes neuronales múltiples

- Una red neuronal múltiple la podríamos definir como aquellla con mas de una capa oculta, la cuál además tienen más de un input por capa.



Redes neuronales múltiples

Tendríamos entonces las siguientes ecuaciones, para una unidad k de la capa oculta número 1.

$$A_k^{(1)} = h_k^{(1)}(X) = g\left(w_{k0}^{(1)} + \sum_{j=1}^{p} w_{kj}^{(1)} X_j\right)$$

Donde g es la función de activación. Para la unidad l de la capa oculta número 2.

$$A_l^{(2)} = h_l^{(2)}(X) = g\left(w_{k0}^{(2)} + \sum_{j=1}^{p} w_{kj}^{(2)} A_j^{(1)}\right)$$

Finalmente para el m-ésimo output.

$$Z_m = \beta_{m0} + \sum_{j=1}^{K_2} \beta_{ml} A_l^{(2)}$$

Finalmente usamos la función de activación softmax que convertirá este input en probabilidades.

$$f_m(X) = \frac{e^{Zm}}{\sum_{m=0}^{M} e^{Zm}}$$

Modelos de Deep Learning

Por último, los modelos de Deep Learning pueden tener las siguientes funciones de perdida. Si se trata de un modelo de regresión. Se puede el Error Cuadrático Medio ante un output continuo.

En cambio, si el modelo es discreto en este caso se usaría la formula de cross-entropía. Cuya formula es la siguiente

$$-\sum_{i=0}^{n}\sum_{m=0}^{9} y_{im}\log\left(f_m(x_i)\right)$$

Minimizar esta función es entonces equivalente a maximizar la función que habíamos construido de máxima versosimilitud.

Algoritmo de Estimación

Tenemos el siguiente algoritmo de estimación para los modelos de redes neuronales. Suponiendo que la función objetivo es la siguiente:

$$\min_{\{w_k\}_1^K} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Donde $f(x_i)$ podemos plantearla para una red neuronal simple por ejemplo:

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij} \right)$$

Esta estimación puede verse afectada por las siguientes dos situaciones: slow learning y regularization

Algoritmo de Estimación

La técnica de regularización básicamente consiste en imponer una penalización sobre los parámetros del modelo del tipo ridge o lasso:

$$R(\theta:\lambda) = -\frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda\sum_{j}\theta_j^2$$

Mientras que la técnica de *slow learning* consiste en usar el método de *gradient descent.* Intentado detener el proceso cuando se halla podido identificar algo cercano al ovefitting. El algoritmo de *gradient decent* se plantea de la siguiente manera:

1. Inicialice el parámetro θ^0 con alguna conjetura que se tenga sobre el

2. Encuentre el gradiente de la función: $\nabla R^{\theta}(\theta^m) = \frac{\partial R(\theta)}{\partial \theta}_{\theta=\theta^m}$

3. Actualice el parámetro $\theta^m - \rho\nabla R^{\theta}(\theta^m) \rightarrow \theta^{m+1}$

Algoritmo de Estimación

La técnica de Backpropagation consiste :

$$\frac{\partial R_i(\theta)}{\partial \beta_k} = \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \frac{\partial f_\theta(x_i)}{\partial \beta_k}$$

$$\frac{\partial R_i(\theta)}{\partial \beta_k} = (y_i - f_\theta(x_i)) \frac{\partial f_\theta(x_i)}{\partial \beta_k}$$

Ahora tenemos la derivada con respecto a los pesos w_{kj}.

$$\frac{\partial R_i(\theta)}{\partial w_{kj}} = \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} = -\left(y_i - f_0(x_i)\right) \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g\,(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}}$$

$$\frac{\partial R_i(\theta)}{\partial w_{kj}} = -\left(y_i - f_0(x_i)\right) \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g\,(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}}$$

En últimas parte del residual se empieza a distribuir de acuerdo con la regla de la cadena. Es precisamente este hecho el que se aprovecha con la técnica de backpropagation. De manera que se obtengan primero las derivadas parciales que no dependen de las demás (las de la última capa) y así progresivamente se computen las subsiguientes.

Modelos de Lenguaje de Gran Tamaño

BCRP – CEFA 2025

Modelos de Lenguaje

Un modelo de lenguaje (LM) es aquel que define una distribución de probabilidad sobre una sequencia de palabras que pertenezcan a un diccionario predefinido. Si $x_1, \dots, x_L \in V$ entonces el modelo obtiene una probabilidad para alguna secuencia.

$$0 \leq p(x_1, \dots, x_L) \leq 1$$

Esta probabilidad ha de incrementar conforme tenemos oraciones de mayor calidad en cuanto su sintaxis o en cuanto a su semántica.

$$p(Ratón\ el\ queso\ el\ comió) = 0.00001 p(El\ ratón\ se\ comió\ el\ queso)$$
$$= 0.01 p(El\ ratón\ se\ comió\ el\ queso) = 0.2$$

Modelos de Lenguaje Autorregresivos

Una forma de escribir la probabilidad de una secuencia es aplicando la probabilidad condicional en cadena. Es decir condicionando la ocurrencia de cada token en las anteriores:

$$= p(El)p(ratón \mid el)p(se|El\ ratón) \dots p(queso|El\ ratón\ se\ comió\ el)p(El\ ratón\ se\ comió\ el\ queso)$$

En términos mas generales:

$$p(x_{1:L}) = \prod_{i=1}^{L} p(x_i|x_{1:i-1})$$

Modelos de Lenguaje

Como vimos los modelos de redes neuronales nos sirven para generar una distribución de probabilidad dado una serie de parámetros e inputs. Por lo que prometen ser útiles para encontrar esta probabilidad condicional.

Inicialmente se emplearon los modelos de redes neuronales recurrentes (RNN) para este fin. No obstante estos modelos tenían dos problemas:

- No es posible paralelizar su entrenamiento, puesto que procesan las palabras de forma secuencial.
- No es fácil lograr que lograran centrar su atención en palabras sin importar lo lejos que estén.

Modelos de Lenguaje

A raíz de un documento publicado por Google “Attention is all you need” se empezaron a implementar la arquitectura de Transformers para llegar a los mejores modelos de lenguaje.

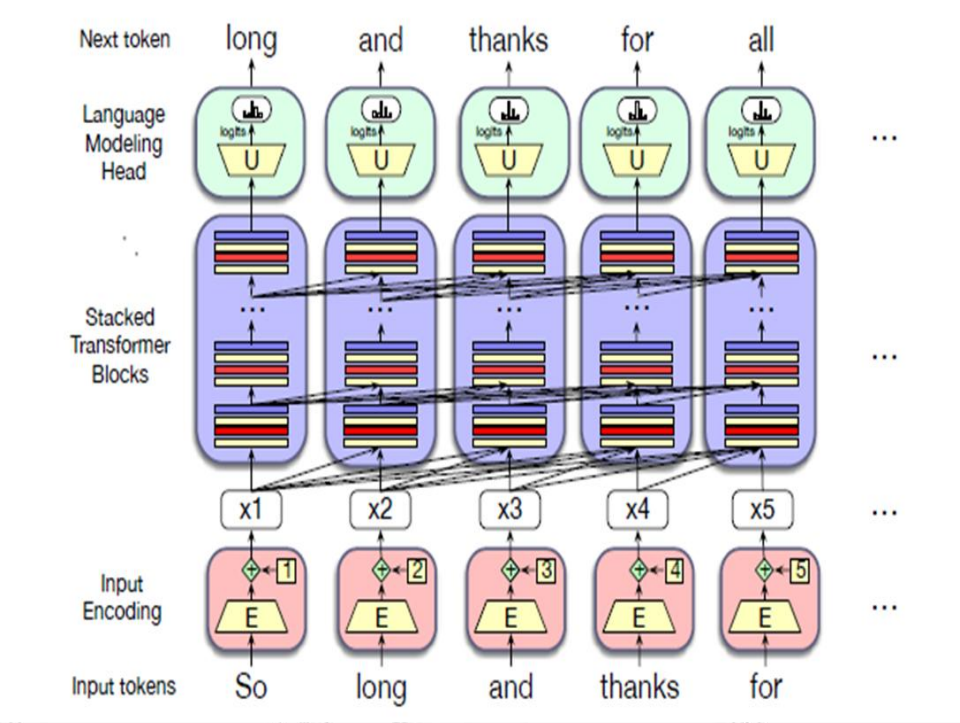
Esta arquitectura de Transformers se basa en el mecanismo de “auto-atención” (*self attention)*

Adicionalmente, se comenta que este tipo de arquitecturas no trabajan con palabras directamente, sino con tokens. Esto permite no solo incluir palabras sino también signos de puntuación, combinaciones de palabras, o a veces también partes de una palabra.

Modelos de Lenguaje

- *(Self Attention)* puede pensarse como una manera de desarrollar una representación del significado de un token prestando atención a las palabras que lo rodean.
- La arquitectura de Transformers involucra las siguientes tres etapas: (1) Un input-encoding de los tokens (2) Bloques de Transformers y un (3) Language model head

Transformer

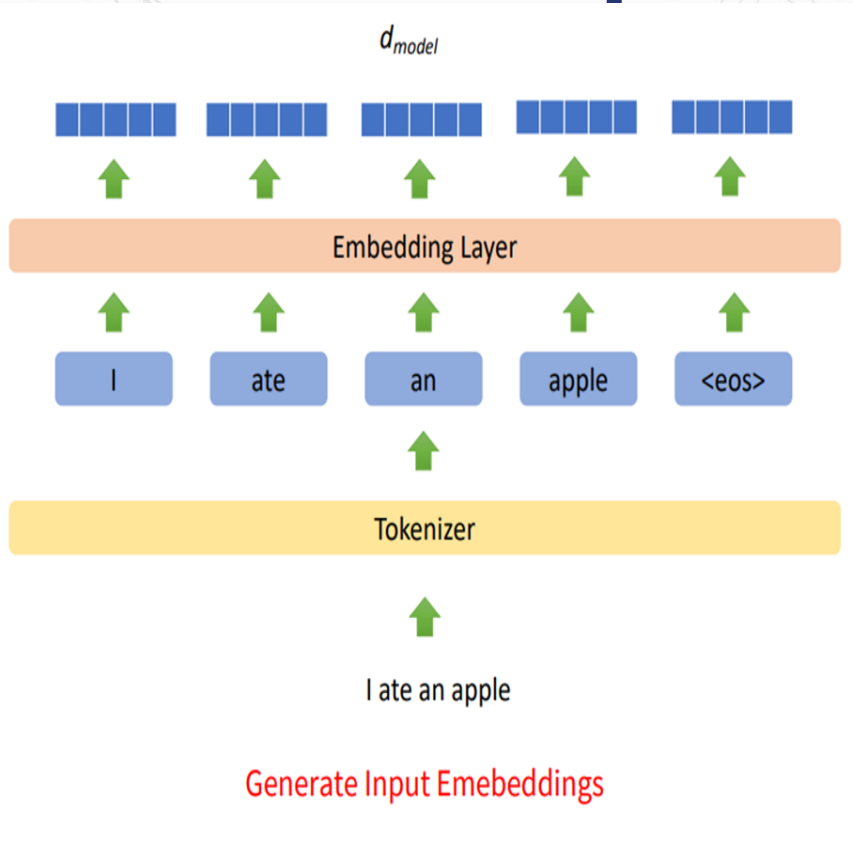


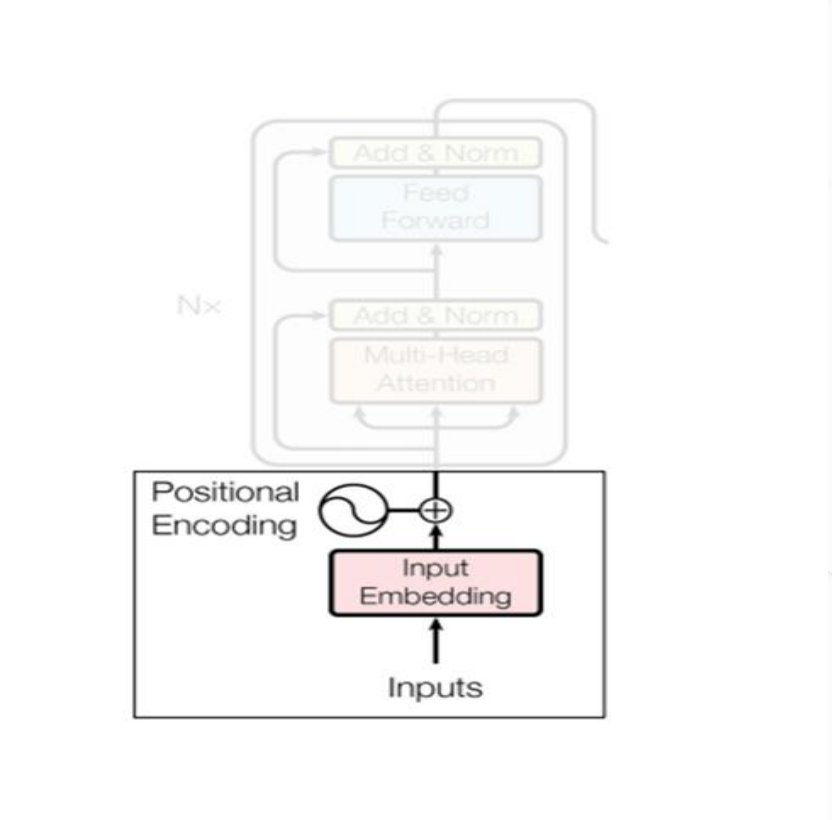
Input Encoding

En la fase del “input encoding” se logra codificar:
(1) una representación vectorizada de la semántica del token.
(2) una representación de la posición del token.

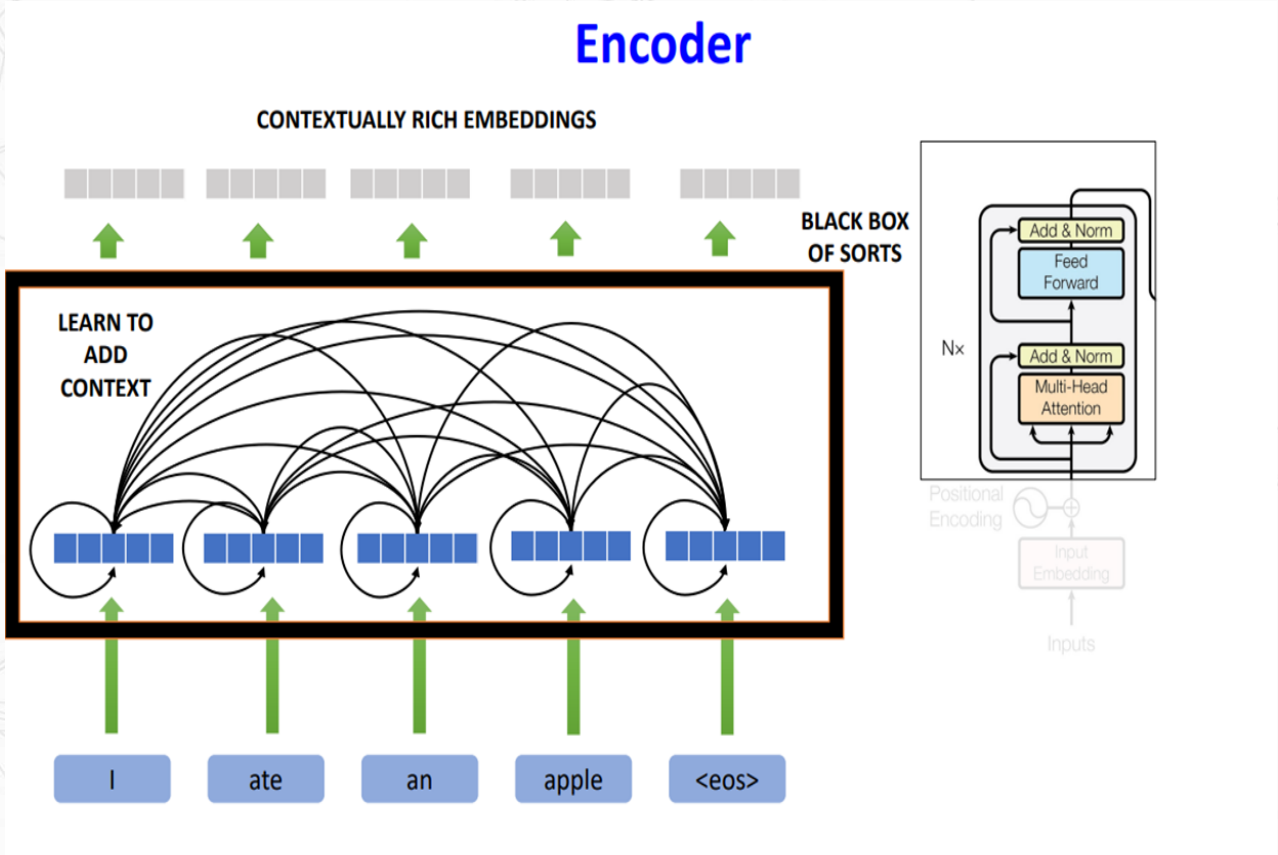
Esta información queda condensada en un vector de [1xd] entiéndase d como la dimensión del modelo que puede ser ajustada para generar mejores respuestas.

Input Encoding



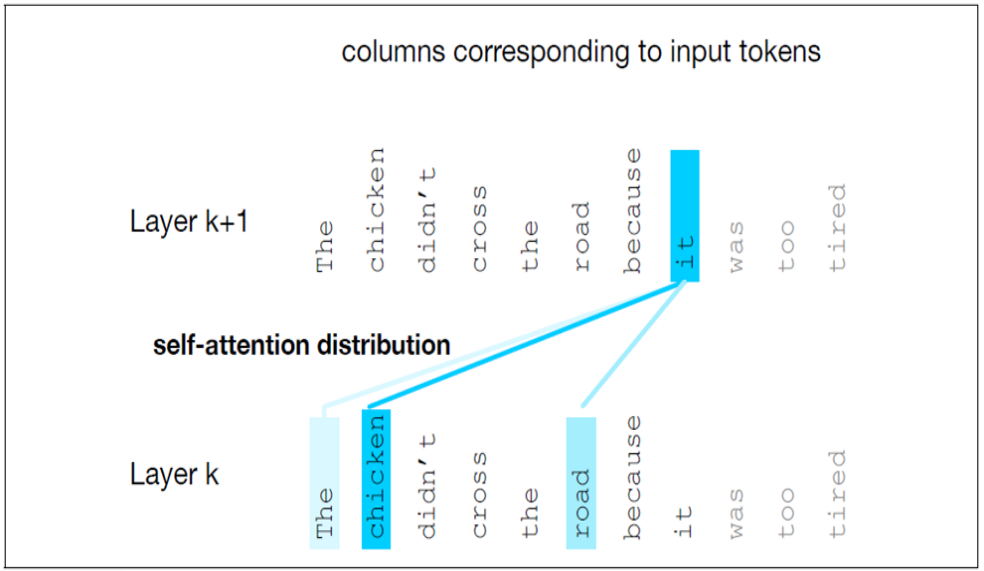


Encoder


Encoder
CONTEXTUALLY RICH EMBEDDINGS
BLACK BOX OF SORTS
LEARN TO ADD CONTEXT
I
ate
an
apple
<eos>
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Atención (simplificado)

El mecanismo de atención se puede identificar en su versión simplificada de la siguiente manera: $a_i = \sum_{j \leq 1} \alpha_{ij} x_j$.Se busca entonces obtener una representación de una palabra con base en todo el cuerpo de palabras que la antecedió:



Atención (simplificado)

Estos pesos del mecanismo de self-attention se calcularían inicialmente con el producto punto y después se los sometería a la función de softmax en miras a tener pesos que sumen en total 1.

$$score(x_i, x_j) = x_i x_j$$
$$\alpha_{ij} = softmax(score(x_i x_j))$$

Atención (Completo)

No obstante para representar de forma mas realista el mecanismo de atención, es necesario tener una representación del vector de entrada por medio de las siguientes tres matrices:

$$q_i = x_i W^Q$$
$$k_i = x_i W^K$$
$$v_i = x_i W^V$$

En el caso del Query, se lo esta definiendo en su rol de elemento actual que será comparado con los inputs que lo preceden.
El key, se lo está definiendo en su rol de elemento que será comparado con los anteriores pesos.
Y finalmente, propiamente el valor con el significado contextualizado que tenga la palabra.

Atención (Completo)

Dadas las anteriores definiciones podemos entonces construir primero un score que identifica la correspondencia entre la llave del anterior input y la consulta que le hace el input propiamente:

$$score(x_i, x_j) = \frac{q_i k_j}{\sqrt{d_k}}$$

Así como un softmax de dicha función, con el propósito de tener pesos que sumen 1.

$$\alpha_{ij} = softmax\left(score(x_i, x_j)\right)$$

Posteriormente ahora podemos computar la representación

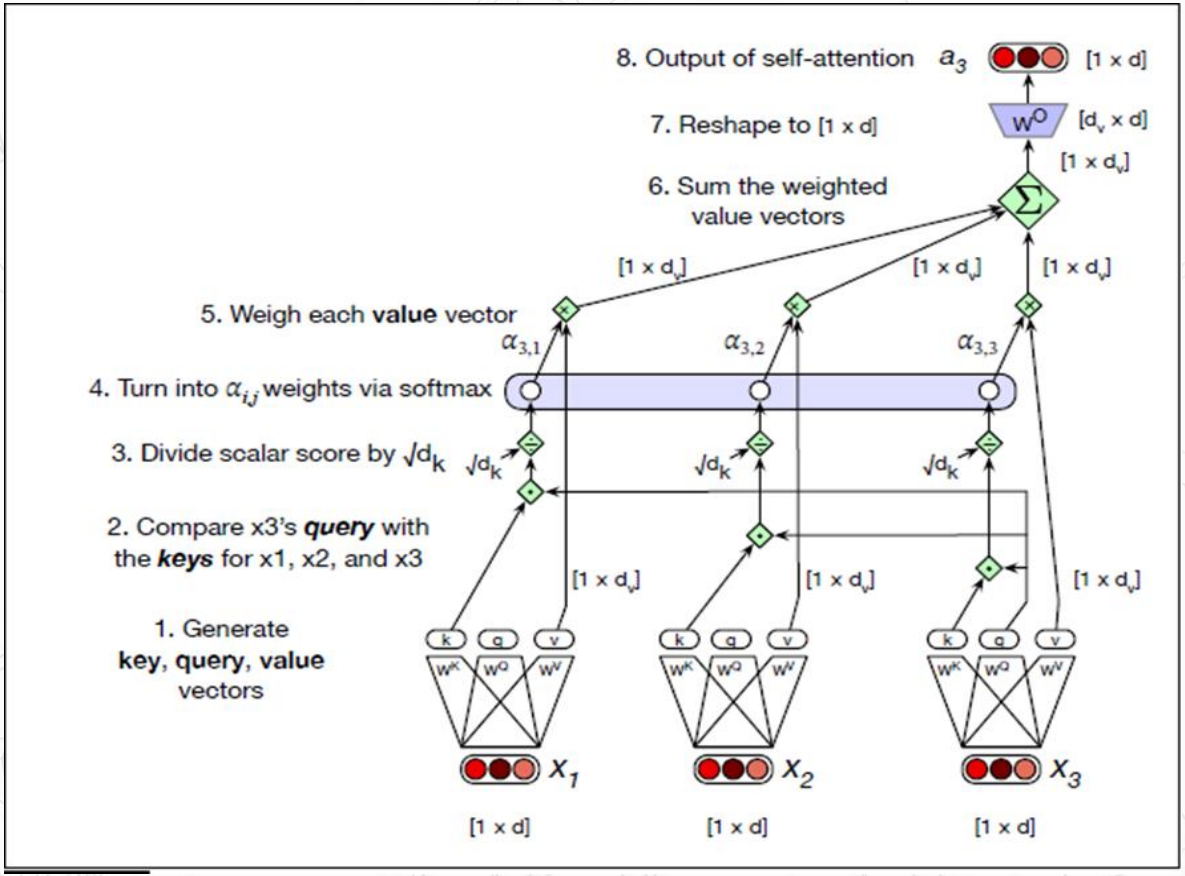
Atención (Completo)

Posteriormente ahora podemos computar la representación en términos de todos los inputs que lo proceden:

$$head_i = \sum_{j \leq i} \alpha_{ij} v_j$$

Esta cabeza esta en dimensiones de $[1 \times d_k]$ por ende resultaría necesario obtener un output del mismo tipo a_i.

Bloques de Transformers



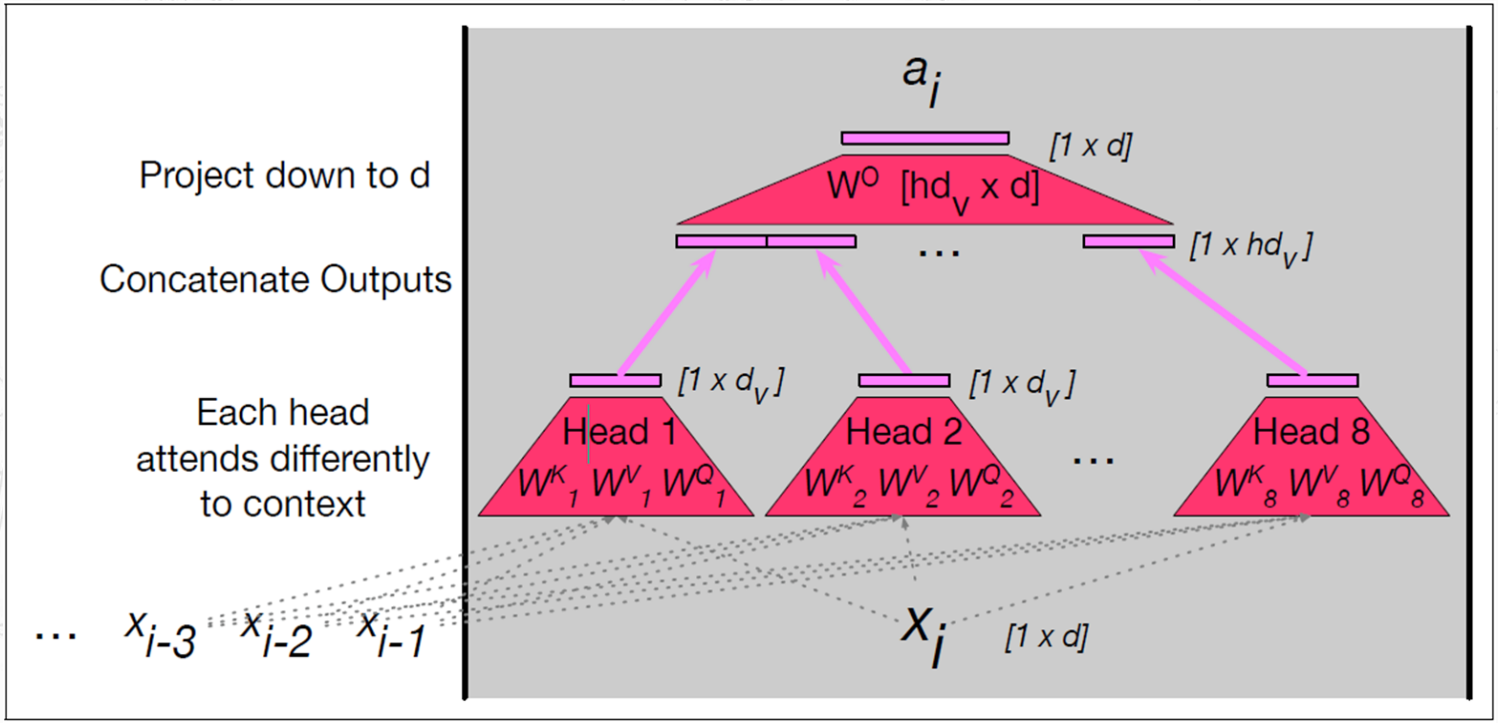
Atención más formalmente

$$q_i^c = x_i W^{Qc}$$
$$k_j^c = x_j W^{K_c}$$
$$v_j^c = x_j W^{V_c}$$

Podemos entonces computar el score entre la llave y el query y representarlo con la siguiente expresión matemática:

$$score^c(x_i, x_j) = \frac{q_i^c k_j^c}{\sqrt{d_k}}$$
$$a_{ij}^c = softmax\left(score^c(x_i, x_j)\right)$$
$$head_i = \sum_{j \le i} a_{ij}^c v_j^c$$
$$a_i = head^1 \oplus head^2 \oplus \cdots \oplus head^4 W^O$$

Atención más formalmente



Atención más formalmente

Podemos entonces representar finalmente un ‘transformer’ como el siguiente conjunto de operaciones sucesivas:

$$t_i^1 = LayerNorm(x_i)$$
$$t_i^2 = MultiHeadAttention\left(t_i^1, [t_1^1, \dots, t_N^1\right)$$
$$t_i^3 = t_i^2 + x_i$$
$$t_i^4 = LayerNorm\left(t_i^3\right)$$
$$t_i^5 = FFN(t_i^4)$$
$$h_i = t_i^5 + t_i^3$$

Generación de secuencias

Hay tres metodologías para generar una secuencia usando un modelo de lenguaje de gran tamaño:
Greedy decoding. Generar el *outcome* más probable a partir de los parámetros recibidos: las palabras que le preceden. Esto se puede representar esquemáticamente de la siguiente manera:

$$\hat{w}_t = argmax_{w \in W} p(w|w_{<t})$$

Este método tiene genera respuestas de forma determinística: ante el mismo input genera entonces la misma secuencia de palabras.
Aunque a primera vista esto podría considerarse como un aspecto positivo, en realidad genera respuestas demasiado predecibles o incluso genéricas.

Métodos de sampleo

Por esta razón es que también se decide introducir métodos de ‘sampleo’.
Sampleo simple: Obtener valores de la distribución de probabilidad generada en el último output.
Top k-sampling: Consiste en:

(1) Preseleccionar un número k de palabras sobre las cuáles se quiere realizar el sampleo, ordenárlas según las que mayor probabilidad asignada tengan.
(2) Ponderar nuevamente las probabilidades sobre estas k palabras para que sumen 1.
(3) Generar aleatoriamente la palabra.

Métodos de sampleo

Otro método de sampleo es el de **Nucleus or top-p sampling.** Consiste en realizar un sampleo sobre las palabras que acumulen determinado porcentaje del total de probabilidad. Esto hace que sea menos sensible a la distribución de probabilidad generada.

Métodos de sampleo

Adicionalmente, también contamos con el mecanismo de sampleo por temperatura. Básicamente esto consiste en aplicar a la función softmax la siguiente transformación.

$$softmax\left(\frac{u}{\tau}\right)$$

En caso tal de que τ este muy cerca de cero, esto básicamente amplificar la diferencia de probabilidad asignada entre las combinaciones con menor probabilidad y las de mayores probabilidad, de manera que finalmente se trata de un greedy decoding.

Métodos de sampleo

$$softmax\left(\frac{u}{\tau}\right)$$

En caso tal de que τ este muy cerca de 1, esto básicamente implica realizar un random sampling.

En caso tal de que τ este muy cerca a infinito, esto significa realizar un sampleo casi de una distribución uniforme del vocabulario.

Se puede tratar de ajustar ese parámetro según el grado de creatividad que se busque en las respuesta.

Métodos de sampleo

$$softmax\left(\frac{u}{\tau}\right)$$

En caso tal de que τ este muy cerca de 1, esto básicamente implica realizar un random sampling.
En caso tal de que τ este muy cerca a infinito, esto significa realizar un sampleo casi de una distribución uniforme del vocabulario.

Se puede tratar de ajustar ese parámetro según el grado de creatividad que se busque en las respuesta.

Métricas de LLM’s

Otra métrica para complementar el entrenamiento de los LLM’s es el Perplexity. Es decir es una medida de la plausibilidad de una secuencia generada:

$$Perplexity_\theta(w_{1n}) = P_\theta(w_{1n})^{\frac{1}{n}}$$

$$Perplexity_\theta(w_{1n}) = \sqrt[n]{\frac{1}{P_\theta(w_{1n})}}$$

En esa medida lo que se busca es minimizar el perplexity, puesto que eso implica maximizar la probabilidad de que la secuencia halla ocurrido.

FLAR
FONDO LATINOAMERICANO DE RESERVAS

Recomendaciones para el uso de Modelos de Inteligencia Artificial

BCRP – CEFA 2025

Recomendaciones para el uso de modelos de IA

Uso de ChatGPT

Los modelos de GPT han estado entrenados mayoritariamente con textos en inglés. Si es una consulta sobre un contenido que no sea referente a temas idiomáticos es preferible hacerla en inglés.

Ser específico: Se ha de ser tan específico como sea posible en la configuración del 'prompt'.

Partir el Query lo más posible: Si una tarea implica varios pasos puede ser buena idea partir el query en varias tareas. Esto le puede ayudar a ChatGPT a procesar de una manera más eficiente los resultados.

Iterar y refinar: Eventualmente un Prompt a la primera puede que no obtenga el resultado correcto por tanto no es mala idea tratar de hacer la misma pregunta nuevamente.

Valerse de ejemplos: Algunas veces proveer de ejemplos a ChatGPT le ayudarán a tener un mejor entendimiento de la tarea que le fue asignada.

Prompt Engineering

Usar los siguientes segmentos en el ‘prompt’ puede generar mejores respuestas:

- "Tomando el rol de un analista financiero con experiencia sólida en mercados de capitales
- “Analice el siguiente titular [Afirmación 1]. Responda “Si” si son buenas noticias, “No” si son malas noticias, o “Indeterminado” de no estar seguro. Después elabore una pequeña y concisa oración en la siguiente línea”
- "¿Es esta afirmación buena, mala para la compañía [Nombre de la compañía] en el corto plazo?"

Rol: El perfil del rol de la persona que mejor contestaría a la pregunta.

Contexto:
La información relevante que el modelo de lenguaje debe usar para formular la respuesta y las características de la respuesta que se está formulando.

Pregunta:
Lo que se espera que responda el modelo.

Recomendaciones generales

Escribir instrucciones claras. Que complementen la tarea que se solicitó
- Si el output obtenido fue muy largo, pregunte por repuestas mas cortas.
- Si usted no está de acuerdo con el formato, muestre un ejemplo del formato que a usted le gustaría ver.
- En general, entre menos dudas halla abiertas para el modelo de lo que se desea hacer, es más probable que usted obtenga exactamente lo que quiere.

Provea referencias en el texto claras

Es muy útil remitirse a las citas en el texto sobre las cuáles usted desea que el portafolio genere la respuesta. De manera equivalente a cuando a un estudiante se le da una pista para responder correctamente un examen.

Separe instrucciones complejas en tareas simples

Separar instrucciones complejas entre varias tareas simples. Al igual que es una buena práctica en la ingeniería de software descomponer un sistema complejo entre diferentes componentes modulares más simple, tareas más complejas pueden ser redefinidas como una secuencia de tareas más sencillas.

Proporcione al modelo tiempo para pensar

Esta técnica obedece al uso del sistema de la cadena de razonamiento para que el modelo llegue a la respuesta de una mejor manera. Al igual que cuando se le pregunta por un cálculo si se le da menos tiempo al modelo para responder una pregunta es más fácil que el modelo arroje una respuesta más arbitraria más fácilmente.

Use herramientas externas.

Compense por las debilidades del modelo valiéndose de otras herramientas, por ejemplo, el uso de una herramienta de RAG puede decirle al modelo cuáles serían los documentos más relevantes. Esta técnica ya está siendo implementada por la aplicación que hemos construido.

Pruebe cambios de forma sistemática

Para estar seguros de que un cambio en un prompt mejora la respuesta obtenida, puede ser necesario crear una batería de problemas. Este ejercicio ya lo estamos haciendo nosotros. Adicionalmente usted puede proveer una serie de tácticas para dar una respuesta. Incluya explicaciones más completas de lo que se desea obtener como respuesta.

Referencias

Wooldridge, J. M. (2010). *Econometric analysis of cross section and panel data*. MIT press.

Jurafsky, D., & Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer. *Machine Learning*