Cursos Extraordinarios

verano 2025

“Inteligencia Artificial y Grandes Modelos de Lenguaje: Funcionamiento, Componentes Clave y Aplicaciones”

Zaragoza, del 30 de junio al 02 de julio de 2025

Transformer

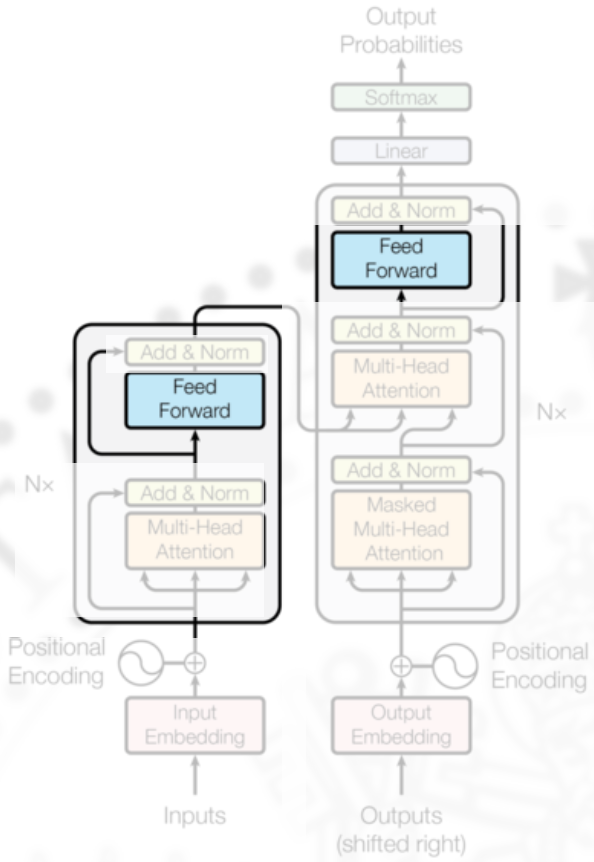
- **Arquitectura**
  - Residual layers
  - Layer normalization
  - Multi-head attention
  - **Feed forward**
  - Positional embedding
  - Embedding de entrada
  - Arquitectura general

Transformer

- **Bloque Feed forward**
  - La mayor parte de los parámetros de un transformer
  - La dimensión de la capa oculta H > D
    - Normalmente H = 4D

$$f(\boldsymbol{x}) = \boldsymbol{x} + \underbrace{\max(\boldsymbol{x} \cdot \boldsymbol{W}_1, 0) \cdot \boldsymbol{W}_2}_{\Delta \boldsymbol{x}}$$

$\boldsymbol{W}_1 \in \mathbb{R}^{D \times H}$
$\boldsymbol{W}_2 \in \mathbb{R}^{H \times D}$



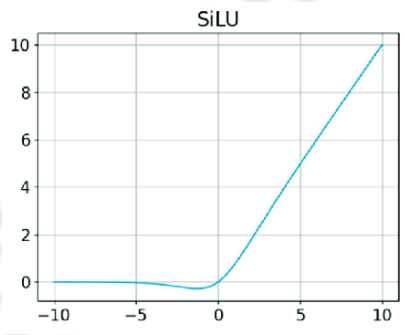
- Ejemplos
  - GPT3-7B:
  - 32 layers D = 4096, H = 16384
  - $|\Theta|_{ff} = 67\text{M}$
  - $|\Theta|_{32 \times ff} = 2.1\text{B}$
  - GPT3-13B:
  - 40 layers D = 5140, H = 20560
  - $|\Theta|_{ff} = 106\text{M}$
  - $|\Theta|_{40 \times ff} = 4.2\text{B}$

Transformer

- **Otras implementaciones**
  - LLAMA, Mistral

$$f(\boldsymbol{x}) = \boldsymbol{x} + \left(\text{silu}\left(\boldsymbol{x} \cdot \boldsymbol{W}_{gate}\right) \circ \left(\boldsymbol{x} \cdot \boldsymbol{W}_{up}\right)\right) \cdot \boldsymbol{W}_{dw}$$

$$\boldsymbol{W}_{up} \in \mathbb{R}^{D \times H}, \boldsymbol{W}_{gate} \in \mathbb{R}^{D \times H}$$
$$\boldsymbol{W}_{dw} \in \mathbb{R}^{H \times D}$$



$$\text{silu}(x) = x \cdot \sigma(x)$$

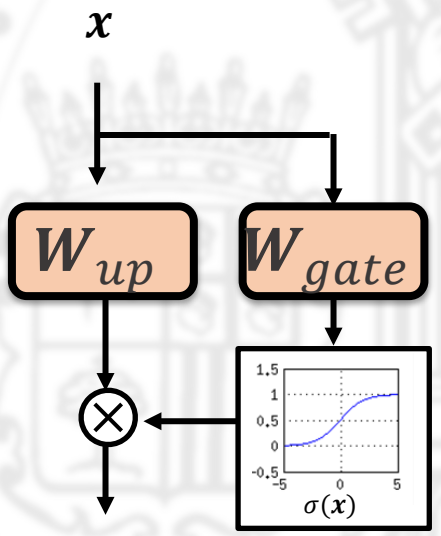
https://github.com/mistralai/mistral-src/blob/main/mistral/model.py#L141C16-L141C68
https://github.com/huggingface/transformers/blob/main/src/transformers/models/llama/modeling_llama.py#L268C12-L268C89

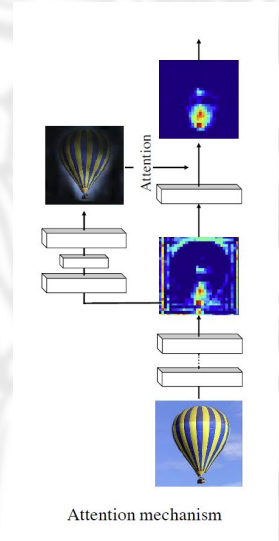
  - Gate attention

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. arXiv preprint arXiv:1505.00387.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

  - Element-wise product:
    - El mecanismo de atención tiupo puerta /gate
    - Deja pasar o inhibe la información

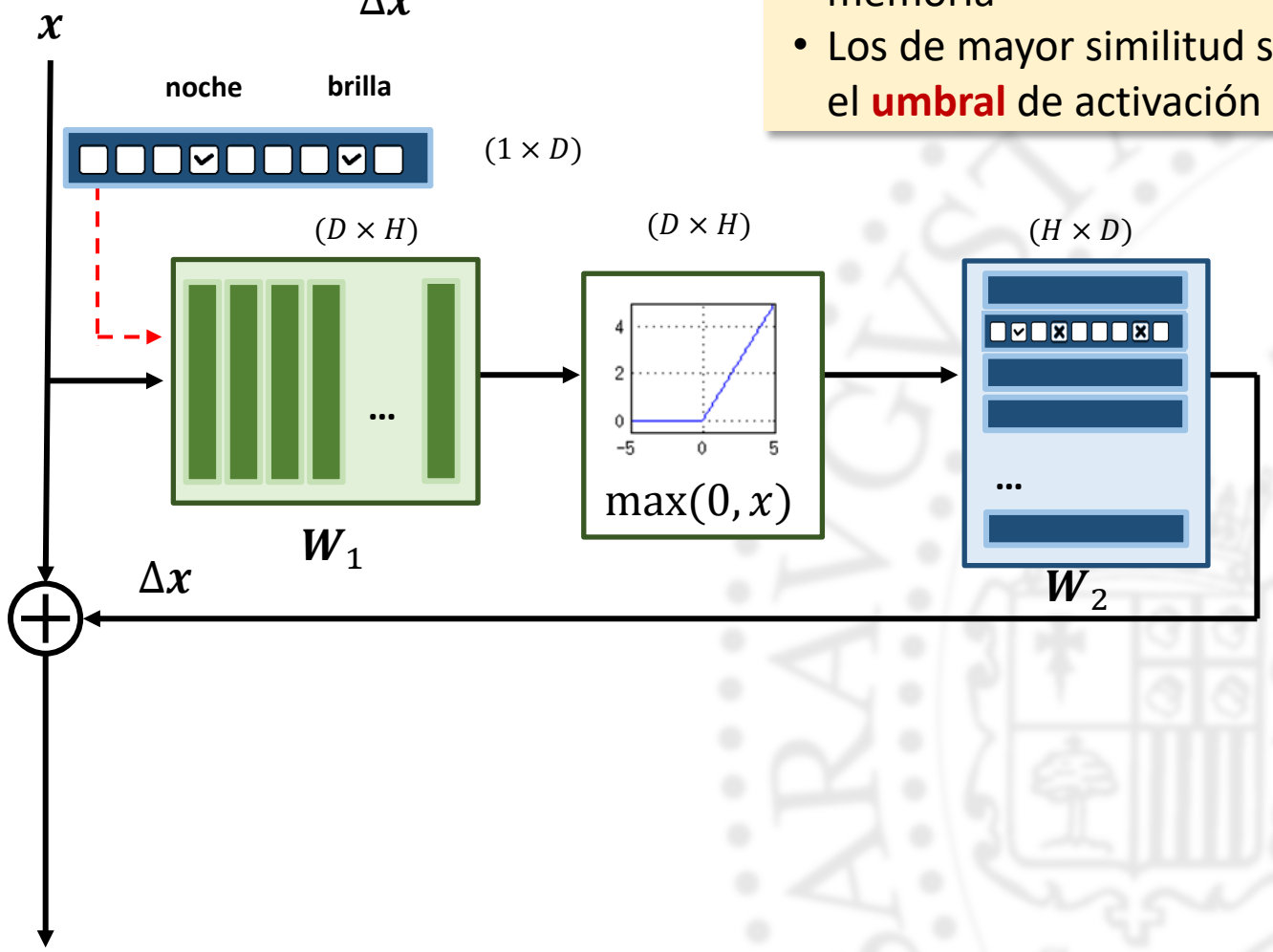




Transformer

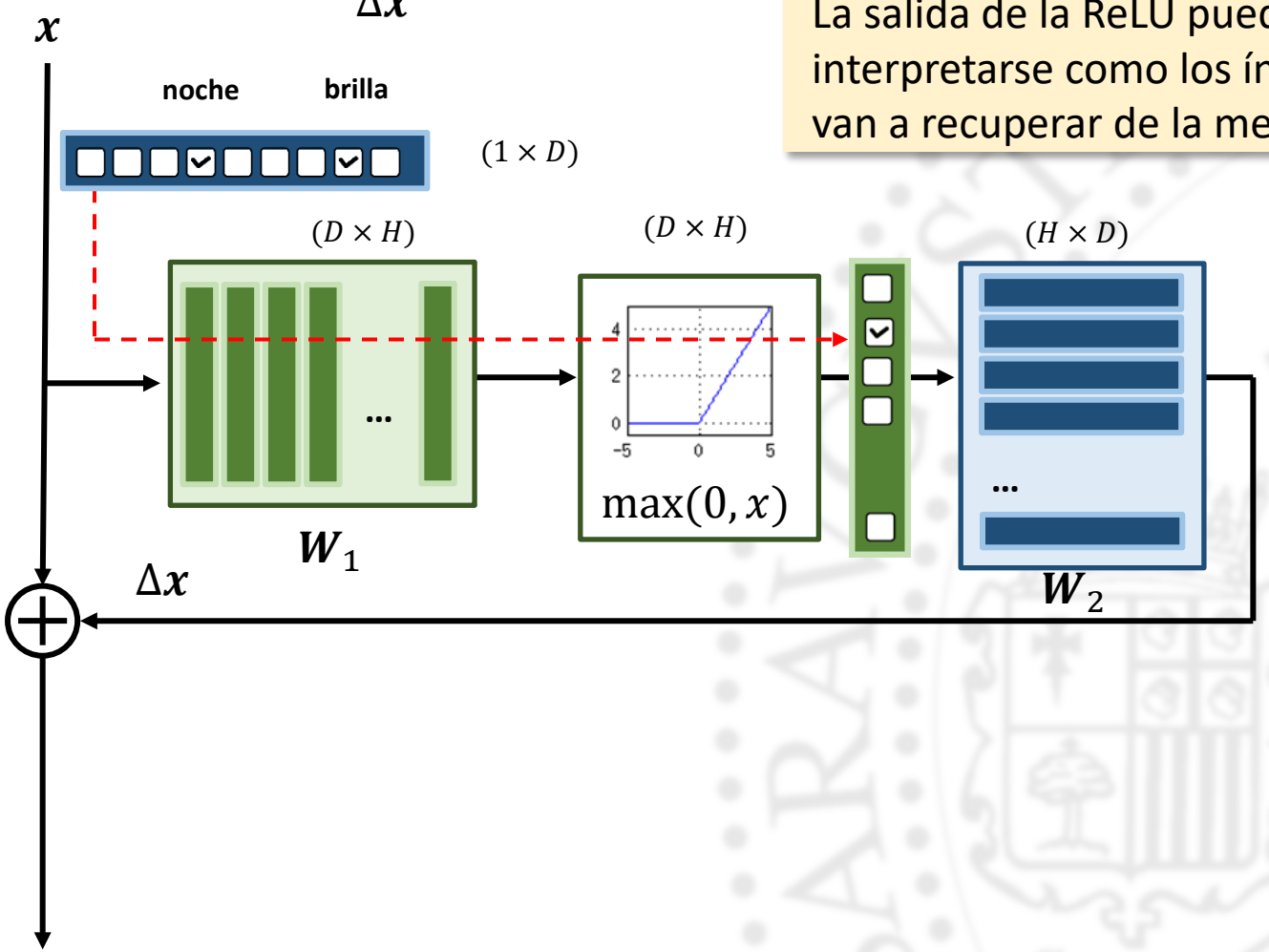
$$f(\boldsymbol{x}) = \boldsymbol{x} + \underbrace{\max(\boldsymbol{x} \cdot \boldsymbol{W}_1, 0) \cdot \boldsymbol{W}_2}_{\Delta \boldsymbol{x}}$$

- $\boldsymbol{W}_1$ contiene los vectores prototipo para **indexar** la memoria
- Los de mayor similitud superan el **umbral** de activación



Transformer

$$f(\boldsymbol{x}) = \boldsymbol{x} + \underbrace{\max(\boldsymbol{x} \cdot \boldsymbol{W}_1, 0) \cdot \boldsymbol{W}_2}_{\Delta \boldsymbol{x}}$$

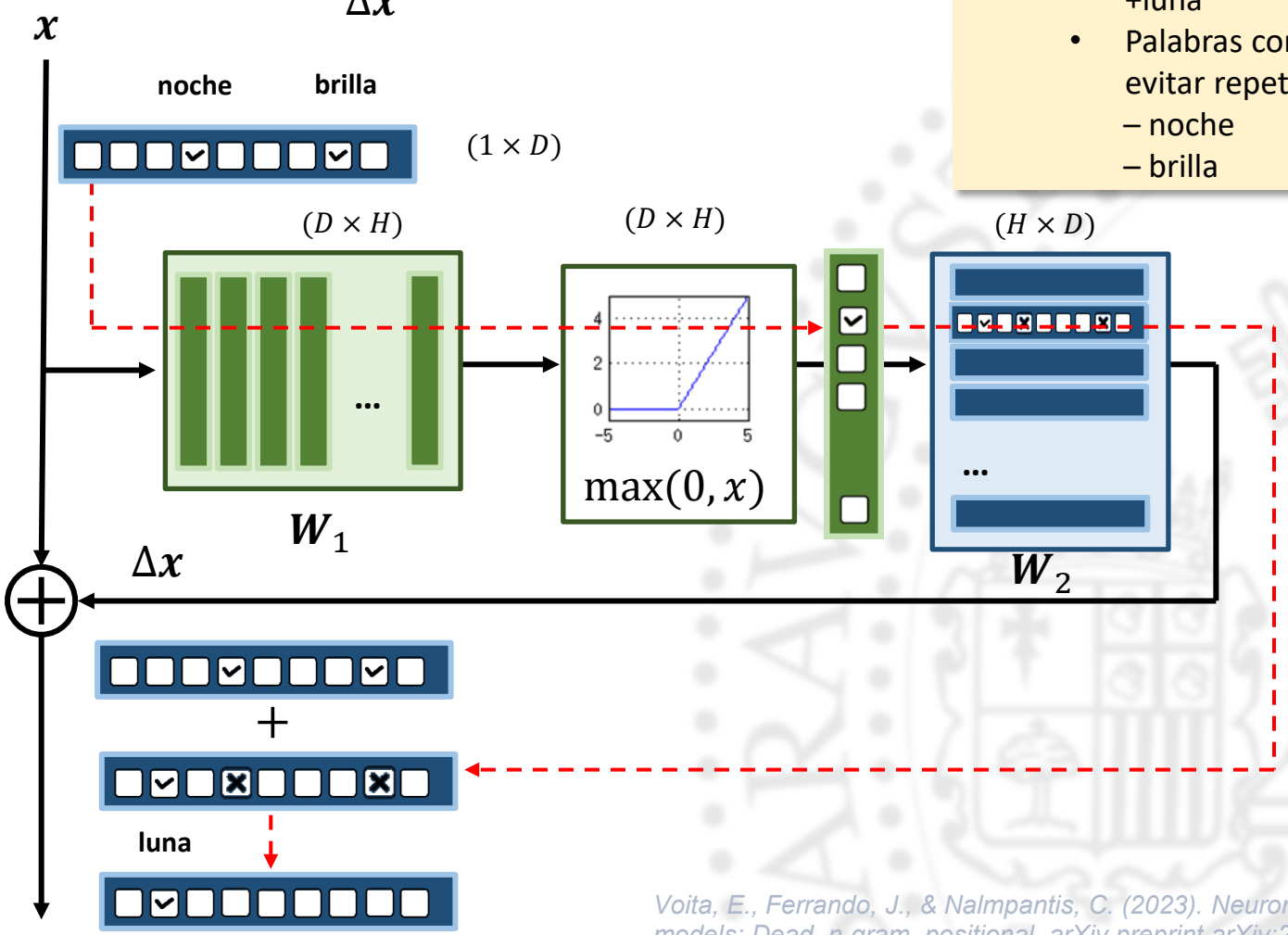


La salida de la ReLU puede interpretarse como los índices que se van a recuperar de la memoria $\boldsymbol{W}_2$

Transformer

$$f(\boldsymbol{x}) = \boldsymbol{x} + \underbrace{\max(\boldsymbol{x} \cdot \boldsymbol{W}_1, 0) \cdot \boldsymbol{W}_2}_{\Delta \boldsymbol{x}}$$

- La memoria $\boldsymbol{W}_2$ almacena los vectores $\Delta\mathbf{x}$ como un almacén
- Puede interpretarse como
  - Palabras/conceptos promocionados +luna
  - Palabras conceptos suprimidos para evitar repeticiones o incoherencias
    – noche
    – brilla



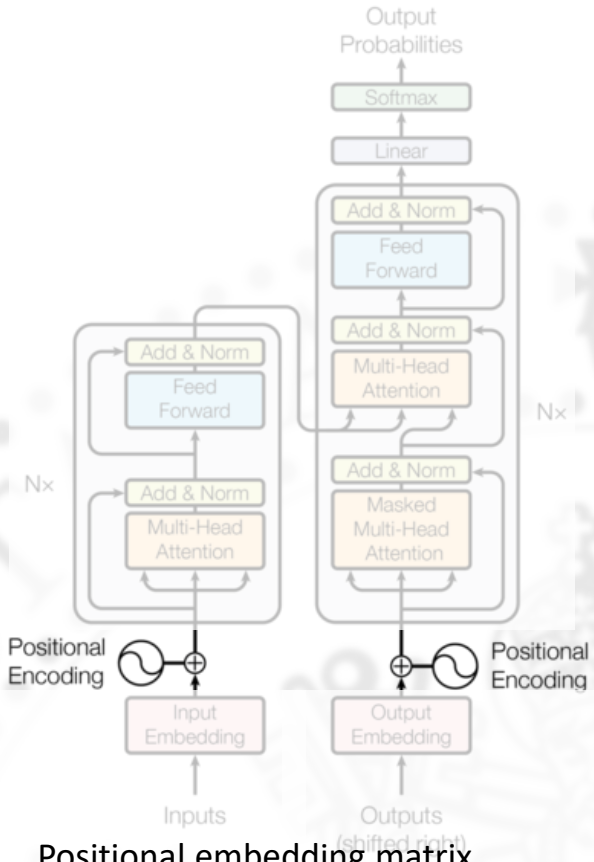
Voita, E., Ferrando, J., & Nalmpantis, C. (2023). Neurons in large language models: Dead, n-gram, positional. arXiv preprint arXiv:2309.04827.

Transformer

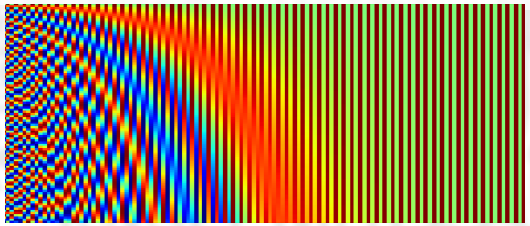
- **Arquitectura**
  - Residual layers
  - Layer normalization
  - Multi-head attention
  - Feed forward
  - **Positional embedding**
  - Embedding de entrada
  - Arquitectura general

Transformer

- **Embedding posicional**
  - Se proporciona a modo de referencia temporal
  - Permite
    - ubicar elementos en la secuencia
    - medir la distancia entre elementos de la secuencia



Positional embedding matrix



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

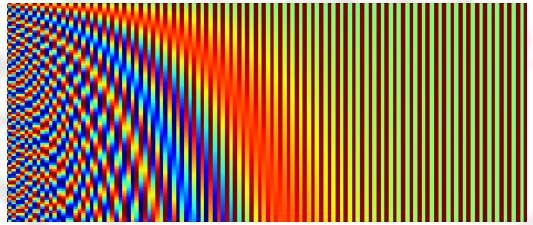
  - En otras modalidades de embedding posicional se deja que toda la matriz sean parámetros entrenables

Transformer

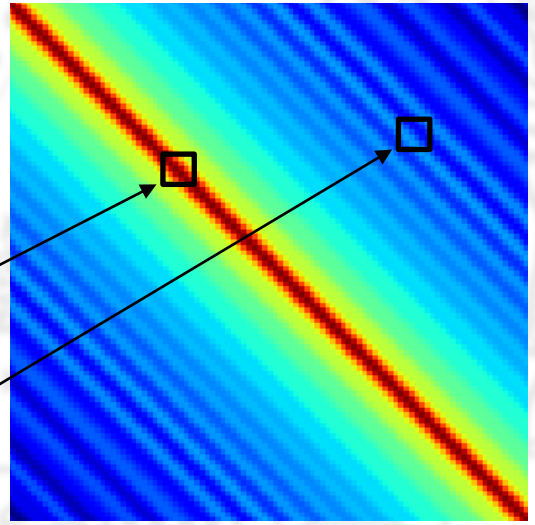
- **Embedding posicional**
  - Ejemplo ilustrativo
    - La figura de abajo indica el producto de

    de todos los vectores del embedding posicional

Positional embedding matrix



Positional embedding similarities



- Los vectores cercanos tienen alta similitud
- Los elementos alejados tienen menor similitud

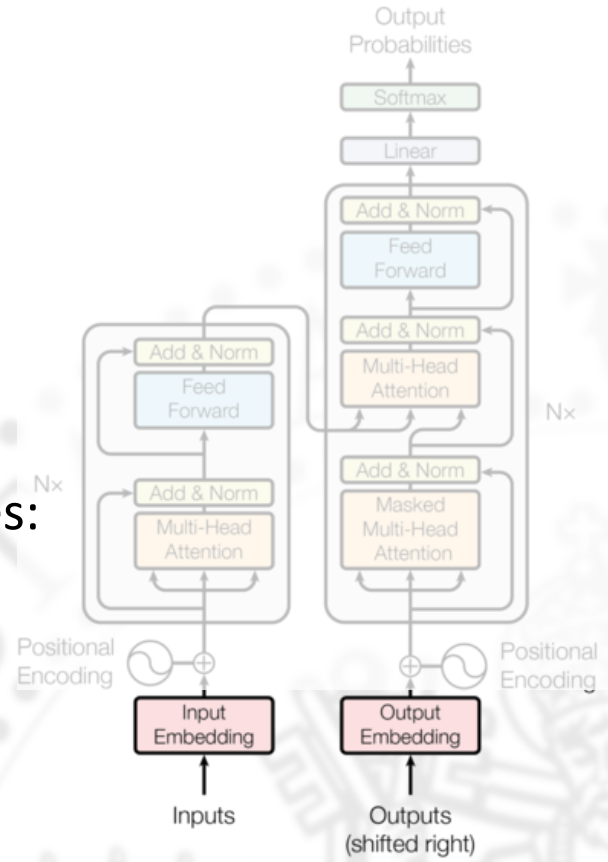
Transformer

- **Arquitectura**
  - Residual layers
  - Layer normalization
  - Multi-head attention
  - Feed forward
  - Positional embedding
  - **Embedding de entrada**
  - Arquitectura general

Transformers

- **Embeddings de entrada**
  - Hemos visto la interpretación simplificada de la representación mediante vectores y sus implicaciones:
    - Medida del parecido
    - Operaciones algebraicas -> analogías

palabra	rey	reina	gato	árbol	ráiz	gustará	odiaba
propiedades							
persona	☑	☑					
monarquía	☑	☑					
vegetal				☑	☑		
animal			☑				
masculino	☑		☑	☑			
femenino		☑			☑		
positivo						☑	
negativo							☑
verbo						☑	☑
presente							
pasado							☑
futuro						☑	



A change in meaning can be associated with changing one of those aspects; they resemble more a Chinese ideogram than a representation through a phonetic alphabet.

木 → 本
Árbol **Raíz**

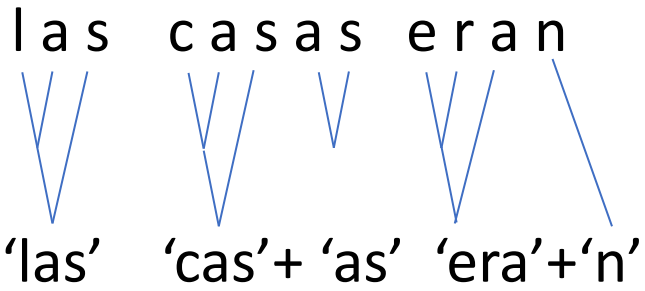
Transformers

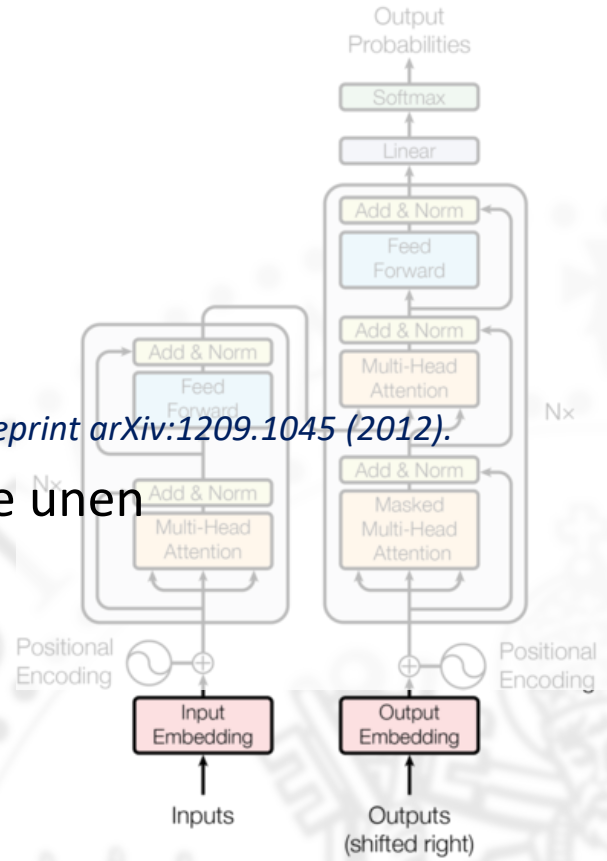
- **Tokenizadores: Byte-Pair Encoding (BPE)**

  Suarjaya, I. M. A. D. "A new algorithm for data compression optimization." arXiv preprint arXiv:1209.1045 (2012).

  – Se analizan los pares de símbolos más frecuentes y se unen

  – Formar tokens o unidades subpalabra más grande.



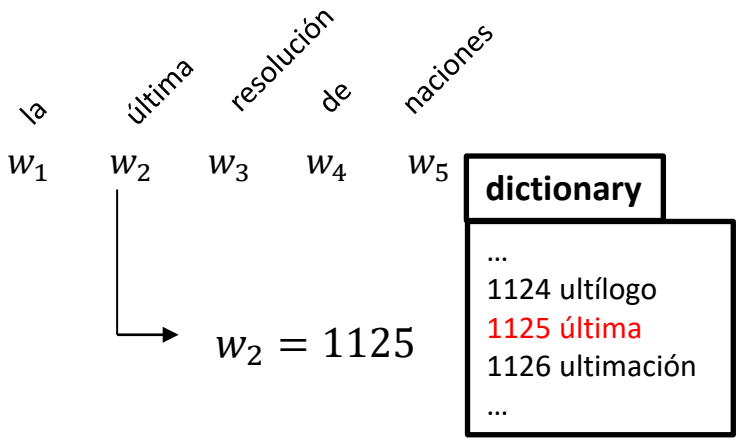


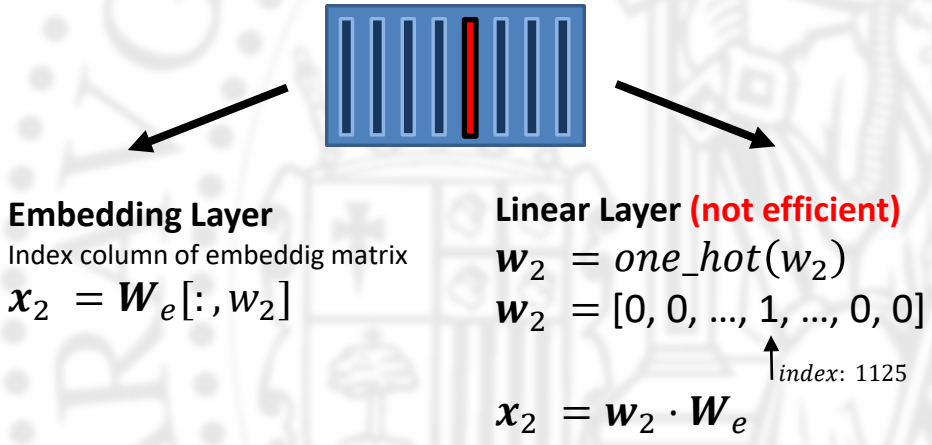
  – Depende mucho de los datos de entrenamiento

  - Si están bien adaptados tokens muy grandes define palabras enteras o casi enteras
  - Si no están bien representados en el entrenamiento las palabras se definen con más unidades

Transformer

- **Capa Embedding**
  - Almacena las representaciones de cada símbolo de entrada
    - En el caso de texto con LMs: tokens de un tokenizador
    - Aunque en realidad es una entrada discreta de tipo one_hot multiplicada por una capa lineal
      - La forma eficiente es evitar multiplicar por 0 y simplemente indexar.



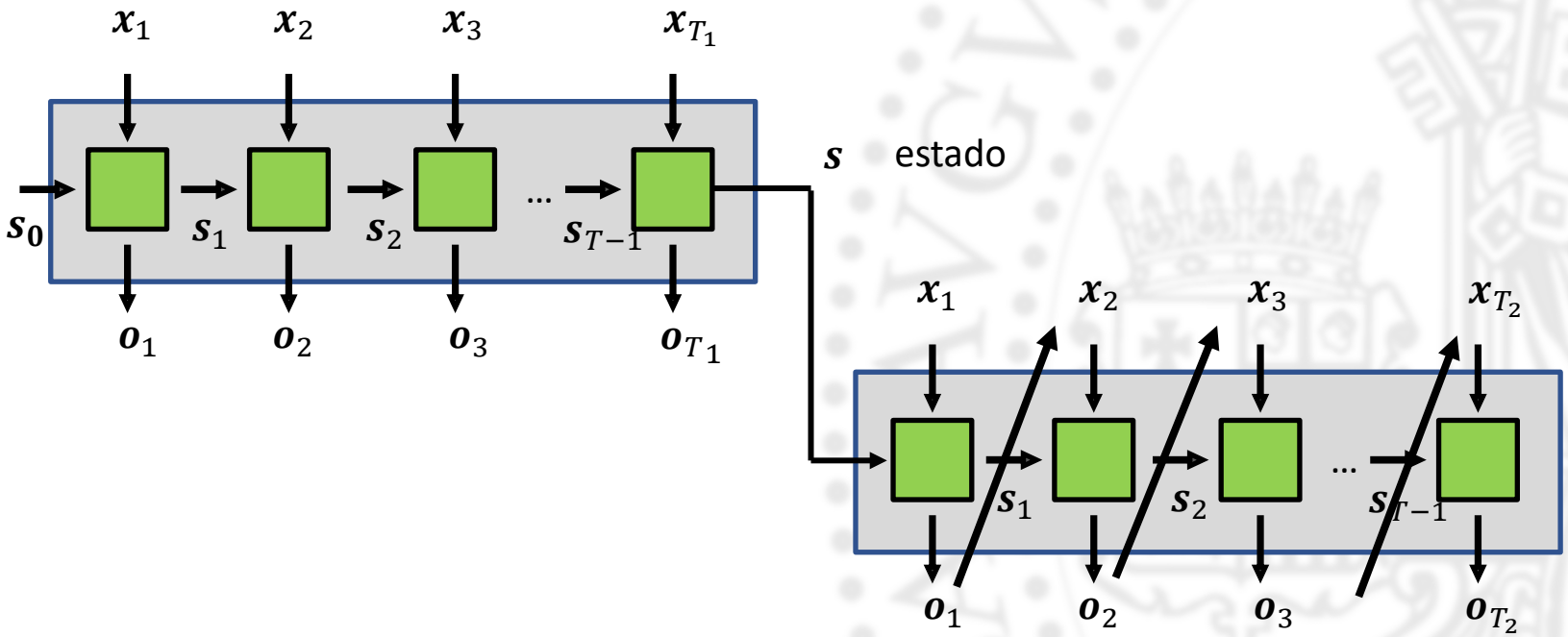


Transformer

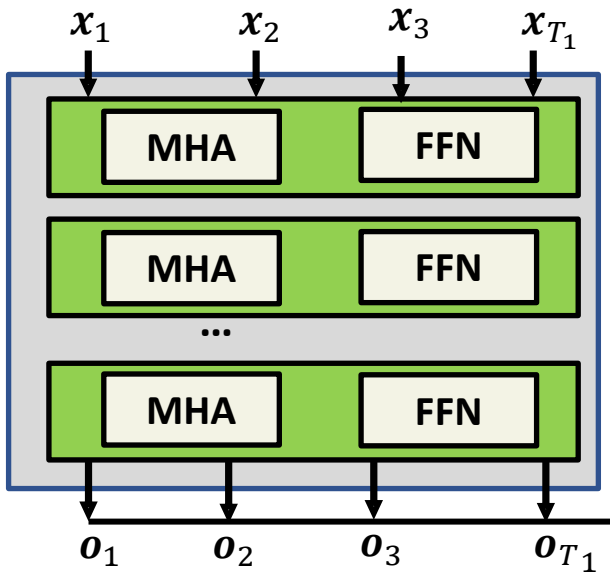
- **Arquitectura**
  - Residual layers
  - Layer normalization
  - Multi-head attention
  - Feed forward
  - Positional embedding
  - Embedding de entrada
  - **Arquitectura general**

Transformer

- **Seq2seq**
  - Los transformer comparte la idea de los modelos seq2seq
    - Se divide principalmente en dos bloques encoder y decoder
      - el problema de estos modelos era el cuello de botella que suponía almacenar todo en un vector de estado



Transformer

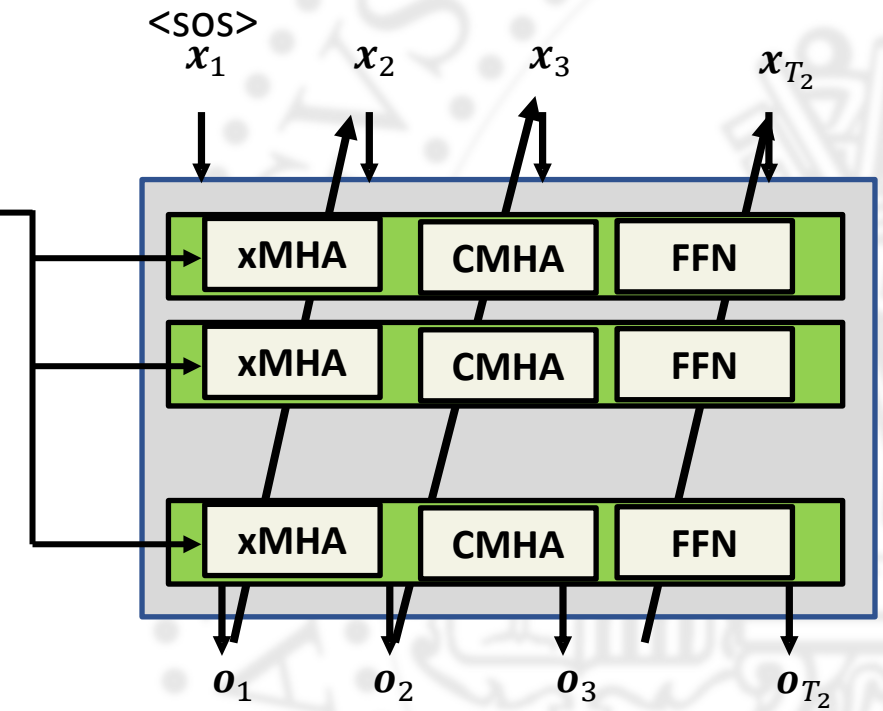


MHA: multihead self att

CMHA: causal multihead self att

xMHA: multihead cross-att

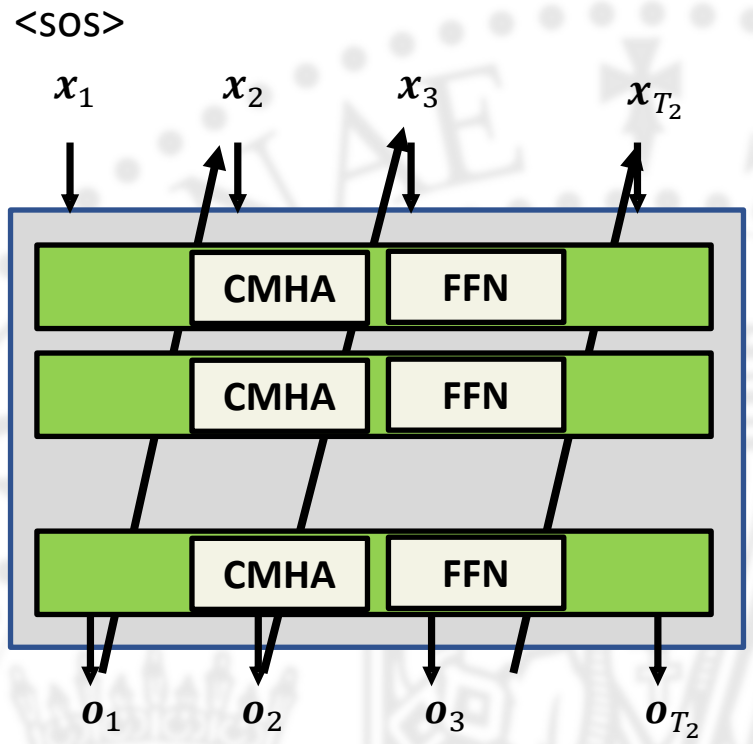
La salida completa del encoder va a todas las capas del decoder



Transformer

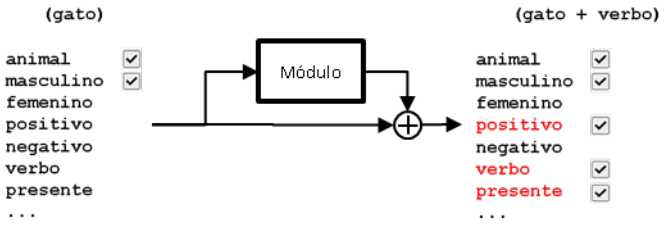
- **Decoder**
  - Un bloque decoder aislado no utiliza cross-att
  - La aplicación de este tipo de modelos es LM
    - Predecir una palabra dadas las anteriores
    - Para ello se realimenta de la salida anterior
    - Para generar cada nueva palabra el mecanismo de

    Self-att tiene acceso a TODA la secuencia previa
      - Alto coste
      - No es tan pronunciado el efecto de cuello de botella



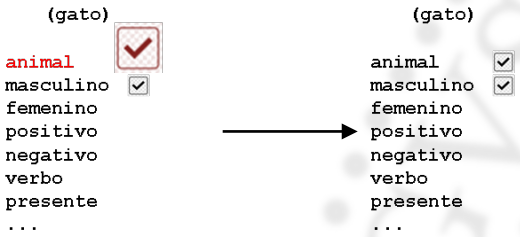
Transformer

- **Transformers (summary)**

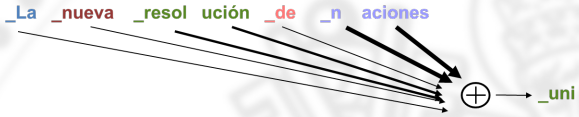


La información se procesa **incrementalmente** (residual)

Thought vectors (G. Hinton)



Es importante para la convergencia normalizar la escala

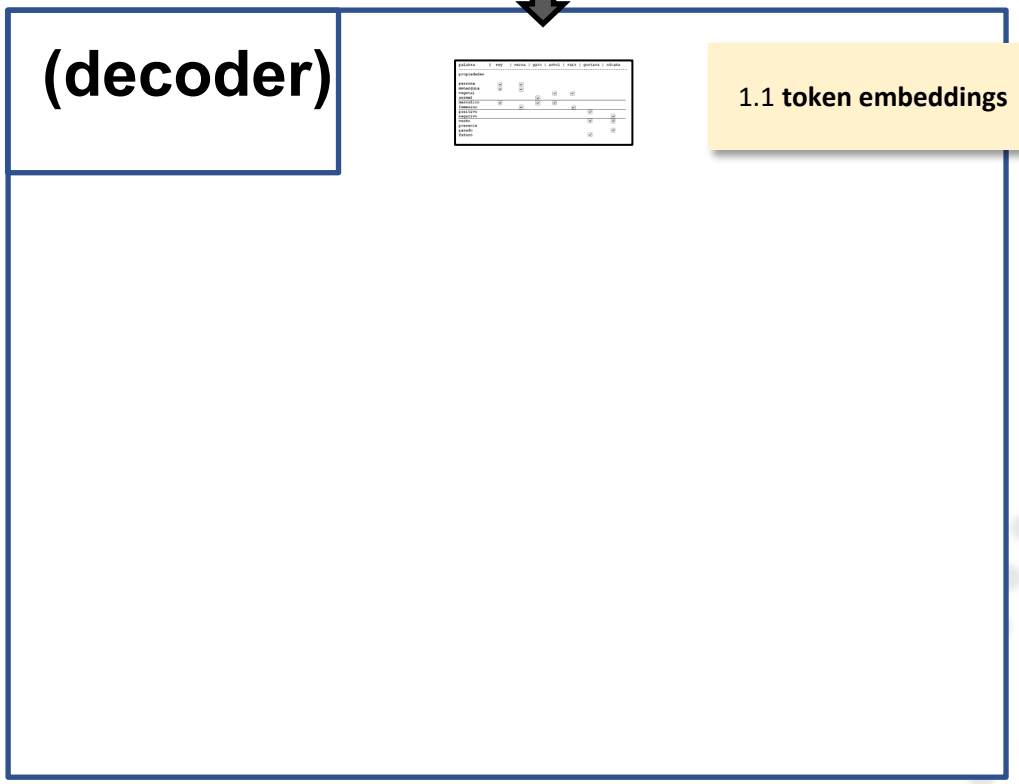


Dos tipos de módulos:
Procesado independiente (vectores)
- FFN

Mezcla de vectores
- **mecanismo de atenció** (self) (weighted sum)

Transformer

GPT-3 13B / LLAMA13B



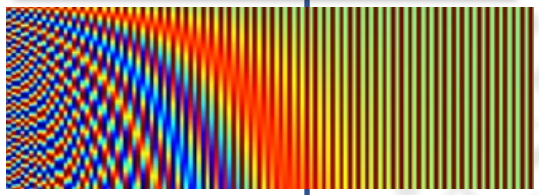
1.1 **token embeddings**

Transformer

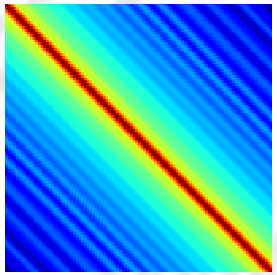
GPT-3 13B / LLAMA13B

(decoder)

1.2 **positional embedding**

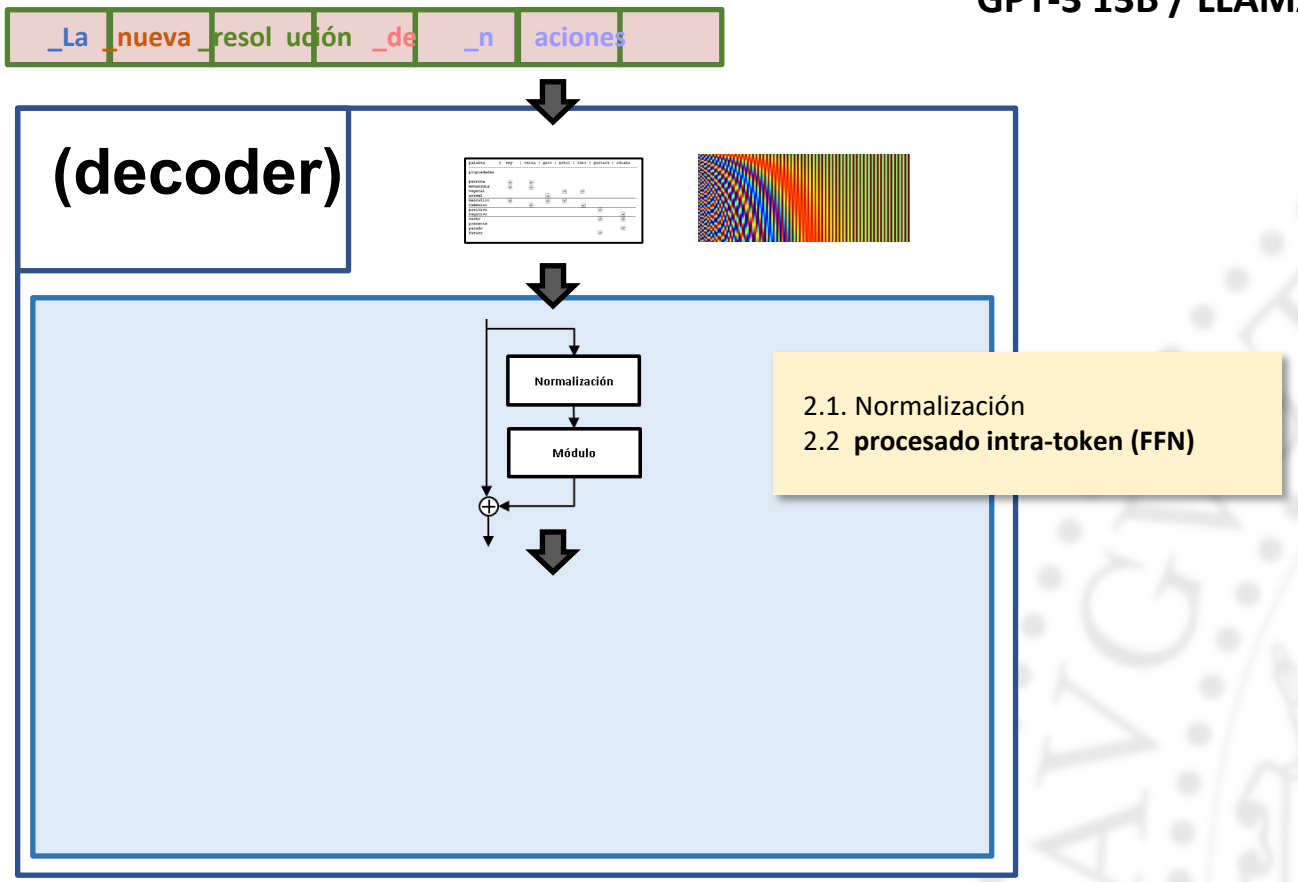


(parecido relativo)



Transformer

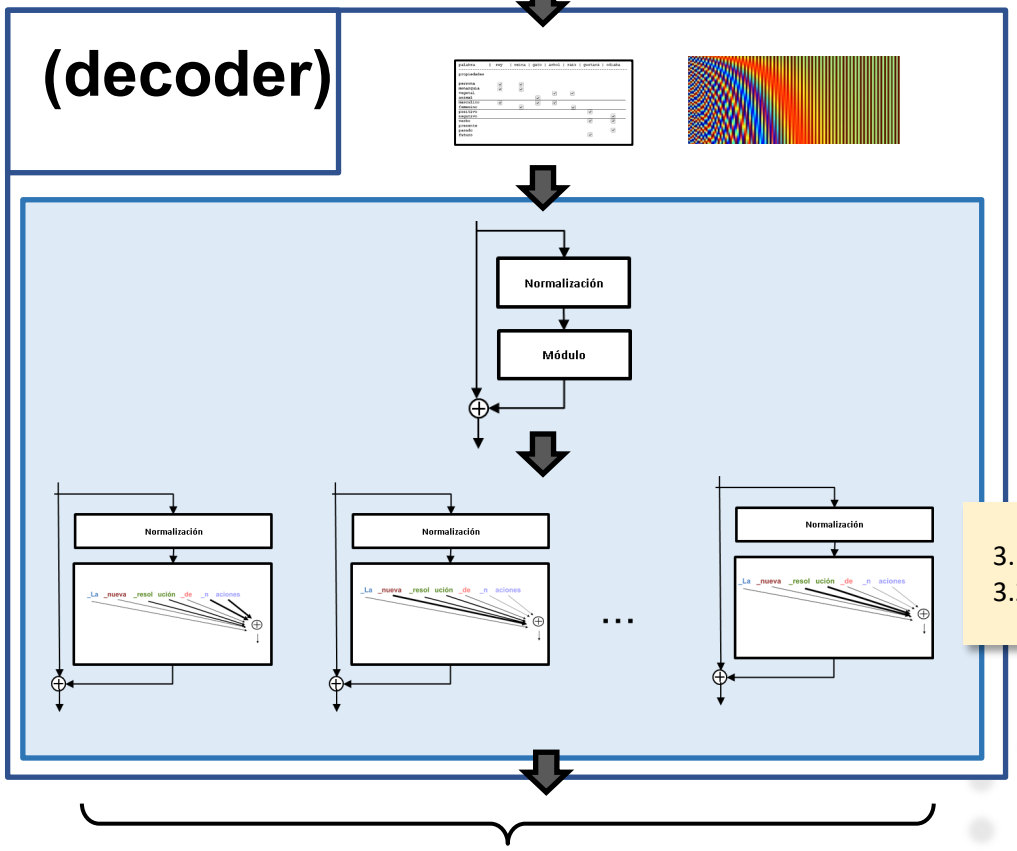
GPT-3 13B / LLAMA13B



2.1. Normalización
2.2 **procesado intra-token (FFN)**

Transformer

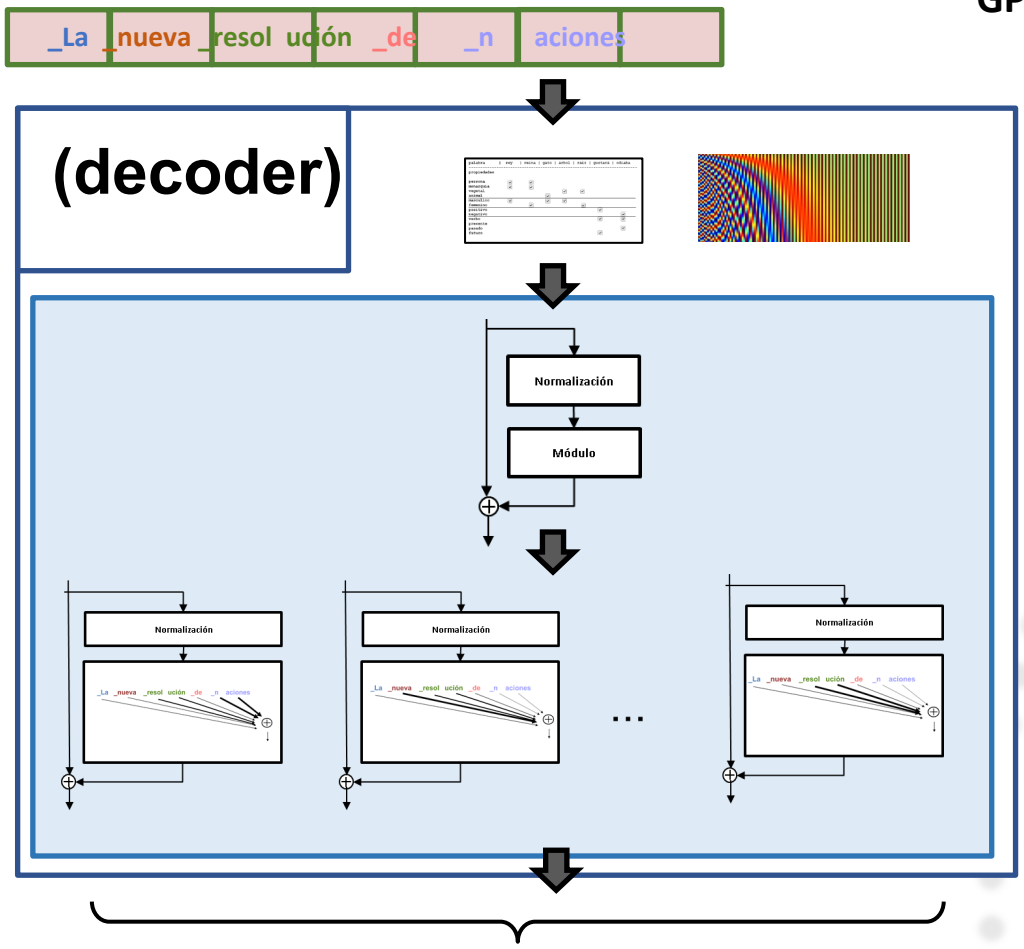
GPT-3 13B / LLAMA13B



3.1 Normalización
3.2 procesado **inter token (MSA)**

x 40 (heads / cabezales)

Transformer



GPT-3 13B / LLAMA13B

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

x 32 layers LLAMA7B
x 40 layers LLAMA13B
x 96 layers GPT-3 175B

Transformer

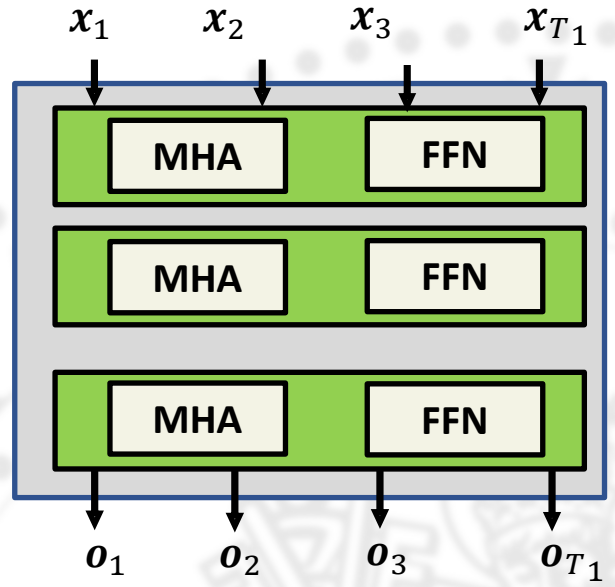
- **Modelos:**
  - Bert
  - Conformer
  - Vit
  - Whisper

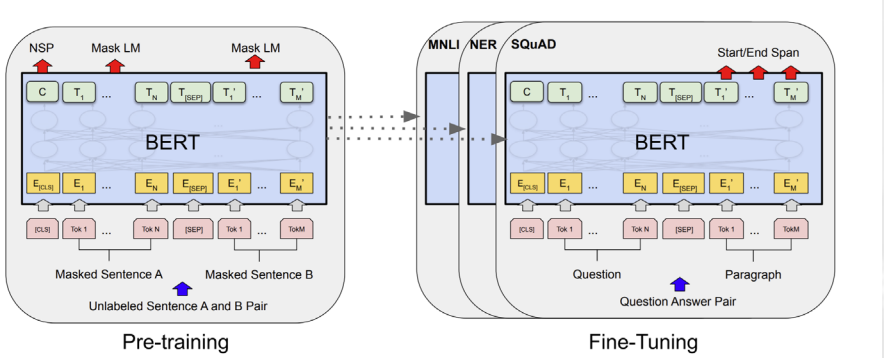
Transformer

- **BERT (Devlin 2018)**

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805..

  - Utiliza solo el bloque encoder (bidireccional)
  - BERT: Bidirectional Encoder Representations from Transformers
  - Su idea es muy utilizada actualmente como modelo para generar **representaciones**
    - Con su representación se han tratado multitud de tareas relacionadas con el texto:
      - Clasificación de tema, autoría, idioma, estado emocional



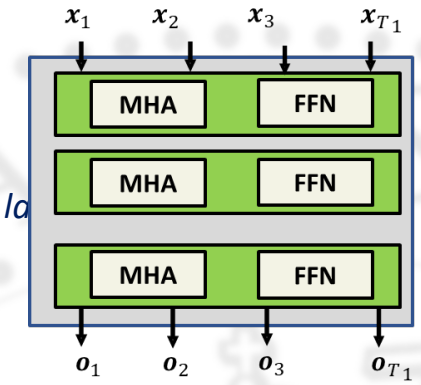


BERT BASE (L=12, H=768, A=12, Total Parameters=**110M**)
BERT LARGE (L=24, H=1024, A=16, Total Parameters=**340M**)

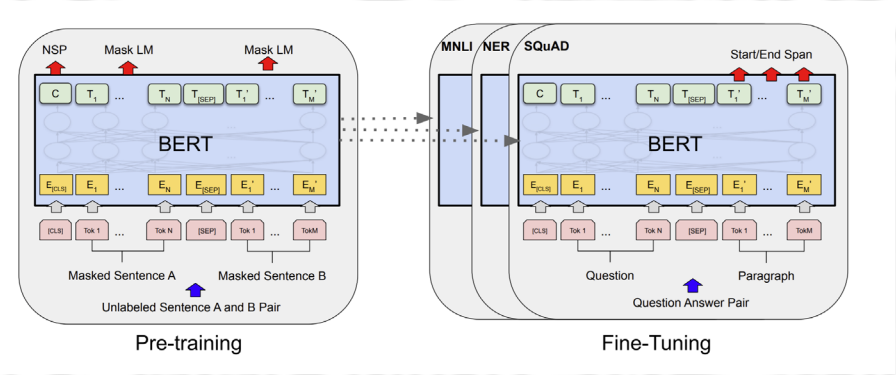
Transformer

- **BERT (Devlin 2018)**

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for la arXiv preprint arXiv:1810.04805..



  - **Tarea 1**
  - Predecir elementos sustuidos por <MASK> (denoising discreto)
  - Las máscaras se generan de esta forma:
    - 80% de los tokens reemplazados por '<MASK>'
      - Example: "My dog is <MASK>"
    - 10% reemplazados por tokens aleatorios
      - Example: "My dog is apple"
    - 10% se dejan intactos
      - Example: "My dog is hairy"



BERT BASE (L=12, H=768, A=12, Total Parameters=**110M**)
BERT LARGE (L=24, H=1024, A=16, Total Parameters=**340M**)

  - **Task 2**
  - Se entregan siempre 2 frases
  - La segunda tarea consiste en decir si las dos frase van seguidas en el texto de origen

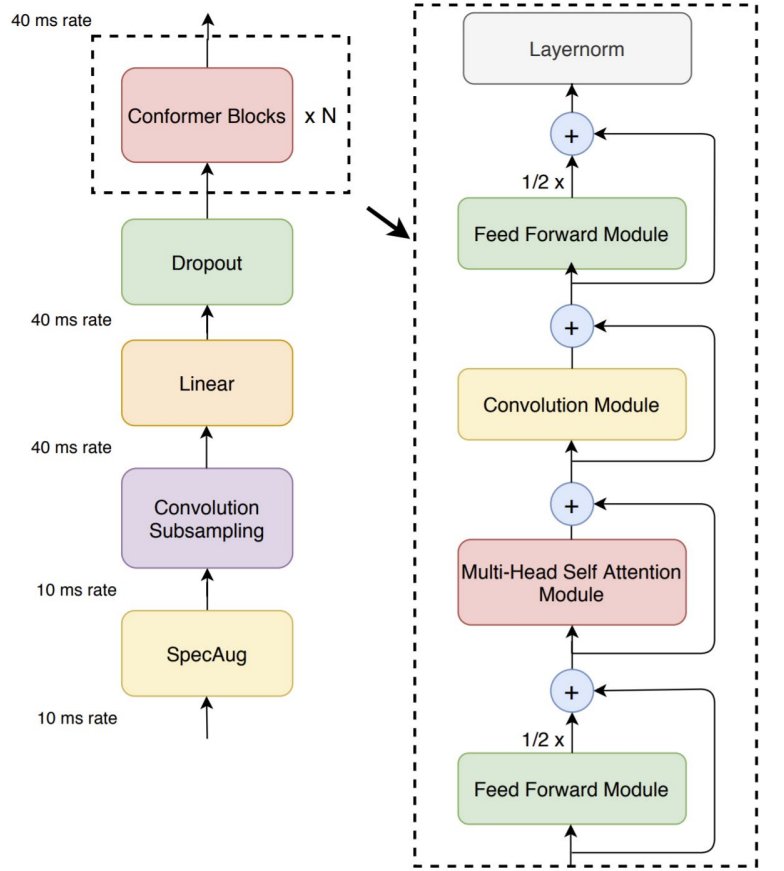
Transformer

- **Conformer(Gulati 2020)**

Gulati, A., Qin, J., Chiu, C. C., Parmar, N., Zhang, Y., Yu, J., ... & Pang, R. (2020). Conformer: Convolution-augmented Transformer for Speech Recognition. arXiv preprint arXiv:2005.08100.

  - combina CNNs y transformers
  - modela dependencias locales and globales
    - buenas prestaciones: ASR

Method	#Params (M)	WER Without LM		WER With LM	
		testclean	testother	testclean	testother
Hybrid					
Transformer [33]	-	-	-	2.26	4.85
CTC					
QuartzNet [9]	19	3.90	11.28	2.69	7.25
LAS					
Transformer [34]	270	2.89	6.98	2.33	5.17
Transformer [19]	-	2.2	5.6	2.6	5.7
LSTM	360	2.6	6.0	2.2	5.2
Transducer					
Transformer [7]	139	2.4	5.6	2.0	4.6
ContextNet(S) [10]	10.8	2.9	7.0	2.3	5.5
ContextNet(M) [10]	31.4	2.4	5.4	**2.0**	4.5
ContextNet(L) [10]	112.7	**2.1**	4.6	**1.9**	4.1
Conformer (Ours)					
Conformer(S)	10.3	**2.7**	**6.3**	**2.1**	**5.0**
Conformer(M)	30.7	**2.3**	**5.0**	**2.0**	**4.3**
Conformer(L)	118.8	**2.1**	**4.3**	**1.9**	**3.9**



Transformer

- **Automatic Speech Recognition ASR**

https://paperswithcode.com/sota/speech-recognition-on-librispeech-test-other

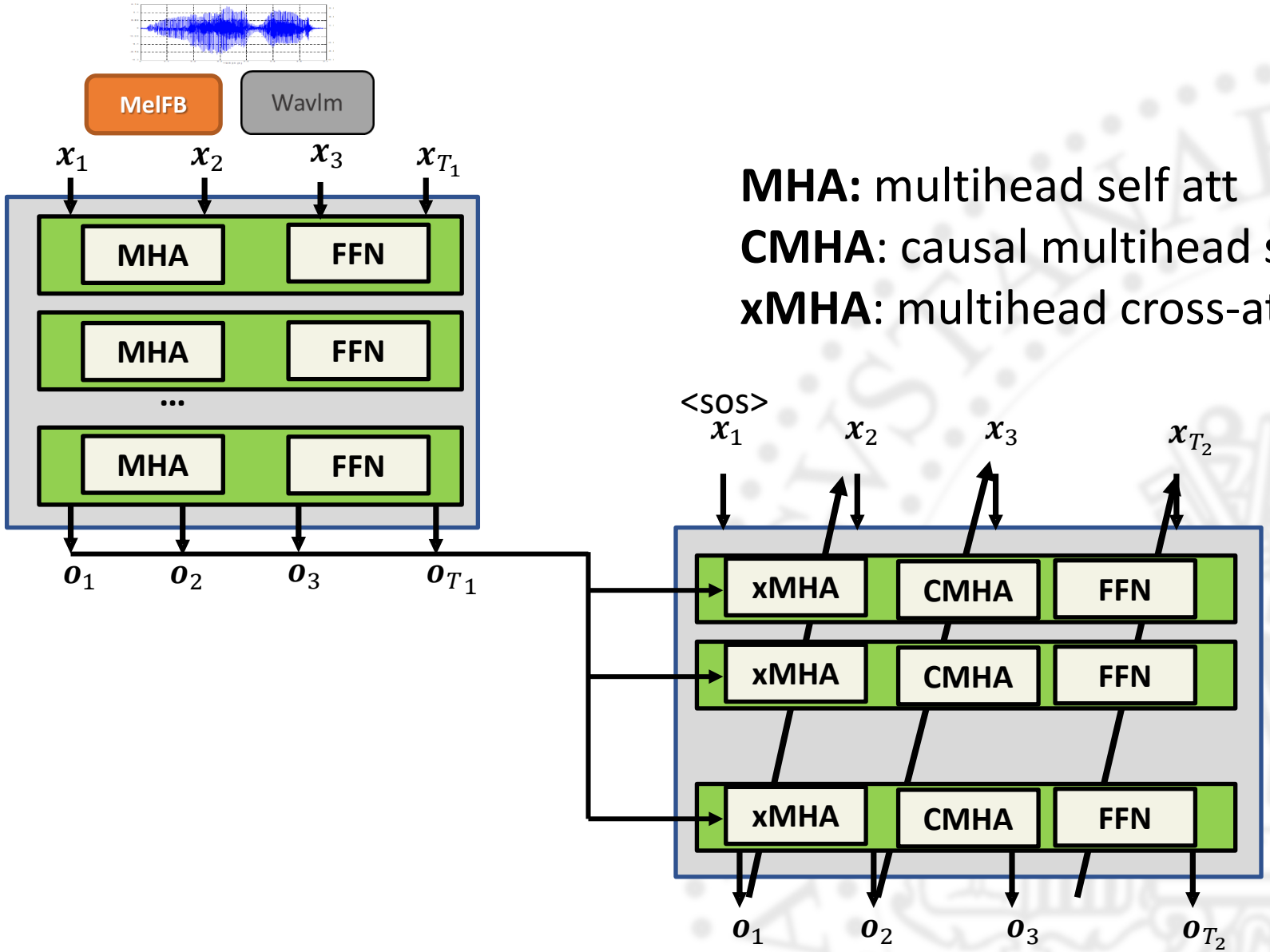
Speech Recognition on LibriSpeech test-other

Lectura de libros inglés (no nativos)

En menos de 5 years
de 13.2% a 2.5% WER

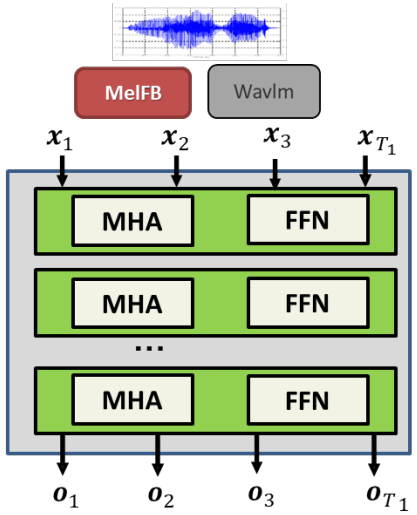
>10 years:
HMM+GMM: 22%

Transformer



MHA: multihead self att
CMHA: causal multihead self att
xMHA: multihead cross-att

Transformer



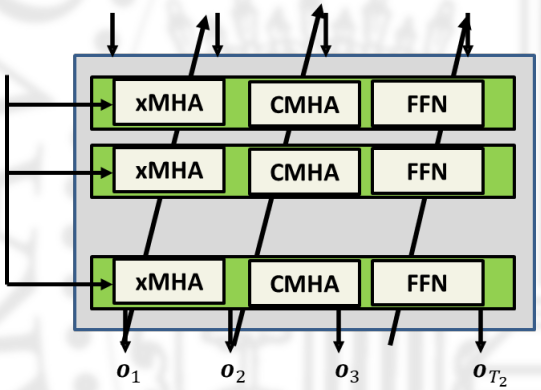

```python
class Encoder(torch.nn.Module):
    def __init__(self, nb_layers=6, seq_len=400, **kwargs):
        super().__init__()
        self.pos = torch.nn.Parameter(torch.randn(1, seq_len, kwargs['d_model']))
        self.att = torch.nn.ModuleList([SelfAttention(**kwargs) for _ in range(nb_layers)])
        self.ff = torch.nn.ModuleList([FeedForward(**kwargs) for _ in range(nb_layers)])

    def forward(self, x):
        b, t, d = x.shape
        x = x + self.pos[:, :t, :]
        for att, ff in zip(self.att, self.ff):
            x = x + att(x)
            x = x + ff(x)
        return x
```

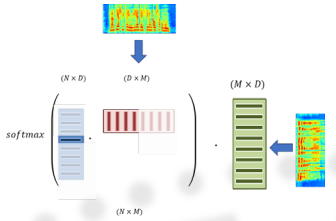
Transformer

```python
class Decoder(torch.nn.Module):
    def __init__(self, nb_layers=6, seq_len=400, **kwargs):
        super().__init__()
        self.pos = torch.nn.Parameter(torch.randn(1, seq_len, kwargs['d_model']))
        self.att = torch.nn.ModuleList([CausalSelfAttention(**kwargs) for _ in range(nb_layers)])
        self.cross_att = torch.nn.ModuleList([CrossAttention(**kwargs) for _ in range(nb_layers)])
        self.ff = torch.nn.ModuleList([FeedForward(**kwargs) for _ in range(nb_layers)])

    def forward(self, x, enc):
        b, t, d = x.shape
        x = x + self.pos[:, :t, :]
        for att, cross_att, ff in zip(self.att, self.cross_att, self.ff):
            x = x + att(x)
            x = x + cross_att(x, enc)[0]
            x = x + ff(x)
        return x
```

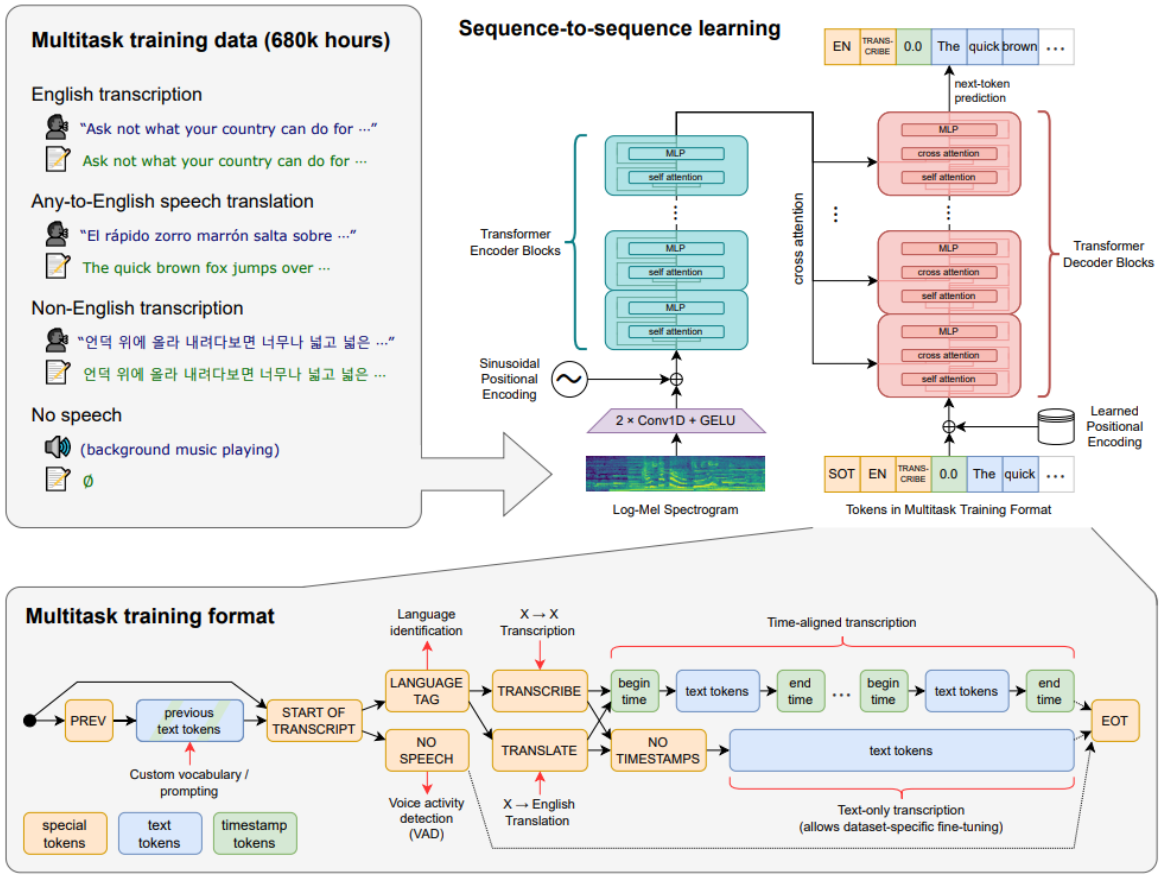


Transformer

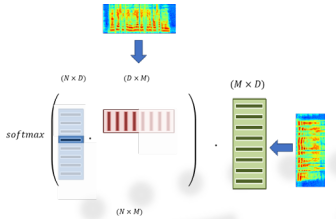


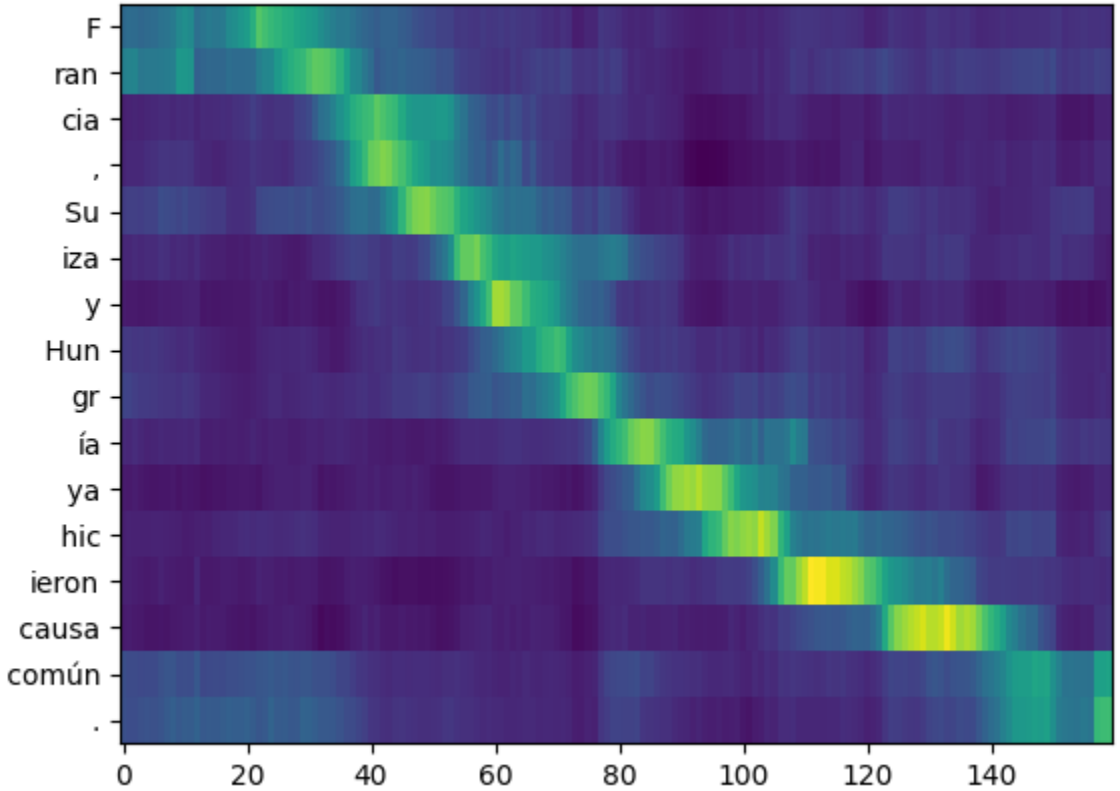
- **Whisper**

Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C. and Sutskever, I., 2023, July. Robust speech recognition via large-scale weak supervision. In International conference on machine learning (pp. 28492-28518). PMLR.



Transformer





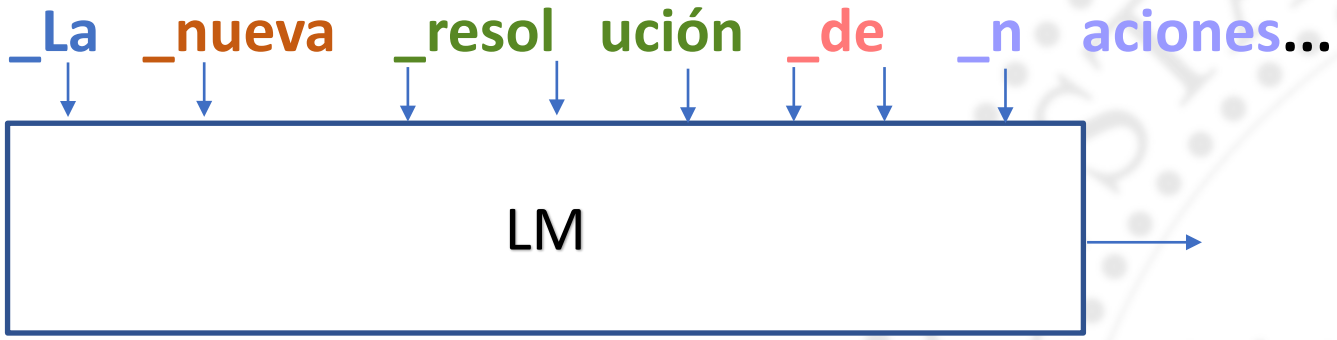
whisper medium model

Transformer

- Técnicas de generación:
  - Muestreo
  - Greedy
  - N-best

Transformer

- **Ejemplo LLAMA 7B 32k tokens**

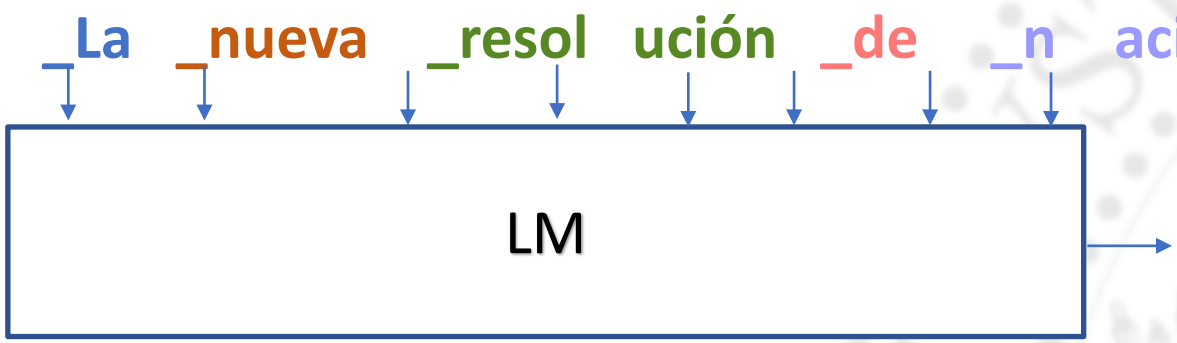


p(unidas | _La, _nueva, _resol, ución, _de, _n, aciones)

token	prob
_un	0.48
_y	0.04
_en	0.03
_Un	0.03
_que	0.02
,	0.02
…	

Transformer

- **Ejemplo LLAMA 7B 32k tokens**



(two steps)

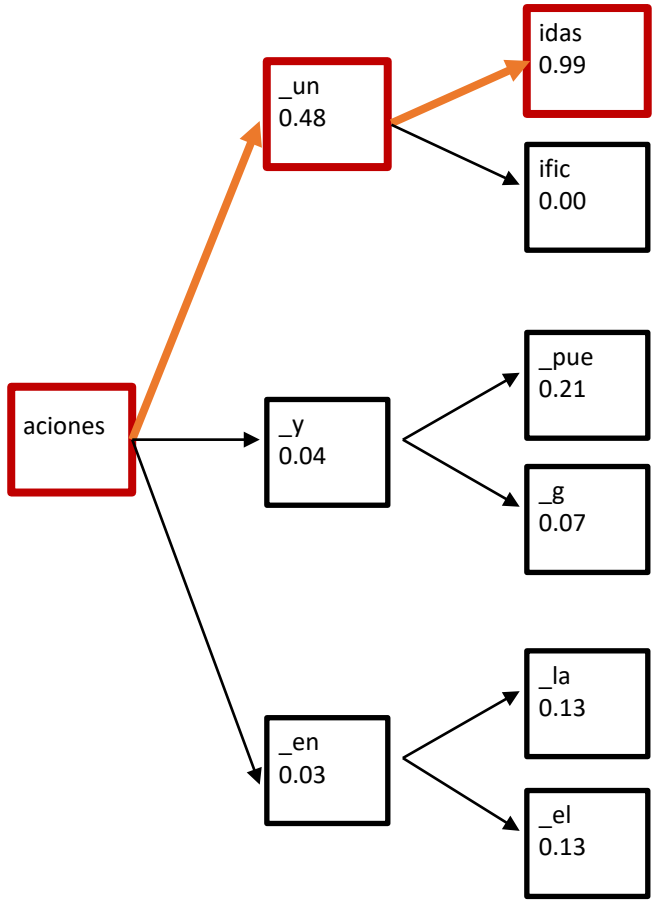
```
--------------------
_un (0.48) ->
        idas        (0.99)
        ific        (0.00)
        ip          (0.00)
        as          (0.00)
        id          (0.00)
--------------------
_y (0.04) ->
        _pue        (0.21)
        _g          (0.07)
        _raz        (0.04)
        _nacional   (0.04)
        _de         (0.04)
--------------------
_en (0.03) ->
        _la         (0.13)
        _el         (0.13)
        _conflic    (0.05)
        _las        (0.04)
        _Europa     (0.03)
```

  - Cómo generamos ejemplos?
    - Greedy
    - Sampling
    - Beam search

Transformer

```
--------------------
_un (0.48) ->
       idas         (0.99)
       ific         (0.00)
       ip           (0.00)
       as           (0.00)
       id           (0.00)
--------------------
_y (0.04) ->
       _pue         (0.21)
       _g           (0.07)
       _raz         (0.04)
       _nacional    (0.04)
       _de          (0.04)
--------------------
_en (0.03) ->
       _la          (0.13)
       _el          (0.13)
       _conflic     (0.05)
       _las         (0.04)
       _Europa      (0.03)
```

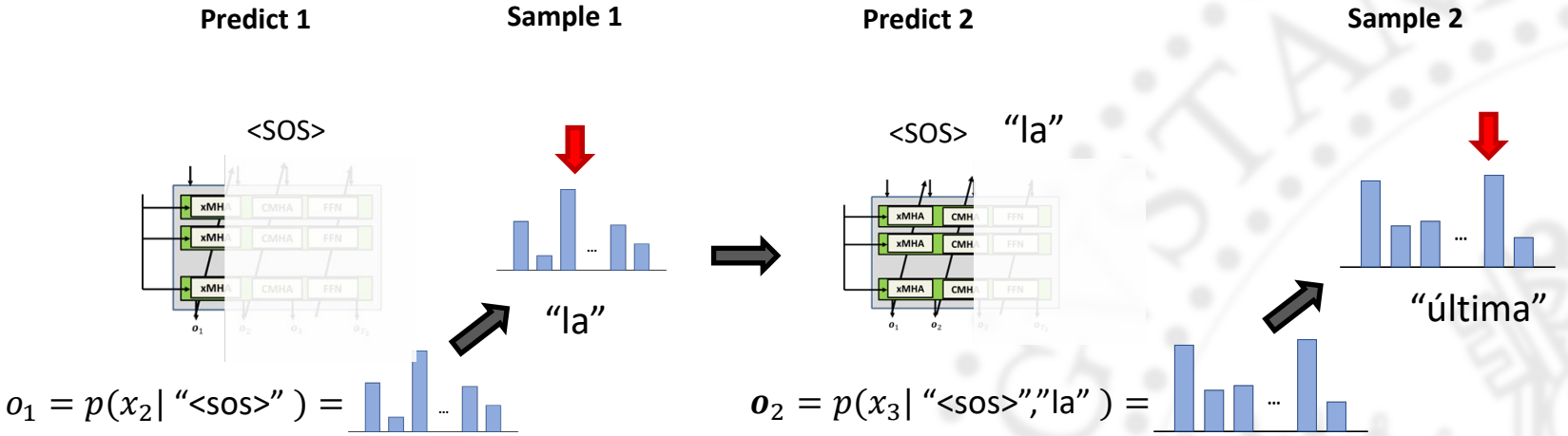
- **Greedy generation**
  - Se podrían representar todas las posibilidades de continuación



- Seleccionar después sobre las opciones la que mejor $p(x_t|x_1^{t-1})$
- <SOS> _La _nueva _resol ución _de _n aciones _un idas

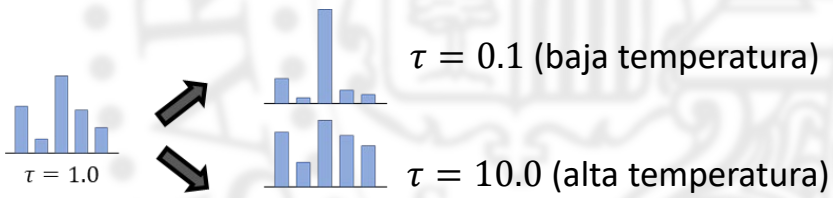
Transformer

- **Random generation**
  - Muestreo de la distribución: $x_t \sim p(x_t | x_1^{t-1})$



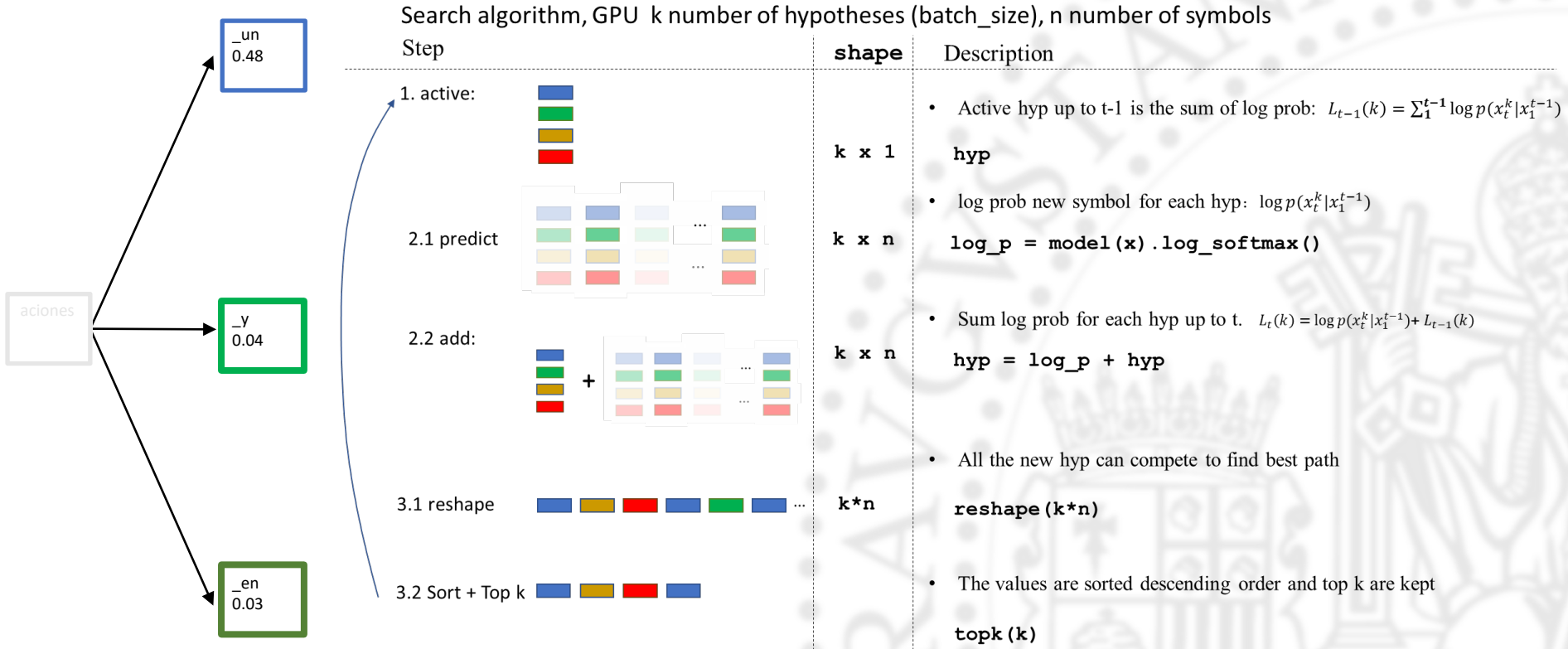
  - Las distribuciones pueden ajustarse para favorecer los símbolos más probables (low temperature) o distribuciones más uniformes (alta)
    - Se escala el vector (logits) antes de la capa softmax

$$softmax_T(\boldsymbol{x}, \tau) = \frac{\exp x_c / \tau}{\sum_{c'} \exp x_{c'} / \tau}$$



Transformer

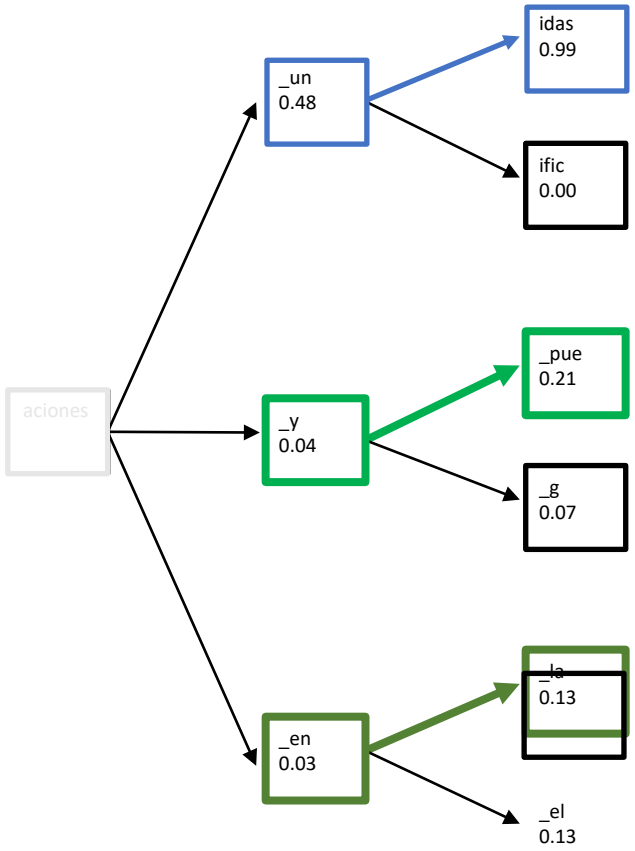
- **Beam search: múltiples hipótesis + beam**

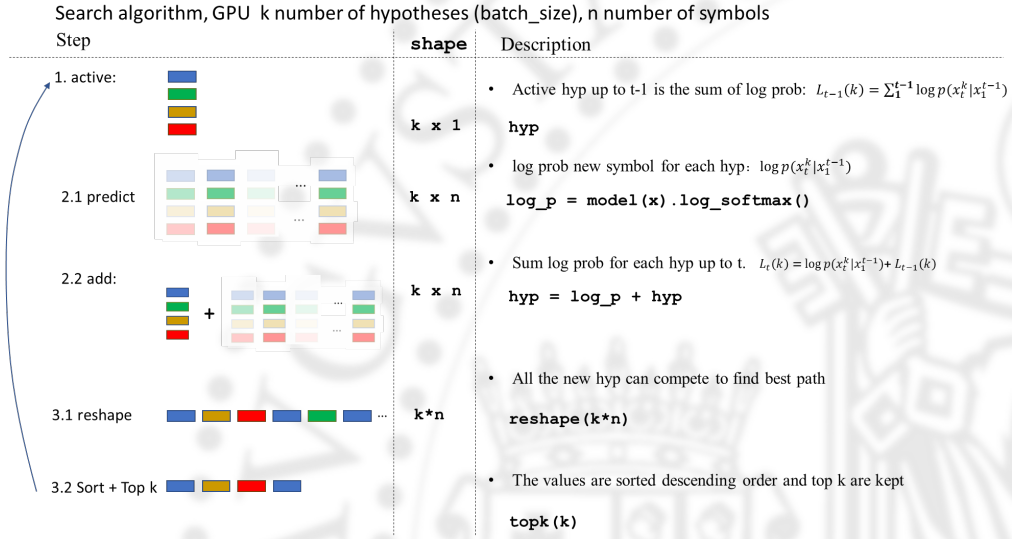


Transformer

Beam search

- *Multiple active hypotheses + beam*





*For example with **beam 3**, active hyp:*

***_un + idas** : 0.48 * 0.99 = 0.4752*

***_y + _pue**: 0.04 * 0.21 = 0.0084*

***_en + _la**: 0.03 * 0.13 = 0.0039*

Transformer

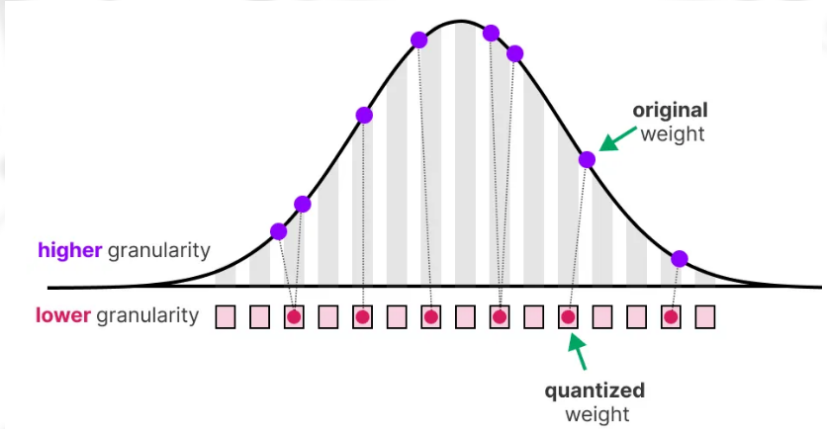
- Reentrenamiento, y optimizaciones:
  - QAT
  - LORA
  - Implementación: einops, rearrange
  - Flash Attention
  - Rotary Positional Embedding
  - Sliding window attention
  - MoE

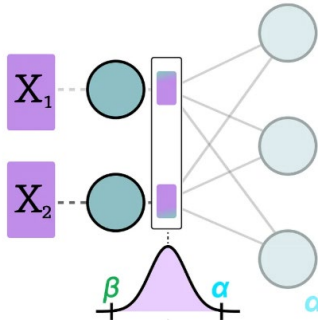
Transformer

- **Quantification**

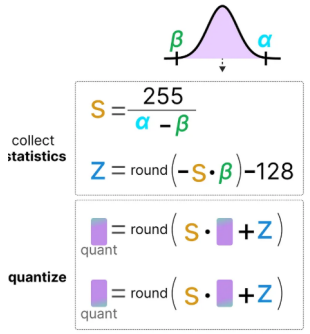
https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

  - Solución: bajos recursos computacionales disponibles
    - memoria de GPU
    - **cuantificar los pesos del modelo**
    - número de bits inferior a fp32 o fp16 (estándar pytorch)
    - Valores típicamente empleados fp8, int8, fp4, int4, o incluso 1 bit y 1.5 bitshttps://ollama.com/library/gemma3/tags





recopilar estadísticas



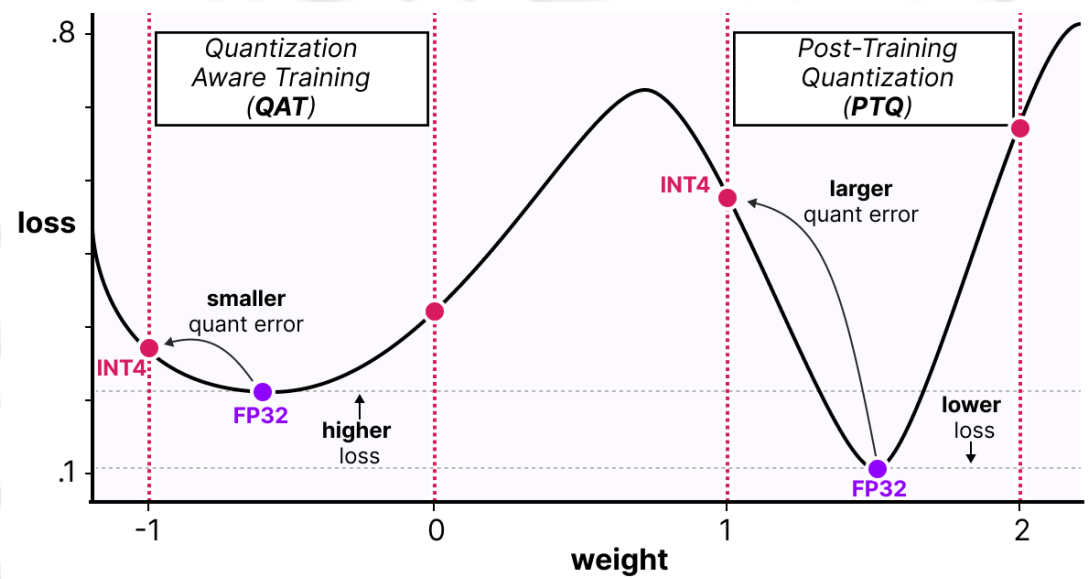
cuantificar

Transformer

- **Quantification**

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

  - Se puede cuantificar un modelo simplemente transformando cada peso
    - Da malos resultados
  - Lo ideal es reentrenar el modelo al menos unos pasos mediante métodos denominados **QAT: Quantification Aware Training**
  - Una de las técnicas más utilizadas es STE




```python
import torch
def STE(x):
    rounded = torch.round(x)  # rounds to nearest integer
    return x + (rounded - x).detach()  # detach blocks gradient backpropagation
```

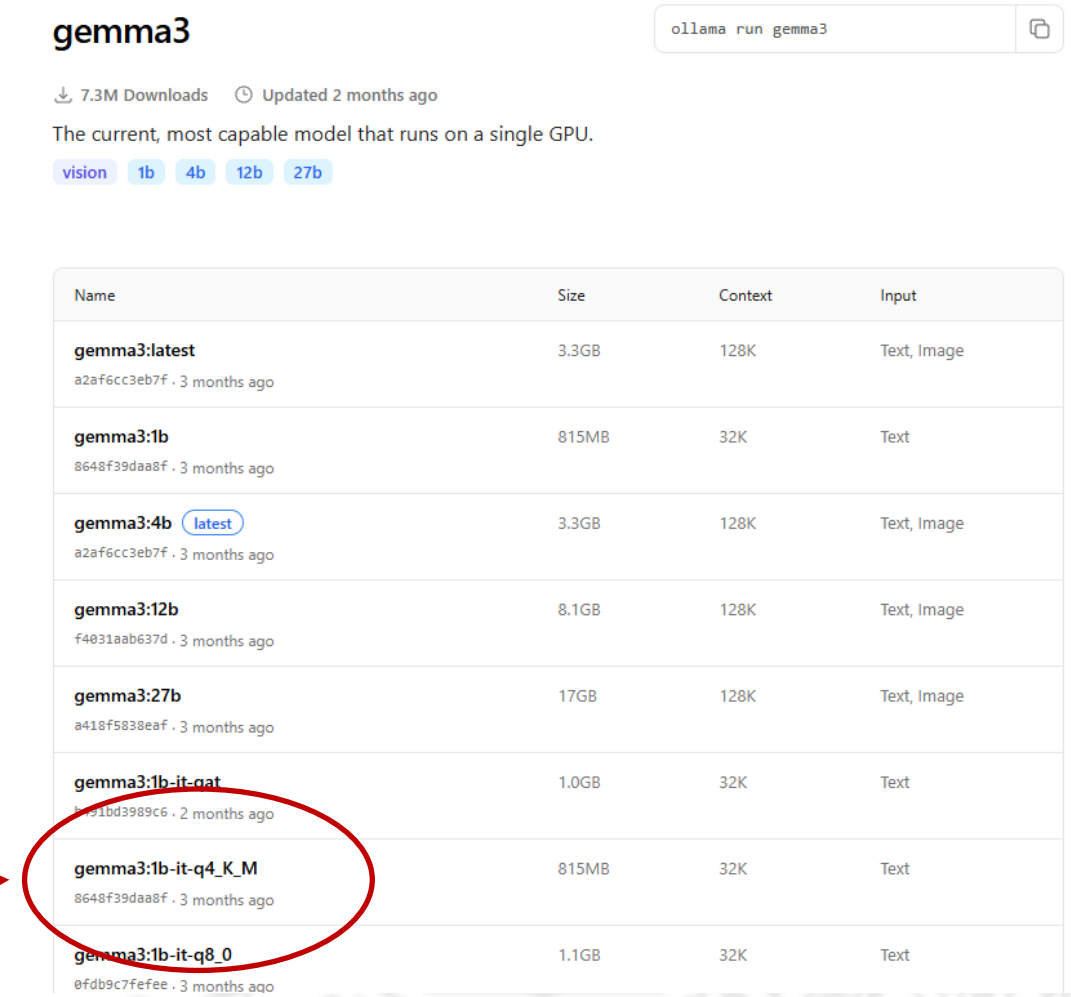
Transformer

- **Quantification**

https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-quantization

  - Modelos opensource
    - Algunos desarrolladores empiezan a proporcionar las versiones cuantificadas directamente por ellos

Se puede ejecutar en una RTX 3090/4090 !



gemma3

ollama run gemma3

7.3M Downloads · Updated 2 months ago

The current, most capable model that runs on a single GPU.

vision · 1b · 4b · 12b · 27b

Name	Size	Context	Input
gemma3:latest a2af6cc3eb7f · 3 months ago	3.3GB	128K	Text, Image
gemma3:1b 8648f39daa8f · 3 months ago	815MB	32K	Text
gemma3:4b latest a2af6cc3eb7f · 3 months ago	3.3GB	128K	Text, Image
gemma3:12b f4031aab637d · 3 months ago	8.1GB	128K	Text, Image
gemma3:27b a418f5838eaf · 3 months ago	17GB	128K	Text, Image
gemma3:1b-it-qat b91bd3989c6 · 2 months ago	1.0GB	32K	Text
gemma3:1b-it-q4_K_M 8648f39daa8f · 3 months ago	815MB	32K	Text
gemma3:1b-it-q8_0 0fdb9c7fefee · 3 months ago	1.1GB	32K	Text

https://ollama.com/library/gemma3/tags

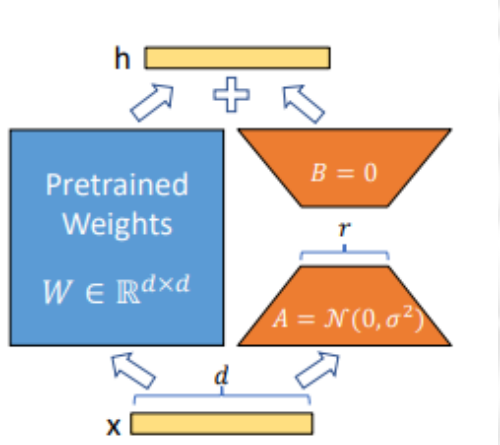
Transformer

- **LORA Low-rank adaptation**

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W., 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.

- Cada vez es más habitual tener acceso a grandes modelos preentrenados
- Si no funcionan perfectamente en nuestra tarea podemos pensar en
  - adaptar el modelo con nuestros datos -> finetuning
- El problema es que es muy difícil disponer de tantos datos como serían necesarios
- Una solución muy eficiente es la adaptación LORA
  - La adaptación consiste en dejar bloqueada la matriz **W**
  - Lo que realmente se entrena con fine-tuning
    - Matrices **A** y **B** de forma que:

$$\boldsymbol{W'} = \boldsymbol{W} + \boldsymbol{A} \cdot \boldsymbol{B}$$



A y **B** son matrices de bajo rango (d x r) (r x d)

Por ejemplo d puede ser 1024 por lo tanto **W** tiene más de 1M de parámetros

r puede ser tan pequeño como 8, 16,… así que **A** y **B** no sumarían más de unos miles de parámetros

Transformer

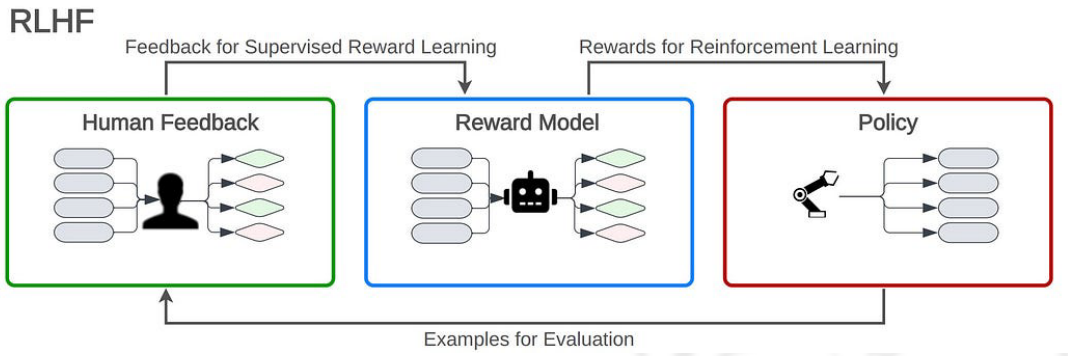
- **RLHF Reinforcement Learning Human Feedback**

https://docs.unsloth.ai/basics/gemma-3-how-to-run-and-fine-tune

https://docs.unsloth.ai/basics/qwen3-how-to-run-and-fine-tune

(muchos otros ejemplos para colab)

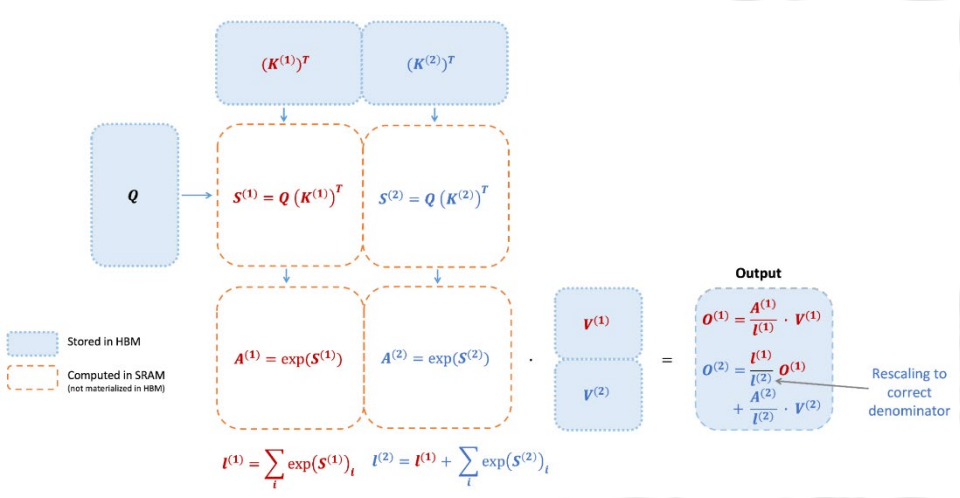
  - RLHF un conjunto de técnicas para entrenar de forma automática modelos basados en las preferencias de etiquetadores humanos
  - Conseguir etiquetas para entrenar un LLM es muy costoso
    - Se sustituye la generación de etiquetas para un gran dataset por el entrenamiento de un modelo que acierte lo suficiente en clasificar lo que es una respuesta válida y lo que no
    - Ahora se puede entrenar la red grande (LLM) utilizando las etiquetas generadas por el segundo modelo, esto no tiene coste de etiquetado



Transformer

- **Flash attention, Flash attention2**

Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in Neural Information Processing Systems, 35, 16344-16359.
Dao, T. (2023). Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv:2307.08691...



  - La matriz de atención no se calcula completamente:
    - Se calculan matrices más pequeñas para cada query (fila) -> salida
  - Muy dependiente de la arquitectura GPU
    - Se puede ganar mucho tiempo en GPUs recientes, especialmente en precisión como float16

Transformer

- **Einops, rearrange**
  - Rearrange permite ejecutar varias operaciones permute, reshape en una llamada

```
q = self.to_q(q).view(b, -1, h, d)
k = self.to_k(k).view(b, -1, h, d)
v = self.to_v(v).view(b, -1, h, d)

q = q.permute(2, 0, 1, 3).contiguous().view(b*h, -1, d)
k = k.permute(2, 0, 1, 3).contiguous().view(b*h, -1, d)
v = v.permute(2, 0, 1, 3).contiguous().view(b*h, -1, d)
```

```
q = rearrange(self.to_q(q), ‘b t (h d) -> (b h) t d’, h=h)
k = rearrange(self.to_k(k), 'b t (h d) -> (b h) t d’, h=h)
v = rearrange(self.to_v(v), 'b t (h d) -> (b h) t d’, h=h)
```

  - Einops sigue la notación de tensores de Einstein y permite calcular productos, reducciones…

```
scores = torch.matmul(q, k.transpose(-1,-2))
```

```
scores = torch.einsum('bihd,bjhd->bhij', q, k)
```

https://einops.rocks/pytorch-examples.html

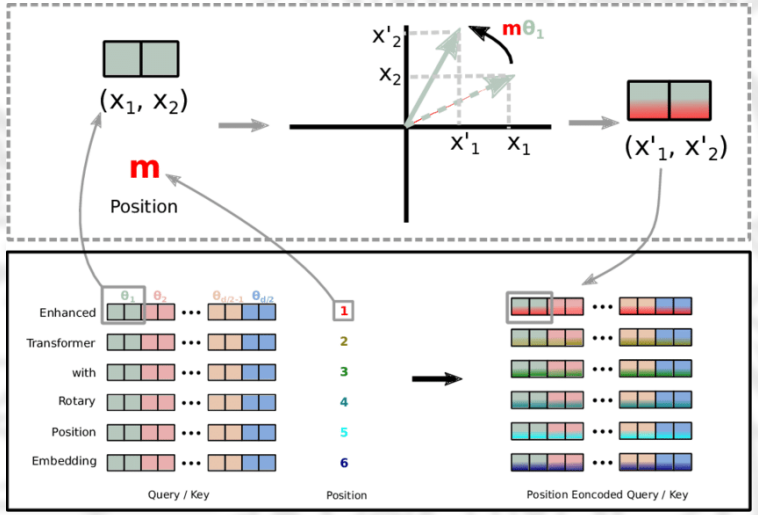
Transformer

- **Rotary Positional Embedding (RoPE)**

  Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864..

  - Las secuencias q,k se modulan con una exponencial compleja
    - El producto escalar de los vectores complejos es equivalente al producto escalar de los vectores originales y un término dependiente de la **diferencia relativa de los índices** (modulación)

$$
\begin{aligned}
\mathrm{RoPE}(x, m) &= x e^{mi\varepsilon} \\
\langle \mathrm{RoPE}(q_j, m), \mathrm{RoPE}(k_j, n) \rangle &= \langle q_j e^{mi\varepsilon}, k_j e^{ni\varepsilon} \rangle \\
&= q_j k_j e^{mi\varepsilon} \overline{e^{ni\varepsilon}} \\
&= q_j k_j e^{(m-n)i\varepsilon} \\
&= \mathrm{RoPE}(q_j k_j, m-n)
\end{aligned}
$$

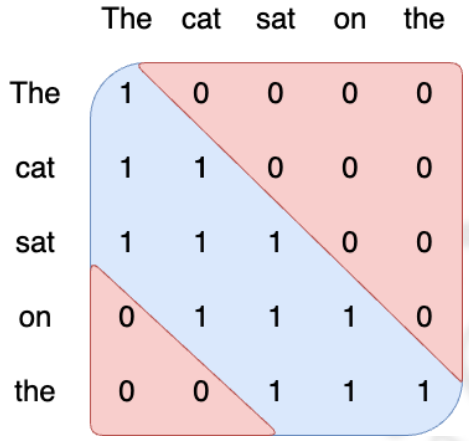


  - Es útil con implementaciones tipo flash-att:
    - Se puede aplicar a q, k antes de la atención

Transformer

- **Sliding window attention**

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150....



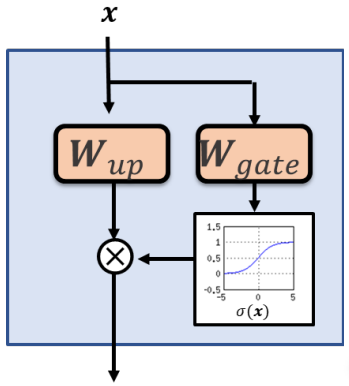
  - Se limita el contexto a un tamaño pequeño, menor que el tamaño de la secuencia
  - Está implementado en flash attention2 con ahorro de operaciones
  - Recuerda a la idea las convoluciones causales (i.e. wavenet)
    - La mascara de atención define un tamaño de relación máximo.

  Ejemplo los modelos recientes de mistral y otros como gemma2

  https://github.com/mistralai/mistral-src

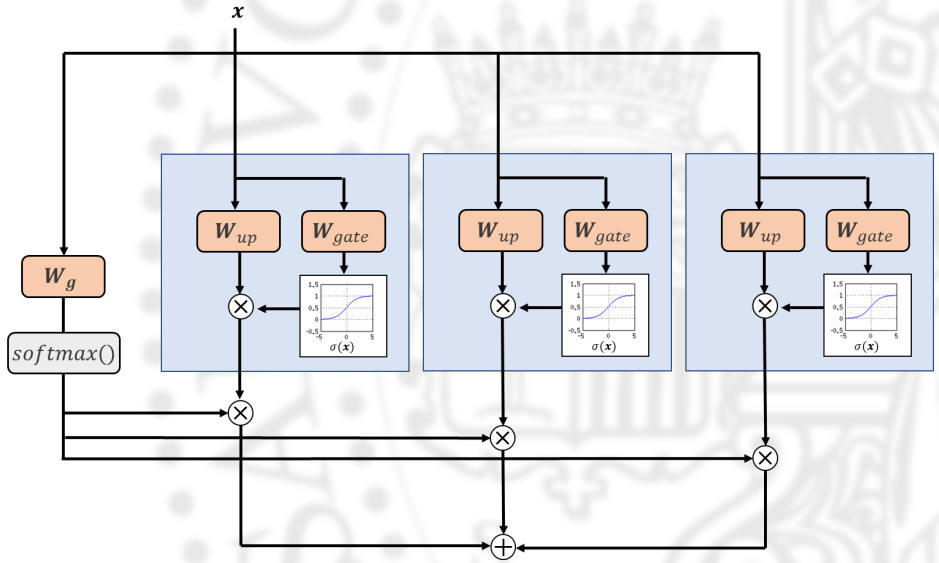
Transformer

- **Mixture of Experts**
  - En lugar de una FFN simple



	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B
MMLU (MCQ in 57 subjects)	69.9%	70.0%	**70.6%**
HellaSwag (10-shot)	87.1%	85.5%	86.7%
ARC Challenge (25-shot)	85.1%	85.2%	**85.8%**
WinoGrande (5-shot)	**83.2%**	81.6%	81.2%
MBPP (pass@1)	49.8%	52.2%	**60.7%**
GSM-8K (5-shot)	53.6%	57.1%	**58.4%**
MT Bench (for Instruct Models)	6.86	**8.32**	8.30

  - Se define una mezcla de FFNs
    - Una capa lineal con una softmax elige los “expertos”
      - Por afinidad de tema
    - Los expertos equivalen a una capa con una memoria mucho mayor pero ahorrando en cómputo
      - Si no se eligen no se computan

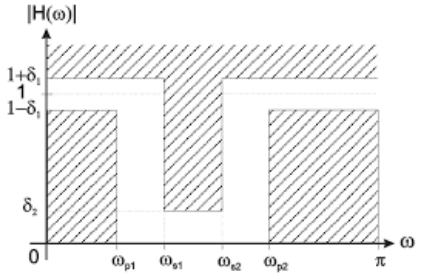


Transformer

- Limites computacionales:
  - Computación diferenciable
  - World models
  - SSMs
  - XLSTMS

Transformer

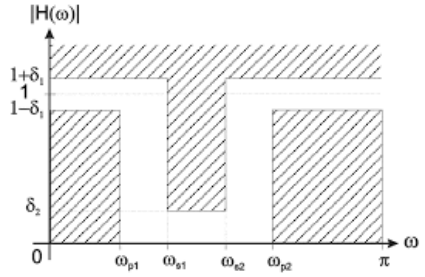
- **Transformers: ¿Máquinas de propósito general?**
  - Planteamiento clásico de un problema de ingeniería



    - se plantean unos requisitos/restricciones
    - se elige un método para la solución, a veces de forma analítica, otras aproximada, otras por optimización iterativa...
    - se comprueba si la solución cumple con las restricciones

Transformer

- **Transformers: ¿Máquinas de propósito general?**
  - Planteamiento clásico de un problema de ingeniería



    - se plantean unos requisitos/restricciones
    - se elige un método para la solución, a veces de forma analítica, otras aproximada, otras por optimización iterativa...
    - se comprueba si la solución cumple con las restricciones

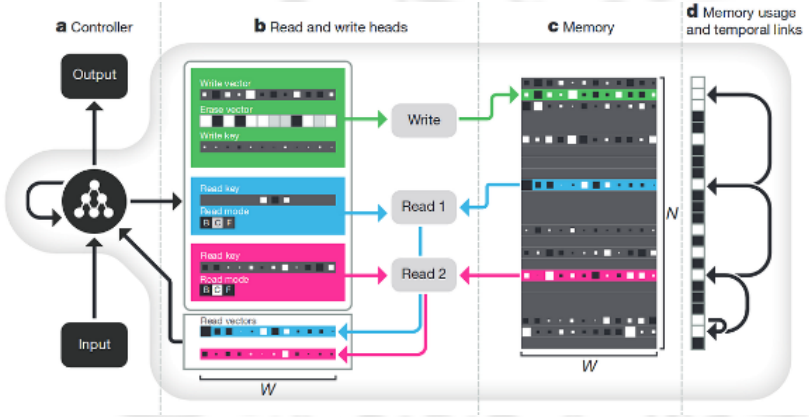
  - En aprendizaje automático presentamos datos y la solución esperada

```
entrada       | salida
------------------------------
0, 2, 1, 5    | 0, 1, 2, 5
4, 5, 3       | 3, 4, 5
0, 1          | 0, 1
8, 4, 5       | 4, 5, 8
...
```

    - - Por ejemplo dar varios números aleatorios y como objetico querer los números ordenados
    - - Preparamos muchos ejemplos distintos y entrenamos un sistema
    - - Medimos la calidad, no siempre acierta el 100% de los casos!

Transformer

- **Transformers: ¿Máquinas programables de propósito general?**
  - "differentiable computer"
    - 2014, Neural Turing Machines
    - 2016, Differentiable Neural Computer
      - Tenía memorías diferenciables:
        - » Escritura / Lectura
        - » **Heads / cabezales**
    - 2017, Transformers

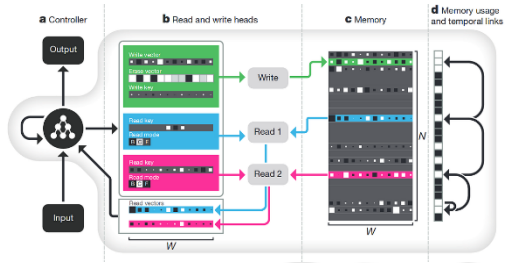


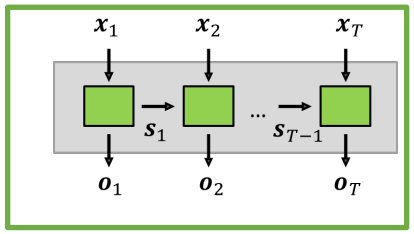
Hybrid computing using a neural network with dynamic external memory

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu & Demis Hassabis

Differentiable computer

2014, Neural Turing Machines
2016, Differentiable Neural Computer
2017, Transformers





Controller (LSTM)

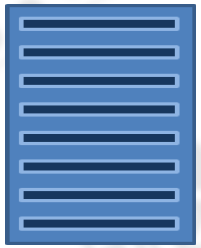
$\boldsymbol{k}_t$ $(key/address)$

$$\boldsymbol{r}_t = \sum_i w_t(i) \cdot \boldsymbol{M}_t[i,:] \qquad \sum_i w_t(i) = 1$$

Read head

$$\boldsymbol{M}_t[i,:] = \boldsymbol{M}_{t-1}[i,:](1 - w_t(i)\boldsymbol{e}_t) + w_t(i)\boldsymbol{a}_t$$

Write head:
erasing $\boldsymbol{e}_t$ adding $\boldsymbol{a}_t$



Memory $\boldsymbol{M}_t$ $(N \times D)$

- The memory indexing is done by measuring similarity: cosine distance
- similarity of the "index" called **key, $\boldsymbol{k}_t$,** to all the memorry positions
- Weigths are normalized to sum 1 using **softmax**

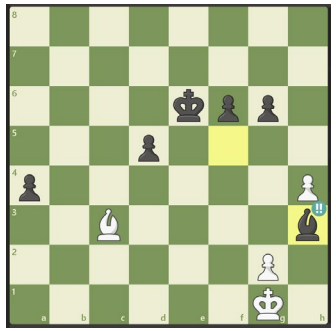
$$K[\mathbf{u}, \mathbf{v}] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}$$

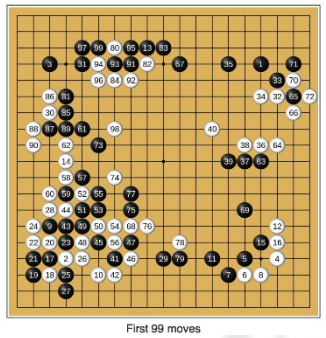
$$w_t(i) = \frac{exp(\beta_t K[\boldsymbol{k}_t, \boldsymbol{M}_t[i,:]])}{\sum_j exp(\beta_t K[\boldsymbol{k}_t, \boldsymbol{M}_t[j,:]])}$$

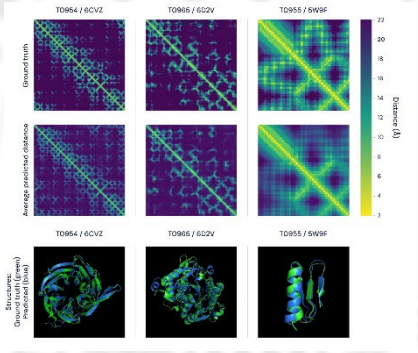
- This can be interpreted as a posterior probability: the higher most similar to key, and it Will be selected to read/write

Transformer

- **¿Qué utilidad tiene aprenda un algoritmo que ya existe? (y que no siempre lo haga bien)**
  - Estos experimentos se han hecho para demostrar esa capacidad de generar pequeños algoritmos si se necesitan
  - Estas capacidad destaca cuando no se conoce un algoritmo para resolver la tarea: traducir entre idiomas, resumir textos, evaluar una posición estratégica, predecir el plegado de proteínas...



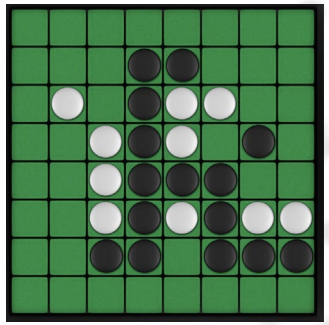




https://www.deepmind.com/blog/alphafold-using-ai-for-scientific-discovery-2020

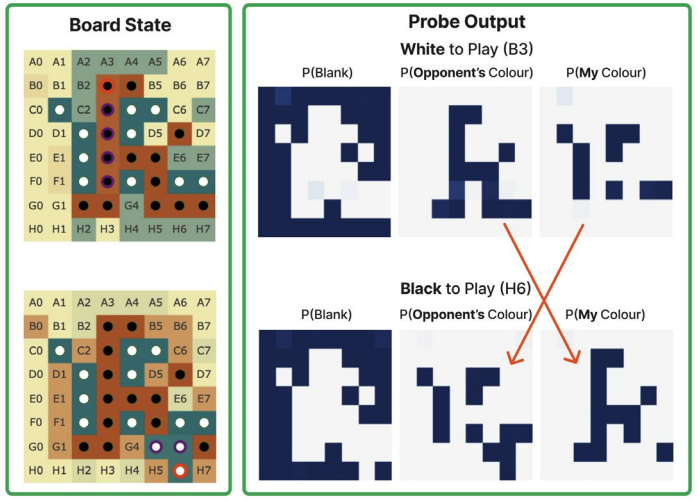
Transformer

- **Ejemplo generación de modelos del mundo**
  - El modelo se entrena para predecir en siguiente movimiento.
  - El modelo tiene una secuencia de movimientos
  - F4, F3, D2, F5, G2, F2, G3, C4, E5, F6, D6, E2, B4, C5, G7, C1, G6, F7, G5, C3, B3, H6
  - Si nos ayudamos de una representación visual entendemos mucho mejor de qué juego se trata y cúal es el estado actual del trablero



Transformer

- **Ejemplo generación de modelos del mundo**
  - En ese ejemplo se muestra cómo procesando los vectores de representación.
    - El modelo representa internamente el mapa del tablero diferenciando el color de las piezas y a quién le toca jugar



https://www.lesswrong.com/posts/nmxzr2zsjNtjaHh7x/actually-othello-gpt-has-a-linear-emergent-world

Transformer

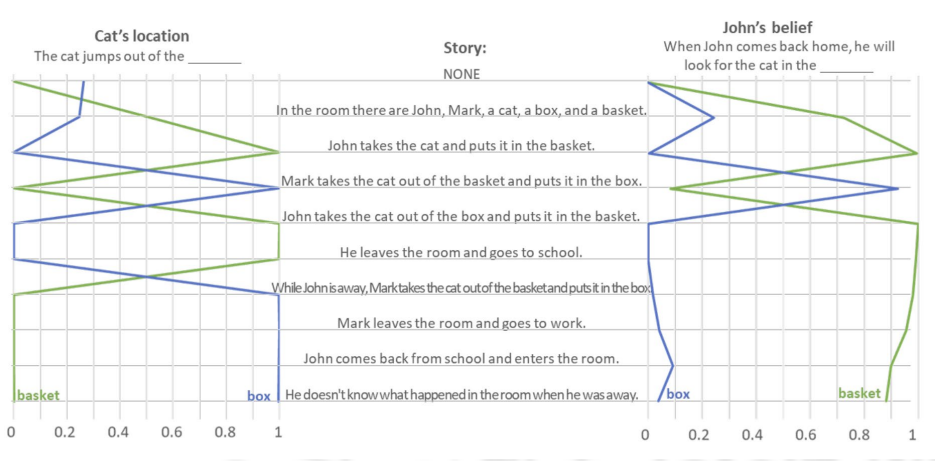
- **Transformers: Razonamiento / modelos del mundo**
  - Existen números trabajos recientes en los que se evaluan o entrenan LMs para resolver multitud de tareas

Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Task 7: Time Reasoning
In the afternoon Julie went to the park.
Yesterday Julie was at school.
Julie went to the cinema this evening.
Where did Julie go after the park? A:cinema
Where was Julie before the park? A:school

Task 8: Positional Reasoning
The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

Xiang, J et al (2023). Language Models Meet World Models: Embodied Experiences Enhance Language Models.

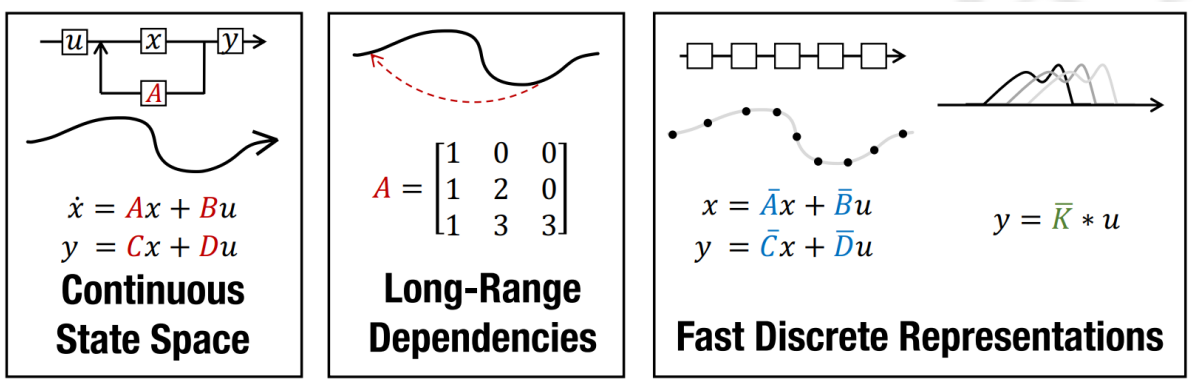


*Theory of Mind May Have Spontaneously Emerged in Large Language Models Authors: Michal Kosinski*1*

SSMs

- **Structured State Spaces**

Gu, A., Goel, K., & Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396...



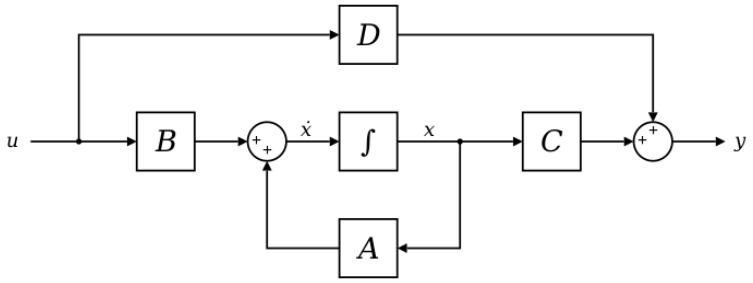
- SSM representa los estados como un sistema dinámico realimentado con una matriz de transición y matrices que controlan la entrada, la salida y la realimentación
- Ampliamente utilizados en Modelado de Sistemas, Sistemas de Control, Protocolos de Red, Diagnóstico de Fallos, Modelado de Procesos de Negocio, Sistemas Biológicos, Robótica, Procesos de Manufactura. …

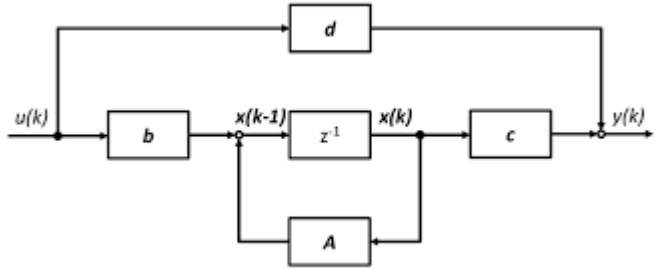
https://srush.github.io/annotated-s4/

SSMs

- **Structured State Spaces**

Gu, A., Goel, K., & Ré, C. (2021). Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396...





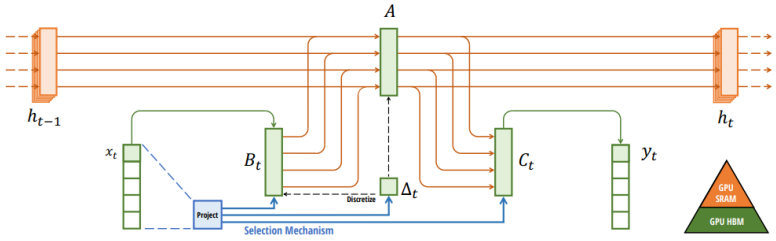
- Si las matrices A,B,C,D son fijas el sistemas invariante
- En ese caso se podría representar su comportamiento como in filtro de longitude infinita IIR
  - Se puede utilizer la convolución o la transformada de Fourier
- El paso de discretización puede ser de paso variable

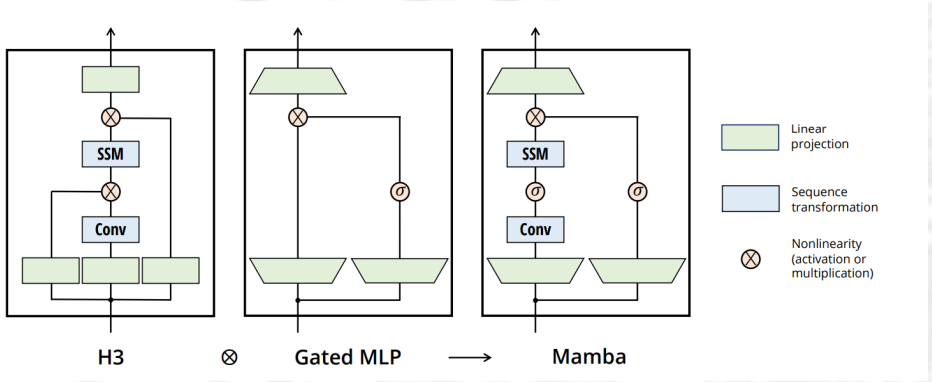
https://en.wikipedia.org/wiki/State-space_representation/

SSMs

- **MAMBA: Linear-time sequence modeling with selective state spaces**

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752.



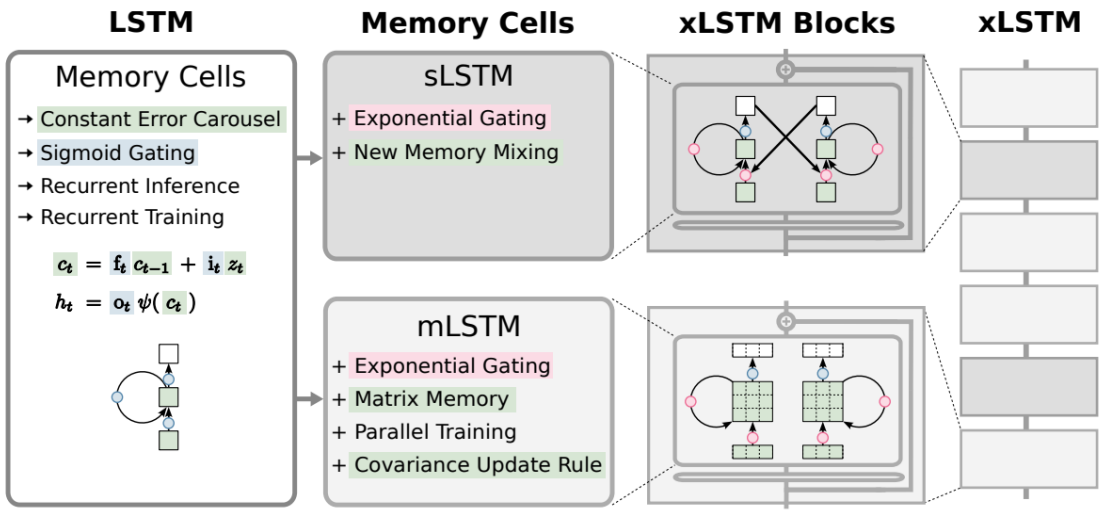


- El filtro en este modelo no es invariante, depende de la entrada, como las RNNs (LSTM, GRU)
- Los valores de las matrices B, C y el paso de discretización se calculan a partir de la entrada
- Se añade además un mecanismo de atención de tipo puerta para controlar la salida
- Se ha implementado mediante un kernel optimizado en cuda
- Responden muy bien a tareas de modelado de secuencias

xLSTM

- **xLSTM: Extended Long Short-Term Memory**

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J. and Hochreiter, S., 2024. xLSTM: Extended Long Short-Term Memory. arXiv preprint arXiv:2405.04517.



Model	#Params M	SlimPajama (15B) ppl ↓
GPT-3	356	14.26
Llama	407	14.25
H3	420	18.23
Mamba	423	13.70
Hyena	435	17.59
RWKV-4	430	15.62
RWKV-5	456	16.53
RWKV-6	442	17.40
RetNet	431	16.23
HGRN	411	21.83
GLA	412	19.56
HGRN2	411	16.77
xLSTM[1:0]	409	**13.43**
xLSTM[7:1]	408	13.48

https://github.com/NX-AI/xlstm.git

- Uno de los autores de las lstms originales renovando el modelo gracias a avances recientes
- Está dando buenos resultados en tareas de modelado de lenguaje, incluso en imagen
- Responden muy bien a tareas de modelado de secuencias y tareas lógicas

xLSTM

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J. and Hochreiter, S., 2024. xLSTM: Extended Long Short-Term Memory. arXiv preprint arXiv:2405.04517.

	Context Sentsitive		Deterministic Context Free		Regular					
	Bucket Sort	Missing Duplicate	Mod Arithmetic (w Brackets)	Solve Equation	Cycle Nav	Even Pairs	Mod Arithmetic (w/o Brackets)	Parity	Majority	Majority Count
Llama	0.92 ± 0.02	0.08 ± 0.0	0.02 ± 0.0	0.02 ± 0.0	0.04 ± 0.01	1.0 ± 0.0	0.03 ± 0.0	0.03 ± 0.01	0.37 ± 0.01	0.13 ± 0.0
Mamba	0.69 ± 0.0	0.15 ± 0.0	0.04 ± 0.01	0.05 ± 0.02	0.86 ± 0.04	1.0 ± 0.0	0.05 ± 0.02	0.13 ± 0.02	0.69 ± 0.01	0.45 ± 0.03
Retention	0.13 ± 0.01	0.03 ± 0.0	0.03 ± 0.0	0.03 ± 0.0	0.05 ± 0.01	0.51 ± 0.07	0.04 ± 0.0	0.05 ± 0.01	0.36 ± 0.0	0.12 ± 0.01
Hyena	0.3 ± 0.02	0.06 ± 0.02	0.05 ± 0.0	0.02 ± 0.0	0.06 ± 0.01	0.93 ± 0.07	0.04 ± 0.0	0.04 ± 0.0	0.36 ± 0.01	0.18 ± 0.02
RWKV-4	0.54 ± 0.0	0.21 ± 0.01	0.06 ± 0.0	0.07 ± 0.0	0.13 ± 0.0	1.0 ± 0.0	0.07 ± 0.0	0.06 ± 0.0	0.63 ± 0.0	0.13 ± 0.0
RWKV-5	0.49 ± 0.04	0.15 ± 0.01	0.08 ± 0.0	0.08 ± 0.0	0.26 ± 0.05	1.0 ± 0.0	0.15 ± 0.02	0.06 ± 0.03	0.73 ± 0.01	0.34 ± 0.03
RWKV-6	0.96 ± 0.0	0.23 ± 0.06	0.09 ± 0.01	0.09 ± 0.02	0.31 ± 0.14	1.0 ± 0.0	0.16 ± 0.0	0.22 ± 0.12	0.76 ± 0.01	0.24 ± 0.01
LSTM (Block)	0.99 ± 0.0	0.15 ± 0.0	0.76 ± 0.0	0.5 ± 0.05	0.97 ± 0.03	1.0 ± 0.0	0.91 ± 0.09	1.0 ± 0.0	0.58 ± 0.02	0.27 ± 0.0
LSTM	0.94 ± 0.01	0.2 ± 0.0	0.72 ± 0.04	0.38 ± 0.05	0.93 ± 0.07	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.82 ± 0.02	0.33 ± 0.0
xLSTM[0:1]	0.84 ± 0.08	0.23 ± 0.01	0.57 ± 0.09	0.55 ± 0.09	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	0.75 ± 0.02	0.22 ± 0.0
xLSTM[1:0]	0.97 ± 0.0	0.33 ± 0.22	0.03 ± 0.0	0.03 ± 0.01	0.86 ± 0.01	1.0 ± 0.0	0.04 ± 0.0	0.04 ± 0.01	0.74 ± 0.01	0.46 ± 0.0
xLSTM[1:1]	0.7 ± 0.21	0.2 ± 0.01	0.15 ± 0.06	0.24 ± 0.04	0.8 ± 0.03	1.0 ± 0.0	0.6 ± 0.4	1.0 ± 0.0	0.64 ± 0.04	0.5 ± 0.0

```
Bucket Sort
  Sequence: 1 4 8 6 1 1 1 4 6 8
Cycle Nav
  Sequence: STAY +1 -1 +1 STAY +1 +1 +1 -1 P3
Even Pairs
  Sequence: a b b a a b a b a a
Majority
  Sequence: 1 7 6 4 3 8 1 7 2 1
Majority Count
  Sequence: 1 7 6 4 4 8 1 7 2 2
Missing Duplicate
  Sequence: 4 8 6 2 5 4 8 6 2 [MIS] 5
Mod Arithmetic (w/o Braces)
  Sequence: 0 - 4 + 0 - 2 = 4 [PAD]
Mod Arithmetic (w Braces)
  Sequence: ( ( ( 2 ) * - 2 ) - ( - 4 - 2 ) ) = 2
Odds First
  Sequence: 2 7 3 2 6 9 [ACT] 2 3 6 7 2 9
Parity:
  Sequence: a b b a a b a b
Repetition
  Sequence: 2 4 8 6 2 [ACT] 2 4 8 6 2
Reverse String
  Sequence: 2 4 8 6 2 [ACT] 2 6 8 4 2
Stack Manipulation
  Sequence: ST1 ST1 ST3 POP POP PS3 PS3 [ACT] ST1 ST3 ST3
Set
  Sequence: 8 6 6 3 5 4 5 3 [ACT] 8 6 3 5 4
Solve Equation:
  Sequence: ( ( ( 2 + 0 ) + - x ) - ( 1 ) ) = 2 [ACT] 2
```

xLSTM

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J. and Hochreiter, S., 2024. xLSTM: Extended Long Short-Term Memory. arXiv preprint arXiv:2405.04517.

- **Bucket Sort** Given a string of tokens of a sorted alphabet, compute the sorted string.
  $|V| = 11 \quad s_{\text{rand}} = \frac{1}{|V|-1}$
- **Cycle Nav** Given a string of "movement tokens" $(+1, -1, \text{STAY})$ compute the end position of the agent with start position 0. The position must be computed modulo the maximum position.
  $|V| = 9 \quad s_{\text{rand}} = \frac{1}{|V|-4}$
- **Even Pairs** Given a binary string of a and b tokens, compute whether the number of ab and ba is even. This task can be solved by checking if the first and last token of the string are equal.
  $|V| = 3 \quad s_{\text{rand}} = 0.5$
- **Majority** Given a string of tokens, compute the token that occurred most often in the sequence.
  $|V| = 64 \quad s_{\text{rand}} = \frac{1}{|V|-1}$
- **Majority Count** Given a string of tokens of an ordered alphabet. Compute the count of the token that occurred most often in the sequence. If the count exceeds the vocab size, the highest vocab token should be outputted.
  $|V| = 64 \quad s_{\text{rand}} = \frac{1}{|V|-1}$
- **Missing Duplicate** Given a string of tokens. The string is repeated but one of the tokens is masked in the repetition. Output the token that is masked.
  $|V| = 11 \quad s_{\text{rand}} = \frac{1}{|V|-2}$
- **Mod Arithmetic (w/o Brackets)** Calculate the result — modulo the max number — of the arithmetic operations in the context. The maximum number is the vocabulary size minus the number of special tokens (+,-,*,=, [PAD]).
  $|V| = 10 \quad s_{\text{rand}} = \frac{1}{|V|-5}$
- **Mod Arithmetic (w Brackets)** Calculate the result — modulo the maximum number — of the arithmetic operations in the context. The maximum number is vocabulary size minus the number of special tokens (+,-,*,=,(,), [PAD]).
  $|V| = 12 \quad s_{\text{rand}} = \frac{1}{|V|-7}$
- **Odds First** An string of tokens $t_1, t_2, t_3, ...t_n$ is given. Output all tokens with and odd index $(t_1, t_3, ...)$ then the token with an even index $(t_2, t_4,..)$. Apart from that keep the ordering of the initial string.
  $|V| = 12 \quad s_{\text{rand}} = \frac{1}{|V|-2}$
- **Parity** Given a binary string of a and b tokens, compute if the number of b's is even. If the number is even output a otherwise b. This is equivalent to sequentially calculating the half-adder sum.
  $|V| = 3 \quad s_{\text{rand}} = 0.5$
- **Repetition** Given a string of tokens — repeat it.
  $|V| = 12 \quad s_{\text{rand}} = \frac{1}{|V|-2}$
- **Reverse String** Given a string of tokens — repeat it in reverse order.
  $|V| = 12 \quad s_{\text{rand}} = \frac{1}{|V|-2}$
- **Stack Manipulation** An initial stack content is given, followed by a sequence of push and pop operations. Compute the stack content after the operations
  $|V| = 11 \quad s_{\text{rand}} = \frac{1}{\lfloor \frac{|V|-3}{2} \rfloor}$
- **Set** Given a string of tokens, compute the ordered set of the tokens. Keep the ordering so that tokens that occurred first are also outputted first.
  $|V| = 128 \quad s_{\text{rand}} = \frac{1}{|V|-2}$
- **Solve Equation** Given is an equation with the operators {+,-,*,=,(,)}, number, and an unknown variable x. Compute the value of the variable modulo the max number. The maximum number is vocabulary size minus the number of special tokens (+,-,*,=,(,), [PAD], [ACT]).
  $|V| = 14 \quad s_{\text{rand}} = \frac{1}{|V|-9}$