Clase 7

Procesamiento del Lenguaje Natural, Transformers y LLMs

Enzo Ferrante

eferrante@sinc.unl.edu.ar

@enzoferrante

Procesamiento del Lenguaje Natural

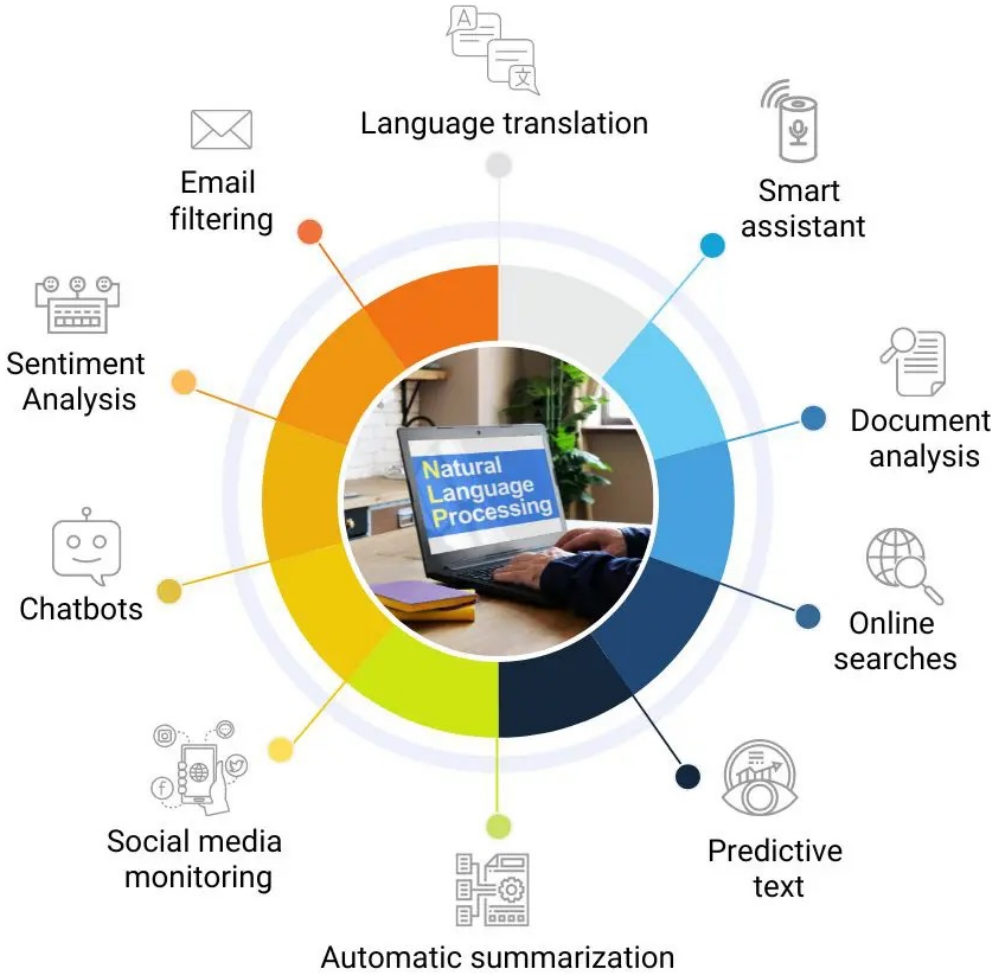
Ejemplo de problemas abordados

Clasificación de Texto

Asignar una categoría predefinida a un texto dado.

Ejemplo: Clasificar correos electrónicos como 'spam' o 'no spam'.

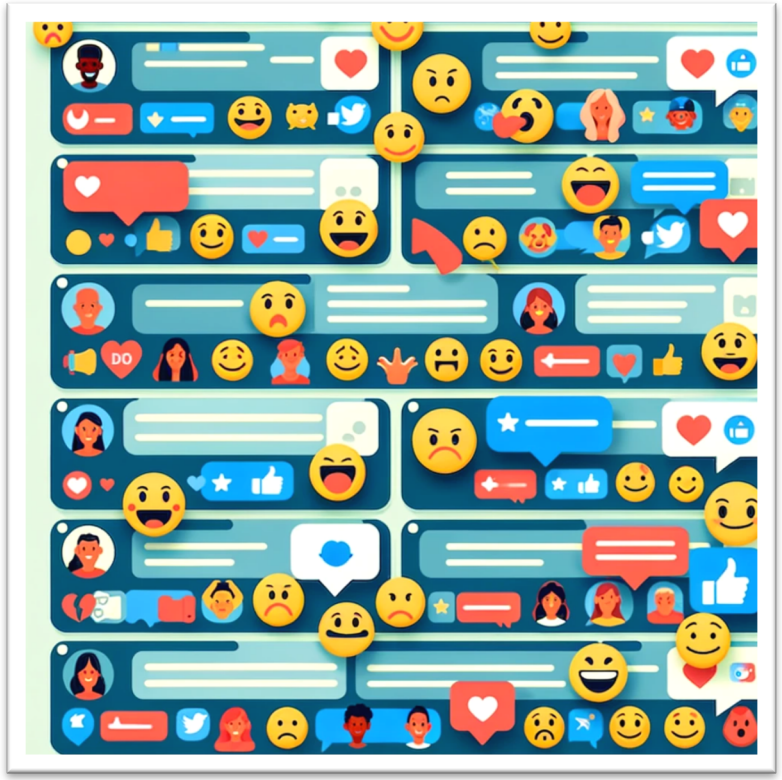
Procesamiento del Lenguaje Natural



- El Procesamiento del Lenguaje Natural se ocupa de las interacciones entre las computadoras y los idiomas humanos (naturales).
- Implica desarrollar algoritmos y modelos que puedan procesar, analizar y generar lenguaje humano.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados



Análisis de Sentimientos

Determinar la actitud o el tono emocional de un texto.

Ejemplo: Analizar opiniones en redes sociales para saber
si los comentarios sobre un producto son positivos, negativos o neutrales.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados

Reconocimiento de Entidades Nombradas (NER)

Identificar y clasificar entidades mencionadas en un texto en categorías como nombres de personas, organizaciones, lugares, etc.

Ejemplo: Extraer nombres de empresas y ubicaciones de un artículo de noticias.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados

Traducción Automática

Convertir texto de un idioma a otro.

Ejemplo: Traducir un documento del inglés al español.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados

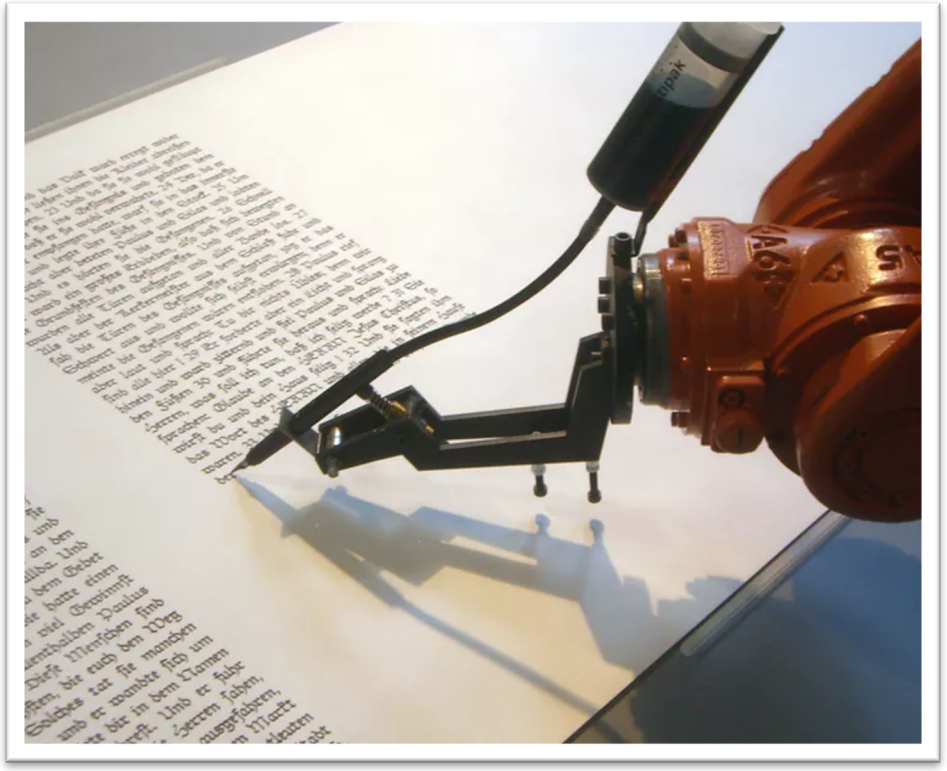
Resumen de Texto

Generar un resumen breve de un texto largo.

Ejemplo: Crear un resumen de un artículo de investigación

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados



Generación de Texto

Generar texto coherente y relevante a partir de una entrada dada.

Ejemplo: Completar automáticamente una frase o párrafo en un procesador de textos.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados

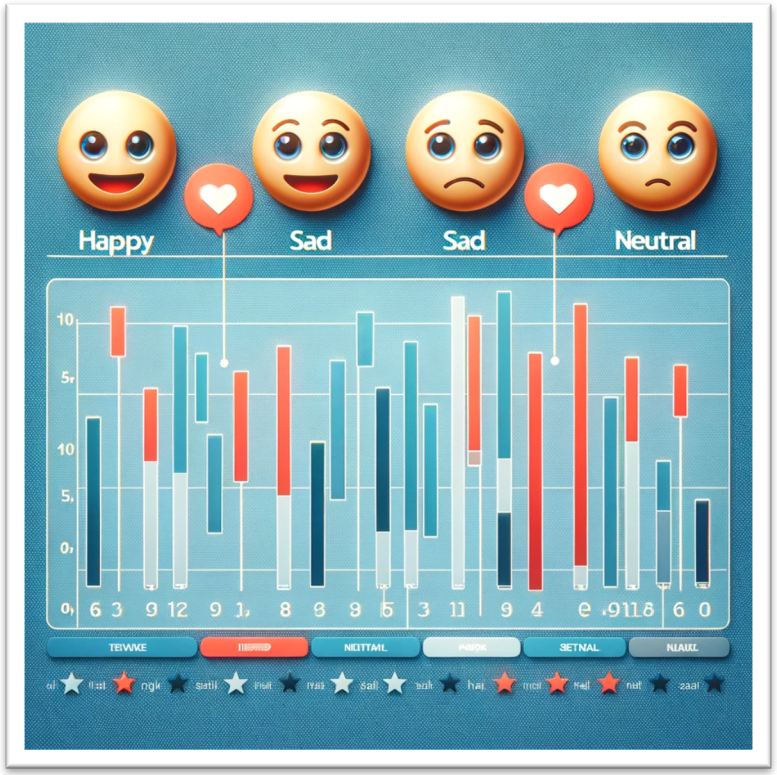
Respuesta a preguntas

Descripción: Responder preguntas específicas

Ejemplo: Responder preguntas de un usuario utilizando información contenida en una base de datos de conocimientos.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados



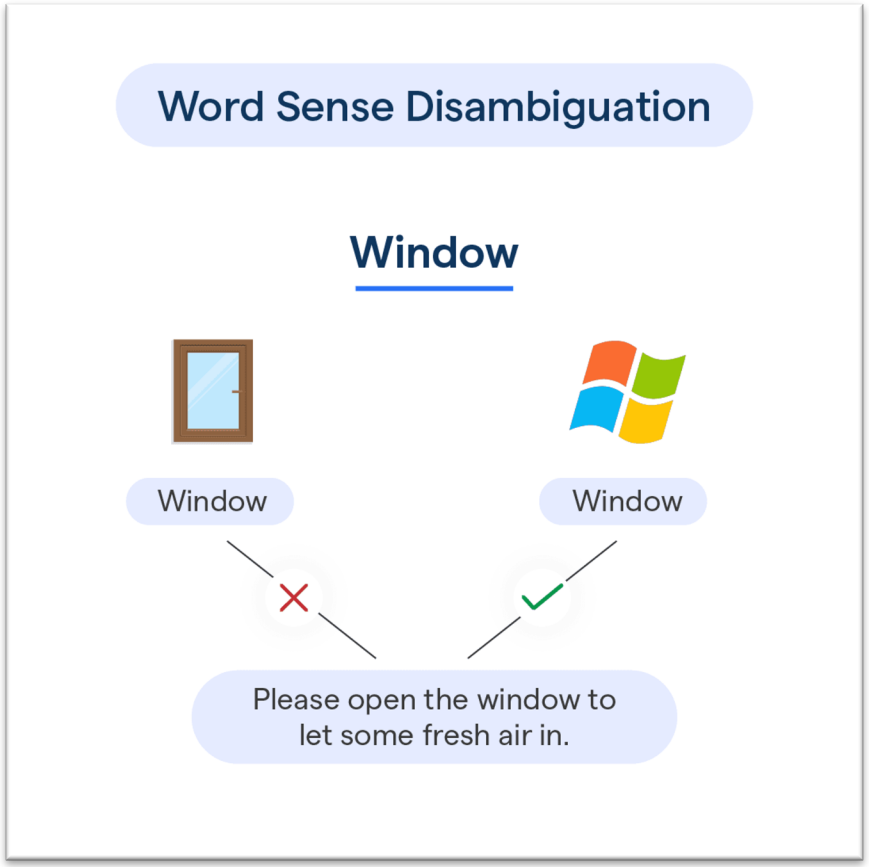
Análisis de sentimientos

Evaluar la polaridad de las emociones en un texto.

Ejemplo: Determinar si una reseña de producto es positiva, negativa o neutral.

Procesamiento del Lenguaje Natural

Ejemplo de problemas abordados

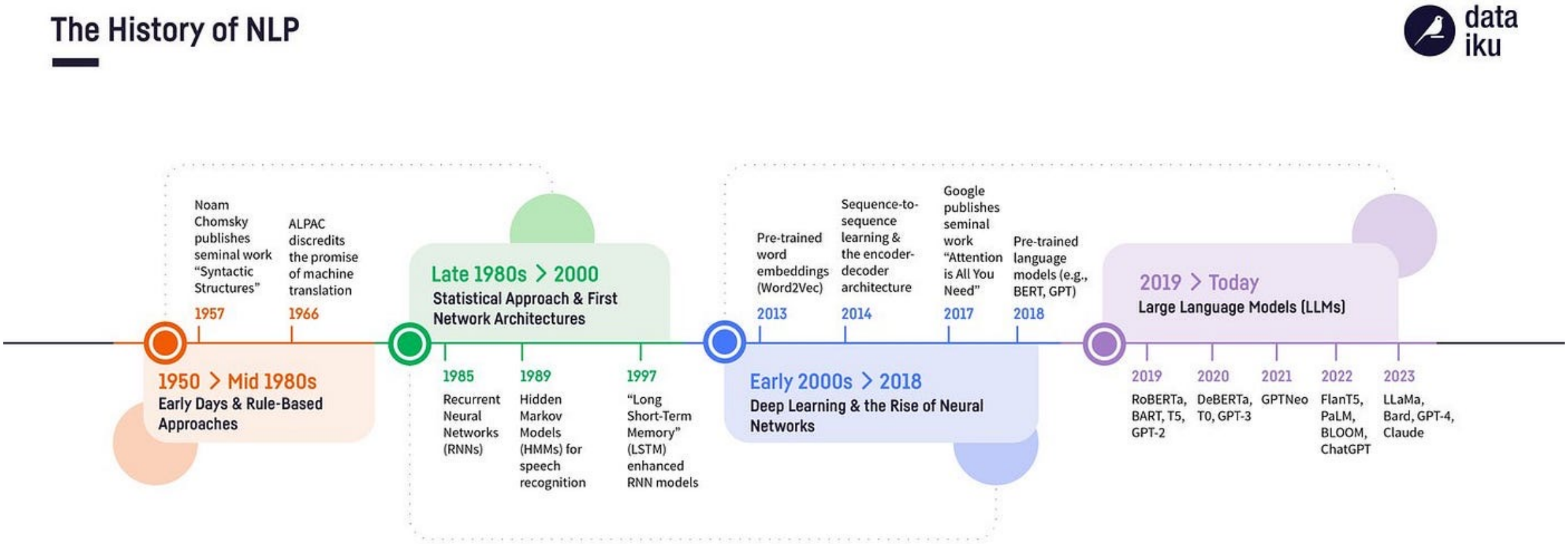


Desambiguación Semántica

Determinar el significado correcto de una palabra que tiene múltiples significados según el contexto.

Ejemplo: Determinar si la palabra 'banco' se refiere a una institución financiera o a un asiento en un parque.

Breve historia de los modelos de lenguaje



Cómo transformar texto en representaciones numéricas?

Frog on a log.

Tokenization

[CLS]	Frog	on	a	log	.	[SEP]
101	2025	2001	1063	8532	2011	102

Tokenization

Tokenization

Corta el texto en unidades que pertenecen a un *vocabulario* de posibles ***tokens***

Imagen extraída de: https://nlpiation.medium.com/how-to-use-huggingfaces-transformers-pre-trained-tokenizers-e029e8d6d1fa

Tokenization

Character Level:

- Permite modelar cualquier palabra, pero require que el modelo aprenda las relaciones entre caracteres

Word level:

-Si las palabras no existen en el corpus de entrenamiento, no las puede modelar

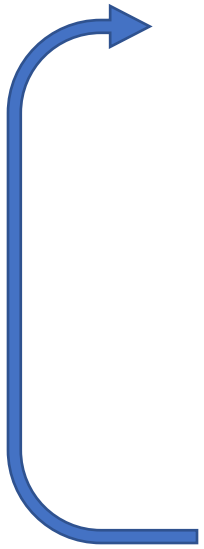
- Palabras que terminan en diferente sufijo son modeladas independientemente (e.g. walk, walked)

Subword level:

- Buen compromiso entre modelar a nivel caracter y a nivel palabra
- El vocabulario final incluye tanto palabras como fragmentos
- Requiere de un proceso de entrenamiento (token learner) y de segmentación (token segmenter)
- Más usado en la práctica para LLMs
- Algoritmos conocidos:
    - **Byte-Pair Encoding (Senrich et al, 2016)**
    - Unigram Language Modeling (Kudo, 2018)

Byte-Pair Tokenizer

(Senrich et al, ACL 2016)



1. Empieza con un **vocabulario** compuesto por **todos los caracteres** individuales

2. Examina el corpus de entrenamiento **eligiendo los dos símbolos más frecuentes**, y los **combina** en uno nuevo.

3. **Reemplaza** todas las ocurrencias de ambos caracteres por el nuevo símbolo

Itera *k* veces, agregando k nuevos símbolos a los caracteres originales

Byte-Pair Tokenizer

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and **Barry Haddow** and **Alexandra Birch**
School of Informatics, University of Edinburgh
{rico.sennrich,a.birch}@ed.ac.uk, bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by

lem, and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasser|behandlungs|anlange* 'sewage water treatment plant', for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et

Byte-Pair Tokenizer

Ejemplo: corpus de entrenamiento con 18 palabras

corpus

```
5   l o w _
2   l o w e s t _
6   n e w e r _
3   w i d e r _
2   n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w
```

Frecuencia = 9

corpus

```
5   l o w _
2   l o w e s t _
6   n e w er _
3   w i d er _
2   n e w _
```

vocabulary

```
_, d, e, i, l, n, o, r, s, t, w, er
```

Byte-Pair Tokenizer

Ejemplo: corpus de entrenamiento con 18 palabras

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

corpus

5 l o w _
2 l o w e s t _
6 ne w er_
3 w i d er_
2 ne w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

Byte-Pair Tokenizer

Ejemplo: corpus de entrenamiento con 18 palabras

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

corpus

5 l o w _
2 l o w e s t _
6 ne w er_
3 w i d er_
2 ne w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

Byte-Pair Tokenizer

Ejemplo: corpus de entrenamiento con 18 palabras

merge	current vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

Luego, el token segmenter se aplica en el corpus de prueba iterativamente, fusionando en el órden de aparición

1. Se segmentan las frases en caracteres indivuales

2. Reemplazo cada instancia de (e,r) por (er)

3. Reemplazo cada instancia de (er,_) por (er_)

….

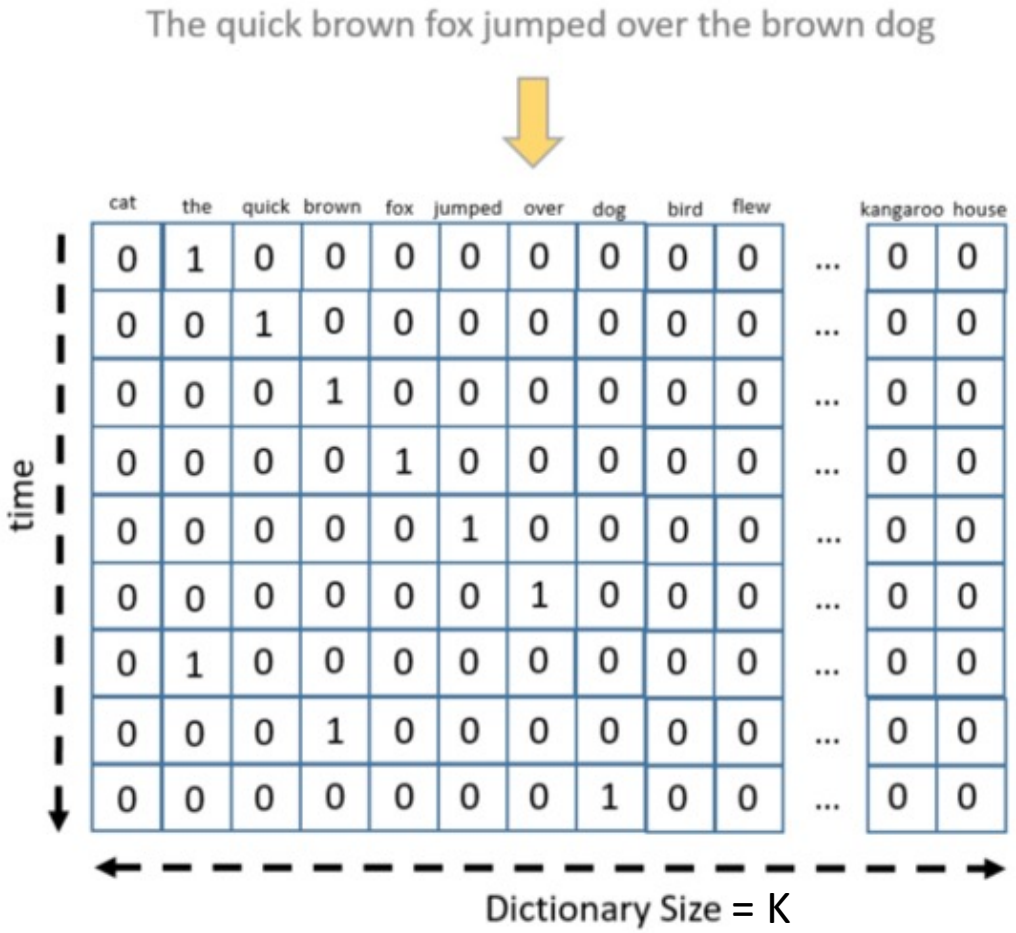
Al final, **“newer_” (palabra muy frecuente) será un solo token** mientras que **“low” “er_” será dos tokens (baja frecuencia)**

Word embeddings

Word embeddings

Cómo podemos representar las palabras para procesarlas con modelos neuronales?

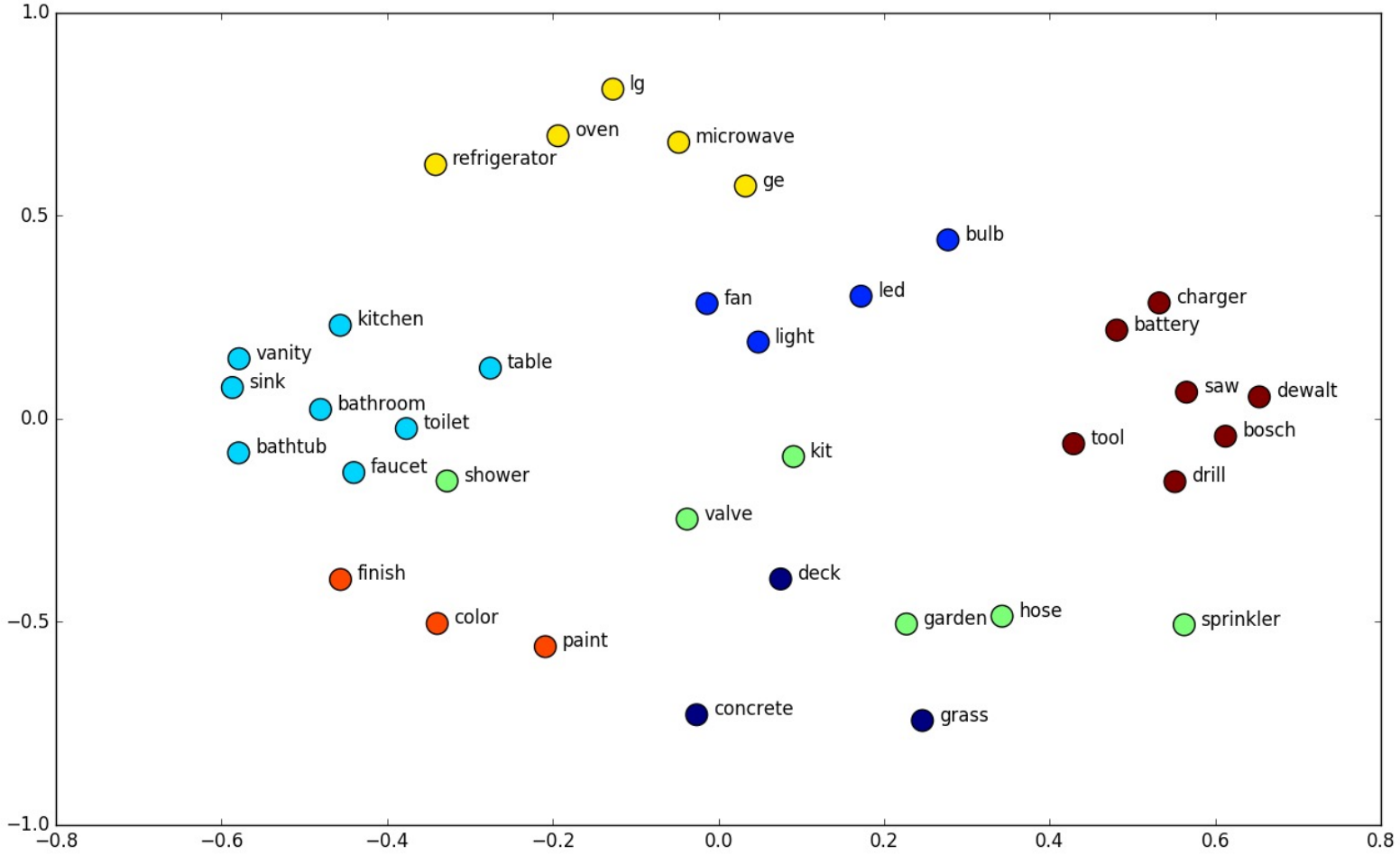
Notación one-hot!



Qué sentido tiene la distancia entre palabras representadas con notación one-hot?

Solución: Word Embeddings

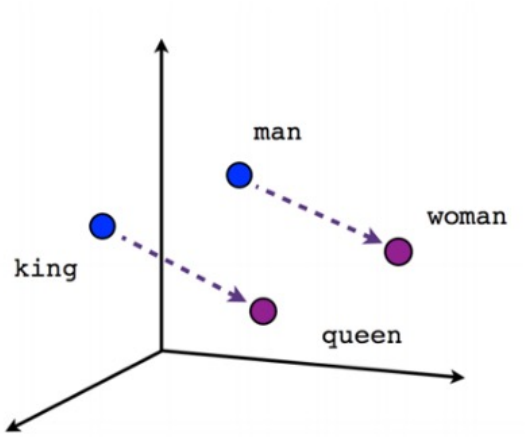
Word embeddings



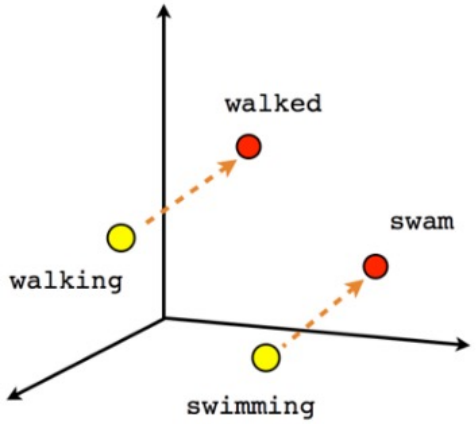
- Suelen ser vectores cortos (dimensión usualmente varia entre 50 y 1000)
- Son densos (distinto de la notación one-hot que es esparsa)

Word embeddings

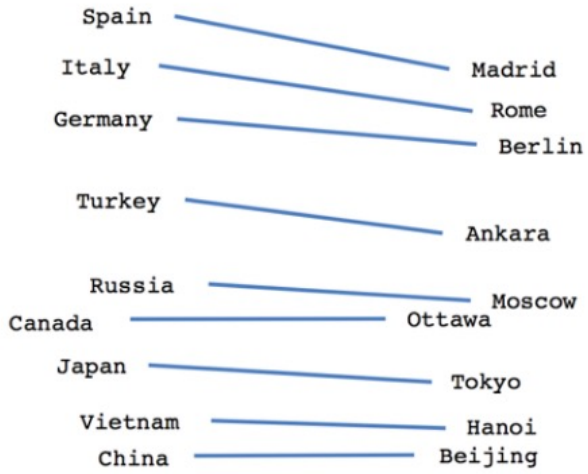
Midiendo similitud semántica



Male-Female



Verb tense



Country-Capital

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Se mueve entre -1 (apuntan en dirección opuesta) y 1 (misma dirección). 0 indica ortogonalidad.

Word embeddings

Matriz de embeddings: al multiplicar por el vector one-hot $\mathbf{x}_n$, recupera el embedding corresponidente $\mathbf{v}_n$

D x 1 (Embedding dim) $\mathbf{v}_n = \mathbf{E}\mathbf{x}_n$ K x 1 (Dictionary Size)

D x K

E es aprendida a partir de un corpus de datos

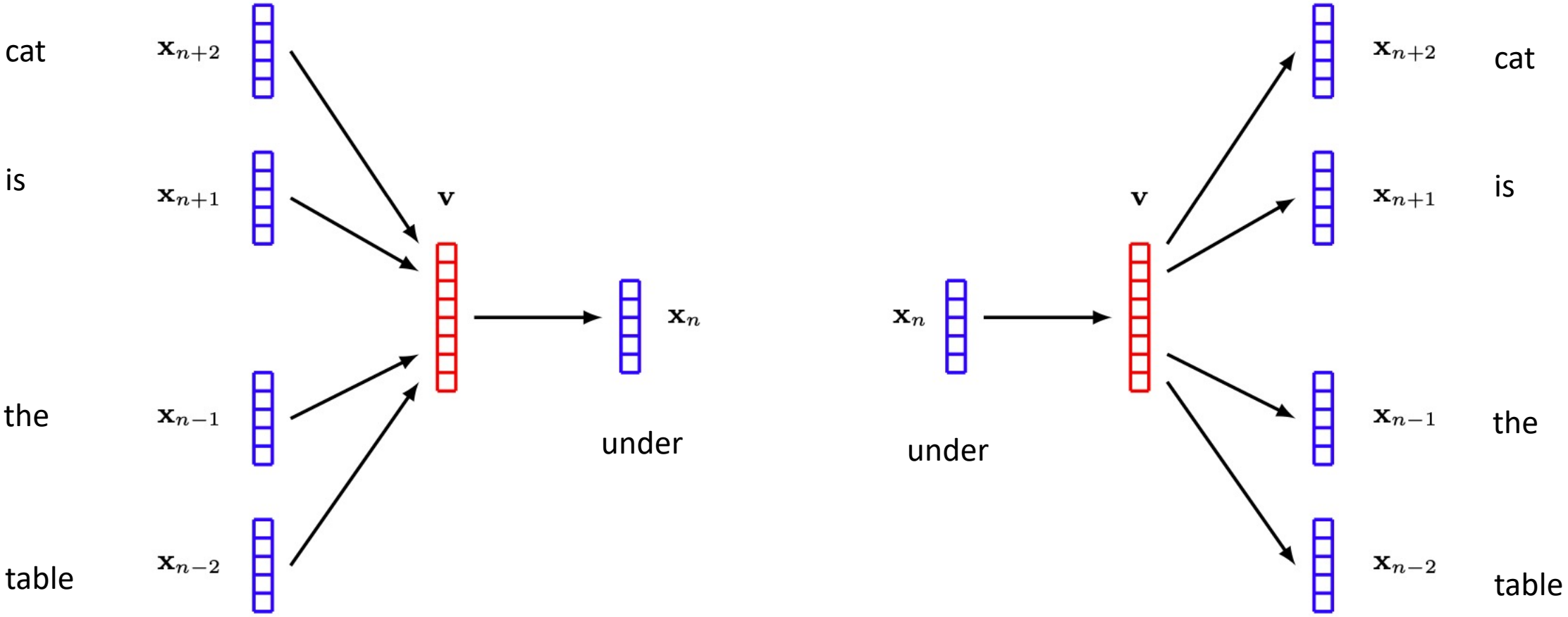
Word2Vec

Técnica para aprender embeddings

- Puede ser visto como una **red neuronal de dos capas**
- Aprende **embeddings estáticos** (no cambian con el contexto de las palabras), distinto a los embeddings contextuales que veremos luego, donde el embedding cambia en función del contexto.
- La idea es que palabras que aparezcan **rodeadas del mismo contexto, tendrán embeddings similares**

Word2Vec

"The cat is under the table"



Continuous Bag of Words (CBOW)

Skip-gram

Vector composition

King – Man + Woman = Queen

Transformers

Transformers

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

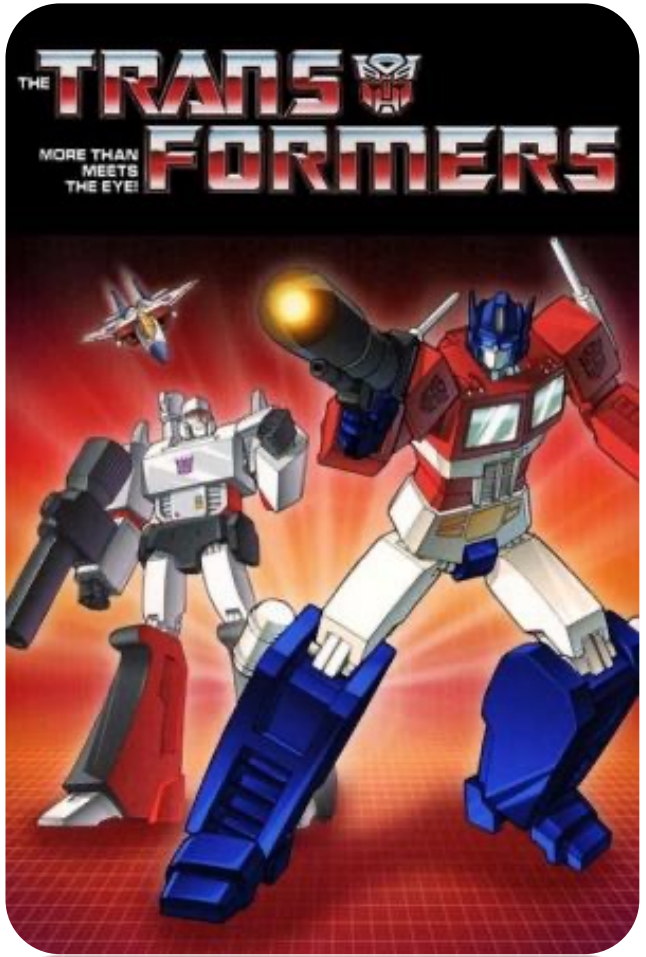
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to



Transformers

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to

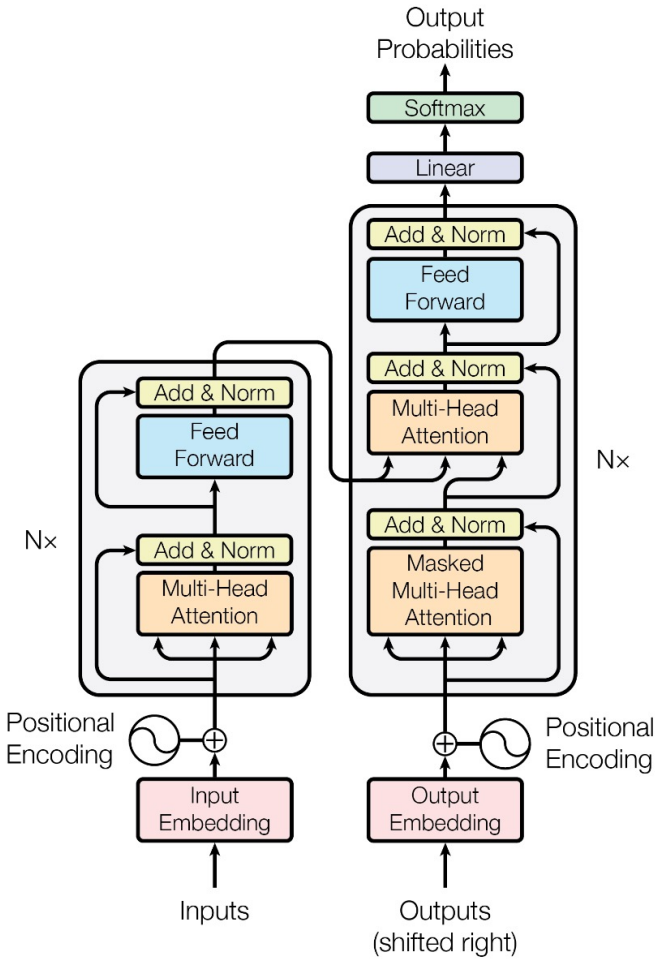


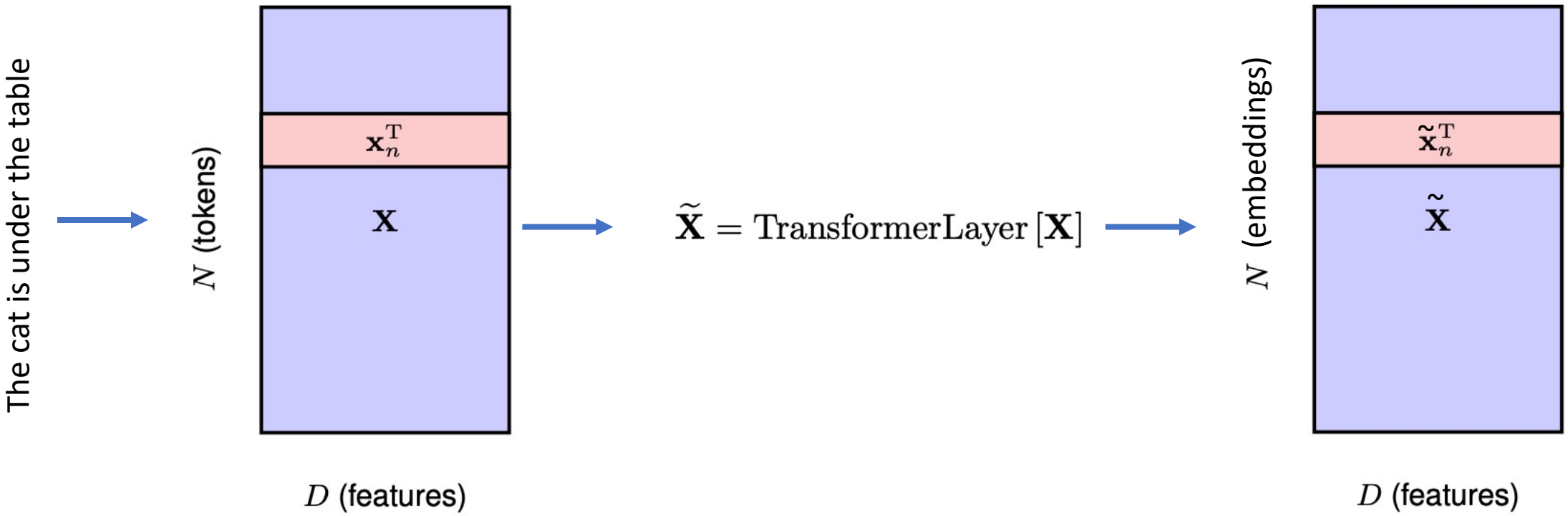
Figure 1: The Transformer - model architecture.

Procesando texto con transformers

The restaurant refused to serve me a ham sandwich because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambiance was just as good as the food and service.

Objetivo: transformar este texto en una nueva **representación** útil para **tareas downstream**

Una capa Transformer



Podemos luego apilar multiples capas para obtener representaciones más ricas donde los tokens incorporen información por medio de ‘prestar atención’ a otros tokens

Atención

Atención

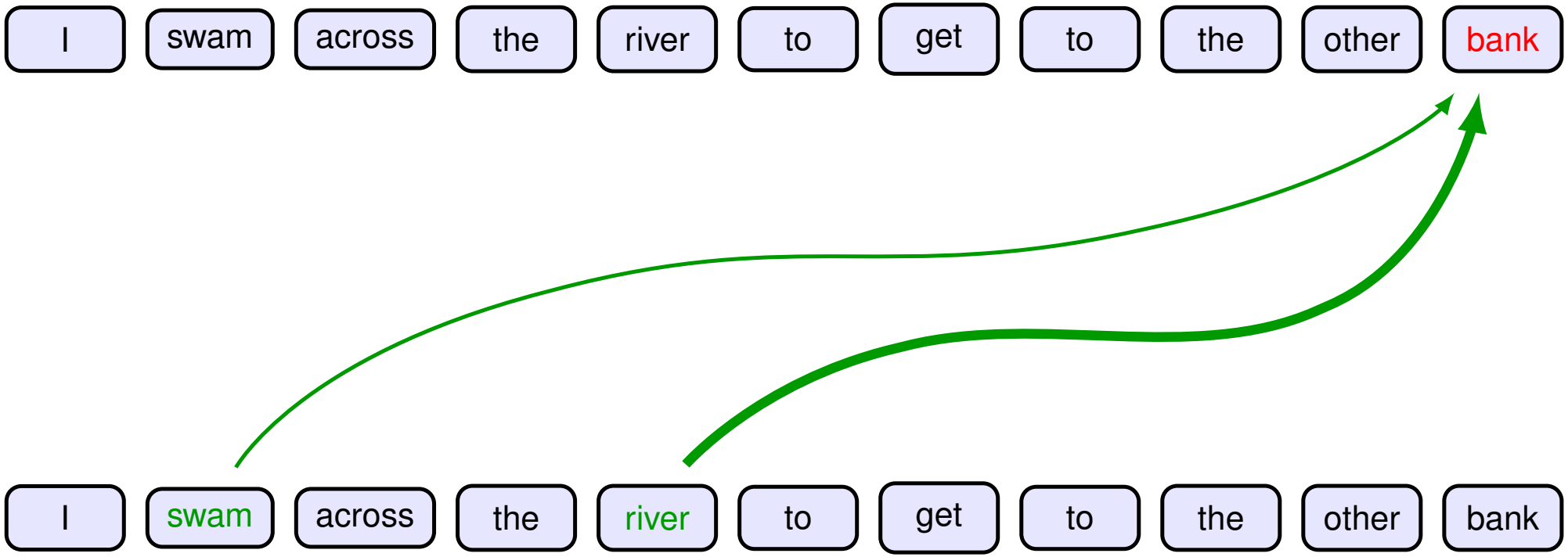
The restaurant refused to serve me a ham sandwich because it only cooks vegetarian food. In the end, they just gave me two slices of bread. Their ambiance was just as good as the food and service.

1. **La entrada puede ser muy larga**, por lo que la representación puede también serlo
   Ej: embeddings de 1024 x 37 palabras = 37888 features → MLP?

2. **Frases de diferente longitud**
   Sería ideal que se compartan los parámetros (similar a una CNN)

3. **Ambiguedad del lenguaje**
   La palabra 'it' necesita contexto para saber qué representa → Atención

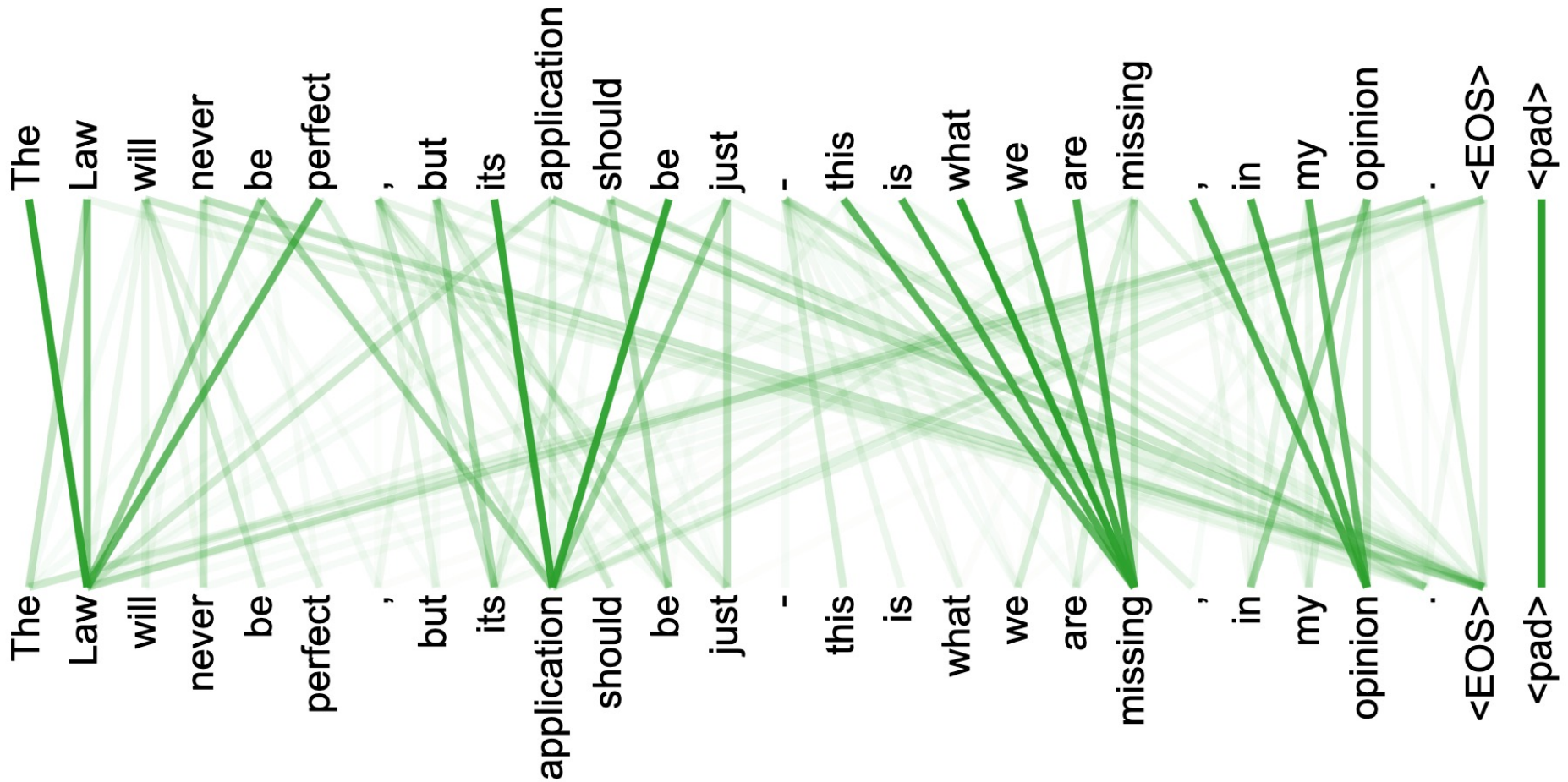
Atención



I swam across the river to get to the other bank.

I walked across the road to get cash from the bank.

Atención


The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>

Atención

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \longrightarrow \mathrm{TransformerLayer}\,[\mathbf{X}] \longrightarrow \mathbf{y}_1, \cdots, \mathbf{y}_N$$

Los y_i deben contener información de los x_i, y prestar más 'atención' a unos tokens que otros

$$\mathbf{y}_n = \sum_{m=1}^{N} a_{nm}\mathbf{x}_m$$

Coeficiente de atención del token de salida **n** al de entrada **m**

$$a_{nm} \geqslant 0$$

$$\sum_{m=1}^{N} a_{nm} = 1$$

Atención

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \longrightarrow \text{TransformerLayer}\,[\mathbf{X}] \longrightarrow \mathbf{y}_1, \cdots, \mathbf{y}_N$$

Los $\boldsymbol{y_i}$ deben contener información de los $\boldsymbol{x_i}$, y prestar más 'atención' a unos tokens que otros

$$\mathbf{y}_n = \sum_{m=1}^{N} a_{nm}\mathbf{x}_m$$

Coeficiente de atención del token de salida **n** al de entrada **m**

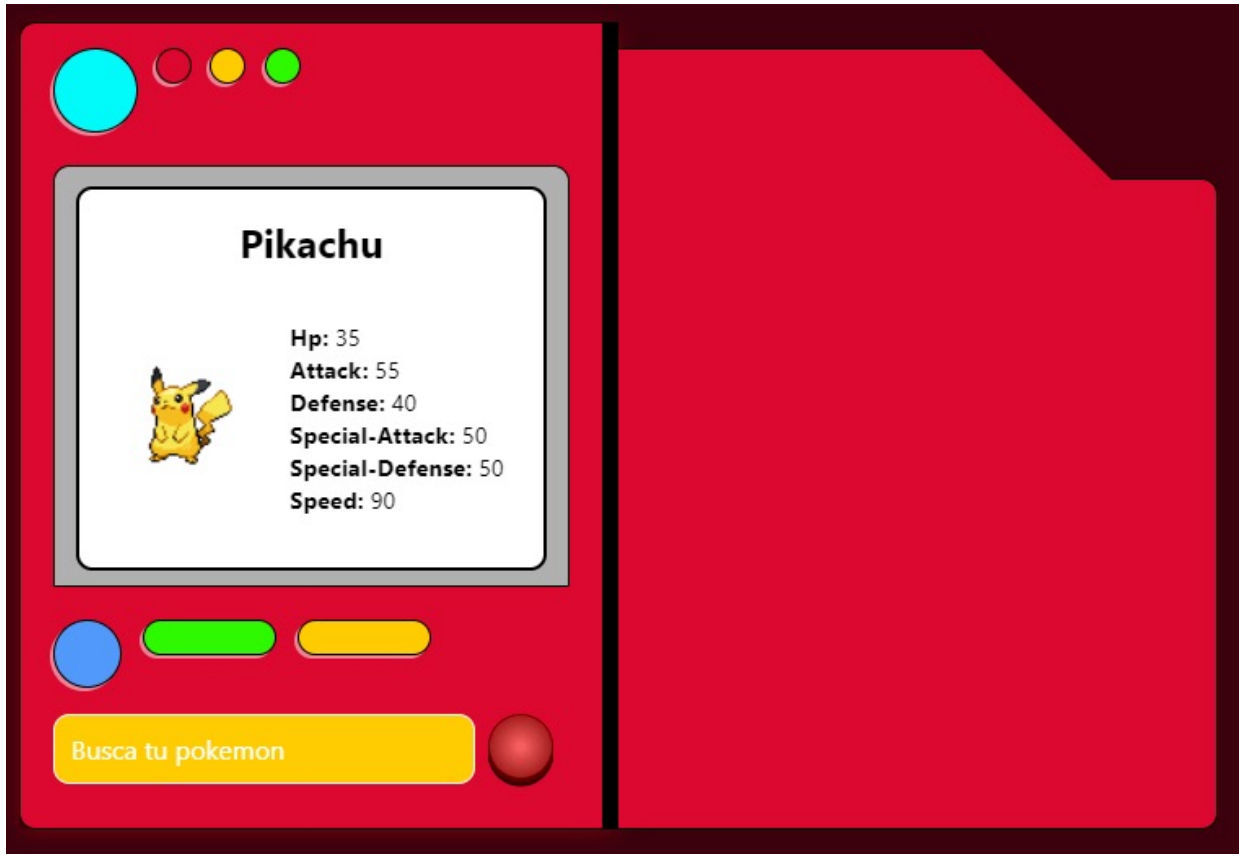
$$a_{nm} \geqslant 0$$

$$\sum_{m=1}^{N} a_{nm} = 1$$

Cuando estos coeficientes se determinan a partir del mismo token de entrada → Self-attention

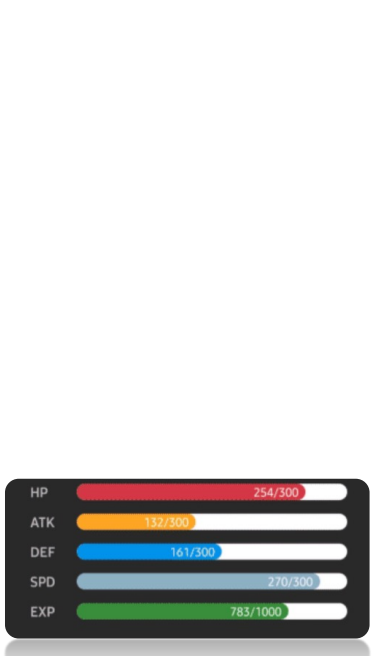
La tríada Query – Key – Value

Conceptos provenientes del campo de Information Retrieval



Fuente: https://github.com/skydoves/Pokedex

La tríada Query – Key – Value

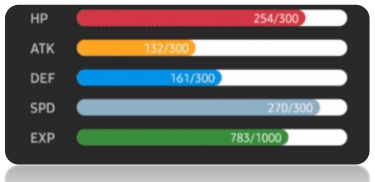


Query

Keys

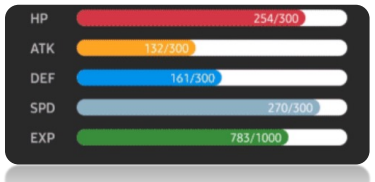
Value

La tríada Query – Key – Value



En la jerga de los Transformers, sería un caso de **hard-attention**

La tríada Query – Key – Value



$$\mathbf{y}_n = \sum_{m=1}^{N} a_{nm}\mathbf{x}_m$$

El resultado de una operación de **soft-attention** sería algo así

La tríada Query – Key – Value

En nuestra primera version de self-attention, usaremos las inputs $\mathbf{x}_i$ como **keys** y **queries**

Cómo comparar si una **key** y una **query** son similares? → $a_{nm} = \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m$

Dot-product self attention

Cómo normalizarlas para que →

$$a_{nm} \geqslant 0$$

$$\sum_{m=1}^{N} a_{nm} = 1$$

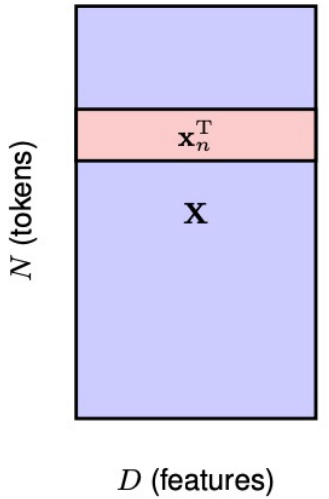
→ $$a_{nm} = \frac{\exp(\mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m)}{\sum_{m'=1}^{N} \exp(\mathbf{x}_n^{\mathrm{T}} \mathbf{x}_{m'})}$$

Dot-product Self-Attention

Queries . Key^T Values

$$\mathbf{Y} = \text{Softmax}\left[\mathbf{X}\mathbf{X}^T\right]\mathbf{X}$$

N x D N x D D x N N x D



Y los parámetros entrenables?

Los agregamos!

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{U}$$

N x D N x D D x D

Dot-product Self-Attention

Queries . Key^T Values

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{X}^{\mathrm{T}}\right]\mathbf{X}$$

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{U}\,\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\right]\mathbf{X}\mathbf{U}$$

$\mathbf{Q}$ $\mathbf{K}^{\mathrm{T}}$ $\mathbf{V}$

Para qué compartir pesos entre Q, K y V?

Dot-product Self-Attention

Cada una puede tener sus propios pesos

Parámetros entrenables

N x D_k $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$

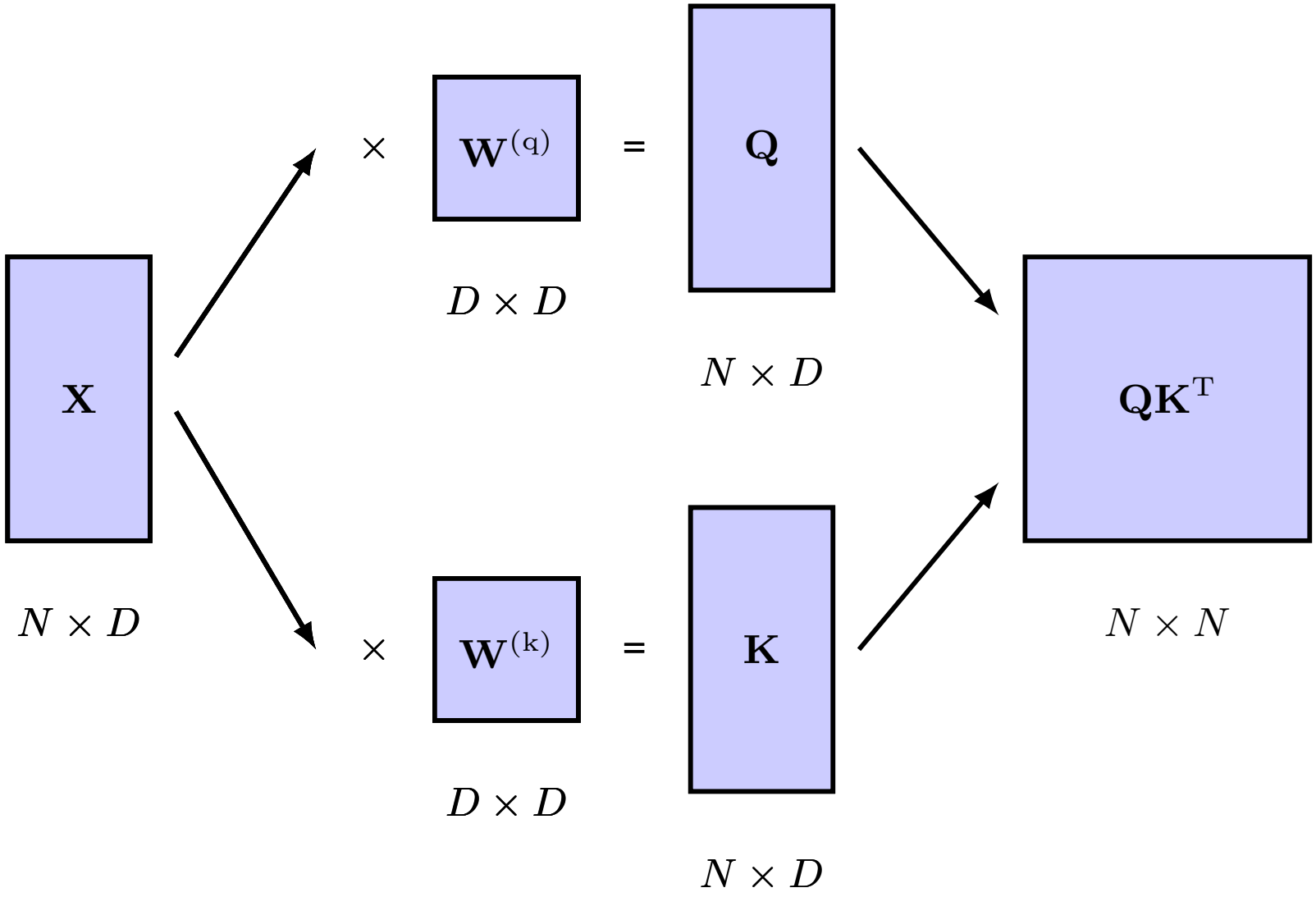
N x D_k $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$

N x D_v $\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$

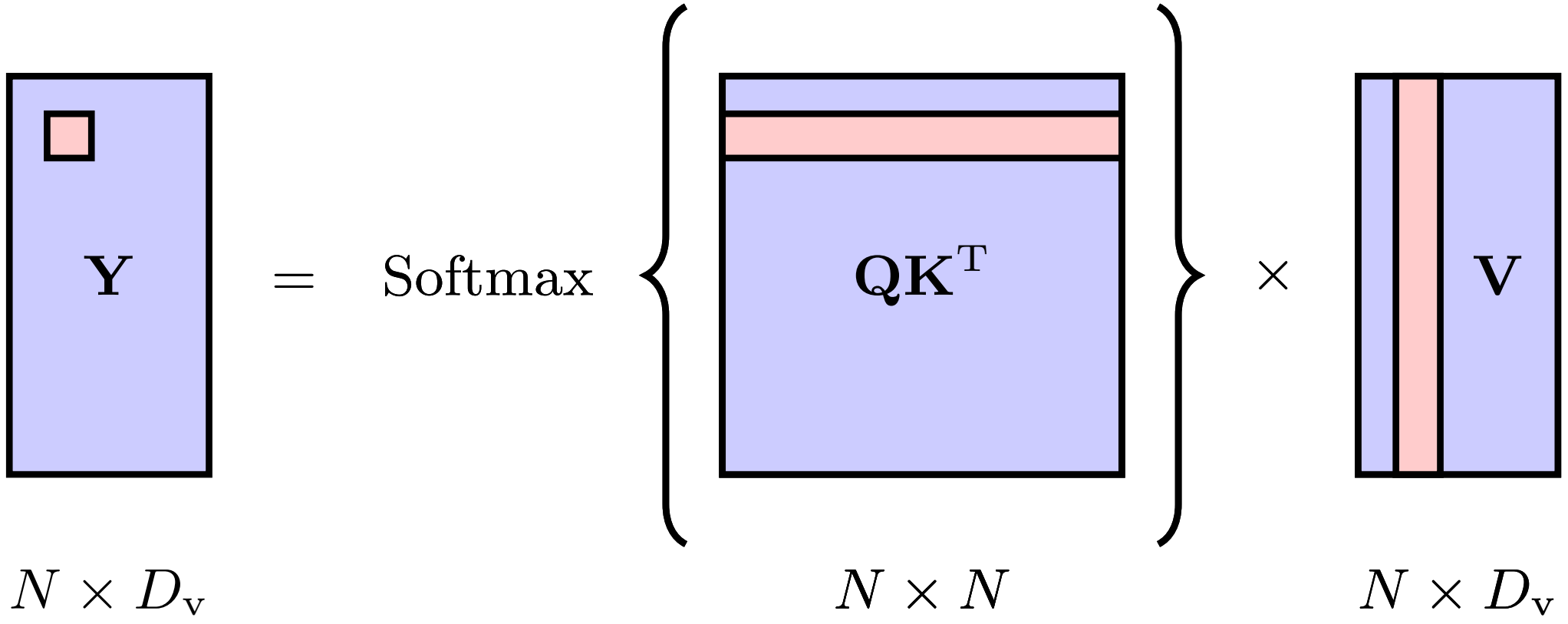
$$\mathbf{Y} = \text{Softmax}\left[\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right]\mathbf{V}$$

N x D_v N x D_k D_k x N N x D_v

Dot-product Self-Attention



Dot-product Self-Attention



Scaled Dot-product Self-Attention Layer

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right]\mathbf{V}$$

Problema: los gradients de Softmax se hacen muy pequeños para grandes valores de QK^T

Solución: escalarlos!

$$\mathbf{Y} = \mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) \equiv \mathrm{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{D_{\mathrm{k}}}}\right]\mathbf{V}$$

Scaled Dot-product Self-Attention Layer

Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Weight matrices $\{\mathbf{W}^{(\mathrm{q})}, \mathbf{W}^{(\mathrm{k})}\} \in \mathbb{R}^{D \times D_\mathrm{k}}$ and $\mathbf{W}^{(\mathrm{v})} \in \mathbb{R}^{D \times D_\mathrm{v}}$

Output: $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{N \times D_\mathrm{v}} : \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$

$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathrm{q})}$ `// compute queries` $\mathbf{Q} \in \mathbb{R}^{N \times D_\mathrm{k}}$

$\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathrm{k})}$ `// compute keys` $\mathbf{K} \in \mathbb{R}^{N \times D_\mathrm{k}}$

$\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathrm{v})}$ `// compute values` $\mathbf{V} \in \mathbb{R}^{N \times D}$

return $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^\mathrm{T}}{\sqrt{D_\mathrm{k}}}\right]\mathbf{V}$

Multi-head attention

- Podrían existir multiples patrones de atención que sean útiles al mismo tiempo.
- En NLP, uno podría estar asociado al tiempo verbal y otro a la semántica del vocabulario
- **Solución:** colocar multiples cabezas de atención en paralelo, con parámetros independientes

$$\mathbf{H}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$$

N x D_v

$$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(\text{q})}$$
$$\mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(\text{k})}$$
$$\mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(\text{v})}$$

Multi-head attention

- Estas cabezas se concatenan en una única matriz
- Y luego el resultado es linealmente transformado usando una nueva matriz de pesos $\mathbf{W}^{(o)}$

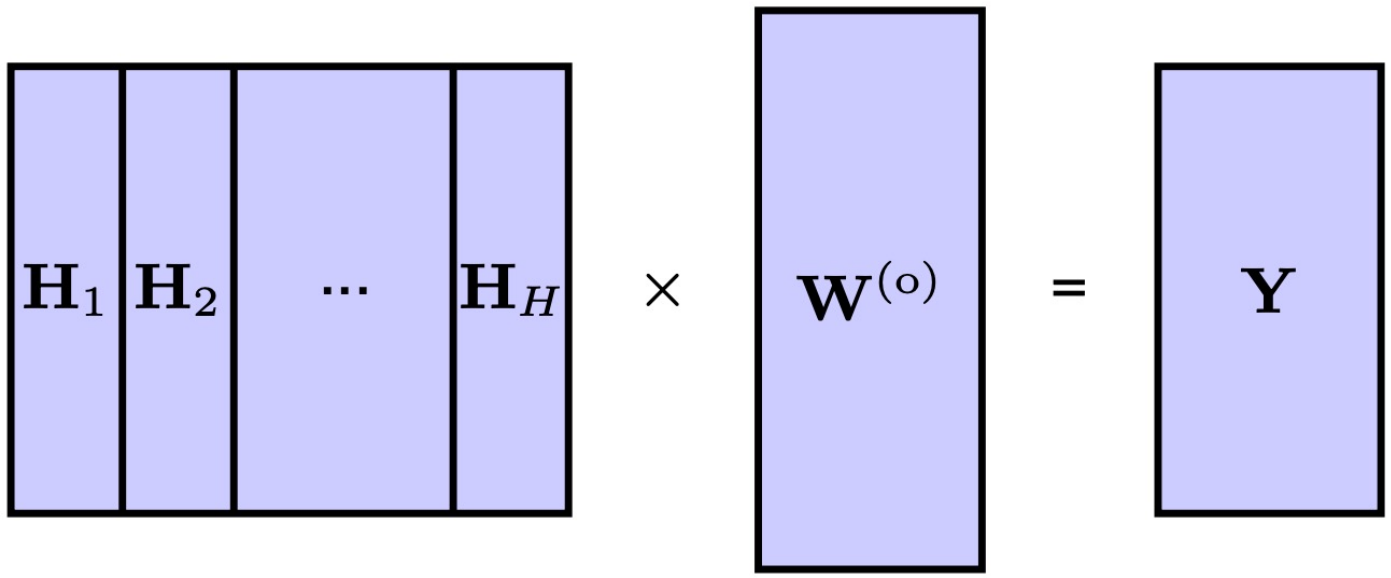
$$\mathbf{Y}(\mathbf{X}) = \text{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_H\right] \mathbf{W}^{(o)}$$

$N \times D_v$ $N \times D_v$

$N \times D_v$ $N \times HD_v$ $HD_v \times D_v$

Multi-head attention

$$\mathbf{Y}(\mathbf{X}) = \text{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_H\right] \mathbf{W}^{(\text{o})}$$



Usualmente, Dv = D/H para que la salida sea de dimension N x D

(recordar que D era la dimension del embedding de entrada)

Multi-head attention

Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Query weight matrices $\{\mathbf{W}_1^{(\mathrm{q})}, \ldots, \mathbf{W}_H^{(\mathrm{q})}\} \in \mathbb{R}^{D \times D}$

Key weight matrices $\{\mathbf{W}_1^{(\mathrm{k})}, \ldots, \mathbf{W}_H^{(\mathrm{k})}\} \in \mathbb{R}^{D \times D}$

Value weight matrices $\{\mathbf{W}_1^{(\mathrm{v})}, \ldots, \mathbf{W}_H^{(\mathrm{v})}\} \in \mathbb{R}^{D \times D_\mathrm{v}}$

Output weight matrix $\mathbf{W}^{(\mathrm{o})} \in \mathbb{R}^{H D_\mathrm{v} \times D}$

Output: $\mathbf{Y} \in \mathbb{R}^{N \times D} : \{\mathbf{y}_1, \ldots, \mathbf{x}_N\}$

// compute self-attention for each head

for $h = 1, \ldots, H$ **do**

$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{q})}, \quad \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{k})}, \quad \mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{v})}$

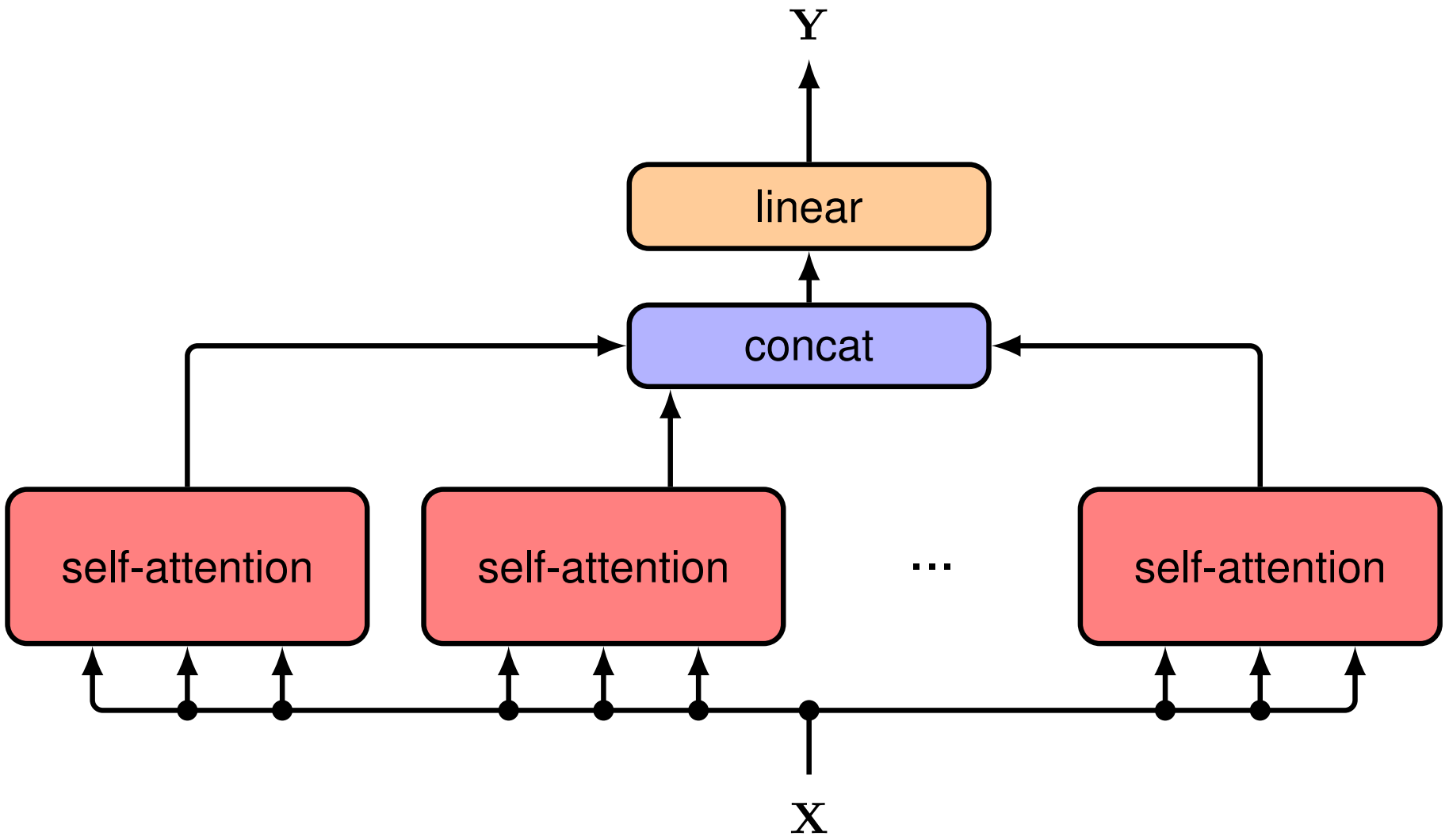
$\mathbf{H}_h = \mathrm{Attention}\left(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h\right)$ // $\mathbf{H}_h \in \mathbb{R}^{N \times D_\mathrm{v}}$

end for

$\mathbf{H} = \mathrm{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_\mathbf{N}\right]$ // concatenate heads

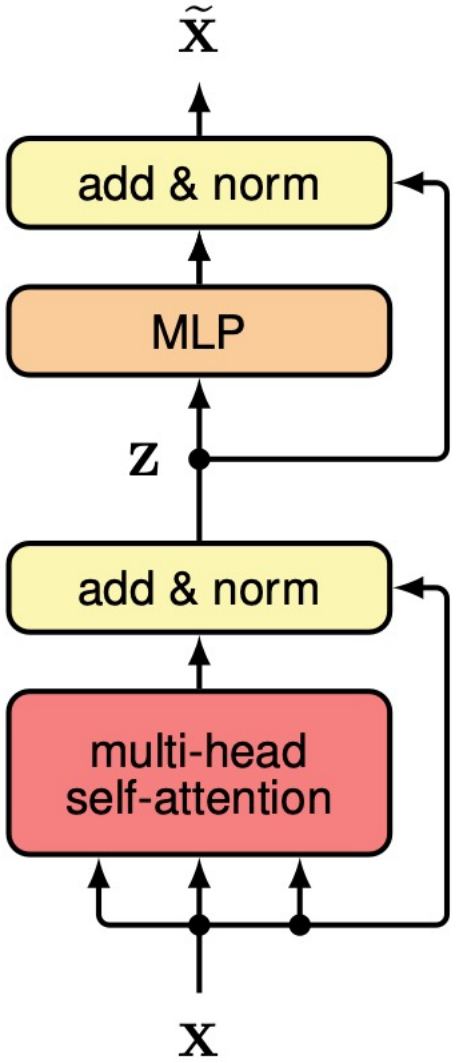
return $\mathbf{Y}(\mathbf{X}) = \mathbf{H}\mathbf{W}^{(\mathrm{o})}$

Multi-head attention



Transformer Layer

Incorpora tres elementos adicionales



Conexiones residuales

Perceptrón Multicapa

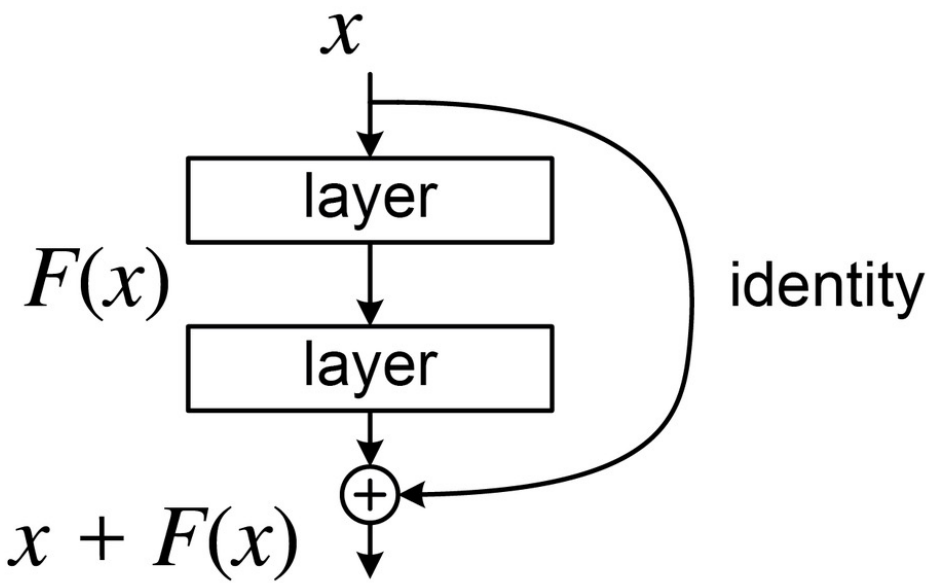
LayerNormalization

Transformer Layer

Incorpora tres elementos adicionales

Conexiones residuales

- Mejoran la dinámica de los gradientes
- Ayudan a entrenar modelos más profundos



Transformer Layer

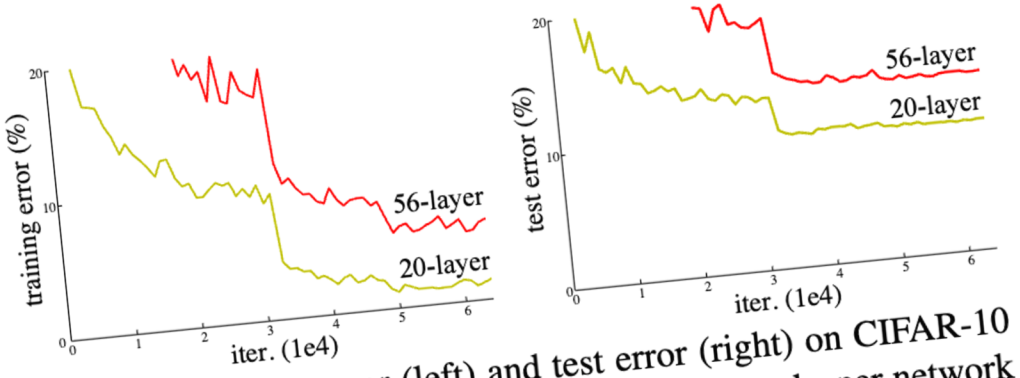
Conexiones Residuales

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used



(left) and test error (right) on CIFAR-10

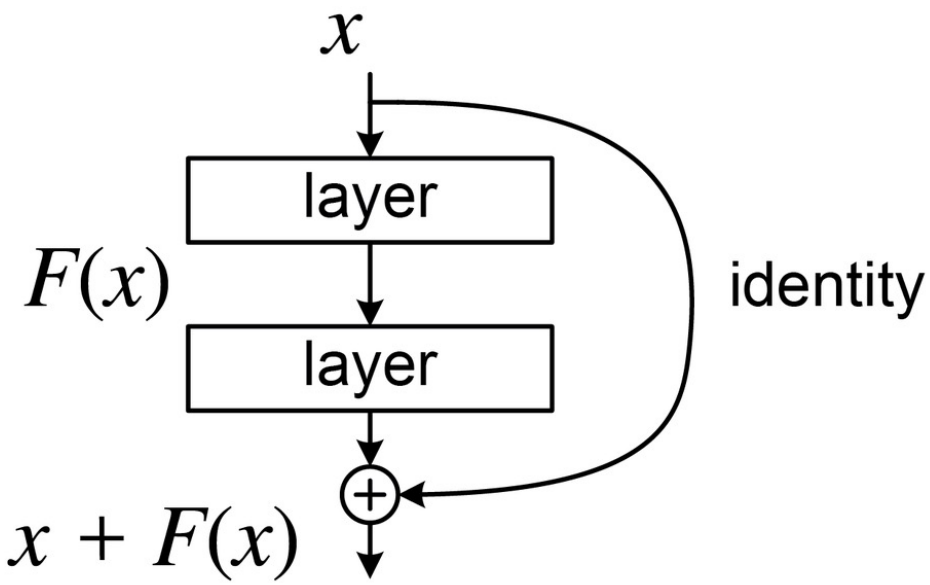
Transformer Layer

Conexiones Residuales

Conexiones residuales

- Mejoran la dinámica de los gradientes
- Ayudan a entrenar modelos más profundos
- En la práctica, implica:

$$\mathbf{Y}(\mathbf{X}) + \mathbf{X}$$

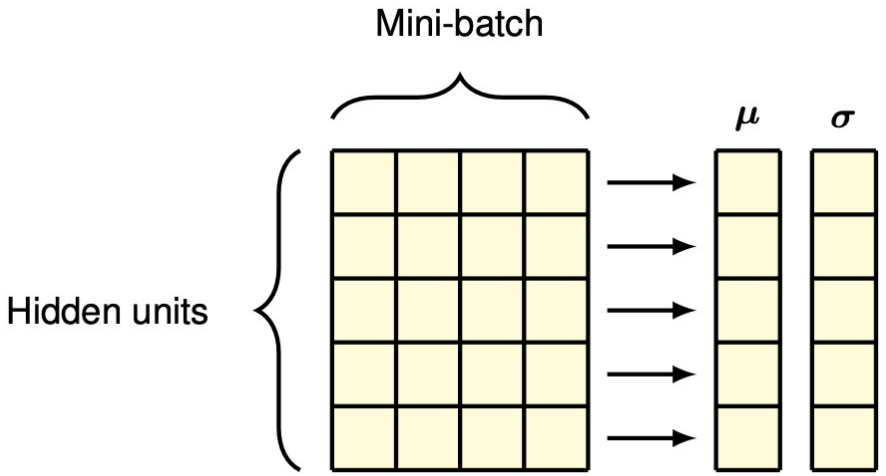


Transformer Layer

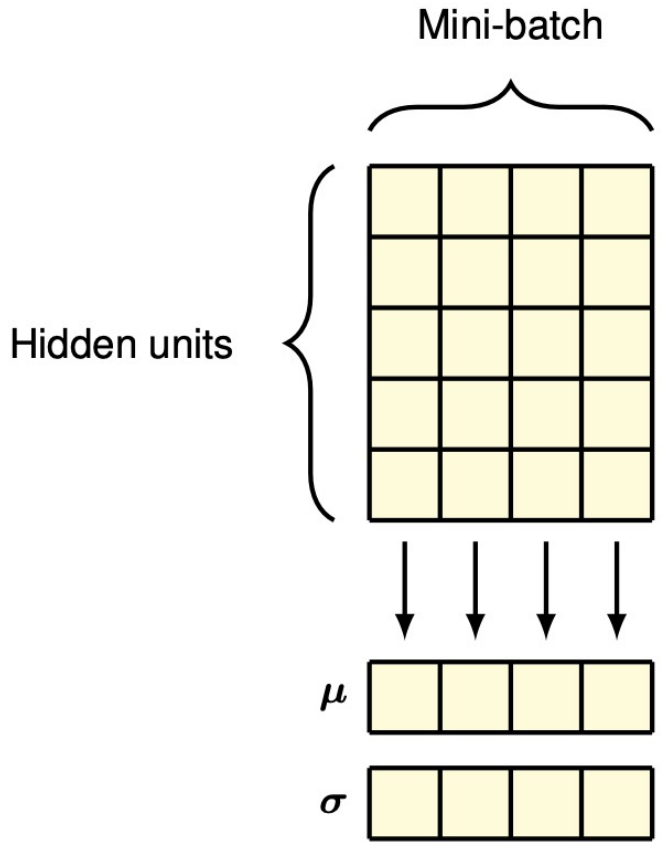
LayerNormalization

Layer Normalization

- Similar a BatchNormalization, pero normalizando a nivel de features en lugar de a nivel de batch sample.



BatchNorm



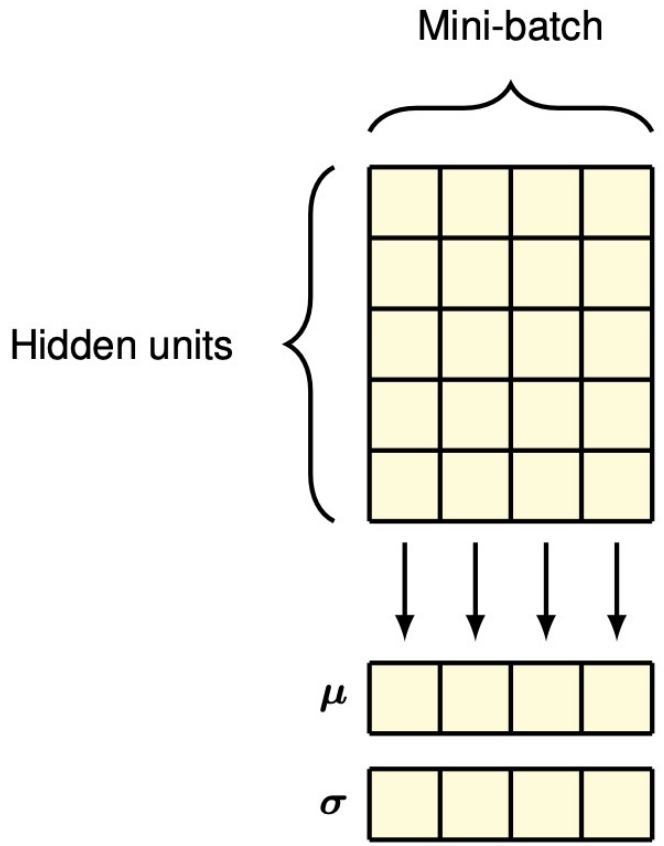
LayerNorm

Transformer Layer

LayerNormalization

Layer Normalization

- En BatchNorm, si el batch es muy pequeño, las estimaciones de media y desvío son muy ruidosas.
- La misma función de normalización se puede usar en training y test, dado que se computa sobre las features para cada sample independientemente.



LayerNorm

Transformer Layer

Conexiones Residuales + Layer Normalization

$$\mathbf{Z} = \mathrm{LayerNorm}\left[\mathbf{Y}(\mathbf{X}) + \mathbf{X}\right]$$

Transformer Layer

MLP Compartido

$$\mathbf{Y}(\mathbf{X}) = \text{Concat}\left[\mathbf{H}_1, \ldots, \mathbf{H}_H\right] \mathbf{W}^{(\text{o})}$$

Observando la composición de la capa de atención, de no ser por la función Softmax, el resto es lineal

Solución: Usar un MLP no lineal, con D inputs y D outputs

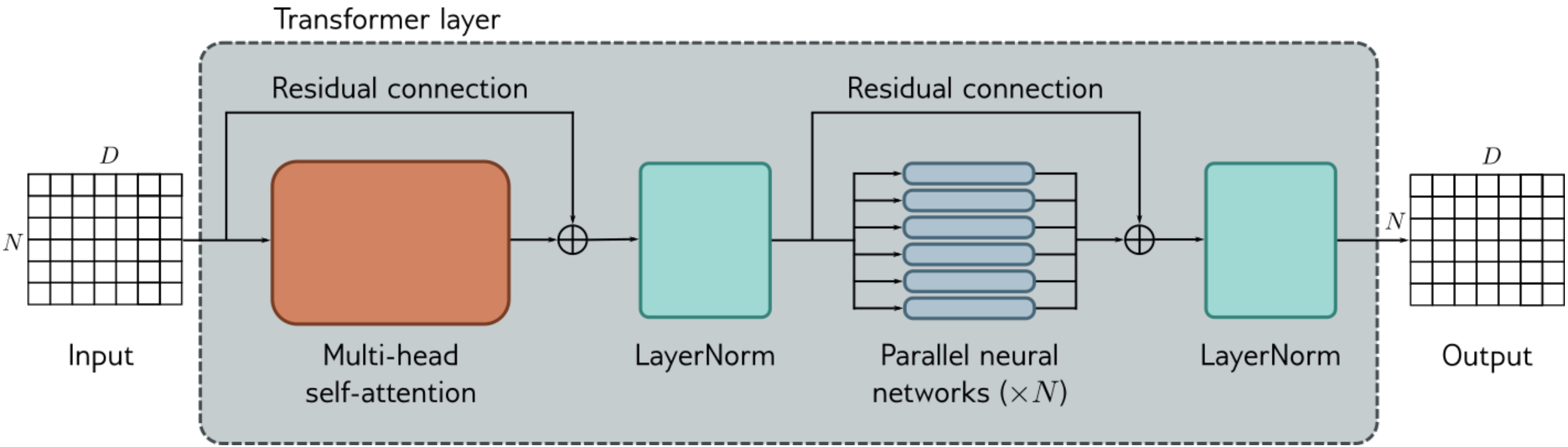
Transformer Layer

Conexiones Residuales + Layer Normalization + Shared MLP

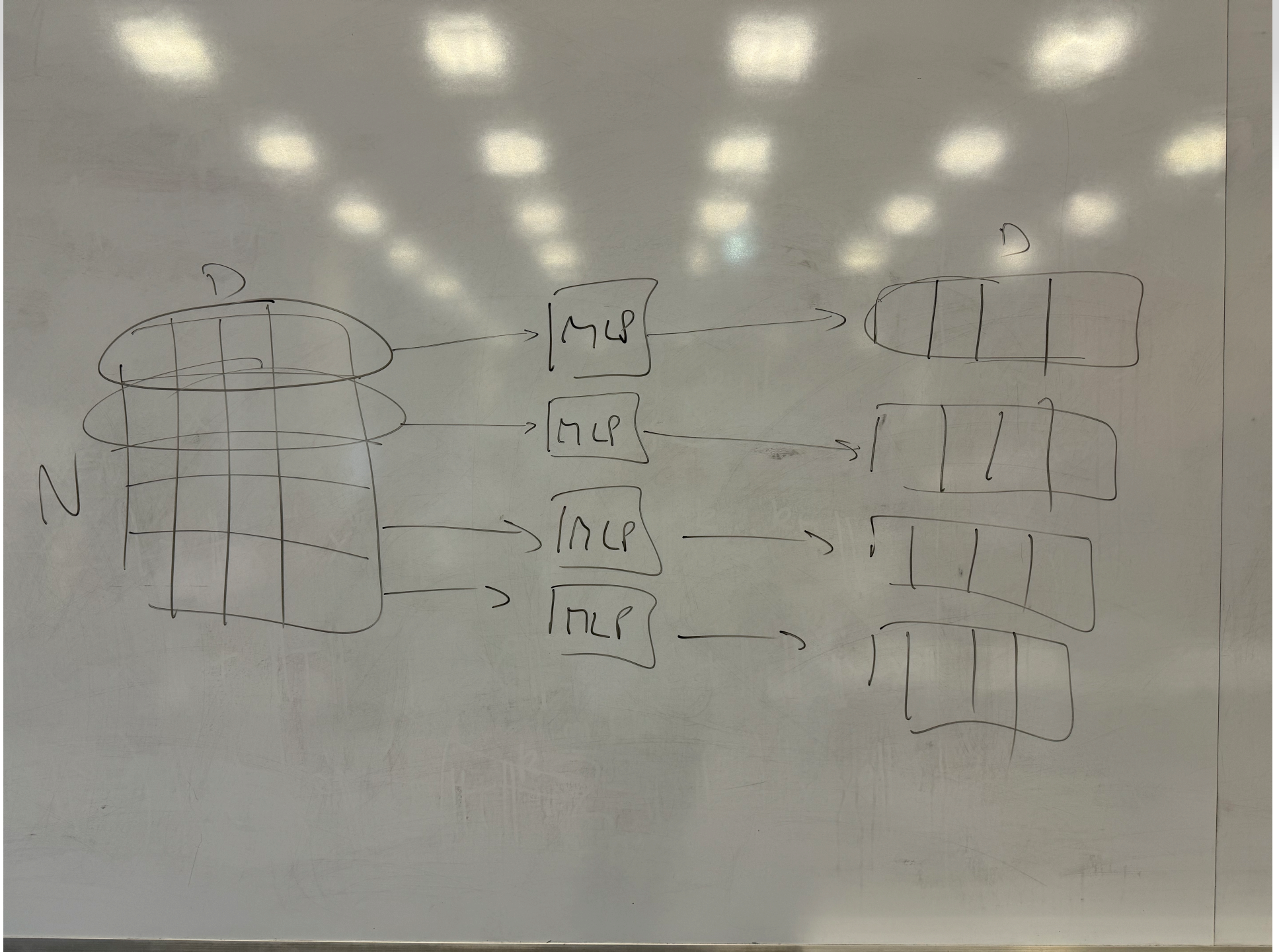
$$\tilde{\mathbf{X}} = \mathrm{LayerNorm}\left[\mathrm{MLP}\left[\mathbf{Z}\right] + \mathbf{Z}\right]$$

El perceptron es compartido por todos los tokens para permitir procesar sequences de tamaño arbitraro

Transformer Layer



El perceptron es compartido por los N tokens para permitir procesar secuencias de tamaño arbitrario


D
N
MLP
MLP
MLP
MLP
D

Positional encoding

Los transformers son equivariantes respecto a permutaciones en la entrada

(Un, perro, mordió, al, hombre) → (v_1, v_2, v_3 , v_4 , v_5)

(Un, hombre, mordió, al, perro) → (v_1, v_5 , v_3 , v_4, v_2)

Cómo tomar en cuenta el órden para definer el significado?

Solución: Positional encondings

Positional embeddings

Codifican la posición del token en el mismo dato de entrada

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$$

Token original Positional embedding

- Tienen la misma dimension que los tokens de entrada
- Tienen un valor diferente para cada posición

Positional embeddings

Codifican la posición del token en el mismo dato de entrada

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$$

Token original Positional embedding

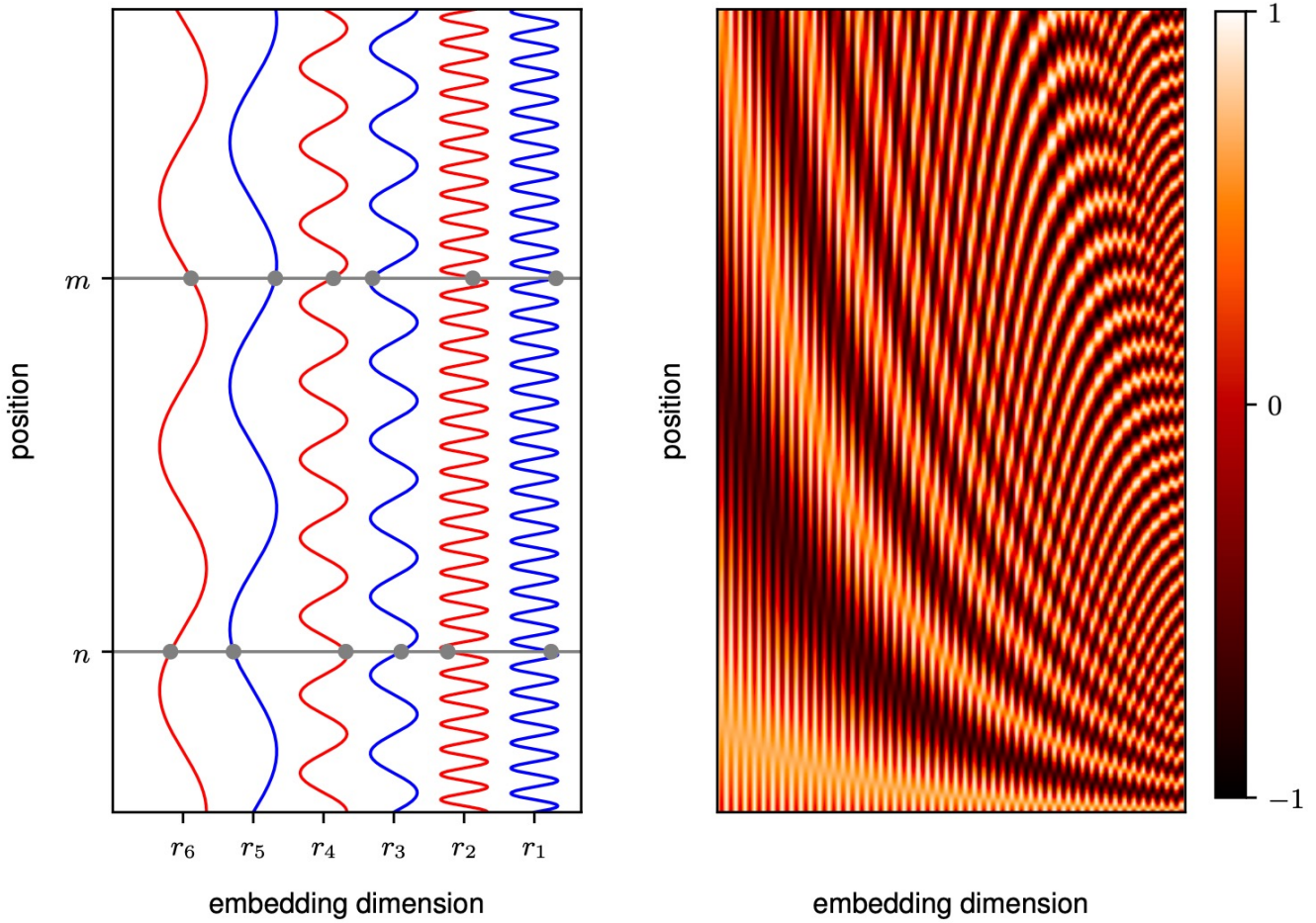
- **Idea 1:** asociar un entero 1, 2, 3, 4, …. a cada posición
  - Problema: la magnitude se incrementa sin restricción de valor
- **Idea 2:** asociar un valor entre (0,1)
  - Problema: si fueran secuencias de diferente longitud, los valores cambiarían

Positional embeddings

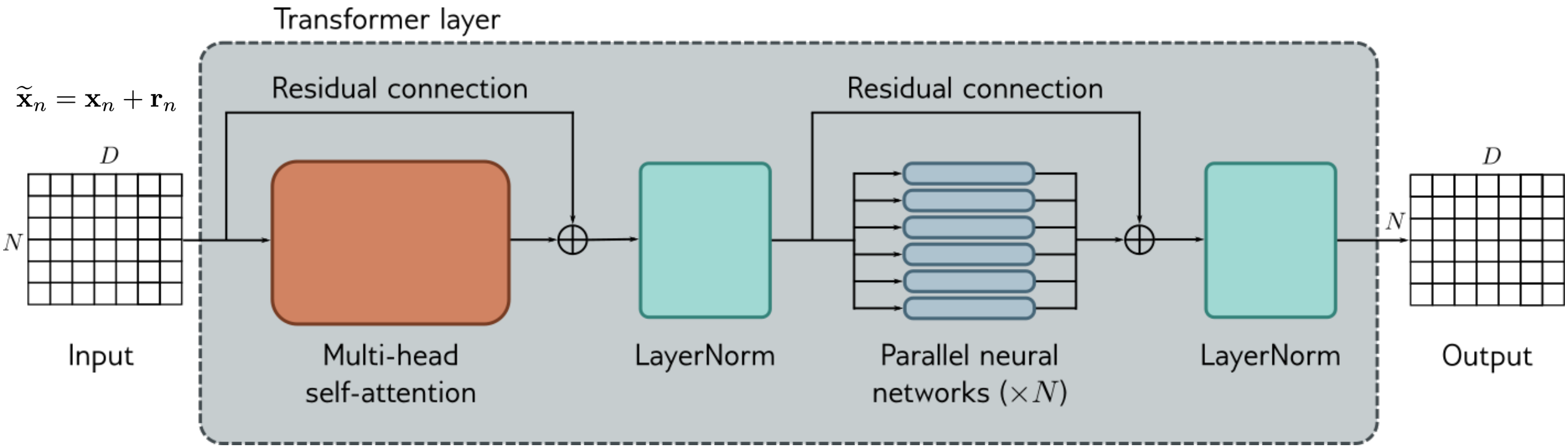
Idea 3: positional encodings basados en sinusoidales

$$\widetilde{\mathbf{x}}_n = \mathbf{x}_n + \mathbf{r}_n$$

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even,} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd.} \end{cases}$$



Transformer Layer



Transformer Models

Los tokes son procesados por sucesivas capas de transfomers

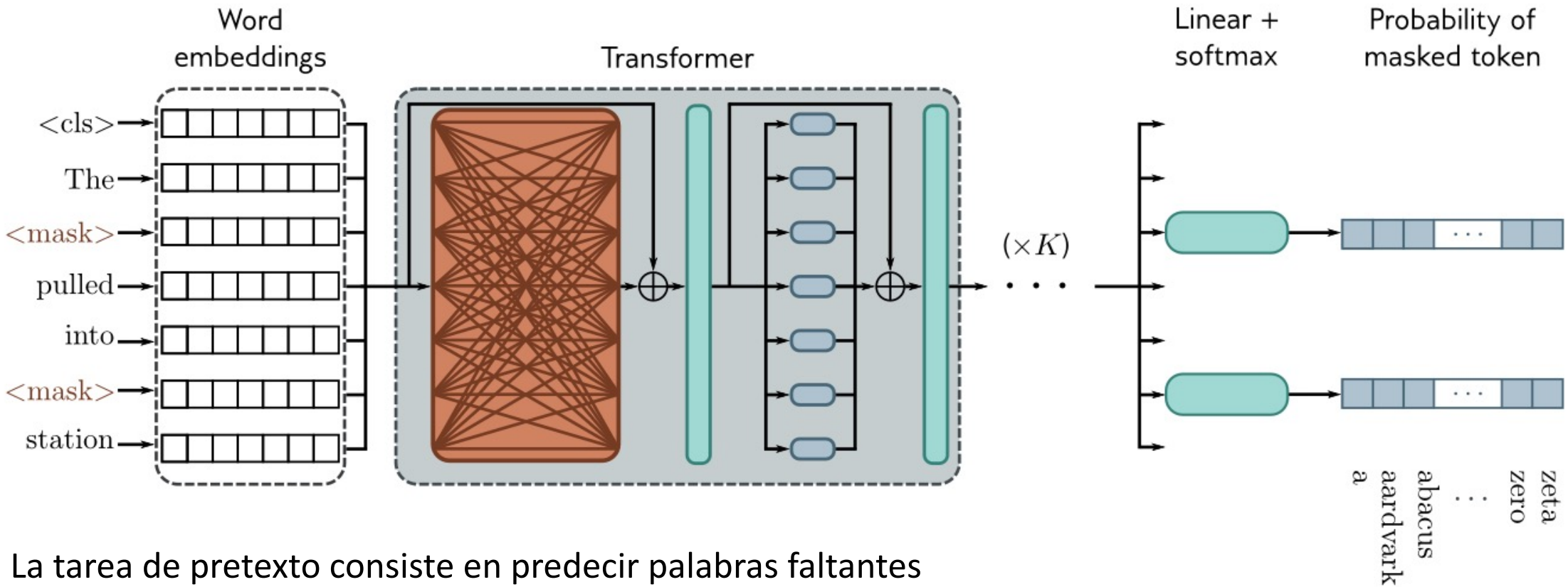
Encoder	Transforma embeddings en nuevas representaciones que pueden ser usadas para downstream tasks como clasificación.
Decoder	Predice el próximo token para continuar el texto de entrada
Encoder-decoder	Son usados en tareas de secuencia a secuencia, donde una cadena es convertida en otra

Modelos disponibles en Huggingface

Model	Examples	Tasks
Encoder	ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa	Sentence classification, named entity recognition, extractive question answering
Decoder	CTRL, GPT, GPT-2, Transformer XL	Text generation
Encoder-decoder	BART, T5, Marian, mBART	Summarization, translation, generative question answering

Modelo Encoder

Pre-training en BERT via Self-supervision



- La tarea de pretexto consiste en predecir palabras faltantes
- En training, la máxima longitud de las entradas es 512 tokens, y el batch-size 256
- El Sistema se entrenó por 1 millón de pasos, aproximadamente 50 épocas en un corpus de 3.3 billones de palabras.

Modelo Encoder

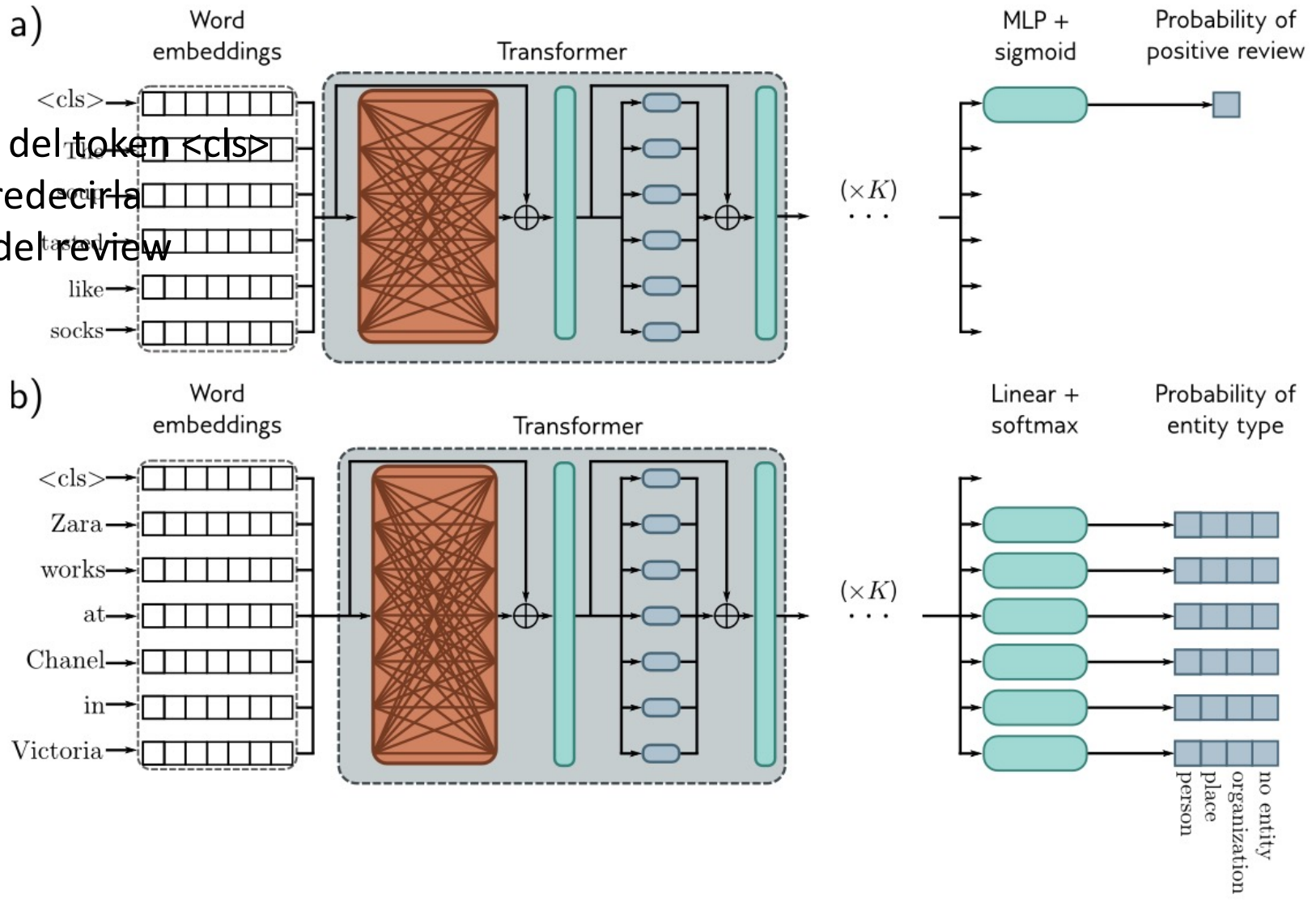
Ejemplo: BERT

- Vocabulary of 30,000 tokens.
- Input tokens are converted to 1024-dimensional word embeddings and passed through 24 transformer layers.
- Each contains a self-attention mechanism with 16 heads.
- The queries, keys, and values for each head are of dimension 64 (i.e., the matrices $\Omega_{vh}, \Omega_{qh}, \Omega_{kh}$ are 1024×64).
- The dimension of the single hidden layer in the fully connected networks is 4096.
- The total number of parameters is $\sim$ 340 million.

Modelo Encoder

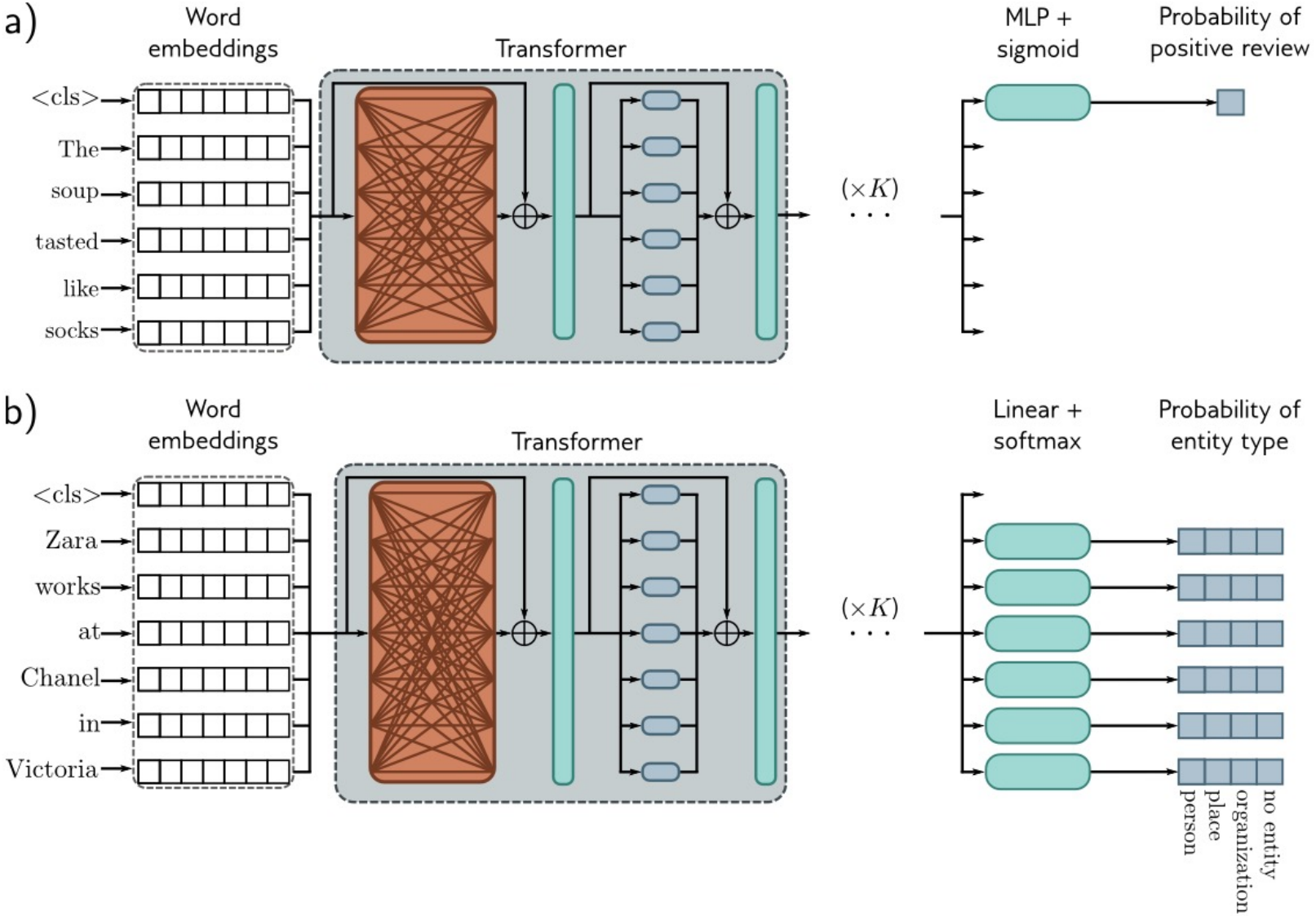
Fine-tunning de BERT para tareas downstream

El embedding del token <cls> se usa para predecirla probabilidad del review positivo



Modelo Encoder

Fine-tunning de BERT para tareas downstream

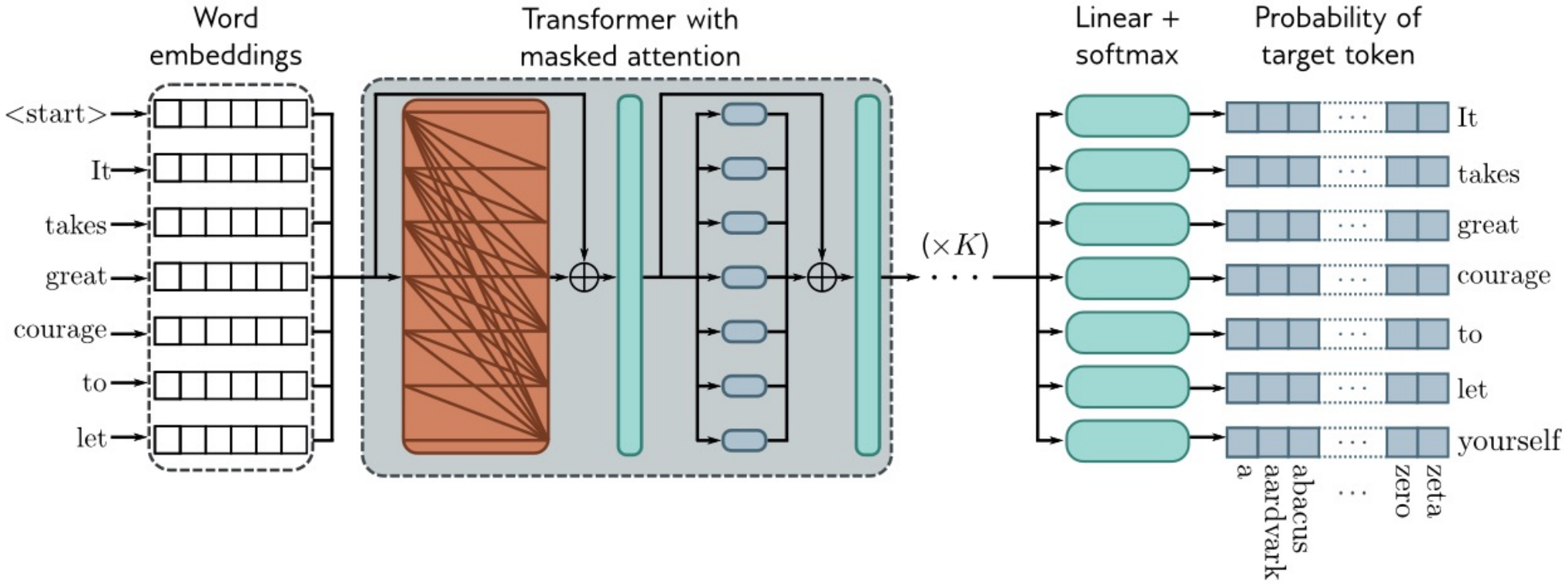


Ej: El embedding del token <cls> se usa para predecirla probabilidad de si review es o nopositivo

Ej: Clasificación de palabras en un problema de named entity recognition

Modelo Decoder

Ejemplo: GPT 3



- La arquitectura es muy similar al modelo Encoder (una serie de bloques Transformer operando sobre embeddings a nivel token)

- Pero el objetivo es diferente: predecir el próximo token en una secuencia. Su salida es la probabilidad por token del vocabulario completo.

Masked Self-Attention

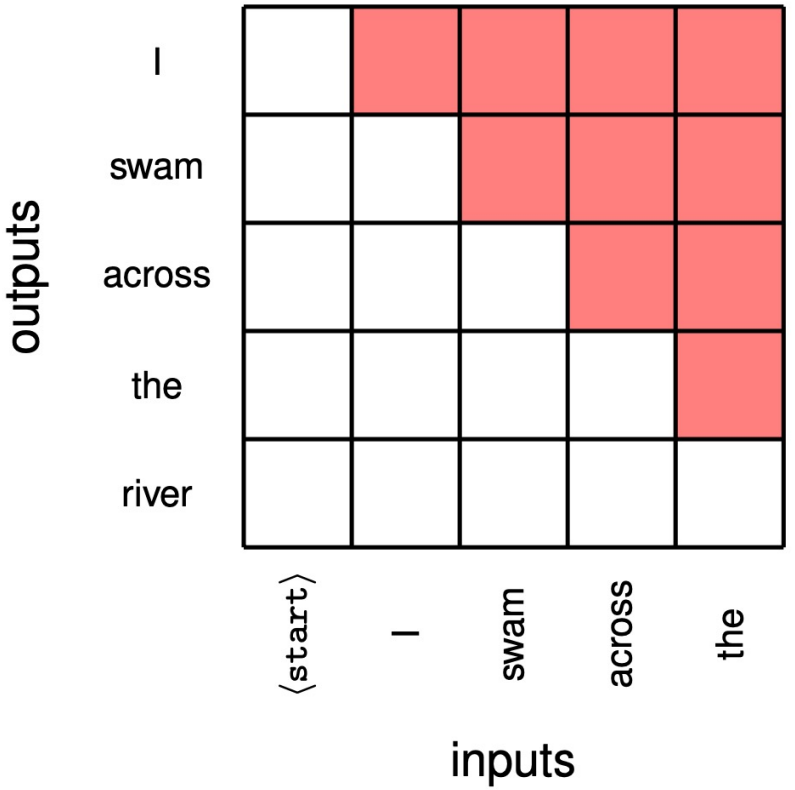
Ejemplo: GPT 3

- Al entrenar un modelo decoder, la loss se modela para maximizar la probabilidad de la próxima palabra en un esquema autoregresivo.

- Si pasamos la frase entera, el modelo tiene acceso a toda la secuencia!

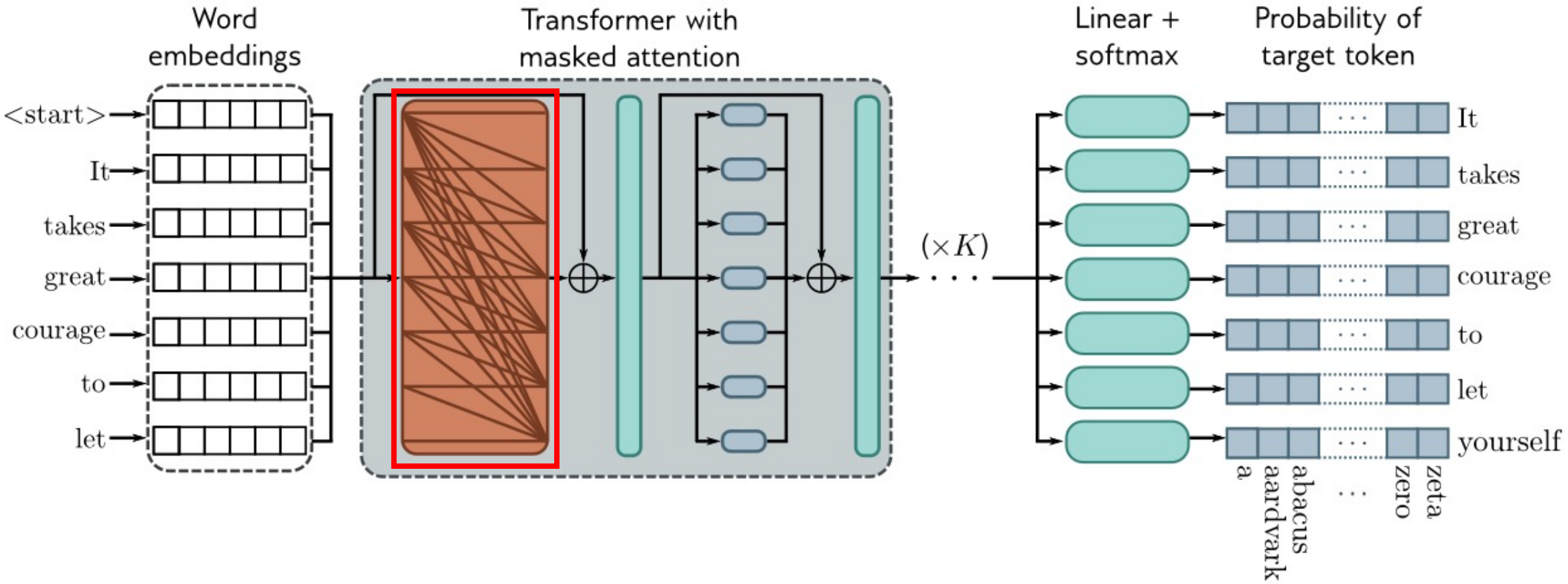
- Solución: **masked-self attention** → setear a 0 los coeficientes de atención de las palabras futuras

- En rojo se muestran los coeficientes en 0. Por ejemplo, al predecir “Across”, el modelo solo puede atender a (<start>, I, swam)

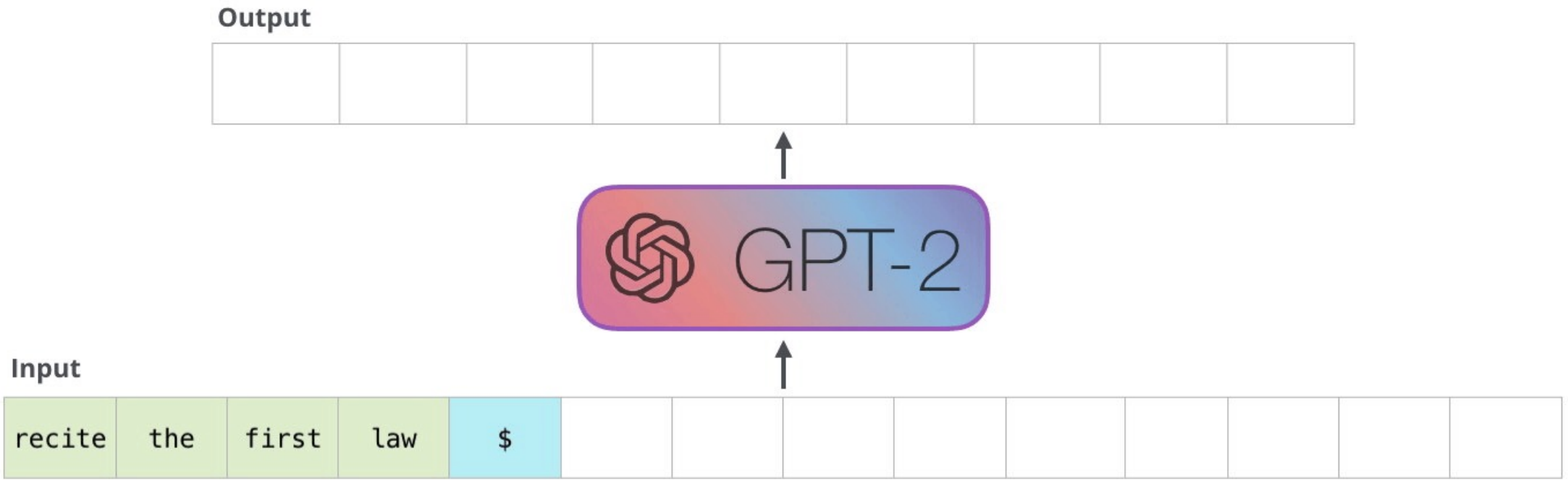


Masked Self-Attention

Ejemplo: GPT 3



Decoder: generando texto



Modelo Decoder

Ejemplo: GPT 3

- GPT3, the sequence lengths are 2048 tokens long, and the total batch size is 3.2 million tokens.
- There are 96 transformer layers, each processing a word embedding of size 12288.
- There are 96 heads in the self-attention layers, and the value, query, and key dimension is 128.
- It is trained with 300 billion tokens and contains 175 billion parameters.

Few Shot Learning

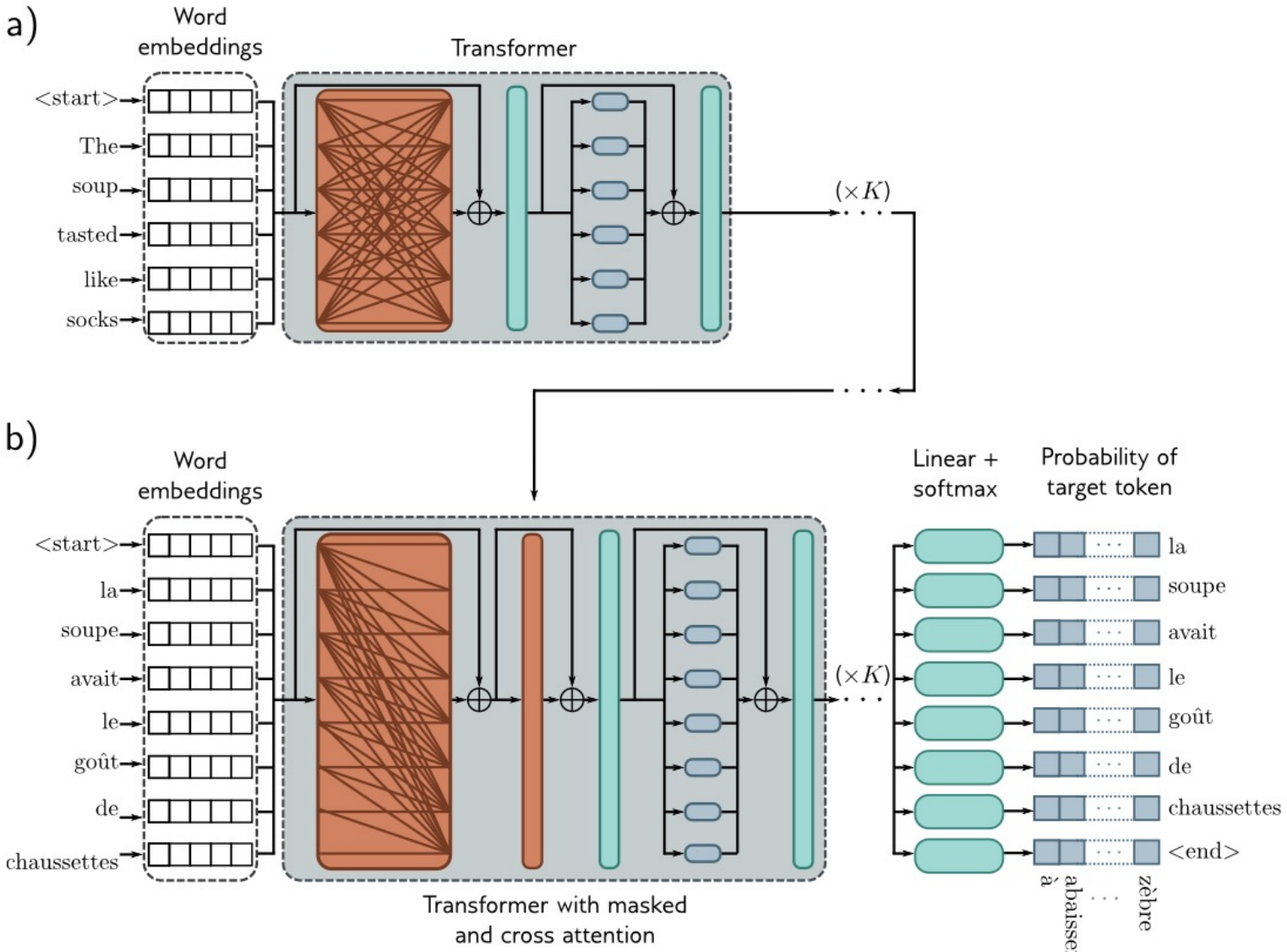
Realizando nuevas tareas sin fine-tunning en GPT-3

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

(result from Brown et al., 2020)

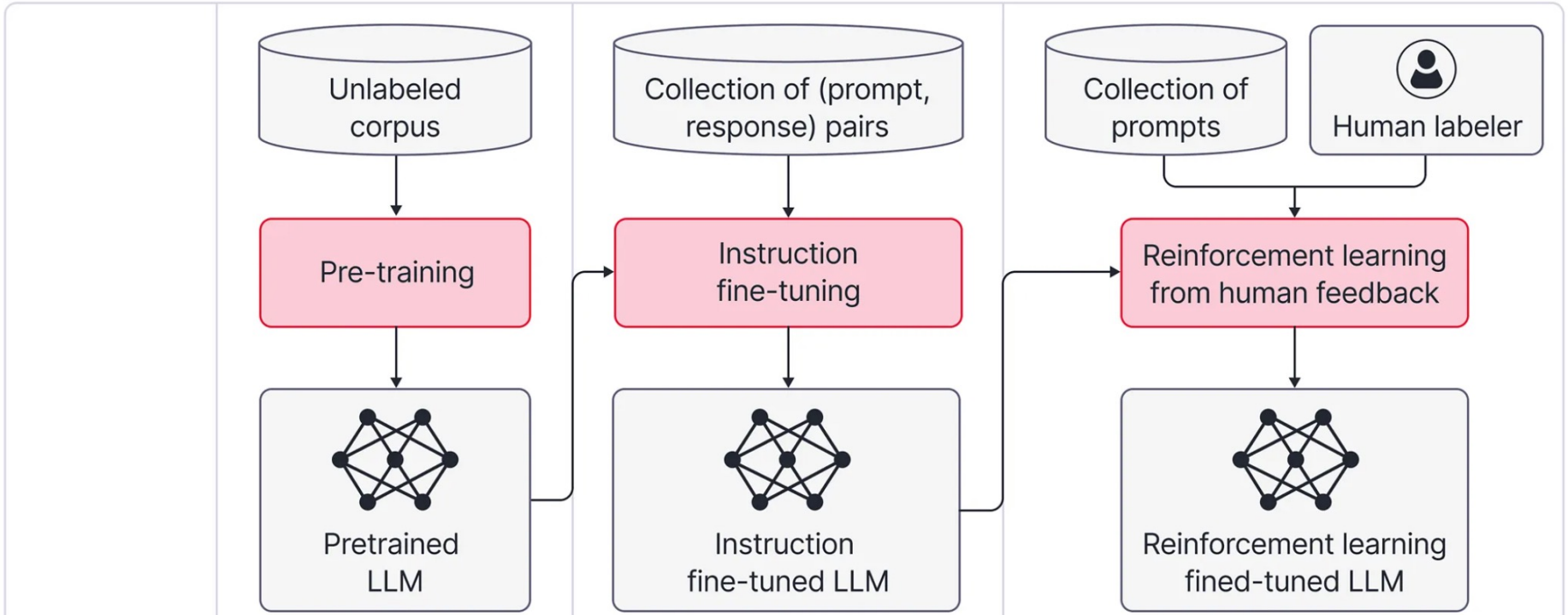
En few-shot learning, se proven una serie de ejemplos de contexto y luego el modelo completa con lo más probable.

Modelos Encoder-Decoder



- La 1era oración se pasa por el encoder, generando embeddings para cada token
- La 2^{nda} oración se pasa por el decoder que usa masked self attention, pero también presta atención a los embeddings de salida del encoder usando cross-attention (rectangulo naranja)
- La loss se computa como en el encoder, maximizando la probabilidad de cada palabra traducida

Entrenando LLMs

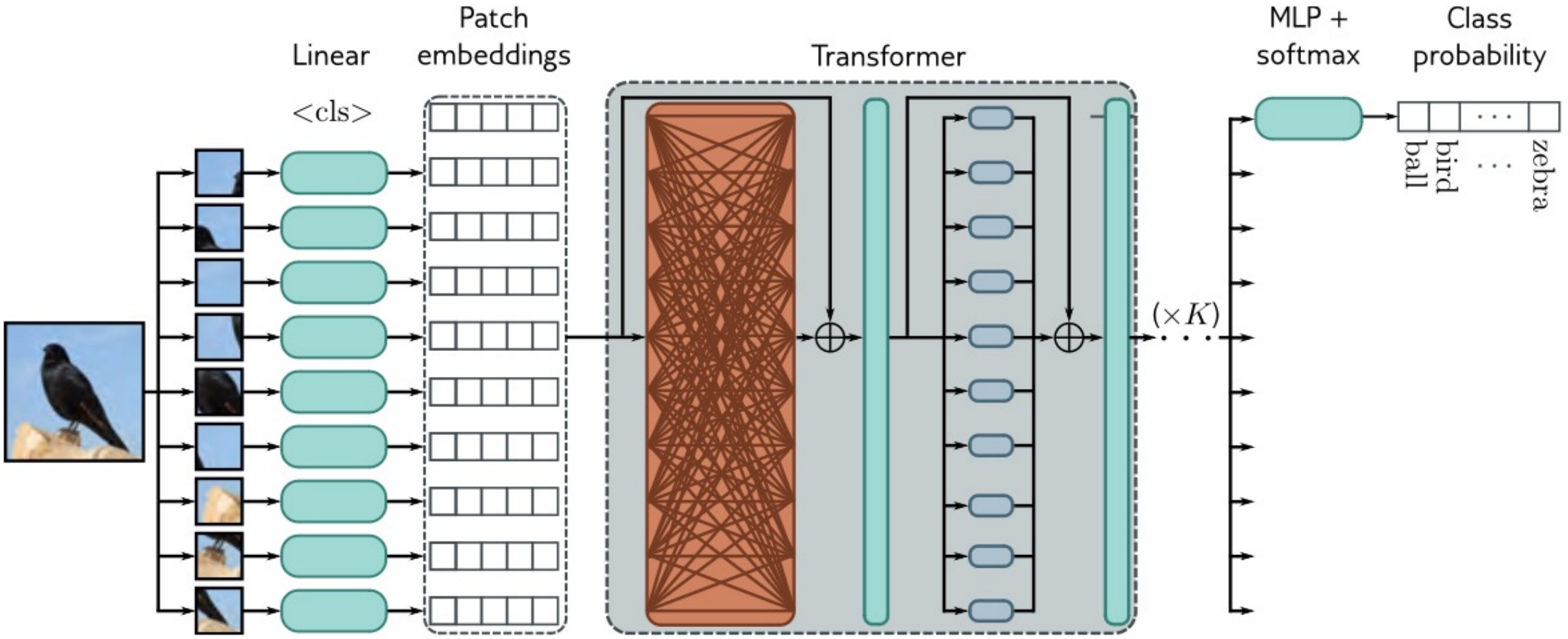


Dataset size	x00 billions to 1.x trillion tokens	~xk to x0k (prompt, response)	~x0k prompt
Example of models	GPT-3, LLaMA, Falcon, BLOOM	Dolly-v2, Falcon-instruct	Claude, GPT-4, ChatGPT
	(a)Pre-training	(b) Instruction fine-tuning	(c) Reinforcement learning from human feedback

Más info: https://www.borealisai.com/research-blogs/a-high-level-overview-of-large-language-models/

LLMs actuales

Vision Transformers



- Las imágenes son transformadas en secuencias de parches, y cada parche es proyectado a un embedding usando una transformación lineal aprendida.
- El modelo es entreando con supervised learning.

Modelos multimodales

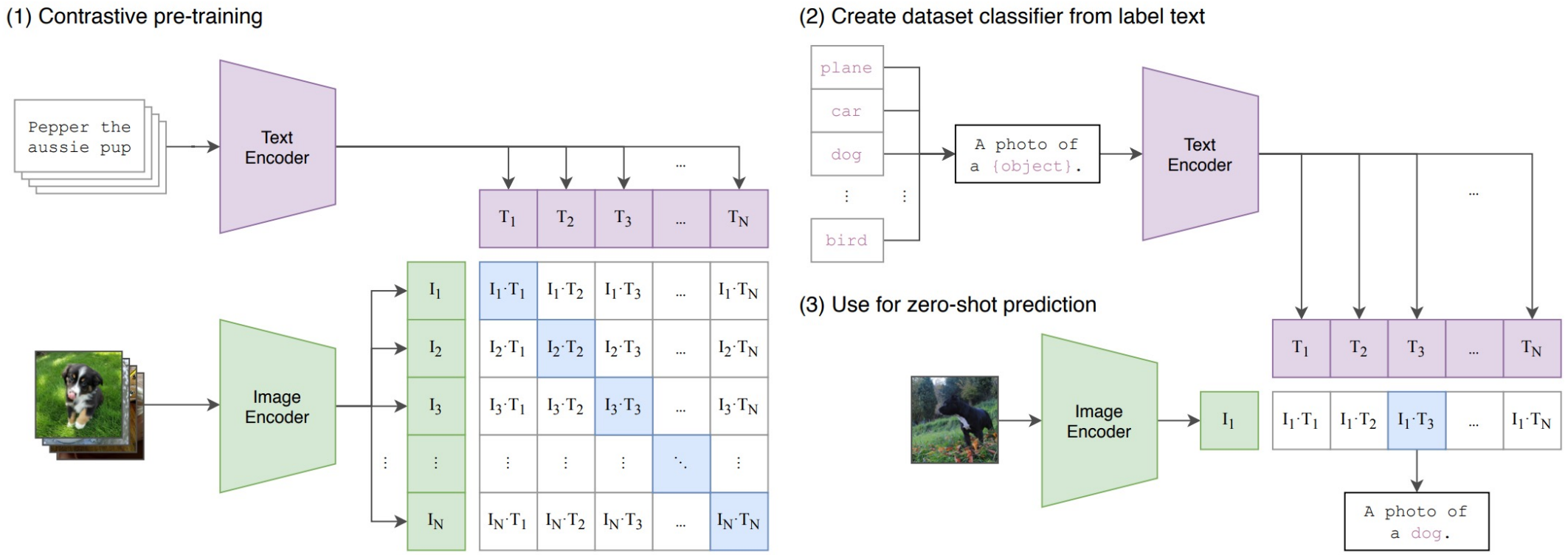


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Modelos multimodales

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] **Jong Wook Kim** [*1] **Chris Hallacy** [1] **Aditya Ramesh** [1] **Gabriel Goh** [1] **Sandhini Agarwal** [1]
Girish Sastry [1] **Amanda Askell** [1] **Pamela Mishkin** [1] **Jack Clark** [1] **Gretchen Krueger** [1] **Ilya Sutskever** [1]

Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which im-

Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of "text-to-text" as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset

ICML 2021