

# Análisis de Textos

Grupo PLN – InCo  
2024

# Deep Learning para PLN

# Deep Learning para PLN

- La entrada de las redes es el texto plano.
- Normalmente no se agregan características (features).
- La red aprende esas características como parte del proceso de entrenamiento.
- Las redes profundas (varias capas) son especialmente buenas para inferir esas características (representation learning).
- Generalmente se usan embeddings preentrenados para convertir las palabras en vectores y usarlos como entrada de la red.
- Si cada entrada es un texto (más de una palabra), se concatenan los vectores (llevando a un largo fijo). También pueden promediarse.
- Arquitectura Transformers: los vectores de cada palabra de la entrada se procesan en paralelo, y se combinan.

# Deep Learning para PLN

- Diferentes arquitecturas:
  - FeedForward: cada entrada va siendo procesada y transferida hasta la capa de salida.
  - Redes recurrentes (RNN): cada entrada es procesada y la información generada se transfiere hacia las capas siguientes y también hacia las neuronas vecinas de la misma capa.
  - Redes convolucionales: se usan principalmente para imágenes pero también para lenguaje. Se suelen combinar con capas LSTM.
- Usadas para diferentes problemas: clasificación, generación de lenguaje, modelos de lenguaje.

# Modelos de Lenguaje

# Modelos de Lenguaje

- Un modelo de lenguaje permite asignar una probabilidad a una secuencia de palabras.

# Modelos de Lenguaje

- Un modelo de lenguaje permite asignar una probabilidad a una secuencia de palabras.
- Podemos usarlo para predecir la palabra más probable para continuar una secuencia.

# Modelos de Lenguaje

- Un modelo de lenguaje permite asignar una probabilidad a una secuencia de palabras.
- Podemos usarlo para predecir la palabra más probable para continuar una secuencia.
- Fueron muy usados para corrección de textos, procesamiento de habla, traducción, entre otros, actualmente se usan para todas las tareas de PLN.



# Modelos de Lenguaje

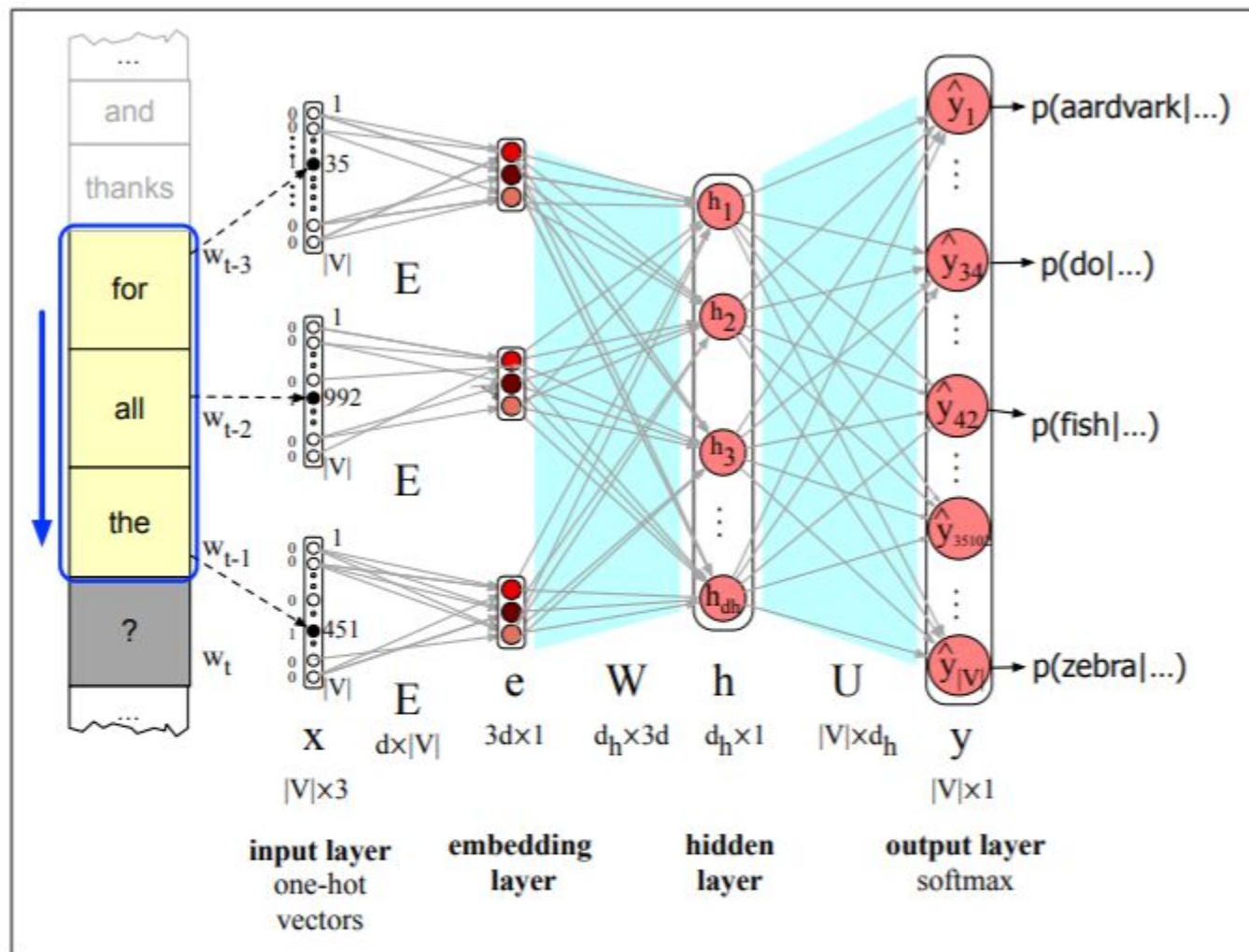
- Un modelo de lenguaje permite asignar una probabilidad a una secuencia de palabras.
- Podemos usarlo para predecir la palabra más probable para continuar una secuencia.
- Fueron muy usados para corrección de textos, procesamiento de habla, traducción, entre otros, actualmente se usan para todas las tareas de PLN.
- Originalmente se construían en base a conteos de n-gramas.
- Luego se construyeron modelos neuronales modelados con redes feed forward o recurrentes.
- El estado del arte son los modelos de lenguaje basados en la arquitectura **Transformers**.

# Modelos de Lenguaje

- Un modelo de lenguaje permite asignar una probabilidad a una secuencia de palabras.
- Podemos usarlo para predecir la palabra más probable para continuar una secuencia.
- Fueron muy usados para corrección de textos, procesamiento de habla, traducción, entre otros, actualmente se usan para todas las tareas de PLN.
- Originalmente se construían en base a conteos de n-gramas.
- Luego se construyeron modelos neuronales modelados con redes feed forward o recurrentes.
- El estado del arte son los modelos de lenguaje basados en la arquitectura **Transformers**.
- Dos grandes familias:
  - bidireccionales o enmascarados (BERT: Bidirectional Encoder Representations from Transformers)
  - generativos o autorregresivos (GPT: Generative Pre-trained Transformer).

# Modelos de Lenguaje

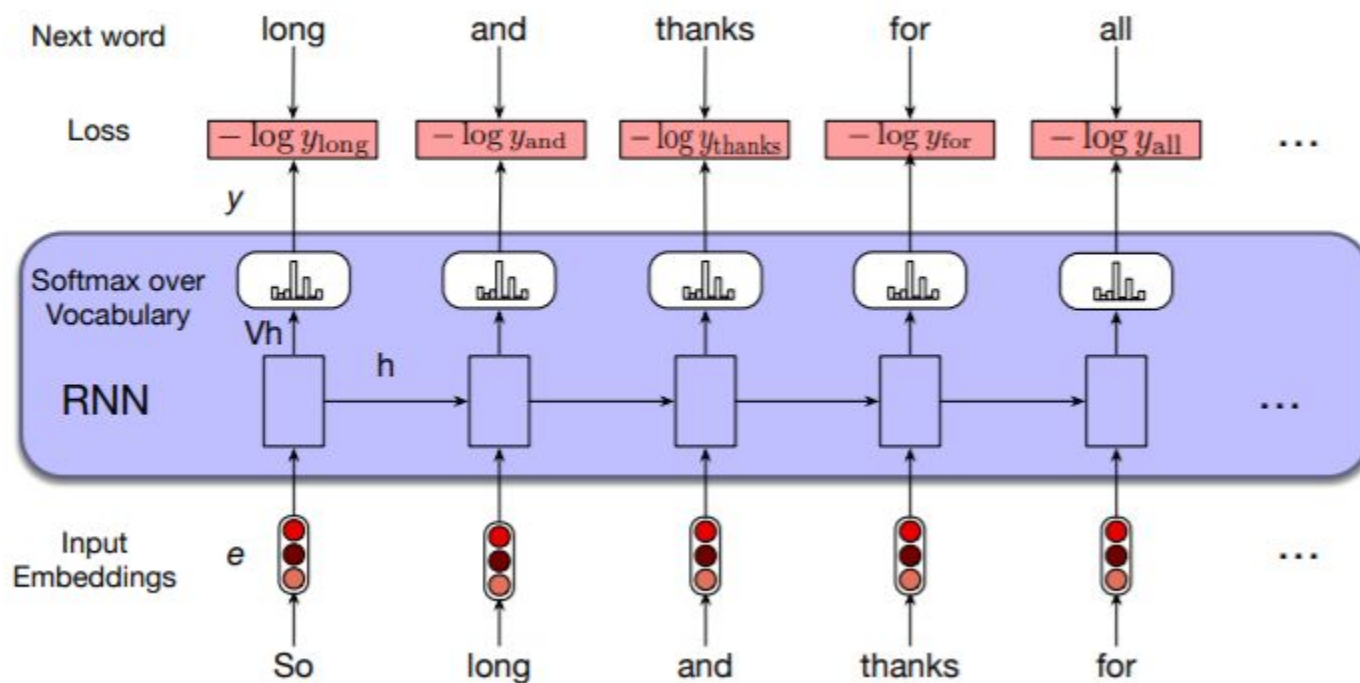
Modelo de lenguaje neuronal (feed forward)



**Figure 7.13** Forward inference in a feedforward neural language model. At each timestep  $t$  the network computes a  $d$ -dimensional embedding for each context word (by multiplying a one-hot vector by the embedding matrix  $\mathbf{E}$ ), and concatenates the 3 resulting embeddings to get the embedding layer  $\mathbf{e}$ . The embedding vector  $\mathbf{e}$  is multiplied by a weight matrix  $\mathbf{W}$  and then an activation function is applied element-wise to produce the hidden layer  $\mathbf{h}$ , which is then multiplied by another weight matrix  $\mathbf{U}$ . Finally, a softmax output layer predicts at each node  $i$  the probability that the next word  $w_t$  will be vocabulary word  $V_i$ .

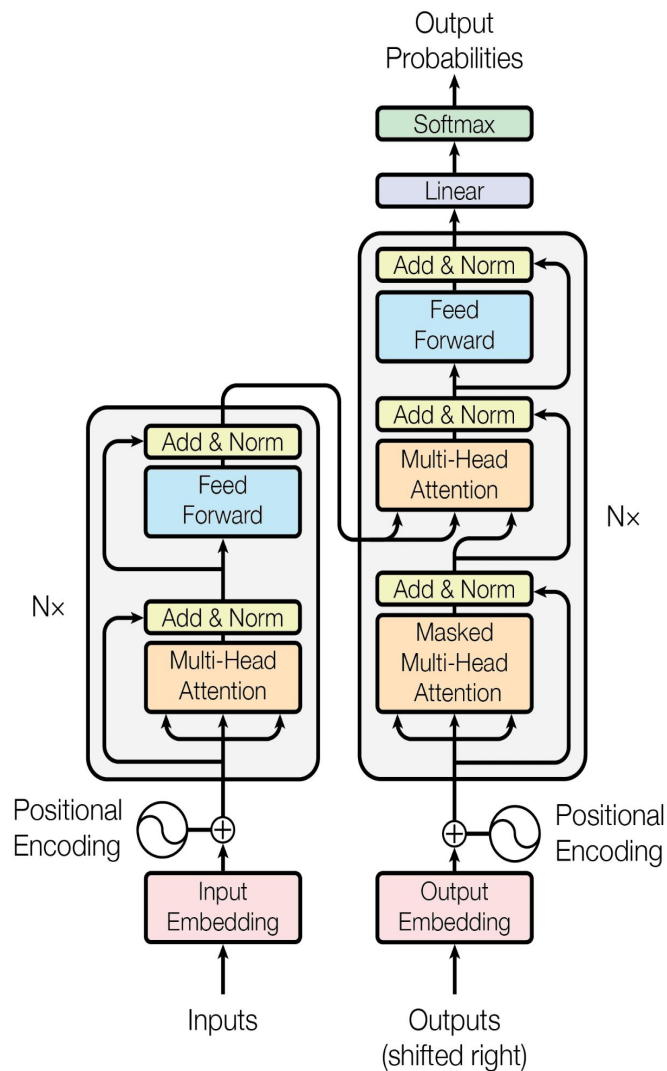
# Modelos de Lenguaje

## Modelo de lenguaje neuronal (RNN)

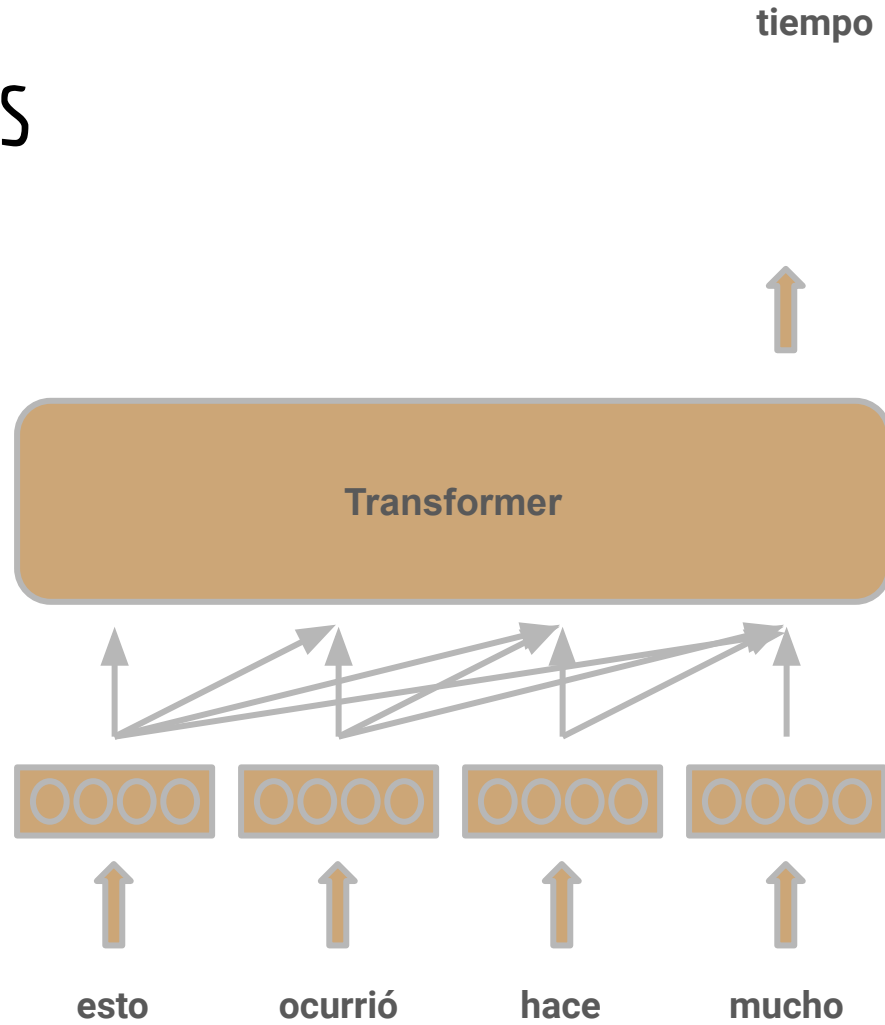


# Modelos de Lenguaje

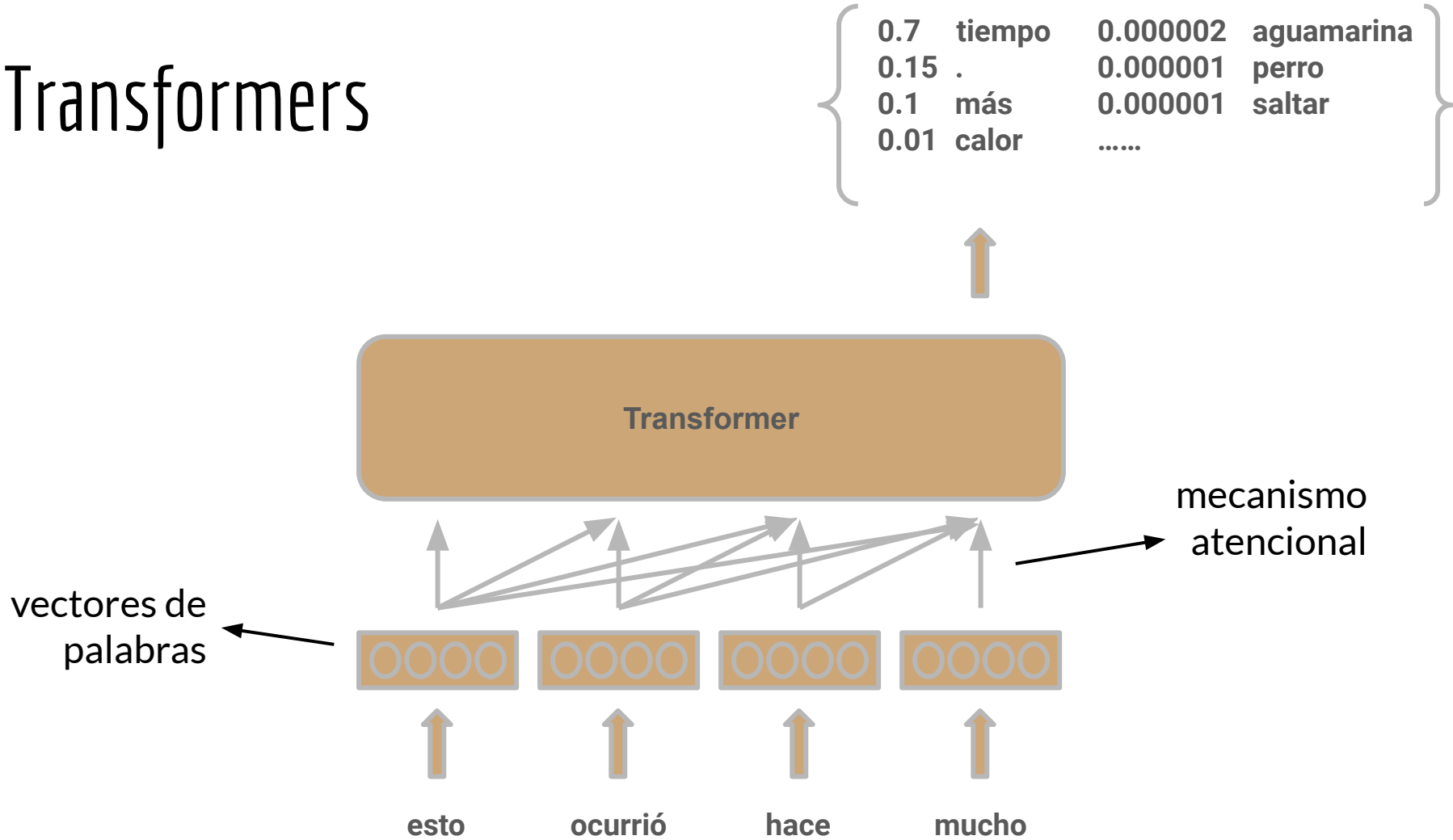
Transformers:



# Transformers

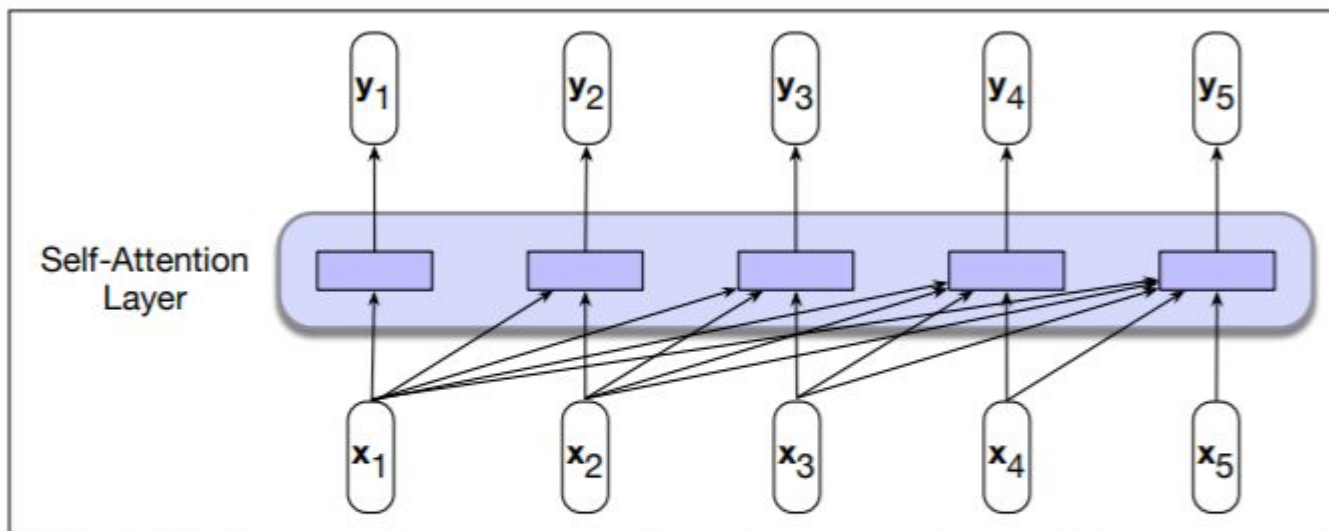


# Transformers



# Modelos de Lenguaje

## Transformers: Self-attention

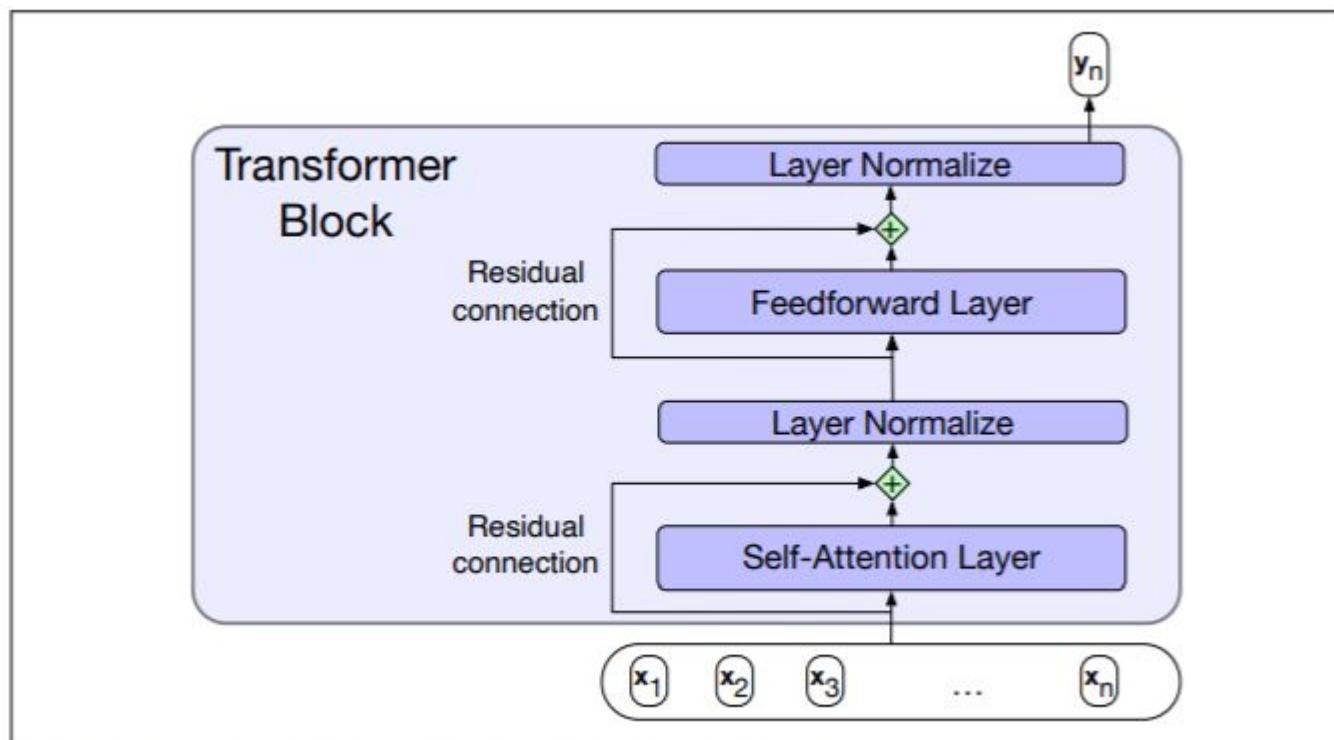


**Figure 9.15** Information flow in a causal (or masked) self-attention model. In processing each element of the sequence, the model attends to all the inputs up to, and including, the current one. Unlike RNNs, the computations at each time step are independent of all the other steps and therefore can be performed in parallel.



# Modelos de Lenguaje

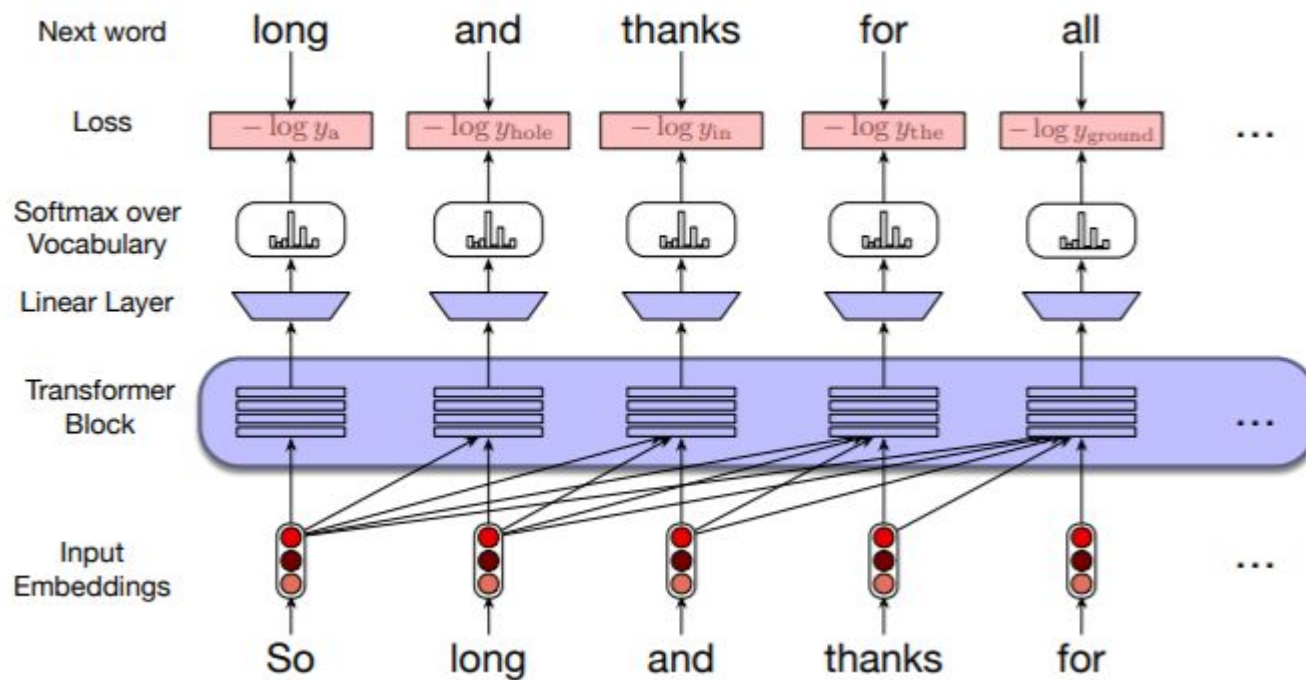
Transformers:  
Transformer block



**Figure 9.18** A transformer block showing all the layers.

# Modelos de Lenguaje

Transformers:  
Modelo de lenguaje

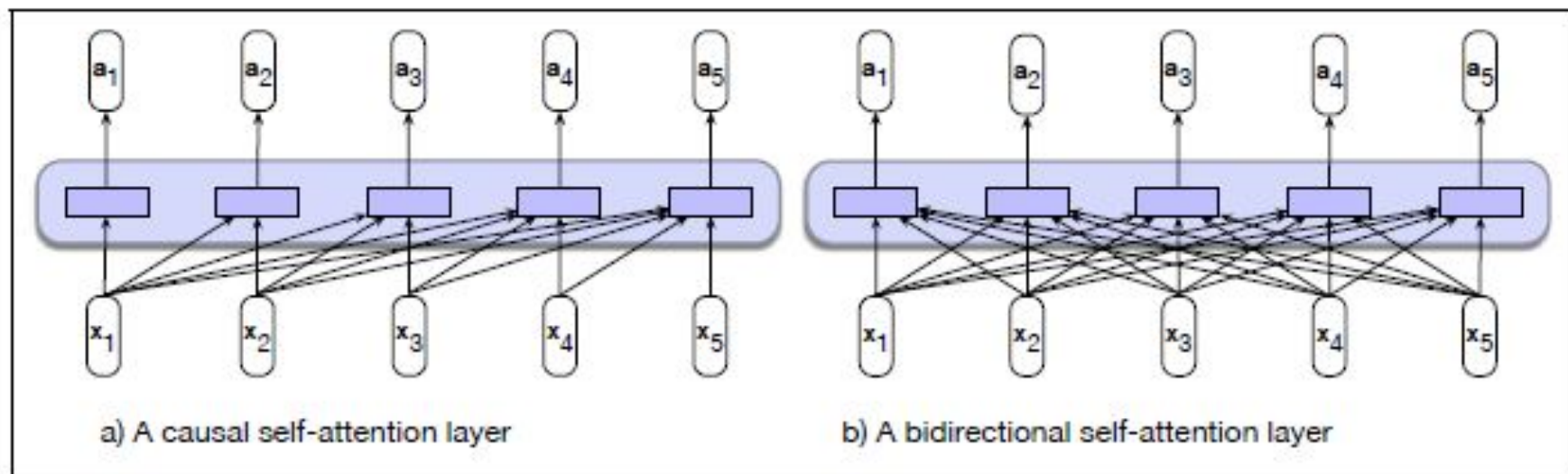


# Modelos de lenguaje

Dos grandes familias de modelos basados en transformers:

- modelos generativos (GPT) (fig. a)
- modelos bidireccionales (BERT) (fig. b)

Jurafsky & Martin 3ª ed.



**Figure 11.1** (a) The causal, backward looking, transformer model we saw in Chapter 10. Each output is computed independently of the others using only information seen earlier in the context. (b) Information flow in a bidirectional self-attention model. In processing each element of the sequence, the model attends to all inputs, both before and after the current one.

# Modelos de lenguaje

- Modelos generativos: muy usados para tareas que implican generación de lenguaje, como chatbots, traducción, resumen, etc.
- Modelos bidireccionales: Al entrenar estos modelos de lenguaje se generan representaciones internas que son de utilidad para otras tareas de PLN
  - word embeddings contextuales
  - sentence embeddings
- Se usan los modelos de lenguaje como capa de entrada y sobre ellos se hace *fine tuning* (caso particular de *transfer learning*) con nuevos datos, por ejemplo, para tareas específicas (clasificación, etc.).

# Modelos de lenguaje

## **Ejemplo de fine tuning: pysentimiento**

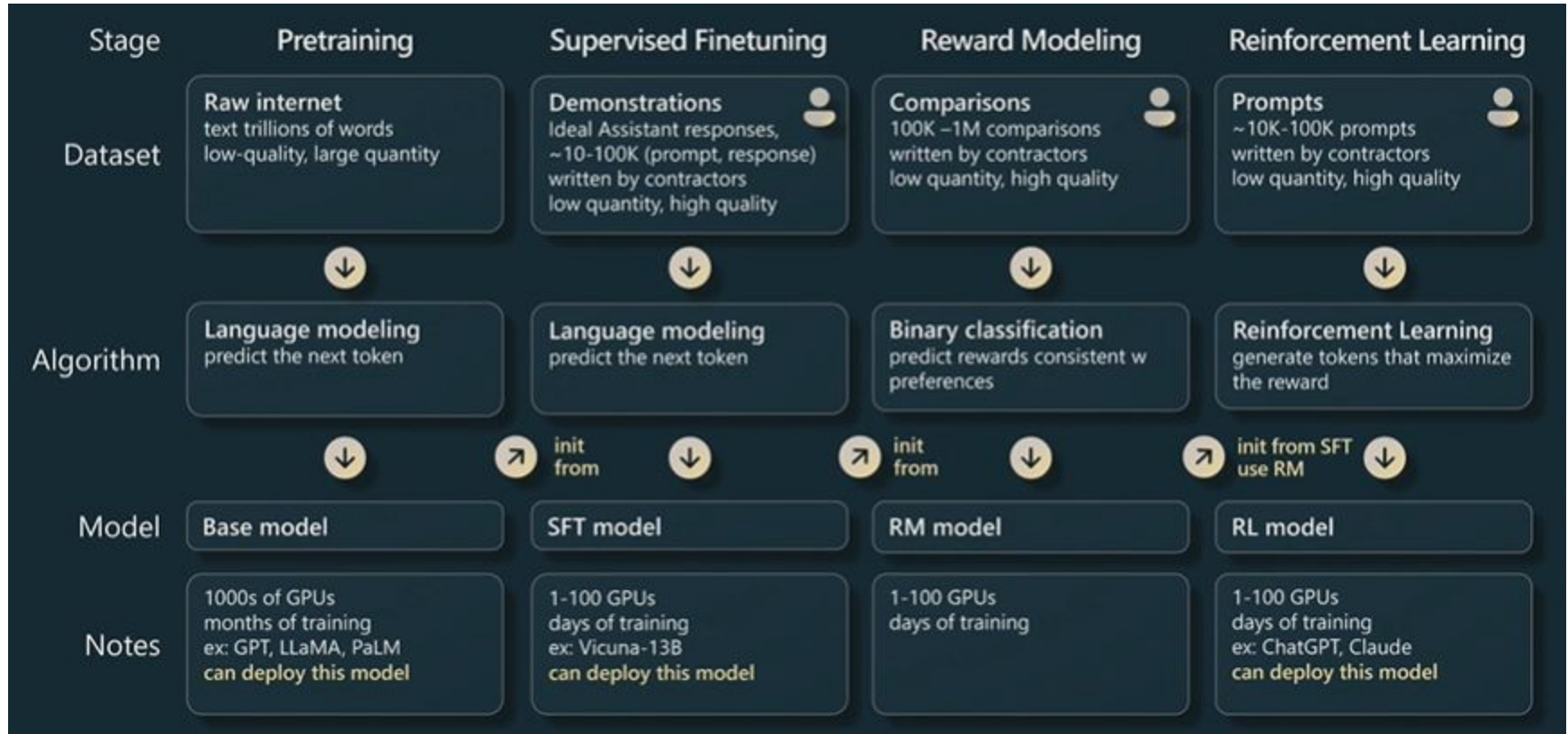
- Entrenamiento de un modelo de lenguaje basado en Roberta (basado a su vez en BERT) con un gran corpus de tweets en español: RroBERTuito (Pérez et al., 2022).
- Fine-tuning para diferentes tareas de análisis de subjetividad, como análisis de sentimiento: pysentimiento (Pérez et al, 2021).
- La mejor configuración alcanza un 70.7% de Macro F para análisis de sentimiento, sobre el dataset de TASS 2020, un valor mayor que los obtenidos en la competencia.

## **Otro ejemplo: ROUBERTa**

- Entrenamiento de un modelo de lenguaje basado en Roberta con un corpus de noticias de prensa uruguaya (Filevich et al, 2024).
- Fine-tuning para varias tareas.

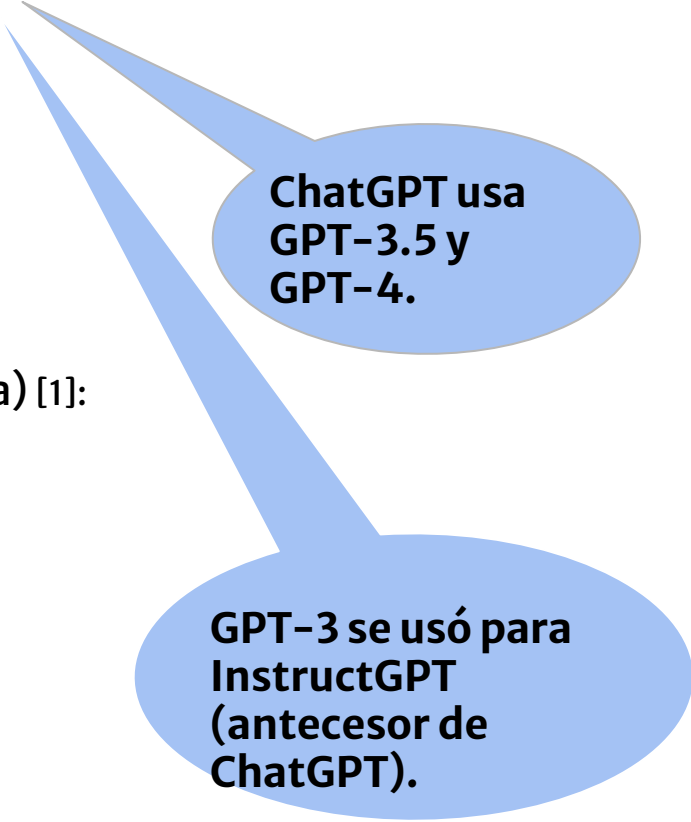
# Modelos de lenguaje usados para chatbots

# Etapas del entrenamiento de Chat GPT [1]



# 1. Pretraining: generación del modelo de lenguaje

- Consume un 99% del tiempo de entrenamiento que insume generar la herramienta completa.
- Corpus de entrenamiento GPT-3 (499 billion tokens) [3]:
  - 60% filtered CommonCrawl (410 billion tokens)
  - 22% WebText2 (19 billion tokens)
  - 16% Books1 y 2 (67 billion tokens)
  - 3% English Wikipedia (3 billion tokens)
- Corpus de entrenamiento Llama (modelo de meta) [1]:
  - 67% CommonCrawl
  - 15% C4
  - 4.5% Github
  - 4.5% Wikipedia
  - 4.5% Books
  - 2.5% ArXiv
  - 2.0% StackExchange.



**ChatGPT usa  
GPT-3.5 y  
GPT-4.**

**GPT-3 se usó para  
InstructGPT  
(antecesor de  
ChatGPT).**



# 1. Pretraining: generación del modelo de lenguaje

- Datos de entrenamiento de GPT3 [3] y LLama:

	GPT3	LLama
vocabulario	50527	32000
contexto	2048	2048
parámetros	175B	65B
entrenamiento	300B tokens (1 mes)	1-1.4 T tokens (21 días)

actualmente se están usando contextos de ¡¡100.000 tokens!!

En ese momento Llama era más potente que GPT3: la cantidad de parámetros no es tan crucial como el tiempo de entrenamiento

# 1. Pretraining: generación del modelo de lenguaje

The inputs to the Transformer are arrays of shape (B,T)

- B is the batch size (e.g. 4 here)
- T is the maximum context length (e.g. 10 here)

Training sequences are laid out as rows, delimited by special <|endof|>

Row 1: Here is an example document 1 showing some tokens.

Row 2: Example document 2 <|endof|>Example document 3 <|endof|>Example document

Row 3: This is some random text just for example <|endof|>This

Row 4: 1,2,3,4,5

GPT-3:  
batch size 3.2M  
context: 2048

tokenización  
y conversión  
a enteros

One training  
batch, array  
of shape (B,T)

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

T = 10

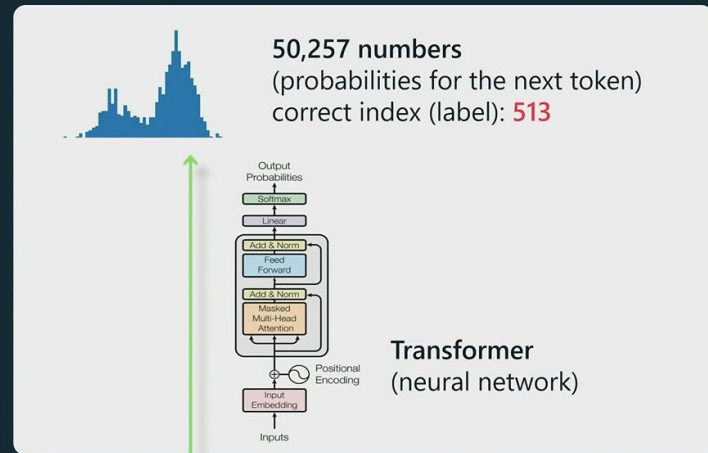
# 1. Pretraining: generación del modelo de lenguaje

Each cell only "sees" cells in its row, and only cells before it (on the left of it), to predict the next cell (on the right of it)

**Green** = a random highlighted token

**Yellow** = its context

**Red** = its target



One training  
batch, array  
of shape (B,T)

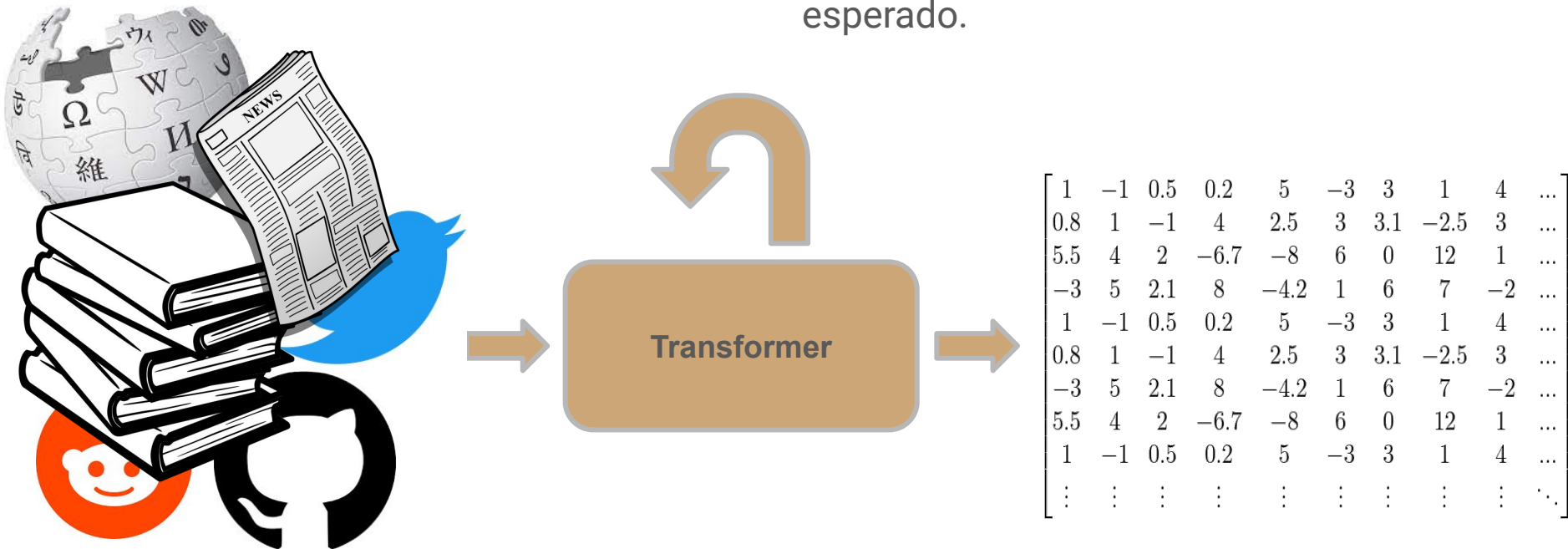
$B = 4$  ↓

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

→  $T = 10$

# 1. Pretraining: generación del modelo de lenguaje

Presentamos los ejemplos muchas veces, modificando los valores internos, hasta que lo predicho se acerque a lo esperado.



El modelo de lenguaje es el conjunto de parámetros (pesos) que se aprendió.

Recordar: el modelo ya no lo verá más todos los ejemplos que usamos.

# 2. Supervised fine-tuning (SFT): modelo para generar respuestas a prompts

- El modelo de lenguaje base podría usarse para obtener respuestas a prompts:
  - se le da como entrada un contexto, una secuencia de preguntas y respuestas, y una pregunta.
  - el modelo debe predecir la continuación, que debería ser una respuesta.

#### Context (passage and previous question/answer pairs)

Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her. "Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says. She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes, but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government, which also provides a high level of care for its people. This level of care costs money.

People in Sweden pay taxes on everything, so aren't surprised to find that owning a dog means more taxes. Some people are paying as much as 500 Swedish kronor in taxes a year for the right to keep their dog, which is spent by the government on dog hospitals and sometimes medical treatment for a dog that falls ill. However, most such treatment is expensive, so owners often decide to offer health and even life insurance for their dog.

In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

Q: How old is Catherine?

A: 54

Q: where does she live?

A:

## 2. Supervised fine-tuning (SFT): modelo para generar respuestas a prompts

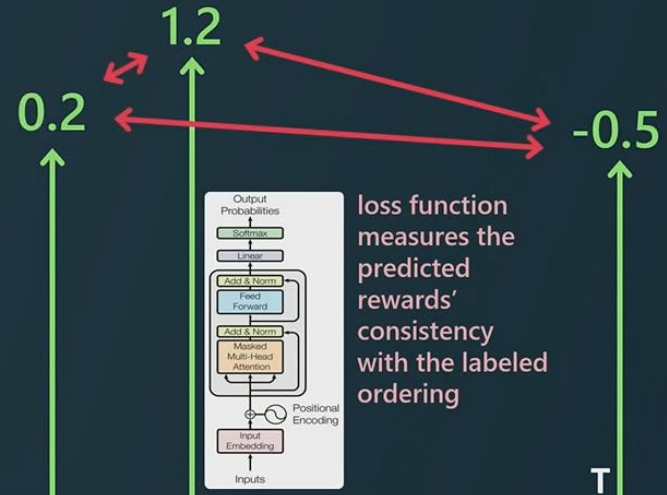
- Una mejor solución es hacer fine-tuning del modelo base usando un dataset de prompts y respuestas, generado en forma manual por anotadores.
- Tamaño del dataset: entre 10 y 100K (low quantity / high quality).
- El modelo SFT ya es un asistente (chat), pero se mejora aún más aplicando RLHF (Reinforcement Learning from Human Feedback), que se compone de un modelo de recompensas y RL.
  - Esto funciona mejor porque es más fácil ordenar salidas ya generadas que generar salidas nuevas.

# 3. Reward Model (RM): se entrena un modelo que rankea respuestas a prompts

- Se entrena un modelo que asigna un peso (recompensa) a cada respuesta que se genera, dado un prompt (Reward Model, RM).
- Nuevo dataset anotado por humanos:
  - dado un prompt,
  - se generan tres respuestas posibles con el modelo SFT,
  - los humanos las ordenan.
- El entrenamiento se hace ajustando las recompensas asignadas a cada una de las tres respuestas, según el gold standard.

# 3. Reward Model (RM): se entrena un modelo que rankea respuestas a prompts<sub>[1]</sub>

Blue are the prompt tokens, identical across rows  
Yellow are completion tokens, different in each row  
Green is the special <|reward|> token "readout"  
Only the outputs at the green cells is used, the rest are ignored



	prompt	...	...	completion 1	...	...	< reward >			
	prompt	...	...	completion 2	...	...	...	...	...	< reward >
B ↓	prompt	...	...	completion 3	...	< reward >				



# 4. Reinforcement Learning (RL): versión final del asistente

- Teniendo un modelo que asigna recompensas es posible entrenar un modelo basado en RL [2]:
  - Se ingresan prompts al modelo SFT.
  - El modelo genera una respuesta.
  - Se asigna una recompensa con el modelo RL.
  - Se ajusta el modelo SFT para maximizar las recompensas.

# 4. Reinforcement Learning (RL): versión final del asistente

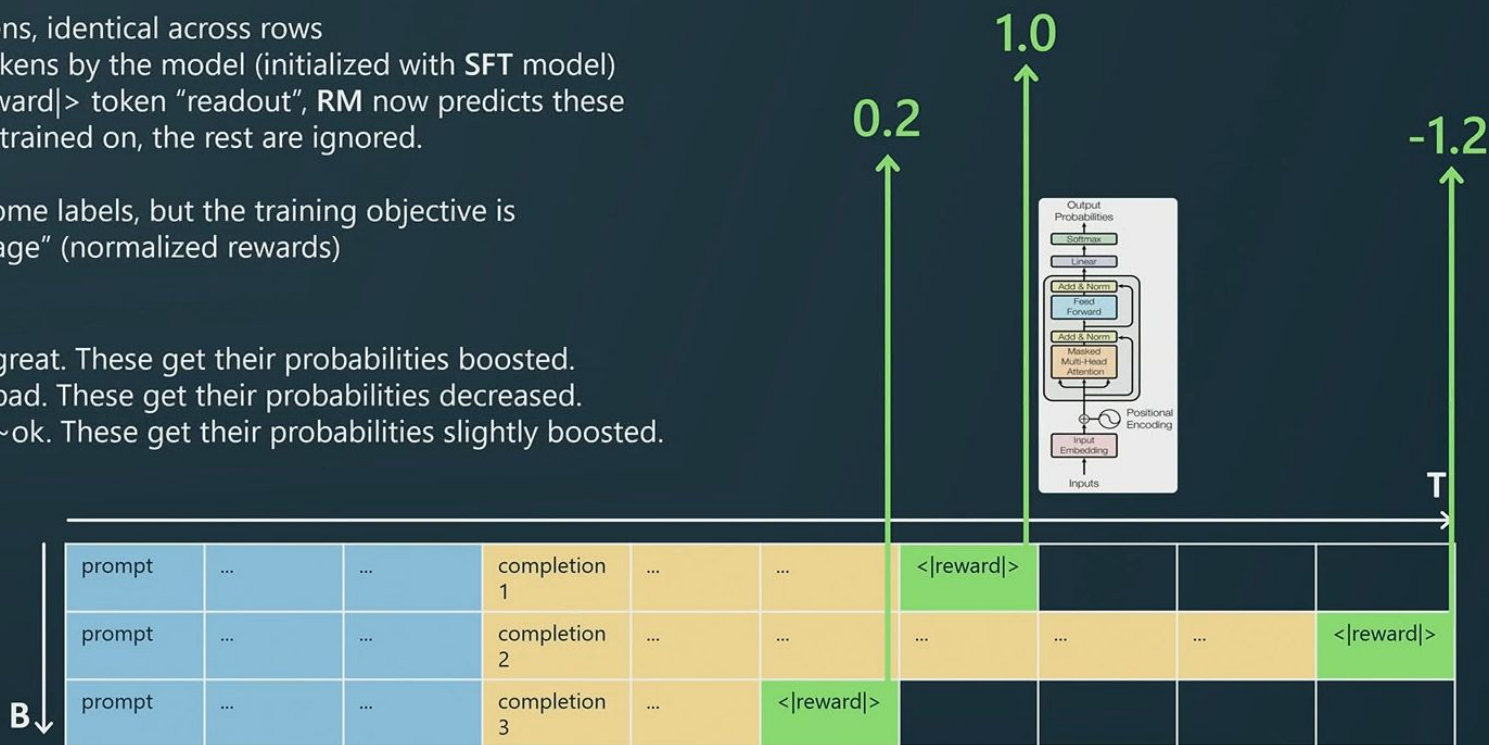
[1]

**Blue** are the prompt tokens, identical across rows  
**Yellow** are completion tokens by the model (initialized with SFT model)  
**Green** is the special `<|reward|>` token "readout", RM now predicts these  
Only the **yellow** cells are trained on, the rest are ignored.

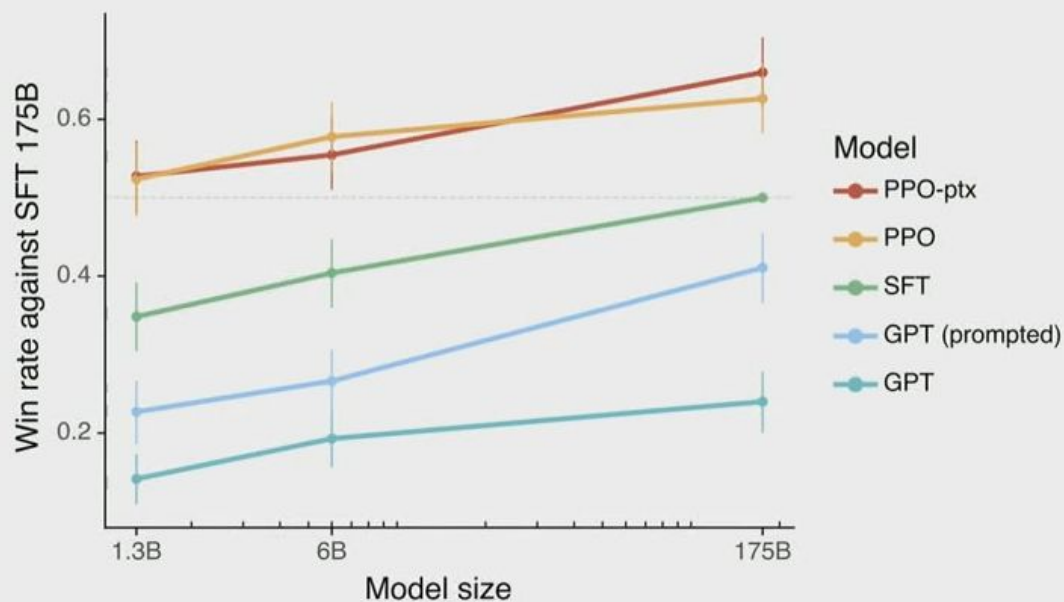
The sampled tokens become labels, but the training objective is weighted by the "advantage" (normalized rewards)

In this example:

- Row #1 tokens were great. These get their probabilities boosted.
- Row #2 tokens were bad. These get their probabilities decreased.
- Row #3 tokens were ~ok. These get their probabilities slightly boosted.



# Comparación GPT, GPT+prompt, SFT y RLHF [1,5]



[Training language models to follow instructions with human feedback, OpenAI, 2022]

# Algunas limitaciones de los LLMs base

- Problema de las “alucinaciones”.
- Tamaño finito de la ventana de contexto.
- No es posible:
  - Obtener las fuentes.
  - Acotar respuestas a un dominio específico.
  - Utilizar fuentes externas como base para las respuestas.
- Y otras más...

# Técnicas de prompting

# Técnicas de prompting

**Prompt:** Contexto que recibe el modelo de lenguaje como entrada. Texto que es tokenizado y procesado por el modelo para generar una continuación.

Las técnicas de prompting intentan controlar el lenguaje para cumplir con un formato particular.

## Primer prompt para un chatbot:

*Lo que sigue es una conversación entre un estudiante y un docente. Las respuestas del docente deben ser siempre correctas y precisas.*

*Estudiante: ¿Qué es un modelo de lenguaje?  
Docente:*

## Segunda prompt para un chatbot:

*Lo que sigue es una conversación entre un estudiante y un docente. Las respuestas del docente deben ser siempre correctas y precisas.*

*Estudiante: ¿Qué es un modelo de lenguaje?  
Docente: Un modelo de lenguaje asigna una probabilidad a una secuencia de palabras.  
Estudiante: ¿Para qué se usan?  
Docente: Los modelos de lenguaje pueden ser utilizados para...*

## Tercera prompt para un chatbot:

*Lo que sigue es una conversación entre un estudiante y un docente. Las respuestas del docente deben ser siempre correctas y precisas.*

*Estudiante: ¿Qué es un modelo de lenguaje?  
Docente: Un modelo de lenguaje asigna una probabilidad a una secuencia de palabras.  
Estudiante: ¿Para qué se usan?  
Docente: Los modelos de lenguaje pueden ser utilizados para...  
Estudiante: ¿Cuál es la arquitectura de los grandes modelos de lenguaje?  
Docente:*

# Zero-Shot, Few-Shot learning (in-context learning)

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language Models are Few-Shot Learners (arXiv:2005.14165). arXiv.

<http://arxiv.org/abs/2005.14165>

# Chain of Thought

## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (arXiv:2201.11903). arXiv. <http://arxiv.org/abs/2201.11903>



# RAG: Retrieval Augmentation Generation

- Utilizar una representación vectorial de secuencias de texto.
- Pueden ser sentence embeddings obtenidas con un modelo de lenguaje tipo BERT.
- Comparar textos en base a distancias, como la distancia coseno para encontrar los más similares.

# RAG: Retrieval Augmentation Generation

Dado un conjunto de textos  $\{t_1, t_2, \dots, t_n\}$  que se quieren utilizar como referencias para generar una respuesta a partir de una pregunta  $q$ :

- Para cada texto  $t_i$  (o fragmento de texto), obtener su representación vectorial  $v(t_i)$ .
- Obtener representación vectorial de  $q$ ,  $v(q)$ .
- Utilizando una medida de distancia (por ej. distancia del coseno), elegir los  $k$  textos  $t_i$  cuya representación  $v(t_i)$  esté más “cerca” de  $v(q)$  (k-NN).
- Agregar los  $k$  textos seleccionados en la prompt del modelo de lenguaje, indicando explícitamente que se deben utilizar estas referencias para responder a la pregunta.

# RAG: Retrieval Augmentation Generation

- **Ventajas: conozco las fuentes de información.**
- **Acierta mucho más si se trata de información fáctica, pero igual se puede equivocar:**
  - **La información puede simplemente no estar en las fuentes consultadas.**
  - **No es bueno haciendo razonamientos complejos a partir de los datos existentes.**
  - **Puede dar respuestas verdaderas pero incompletas.**

# Una aplicación de RAG

Proyecto de grado para responder preguntas sobre reglamentos de la Facultad de Ingeniería:

- Consultas de estudiantes sobre requisitos de ingreso, previaturas, calidad de libre, ...

# Una aplicación de RAG

Proyecto de grado para responder preguntas sobre reglamentos de la Facultad de Ingeniería:

- Consultas de estudiantes sobre requisitos de ingreso, previaturas, calidad de libre, ...
- Respuestas correctas / correctas pero incompletas / incorrectas / no hay respuesta.

# Una aplicación de RAG

Proyecto de grado para responder preguntas sobre reglamentos de la Facultad de Ingeniería:

- Consultas de estudiantes sobre requisitos de ingreso, previaturas, calidad de libre, ...
- Respuestas correctas / correctas pero incompletas / incorrectas / no hay respuesta.
- Conjunto de referencia: preguntas reales de estudiantes de los últimos años, respondidas por integrantes del Espacio de Orientación y Consulta.
  - Problema: las respuestas agregan conocimiento de la persona que responde.
  - ¿Cómo evaluar?

# Una aplicación de RAG

Proyecto de grado para responder preguntas sobre reglamentos de la Facultad de Ingeniería:

- Consultas de estudiantes sobre requisitos de ingreso, preiaturas, calidad de libre, ...
- Respuestas correctas / correctas pero incompletas / incorrectas / no hay respuesta.
- Conjunto de referencia: preguntas reales de estudiantes de los últimos años, respondidas por integrantes del Espacio de Orientación y Consulta.
  - Problema: las respuestas agregan conocimiento de la persona que responde.
  - ¿Cómo evaluar?
- Enfoque:
  - Siempre incluir links a documentos que generaron la respuesta.
  - Mostrar pasos de inferencia aplicados por el modelo.

# Una aplicación de RAG

Proyecto de grado para responder preguntas sobre reglamentos de la Facultad de Ingeniería:

- Consultas de estudiantes sobre requisitos de ingreso, previaturas, calidad de libre, ...
- Respuestas correctas / correctas pero incompletas / incorrectas / no hay respuesta.
- Conjunto de referencia: preguntas reales de estudiantes de los últimos años, respondidas por integrantes del Espacio de Orientación y Consulta.
  - Problema: las respuestas agregan conocimiento de la persona que responde.
  - ¿Cómo evaluar?
- Enfoque:
  - Siempre incluir link al documento que generó la respuesta.
  - Mostrar pasos de inferencia aplicados por el modelo
- Otros desafíos: recursos computacionales para entrenar/ejecutar localmente grandes modelos. ¿Optimizar? ¿Consultar modelos externos? ¿Modelos pagos?



# Referencias

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, 3rd edition draft. Stanford. 2024.

[[https://web.stanford.edu/~jurafsky/slp3/ed3bookfeb3\\_\\_2024.pdf](https://web.stanford.edu/~jurafsky/slp3/ed3bookfeb3__2024.pdf)

Acceso: junio 2024].

García Vega, M. et al.. 2020. Overview of TASS 2020: Introducing Emotion Detection . Proceedings of TASS 2020: Workshop on Semantic Analysis at SEPLN (TASS 2020), co-located with 36th SEPLN Conference (SEPLN 2020), Malaga, Spain.

Pérez, JM., Furman, D., Alemany, L., Luque, F. RoBERTuito: a pre-trained language model for social media text in Spanish. Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2022.

Pérez, JM., Giudici, JC., Luque, F. pysentimiento: A Python Toolkit for Sentiment Analysis and SocialNLP tasks, arXiv cs.CL, 2021. <https://arxiv.org/abs/2106.09462>

Filevich, J. P., Marco, G., Castro, S., Chiruzzo, L., & Rosá, A. (2024, May). A Language Model Trained on Uruguayan Spanish News Text. In Proceedings of the Second International Workshop Towards Digital Language Equality (TDLE): Focusing on Sustainability@ LREC-COLING 2024 (pp. 53-60).

Recursos sugeridos:

<http://jalamar.github.io/illustrated-transformer/>

<https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a>

# Referencias entrenamiento de ChatGPT

[1] Andrej Karpathy. State of GPT. Mayo 2023.

<https://build.microsoft.com/en-US/sessions/db3f4859-cd30-4445-a0cd-553c3304f8e2>

[2] DCC UChile. Mayo 2023. Cómo entrenar a GPT: Un repaso general de cómo se logró crear y domar a un gran modelo de lenguaje.

[https://www.youtube.com/watch?v=4jOY4i0BSSc&t=1362s&ab\\_channel=DCCUChile](https://www.youtube.com/watch?v=4jOY4i0BSSc&t=1362s&ab_channel=DCCUChile)

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei. Language Models are Few-Shot Learners. 2020.

<https://arxiv.org/abs/2005.14165>

[4] Yi Liao, Xin Jiang, and Qun Liu. 2020. Probabilistically Masked Language Model Capable of Autoregressive Generation in Arbitrary Word Order. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 263–274, Online. Association for Computational Linguistics.

[5] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L.E., Simens, M., Askell, A., Welinder, P., Christiano, P.F., Leike, J., & Lowe, R.J. (2022). Training language models to follow instructions with human feedback. ArXiv, abs/2203.02155.