



SPAIN AI  
AI WORKSHOPS

WORKSHOP GRATUITO EN ESPAÑOL

# NLP de 0 a 100 con Hugging Face

Apúntate en [eventbrite](#)



 7 SESIONES MARTES (quincenal)  
DEL 13/07 AL 5/10 (18-18:40 CET)

Imparten:

- María Grandury
- Manuel Romero
- Omar Sanseviero
- Lewis Tunstall

¡SÍGUENOS!

@NLP\_en\_ES

@Spain\_AI\_

# Enlaces útiles

---



[@nlp-en-es/nlp-de-cero-a-cien](#)



[#nlp-de-cero-a-cien](#)



[playlist: NLP de 0 a 100 con 🙌](#)



[@spain\\_ai](#)

[@nlp en es](#)



[@company/spainai](#)

[@company/nlp-en-es](#)



[spain-ai.com](#)

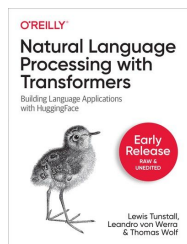
[nlp-en-es.org](#)




# Sobre Lewis Tunstall @\_lewtun



[huggingface.co/course/](https://huggingface.co/course/)



[NLP with Transformers](#)




## Hugging Face

The AI community building the future.

📍 NYC + Paris 🔗 <https://huggingface.co/> Verified


[Overview](#) [Repositories 189](#) [Packages](#) [People 100](#) [Teams 6](#) [Projects 4](#) [Sponsoring 4](#)

Pinned

 **transformers**


📖 Transformers: State-of-the-art Natural Language Processing for Pytorch, TensorFlow, and JAX.

🐍 Python ⭐ 50.3k 📄 11.9k

 **datasets**


📖 The largest hub of ready-to-use datasets for ML models with fast, easy-to-use and efficient data manipulation tools

🐍 Python ⭐ 8.8k 📄 1.1k

 **tokenizers**


📖 Fast State-of-the-Art Tokenizers optimized for Research and Production

🦀 Rust ⭐ 4.8k 📄 375

 **knockknock**


📖 Knock Knock: Get notified when your training ends with only two additional lines of code

🐍 Python ⭐ 2.2k 📄 189

 **accelerate**

📖 A simple way to train and use PyTorch models with multi-GPU, TPU, mixed-precision

🐍 Python ⭐ 1.7k 📄 86

 **huggingface\_hub**

all the open source things related to the Hugging Face Hub.

🐍 Python ⭐ 191 📄 34



# Plan de ataque

---

## Conceptos básicos

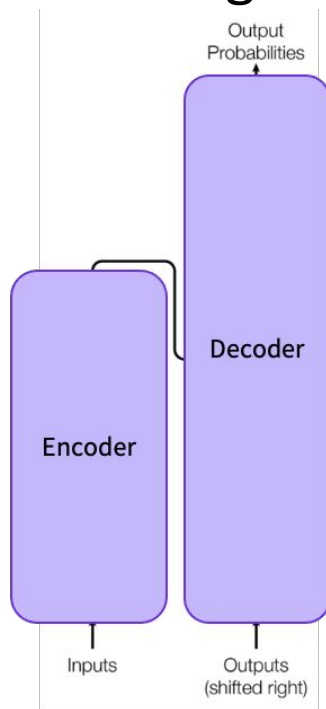
- la diferencia entre los Transformers encoders y decoders
- ejemplos populares de decoders
- diferentes estrategias de "decoding" o generación de textos

## Aplicación

- generación de textos con un GPT-2 español
- [Google Colab link](#)



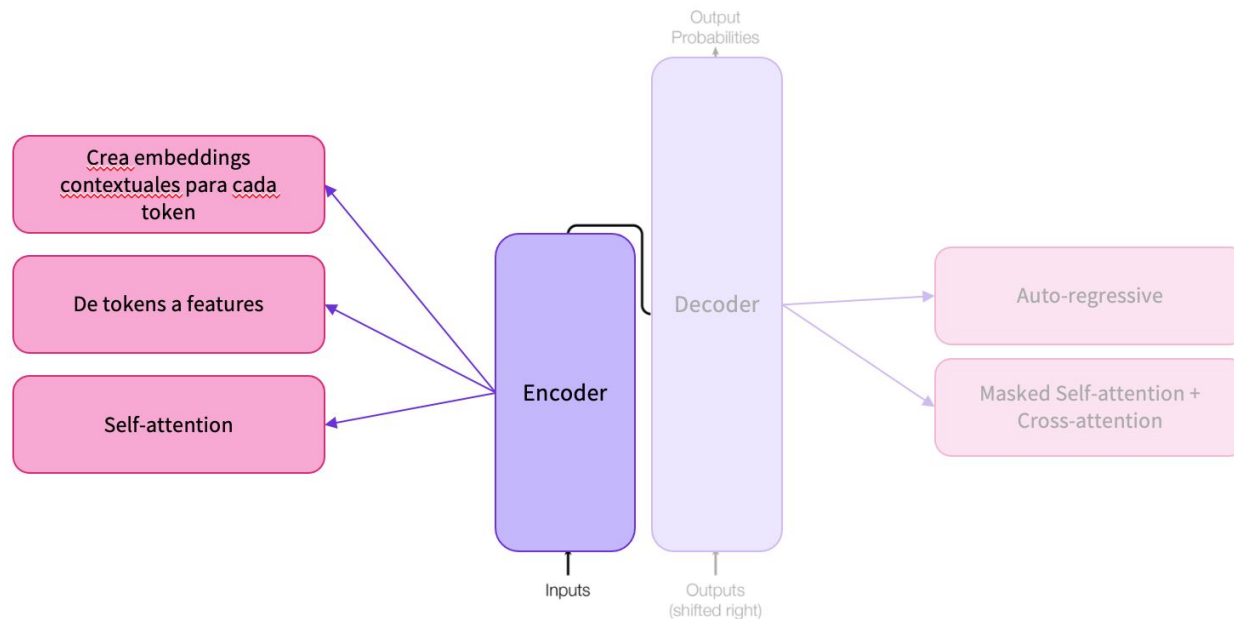
# Encoder-decoder: el transformer original, T5, M2M100, ...



[Fuente de imágenes](#)



# Encoder-based: BERT, DistilBERT, RoBERTa ...

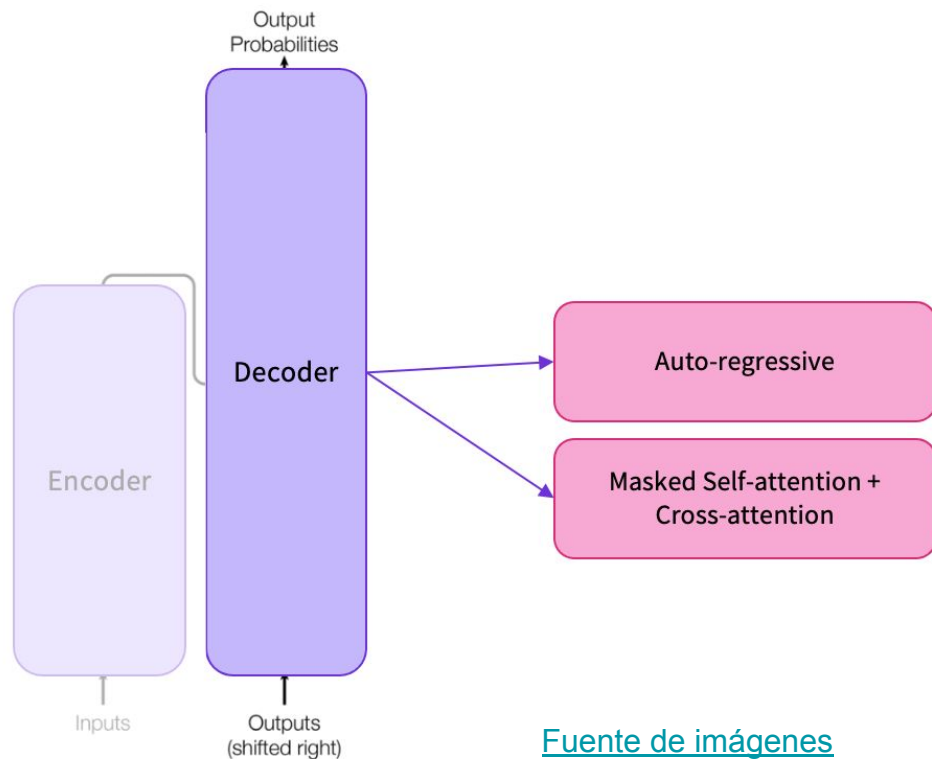


[Fuente de imágenes](#)



# Decoder-based: GPT-1/2/3, CTRL, GPT Neo, GPT-6J-B, ...

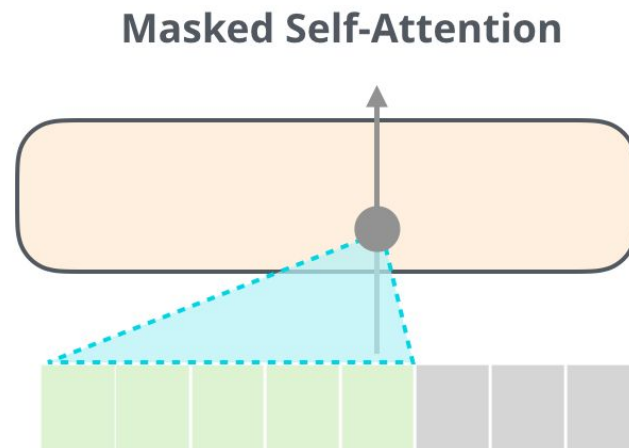
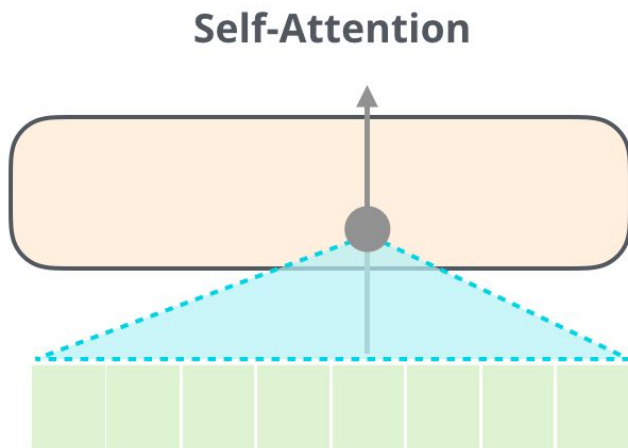
---



[Fuente de imágenes](#)



# Self-attention para encoders y decoders





# Un ejemplo famoso

---

SYSTEM PROMPT  
(HUMAN-WRITTEN)

*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*

MODEL COMPLETION  
(MACHINE-WRITTEN,  
10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

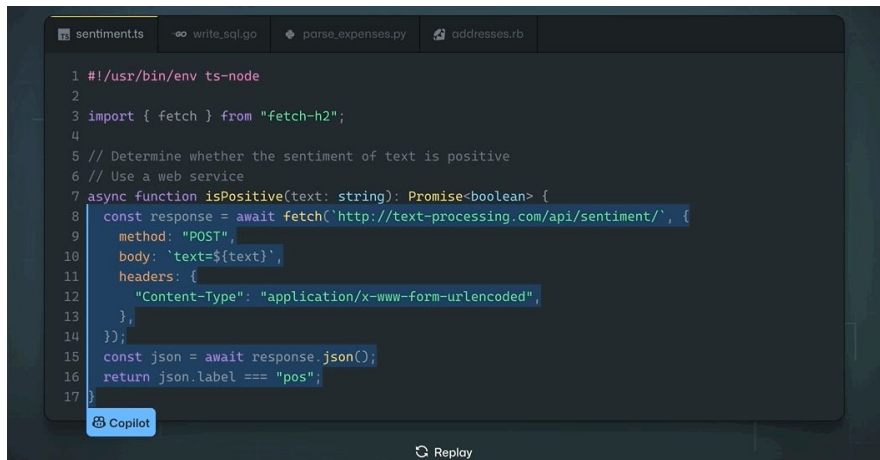
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that



# Una gran variedad de aplicaciones

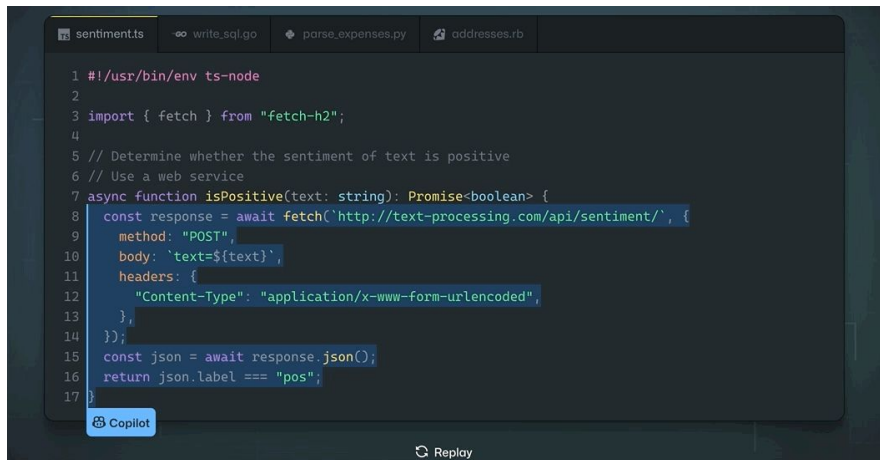


```
1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of text is positive
6 // Use a web service
7 async function isPositive(text: string): Promise<boolean> {
8   const response = await fetch("http://text-processing.com/api/sentiment/", {
9     method: "POST",
10    body: `text=${text}`,
11    headers: {
12      "Content-Type": "application/x-www-form-urlencoded",
13    },
14  });
15  const json = await response.json();
16  return json.label === "pos";
17 }
```

[GitHub Copilot](#)

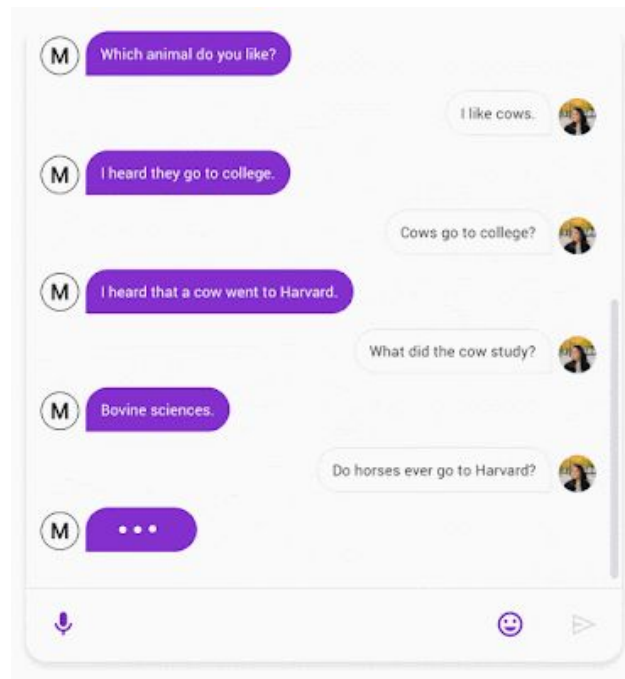


# Una gran variedad de aplicaciones



```
1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of text is positive
6 // Use a web service
7 async function isPositive(text: string): Promise<boolean> {
8   const response = await fetch("http://text-processing.com/api/sentiment/", {
9     method: "POST",
10    body: `text=${text}`,
11    headers: {
12      "Content-Type": "application/x-www-form-urlencoded",
13    },
14  });
15  const json = await response.json();
16  return json.label === "pos";
17 }
```

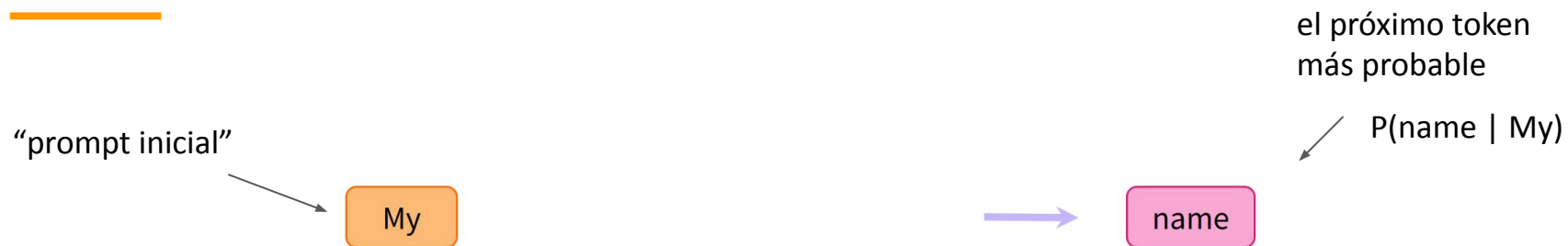
[GitHub Copilot](#)



[Google's Meena](#)

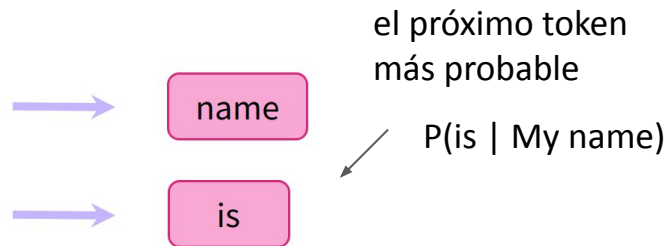


# ¿Cómo generan los decoders el texto?



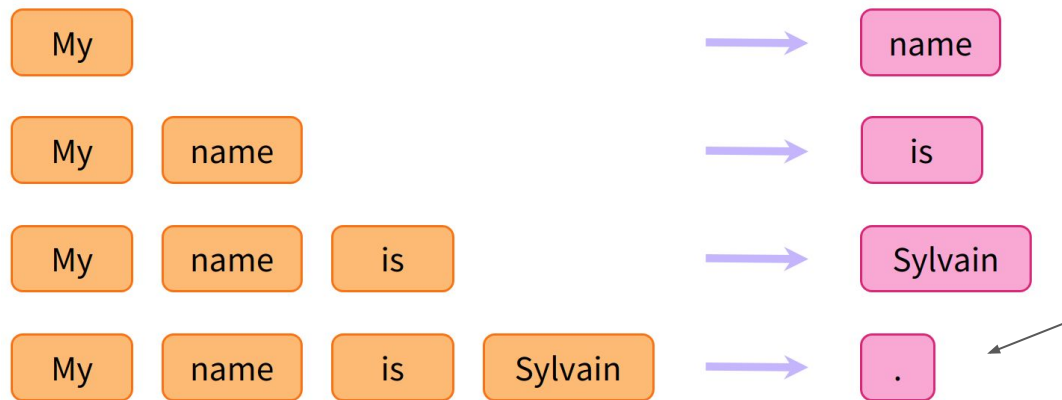
# ¿Cómo generan los decoder el texto?

combinar el token original y la predicción para producir un nuevo "prompt"



# ¿Cómo generan los decoder el texto?

---



repetir el proceso hasta que se alcance un token de fin de frase



# Estrategias de generación de textos

🤔 ¿Cómo predice el modelo el siguiente token?

distribución de probabilidad  
sobre el siguiente token  $w_i$

$$P(y_t = w_i | y_{<t}, \mathbf{x}) = \text{softmax}(z_{t,i})$$

logits del modelo de lenguaje  
para cada token



# Estrategias de generación de textos



¿Cómo predice el modelo el siguiente token?

distribución de probabilidad  
sobre el siguiente token  $w_i$

$$P(y_t = w_i | y_{<t}, \mathbf{x}) = \text{softmax}(z_{t,i})$$

logits del modelo de lenguaje  
para cada token

búsqueda de la secuencia  
global más probable

$$\hat{\mathbf{y}} = \underset{y_t}{\operatorname{argmax}} P(y_t | y_{<t}, \mathbf{x})$$

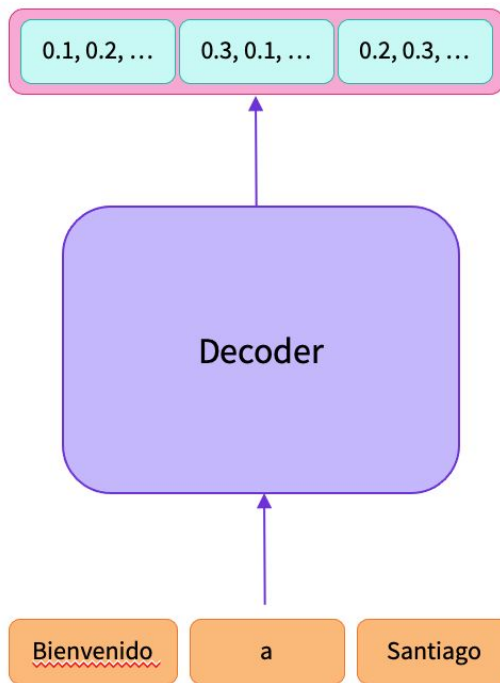
- algoritmos greedy (“greedy search decoding”)
- “beam search” decoding
- “temperatura” y sampling





# Los decoders también crean tensores de features

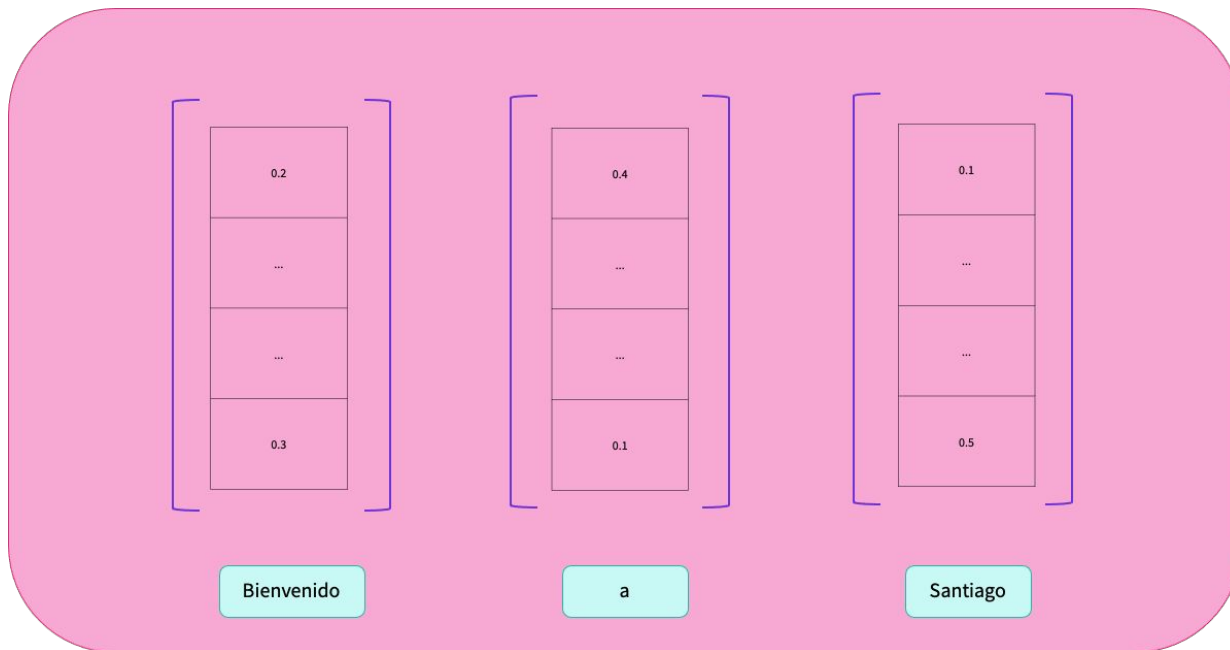
---



[Fuente de imágenes](#)



# Cada palabra se representa como un vector

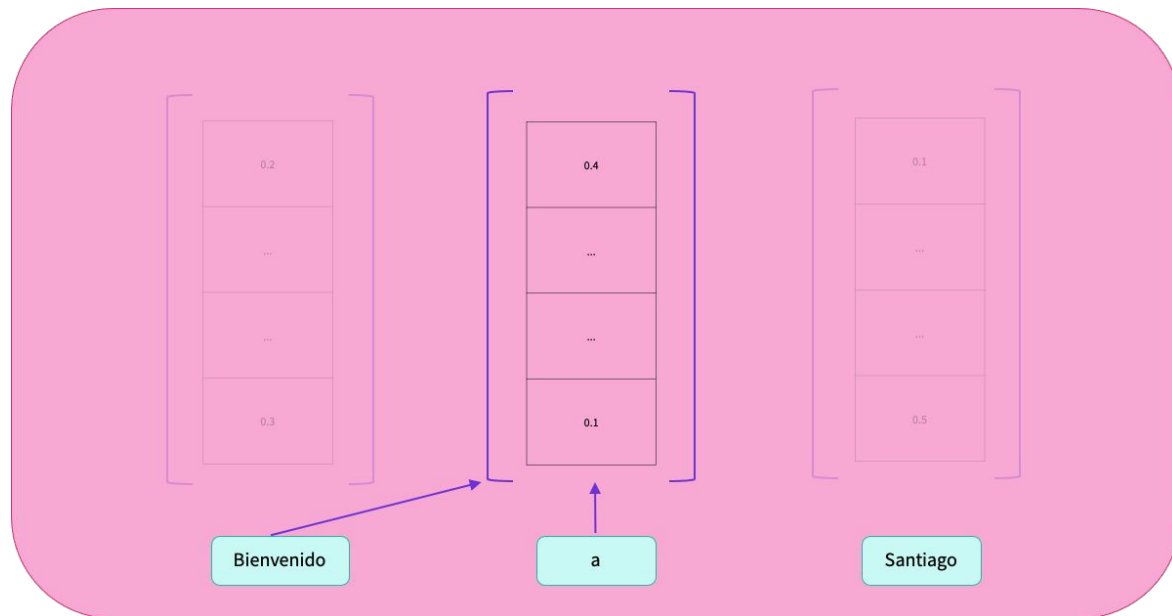


[Fuente de imágenes](#)



# Pero las palabras sólo pueden ver las de la izquierda

---



[Fuente de imágenes](#)



# Masked self-attention oculta los valores de contexto de la derecha

---



[Fuente de imágenes](#)



# ¡Hora de programar!

---



[Google Colab link](#)

@NLP\_en\_ES

