Introducción al NLP y LLM's

Introducción a la IA

2025

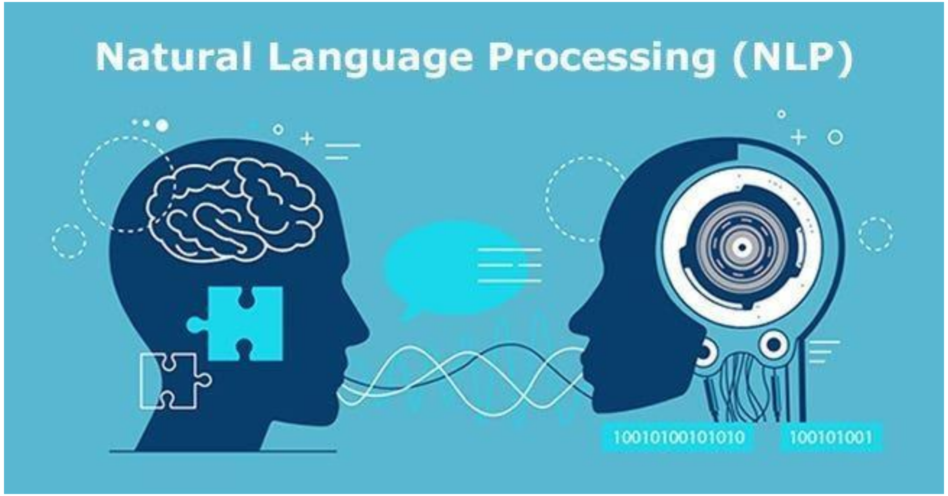
UNIVERSIDAD EAFIT®

Agenda

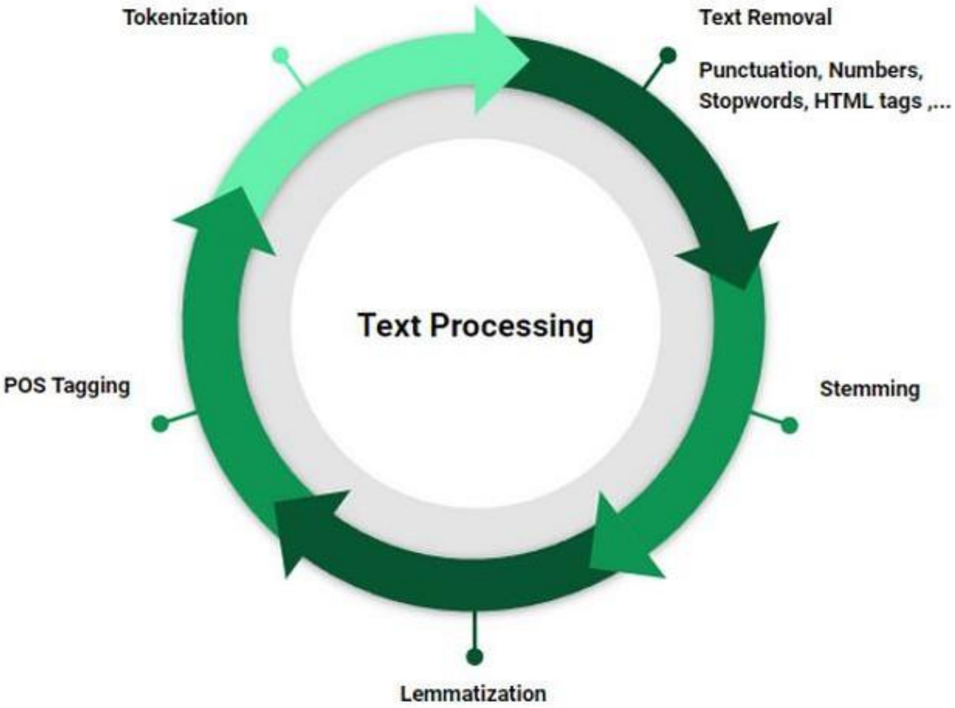
- Introducción a NLP
- Aplicaciones de NLP
- NLP con transformers
- Ejemplo práctico de Q&A usando Hugging Face.

Introducción a NLP

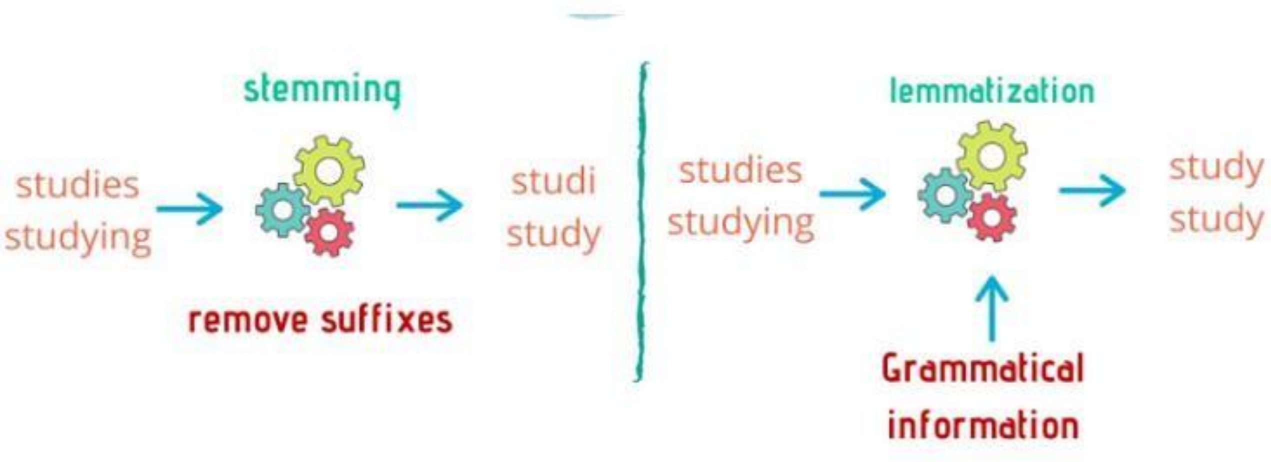
El Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) es una rama de la inteligencia artificial que se enfoca en la interacción entre computadoras y el lenguaje humano. NLP tiene como objetivo ayudar a las computadoras a entender, interpretar y generar lenguaje de una manera que sea útil para tareas como la traducción automática, el análisis de sentimientos, la generación de texto, etc.



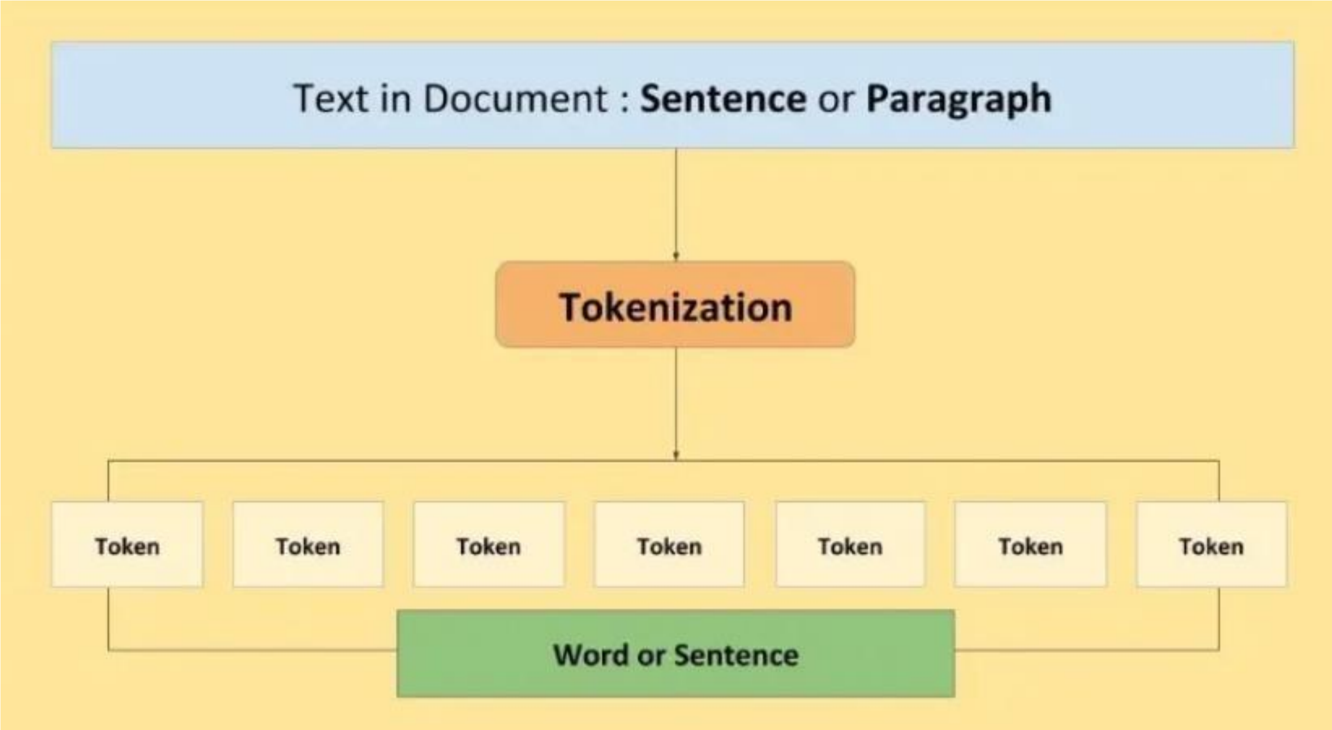
Introducción a NLP



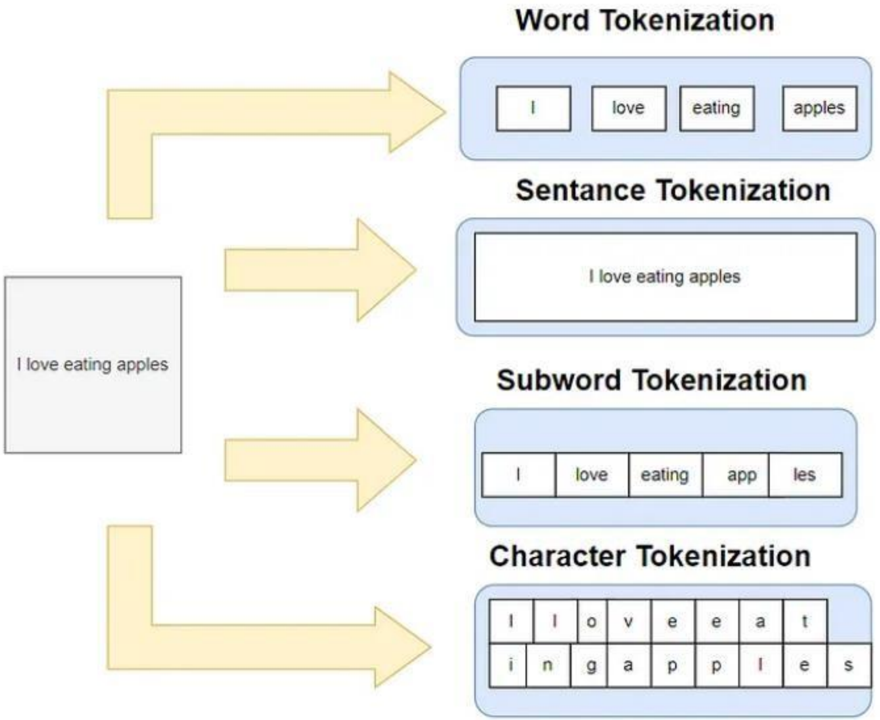
Stemming and Lemmatization



Tokenization

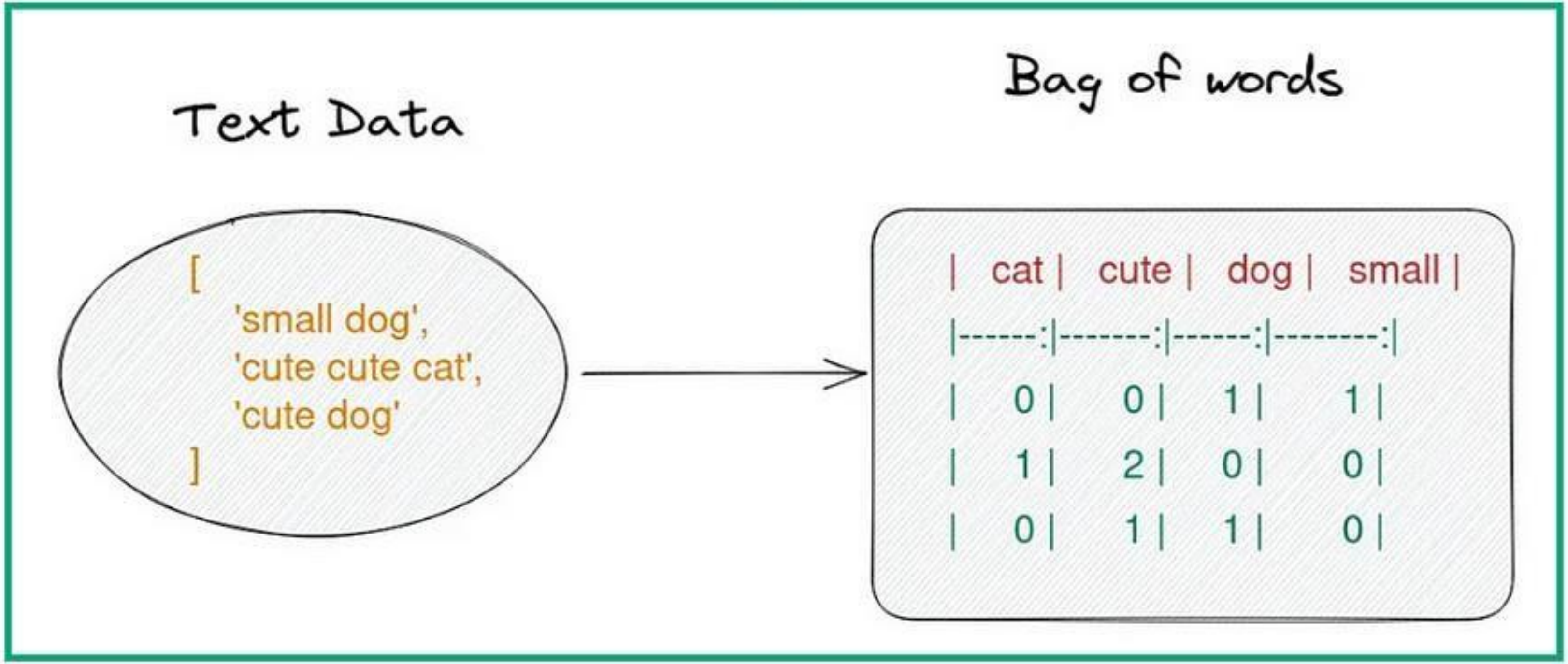


Tokenization



Tokenization techniques

Bag of words



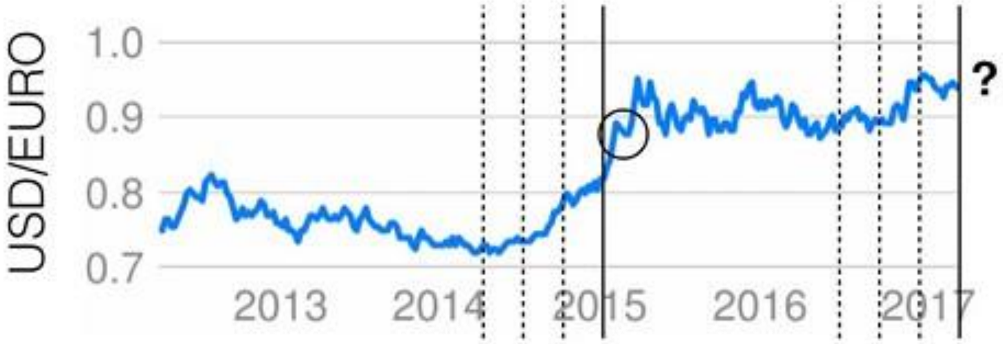
Text Data
[
'small dog',
'cute cute cat',
'cute dog'
]
Bag of words
cat	cute	dog	small
0	0	1	1
1	2	0	0
0	1	1	0

Bag of words

EJEMPLO BoW

DATOS SECUENCIALES

Cómo crear un problema de aprendizaje supervisado?



$$\underbrace{\begin{bmatrix} 0.82 \\ 0.80 \\ 0.73 \\ 0.72 \end{bmatrix}}_{\phi(t)} \qquad \underbrace{0.89}_{y^{(t)}}$$

Los datos se pueden almacenar en vectores de características y "targets" utilizando ventanas deslizantes

DATOS SECUENCIALES

Modelado de lenguaje: qué viene después?

This course has been a tremendous ...

tremendous $\begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$

a $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

?

$\phi(t)$ $\quad$ $y^{(t)}$

DATOS SECUENCIALES

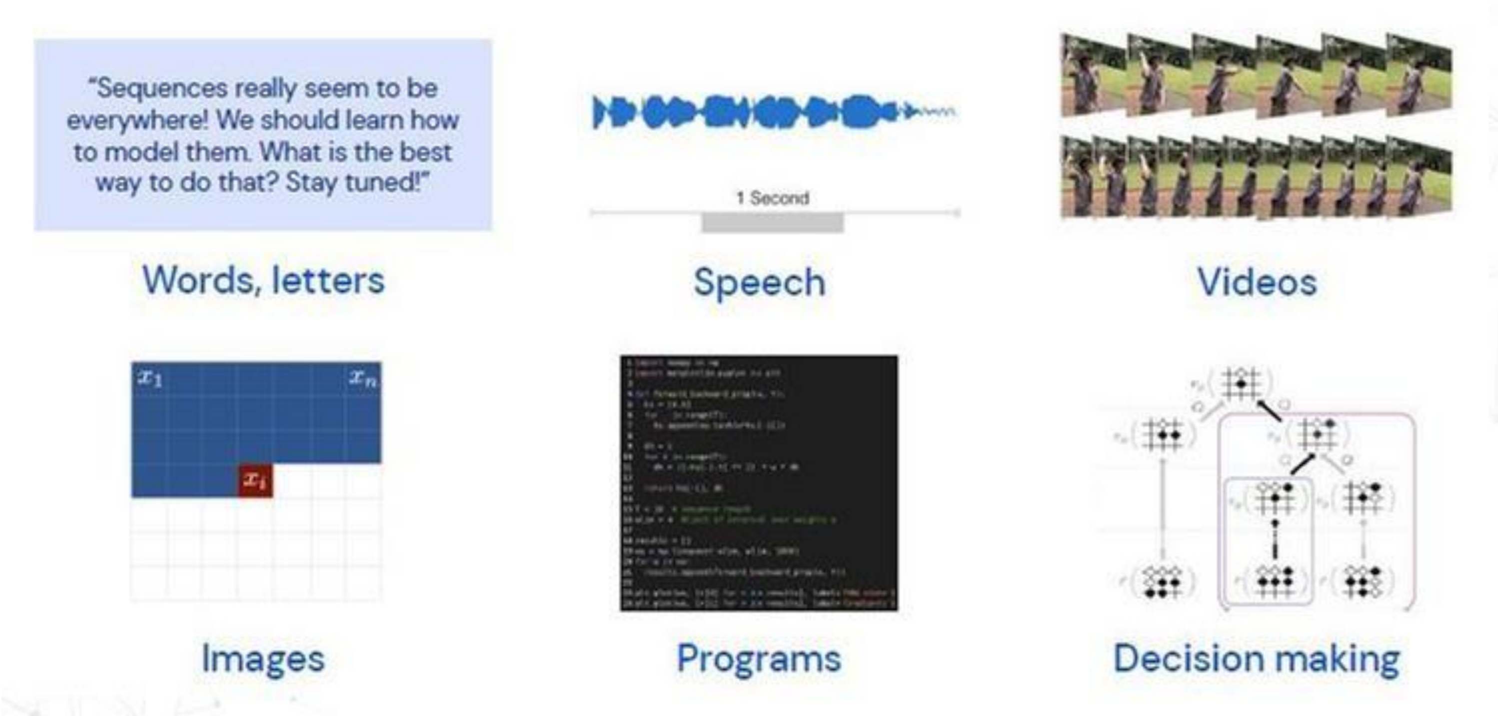
Collections of elements where:

- Elements can be **repeated**
- **Order** matters
- Of **variable** (potentially infinite) length

DATOS SECUENCIALES

- Los problemas de predicción de secuencias se pueden formular para entrenar una red neuronal feed-forward.
- Sin embargo, tenemos que ingeniar como mapear los datos históricos a un vector.
    - Cuántos pasos atrás debemos considerar?
    - Cómo mantener ítems importantes mencionados anteriormente?
- Alternativamente, nos gustaría aprender como codificar la “historia” de la secuencia en un vector.

DATOS SECUENCIALES



DATOS SECUENCIALES

	Supervised learning	Sequence modelling
Data	$\{x, y\}_i$	$\{x\}_i$
Model	$y \approx f_\theta(x)$	$p(x) \approx f_\theta(x)$
Loss	$\mathcal{L}(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$	$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(f_\theta(x_i))$
Optimisation	$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$	$\theta^* = \arg\max_\theta \mathcal{L}(\theta)$

DATOS SECUENCIALES

"Modeling word probabilities is really difficult"

DATOS SECUENCIALES

Simplest model:
Assume independence of words

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t)$$

p("modeling") × p("word") × p("probabilities") × p("is") × p("really") × p("difficult")

Word	$p(x_i)$
the	0.049
be	0.028
...	...
really	0.0005
...	...

However:

Most likely 6-word sentence:

"The the the the the the."

→ Independence assumption does not match sequential structure of language.

DATOS SECUENCIALES

More realistic model:
Assume conditional dependence of words

$$p(x_T) = p(x_T | x_1, \ldots, x_{T-1})$$

Modeling word probabilities is really ?

Context — Target

Target	p(x|context)
difficult	0.01
hard	0.009
fun	0.005
...	...
easy	0.00001

DATOS SECUENCIALES

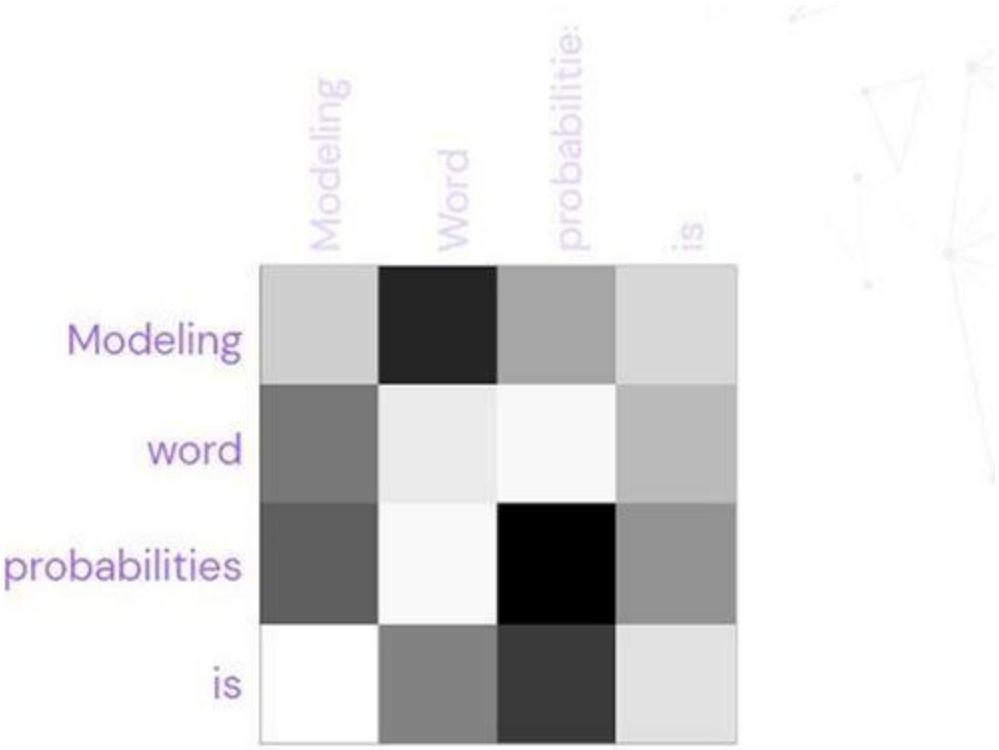
The chain rule
Computing the joint p(**x**) from conditionals

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

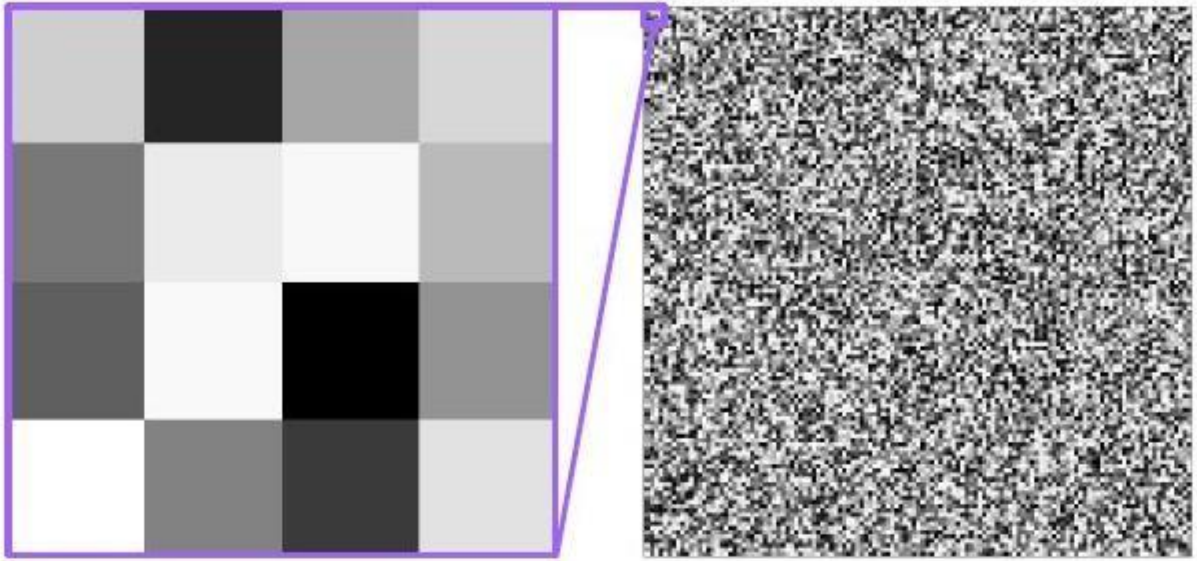
Modeling	$p(x_1)$
Modeling **word**	$p(x_2 \| x_1)$
Modeling word **probabilities**	$p(x_3 \| x_2, x_1)$
Modeling word probabilities **is**	$p(x_4 \| x_3, x_2, x_1)$
Modeling word probabilities is **really**	$p(x_5 \| x_4, x_3, x_2, x_1)$
Modeling word probabilities is really **difficult**	$p(x_6 \| x_5, x_4, x_3, x_2, x_1)$

PROBLEMA DE ESCALAMIENTO

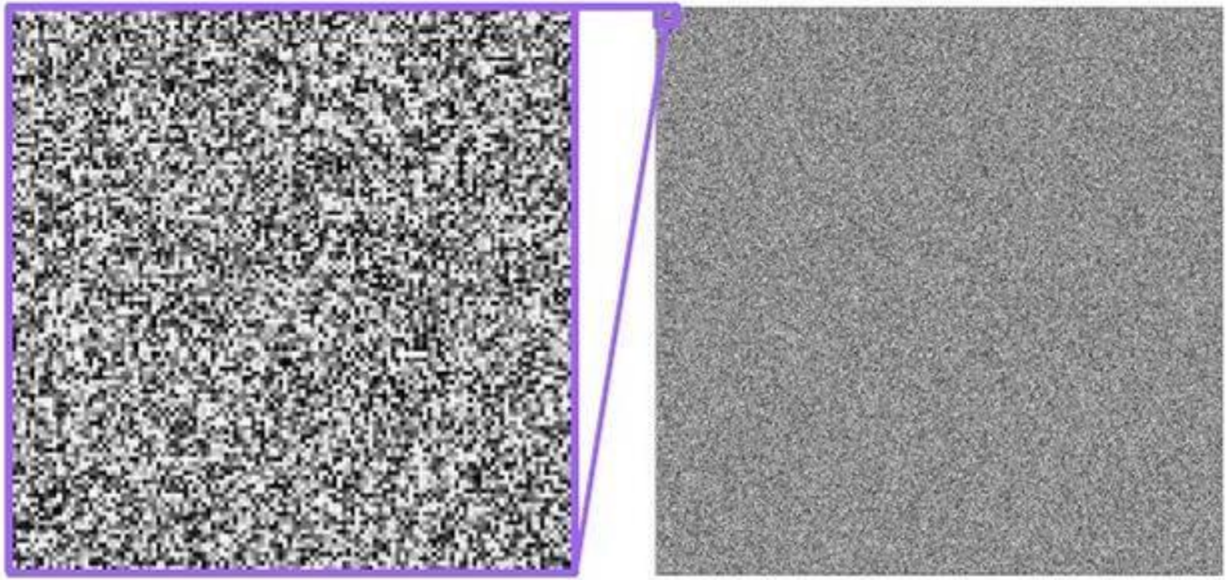
$p(x_2|x_1)$



PROBLEMA DE ESCALAMIENTO



Escalamiento



These images are only for context of size N=1!

The table size of larger contexts will grow with **vocabulary**^{N}

N-gramas

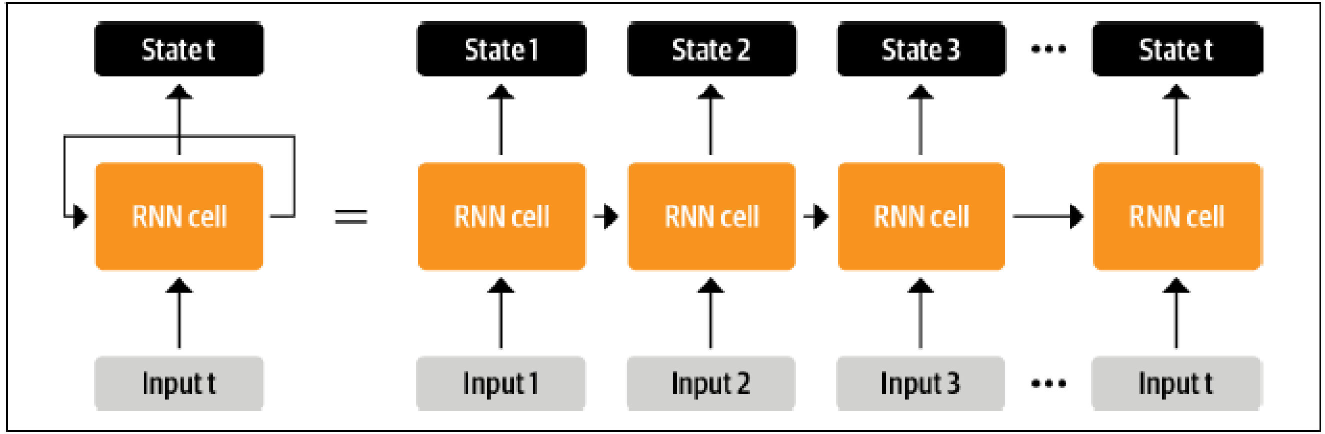
Only condition on N previous words

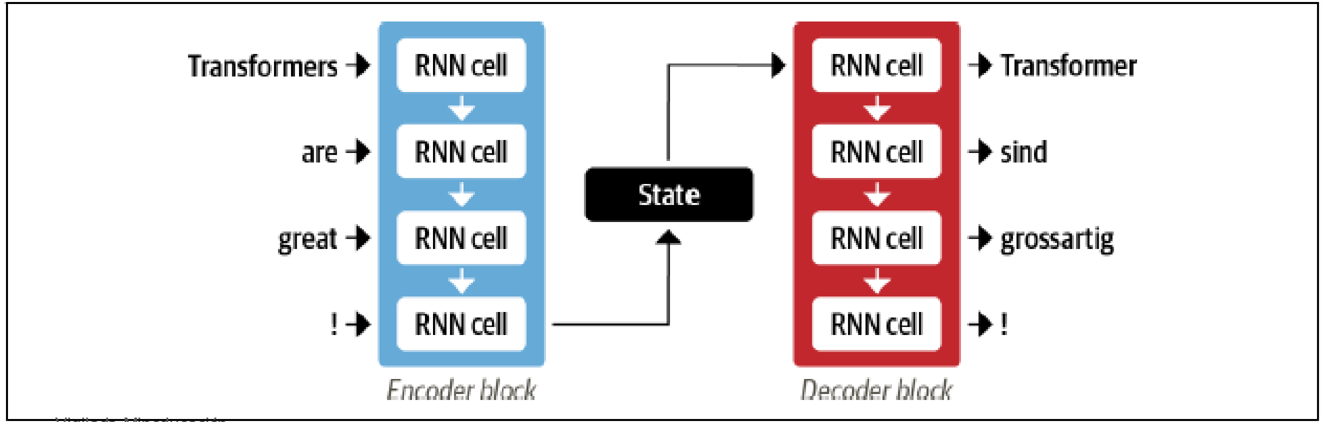
$$p(\mathbf{x}) \approx \prod_{t=1}^{T} p(x_t | x_{t-N-1}, \ldots, x_{t-1})$$

Modeling
Modeling **word**
Modeling word **probabilities**
word probabilities **is**
probabilities is **really**
is really **difficult**

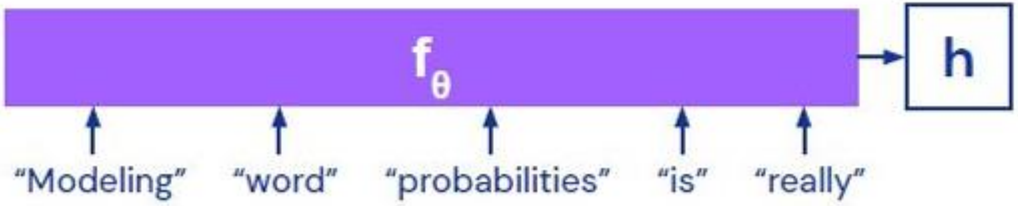
$$p(x_1)$$
$$p(x_2 | x_1)$$
$$p(x_3 | x_2, x_1)$$
$$p(x_4 | x_3, x_2)$$
$$p(x_5 | x_4, x_3)$$
$$p(x_6 | x_5, x_4)$$

RNN


State t
RNN cell
Input t
=
State 1
State 2
State 3
•••
State t
RNN cell
RNN cell
RNN cell
RNN cell
Input 1
Input 2
Input 3
•••
Input t


Transformers
are
great
!
RNN cell
RNN cell
RNN cell
RNN cell
Encoder block
State
RNN cell
RNN cell
RNN cell
RNN cell
Decoder block
Transformer
sind
grossartig
!

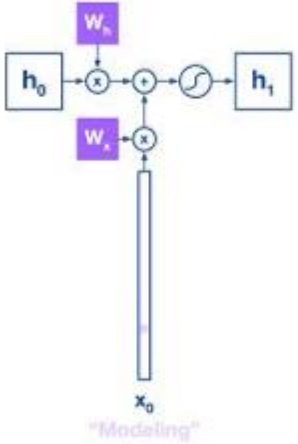
RNN



f_θ summarises the context in $\boxed{h}$ such that:

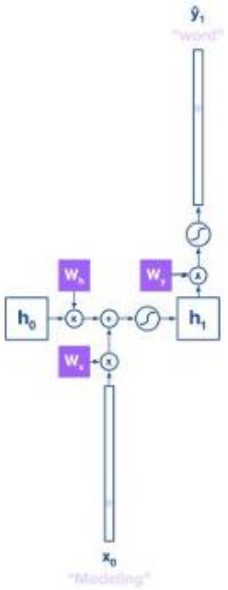
$$p(x_t|x_1, \ldots, x_{t-1}) \approx p(x_t|h)$$

RNN



$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t)$$

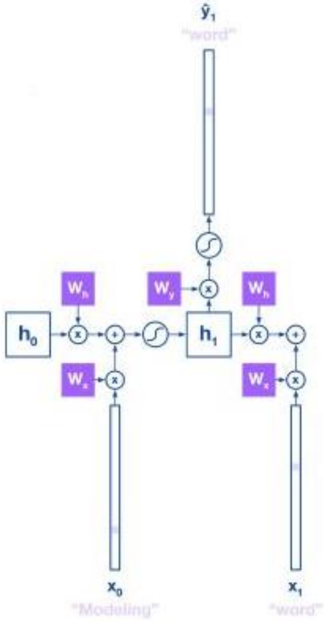
RNN



RNNs predict the target **y** (the next word) from the state **h.**

$$p(\mathbf{y_{t+1}}) = softmax(\mathbf{W}_y \mathbf{h}_t)$$

Softmax ensures we obtain a distribution over all possible words.



Input next word in sentence x_1

Hello Transformers

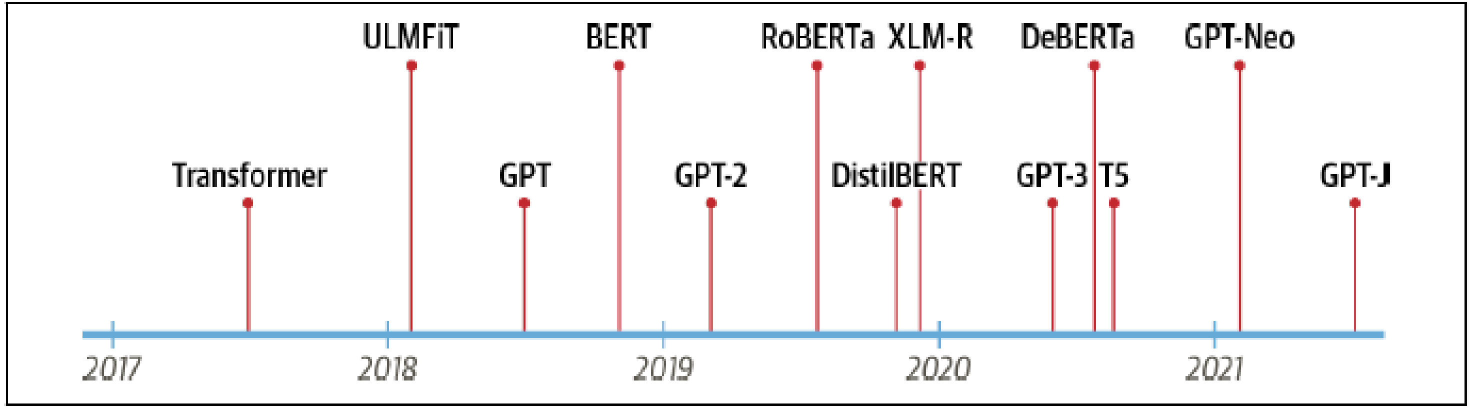
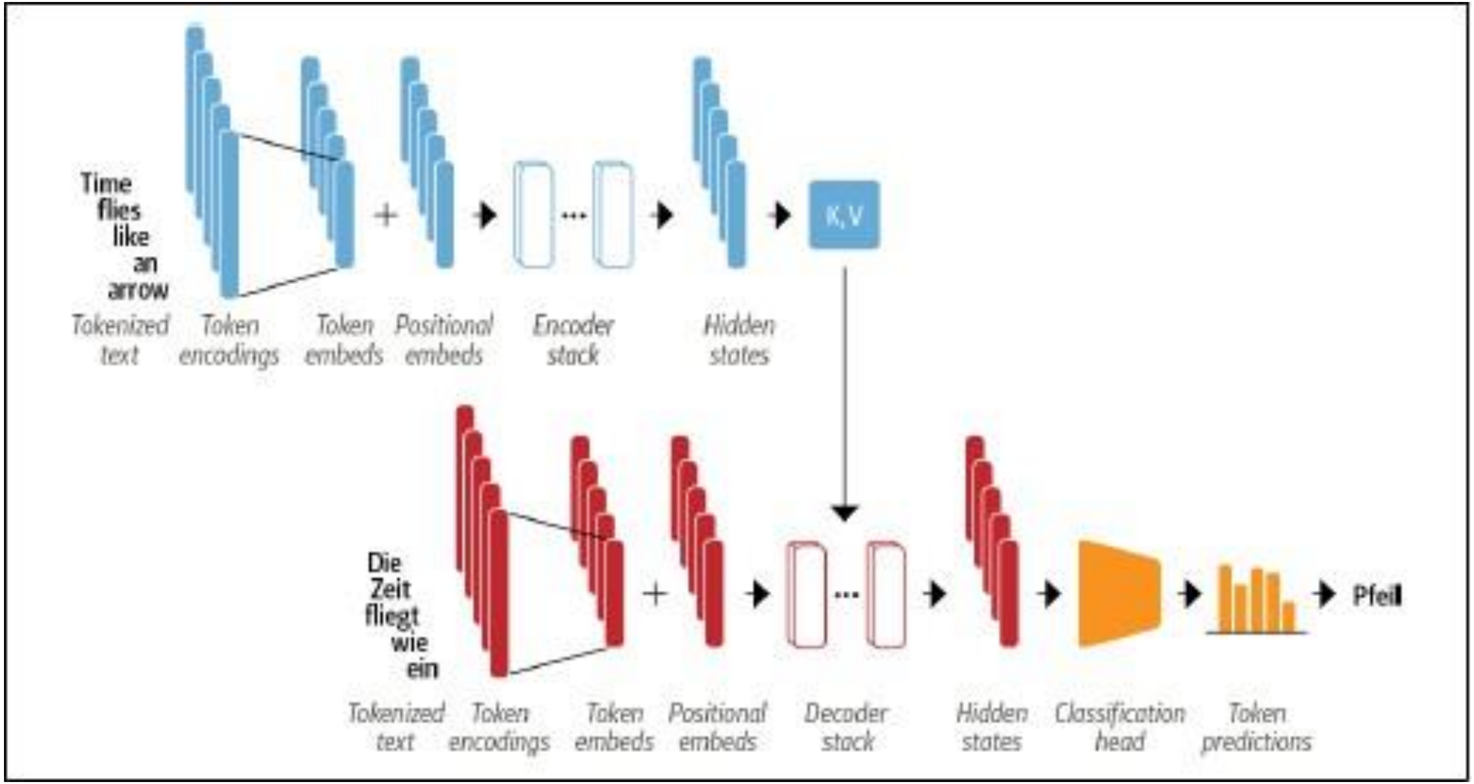
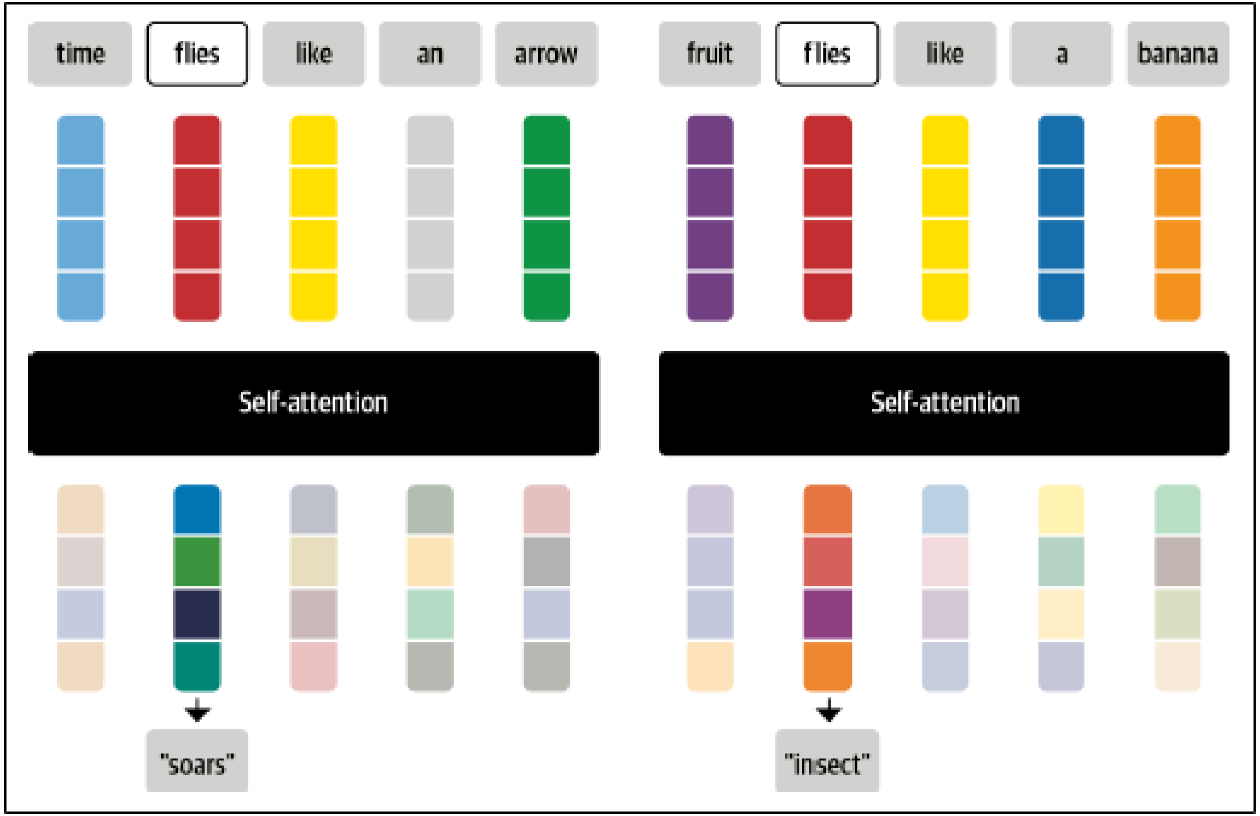


Figure 1-1. The transformers timeline

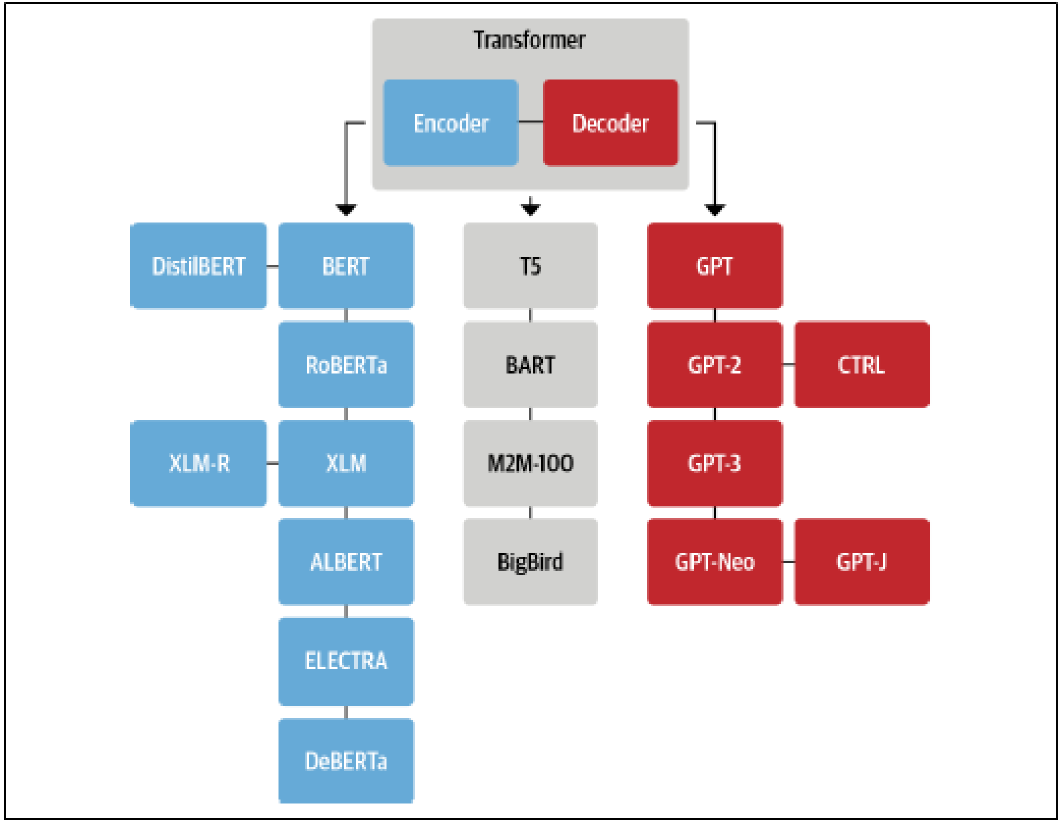
Transformer Arquitecture



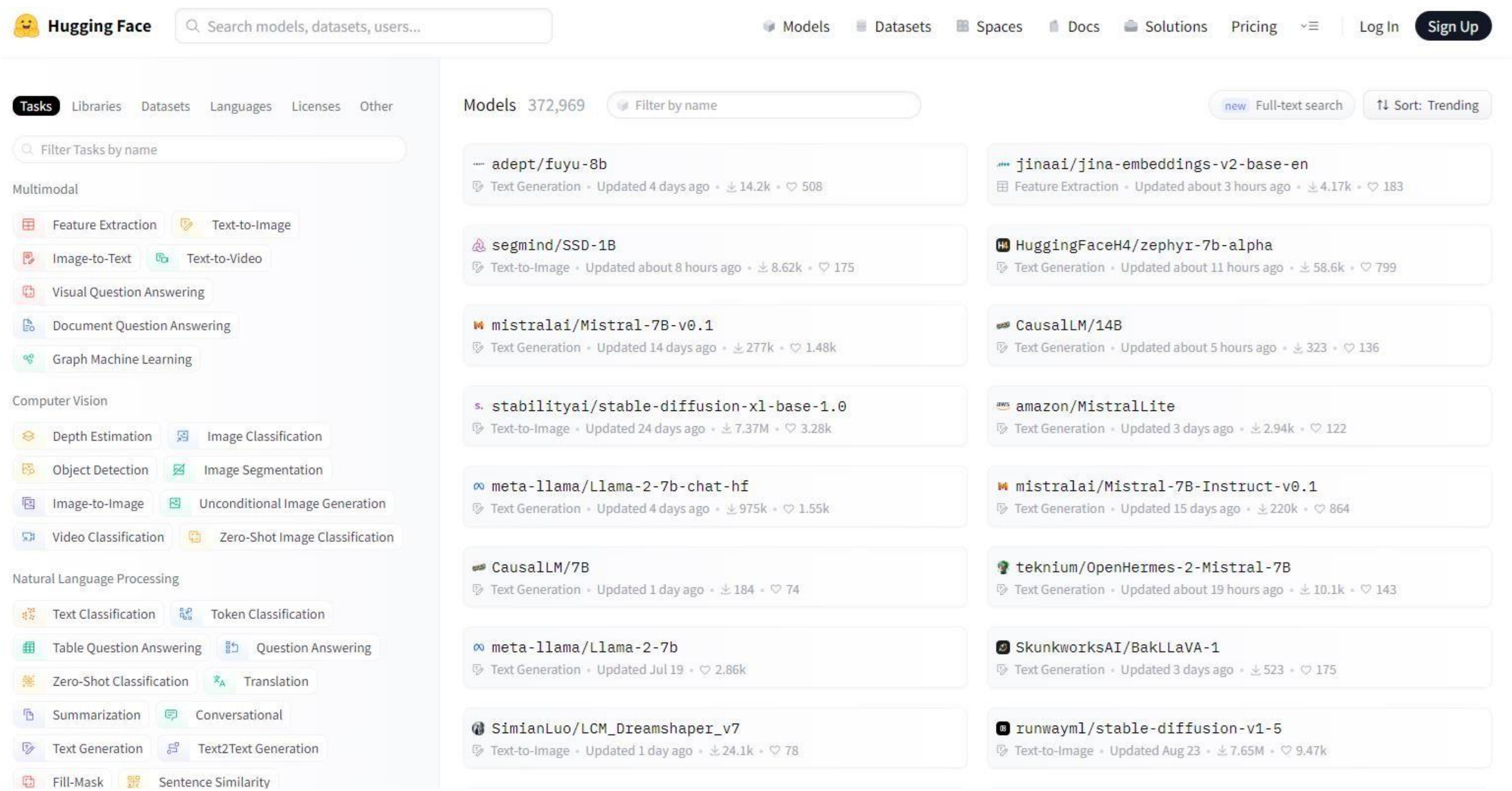
Self-Attention



Algunas Arquitecturas de Transformers

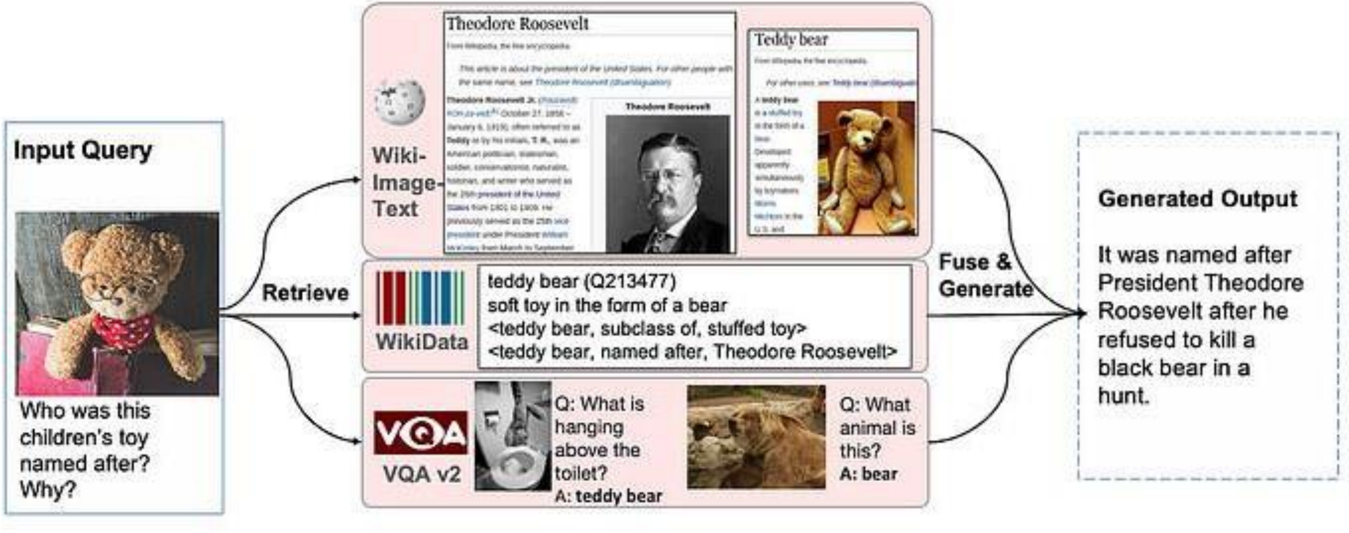


Hugging Face Hub

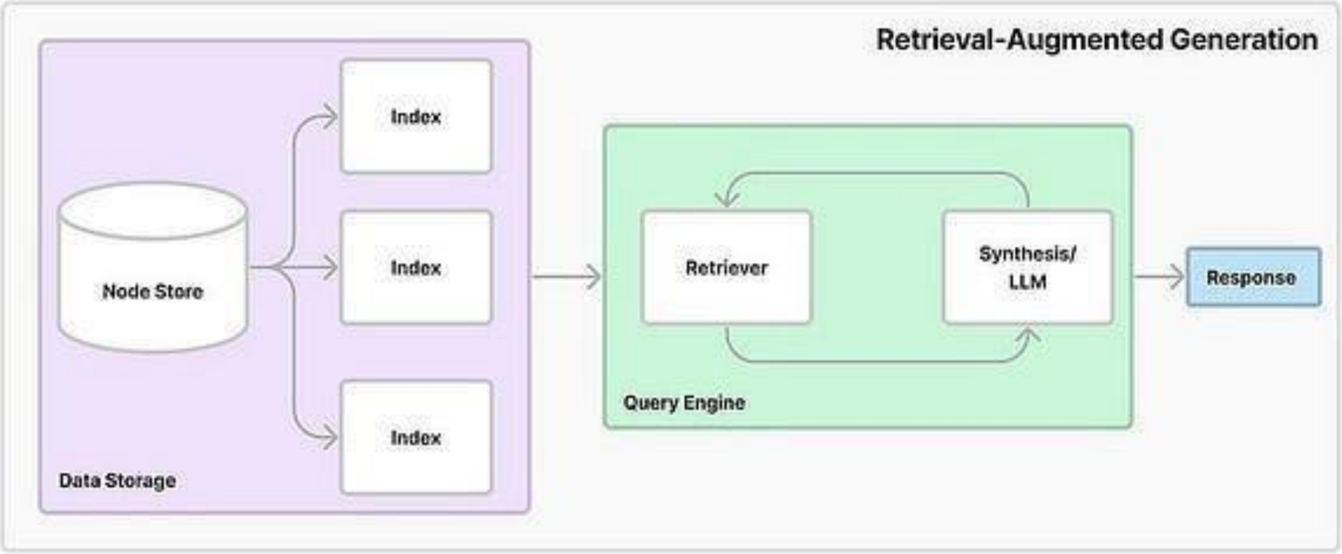


RAG

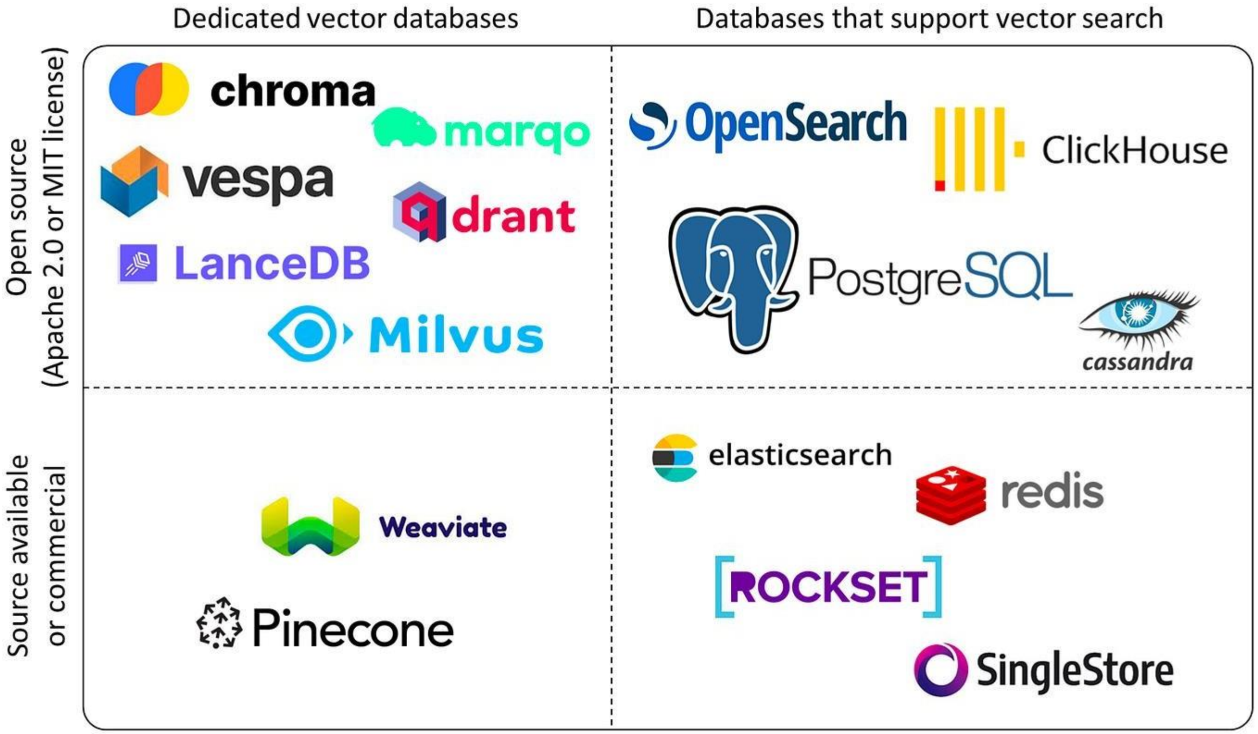
Retrieval-Augmented Generation (RAG)
Retrieval-augmented generation (RAG) is a natural language processing (NLP) approach that combines the benefits of both retrieval-based and generation-based methods for content generation tasks. It aims to improve the quality and controllability of the generation tasks by leveraging a pre-trained language model in conjunction with a retrieval mechanism.



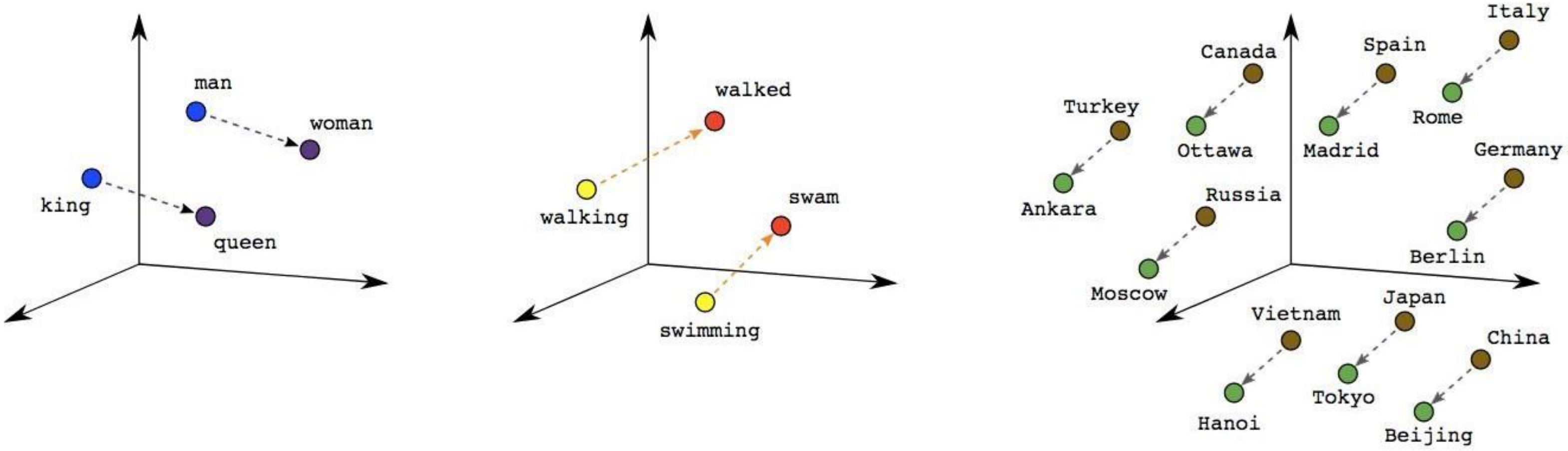
RAG: Componentes



RAG: Database



RAG: Embeddings



Male-Female Verb Tense Country-Capital

RAG: LLM

Hugging Face

The task illustrated in this tutorial is supported by the following model architectures:

ALBERT, BART, BERT, BigBird, BigBird-Pegasus, BLOOM, CamemBERT, CANINE, ConvBERT, Data2VecText, DeBERTa, DeBERTa-v2, DistilBERT, ELECTRA, ERNIE, ErnieM, Falcon, FlauBERT, FNet, Funnel Transformer, OpenAI GPT-2, GPT Neo, GPT NeoX, GPT-J, I-BERT, LayoutLMv2, LayoutLMv3, LED, LiLT, Longformer, LUKE, LXMERT, MarkupLM, mBART, MEGA, Megatron-BERT, MobileBERT, MPNet, MPT, MRA, MT5, MVP, Nezha, Nyströmformer, OPT, QDQBert, Reformer, RemBERT, RoBERTa, RoBERTa-PreLayerNorm, RoCBert, RoFormer, Splinter, SqueezeBERT, T5, UMT5, XLM, XLM-RoBERTa, XLM-RoBERTa-XL, XLNet, X-MOD, YOSO