

MINERÍA DE DATOS

Maximiliano Ojeda

muojeda@uc.cl



IIC-2433



Texto en Minería de Datos

Motivación

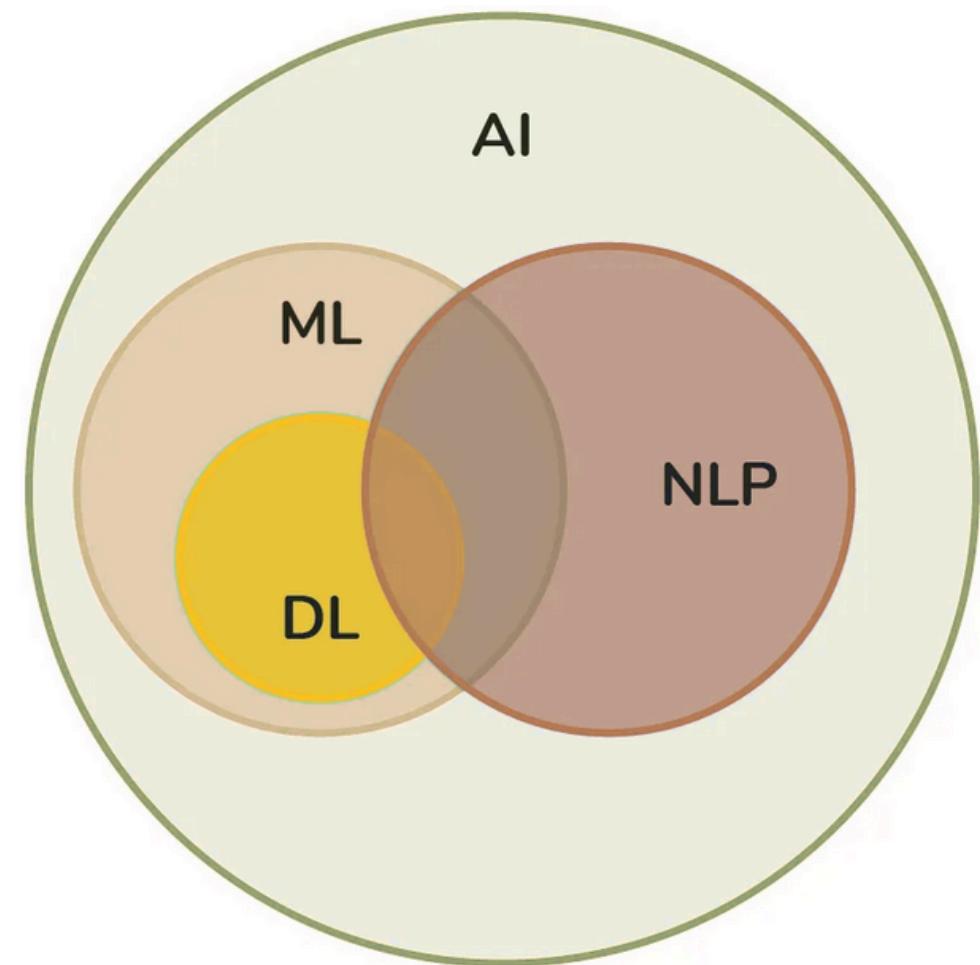
◆ Importancia hoy en día



Conceptos Importantes

Natural Language Processing

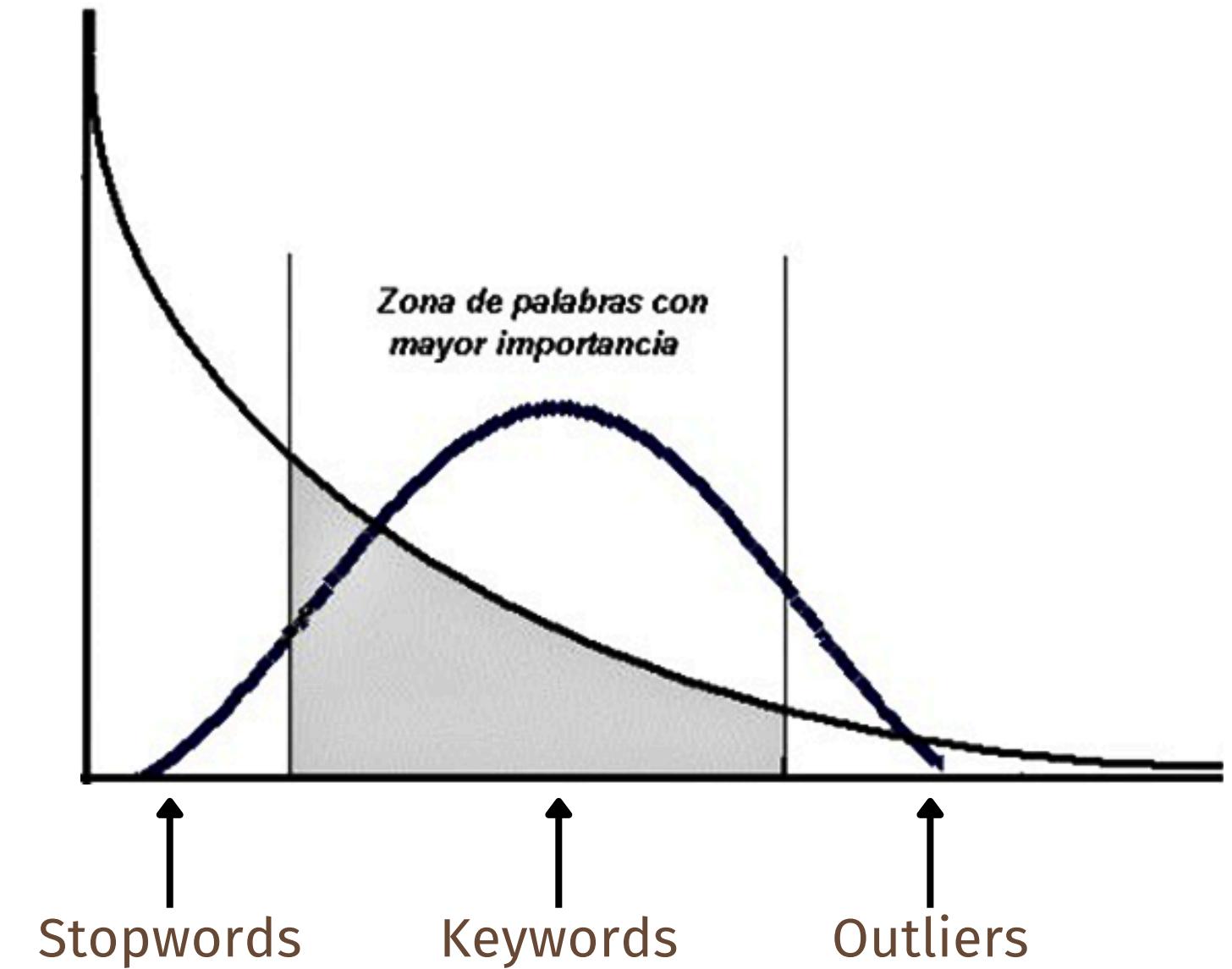
NLP es la rama de la IA que permite a los computadores entender, generar y razonar con el lenguaje humano combinando lingüística computacional, aprendizaje estadístico y deep learning.



Stopwords

Las **stopwords** son palabras muy frecuentes en un idioma (artículos, preposiciones, etc.) que aportan poco significado léxico (“el”, “la”, “de”, “y”). Se suelen eliminar para:

- Reducir el tamaño del vocabulario y el índice.
- Mejorar eficiencia y relevancia al enfocar el análisis en términos con más peso semántico.



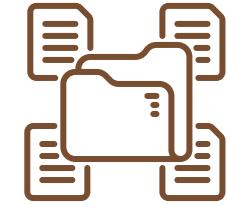
Stopwords

Las **stopwords** son palabras muy frecuentes en un idioma (artículos, preposiciones, etc.) que aportan poco significado léxico (“el”, “la”, “de”, “y”). Se suelen eliminar para:

- Reducir el tamaño del vocabulario y el índice.
- Mejorar eficiencia y relevancia al enfocar el análisis en términos con más peso semántico.

```
import nltk  
nltk.download('stopwords')  
from nltk.corpus import stopwords  
  
# Stopwords español  
spanish_sw = stopwords.words('spanish')  
print(spanish_sw)  
  
['de', 'la', 'que', 'el', ... , 'tenidas', 'tened']
```

Conceptos de NLP



Corpus

Colección estructurada de textos usada como datos de entrenamiento o análisis en NLP (p. ej. Brown, Wikipedia, tweets).

0	The Fulton County Grand Jury said Friday an investigation of Atlanta's recent
1	The jury further said in term-end presentations that the City Executive Comm
2	The September-October term jury had been charged by Fulton Superior Cou
3	“ Only a relative handful of such reports was received ”, the jury said, “ con:
4	The jury said it did find that many of Georgia’s registration and election laws
5	It recommended that Fulton legislators act “ to have these laws studied and
6	The grand jury commented on a number of other topics , among them the A



Documento

Unidad individual dentro del corpus, como un artículo, un párrafo o un mensaje.

The jury said it did find that many of Georgia's registration and election laws "are outmoded or inadequate and often ambiguous".



Token

Elemento atómico extraído del texto tras la tokenización (palabras, números o símbolos), que sirve como entrada.

```
['The', 'jury', 'said', 'it', 'did', 'find', 'that', 'many', 'of', 'Georgia', "s", 'registration', 'and', ...]
```

Tokenización

La tokenización es el proceso de segmentar un texto en tokens (por ejemplo palabras, signos de puntuación o subpalabras), que sirven como entrada para algoritmos de NLP

Tokenización con NLTK

```
from nltk.tokenize import word_tokenize, sent_tokenize, RegexpTokenizer

text = """Coming down like an Armageddon flame (hey!).
The shame, the ones who died without a name"""

# Sentence tokenizer
sentences = sent_tokenize(text)

# Word tokenizer
words = word_tokenize(text)

# Regexp tokenizer
regexp = RegexpTokenizer(r'\w+')
regex_tokens = regexp.tokenize(text)
```

Sentence Tokenizer:

```
['Coming down like an Armageddon flame (hey!).', 'The shame, the ones who died without a name']
```

Word Tokenizer:

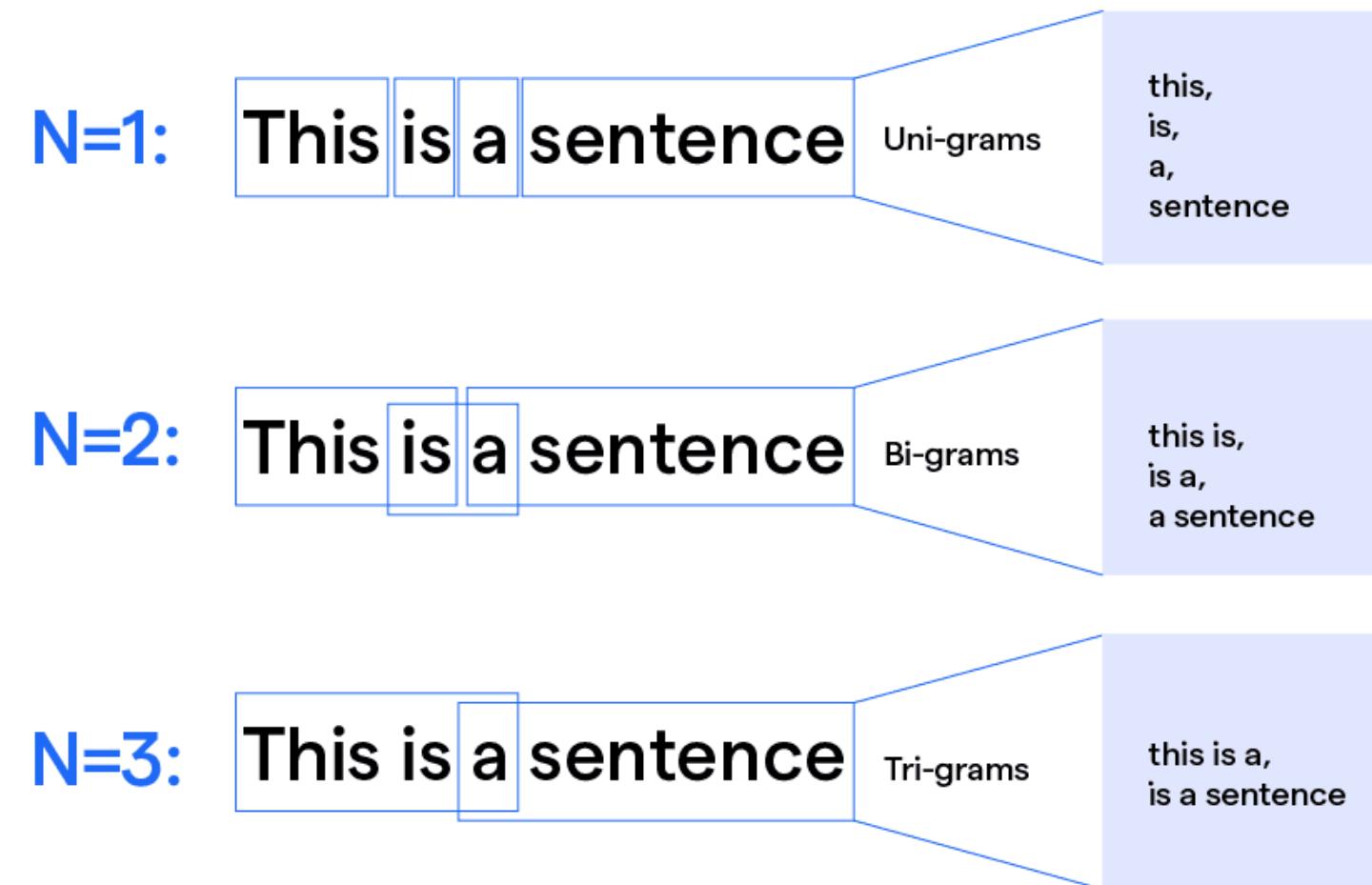
```
['Coming', 'down', 'like', 'an', 'Armageddon', 'flame', '(', 'hey', ')', '.', 'The', 'shame', ',', 'the', 'ones', 'who', 'died', 'without', 'a', 'name']
```

Regexp Tokenizer (\w+):

```
['Coming', 'down', 'like', 'an', 'Armageddon', 'flame', 'hey', 'The', 'shame', 'the', 'ones', 'who', 'died', 'without', 'a', 'name']
```

N-gram

Un n-grama es una secuencia contigua de n unidades (palabras o caracteres) extraídas de un texto. Los n-gramas añaden contexto local que los tokens aislados no capturan (colocaciones, ambigüación).

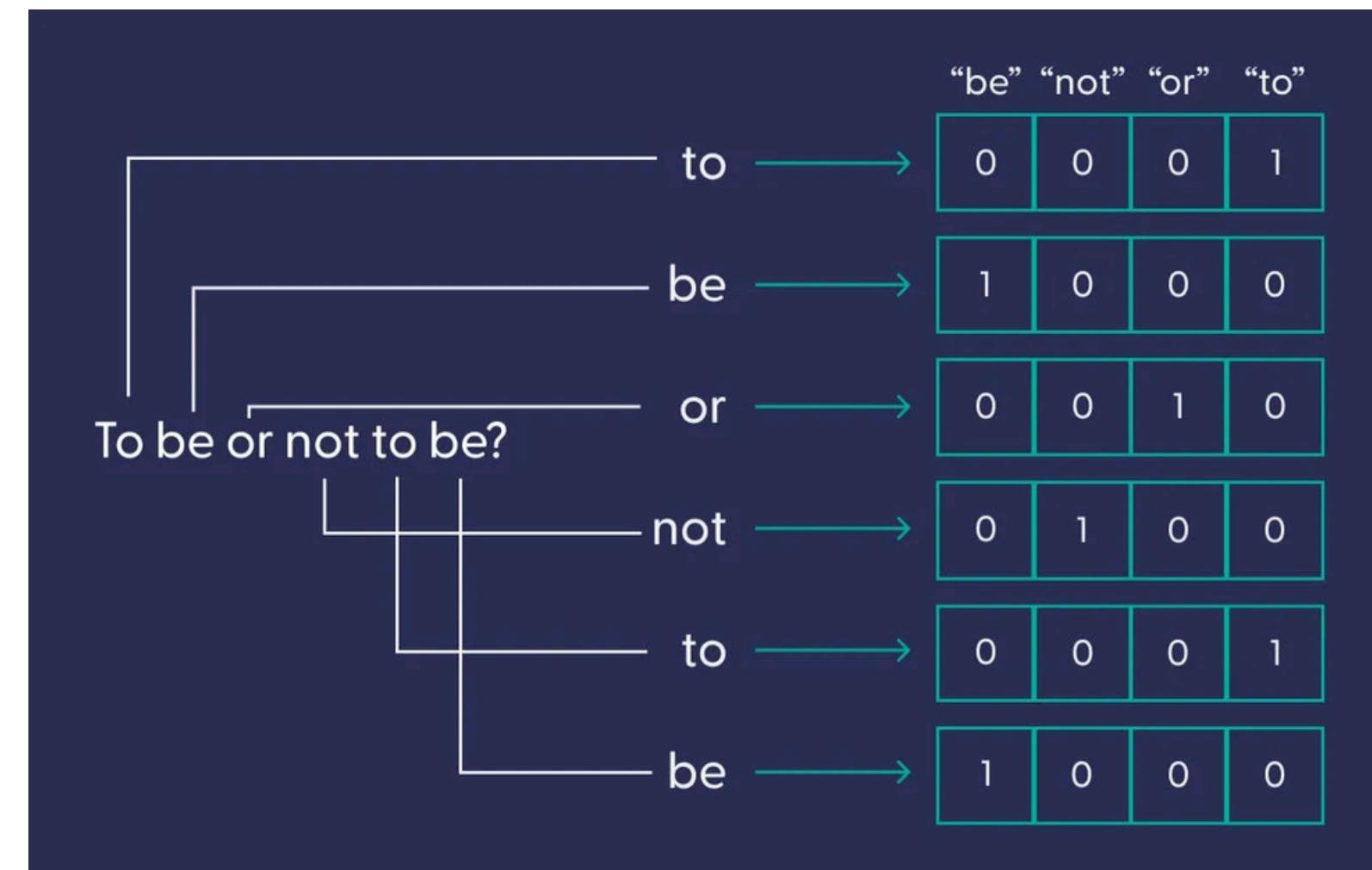




Representación de Texto

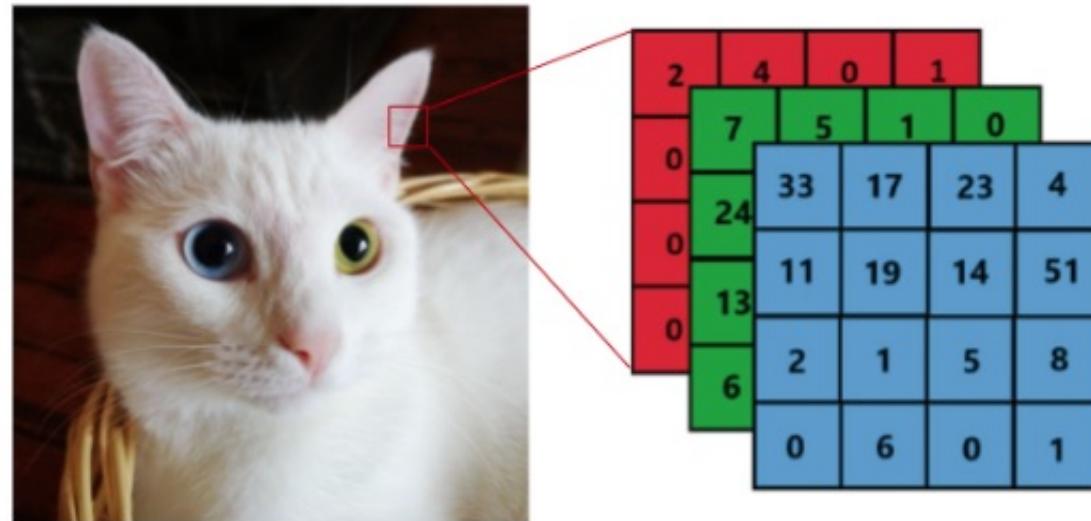
¿Qué se entiende por “representar”?

Nos referimos a cómo la información del mundo se transforma a una forma que una máquina pueda manejar y razonar sobre ella.

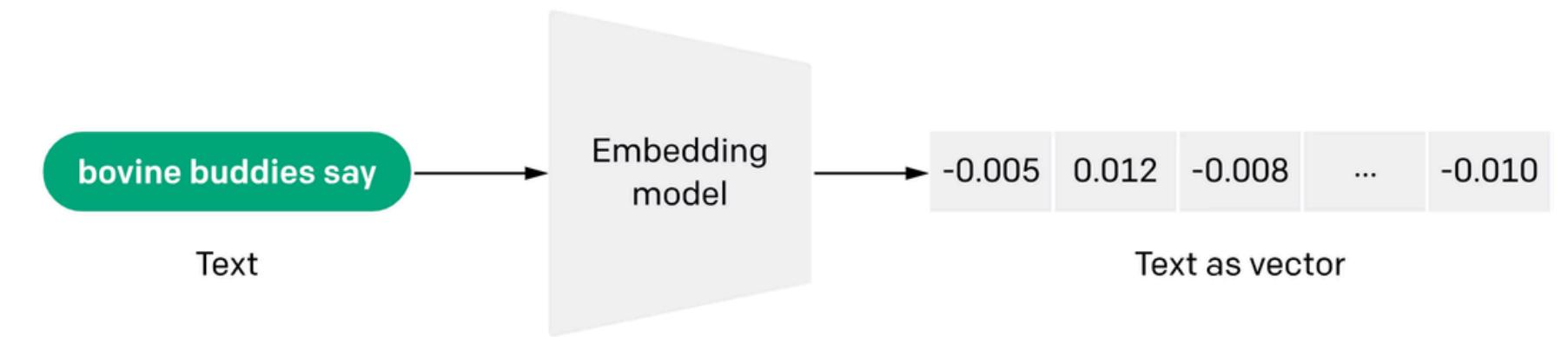


¿Qué se entiende por “representar”?

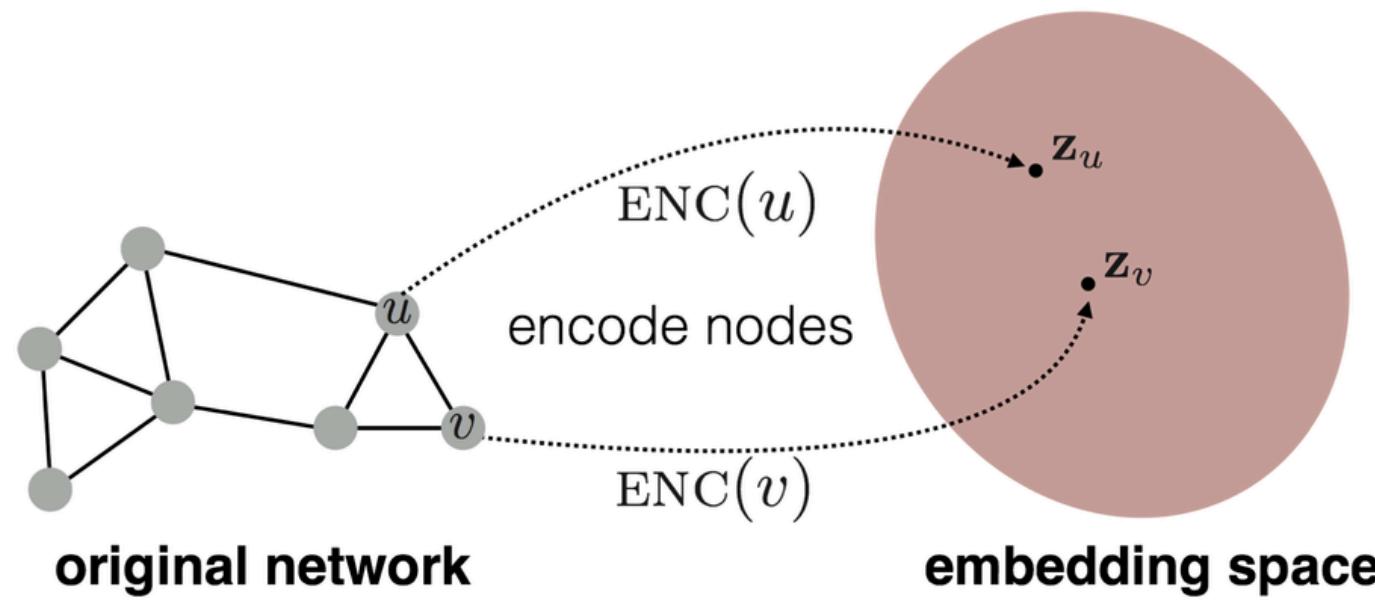
Computer Vision

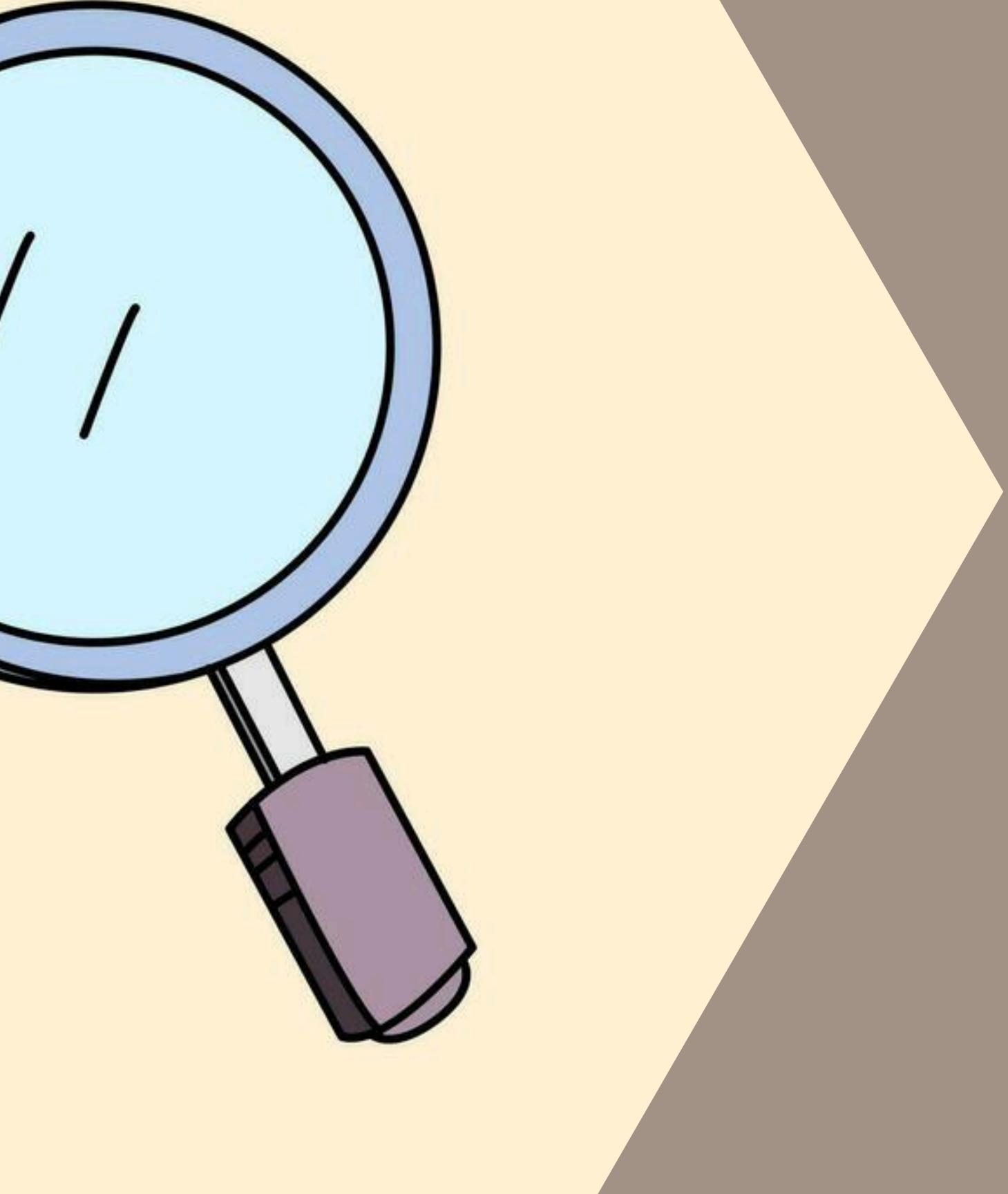


NLP



Grafos





Bag of Words

¿Qué es Bag of Words (BoW)?

Es una de las formas más básicas de **representar texto** en NLP e IR. Consiste en:

1. Tomar un texto (frase, párrafo, documento).
2. Dividirlo en sus palabras (tokens).
3. Contar cuántas veces aparece cada palabra.
4. Guardar esas cuentas en un vector.

	about	bird	heard	is	the	word	you
About the bird , the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Bag of Words (BoW)

```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
    "El perro corre rápido y el perro ladra fuerte.",
    "La casa es grande, la casa es bonita.",
    "Me gusta la música, la música es mi pasión.",
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)

print("Vocabulario:", vectorizer.get_feature_names_out())
print(X.toarray())

Vocabulario: ['bonita' 'casa' 'corre' 'el' 'es' 'fuerte' 'grande' 'gusta' 'la' 'ladra'
 'me' 'mi' 'música' 'pasión' 'perro' 'rápido']
[[0 0 1 2 0 1 0 0 0 1 0 0 0 0 0 2 1]
 [1 2 0 0 2 0 1 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 1 2 0 1 1 2 1 0 0]]
```

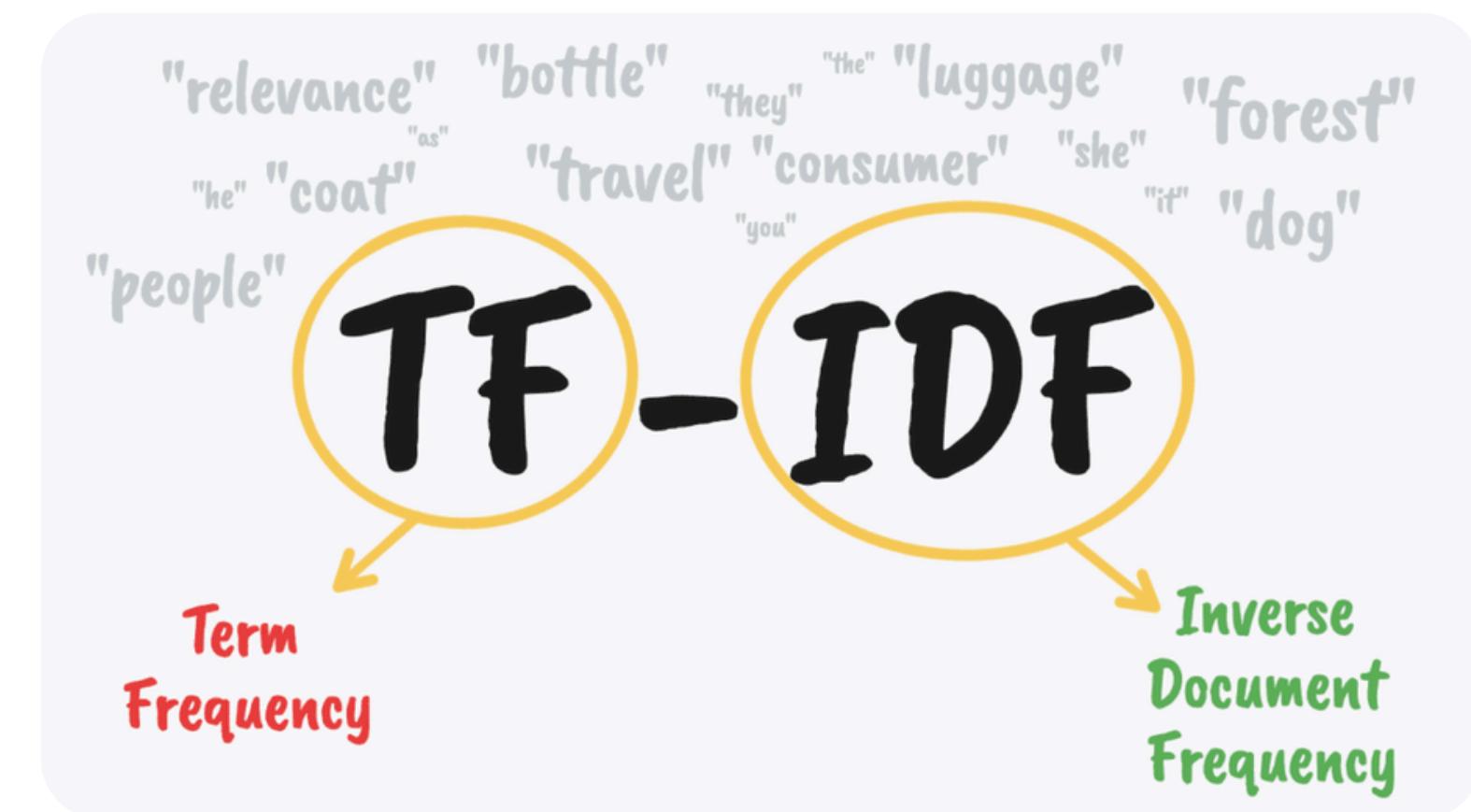


TF-IDF

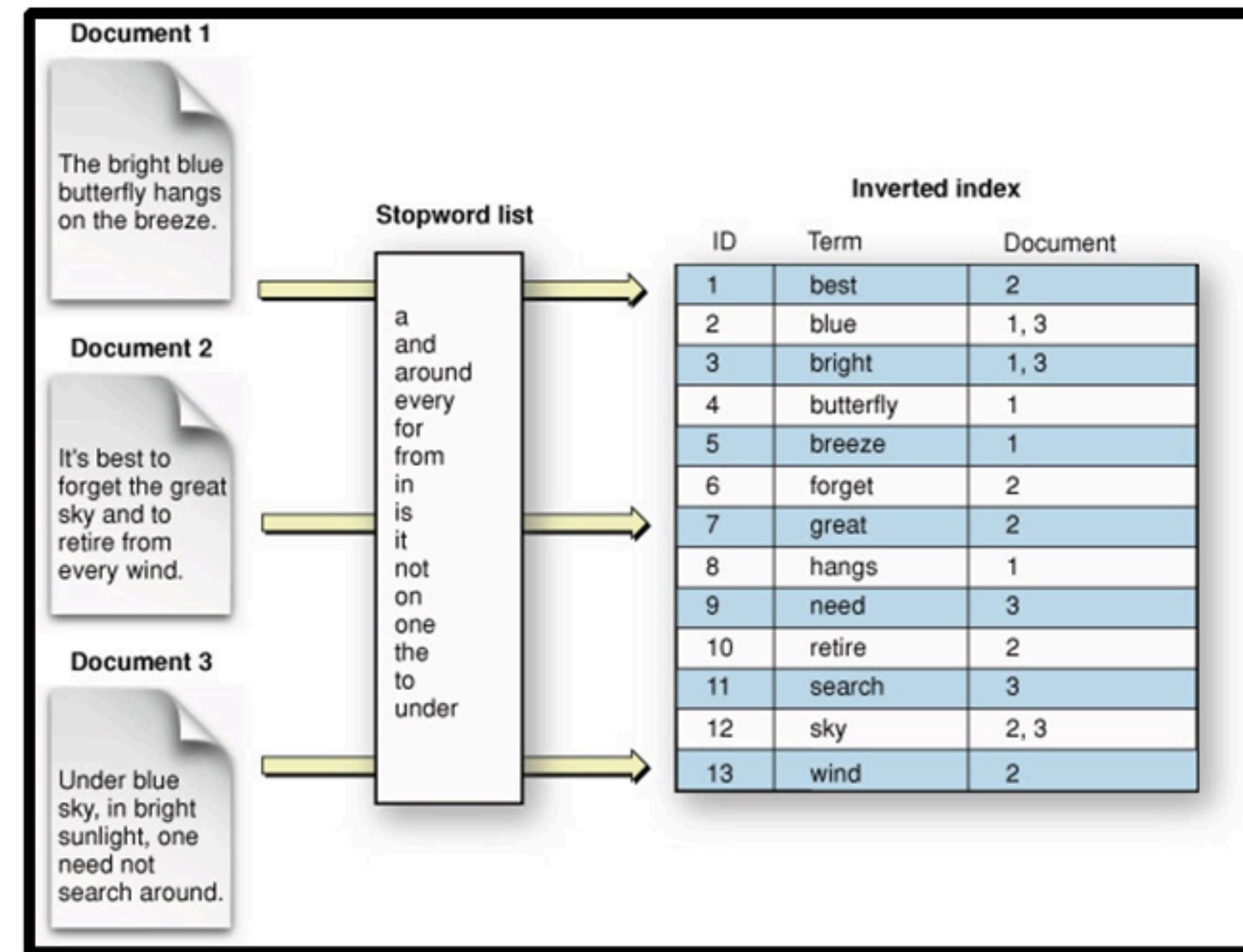
Term Frequency – Inverse Document Frequency

Es una **mejora sobre Bag of Words** porque no solo cuenta cuántas veces aparece una palabra en un documento, sino también **qué tan importante es esa palabra** en todo el corpus.

- En los 70s el área de IR estaba en auge por la necesidad de organizar grandes colecciones de documentos
- Se usaban conteos brutos (BoW) que no distinguían palabras comunes de palabras clave.
- En este contexto, varios investigadores (notablemente Karen Spärck Jones en 1972) introdujeron la idea de la **frecuencia inversa de documentos (IDF)**.



Índice Invertido



Term Frequency – Inverse Document Frequency

Cuántas veces aparece una palabra en un texto, comparado con la longitud total de ese texto.

Le da **más valor a las palabras que aparecen en pocos documentos** de todo el conjunto, y menos a las que salen en casi todos

Combina ambos para **destacar las palabras que salen con frecuencia en un texto pero son raras en el resto del corpus.**

$$TF(t, d) = \frac{\text{Cantidad de veces que } t \text{ aparece en } d}{\text{Total de terminos en } d}$$

$$IDF(t) = \log \left(\frac{\text{Cantidad total de documentos}}{\text{Cantidad de documentos que contienen el termino } t} \right)$$

$$\text{TD-IDF}(t, d) = TF(t, d) \times IDF(t)$$

Term Frequency – Inverse Document Frequency

```
from nltk.corpus import stopwords
spanish_stopwords = stopwords.words("spanish")
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
    "La inflación sube en Chile: inflación, inflación e IPC empujan precios; ",
    "el IPC mensual refleja inflación en alimentos y energía.",",
    "Bancos centrales monitorean inflación e IPC; inflación moderada, proyecciones de "
    "crecimiento y crecimiento global.",",
    "Chile gana el clásico: goles, goles y más goles; la selección de Chile presiona alto "
    "y mantiene posesión.",",
    "Autos eléctricos, autos eléctricos y más autos eléctricos: mejora de batería, batería "
    "y autonomía en nuevos modelos.",",
    "El Congreso de Chile discute reforma, reforma constitucional; la reforma busca "
    "descentralización y consenso.",",
    "Santiago incorpora autos eléctricos al transporte público: recarga, recarga rápida y "
    "puntos de recarga para autos."
]

# Vectorizador TF-IDF
tfidf = TfidfVectorizer(stop_words=spanish_stopwords, lowercase=True, max_features=30,
ngram_range=(1,1))

X = tfidf.fit_transform(corpus)
vocab = np.array(tfidf.get_feature_names_out())
X_dense = X.toarray()
```

¿Para qué sirven los vectores de texto?

Los vectores son la puerta de entrada para aplicar algoritmos de Machine Learning.

1. Clasificación de Texto

- Spam vs No Spam
- Sentiment Analysis
- Clasificación de Noticias

2. Clustering

- Encontrar grupos de documentos similares

3. Búsqueda e IR

- Comparar consultas vs documentos usando similaridad de coseno.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

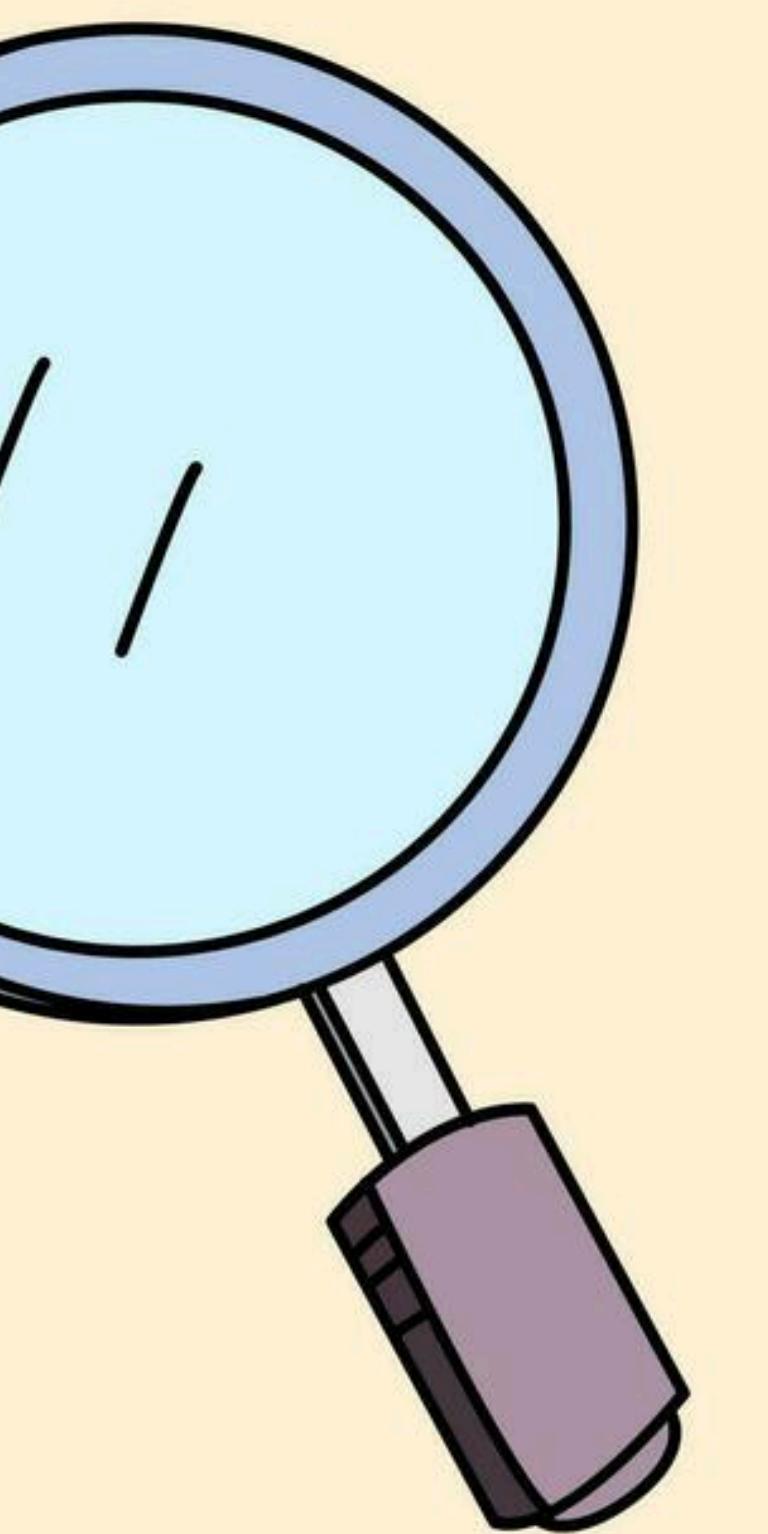
# Corpus y etiquetas (1=positivo, 0=negativo)
corpus = [
    "Me encanta este producto de gran calidad",
    "El servicio fue excelente y muy rápido",
    "Estoy muy satisfecho con mi compra",
    "La comida estaba deliciosa y bien servida",
    "Recomiendo totalmente esta tienda a mis amigos",
    "Odio este producto tan malo y defectuoso",
    "El servicio fue terrible y muy lento",
    "Estoy muy decepcionado con la experiencia",
    "La comida estaba fría y mal servida",
    "No recomiendo esta tienda por su atención"
]
labels = [1,1,1,1,1, 0,0,0,0,0]

# Vectorizar
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)

# Modelo
clf = LogisticRegression()
clf.fit(X, labels)

# Predicción de ejemplo
texto = ["El producto llegó rápido y funciona bien"]
print("Predicción:", clf.predict(vectorizer.transform(texto)))
print("Probabilidades:", clf.predict_proba(vectorizer.transform(texto)))

Predicción: [1]
Probabilidades: [[0.44268013 0.55731987]]
```



Modelos de Lenguaje

Vector-Space Model

Cada término se representa como una dimensión del vector

The diagram illustrates the Vector-Space Model for four terms: Rome, Paris, Italy, and France. Each term is represented by a vector in a high-dimensional space. The vectors are as follows:

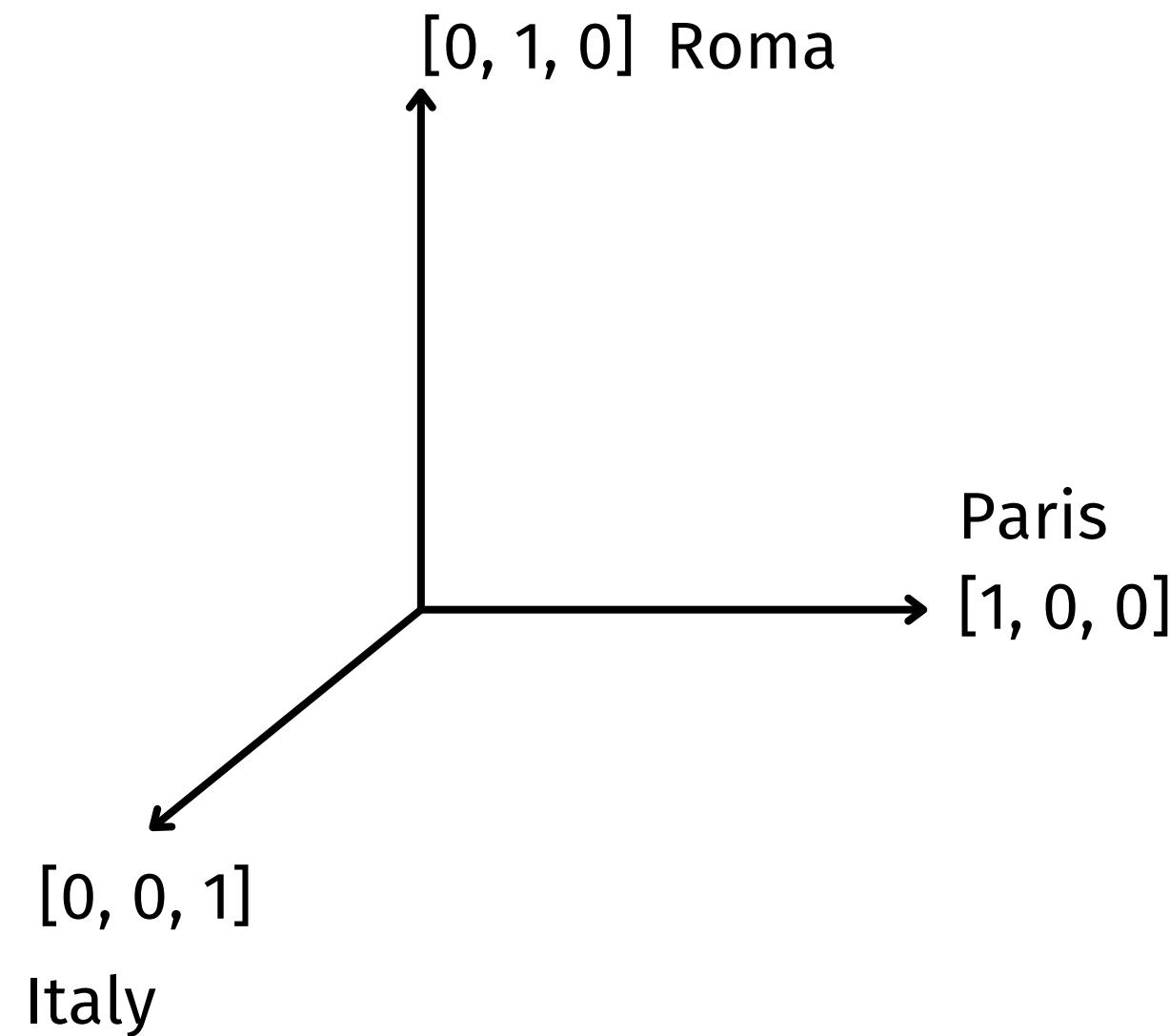
- Rome = [1, 0, 0, 0, 0, 0, ..., 0]
- Paris = [0, 1, 0, 0, 0, 0, ..., 0]
- Italy = [0, 0, 1, 0, 0, 0, ..., 0]
- France = [0, 0, 0, 1, 0, 0, ..., 0]

Annotations with arrows point to specific dimensions in the vectors:

- An arrow labeled "Rome" points to the first dimension (index 0) of the Rome vector.
- An arrow labeled "Paris" points to the second dimension (index 1) of the Paris vector.
- An arrow labeled "Italy" points to the third dimension (index 2) of the Italy vector.
- An arrow labeled "France" points to the fourth dimension (index 3) of the France vector.
- An arrow labeled "word V" points to the last dimension (index n-1) of all vectors.

Vector-Space Model

Cada término se representa como una dimensión del vector



Vector-Space Model

Cuando tenemos una situación de polisemia los vectores se representan de la misma forma sin hacer distinción del contexto

Tengo que ir a pagar al **banco**

Deberían arreglar el **banco** de la plaza

banco en Doc 1: [0, 0, 1, 0, 0]
banco en Doc 2: [0, 0, 1, 0, 0]

Limitaciones

Pensemos en que tenemos un vocabulario de más de 30.000 palabras. Cada palabra está representado por un vector de 30.000 dimensiones (**v**)

I: [0, 0, 0, 0, 0, **1**, 0, ..., ..., 0, 0]

cat: [0, 0, 0, **1**, 0, 0, 0, ..., ..., 0, 0]

dog: [0, **1**, 0, 0, 0, 0, 0, ..., ..., 0, 0]

have: [0, 0, 0, 0, 0, 0, 0, ..., ..., **1**, 0]

a: [0, 0, 0, 0, 0, 0, 0, ..., ..., 0, **1**]



30.000 dim

- Muy disperso (alta dimensionalidad).
- No captura relaciones semánticas (no sabe que gato y perro son parecidos).

Word2Vec

Word2Vec es un modelo creado por Tomas Mikolov y su equipo en Google (2013) que aprende a **representar palabras como vectores numéricos densos** que capturan el significado

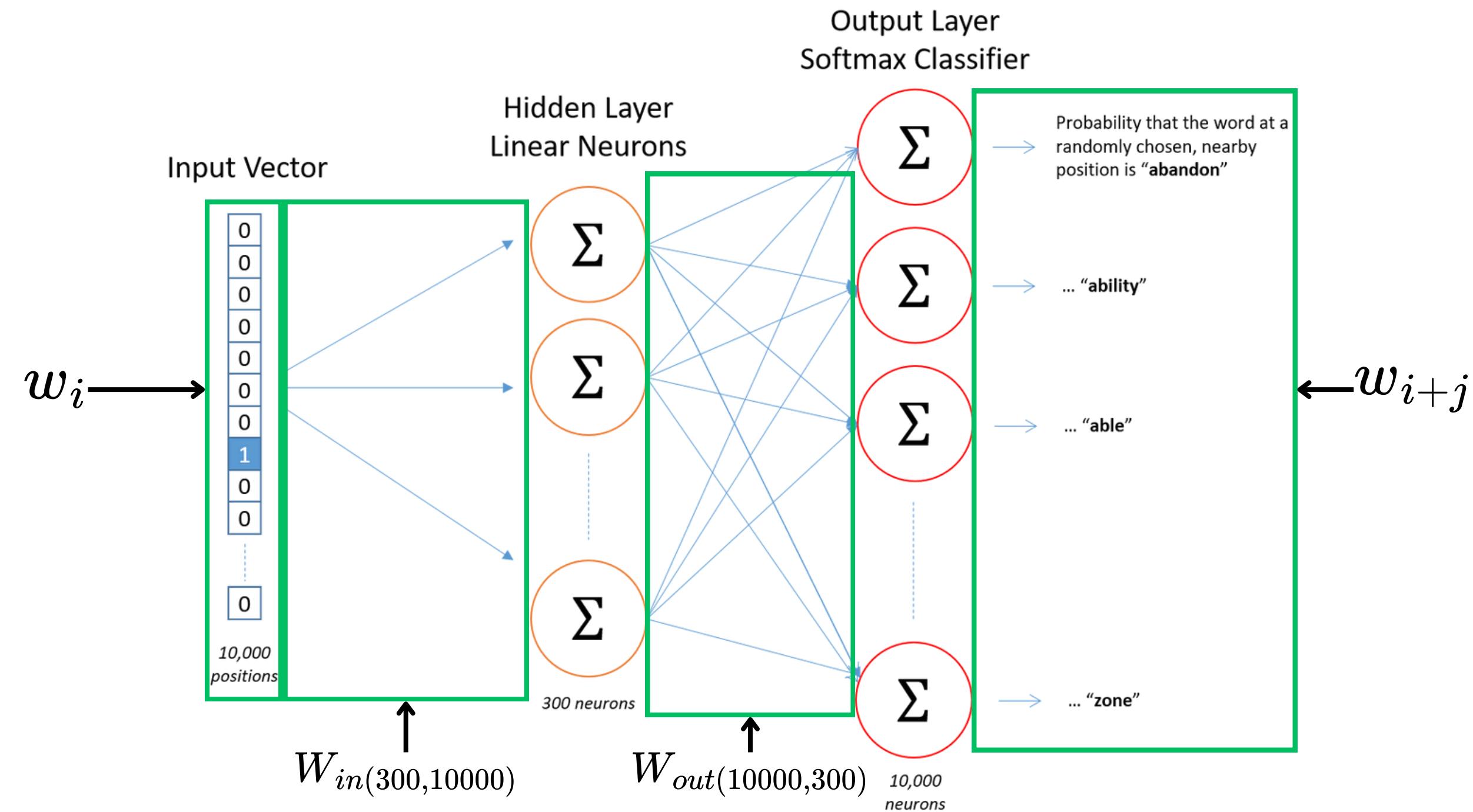
Palabra	Vector (dim=4)
gato	[0.21, -0.34, 0.11, 0.78]
perro	[0.25, -0.31, 0.09, 0.74]
mesa	[-0.51, 0.22, -0.33, 0.05]

Cómo funciona

Word2Vec es una red neuronal muy pequeña que aprende de texto sin etiquetas

Variante	Qué predice	Ejemplo
CBOW	Predice la palabra central a partir de las palabras del contexto.	Contexto: “El __ corre” → predice “perro”
Skip-Gram	Predice el contexto a partir de la palabra central.	“perro” → predice “El”, “corre”

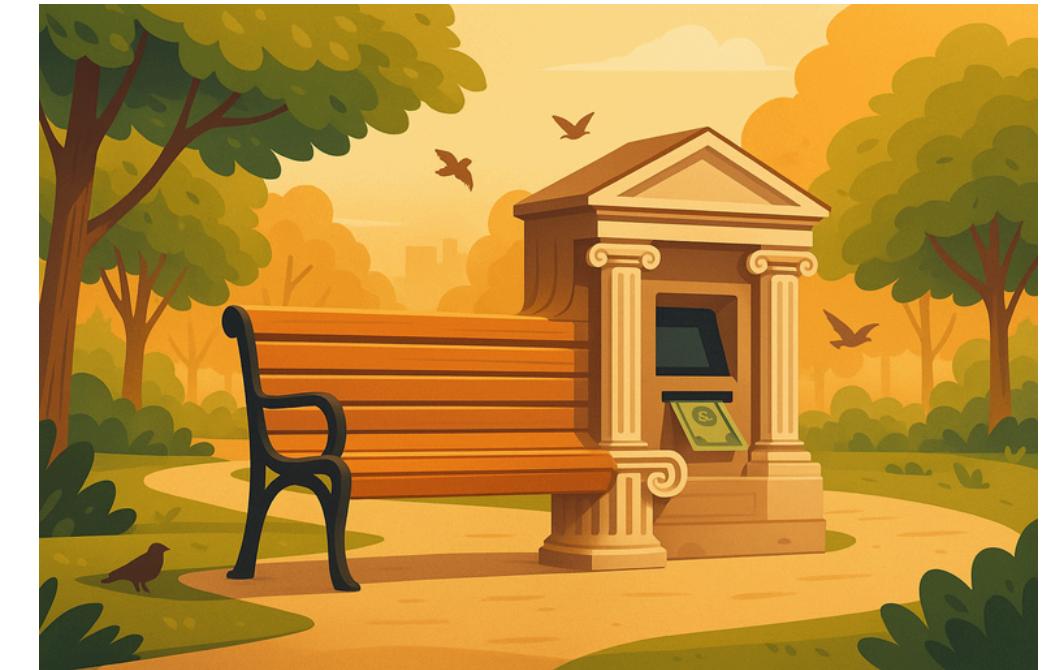
Word2Vec



Limitaciones Word2Vec

Embeddings estáticos (sin contexto)

La palabra tiene un solo vector sin importar la frase. “banco” (parque vs. entidad financiera) queda mezclado.



Out-of-Vocabulary (OOV) y morfología

No modela prefijos/sufijos ni caracteres. Palabras nuevas o con faltas → UNK.
En lenguas ricas en morfología (español), pierde información.

“barato” → “baratísimo” → “baratito”

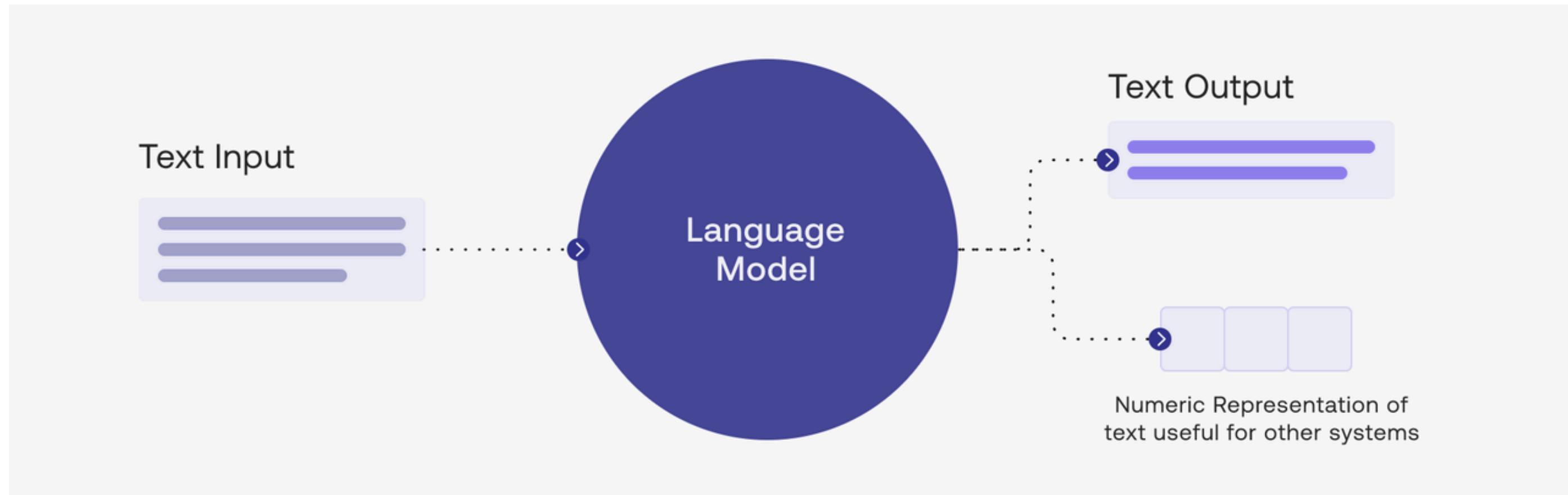
Vocabulario cerrado y corte por frecuencia

Hay que fijar un vocabulario; las palabras bajo el umbral de frecuencia quedan fuera o con vectores pobres.

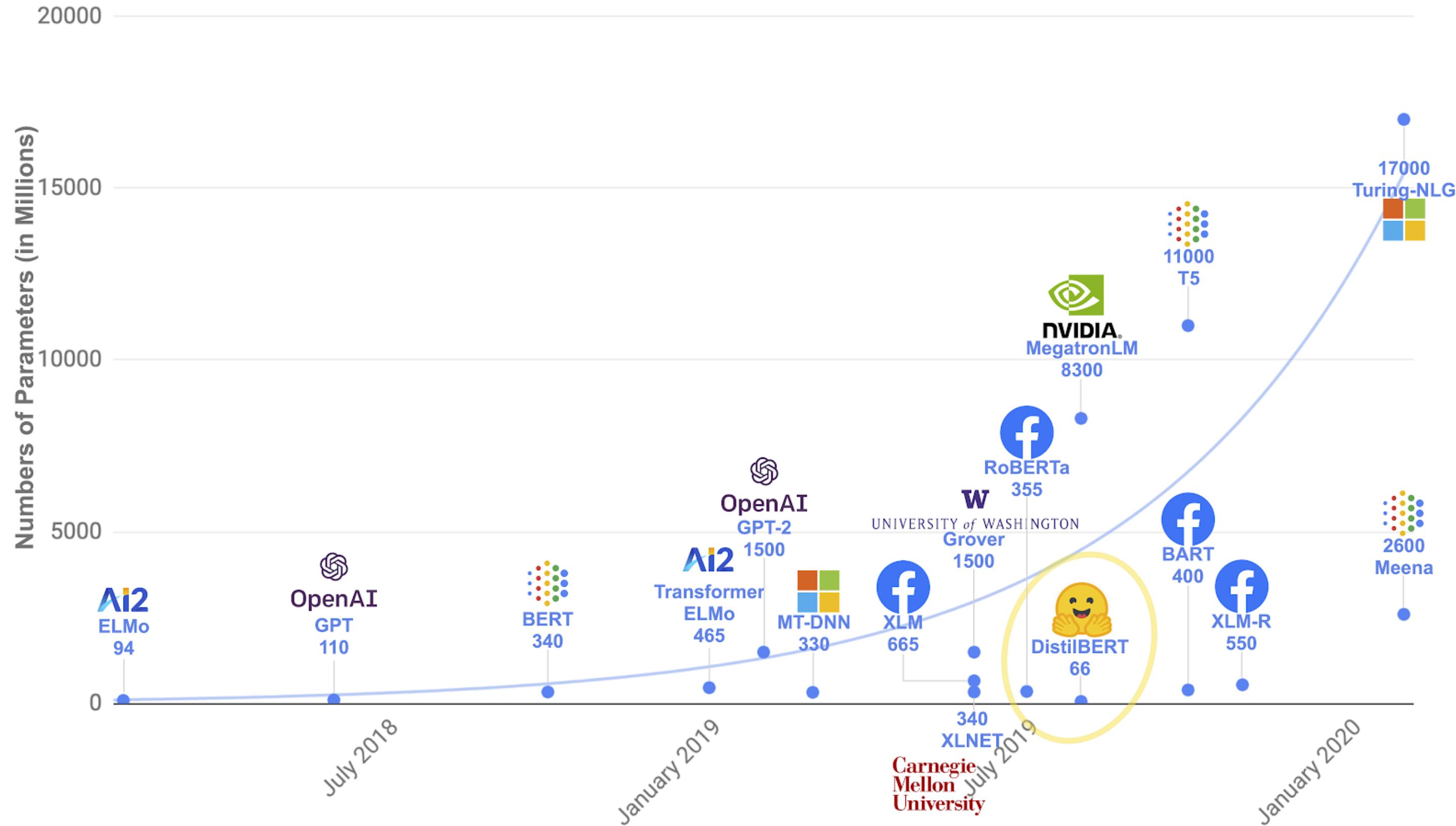
Modelos de lenguaje

Un modelo de lenguaje es un sistema que aprende a predecir la probabilidad de una secuencia de palabras en un idioma.

- Representan texto como vectores semánticos en espacios de alta dimensión.
- Capturan orden, contexto y significado.



Modelos de lenguaje



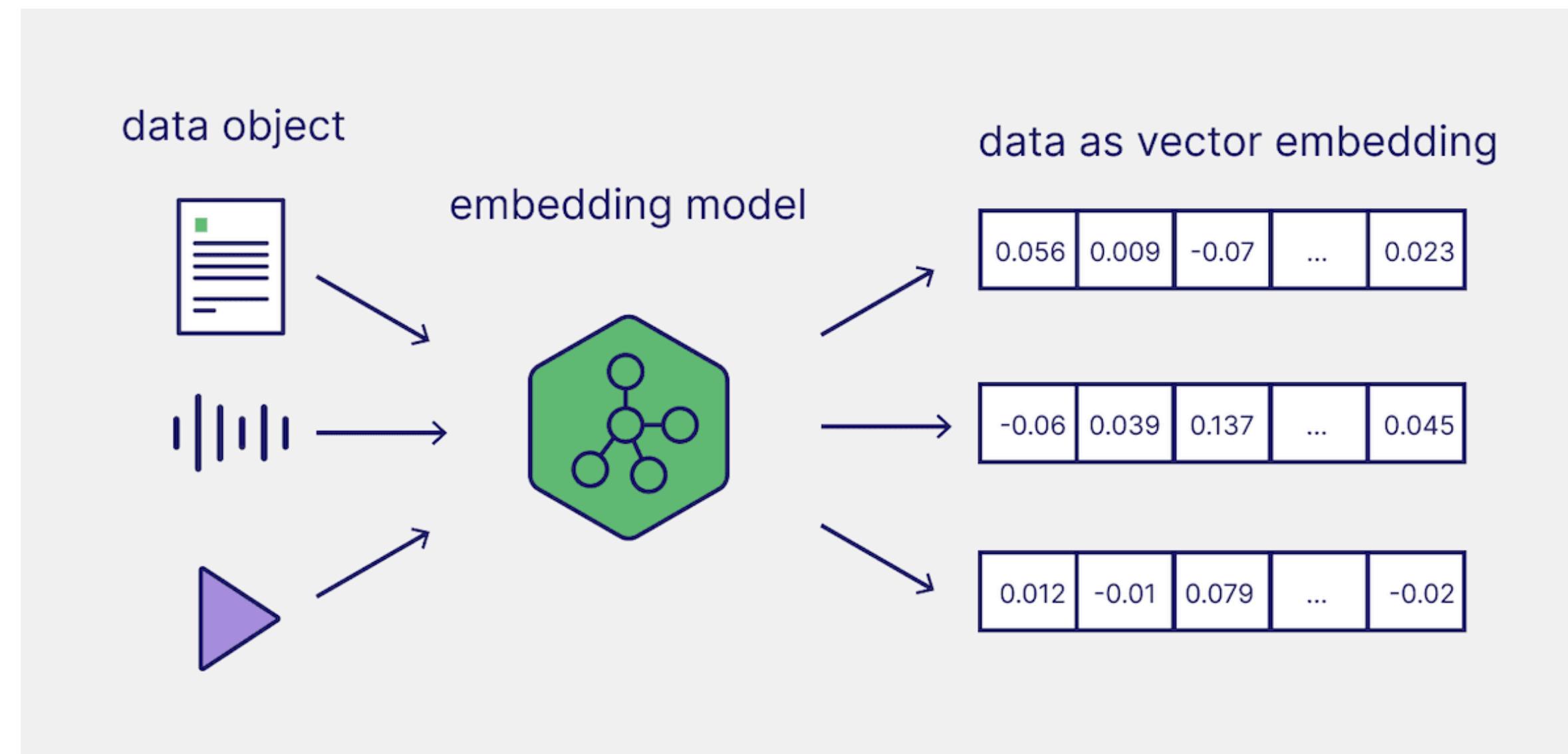
Modelos de lenguaje

Un **modelo de lenguaje** es cualquier sistema que predice o genera texto en base a probabilidades de secuencias de palabras. Un **Large Language Model (LLM)** es un tipo de modelo de lenguaje mucho más grande y poderoso, entrenado con miles de millones de parámetros y enormes cantidades de datos, lo que le permite no solo predecir palabras, sino también generar texto coherente, responder preguntas, razonar y adaptarse a múltiples tareas.

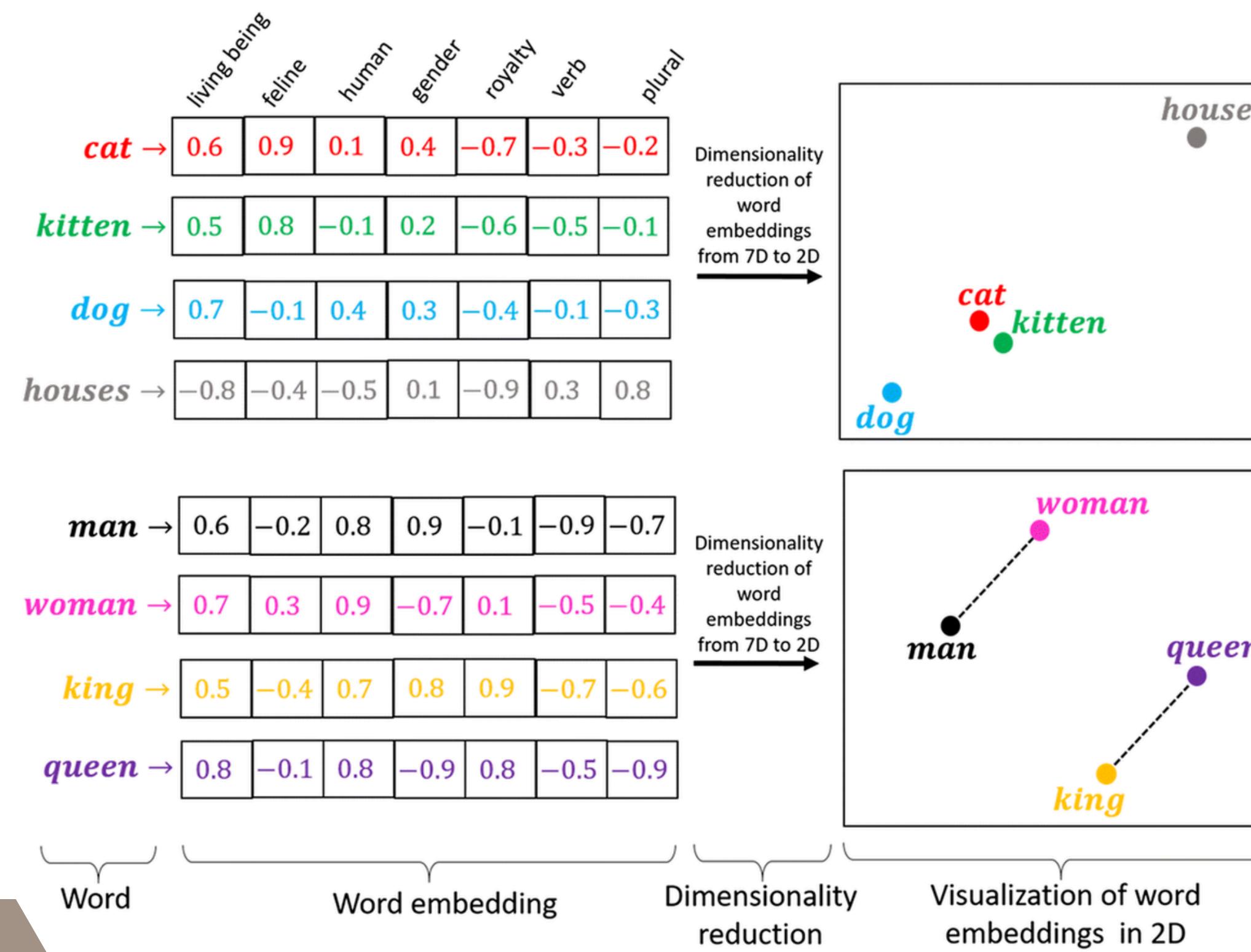
Metric	LLaMA 3.1	GPT-4	BERT	T5
Parameter Count	300 billion	175 billion	340 million	11 billion
Training Data Size	2.5 TB of text data	1.5 TB of text data	3.3 billion words	750 GB of text data
GLUE Score	92.5	90.5	80.5	89.9
SQuAD Score	90.3	88.9	86.7	87.7
SuperGLUE Score	89.7	87.4	78.3	86.2

¿Qué es un embedding?

Un embedding es una representación numérica de un objeto (palabra, frase, documento, imagen, etc.) en forma de vector dentro de un espacio matemático.



Embedding de Texto





Atención

Attention Is All You Need (vaswani, 2017)

Marcó un antes y un después en la **historia de la inteligencia artificial**. Su impacto fue tan grande que transformó completamente el campo de NLP y, con el tiempo, toda la IA moderna.

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,

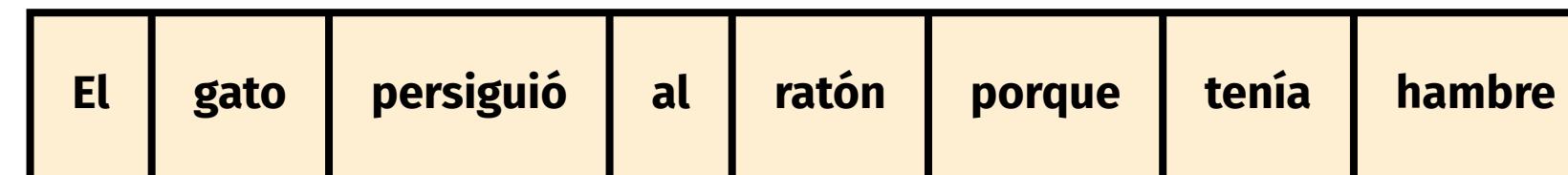
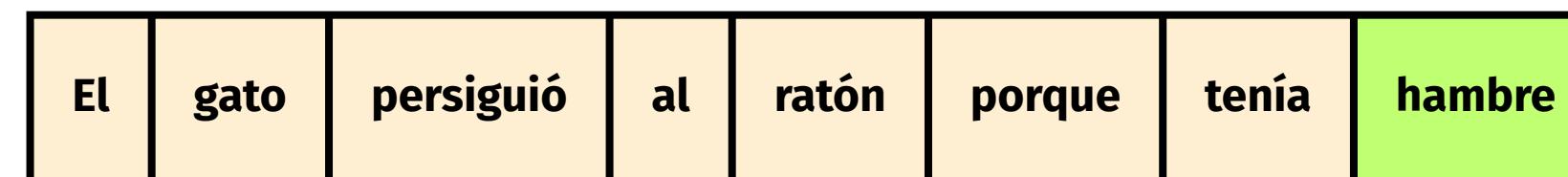
Attention Is All You Need (vaswani, 2017)

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 other tasks by applying it successfully to English constituency parsing both with large and limited training data.

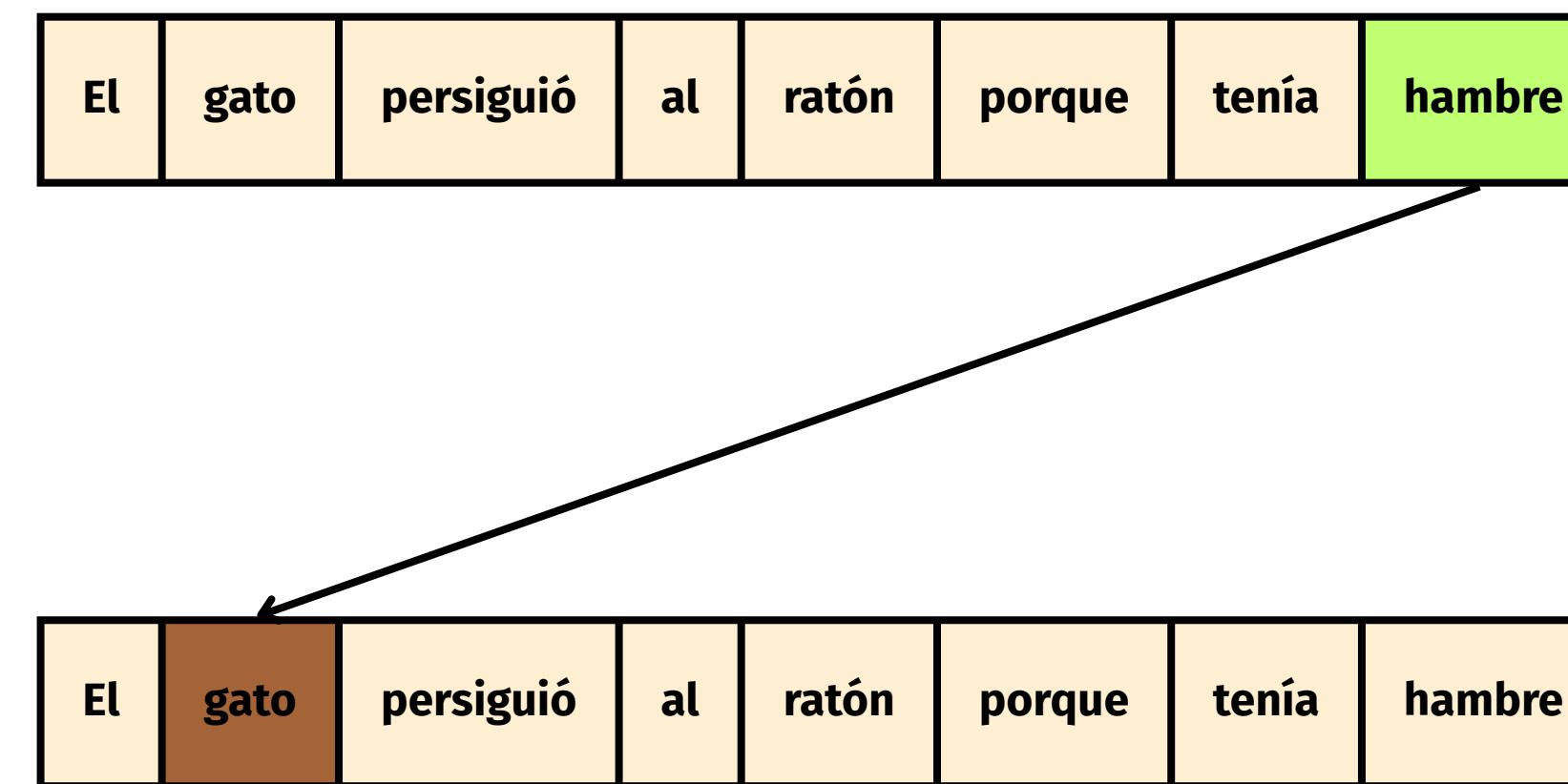
¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



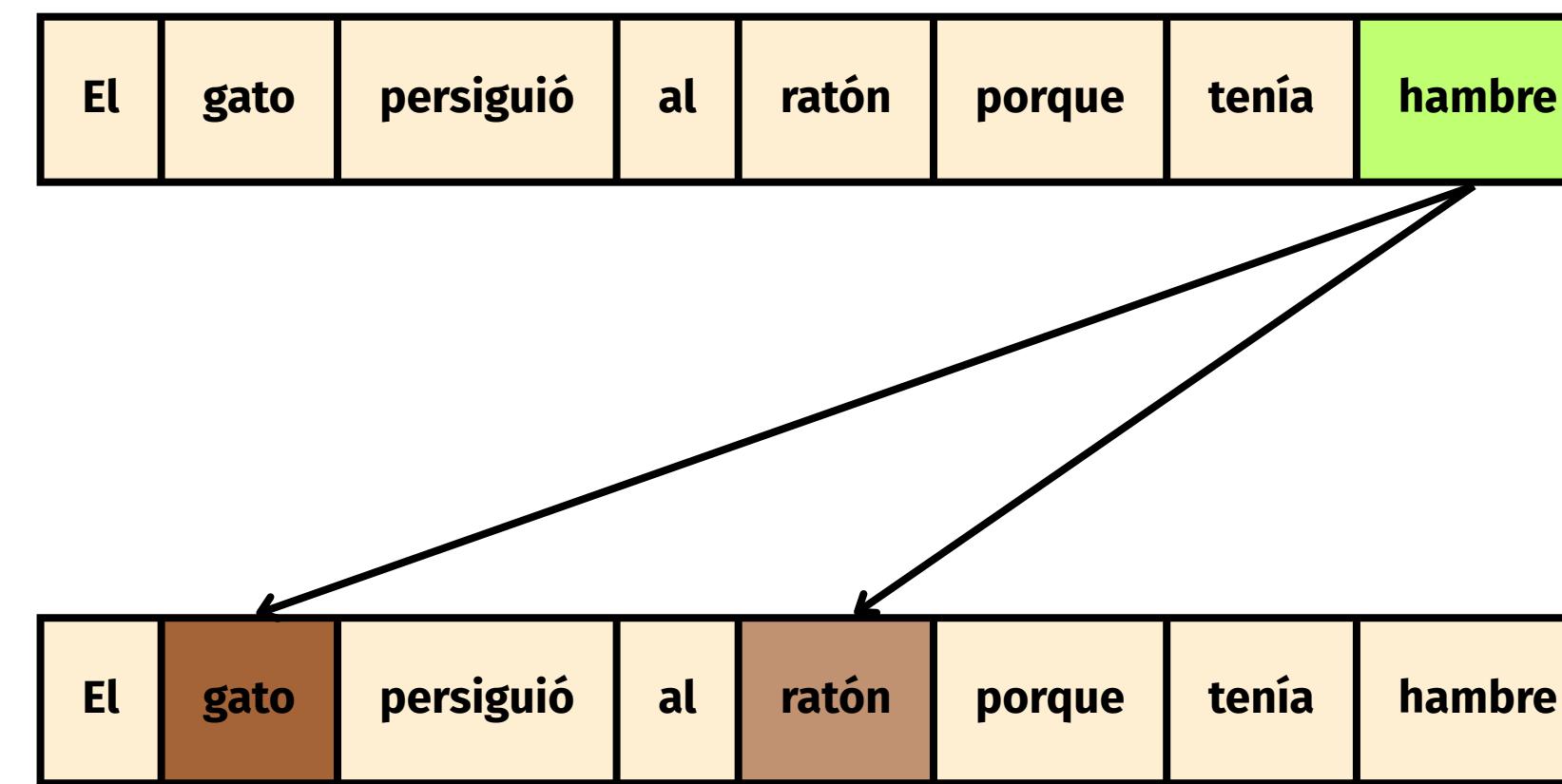
¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



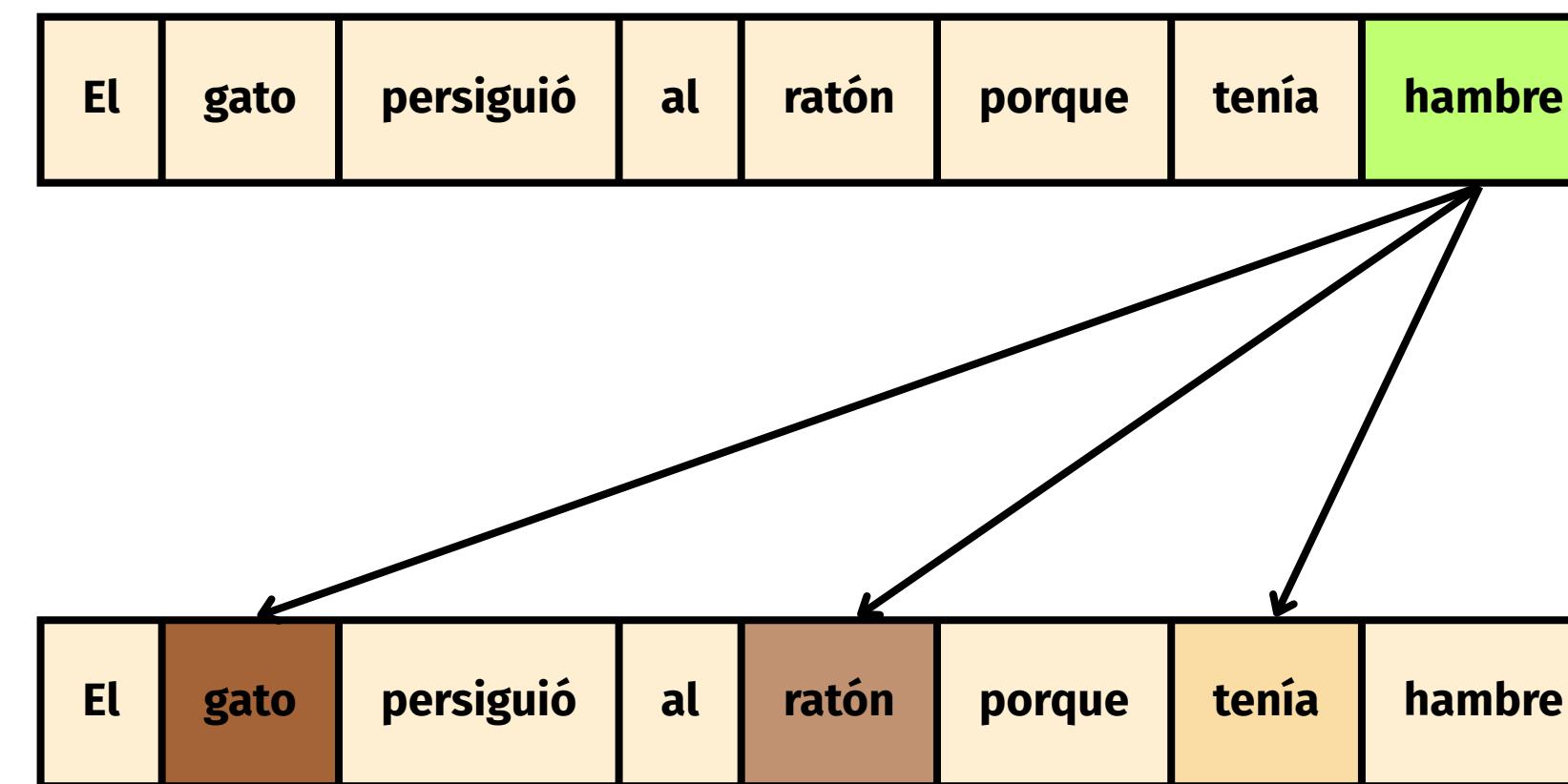
¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



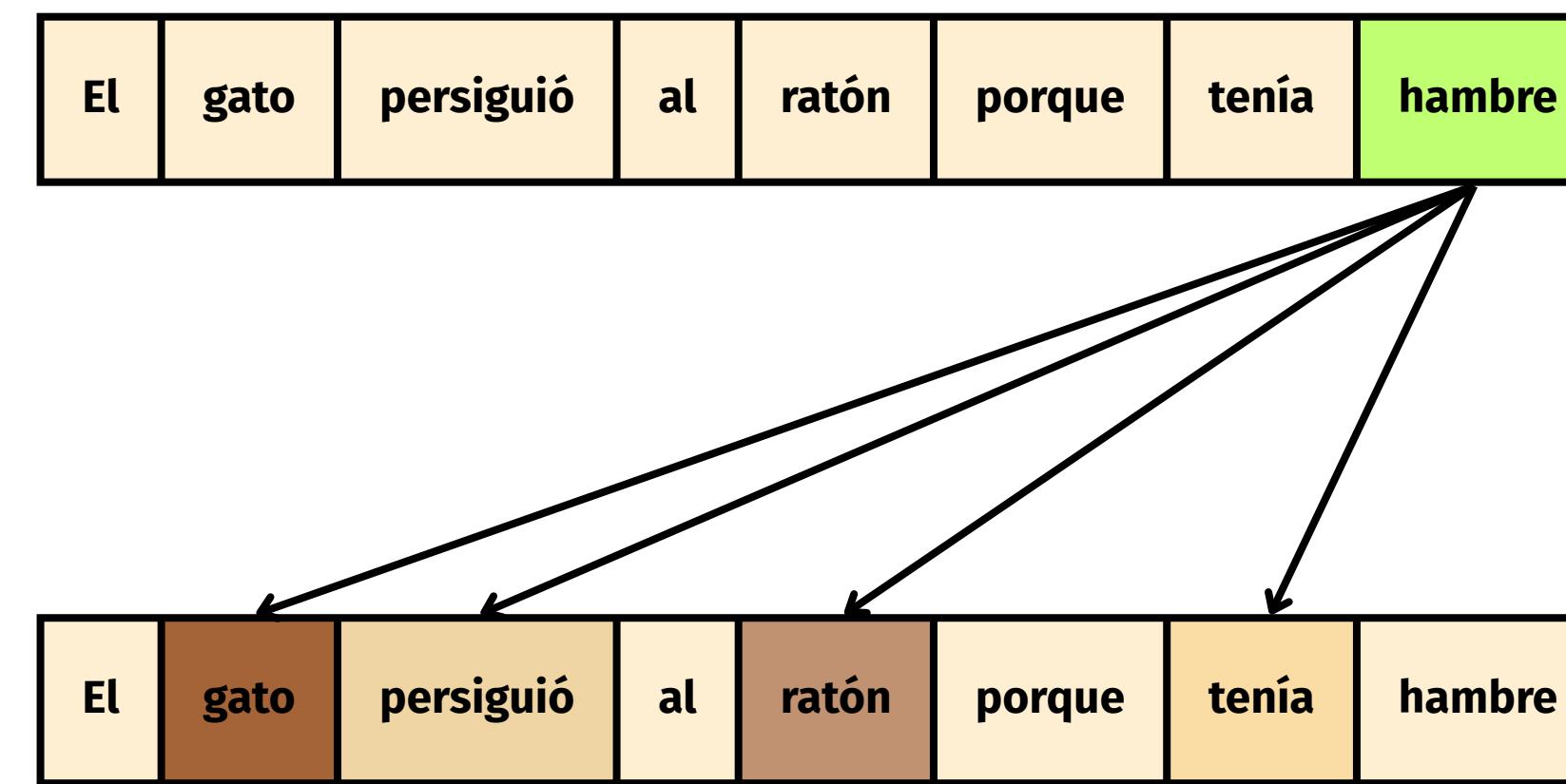
¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



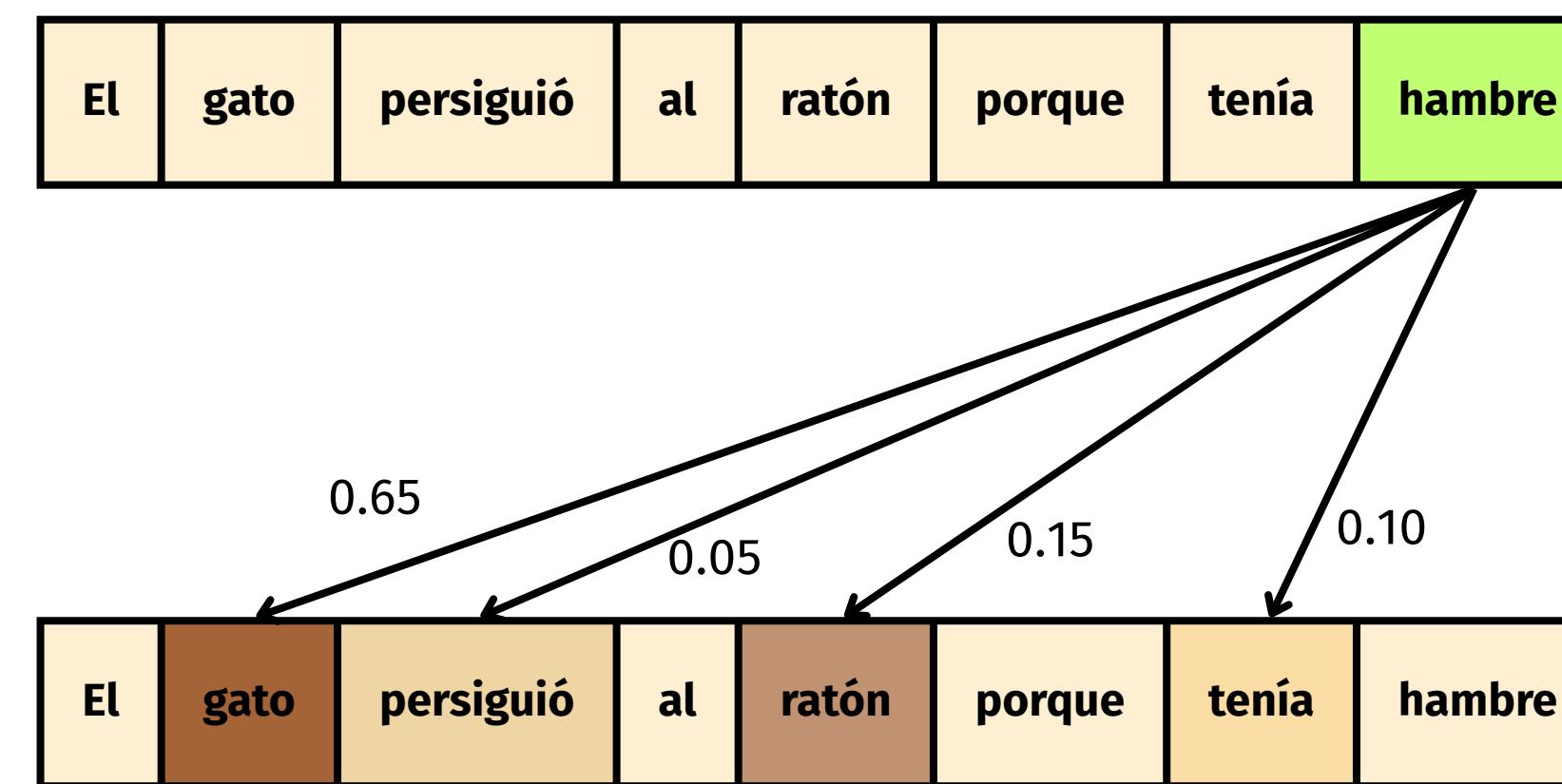
¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



¿Qué es la atención?

El **mecanismo de autoatención (self-attention)** permite que cada palabra aprenda su relación con todas las demás sin depender de un orden secuencial



¿Qué es la atención?

- Ya no hay hidden state que se pasa paso a paso.
- El modelo ve todo el contexto a la vez.
- Esto permitió paralelización masiva, dependencias largas y mayor estabilidad de entrenamiento.





Transformer

Transformers

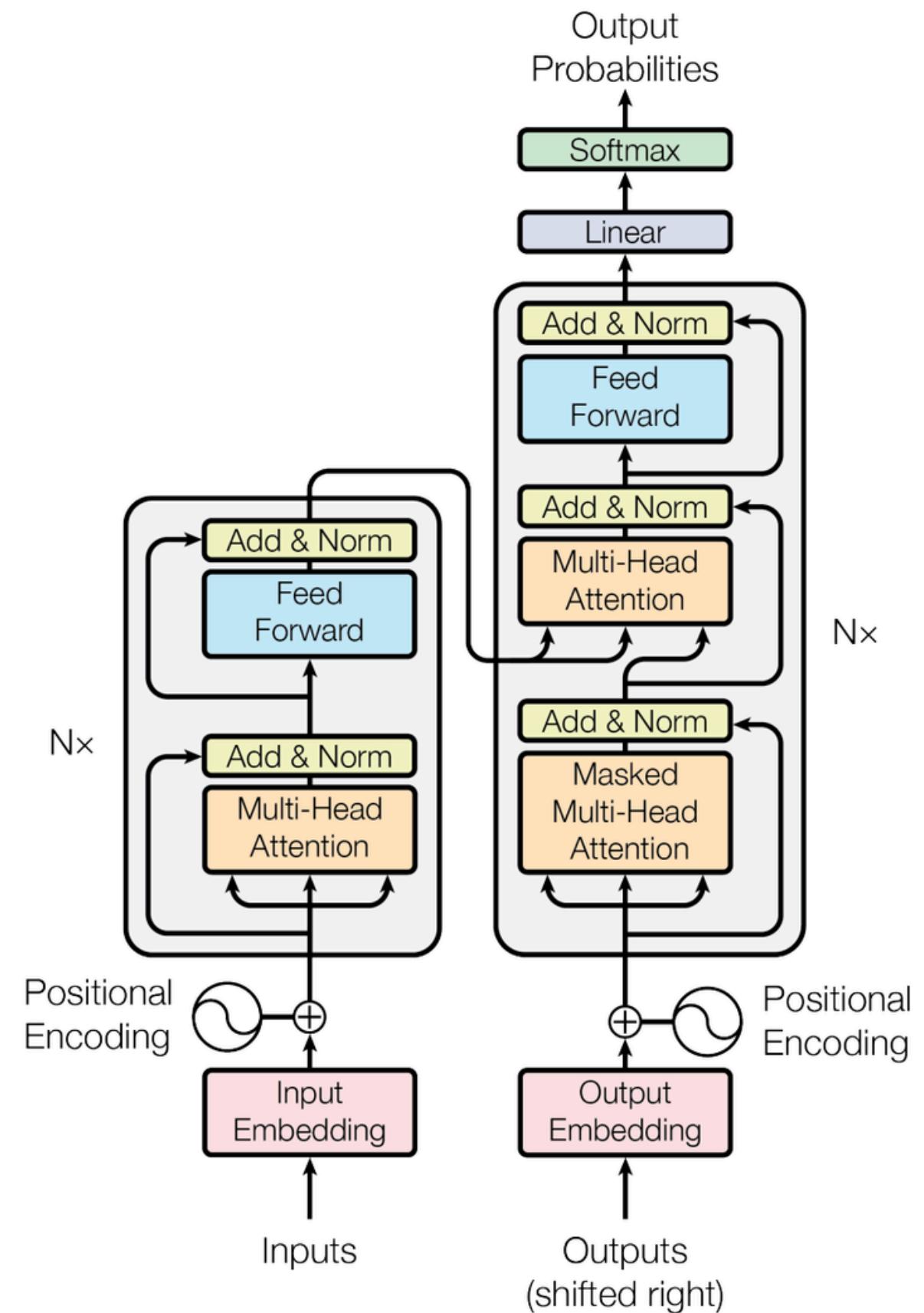
En lugar de procesar la información palabra por palabra (como lo hacían las RNN o LSTM), el Transformer procesa toda la secuencia a la vez y aprende qué partes del texto deben prestarse más atención unas a otras.

Encoder

Lee la secuencia de entrada completa . Calcula representaciones internas (embeddings contextuales) para cada palabra, entendiendo su relación con las demás.

Decoder

Genera una salida paso a paso (por ejemplo, una traducción). Usa la información del encoder + atención hacia palabras ya generadas.

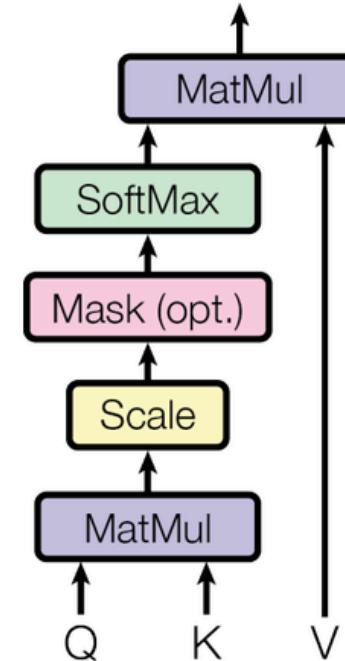


Transformers

Self-Attention

Cada palabra genera tres vectores: **Q (Query)**, **K (Key)** y **V (Value)**. El modelo calcula qué tan relevante es cada palabra respecto a las demás usando el producto QK. Los pesos resultantes se usan para combinar los valores V.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

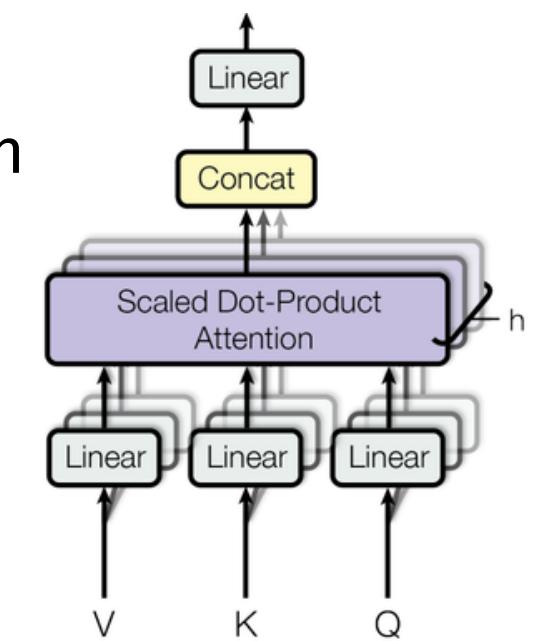


Multi-Head Attention

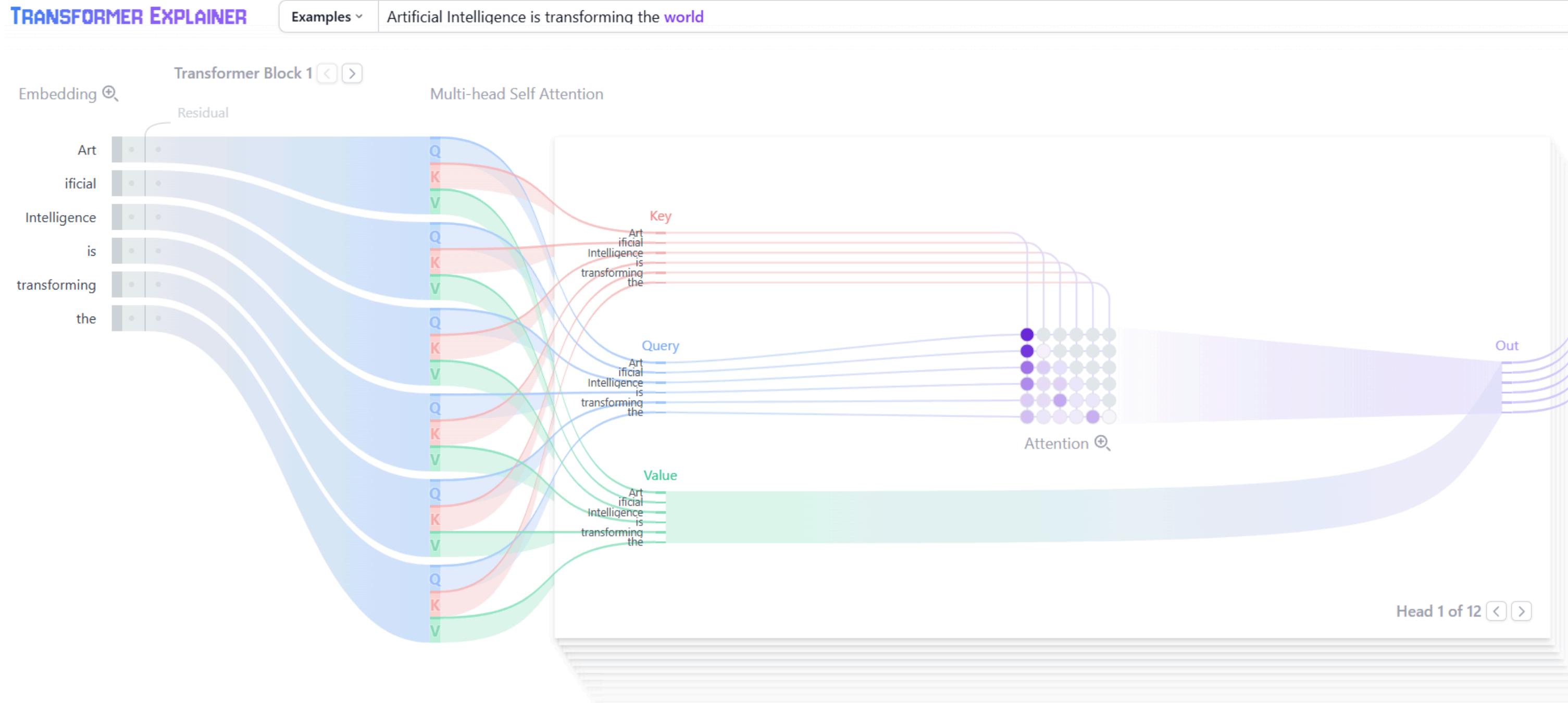
En lugar de una sola atención, el modelo usa varios “cabezales” que observan la secuencia desde diferentes perspectivas. Luego concatena sus resultados para formar una representación más rica del contexto.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Transformers



Transformer Explainer: LLM Transformer Model Visually Explained



BERT

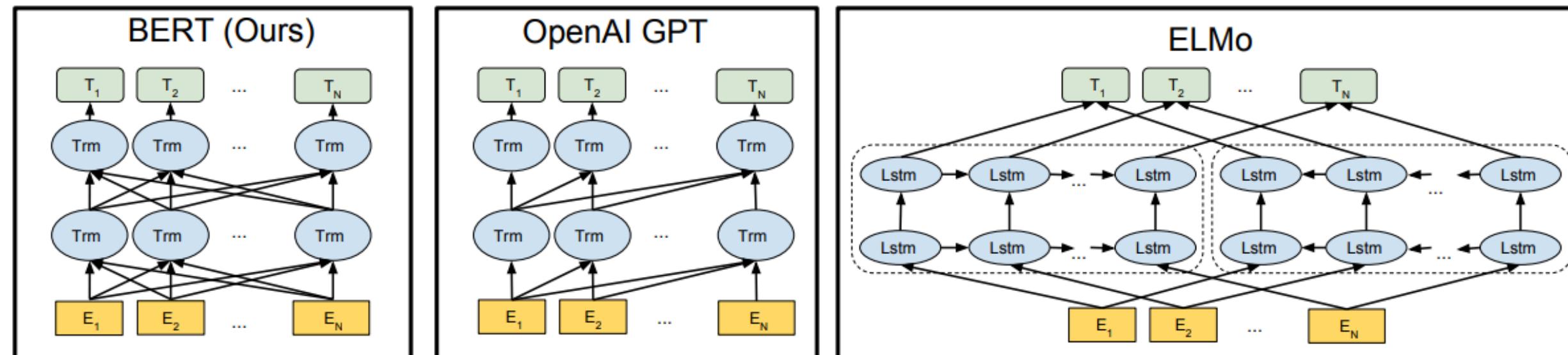
BERT

BERT (Bidirectional Encoder Representations from Transformers) fue propuesto por Devlin et al. (Google, 2018) y cambió por completo la forma de entrenar modelos de lenguaje.

Aprende embeddings del lenguaje usando solo la parte **encoder** del Transformer.

A diferencia de modelos anteriores (como GPT-1 o ELMo), BERT:

- Lee el texto **en ambas direcciones (izquierda y derecha) al mismo tiempo** → “bidireccional”.
- Captura relaciones de contexto mucho más robustas.



¿Cómo se entrena?

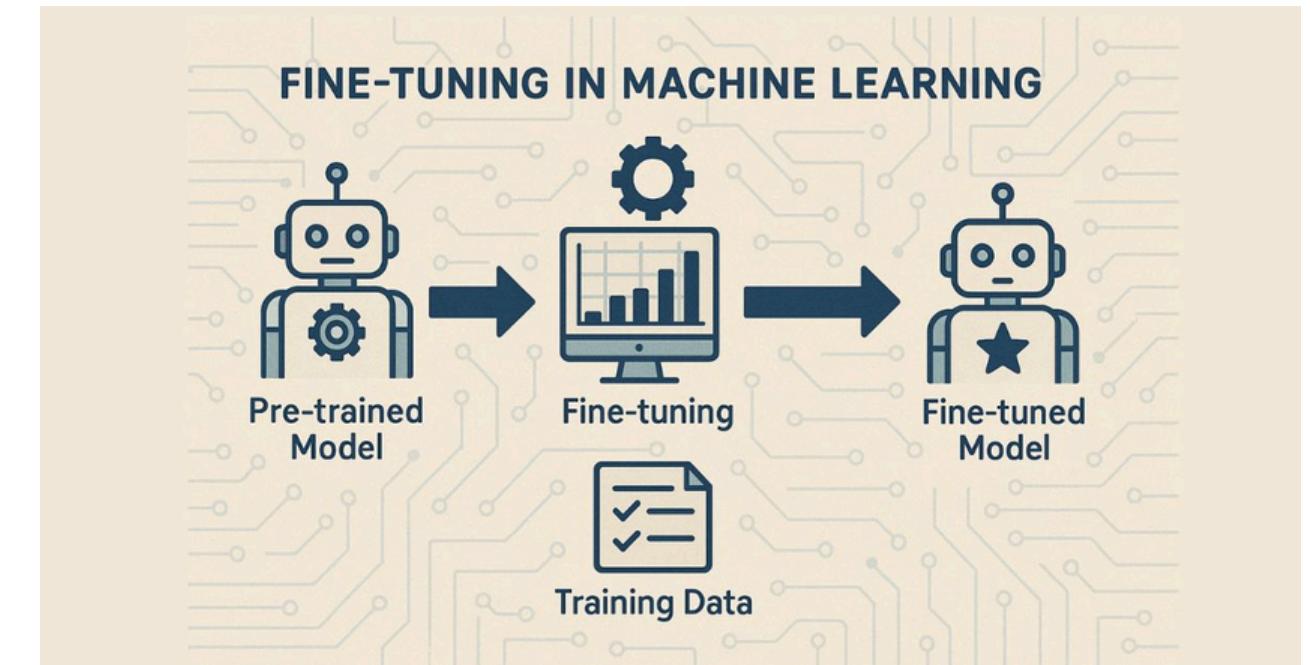
BERT no se entrena directamente para una tarea específica, sino con tareas de preentrenamiento:

1. Masked Language Modeling (MLM)

- Se ocultan algunas palabras del texto (por ejemplo, 15%)
- El modelo debe predecir las palabras faltantes.
- Ejemplo:
- “El [MASK] persiguió al ratón” → predice “gato”.

1. Next Sentence Prediction (NSP)

- Se le dan dos frases y debe predecir si la segunda sigue a la primera.
- Ayuda a aprender relaciones entre oraciones.

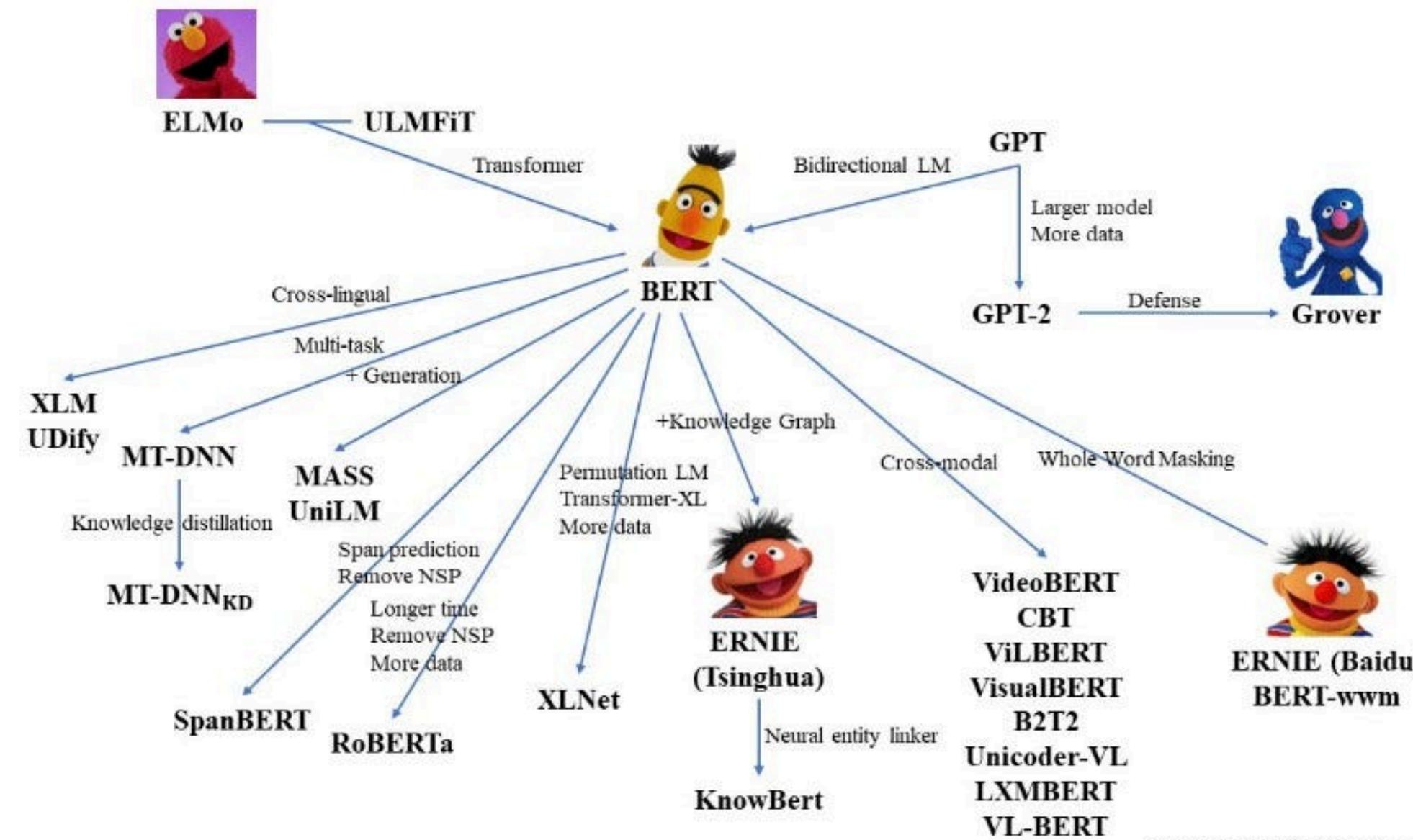


Luego, el **modelo puede ajustarse (fine-tuning) para tareas específicas**:

clasificación de texto, análisis de sentimientos, NER, QA, etc.

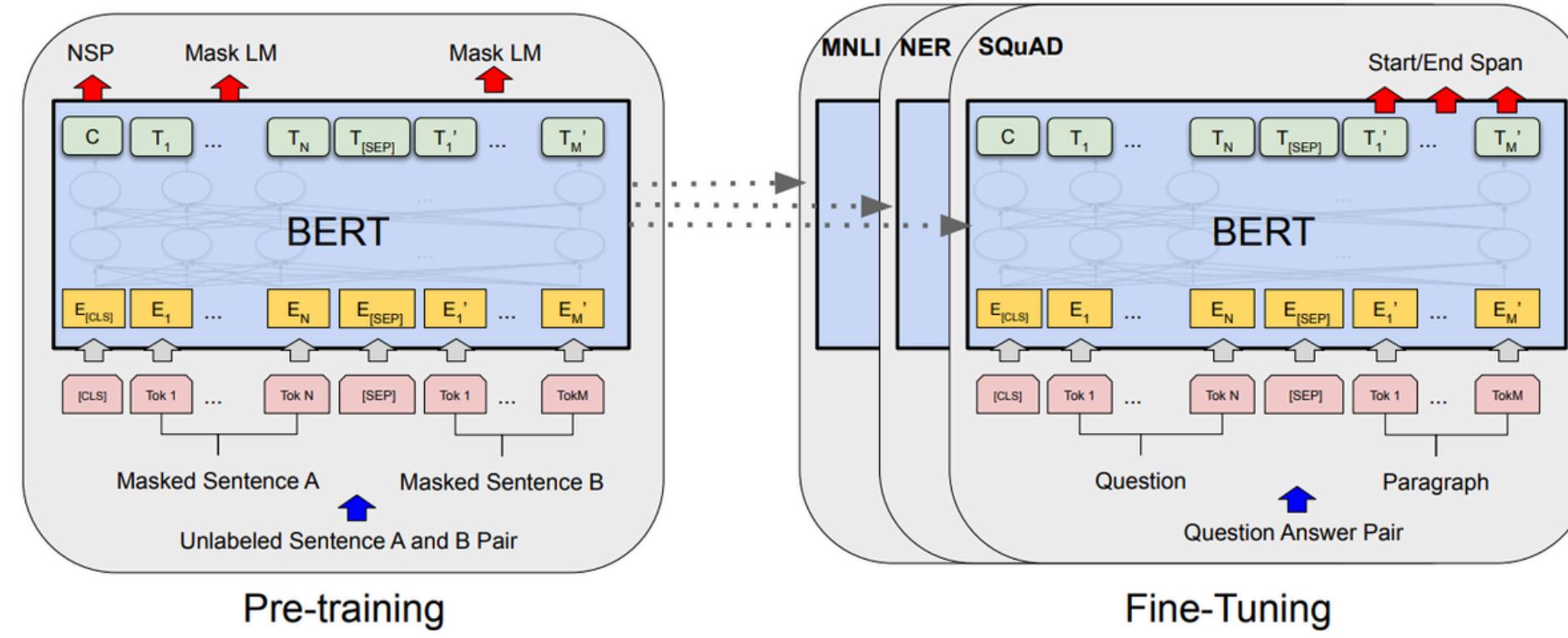
Modelos Pre-entrenados

Un modelo preentrenado es un modelo de machine learning o deep learning que ya fue entrenado previamente en una gran cantidad de datos y luego se pone a disposición para que otras personas lo usen o lo ajusten (fine-tuning) en tareas específicas.



Fine-Tuning

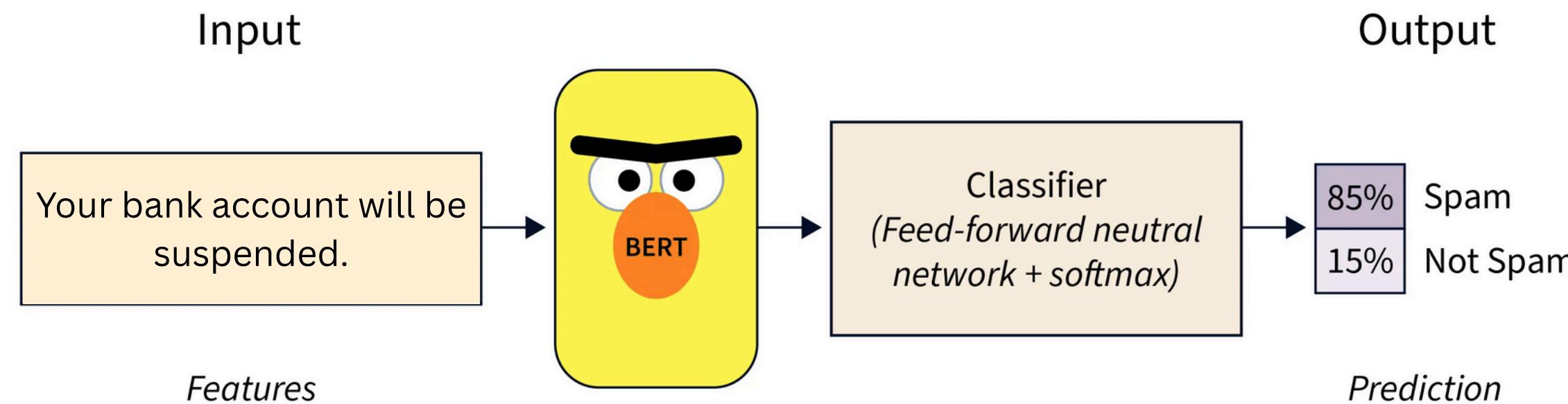
El fine-tuning es el proceso de tomar un modelo ya preentrenado (como BERT o GPT) y ajustarlo con un conjunto de datos específico para una tarea concreta.



System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Fine-Tuning

El fine-tuning (ajuste fino) es el proceso de tomar un modelo preentrenado y entrenarlo un poco más en un conjunto de datos específico, para adaptarlo a una tarea particular.



Fine-Tuning

The screenshot shows the Hugging Face Model Hub interface. At the top, there's a navigation bar with a search bar labeled "Search models, datasets, users...". Below the search bar, the "Models" tab is selected. The main content area displays a model card for "google-bert/bert-base-uncased". The card includes the repository name, a "like" count of 2.45k, a "Follow" button for the BERT community (828 members), and various framework support buttons for Fill-Mask, Transformers, PyTorch, TensorFlow, JAX, Rust, Core ML, and ONNX. A note indicates the license is apache-2.0. Below the card, there are links for "Model card", "Files and versions", and "Community" (84 members). The main text on the card describes the "BERT base model (uncased)" as a pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in [this paper](#) and first released in [this repository](#). The model is uncased, meaning it does not make a difference between "english" and "English". A disclaimer at the bottom states that the team releasing BERT did not write a model card for this model, so this one was written by the Hugging Face team.

Toxic-BERT

Fake-News-BERT

BERT-Sentiment

MINERÍA DE DATOS

Maximiliano Ojeda

muojeda@uc.cl



IIC-2433