



DataSpartan



Construyendo un Buscador Semántico con NLP

¿Qué es un buscador semántico? ¿Cómo funciona? ¿Cómo puedo construir uno?



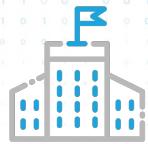
[https://github.com / rorisDS / workshop_semantic_search](https://github.com/rorisDS/workshop_semantic_search)





DataSpartan

Expertos en las áreas de Inteligencia Artificial (IA), Big Data, Investigación Cuantitativa, Computación de Alto Rendimiento (HPC), Machine Learning (ML) y Modelización de Riesgos.



PYMES



EMPRENEDORES



EQUIPOS I+D

Investigamos, diseñamos, construimos, implementamos y validamos soluciones, para facilitar que las empresas puedan alcanzar su potencial, de forma más rápida y asequible de lo que sería posible contando únicamente con sus propios medios.



Víctor Manuel Alonso Rorís



- Ingeniería de Telecomunicaciones (2003 - 2009)
- Doctor en Telemática (2017)
 - *Tecnologías Semánticas - Representación de conocimiento*
 - *34 publicaciones científicas (2011-2019)*
- Investigador en la Universidad de Vigo (9 años)
- Data Scientist en DataSpartan (2018-presente)
 - *Machine Learning Engineer – NLP specialist*

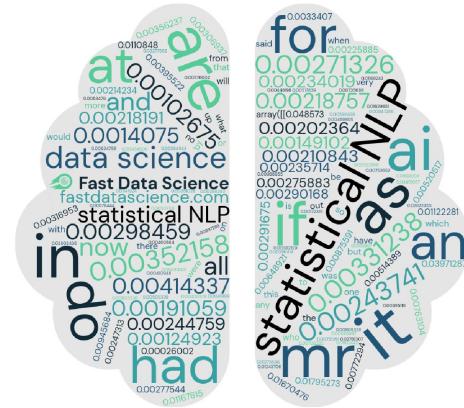
4 Workshop en el Foro de Empleo en la UVigo

- 2022-2023: *Construir un chatbot* [[link](#)]
- 2024: *LLMs* [[link](#)]
- 2025: *Construir un Buscador Semántico* [[link](#)]

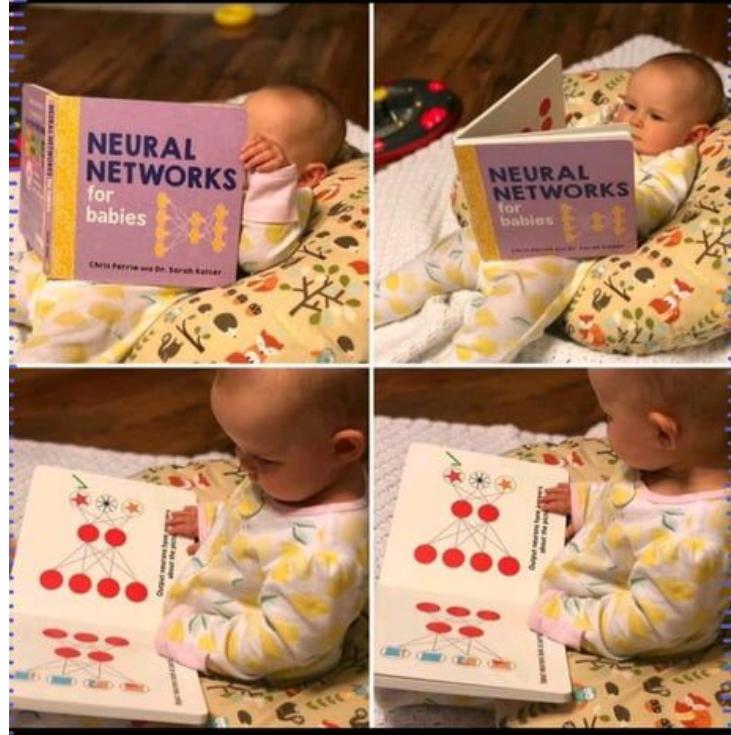


Contenidos

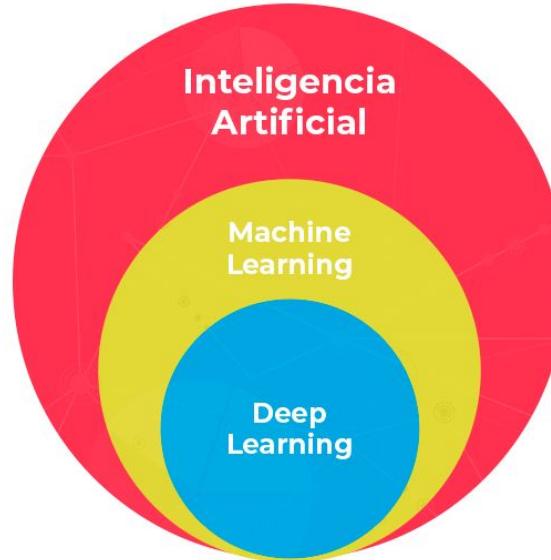
1. Conceptos básicos de Machine Learning
 2. ¿Qué es NLP?
 3. Matemáticas del lenguaje
 4. Construyendo un buscador semántico



Conceptos básicos de Machine Learning



Definiciones



IA: Combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.

Machine Learning: Rama de la Inteligencia artificial (IA) que estudia como dotar a las máquinas de capacidad de aprendizaje

Deep Learning: algoritmo automático jerárquico que emula el aprendizaje humano con el fin de obtener ciertos conocimientos.

[source](#)

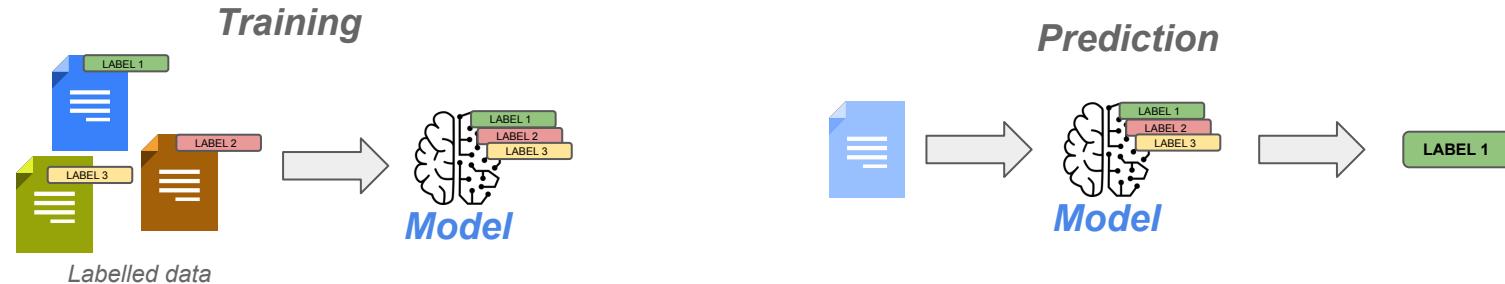


Supervised Learning

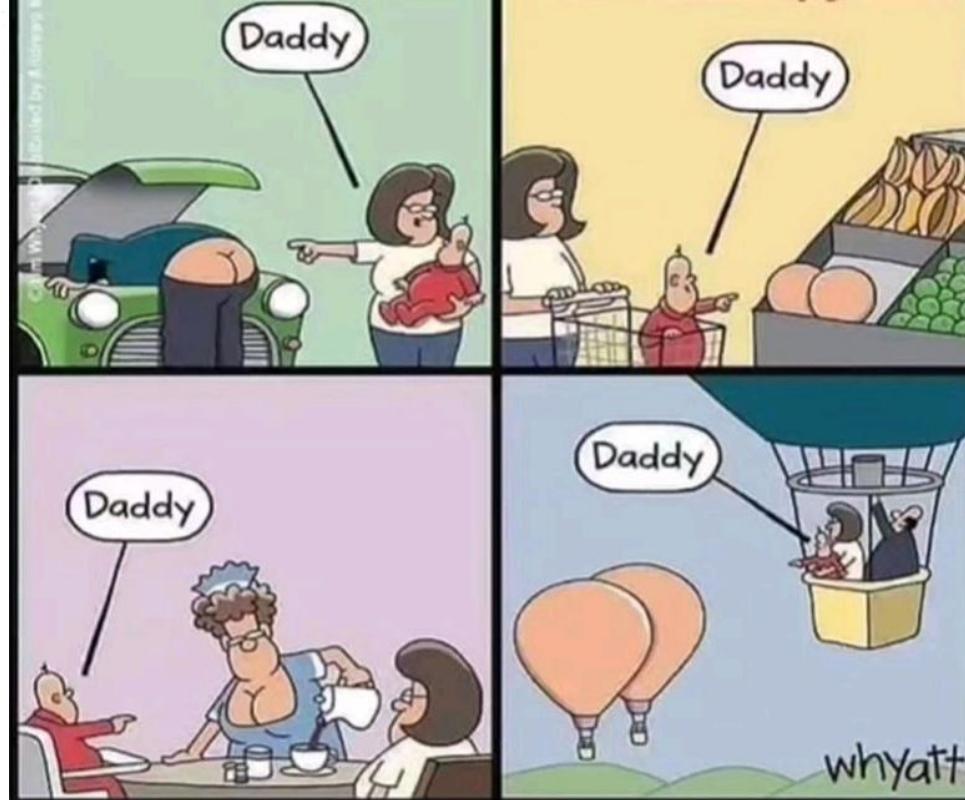
Tipo de ML que utiliza datos etiquetados para entrenar models ML.

Datos etiquetados significa que la salida esperada es conocida.

Los algoritmos miden su precisión a través del error de predicción, ajustándose hasta que el error sea suficientemente minimizado.



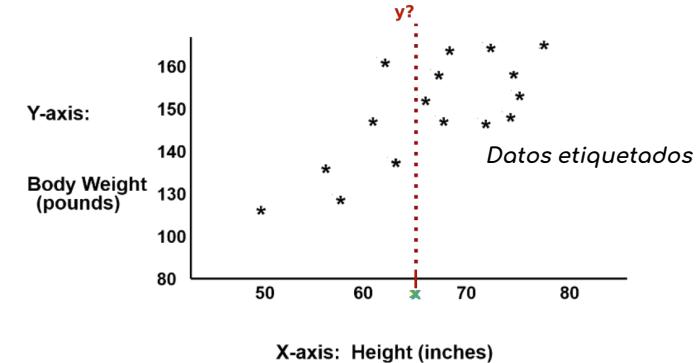
Supervised Learning



Simple Linear Regression

Predecir un simple output (y) a traves de 1-d input (x)

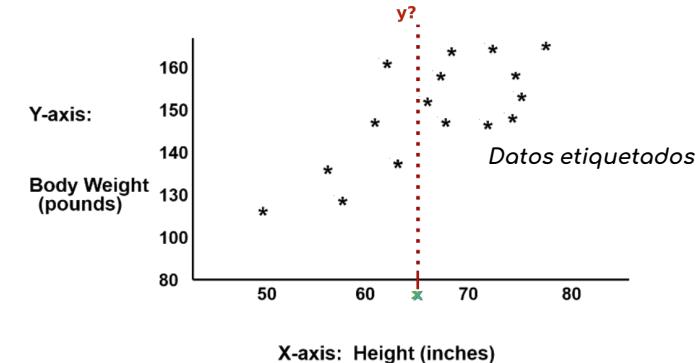
Por ejemplo, predecir el peso corporal (y) por la altura (x)



Simple Linear Regression

Predecir un simple output (y) a traves de 1-d input (x)

Por ejemplo, predecir el peso corporal (y) por la altura (x)

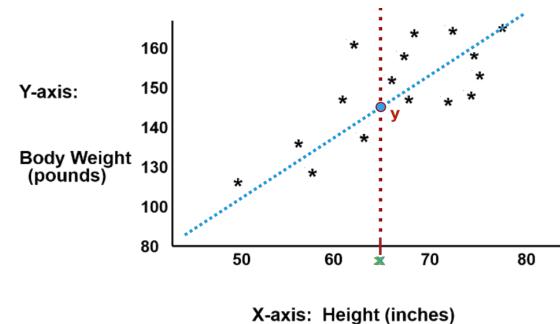


Formula de la recta:

$$h(x) = m \cdot x + b = y'$$

Donde:

- m es la pendiente
- b es el término independiente



Simple Linear Regression

1. Valor "aleatorio" para m y b
2. Introduce x en la función \Rightarrow predice y'
3. Calcula error entre y' y el y esperado
4. Ajusta m y b en base al error
 \Rightarrow Descenso del gradiente
5. Repite el proceso múltiples veces para cada ejemplo del dataset de entrenamiento

$$h(x) = m*x + b = y'$$

$$h_0(x) = 2.3*x + 1 = y'$$

$$h_1(x) = 4.1*x + 5 = y'$$

...

$$h_M(x) = 3.032*x + 2.95 = y'$$



Simple Linear Regression

1. Valor "aleatorio" para m y b
2. Introduce x en la función \Rightarrow predice y'
3. Calcula error entre y' y el y esperado
4. Ajusta m y b en base al error
 \Rightarrow Descenso del gradiente
5. Repite el proceso múltiples veces para cada ejemplo del dataset de entrenamiento

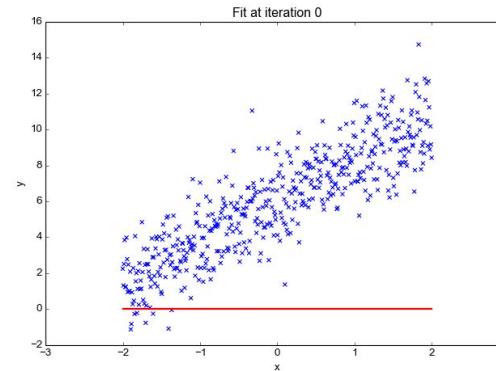
$$h(\mathbf{x}) = m^*\mathbf{x} + b = y'$$

$$h_0(x) = 2.3*x + 1 = y'$$

$$h_1(x) = 4.1*x + 5 = y'$$

...

$$h_M(x) = 3.032*x + 2.95 = y'$$



Simple Linear Regression

1. Valor "aleatorio" para m y b
2. Introduce x en la función \Rightarrow predice y'
3. Calcula error entre y' y el y esperado
4. Ajusta m y b en base al error
 \Rightarrow Descenso del gradiente
5. Repite el proceso múltiples veces para cada ejemplo del dataset de entrenamiento

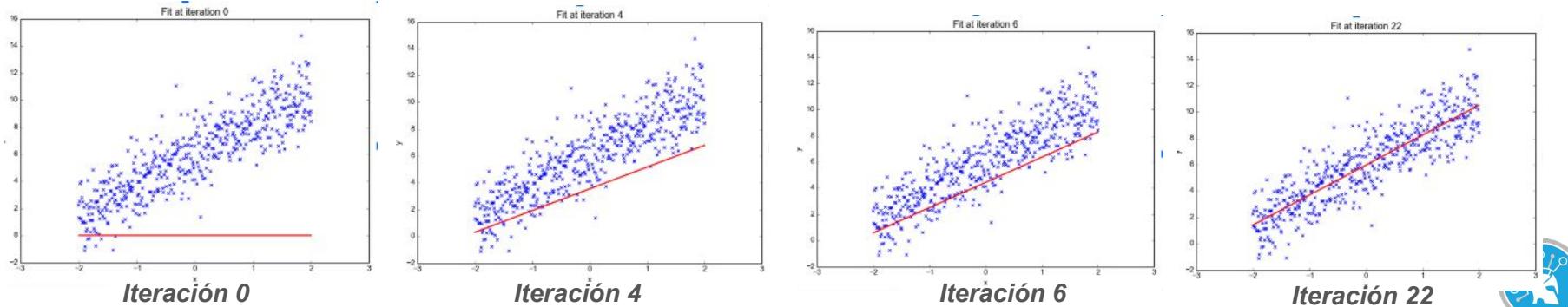
$$h(\mathbf{x}) = m^*\mathbf{x} + b = y'$$

$$h_0(x) = 2.3*x + 1 = y'$$

$$h_1(x) = 4.1*x + 5 = y'$$

...

$$h_M(x) = 3.032*x + 2.95 = y'$$



Multiple Linear Regression

Vector n-dimensional de input : $\mathbf{x} = (x^1, x^2, x^3, \dots, x^N)$

$$h(\mathbf{x}) = \theta_0 * x_0 + \theta_1 * x_1 + \dots + \theta_N * x_N + b = y'$$



Multiple Linear Regression

Vector n-dimensional de input : $\mathbf{x} = (x^1, x^2, x^3, \dots, x^N)$

Output m-dimensional : $\mathbf{y} = (y^1, y^2, y^3, \dots, y^M)$

$$h_1(\mathbf{x}) = \theta_0^1 * x_0 + \theta_1^1 * x_1 + \dots + \theta_N^1 * x_N + b^1 = y_1,$$

$$h_2(\mathbf{x}) = \theta_0^2 * x_0 + \theta_1^2 * x_1 + \dots + \theta_N^2 * x_N + b^2 = y_2,$$

$$h_3(\mathbf{x}) = \theta_0^3 * x_0 + \theta_1^3 * x_1 + \dots + \theta_N^3 * x_N + b^3 = y_3,$$

...

$$h_M(\mathbf{x}) = \theta_0^M * x_0 + \theta_1^M * x_1 + \dots + \theta_N^M * x_N + b^M = y_M,$$



Multiple Linear Regression

Vector n-dimensional de input : $\mathbf{x} = (x^1, x^2, x^3, \dots, x^N)$

Output m-dimensional : $\mathbf{y} = (y^1, y^2, y^3, \dots, y^M)$

$$h_1(\mathbf{x}) = \theta^1_0 * x_0 + \theta^1_1 * x_1 + \dots + \theta^1_N * x_N + b^1 = y_1,$$

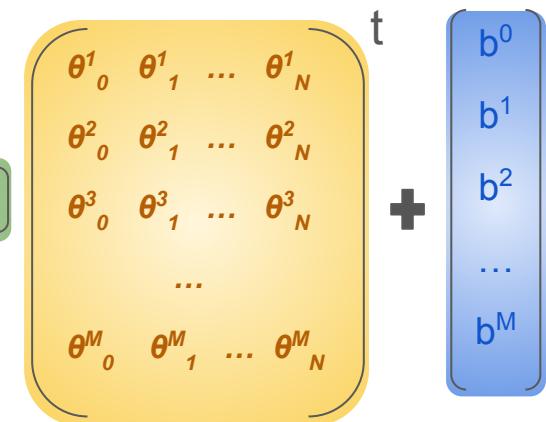
$$h_2(\mathbf{x}) = \theta^2_0 * x_0 + \theta^2_1 * x_1 + \dots + \theta^2_N * x_N + b^2 = y_2,$$

$$h_3(\mathbf{x}) = \theta^3_0 * x_0 + \theta^3_1 * x_1 + \dots + \theta^3_N * x_N + b^3 = y_3,$$

...

$$h_M(\mathbf{x}) = \theta^M_0 * x_0 + \theta^M_1 * x_1 + \dots + \theta^M_N * x_N + b^M = y_M,$$

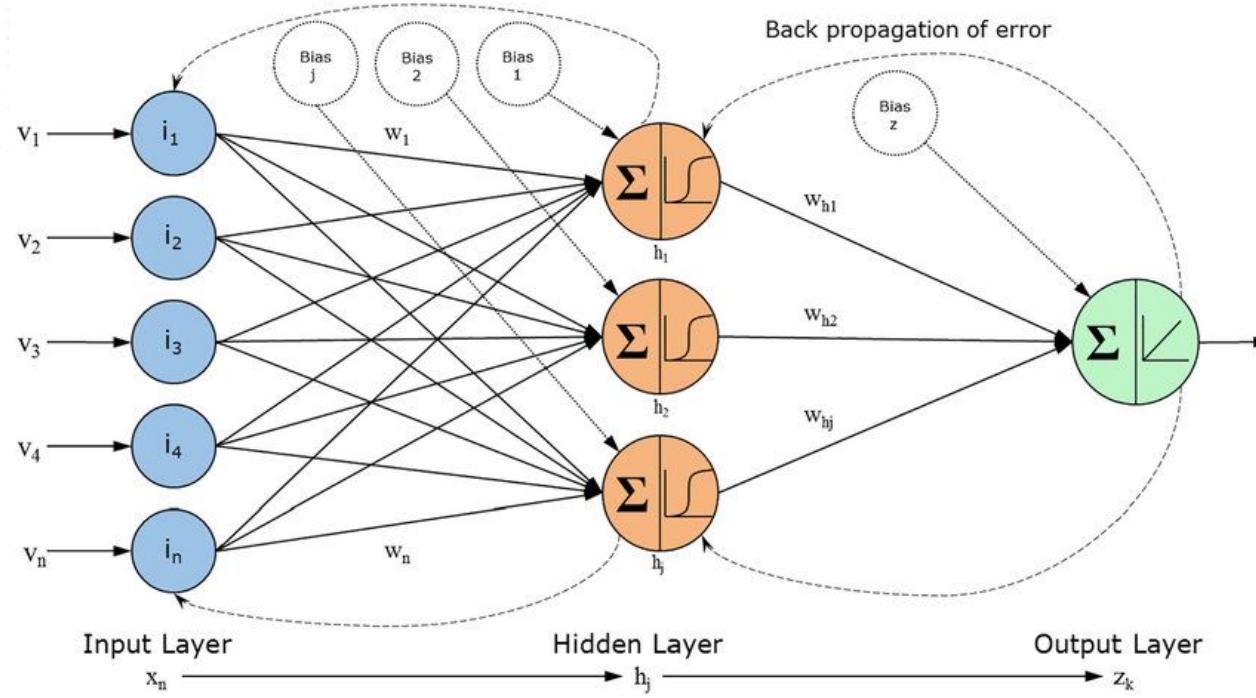
$$\left[\begin{array}{cccc} x_0 & x_1 & x_2 & \dots & x_N \end{array} \right]$$



OPERACIONES CON MATRICES



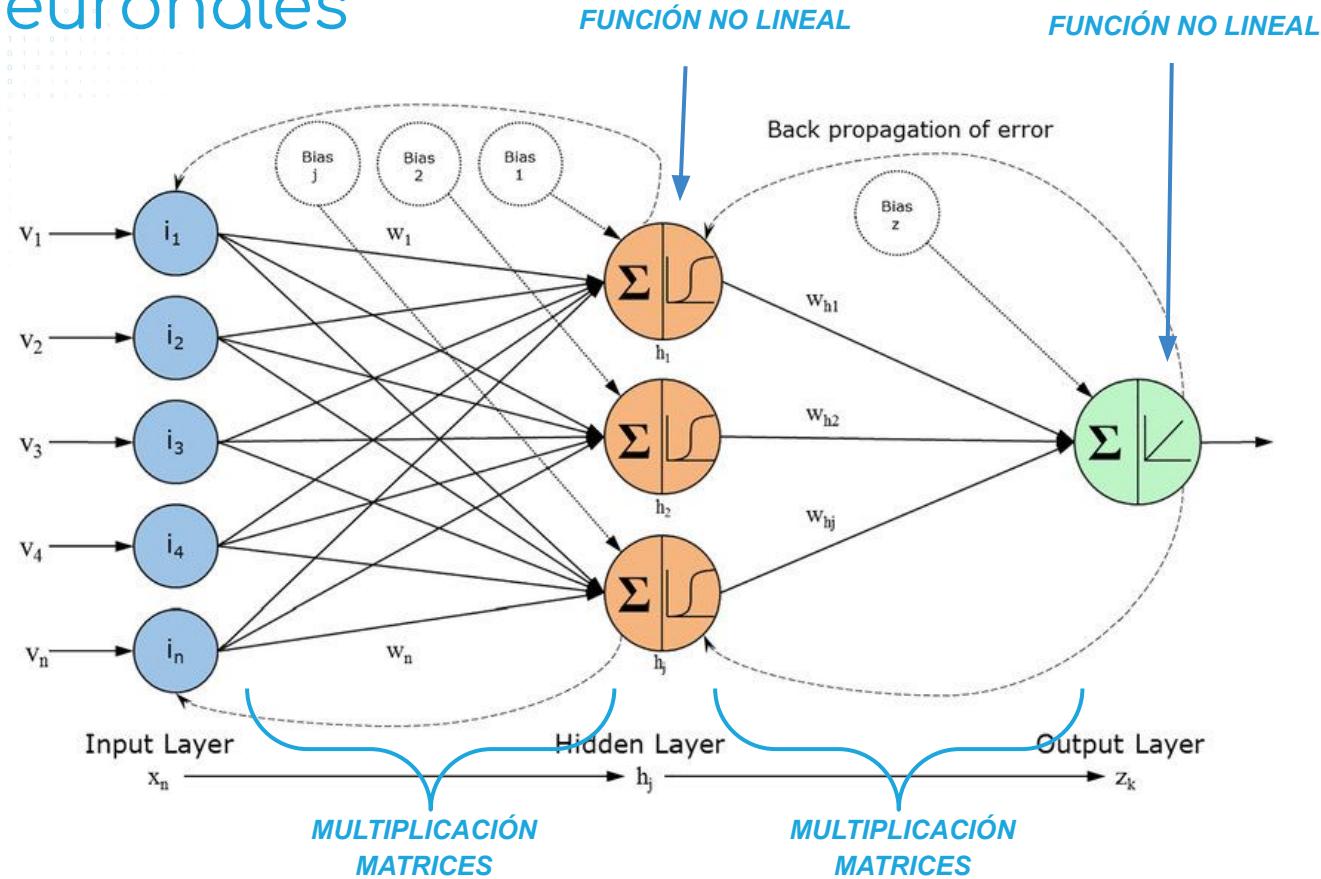
Redes Neuronales



[source](#)



Redes Neuronales



Redes Neuronales - Let's Play

Let's play

Definimos un modelo (*red neuronal*) simple usando PyTorch

```
import torch

class TinyModel(torch.nn.Module):

    def __init__(self):
        super(TinyModel, self).__init__()

        # Layer 1
        self.linear1 = torch.nn.Linear(100, 200) # Matriz de dimension 100x200
        self.activation = torch.nn.ReLU()         # Function no lineal

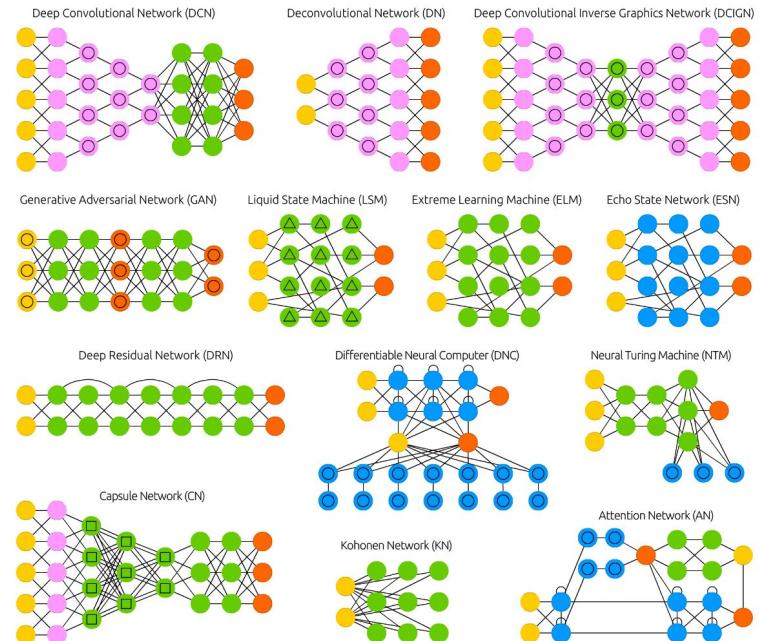
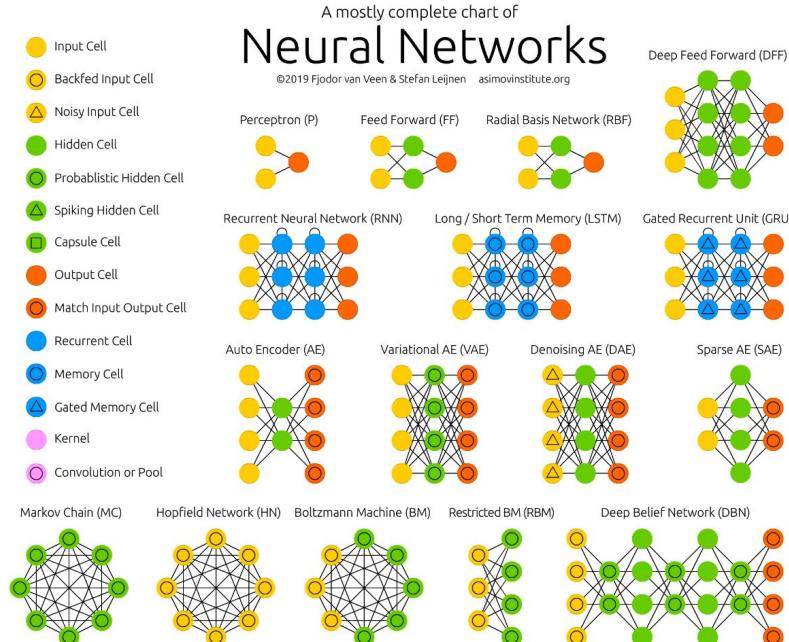
        # Layer 2
        self.linear2 = torch.nn.Linear(200, 10)   # Matriz de dimension 200x10
        self.softmax = torch.nn.Softmax()         # Function no lineal

    def forward(self, x):
        # Siendo x vector de dimension: 1x100
        x = self.linear1(x)      # Multiplicacion de matrices: 1x100 * 100x200 = 1x200
        x = self.activation(x)   # f(1x200) => 1x200
        x = self.linear2(x)      # Multiplicacion de matrices: 1x200 * 200x10 = 1x10
        x = self.softmax(x)      # f(1x10) => 1x10
        return x
```

[Notebook link](#)



Arquitecturas de Redes Neuronales



[source](#)



Conceptos Machine Learning

Operaciones matemáticas ⇒ Identificación de patrones

No hay conciencia! Lo siento Sarah Connor!

Comportamiento modelado por datos de entrenamiento

Modelos racistas o sexistas ⇒ Tus datos apestan!

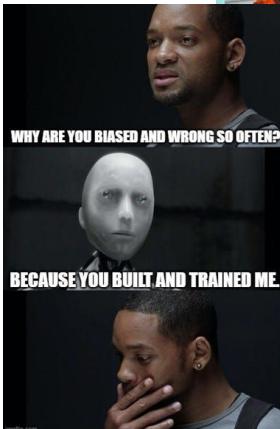
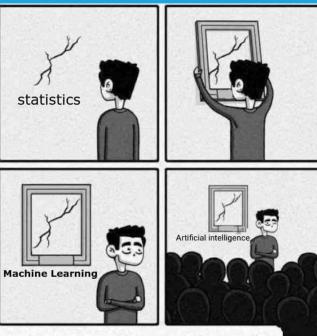
Librerías abstractas matemáticas: [Pytorch](#), [Tensorflow](#), [Keras](#), [Theano](#), ...

Gente mucho más inteligente que yo!

Quieres aprender más:

- [Curso de Coursera](#)
- [Curso de Google](#)
- [Curso de FastAI](#)

Me patiently waiting for AI to begin its war against humanity



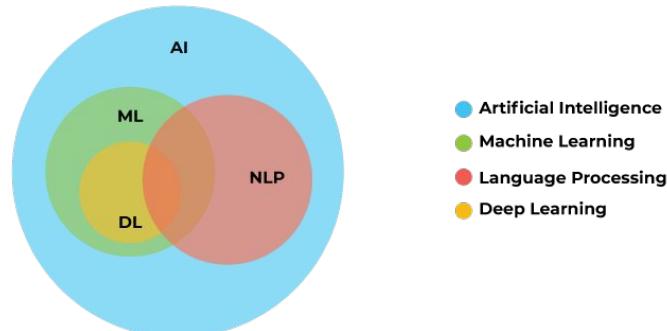
¿Qué es NLP?



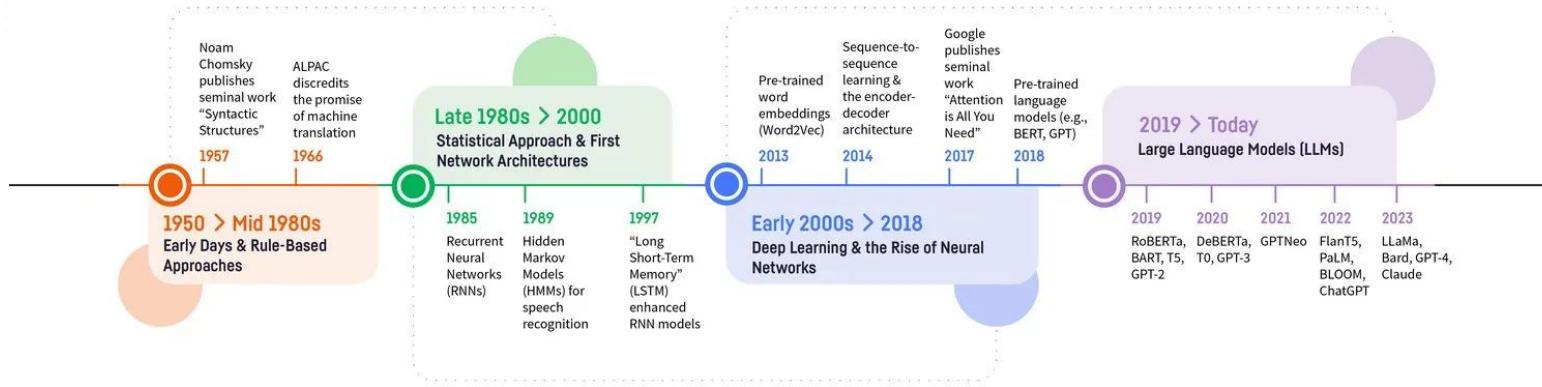
Natural Language Processing

Lenguaje Natural es la lengua o idioma hablado o escrito por seres humanos para propósitos generales de comunicación ⇒ Ambiguo y desestructurado

Procesado del Lenguaje Natural (PLN), **Natural Language Processing (NLP)** en inglés, es la rama de la IA encargada de dar a los ordenadores la capacidad de comprender textos en lenguaje natural.

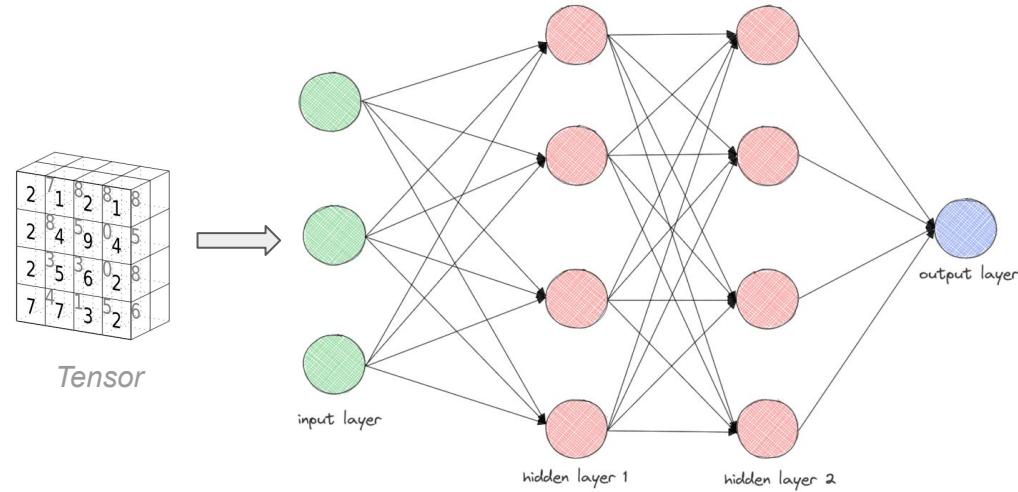


The history of NLP



Neural NLP

Los modelos ML usan tensores como input ⇒ Vectores numéricos

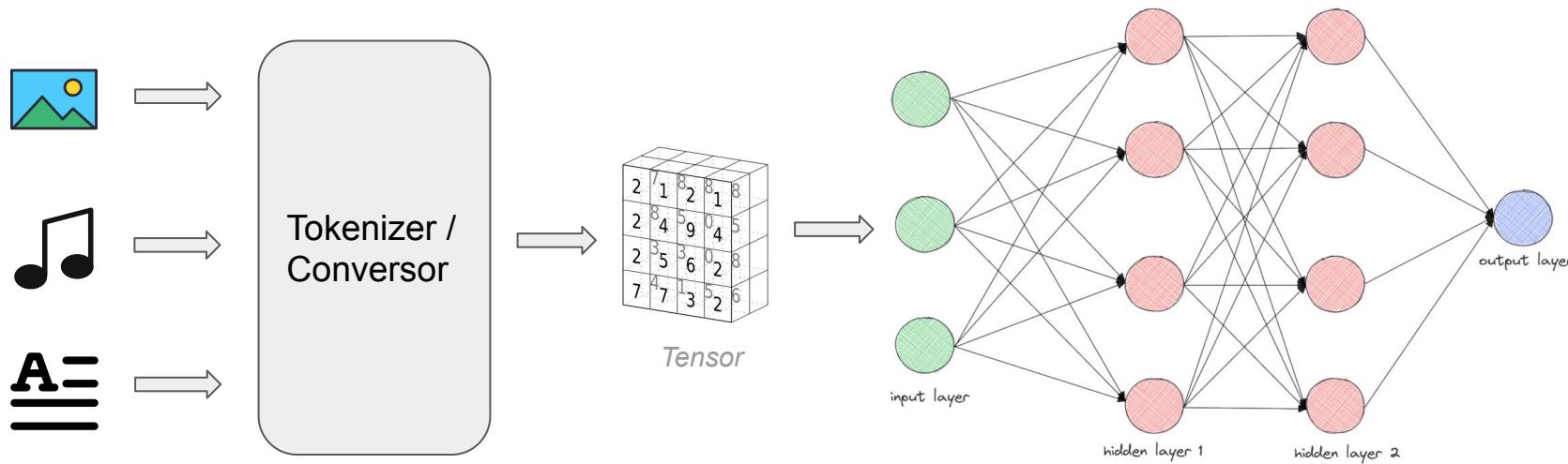


Neural NLP

Los modelos ML usan tensores como input ⇒ Vectores numéricos

¿Qué pasa con los modelos que toman como input: Imágenes? Audio? **Texto?**

⇒ Antes es necesario convertir el input a tensor (features)



Neural NLP - Let's Play

Let's play!

Seleccionamos un modelo a usar

```
[ ] # Usando el Hub de modelos de HuggingFace, seleccionamos un model  
# En este caso, un modelo basado en lenguaje español  
model_id = "dccuchile/bert-base-spanish-wwm-cased"  
  
# Siquieres, puedes ir al Hub y buscar cualquier otro modelo: https://huggingface.co/models
```

Instanciamos el Tokenizador para el modelo seleccionado

```
[ ] from transformers import AutoTokenizer  
  
tokenizer = AutoTokenizer.from_pretrained(model_id)
```

Escribimos un texto a tokenizar

```
[ ] sequence = "Aprendiendo como funciona un tokenizer"
```

Rompemos el texto en tokens

```
[ ] tokens = tokenizer.tokenize(sequence)  
  
print(tokens)
```

→ ['Apren', '##diendo', 'como', 'funciona', 'un', 'to', '##k', '##en', '##iz', '##er']

Mapeamos cada token a un id numerico (conversion directa usando el diccionario interno del Tokenizer!)

```
[ ] ids = tokenizer.convert_tokens_to_ids(tokens)  
  
print(ids)
```

→ [12340, 3052, 1184, 6023, 1049, 1166, 981, 1014, 1376, 1015]

[Notebook link](#)



Explosión de NLP

2017 - Google publica "Attention is all you need"

⇒ Se propone la arquitectura ML *Transformers*

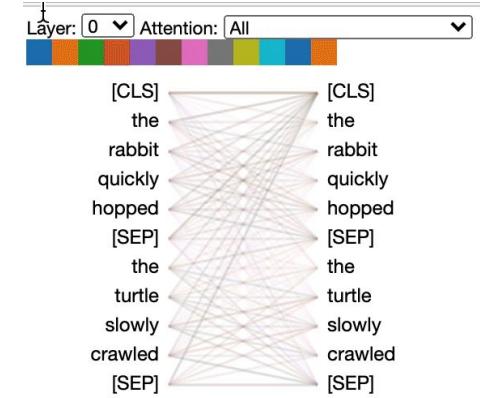
2018 - Pre-trained Language models ⇒ *Fine-tuning*

⇒ *ULMFit, Bert*

⇒ *Datasets de entrenamiento mucho más pequeños*

2022 - OpenAi anuncia chatGPT

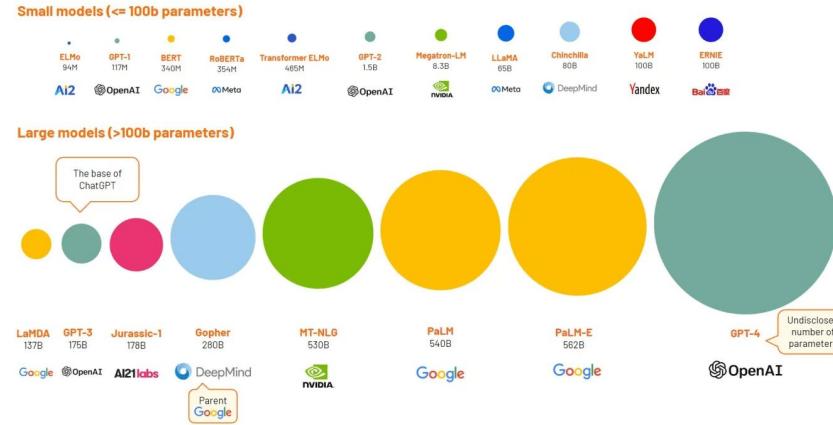
⇒ *Populariza el término LLM*



Definición LLMs

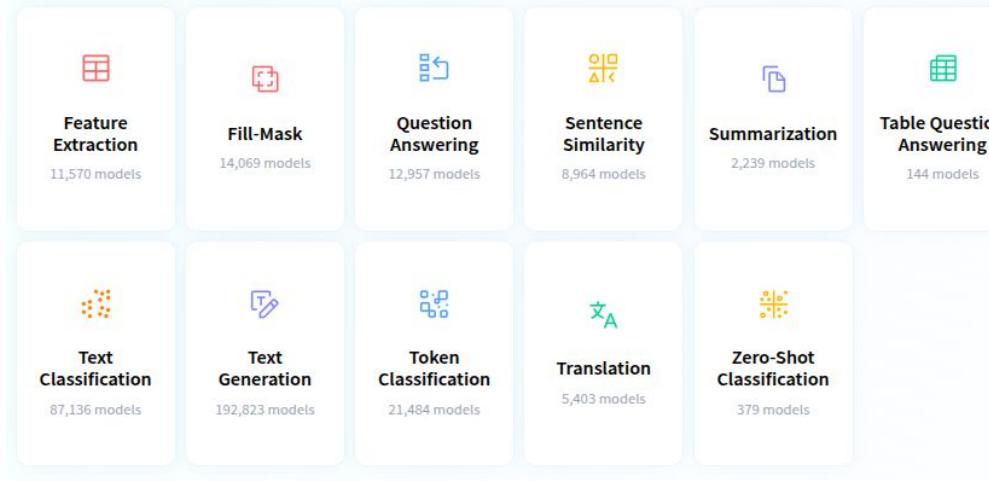
Large Language Model (**LLM**) es:

- Modelo ML para **NLP** basado en la arquitectura **Transformers**,
- Contiene **millones de parámetros** (conocidos como: m , θ , b , pesos, ...),
- Ha sido entrenado en una **cantidad ingente de textos** (corpus)
- **Propósito general** aunque pueda refinarse para dominios o tareas concretas (fine-tuning) de procesamiento de lenguaje natural.



Tareas de NLP

Natural Language Processing



<https://huggingface.co/tasks>



Tareas NLP - Let's Play

Let's play

A continuación, se presentan algunos ejemplos de código para realizar diversas tareas de Procesamiento del Lenguaje Natural (NLP) utilizando la biblioteca Transformers de Hugging Face. Para ello, se emplearán modelos disponibles en el [Hugging Face Model Hub](#), explorando distintas formas de realizar predicciones mediante modelos, tokenizers y [pipelines](#).

Traducción

En este ejemplo, vamos a utilizar el directamente `modelo` y el `tokenizer` para generar la predicción.

```
[57] from transformers import AutoModelForSeq2SeqLM
      from transformers import AutoTokenizer

      # Modelo disponible a través del hub de huggingface:
      #   https://huggingface.co/Helsinki-NLP/opus-mt-en-es
      model_key = "Helsinki-NLP/opus-mt-en-es"

      model = AutoModelForSeq2SeqLM.from_pretrained(model_key)
      tokenizer = AutoTokenizer.from_pretrained(model_key)
      model.eval()

      sentence = 'flowers are white'
      input_ids = tokenizer(sentence, return_tensors='pt')
      outputs = model.generate(**input_ids)
      # outputs = miak xochitl istak
      outputs = tokenizer.batch_decode(outputs, skip_special_tokens=True)[0]
      print(outputs)
```

☞ las flores son blancas

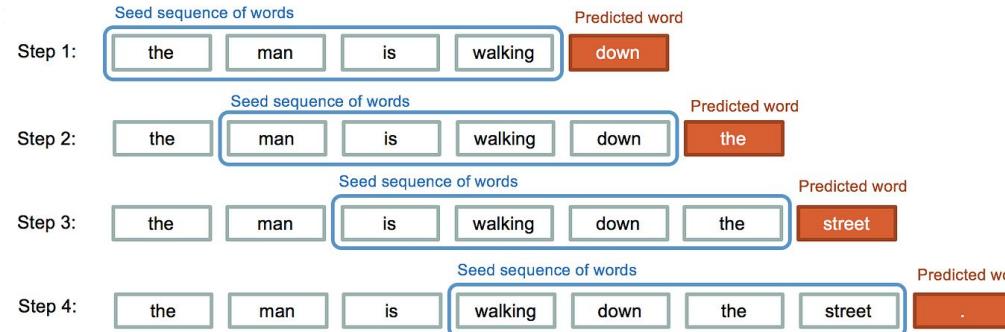
[Notebook link](#)



Tareas de NLP: Text Generation

Genera nuevo texto a partir de un texto dado

⇒ predecir la siguiente palabra más probable (o final de texto)



Los datos de entrenamiento son claves para el desempeño del modelo
⇒ No significa que copie textos!

¿Deberían estar los escritores/guionistas/periodistas preocupados? ¿de dónde se sacan los textos de entrenamiento?
¿es lícito usar textos de internet? ¿es lícito reemplazar trabajos por modelos de generación de texto?



Conceptos NLP

Explosión NLP por uso de ML

→ Explotar las relaciones semánticas del lenguaje

LLM son modelos enormes

→ Demanda recursos computacionales
Tu portatil no esta preparado!

Librerías NLP: [Transformers de HuggingFace](#), [Spacy](#), [NLTK](#), ...

Librerías LLM: [HuggingFace](#), [LangChain](#)

Gente mucho más inteligente que yo!

Quieres aprender más de NLP:

- [Curso de HuggingFace](#)
- [Curso de Stanford](#)
- [Curso de DeepLearning.ai](#)

Quieres aprender más de LLMs:

- [LLM explained](#)
- [Build a chatGPT from scratch](#)



Tell Me This 20 hours ago (edited)

Human: What do we want!?

Computer: Natural language processing!

Human: When do we want it!?

Computer: When do we want what?

Reply • 203

[View reply](#) ▾



2:19 AM - Jun 13, 2023

boredpanda.com

Matemáticas del lenguaje



Semántica del lenguaje

Semántica de cualquier idioma

⇒ Siguen reglas estadísticas

Técnica Masking

⇒ Introducir ruido en frases y entrenar el modelo para corregirlo

⇒ Dataset Auto-supervisado

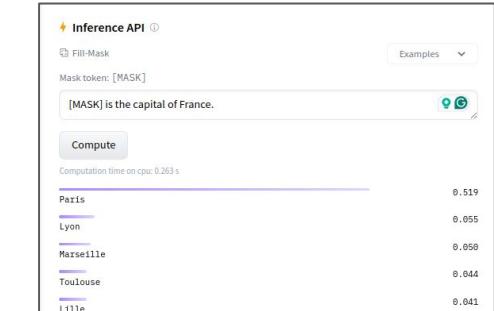
La capital de Francia es [MASK]

[MASK], capital de Francia, es conocida como la ciudad de las luces

Madrid es a España lo que [MASK] es a Francia

En mi visita a Francia, he estado en [MASK] visitando la Torre Eiffel

Las grandes capitales europeas, Madrid, Londres, [MASK], Roma y Berlín, acogen ...



¿Y si fuera un idioma extraterrestre pero dispusieras de millones de textos de ejemplo?



Embeddings

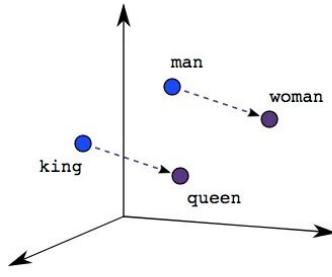
Representación numérica de palabras o frases ⇒ vectores

Word	Embedding			
ábaco	0.19	-0.32	...	0.75
abad	0.41	-1.27	...	-0.06
...			...	
zueco	-0.76	-1.09		1.35

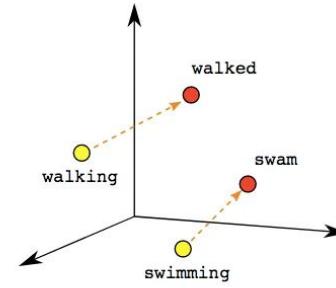
¿Tienen estos embeddings algún sentido?



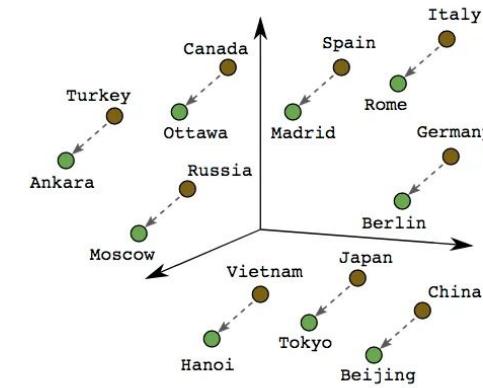
Matemáticas del lenguaje



Male-Female



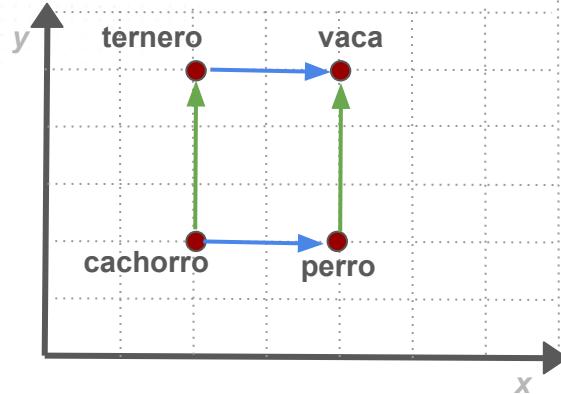
Verb Tense



Country-Capital



Matemáticas del lenguaje



Analogía

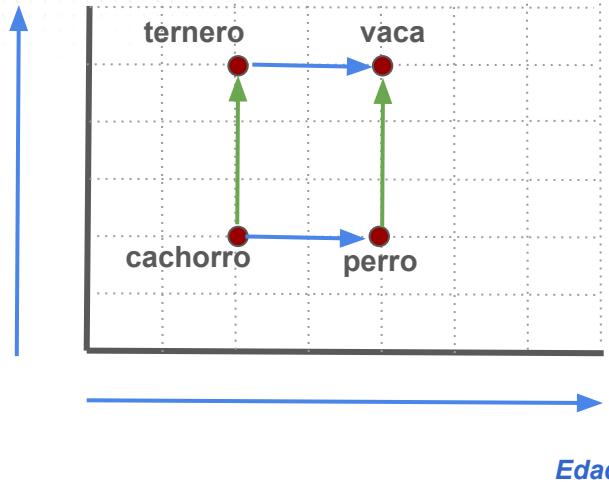
Un cachorro es a un perro lo que un ternero a una vaca

Un cachorro es a un ternero lo que un perro a una vaca



Matemáticas del lenguaje

Tamaño



Cada eje representa una propiedad

⇒ Embeddings (vector de multiples dimensiones)

Modelos ML \Rightarrow reajuste automático basado en textos de ejemplos

⇒ Propiedad de cada componente es desconocida

Dense Vectors

⇒ Información densamente empaquetada en cada dimensión

¿Dónde ubicarías una ballena? ¿Y un novillo?



Sentence-Transformers - Let's play

▼ Let's play

Comprobemos como poder obtener embeddings usando modelos de NLP.

```
[ ] # Modelo disponible a través del hub de huggingface:  
# https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2  
model_key = "sentence-transformers/paraphrase-multilingual-mpnet-base-v2"  
  
[ ] # Frases a generar los embeddings  
sentences = ["Hola", "Buenos días", "Adios"]
```

La obtención del embedding usando transformers es algo compleja, ya que implica varios cálculos sobre el vector de salida del modelo.

```
⌚ from transformers import AutoTokenizer, AutoModel  
import torch  
  
# Carga el modelo a partir del HuggingFace Hub  
tokenizer = AutoTokenizer.from_pretrained(model_key)  
model = AutoModel.from_pretrained(model_key)  
model.eval()  
  
# Tokeniza las frases  
encoded_input = tokenizer(sentences, padding=True, truncation=True, return_tensors='pt')  
  
# Calcula los embeddings de los token  
model_output = model(**encoded_input)  
  
# Mean Pooling - Take attention mask into account for correct averaging  
def mean_pooling(model_output, attention_mask):  
    token_embeddings = model_output[0] #first element of model output contains all token embeddings  
    input_mask_expanded = attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()  
    return torch.sum(token_embeddings * input_mask_expanded, 1) / torch.clamp(input_mask_expanded.sum(1), min=1e-9)  
  
# Agrupa los embeddings de los tokens para obtener el embedding de las frases  
sentence_embeddings = mean_pooling(model_output, encoded_input['attention_mask'])  
  
print("Embeddings de las frases:")  
print(sentence_embeddings)  
  
sentence_embeddings.shape
```

[Notebook link](#)



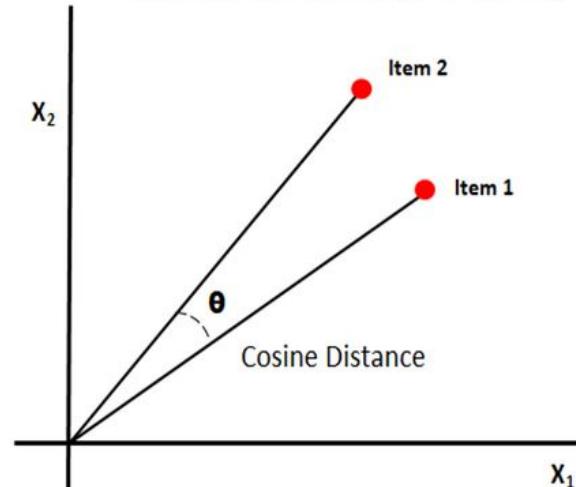
Distancia entre embeddings

Varias técnicas para el cálculo de la distancia o similitud entre puntos en el espacio

⇒ Similitud del coseno estandard NLP

- Distancia del coseno: Mide la diferencia en ángulo entre dos vectores, ignorando la magnitud.
- Similitud del coseno: Variante normalizada que devuelve valores entre -1 y 1.

Cosine Distance/Similarity



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



Distancia entre embeddings - Let's play

```
[2] from sentence_transformers import SentenceTransformer
sentences = ["Hola", "Buenos dias", "coche rojo"]

# Modelo disponible a través del hub de huggingface:
# https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
model_key = "sentence-transformers/paraphrase-multilingual-mpnet-base-v2"
model = SentenceTransformer(model_key)
sentence_embeddings = model.encode(sentences)

[3] embedding1 = sentence_embeddings[0]
embedding2 = sentence_embeddings[1]
embedding3 = sentence_embeddings[2]

Calculo de la similitud del coseno usando NumPy

[4] import numpy as np

dot_product = np.dot(embedding1, embedding2)
magnitude_1 = np.linalg.norm(embedding1)
magnitude_2 = np.linalg.norm(embedding2)

cosine_similarity = dot_product / (magnitude_1 * magnitude_2)
print(f"Similitud del coseno usando NumPy: {cosine_similarity}")

Similitud del coseno usando NumPy: 0.7888635396957397

Calculo de la similitud del coseno usando scikit-learn

[5] from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity_result = cosine_similarity(embedding1.reshape(1, -1), embedding2.reshape(1, -1))
print(f"Similitud del coseno usando scikit-learn: {cosine_similarity_result[0][0]}")

Similitud del coseno usando scikit-learn: 0.7888635396957397
```

[Notebook link](#)



Conceptos NLP y embeddings

Explosión NLP por uso de ML

⇒ Explotar las relaciones semánticas del lenguaje

Representar texto como embeddings

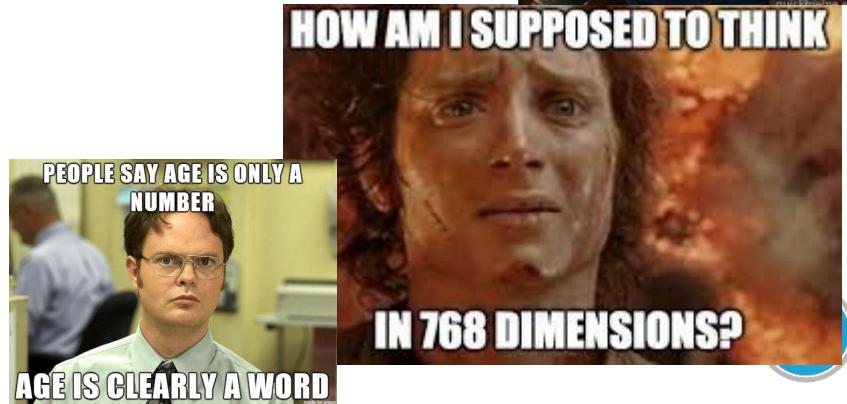
⇒ Encapsulan las relaciones semánticas del texto

Librerías NLP: [HuggingFace](#), [Sentence-Transformers](#), ...

Gente mucho más inteligente que yo!

Quieres aprender más:

- [Curso de Google](#)
- [Curso de DeepLearning.ai](#)



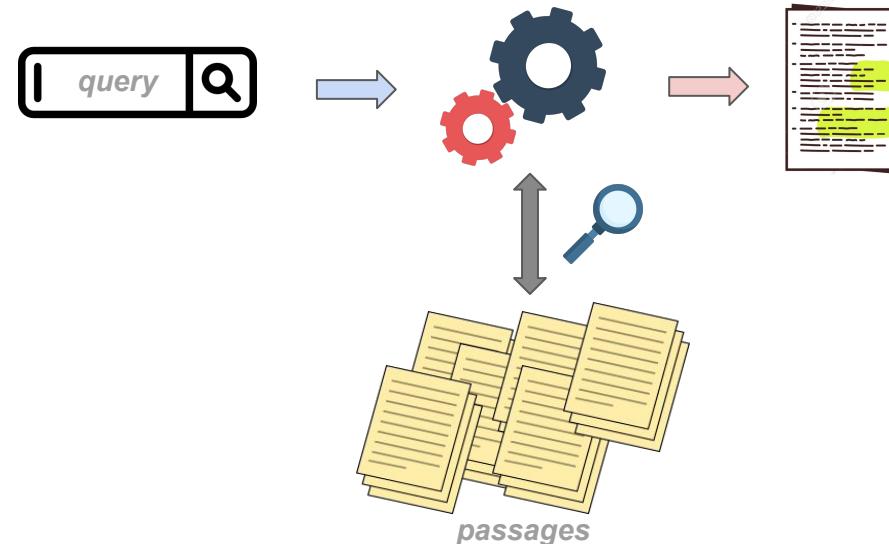
Construyendo un buscador semántico



Lógica de un buscador semántico

Buscador:

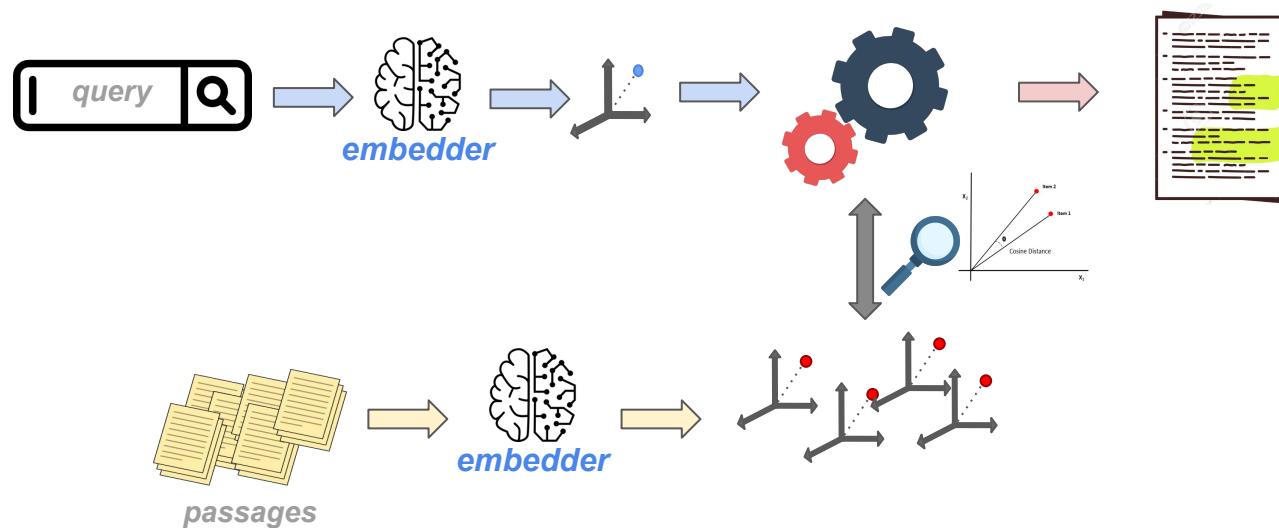
- Identificar textos (passages) que guarden relación con un texto de consulta (query)



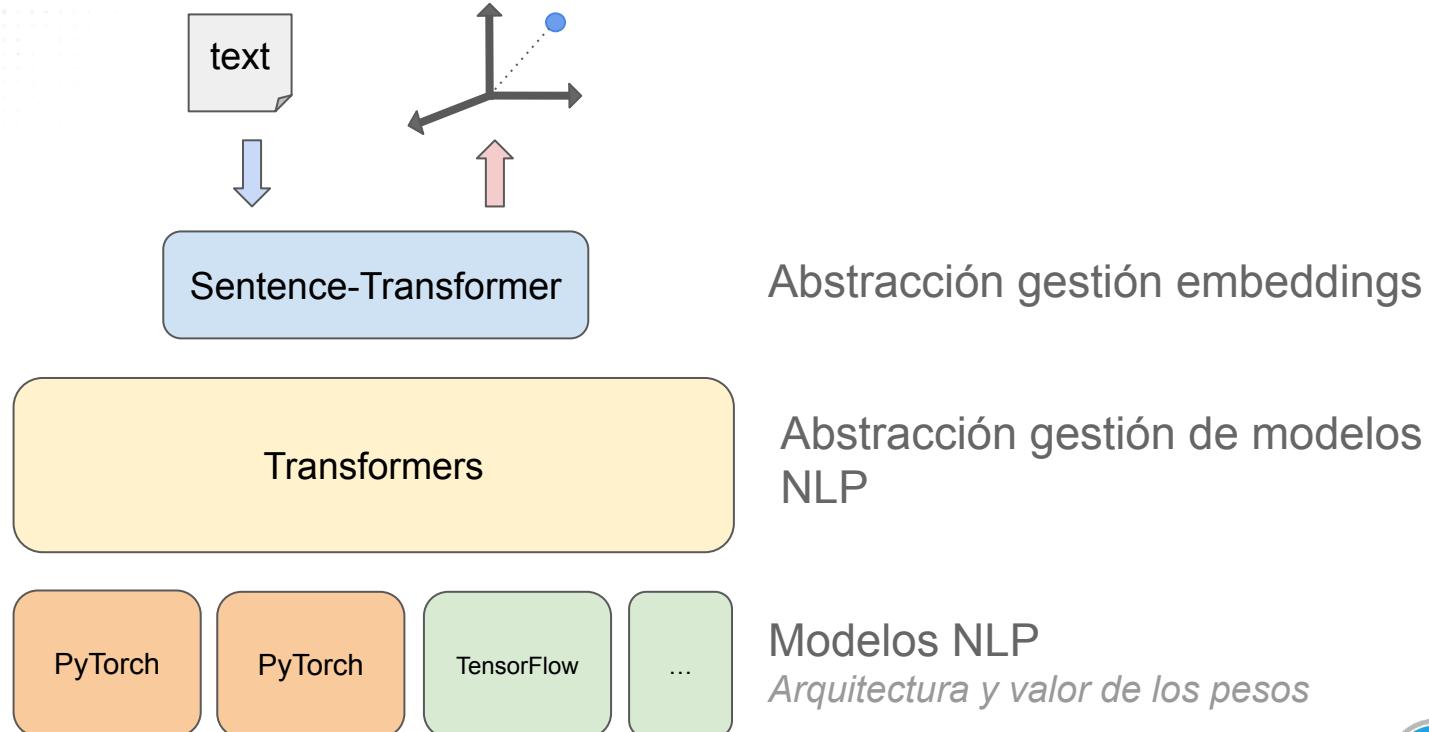
Lógica de un buscador semántico

Buscador semántico:

- Identificar embeddings similares/cercanos al embedding de consulta
 - Convertir a embedding los textos de referencia (**passages**)
 - Convertir a embedding el texto de consulta (**query**)



Stack tecnologías Embedder



Base de datos de vectores

Buscador semántico:

- ⇒ Muchos passages ⇒ Muchos embeddings
- ⇒ Gran cantidad de cálculos (distancia de embeddings) continuamente

Vector Databases:

Abstraen la indexación y búsqueda por similitud de vectores ⇒ Dense Search

Ejemplos: [Faiss](#), [Annoy](#), [NMSLIB](#), [Milvus](#), [Pinecone](#), [Zilliz](#), ...



Faiss is a library for **efficient** similarity search and clustering of dense vectors. It contains algorithms that search in sets of **vectors of any size**, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning. Faiss is written in C++ with complete wrappers for Python/numpy.



Base de datos de vectores - Let's play

```
[26] from sentence_transformers import SentenceTransformer
     from typing import List

class Embedder:

    def __init__(self, model_key: str = None):
        if model_key is None:
            # Modelo disponible a traves del hub de huggingface:
            # https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
            model_key = "sentence-transformers/paraphrase-multilingual-mpnet-base-v2"
        self.model = SentenceTransformer(model_key, trust_remote_code=True)

    def embed_passages(self, passages: List[str]):
        return self.model.encode(passages, convert_to_numpy=True)

    def embed_query(self, query: str):
        # Some models can have a different function to encode queries!
        return self.model.encode(query, convert_to_numpy=True)

embedder = Embedder()

[27] # Indexar passages
passages = [
    # Ruido
    "hola", "buenos dias", "coche rojo", "workshop", "la estufa esta encendida",
    "Pienso, luego existo",
    # Textos sobre comida
    "la comida del campus sabe a carton",
    "comer en un restaurante de estrella michelin es digno de estudio",
    "la comida de mi madre deberia estudiarse en la universidad"
]
passage_embeddings = embedder.embed_passages(passages)
```

Construimos el indice

```
import faiss

# dimension de los embeddings
d = passage_embeddings[0].shape[0]
print(f"Dimension de los embeddings: {d}")

# Construimos un indice para embeddings de dimension d
index = faiss.IndexFlatL2(d)

Dimension de los embeddings: 768
```

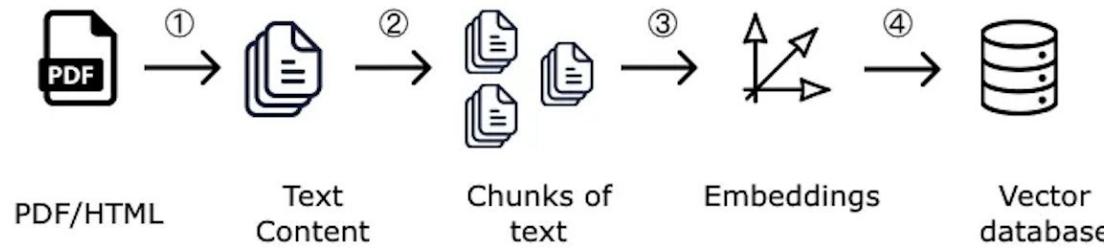
[Notebook link](#)



Lógica de negocio - Indexación

Fase de ingestión (ETL):

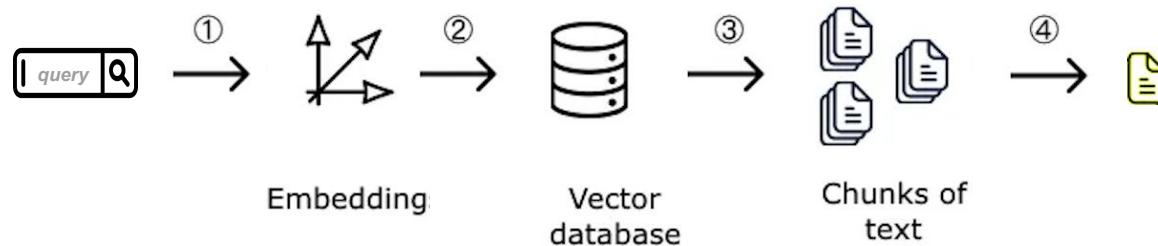
- 1) Documentos a indexar ⇒ extraer texto
- 2) Chunking ⇒ divide textos grandes en fragmentos (passages)
- 3) Convierte cada passage a embedding
- 4) Indexa los embeddings en una base de datos de vectores



Lógica de negocio - Búsqueda

Fase de búsqueda:

- 1) Convertir consulta (query) a embedding
- 2) Busca por embedding en la base de datos de vectores
- 3) Identifica los passages más similares a la query



Lógica de negocio - Let's play

```
✓ [31] import requests
     import fitz # PyMuPDF
     import os
     import faiss
     import numpy as np
     from sentence_transformers import SentenceTransformer

def download_pdf(url, output_path: str):
    """Descarga un PDF de una url"""
    response = requests.get(url)
    with open(output_path, "wb") as f:
        f.write(response.content)
    print(f"PDF descargado: {output_path}")

def pdf_to_text(pdf_path: str) -> str:
    """Convierte el contenido de un PDF a texto"""
    doc = fitz.open(pdf_path)
    text = "\n".join([page.get_text("text") for page in doc])
    return text

def split_text(text: str, chunk_size: int=200) -> list:
    """Rompe el texto en trozos"""
    words = text.split()
    return [" ".join(words[i:i+chunk_size]) for i in range(0, len(words), chunk_size)]

def create_embeddings(passages: list, model_name="sentence-transformers/all-MiniLM-L6-v2"):
    """Convierte textos a embeddings"""
    model = SentenceTransformer(model_name)
    embeddings = model.encode(passages, convert_to_numpy=True)
    return embeddings

def store_embeddings(embeddings, index_path="faiss_index.bin"):
    """Indexa embeddings en Faiss"""
    dimension = embeddings.shape[1]
    index = faiss.IndexFlatL2(dimension)
    index.add(embeddings)
    faiss.write_index(index, index_path)
    print(f"Embeddings guardados en {index_path}")

# 0. Obtengo los documentos a procesar
pdf_url = "https://dl.awsstatic.com/aws-analytics-content/OReilly_book_Natural-Language-and-Search_web.pdf" # URL de ejemplo
pdf_path = "documento.pdf"
download_pdf(pdf_url, pdf_path)

# 1. Extraccion de texto
text = pdf_to_text(pdf_path)

# 2. Division en fragmentos
passages = split_text(text)

# 3. Genera embeddings
embeddings = create_embeddings(passages)

# 4. Guarda en la base de datos de vectores
index_path = "faiss_index.bin"
store_embeddings(embeddings, index_path)
print("Indexación completada.")
```

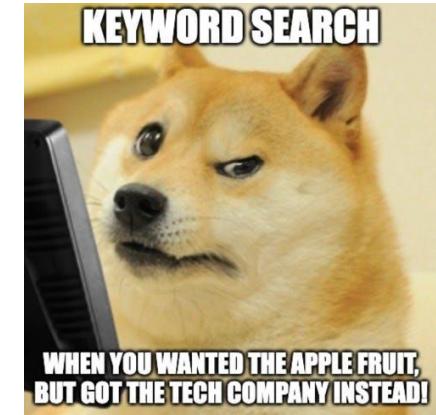
[Notebook link](#)



Ventajas

Ventajas con respecto a los buscadores tradicionales:

- Comprensión del Significado y Contexto:
 - No buscan por términos
 - Permite sinónimos y relaciones semánticas
(buscar AI y recibir documentos sobre ML)
- Búsqueda en Lenguaje Natural
- Escalabilidad y Eficiencia con Bases de Datos de Vectores



WHEN YOU SEARCH UP "VECTOR"
AND FIND TONS OF IMAGES OF A CHARACTER
FROM A RANDOM MOVIE, INSTEAD OF MATH:



Aplicaciones

Aplicaciones en múltiples industrias, incluyendo:

- **Salud:** Encontrar estudios médicos y artículos relevantes para diagnósticos.
- **Educación:** Recuperar materiales de aprendizaje basados en conceptos clave.
- **Legal:** Buscar documentos jurídicos sin necesidad de coincidencia exacta de términos.
- **Empresas:** Mejorar la búsqueda en bases de conocimiento y soporte al cliente.

Integración con IA y Sistemas Avanzados

⇒ RAG (Retrieval Augmented Generation)



RAG

RETRIEVAL-AUGMENTED GENERATION O RAG

Hay 3 fases:

1 RETRIEVAL
RECUPERACIÓN

2 AUMENTED
AUMENTO

3 GENERATION
GENERACIÓN

El proceso es el siguiente:



Fuente: Adaptación de <https://www.ml6.eu/blogpost/leveraging-langs-on-your-domain-specific-knowledge-base>



RAG - Let's play

Let's play!

- OpenAI API KEY

Indica tu OpenAI API Key para poder usar la API de OpenAI (cómo obtener la API key: <https://platform.openai.com/docs/quickstart>)

```
[35] # Indica la API Key de OpenAI
openai_api_key = "TU_API_KEY_DE_OPENAI"

# --- Este proceso es para mí para no compartir mi token con todo el mundo mientras hago el workshop!!!
from google.colab import userdata
try:
    secret_openai_api_key = userdata.get('OPENAI_API_KEY')
except userdata.SecretNotFoundError:
    secret_openai_api_key = None # Assign None if secret is not found

if secret_openai_api_key is not None:
    openai_api_key = secret_openai_api_key

import os
# LangChain toma el API KEY de las variables de entorno
os.environ['OPENAI_API_KEY'] = openai_api_key
```

- Sin RAG

Lanzamos una consulta a OpenAI sin contexto

```
[55] from langchain.chat_models import ChatOpenAI

chat = ChatOpenAI(
    model="gpt-3.5-turbo",
    temperature=0.0
)

# Prueba 1: Pregunta sobre un documento técnico
# query = "¿Cuál es la cuenta y password por defecto del router Livebox Fibra? Di 'No lo sé' si no sabes la respuesta" # Siquieres forzar que no se invente la respuesta
query = "¿Cuál es la cuenta y password por defecto del router Livebox Fibra?"
response = chat.predict(query)
print("Pregunta:", query)
print("Respuesta del modelo:", response)
print("\n-----\n")
```

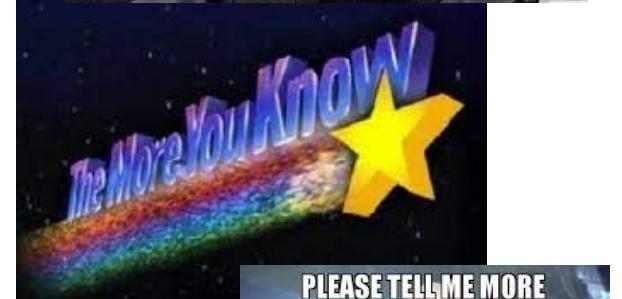
[Notebook link](#)

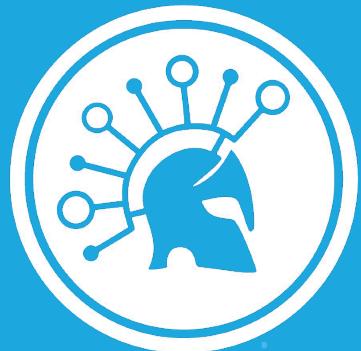


Saber más

Quieres aprender más?

- [Embeddings, Models, Vector Databases and RAG](#)
- [LLM Embeddings Explained: A Visual and Intuitive Guide](#)
- [What are Vector Embeddings? An Intuitive Explanation](#)
- [Introduction to Embeddings & Vector Stores](#)
- [Embeddings & Vector Stores: A Beginner's Guide](#)
- [RAG with LangChain](#)
- [Vector Embeddings in RAG Applications](#)





DataSpartan

Rétanos hoy

informacion@dataspartan.com

www.es.dataspartan.com