UNIVERSIDAD DE ALMERÍA

Introducción al uso de la Inteligencia Artificial Generativa

Octubre, 2024

Marcos Lupión Lorente

Índice

1 Herramientas

UNIVERSIDAD DE ALMERÍA

Librerías y herramientas

- **Ciencia de Datos**
  - NumPy: Funciones matemáticas y gestión de datos multidimensionales.
  - Pandas: Manipulación y análisis de datos.
  - Matplotlib: Gráficos y visualizaciones.
  - OpenCV: Visión por computadora. Herramientas y algoritmos para imágenes y videos.
- **Aprendizaje Automático**
  - Scikit-learn: Biblioteca para ML en Python. Proporciona una variedad de algoritmos de aprendizaje supervisado y no supervisado, junto con herramientas para el preprocesamiento de datos, selección de modelos y evaluación.
- **Aprendizaje Profundo**
  - TensorFlow: Construcción y entrenamiento de redes neuronales.
  - PyTorch: Construcción y entrenamiento de redes neuronales.
  - Keras: API de alto nivel que facilita la construcción y el entrenamiento de redes neuronales.
- **Google Colaborate**: Construcción y entrenamiento de modelos de ML en la nube. Plan gratuito. Recursos de GPU.
- **Google AutoML**: Construcción, entrenamiento y despliegue de modelos de ML en la nube. Ayuda a los usuarios a desarrollar modelos de alto rendimiento con los conjuntos de datos proporcionados.

transformers

Hugging Face

- Enlace:
  - https://huggingface.co/docs/transformers/index
- **Pipeline**: Usar para la inferencia
  - https://huggingface.co/docs/transformers/main_classes/pipelines
  - https://huggingface.co/docs/transformers/pipeline_tutorial
  - https://huggingface.co/docs/transformers/task_summary
  - Hay mútliples tareas predefinidas.
    - Parámetros de entrada/salida pre-definidos.
  - Se puede seleccionar un modelo (que hará una tarea específica):
    - https://huggingface.co/docs/transformers/model_summary

transformers

Hugging Face

- Enlace:
  - https://huggingface.co/docs/transformers/index
- **Pipeline**: Usar para la inferencia
  - https://huggingface.co/docs/transformers/main_classes/pipelines
  - https://huggingface.co/docs/transformers/pipeline_tutorial
  - https://huggingface.co/docs/transformers/task_summary
  - Hay mútliples tareas predefinidas.
    - Parámetros de entrada/salida pre-definidos.
  - Se puede seleccionar un modelo (que hará una tarea específica):
    - https://huggingface.co/docs/transformers/model_summary

- ¡Manos a la obra!

- Acceder a Google Colaboratory
- Importar un Notebook

- *CursoGenAI_UAL_00_Transformers.ipynb*

- **Probar diferentes modelos pre-entrenados**

Task	Description	Modality	Pipeline identifier
Text classification	assign a label to a given sequence of text	NLP	pipeline(task="sentiment-analysis")
Text generation	generate text given a prompt	NLP	pipeline(task="text-generation")
Summarization	generate a summary of a sequence of text or document	NLP	pipeline(task="summarization")
Image classification	assign a label to an image	Computer vision	pipeline(task="image-classification")
Image segmentation	assign a label to each individual pixel of an image (supports semantic, panoptic, and instance segmentation)	Computer vision	pipeline(task="image-segmentation")
Object detection	predict the bounding boxes and classes of objects in an image	Computer vision	pipeline(task="object-detection")
Audio classification	assign a label to some audio data	Audio	pipeline(task="audio-classification")
Automatic speech recognition	transcribe speech into text	Audio	pipeline(task="automatic-speech-recognition")
Visual question answering	answer a question about the image, given an image and a question	Multimodal	pipeline(task="vqa")
Document question answering	answer a question about the document, given a document and a question	Multimodal	pipeline(task="document-question-answering")
Image captioning	generate a caption for a given image	Multimodal	pipeline(task="image-to-text")

transformers

Hugging Face

- **Auto class**: Clases para cargar y hacer inferencia sobre los modelos
  - AutoModel
    - Carga modelos ya sean pre-entrenados o sin entrenar.
    - https://huggingface.co/models
  - AutoTokenizer
    - Carga el tokenizer adecuado para el modelo cargado anteriormente. Normalmente tienen el mismo nombre que los modelos.
  - AutoConfig
    - Cuando se carga un modelo, pero se quiere modificar su arquitectura.
- Si se quieren cargar modelos específicos para una tarea dada, estos sobre-escriben:
  - Clasificación: AutoModelForSequenceClassification

transformers

Hugging Face

- **Trainer**: Clase para entrenar modelos sobre datasets.
  - Trainer:
    - Clase para realizar bucles de entrenamiento de un modelo sobre un dataset dado.
  - TrainingArguments:
    - Clase para configurar el bucle de entrenamiento.

- Para el caso de modelos de **secuencia a secuencia**, se tiene:
  - Seq2SeqTrainingArguments y Seq2SeqTrainer

datasets

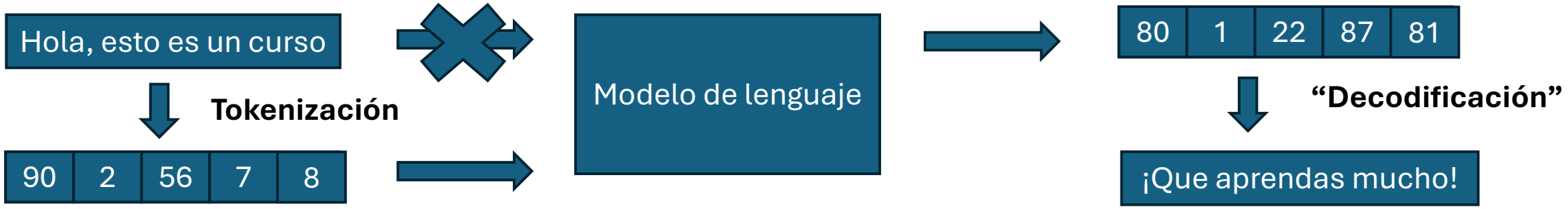
Hugging Face

- https://huggingface.co/docs/datasets/index
- Cargar un dataset del “**hub**”. Hay muchos datasets.
  - https://huggingface.co/datasets
- También se pueden cargar datasets de archivos **propios**.
  - https://huggingface.co/docs/datasets/loading
  - CSV, JSON, txt, …
- **Data collators**: Procesar el dataset, generar los batches de entrada, generar mascaras, etc.

2 Tokenización

Tokenización
Concepto

- Los modelos de aprendizaje automático están **basados en números**.
  - Modelos de **machine learning** tradicionales tienen como entradas atributos **numéricos**.
    - Los atributos de **texto** se **transforman** en números (one-hot y label encoding).
  - Redes neuronales **convolucionales (imágenes)** codifican las imágenes en valores entre **0-1** (o 0-255).
- **¿Cómo se codifica el texto** para que lo entienda el modelo?

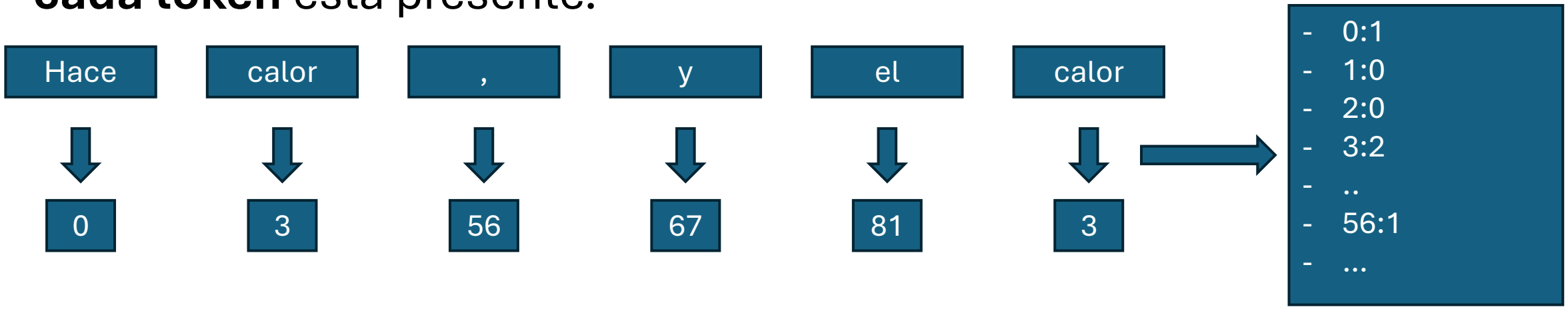


Tokenización
Mochila de palabras (Bag of words) (I)

- **Primer enfoque** para codificar texto en valores numéricos.
- El **tamaño total del vocabulario** es igual al **número máximo** de palabras, caracteres a representar, números que se deseen incorporar.
- Cada **elemento** en el vocabulario es un **token.**
  - “Hola” podría ser el token con identificador “1”
  - “!” Podría ser el token con identificador “2”

Tokenización
Mochila de palabras (Bag of words) (II)

- Cada **frase** se puede representar como el **número de veces que cada token** está presente.



- Inconvenientes
  - Los tokens **no codifican la relación entre las palabras**.
    - "Hola" tiene el token "1", "saludos" tiene el token con identificador "2000" a pesar de ser muy similares en significado.
  - No se tiene en cuenta el **orden** de las palabras.

Tokenización
Definiendo qué es un token

- En los métodos actuales, los **tokens codifican el vocabulario** que el modelo de lenguaje va a entender.
- Lo que no se pueda “escribir” con un token, no se podrá procesar ni generar.
- ¿Cómo definimos qué es un token?
  - 1 palabra 1 token
    - Palabras muy similares tokens diferentes
    - Reentrenamiento cuando aparezcan palabras nuevas
  - 1 sílaba 1 token
    - Permite combinar sílabas y formar palabras.
    - Permite reusar prefijos y sufijos.
    - El más usado.
  - 1 carácter 1 token.
    - Mucha flexibilidad pero complicado generar palabras completas correctamente.

Tokenización
Seleccionar un tokenizer

- Comprobar el **tamaño del vocabulario.**
- Qué tokens hay disponibles. ¿Necesito **tokens especiales**?
  - Si mi texto tiene emojis, el tokenizer debe poder codificarlos.
- ¿Soporta **mayúsculas**?
  - Para aplicaciones de búsqueda de nombres, el hecho de incorporar mayúsculas facilita la tarea.
- ¿**En qué dataset** se ha entrenado?
  - Un tokenizer de un lenguaje de programación será diferente a uno de literatura china.

Tokenización
Tokenizers populares

	WordPiece	Byte Pair Encoding (BPE)
Tamaño del vocabulario	32.522	50.257 - >100.000
Caracteres especiales	<UNK>,<SEP>,<PAD>...	<UNK><|endoftext|>
Modelos que lo usan	BERT	GPT, GPT-4

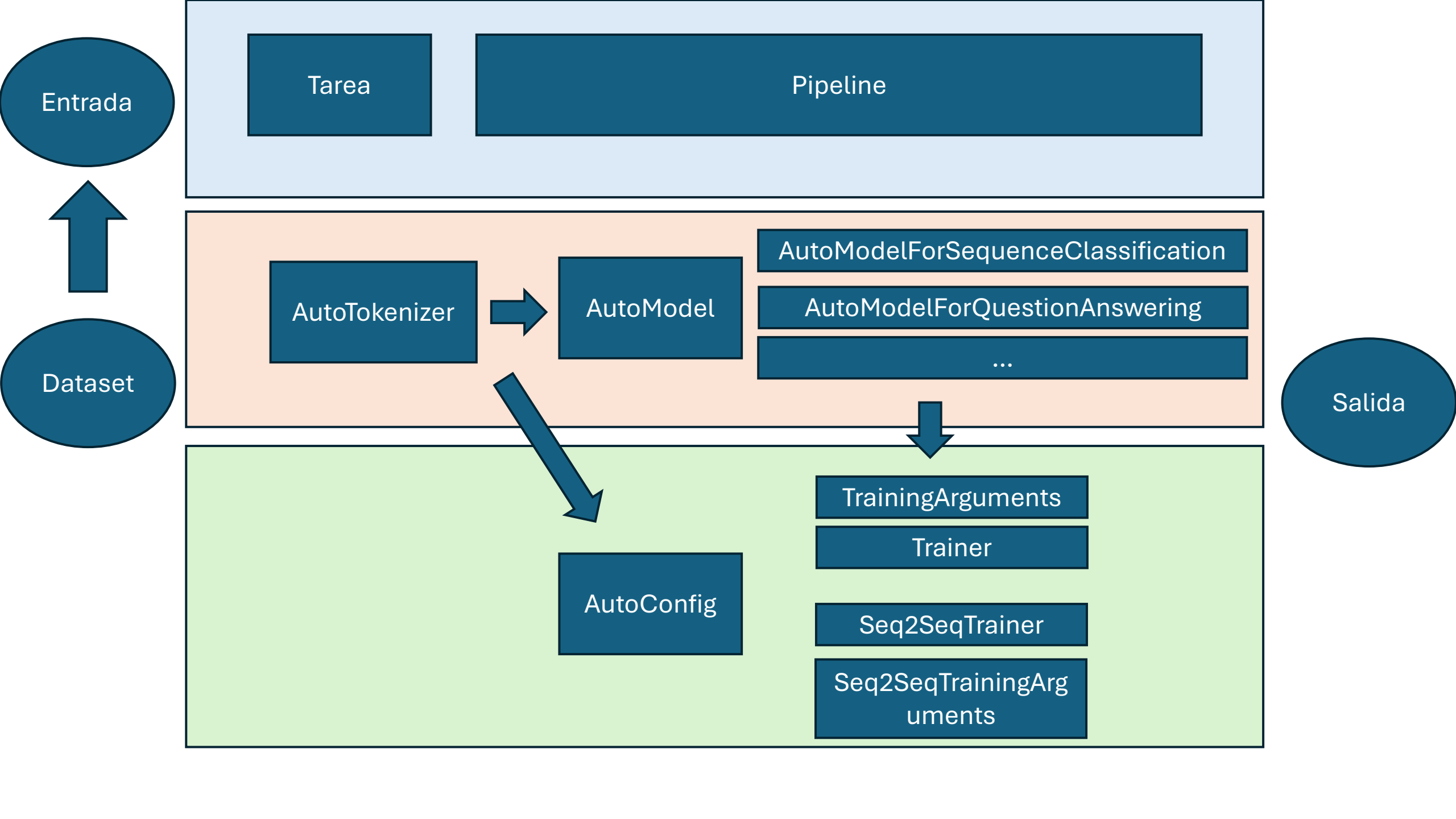
['transform', '##ers', 'are', 're', '##vol', '##ution', '##izing', 'ai', '.']

['Machine', 'Ġlearning', 'Ġis', 'Ġfun']

Tokenización
[Implementación] - Librerías

- Librería “**tokenizers**” de “Hugging Face”
  - https://huggingface.co/docs/tokenizers/index
  - Permite crear **tokenizers desde 0** a partir de uno o varios archivos.
  - Permite usar tokenizers pre-entrenados.

- Librería “**transformers**” de la comunidad “Hugging Face”.
  - Permite usar **tokenizers pre-entrenados** en diferentes datasets.
    - https://huggingface.co/docs/transformers/tokenizer_summary
    - AutoTokenizer
    - XLNetTokenizer
    - BertTokenizer
  - Permite importar los tokenizers definidos en “tokenizers”
- https://huggingface.co/transformers/v3.0.2/quicktour.html


Entrada
Dataset
Tarea
Pipeline
AutoTokenizer
AutoModel
AutoModelForSequenceClassification
AutoModelForQuestionAnswering
...
Salida
AutoConfig
TrainingArguments
Trainer
Seq2SeqTrainer
Seq2SeqTrainingArguments

tokenizers

- https://huggingface.co/docs/transformers/tokenizer_summary
- Incorpora funciones para **cargar tokenizers pre-entrenados**
  - AutoTokenizers
- Incorpora tokenizers **sin pre-entrenar** para realizar un entrenamiento propio.
  - Permite guardar tokenizers propios.

Ejercicio 0

- Vamos a probar los tokenizers.

- Archivo: CursoGenAI_UAL_01_tokenizer.ipynb

- **Probar diferentes modelos pre-entrenados**

- Visualización del tokenizer de OpenAI
  - https://platform.openai.com/tokenizer

Ejercicio 1

- Entrenar un tokenizer propio sobre unos datos almacenados en memoria.
  - https://huggingface.co/docs/tokenizers/pipeline
- Guardarlo en formato .json
- Incorporar el archivo “CursoGenAI_UAL_01_testFile.py”

- **Ejercicio 1**: Re-entrenar el tokenizer con un nuevo archivo más apropiado (buscar), ajustando los parámetros del tokenizer. Ver el resultado tras ejecutarlo sobre “CursoGenAI_UAL_01_testFile.py”.
- **Ejercicio 2**: Descargar quijote.txt y ejecutar el tokenizer.
- **Ejercicio 3**: Ejecutar un tokenizer pre-entrenado (AutoTokenizer de transformers).

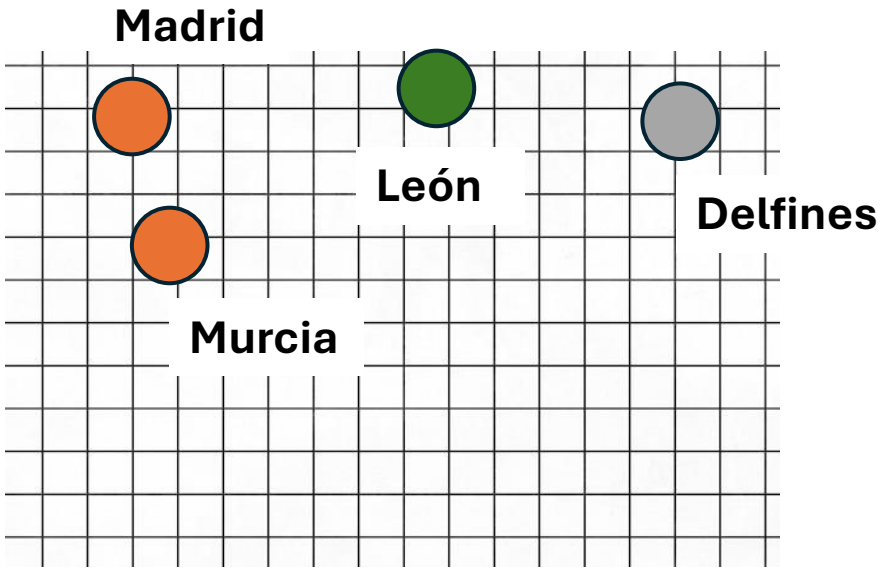
3 Embeddings

Embeddings
Conceptos generales (I)

- ¿Cómo entiende un modelo la relación entre dos tokens similares?
- Un token es un valor **unidimensional**.
  - El token “perro” puede ser el “128” y “dog” el “22.980”. Cuando realmente son la misma palabra, solamente que cambiando el idioma.
- Ignoran la naturaleza **semántica** y el **significado** del texto.
- Para que los tokens sean más **característicos** del **significado** de la palabra o sílaba subyacente, se incorporan los **embeddings**.

Embeddings
Conceptos generales (II)

- Los embeddings son **vectores multidimensionales** que permiten capturar el **significado** de las palabras.
- Palabras **similares** tienen **tokens** similares.



	Plural	Humano	Objeto	Animal	Servicio
Madrid	0.5	0.1	0.5	0.5	0.1
León	0.9	0.4	0.2	0.1	0.9
Delfines	0.9	0.3	0.3	0.1	0.9
Murcia	0.5	0.1	0.5	0.5	0.2

Embeddings
Entrenamiento – Word2Vec (I)

- Los embeddings son vectores que necesitan **“entrenarse”** para que **codifiquen** los mejor posible las palabras que representan.
- Se entiende que **palabras muy relacionadas** (en significado) suelen estar “**cerca**” entre sí.
- ¿Cómo se entrenan?
  - Usando **redes neuronales**…
  - Y “**Contrastive learning**”.
- Se cogen **pares de palabras relacionadas** y **no relacionadas** entre sí.
- La **red neuronal predice** si ambas palabras están relacionadas o no.

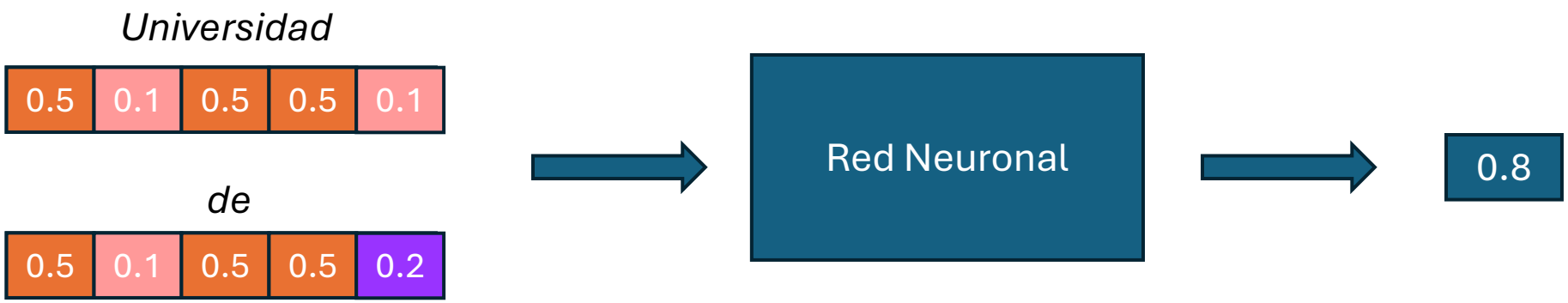
Embeddings
Entrenamiento – Word2Vec (II)

- *Yo voy a ser ingeniero informático por la **Universidad de Almería**.*
  - Ejemplos de entrenamiento:

Palabra 1	Palabra 2	Clase
Yo	voy	1
voy	a	1
yo	por	0

Palabra 1	Palabra 2	Clase
Universidad	almería	1
Universidad	de	1
Almería	informático	0

- *La **Universidad de Almería** está creciendo mucho.*



Ejercicio 2a

- Ejercicio: Estudiar el embedding de Word2Vec

- Ejecutar el archivo: CursoGenAI_UAL_02_embeddingsWord2Vec.ipynb

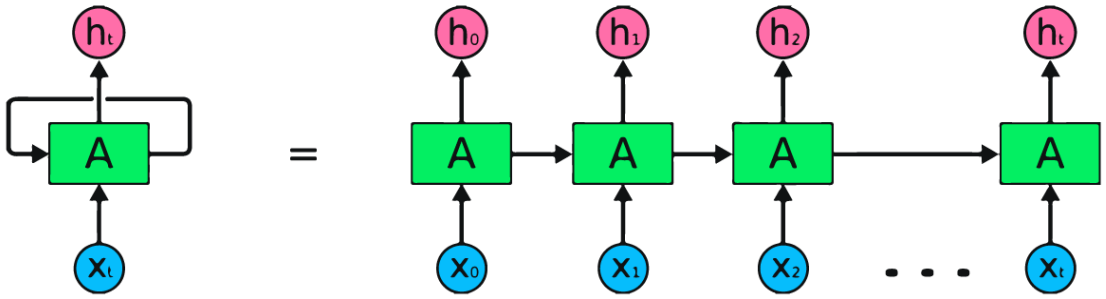
- Obtener las películas más relacionadas con una película dada.
- Examinar los embeddings generados.
- Entrenar un modelo sobre un dataset propio.

Embeddings
Problemas de Word2Vec

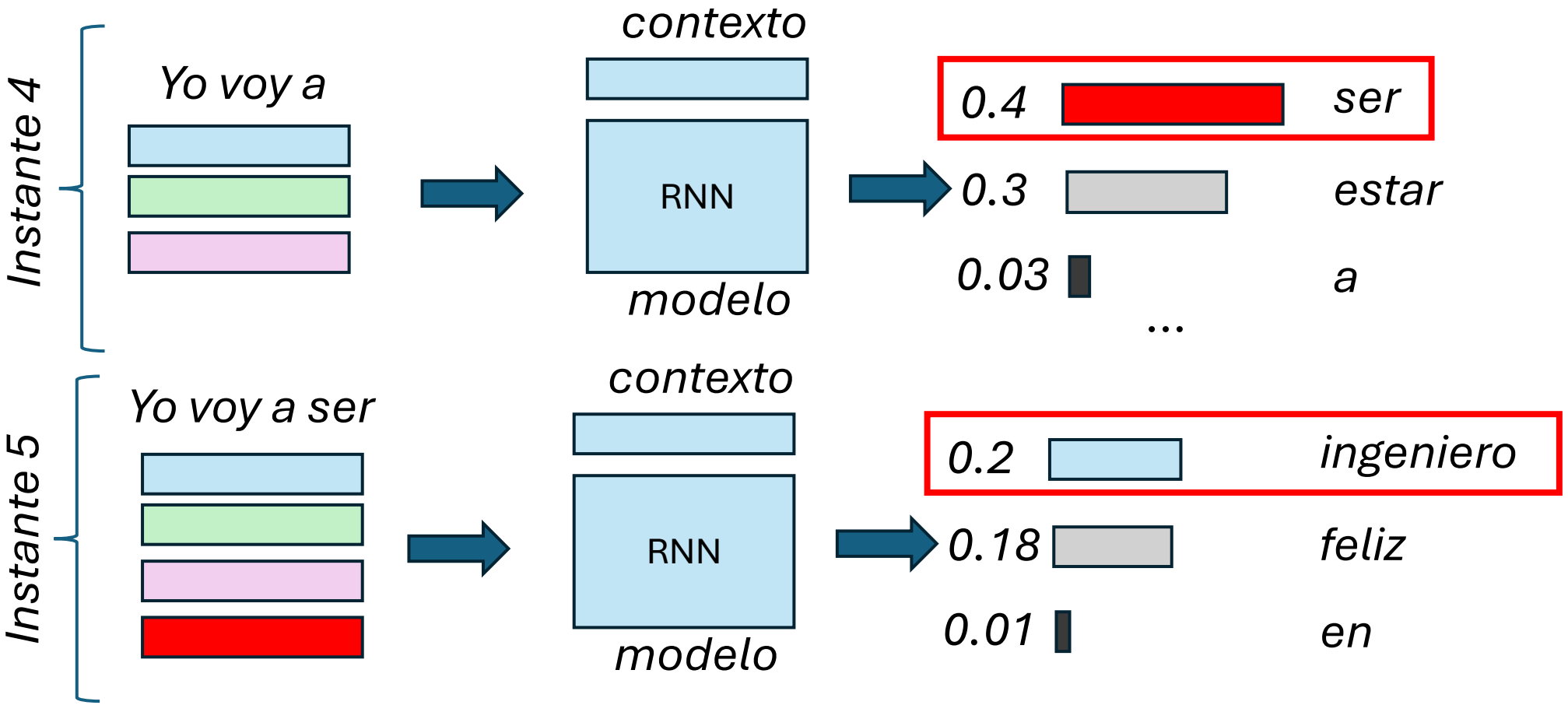
- Por ejemplo, si tenemos estas frases:
  - Fui a sacar dinero al **banco**.
  - El **banco** de peces desapareció al instante.
- **Banco** es una palabra **cuyo embedding es el mismo (estático)** en ambos casos.
- ¿Significa realmente siempre lo mismo? Depende del **contexto**.
  - Si tengo que generar frases de un **banco de peces**, pues la siguiente palabra va a ser muy probable que sea “de” y luego “peces”, pero si se está hablando de **dinero**, no.

Embeddings
Generación de contexto con RNNs (I)

- El principal objetivo de un **embedding con contexto**, es **cambiar** la **representación** de una palabra dependiendo del **contexto**.
- **Tarea**
  - Generación de la próxima palabra a partir de una frase dada.
  - Para generar la palabra, un modelo parte de los embeddings de las anteriores palabras, y de un contexto (formado por estas palabras juntas).
- Uso de RNNs (Redes Neuronales Recurrentes)



Embeddings
Generación de contexto con RNNs (II)



- Si la frase es muy **larga**, no se puede mantener un **contexto**; se **olvida**. Además, son muy **costosas** de entrenar.

Ejemplo

- Archivo: CursoGenAI_UAL_03_LSTM_autoregressive.ipynb

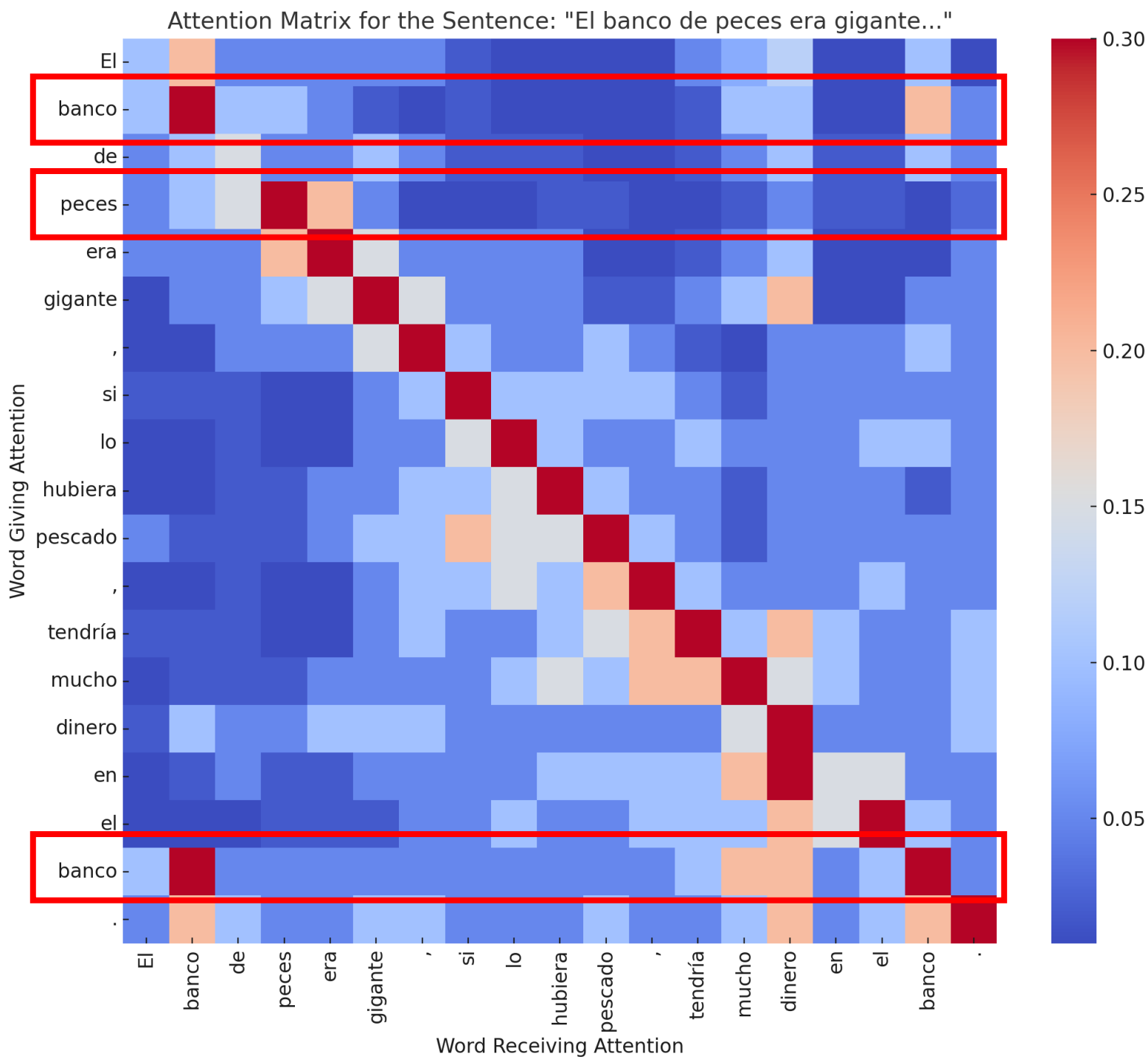
- Implementar de una red autoregresiva de generación de texto.
- Observaciones:
  - Visualizar los datos de entrenamiento.
  - Entrada/salida en cada paso.
  - Visualizar el embedding de la misma palabra en varias generaciones.

- Probar a generar diferentes frases.

4 BERT

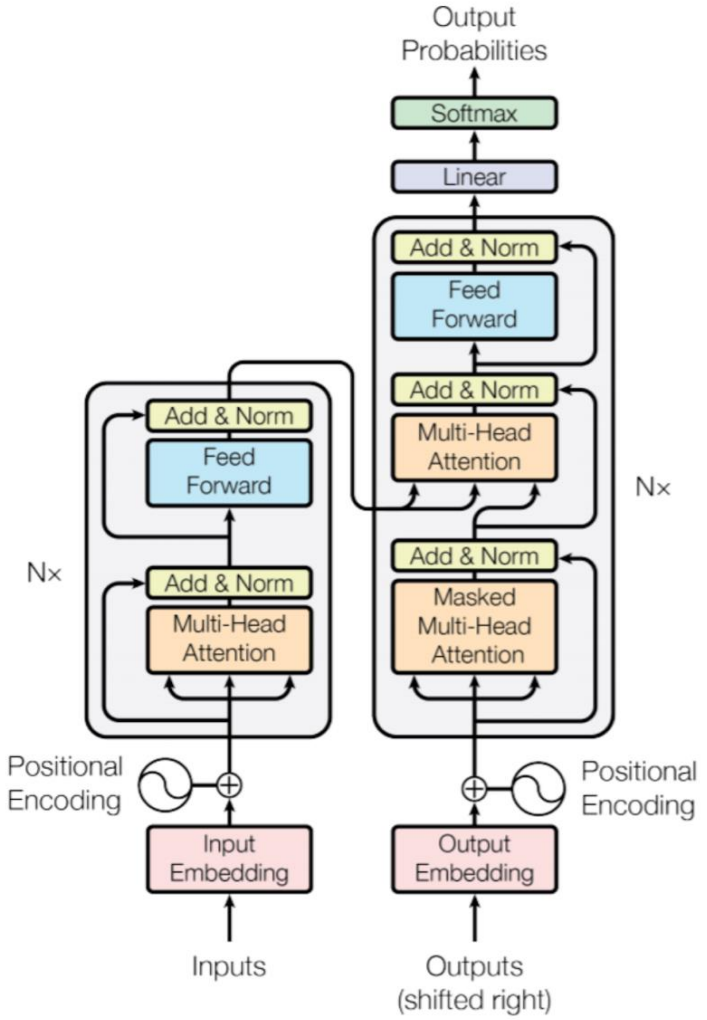
Attention is all you need

- En el anterior enfoque, el **embedding de la palabra NO CAMBIA**. Únicamente lo hace el **contexto** (que almacena la RNN).
- Además, es muy ineficiente: se calcula el **contexto** de forma **secuencial** y se pierde el contexto confome se tienen más palabras, se "**olvida**".
- Solución: Uso de **ATENCIÓN**.
  - La atención permite que el modelo evalúe la **relevancia de cada palabra** en la secuencia en **relación con las demás**, haciendo un análisis **global**.
  - Hay **palabras** más **importantes** que otras (afectan más al contexto).
  - El **vector de embeddings cambia en cada palabra**; depende del contexto.


Attention Matrix for the Sentence: "El banco de peces era gigante..."
Word Giving Attention
El
banco
de
peces
era
gigante
,
si
lo
hubiera
pescado
,
tendría
mucho
dinero
en
el
banco
.
Word Receiving Attention
El
banco
de
peces
era
gigante
,
si
lo
hubiera
pescado
,
tendría
mucho
dinero
en
el
banco
.
0.30
0.25
0.20
0.15
0.10
0.05

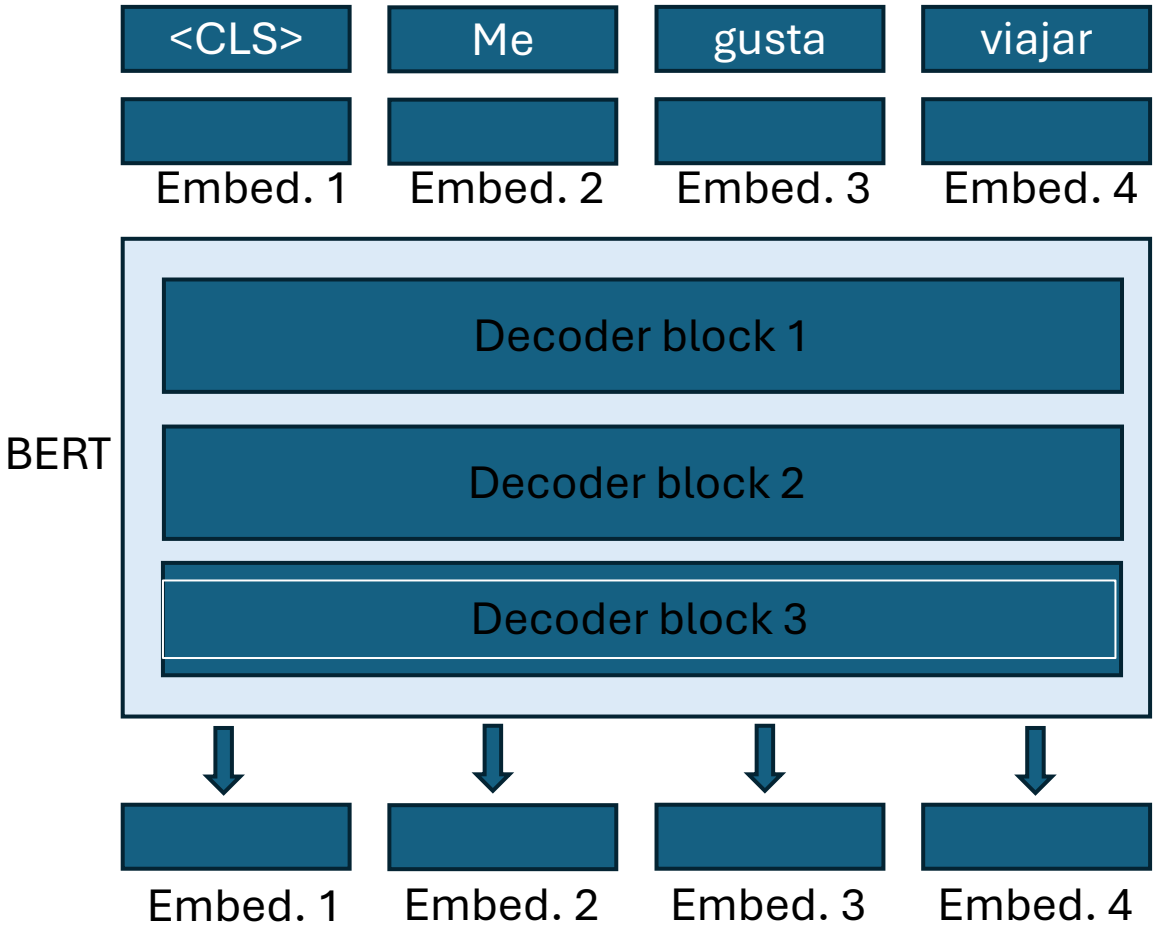
Transformers
Concepto general

- **Encoder**:
  - “Entiende el texto” y genera embeddings.
    - *Extractor de características*.
- **Decoder**
  - “Transforma” los embeddings en nuevos datos a generar.
    - *Respuesta de preguntas*.
    - *Resumen de texto*
    - *Traducción de texto*
    - *Generación de imágenes*



BERT
Arquitectura

- Es un transformer **ENCODER**.
- También llamado "**representation model**" o "**foundation model**".
- Entrada:
  - **Cadena** de texto
- Salida:
  - **Embeddings** para el texto de entrada: 1 palabra 1 embedding.
- Disponibles modelos **BERT pre-entrenados** en datasets gigantes.



Ejercicio 2b

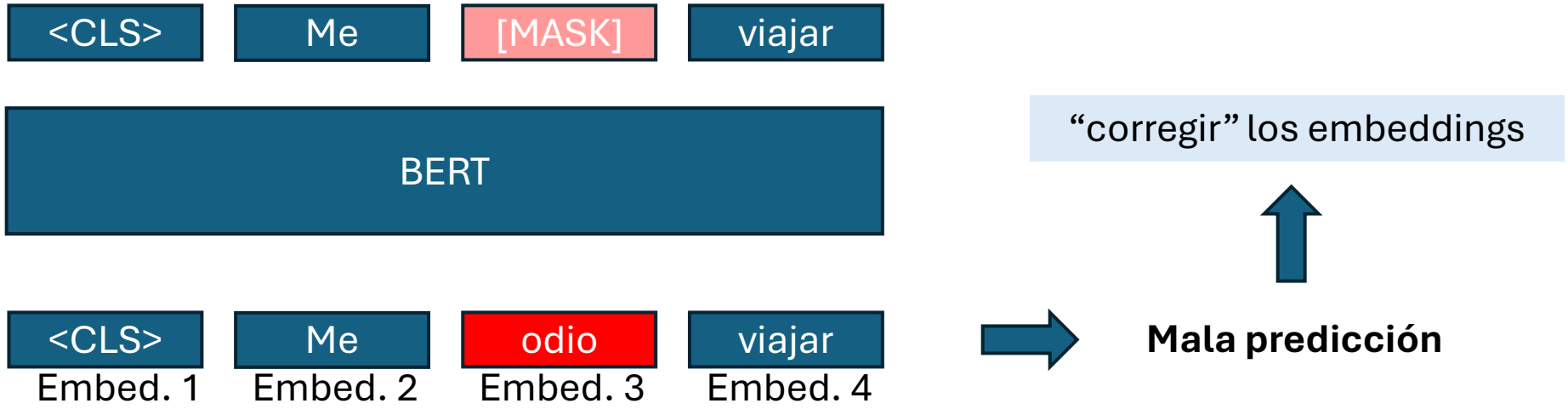
- Ejercicio: Usar modelos de embedding pre-entrenados:
  - https://huggingface.co/models

- Ejecutar el archivo: CursoGenAI_UAL_02_ejercicio_embeddings.ipynb

- Usar el embedding. ¿Cuántas dimensiones tiene cada token?
- Obtener las 10 palabras más “similares” a una palabra dada. Por ejemplo “Almería”, o “Banco”.
- Obtener los embeddings en diferentes frases que incorporan la misma palabra. ¿Qué embedding tiene la palabra dada en cada caso?

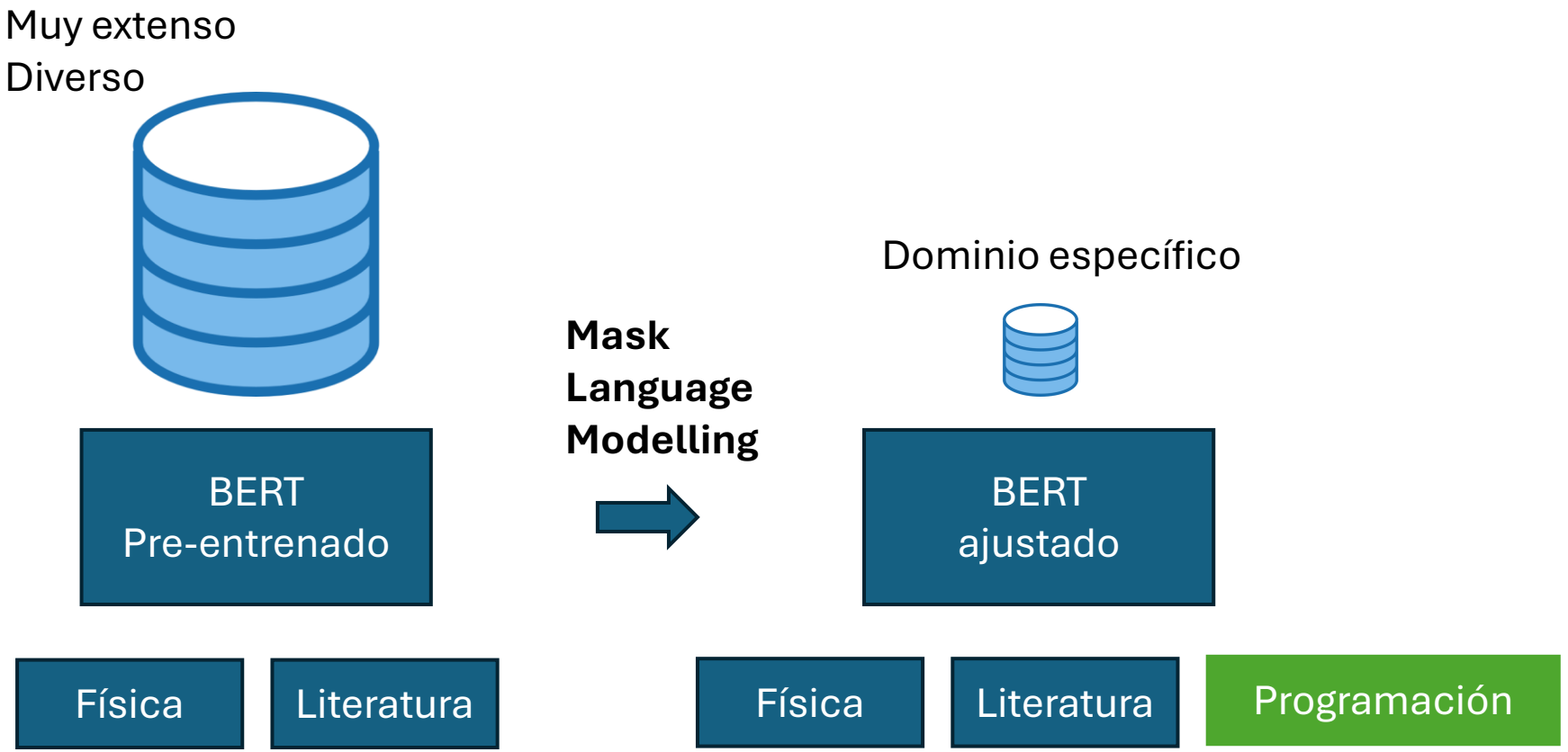
BERT
Entrenamiento

- Entrenamiento con “**Mask language modelling**”.
  - Se incorporan máscaras para **ocultar** algunos **tokens** de la frase.
    - 15% de los tokens.
  - BERT tiene que ser capaz de **predecirlos** por el contexto.
  - Función de pérdida “**cross-entropy**” para calcular el error en la tarea de **clasificación**.



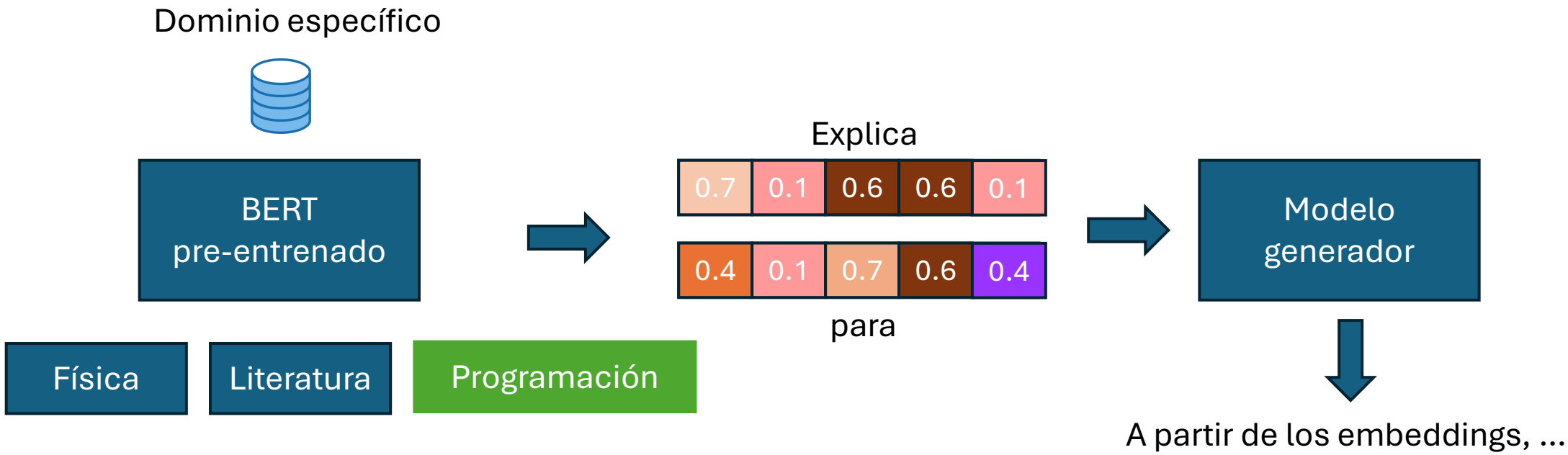
BERT
Fine-Tuning en un dataset propio

- **Adaptar** BERT para que entienda un "**nuevo dominio**".
- Hacer **mask language modelling** en el nuevo dataset.



BERT
Para qué se usa

- **Complicado** de **entrenar**. Muy **costoso**. ¿Cómo se usa realmente?
- Para **otener los embeddings** de una frase/palabras.
- Después con esos embeddings, se pueden **entrenar** otros **modelos** que hagan otras tareas. Por ejemplo, **GENERAR TEXTO**.



Ejercicio 3

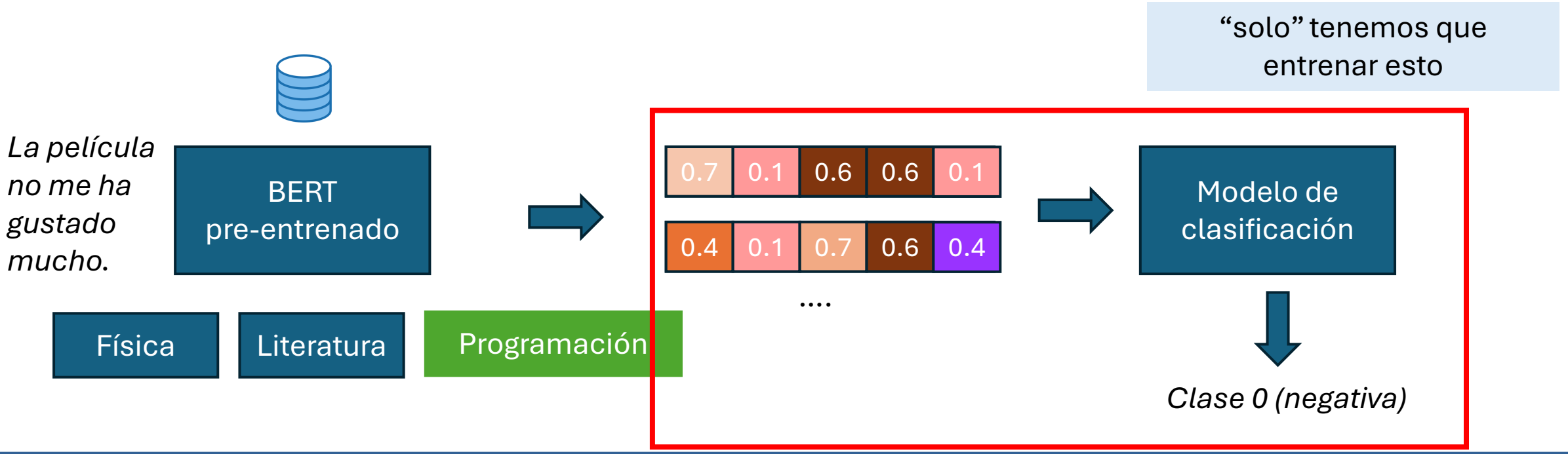
- Archivo: CursoGenAI_UAL_04_ejercicio_03_bert_entrenamiento.ipynb

- Librería “transformers”
  - https://huggingface.co/docs/transformers/v4.45.1/en/model_doc/bert#overview

- Observaciones:
  - Cargar un modelo de BERT sin entrenar.
  - Cargar los datos de un dataset.
  - Aplicar Mask Language Modelling.
  - Guardar el modelo para su posterior uso.
  - Hacer fine-tuning.
  - Evaluar el modelo en el dataset.

BERT
Clasificación a partir de embeddings

- Los embeddings de por sí solos, no sirven para “nada”… ¿o sí?
- Se puede clasificar un conjunto de palabras teniendo en cuenta su semántica y su contexto.



Ejercicio 4

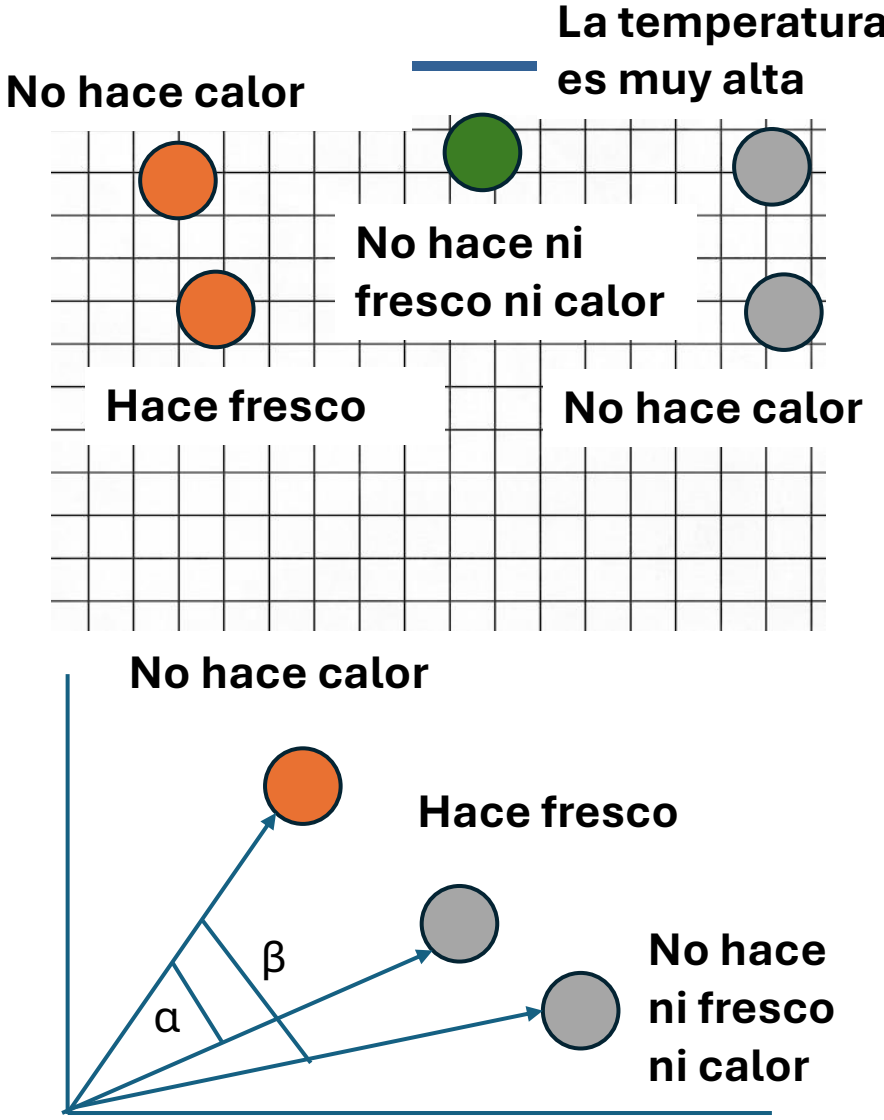
- Archivo: CursoGenAI_UAL_05_ejercicio_04_bert_clasificacion.ipynb

- Usar el dataset dado.
- Usar un modelo pre-entrenado para obtener embeddings.
- Entrenar un modelo simple de ML (kNN, DecisionTree) con los embeddings generados.
- Evaluar el modelo con el dataset de test.
- Proponer mejoras.

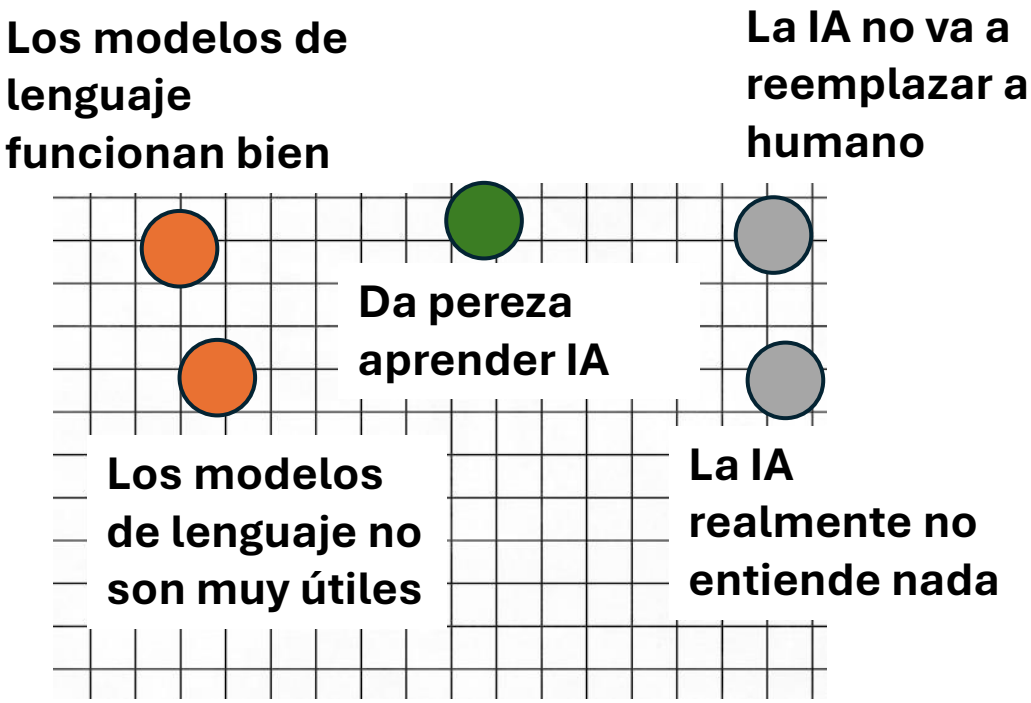
5 S-BERT

S-BERT
Conceptos generales

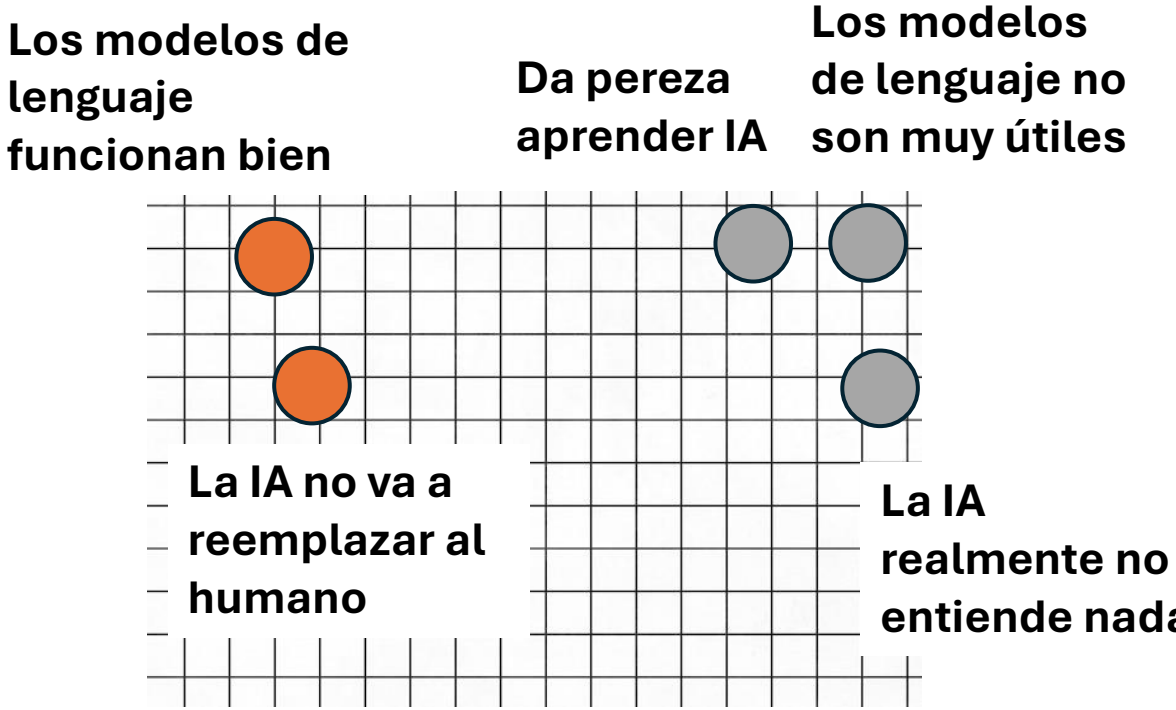
- Generación de **embeddings para frases completas**.
  - Comprimir información.
  - Misma longitud todos los elementos del dataset.
- Cada frase tiene un embedding diferente.
- Entrenamiento usando "**contrastive learning**".
  - Se cogen **parejas** de frases **iguales** y **diferentes**.
  - S-BERT intenta representar las frases "iguales" de forma muy similar, y separarlas de las "diferentes".
- Definir la **función de pérdida**:
    - **Softmax**: Clasificación entre "igual", "diferente"o "neutral".
    - **Cosine Similarity**: Calcula el factor de similitud entre las dos frases.



S-BERT
Ejemplo



S-BERT



S-BERT aplicado a una tarea de clasificación de comentarios positivos/negativos

Ejercicio 5

- Archivo: CursoGenAI_UAL_06_ejercicio_05_SBERT_similarity.ipynb

- Generación de una base de datos de embeddings con un modelo S-BERT pre-entrenado.
- Cálculo de similitud entre embeddings.

- Ejercicio: Obtención de la mejor respuesta a una pregunta dada.

S-BERT
Fine-tuning supervisado en un nuevo dataset

- ¿Por qué? Para lidiar mejor con la **tarea**, o cuando se tiene un **nuevo contexto** o **dominio**.
- 1) Si S-BERT ha sido entrenado en un dominio **similar**.
  - Se **carga** un S-BERT pre-entrenado en otro contexto/dominio.
  - Se hace **fine-tuning** usando **contrastive** learning.
- 2) Si S-BERT se ha pre-entenado en un dominio **muy diferente**.
  - Se aplica **mask language modelling** para entender el nuevo dominio.
  - Se genera un modelo **pre-entrenado**.
  - Se entrena el modelo con **frases** en el nuevo dominio usando contastive learning.
- Lo más **importante**: Encontrar frases que tengan diferente significado pero que sean muy similares: **Hard negatives**.
- Principal **problema**: Se necesitan **muchos pares** de frases.

Ejemplo

- Archivo: CursoGenAI_UAL_07_ejemplo_SBERT_finetune.ipynb

- Entrenamiento de SBERT con un dataset particular

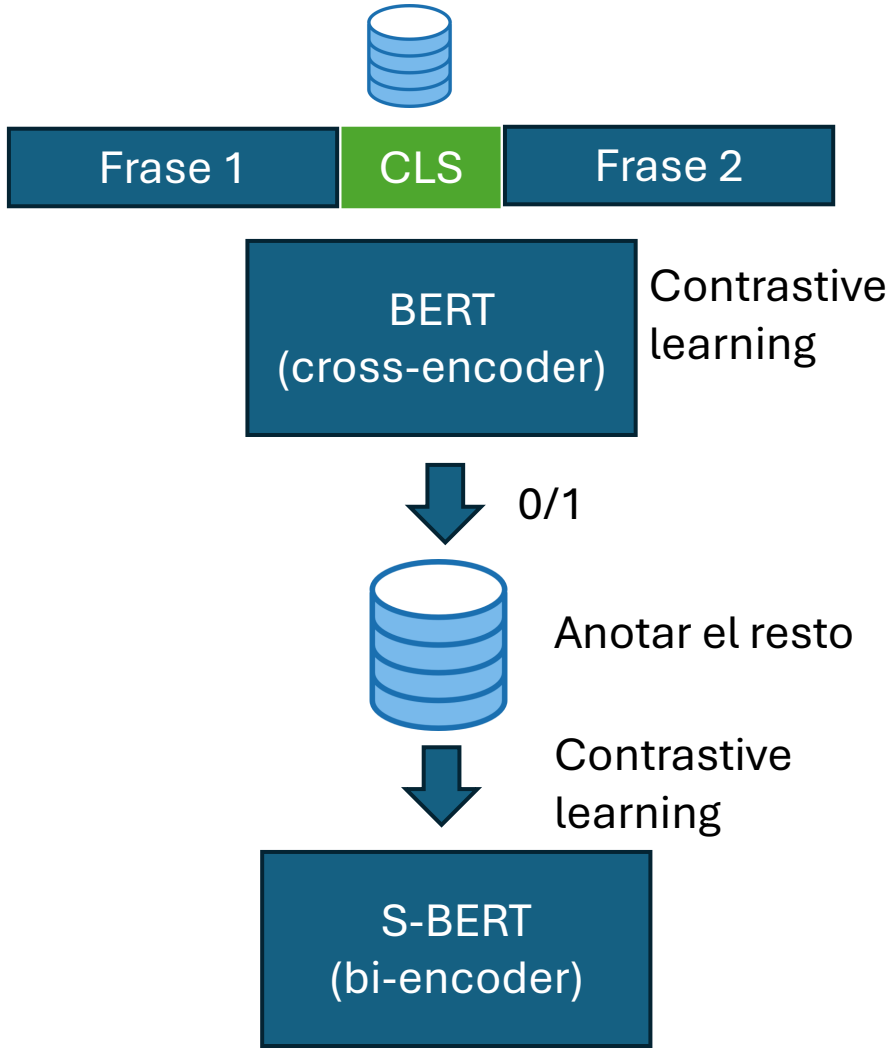
S-BERT
Evaluación

- **Benchmarks**
  - GLUE: 9 tareas de comprensión del lenguaje.
    - **MNLI** : Multi-Genre Natural Language Inference corpus
      - Pares de frases anotadas con “contradicción”, “neutral” y “similares”.
  - **STSB**: Semantic Textual Similarity Benchmark
    - Pares de frases anotadas con valores entre 1 y 5 según su similitud.

- Definir un **evaluador** de nuestro modelo
- Métrica más importante: “**pearson cosine**”

S-BERT
Fine-Tuning semi-supervisado

- En la mayor parte de las ocasiones, **adquirir datasets** completamente anotados es **complicado**.
- Para ello, se proponen varias etapas:
  - Coger un **dataset anotado pequeño** (Golden dataset).
  - Entrenar **BERT** (cross-encoder) usando "**contrastive learning**". La salida de BERT ahora es un valor entre 0 y 1 indicando la similitud entre dos frases de entrada.
  - Usar el **BERT** pre-entrenado para **anotar** el resto del dataset.
  - Usar **contrastive learning** para entrenar **S-BERT**.



Ejercicio 6

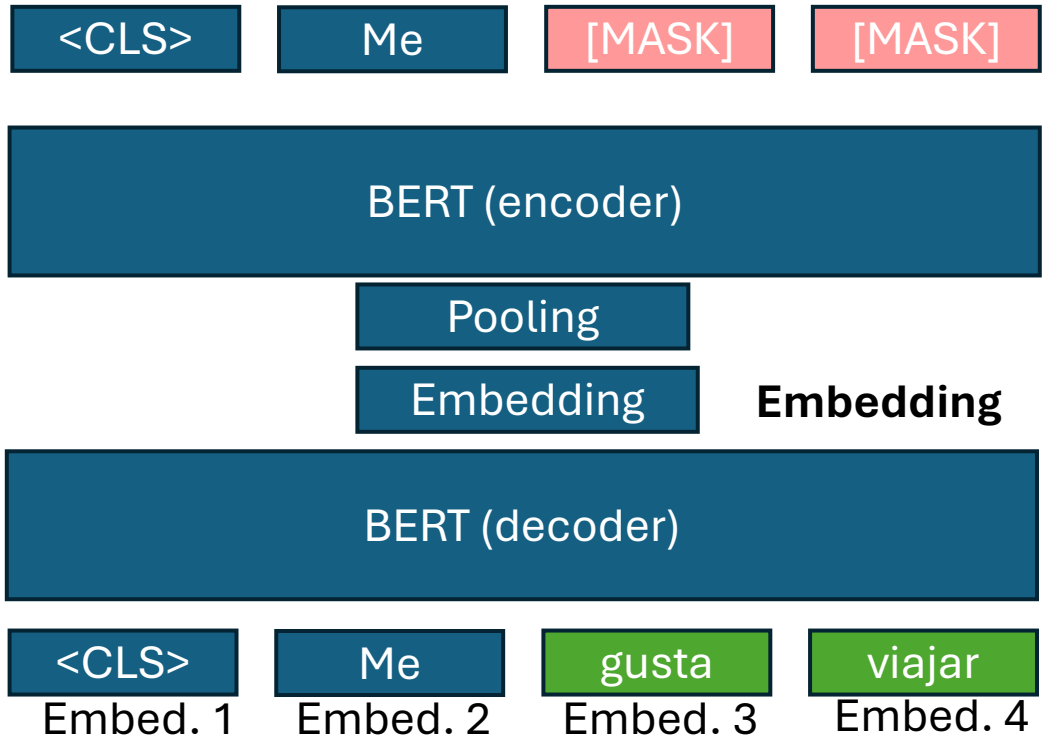
- Archivo: CursoGenAI_UAL_08_ejercicio_06_SBERT_semisupervisado.ipynb

- Fine-tuning de S-BERT en un dataset particular.
- Uso de BERT para anotar datos.

- Ejercicios:
  - Comparar entre el entrenamiento con un dataset pequeño y un dataset anotado con BERT.
  - Probar modelos pre-entrenados de S-BERT
  - Crear un conjunto de test propio y evaluar S-BERT.

Embeddings de frases
Entrenamiento NO supervisado

- *"Un buen embedding de frase permite a un decodificador reconstruir la frase original".*
- Entrenamiento usando **"Mask Language Modelling".**



Codifica la frase de tal manera que permite a un decodificador, generar la frase original a partir de sus valores.

6 Clasificación

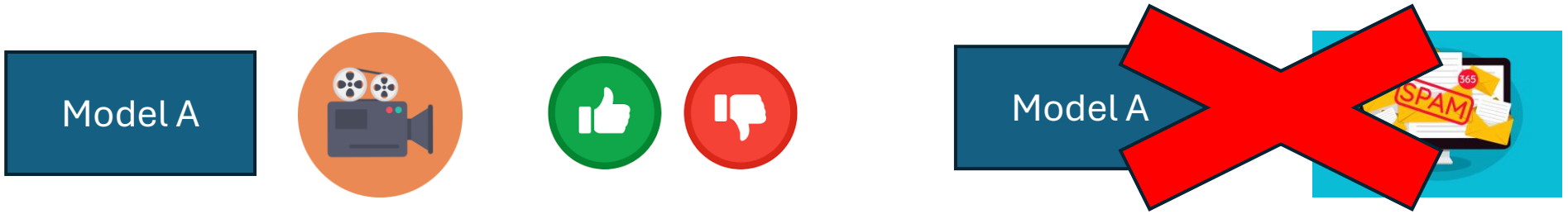
UNIVERSIDAD DE ALMERÍA

Clasificación de texto
Conceptos

- Consiste en **etiquetar** una palabra o frase entre una o varias **clases**.
- Usando modelos de lenguaje:
    1. Un modelo de representación **pre-entrenado como BERT o S-BERT** extrae los **embeddings** de la frase.
    2. Se añade un **clasificador** (o capa) para clasificar los embeddings.
- *Ejemplos: detección de spam en mails, clasificación de opiniones en amazon, detección de discurso ofensivo en redes sociales.*

Clasificación de texto
Uso de modelos de clasifación pre-entrenados

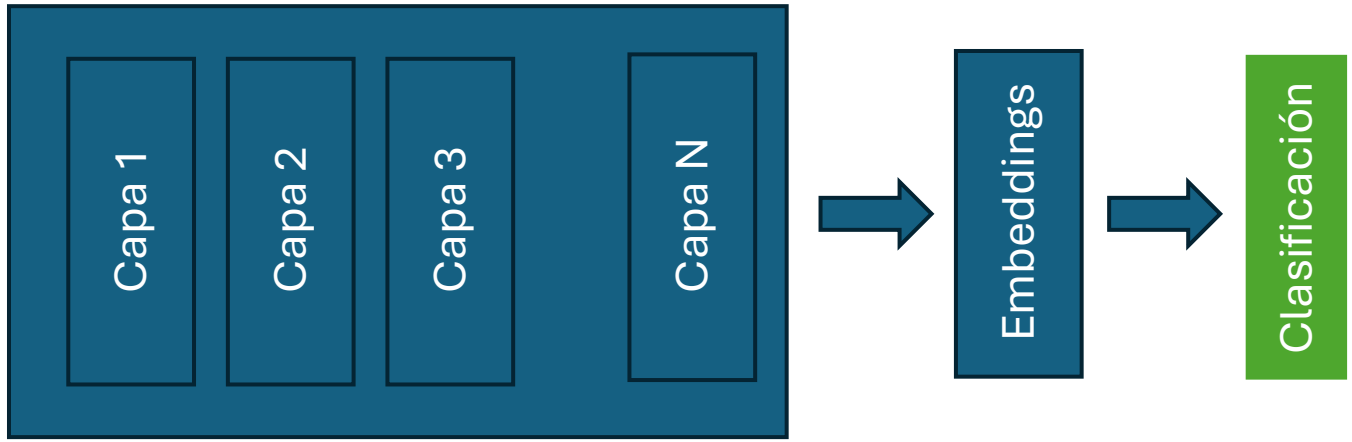
- Entrenados sobre **un dataset específico**.
- Puede que **no funcionen** del todo bien en **nuevos** datasets/contextos.
  - Por ejemplo, si he sido entrenado en un dataset de reviews de películas, puede funcionar bien en un dataset de review de literatura. Pero a lo mejor no funciona muy bien en la clasificación de correos de SPAM.



- ¿Habría que reentrenar desde 0 un nuevo modelo en mi dataset? ¿Puedo usar un modelo de lenguaje pre-entrenado?

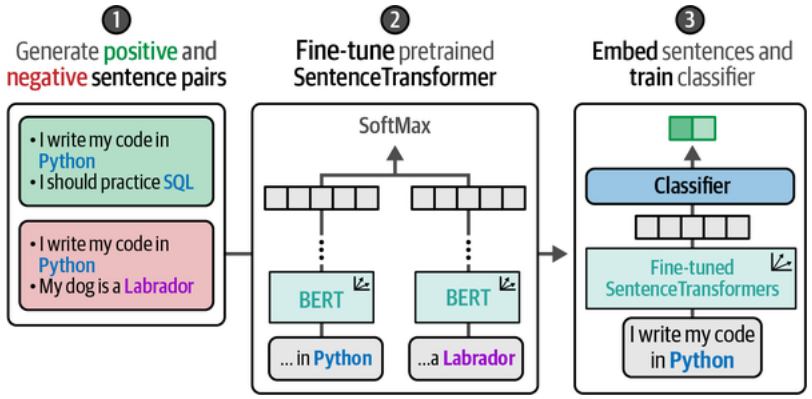
Clasificación de texto
Entrenamiento supervisado

- Partir de un **modelo de representación del lenguaje**.
  - **Descongelar** los pesos (BERT).
  - **Congelar** todo el modelo menos la última capa.
  - **Congelar** solamente algunas capas.
- Añadir una capa final **totalmente conectada**.



Clasificación de texto
Entrenamiento supervisado few-shot

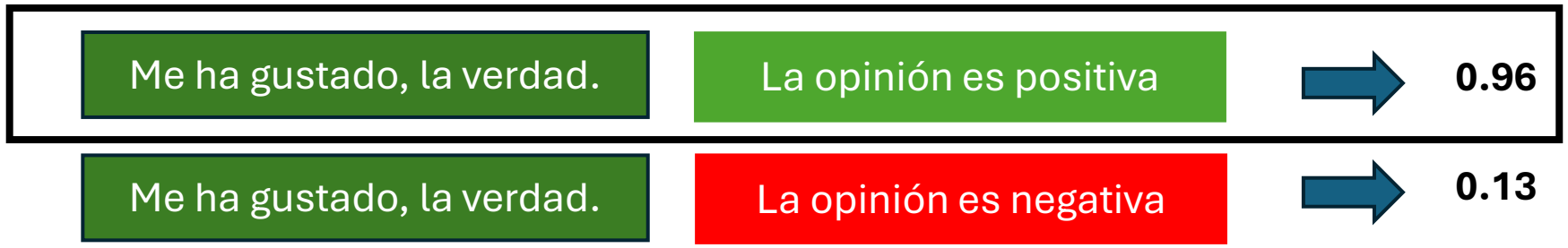
- Coger un **modelo pre-entrenado** S-BERT.
- Hay **muy pocos datos** de entrenamiento.
- **Generar un dataset** para **contrastive** learning.
  - **Emparejar** frases de la **misma** clase como positivas
  - **Emparejar** frases de **diferentes** clases como negativas
- **Fine-Tune** de S-BERT usando contrastive learning.
  - Los embeddings que se generan, harán que las **frases de la misma clase sean muy similares** y los de otra clase muy diferentes.



- Entrenar un clasificador a parte, o incorporar una capa de clasificación.

Clasificación de texto
Entrenamiento no supervisado

- Usando S-BERT pre-entrenado:
  - Se obtiene el embedding de la frase.
  - Se obtienen los embeddings de las “clases”.
- *¿Cómo se definen las clases?*
  - *Ver cómo varía la clasificación dependiendo de las clases que se le den al problema.*
- Se mide la distancia.
- Se coge la clase con menor distancia (cosine similarity)



Ejercicio 7

- Ejercicio:
  - Usar el siguiente dataset:
    - data = load_dataset("rotten_tomatoes")
  - Crear un array que contenga frases de los siguientes temas:
    - Naturaleza
    - Tecnología
    - Deporte
  - Crear un clasificador NO SUPERVISADO que clasifique las frases dependiendo de su tema. Habrá tres clases posibles.

7 Búsqueda semántica y RAG

UNIVERSIDAD DE ALMERÍA

Búsqueda semántica
Concepto general

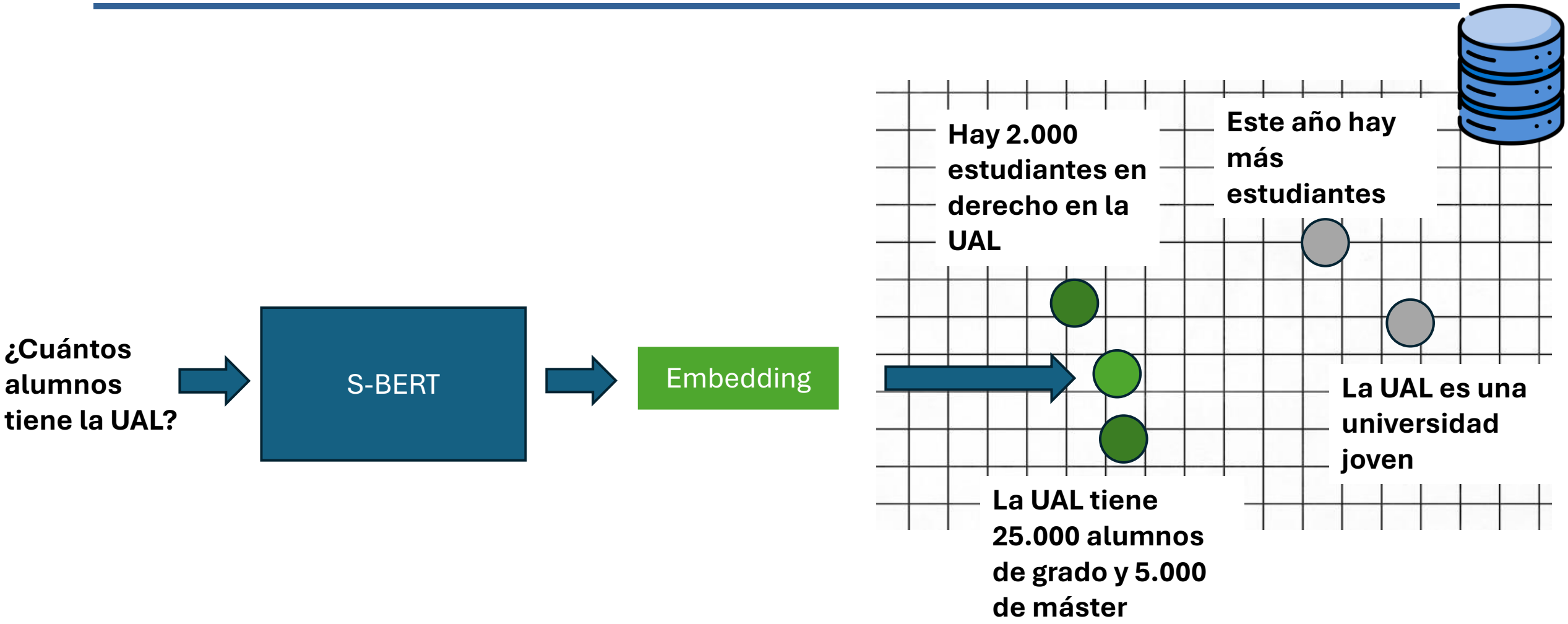
- Las **búsquedas** en grandes bases de datos de documentos se hacían con "**palabras clave**" hasta hace muy poco.
- Ahora, se puede buscar por "**significado**".
- Se hace una pregunta a un LLM, ¿qué pasa si **no sabe responder**?
  - **Hallucination**
  - Se propone "**aumentar su conocimiento**" sin necesidad de reentrenar.

- Pasos
  1. Dense retrieval
  2. Rerank
  3. Retrieval Augmented Generation

Búsqueda semántica
Búsqueda densa (I)

- Se tiene una **base de datos** que no contiene documentos, sino **embeddings**.
- A partir de una **pregunta**, se genera el **embedding** y se obtienen los **K** embeddings más "**próximos**".
- Ambos **embeddings** serán muy **similares** (el entrenamiento fuerza a ello al modelo).
- Normalmente, el LLM debe haber sido entrenado muy bien en datasets de **preguntas/respuestas**.

Búsqueda semántica
Búsqueda densa (II)

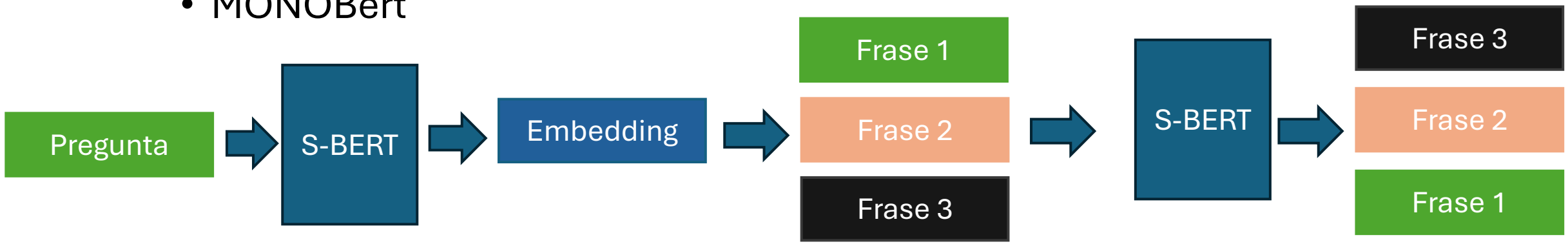


Búsqueda semántica
Búsqueda densa (III)

- ¿Cómo se genera la **base de datos**?
- Se **dividen** los documentos
  - Documento **completo**: Se **comprime** mucho, se pierde mucha información.
  - **Frases**: Muy **granular**, pueden incorporar contenido **incompleto**.
  - **Párrafos**: Pueden ser muy **grandes**.
  - **Ventanas deslizantes** de párrafos: Tiene en cuenta el **contexto**.
- Se obtiene el **embedding** de la información.
  - Usando un modelo **pre-entrenado**.
- Se genera un **índice** (para su búsqueda eficiente)
- Se **consultan** los puntos más cercanos a la pregunta.
  - Método de los vecinos cercanos.
  - Bases de datos.
- Es muy importante el **contexto** del modelo de **embedding**.

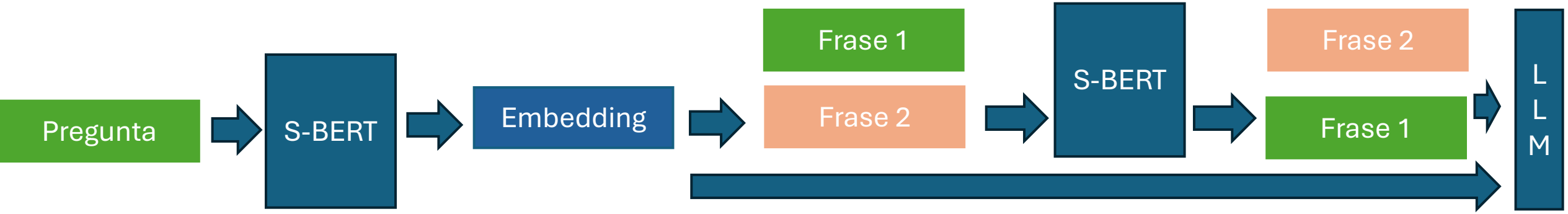
Búsqueda semántica
Reranking

- **Ordenar** los documentos extraidos por **relevancia** para responder a la pregunta original.
- Se pasan los **documentos** y la **pregunta** al modelo.
- Métricas:
  - Mean Average precisión (**MAP**)
- Modelos
  - https://www.sbert.net/
  - MONOBert



Búsqueda semántica
Retrieval Augmented Generation (RAG)

- Dotar al modelo LLM de la habilidad de busar en **bases de datos específicas** para contestar a las preguntas.
- Capacidad de **búsqueda** y de **generación de respuestas**.
- Permite “hablar” con los documentos/datos aportados.
- Funcionamiento
  - Se obtienen los **documentos más relacionados** con la pregunta.
  - Se añade al **prompt del LLM** que se genere una respuesta para constestar a la pregunta teniendo en cuenta los documentos obtenidos.



Ejercicio 8

- Ejercicio: CursoGenAI_UAL_09_ejercicio_08_busquedaSemantica_RAG.ipynb
  - Usar la base de datos que se ha usado en el ejercicio 5.
  - Guardar los datos en una base de datos indexada.
  - Obtención de embedding de una pregunta dada.
  - Comparativa de embeddings de la pregunta con el resto de la base de datos.
  - Obtención de la respuesta más adecuada.
  - Creación de un bot

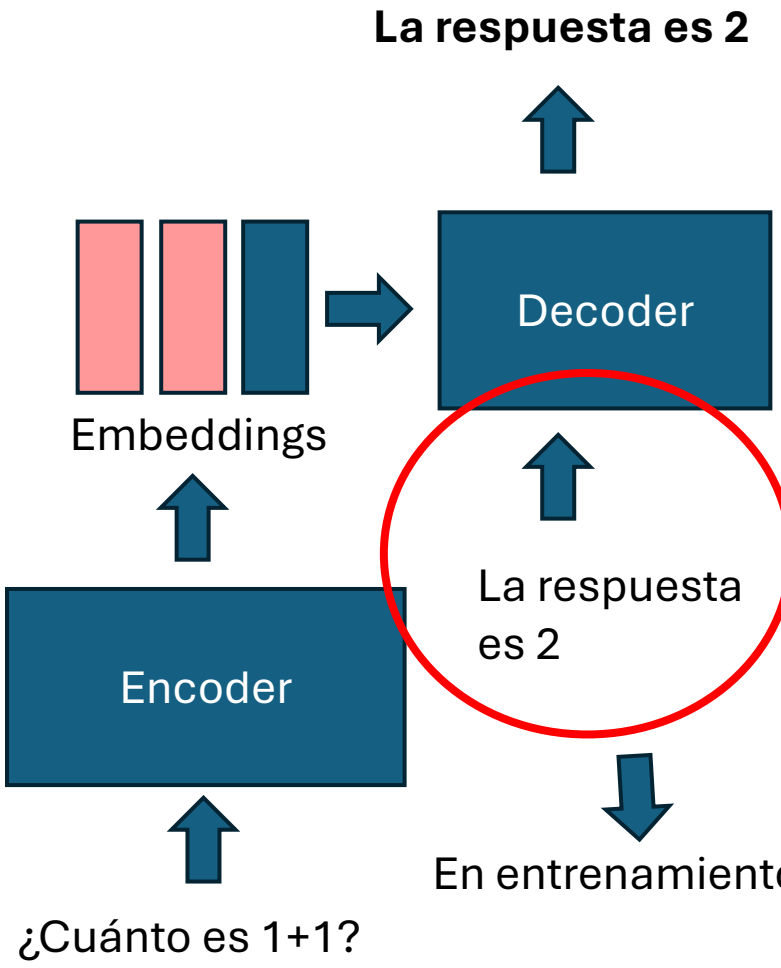
- Ejercicio
  - Incorporar información sobre el grado en ingeniería informática de la UAL.
  - Incorporación de un modelo de “transformers” generativo, tipo gpt-2.
  - Investigar la incorporación de interfaz gráfica de interacción.

8 LLMs generativos

UNIVERSIDAD DE ALMERÍA

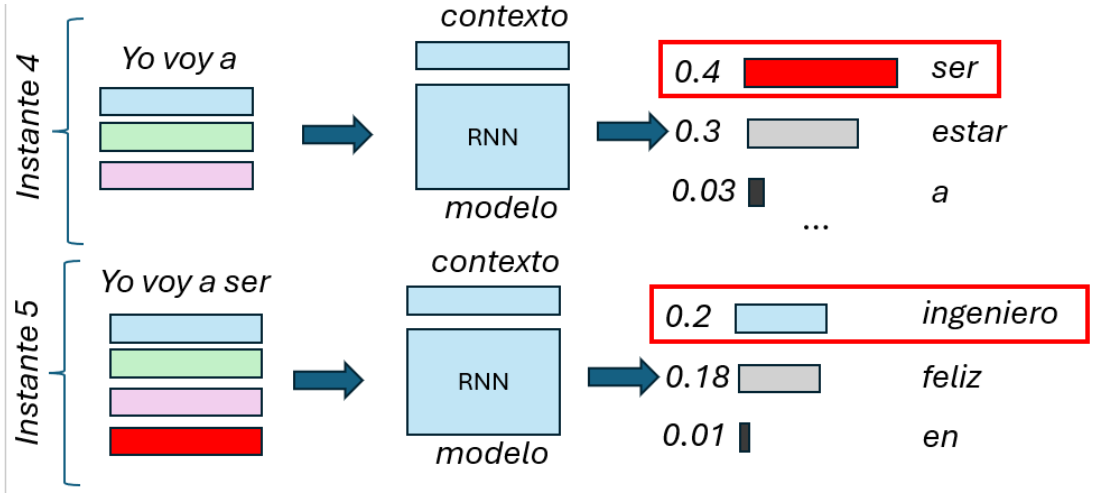
Transformers
La parte generativa (I)

- Decoder
  - **Generación de contenido** a partir de una entrada textual.
  - **Basados** en los **embeddings** del encoder (BERT, S-BERT o similares).
  - Aprender a **generar una salida** a partir de una entrada especificada (por el usuario).
- Principales aplicaciones:
  - Generación automática de texto coherente.
  - Respuesta a preguntas basadas en una entrada
  - Traducción automática entre distintos lenguajes.
  - Creación de imágenes a partir de descripciones textuales.
  - Descripción de imágenes.



Transformers
La parte generativa (II)

- Comportamiento: **Auto-regresivos**.
- Entrada
  - **Instrucción** del usuario y **tokens anteriores** generados.
- Salida
  - Capa totalmente conectada. **1 token**.
  - **Probabilidad** del siguiente token.
- Max.tokens: **Número máximo** de tokens que se pueden procesar.



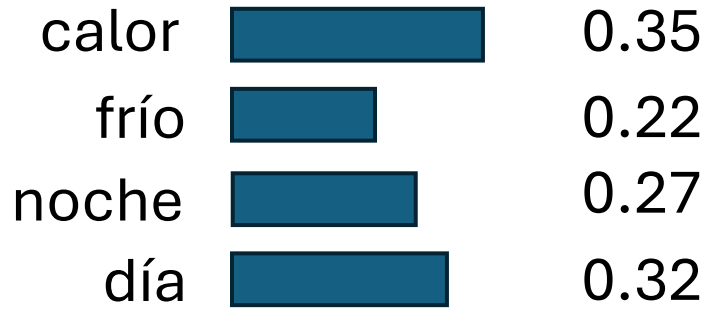
LLM
Control de la salida

- A partir de unas instrucciones y los tokens anteriores, generan el siguiente token. ¿Qué token se escoge?
  - Coger siempre el **token más probable**: Determinista.
  - Incorporar **probabilidades**.
    - **Top_p**: Considerar los tokens **hasta** que se alcance una **probabilidad acumulada** de "top_p".
    - **Temperatura**: Valores más **altos** hacen que los **tokens con menos probabilidad** puedan ser **escogidos**.

Voy a coger una chaqueta, hace...

Top_p = 0.6

Temperatura = 0.8

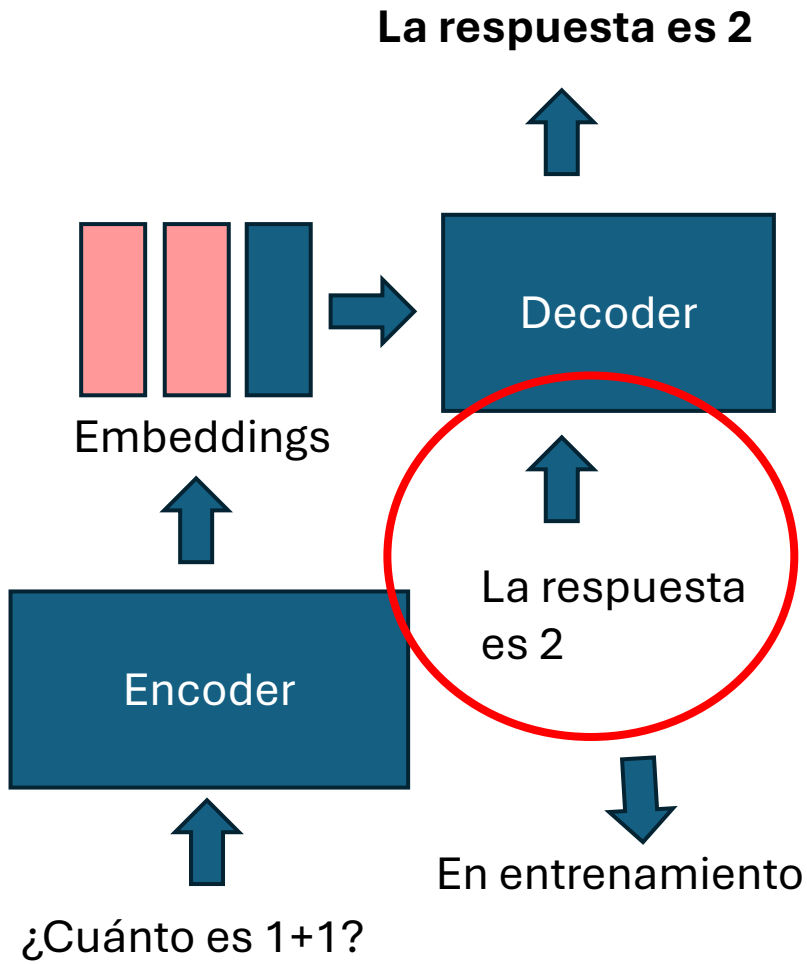


Ejercicio 9

- Usa la API de OpenAI en Databricks.
- Se va a estudiar el impacto de los parámetros “top_n” y “temperature”.
- El prompt de entrada es:
  - “Escribe una breve historia sobre un alumno de ingeniería informática en la Universidad de Almería”.
- Genera 4 combinaciones de parámetros “top_n” y “temperature”
  - top_n alto y temperatura alta
  - Top_n bajo y temperatura alta
  - Top_n alto y temperatura baja
  - Top_n bajo y temperatura baja.
- ¿Qué observas?

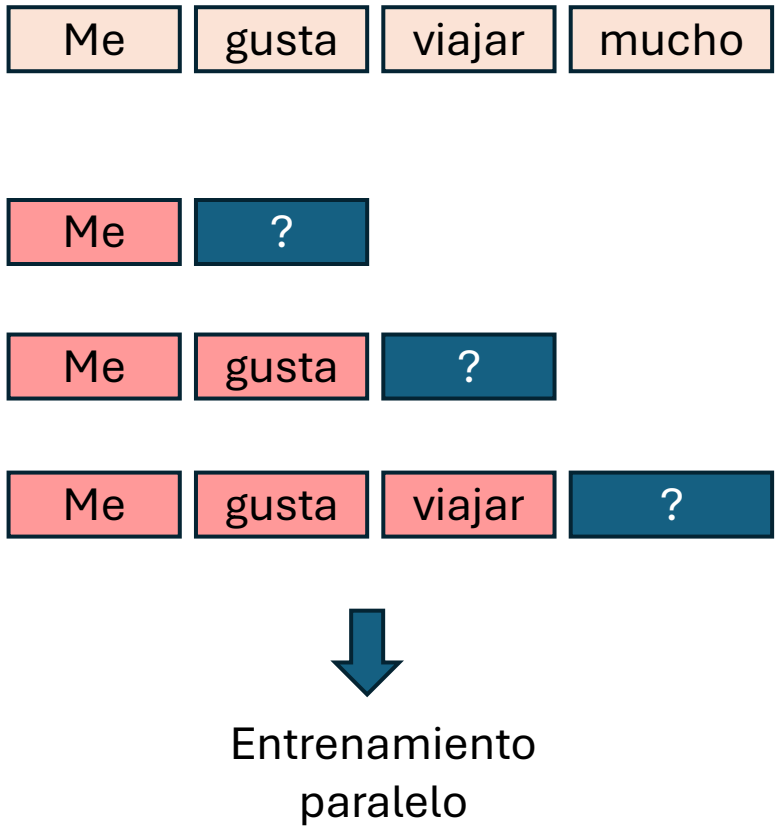
LLM
Entrenamiento

- Construcción y entrenamiento del **encoder**. **Mask Language Modelling**.
- Entrenamiento del **decoder**:
  - **Pre-entrenamiento no supervisado**
    - Usando **máscaras**, **predicción** de la **siguiente** palabra.
  - **Fine-tuning** supervisado
    - **Predicción de la siguiente palabra**.
    - Se entrena para realizar una tarea específica.
    - Algunas tareas:
      - Respuesta a preguntas.
      - Creación de resumen de una frase.
  - **Preference-tuning** (supervisado)
    - Mejora la calidad de las respuesta del modelo porque se tiene en cuenta la respuesta esperada por los humanos.



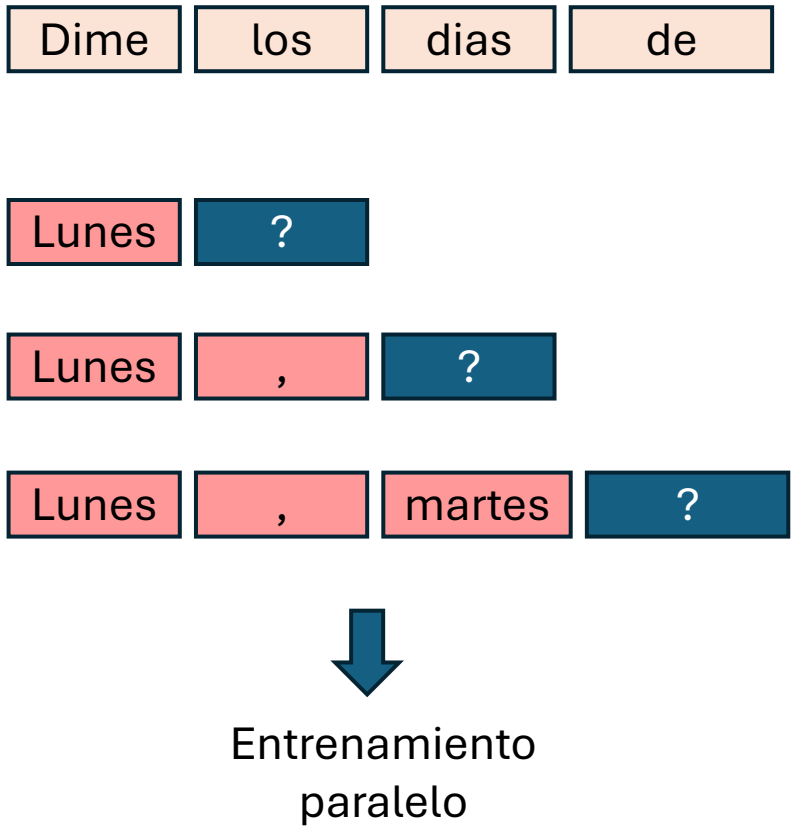
LLM
Entrenamiento – Pre-entrenamiento no supervisado

- De igual forma que los modelos **auto-regresivos**.
- “**Causal Masking Modelling**”.
- Toda la **frase**/input **a la vez**.
- Se intenta predecir el **siguiente token** a partir de la **entrada** y los tokens **anteriores**.
- **Paralelo**.
- **Dataset**: Frases.
- Resultado: Modelo capaz de **generar texto.**



LLM
Entrenamiento – Fine-tuning supervisado

- De igual forma que los modelos **auto-regresivos**.
- “**Causal Masking Modelling**”.
- El modelo aprende a **generar el contenido** que requiere la instrucción del prompt.
- No es un self-training.
- **Dataset**:
  - Entrada: Instrucción
  - Salida: El texto con el que se responde a dicha instrucción.



LLM
Entrenamiento – Preference tuning

- Se hace un **fine-tuning** con el **dataset específico** (tarea).
- A partir de una entrada, aprende a predecir la salida deseada.
- Se **evalúa cada respuesta** generada por el LLM para mejorar.
- Además, las respuestas están optimizadas para contentar al usuario.
- ¿Cómo? Sistema de **recompensa**.
  - Entrenamiento de un sistema de recompensa.
  - Entrada: Prompt y la respuesta
  - Salida: Valor
  - Inconvenientes:
    - Entrenamiento de un nuevo modelo
  - Dataset: Prompt + Buenas y malas respuestas.

LLM
Librerías

Step 1: **SFTTrainer**
Train your model on your favorite dataset

```
from trl import SFTTrainer

trainer = SFTTrainer(
    "facebook/opt-350m",
    train_dataset=dataset,
    dataset_text_field="text",
    max_seq_length=512,
)

trainer.train()
```

Step 2: **RewardTrainer**
Train a preference model on a comparison data to rank generations from the supervised fine-tuned (SFT) model

```
from trl import RewardTrainer

trainer = RewardTrainer(
    model=model,
    args=training_args,
    tokenizer=tokenizer,
    train_dataset=dataset,
)

trainer.train()
```

Step 3: **PPOTrainer**
Further optimize the SFT model using the rewards from the reward model and PPO algorithm

```
from trl import PPOConfig, PPOTrainer

trainer = PPOTrainer(
    config,
    model,
    tokenizer=tokenizer,
)

for query in dataloader:
  response = model.generate(query)
  reward = reward_model(response)
  trainer.step(query, response, reward)
```

https://huggingface.co/docs/trl/main/en/index

Ejemplo

- Archivo: CursoGenAI_UAL_10_LLM_entrenamiento.ipynb

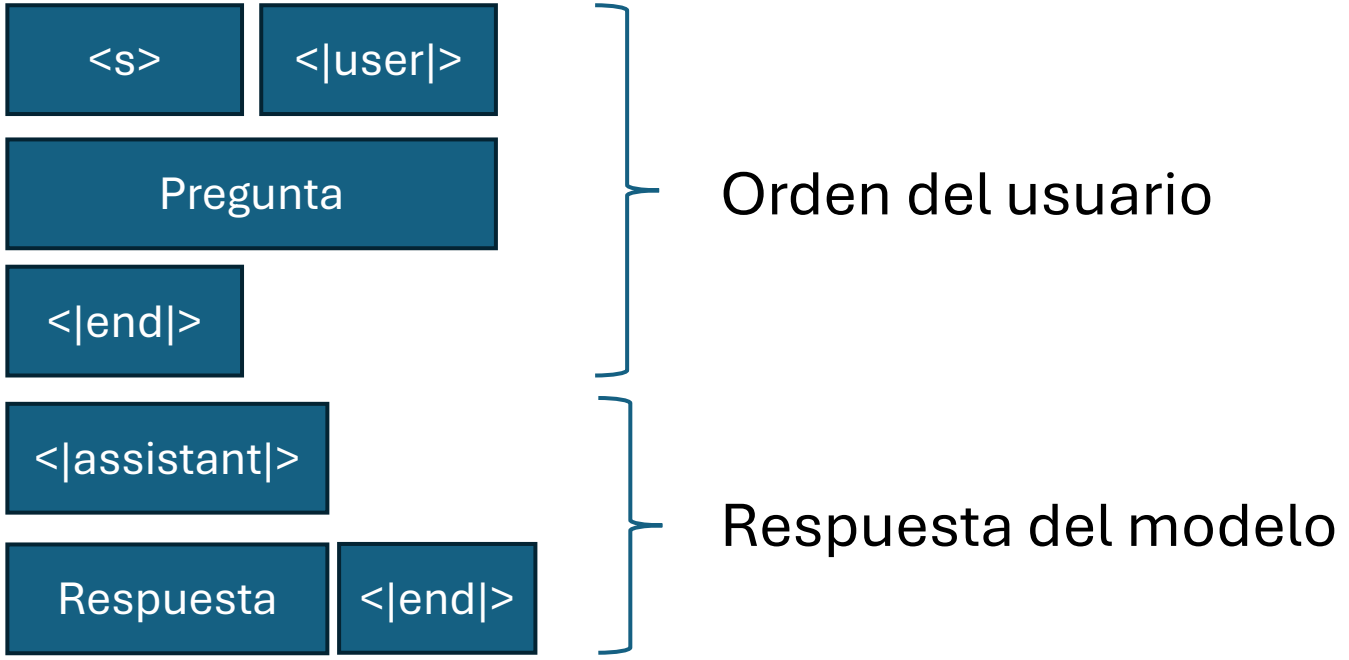
- Entrenamiento de un modelo generativo haciendo uso de Causal Masking Modelling.
- Incorporación de LoRa para hacer dicho entrenamiento menos costoso.

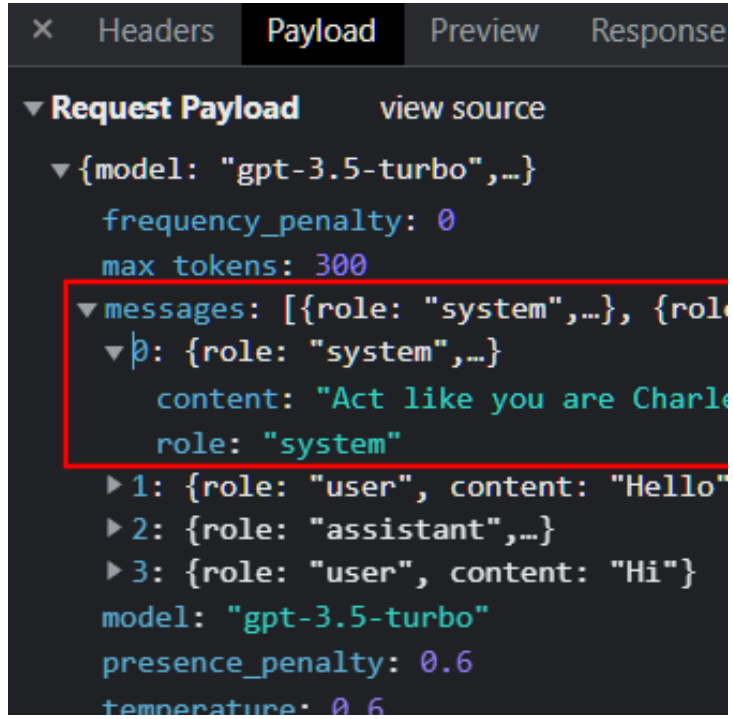
Ejercicio 8*

- Archivo: CursoGenAI_UAL_09_ejercicio_08_semanticSearchRAG.ipynb
- Incorporar un decoder en el ejercicio sobre RAG.

LLM
Control de la entrada

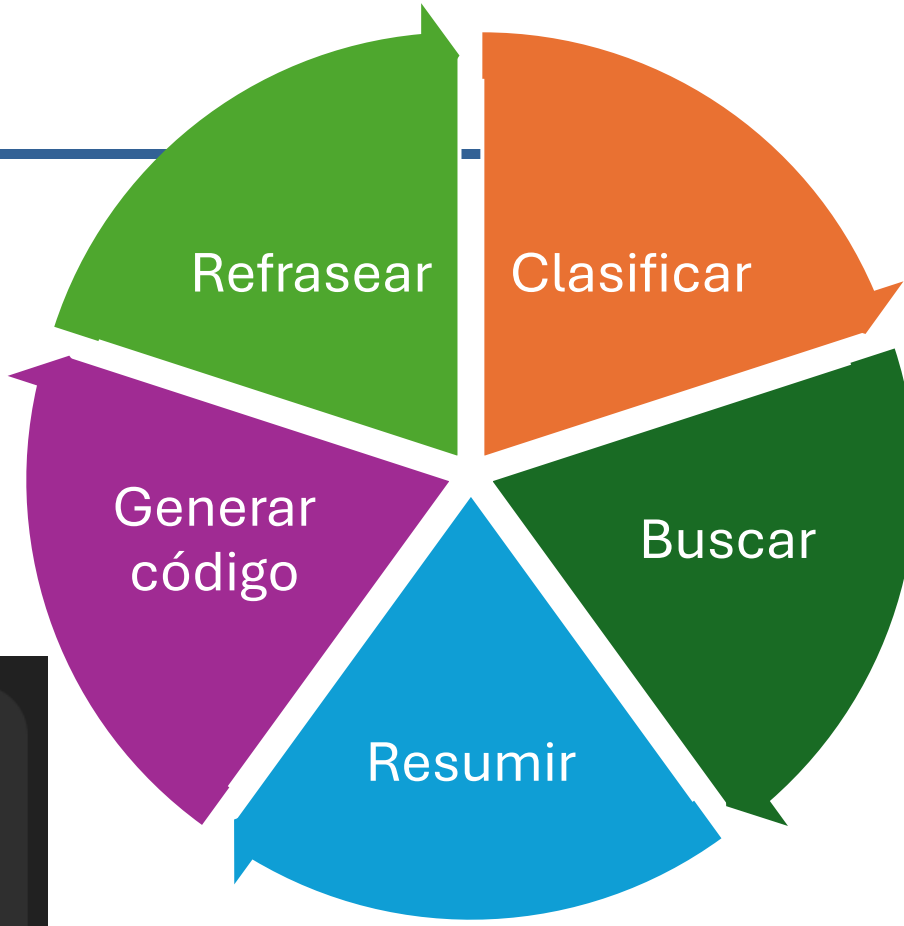
- Para dar instrucciones al modelo, se debe seguir una **plantilla** incorporando tokens específicos.
- La mayoría de las **APIs** lo generan **automáticamente**, pero en algunos casos el usuario tiene que especificarlo a mano.

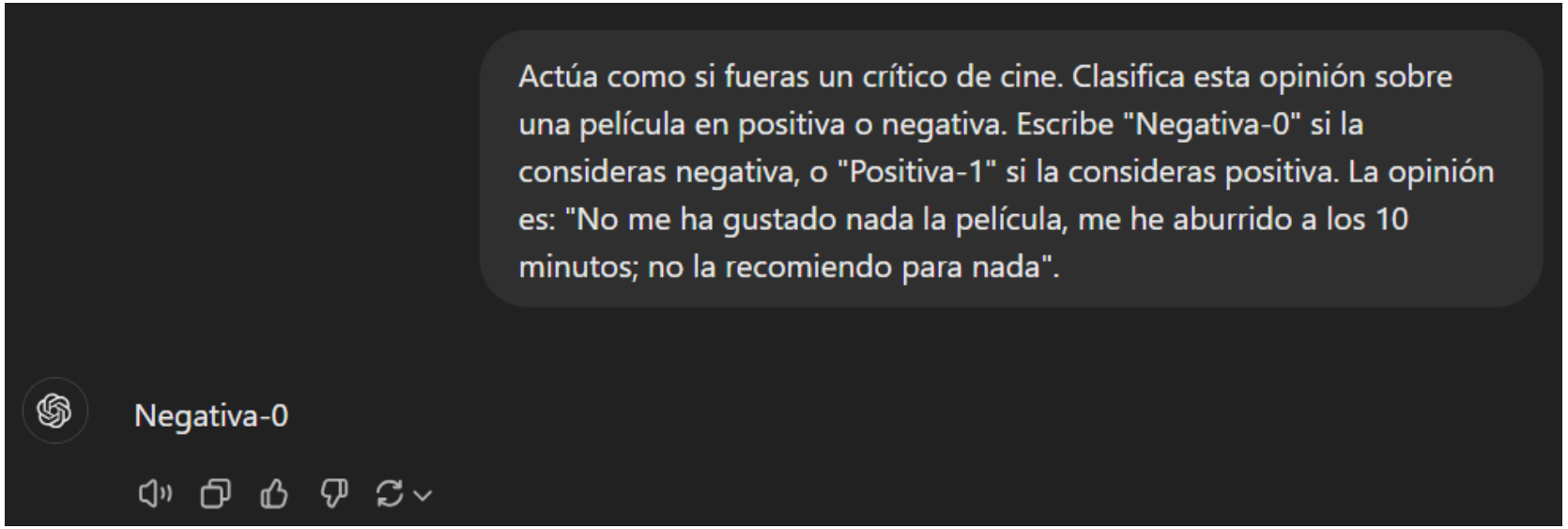




LLM
Ingeniería de prompts (I)

- Instrucciones claras.
- Especificar claramente la **salida** que se **espera**.





LLM
Ingeniería de prompts (II)

- **Identidad**:
  - El **rol** que tiene que tener el LLM. Contextualiza.
  - *Eres un médico radiólogo, experto en detección de células cancerosas en tejidos pulmonares.*
- **Instrucción**:
  - **Tarea** que debe realizar.
  - *Segmenta la imagen y muestra las partes donde se detecten las células cancerosas. Explica por qué resaltas cada parte importante.*
- **Contexto**:
  - Por qué se está realizando esta tarea. Para qué.
  - *Se va a incorporar en una unidad de radiología en el hospital para aliviar la carga de trabajo.*

LLM
Ingeniería de prompts (III)

- **Formato**
  - Formato de salida/respuesta.
  - *Proporciona una imagen en blanco y negro, pero con las partes resaltadas en colores. También, una lista con la explicación de por qué cada elemento ha sido resaltado.*
- **Audiencia**
  - Quién va a leer la respuesta o se va a beneficiar de ella.
  - *Explícalo para un médico radiólogo, con conocimientos avanzados.*
- **Tono**
  - Formato y tono del texto a generar.
  - *La explicación debe estar escrita en formato médico.*

LLM
Ingeniería de prompts (IV)

- **One-shot prompting**
  - Se proporciona al LLM una tarea y la respuesta que nosotros queremos que nos proporcione de dicha tarea.

Instrucción: Clasifica esta opinión sobre una película en "Positiva-1" si es positiva, o "Negativa-0" si es negativa.

Ejemplos:

1. **Opinión:** "Me ha encantado la película, los efectos especiales son increíbles y el guion está muy bien escrito." **Clasificación:** Positiva-1
2. **Opinión:** "La trama es predecible y los actores no están bien elegidos. No la recomendaría." **Clasificación:** Negativa-0
3. **Opinión:** "Es una película emocionante que me mantuvo pegado a la pantalla todo el tiempo. Sin duda la volvería a ver." **Clasificación:** Positiva-1

Nueva entrada: Opinión: "No me ha gustado nada la película, me he aburrido a los 10 minutos; no la recomiendo para nada."

Salida esperada:
Clasificación: Negativa-0

LLM
Ingeniería de prompts (IV)

- **Chain-prompting.**
- **Dividir** el **prompt** en **varios prompts**, en lugar de hacerlo todo a la vez.
- Se da una **especificación inicial**, y se va yendo de lo más **general**, a lo más **particular**.

Instrucción: Analiza la siguiente opinión sobre una película y proporciona un resumen, la clasificación de la opinión y una breve explicación del porqué.

Opinión: "Me ha encantado la película, la actuación fue impresionante y la trama era innovadora. Sin duda, la recomendaría a todos mis amigos."

Paso 1: Resumir la opinión.

- **Resumen:** La persona disfrutó de la película, elogiando la actuación y la originalidad de la trama.

Paso 2: Clasificar la opinión.

- **Clasificación:** Positiva-1

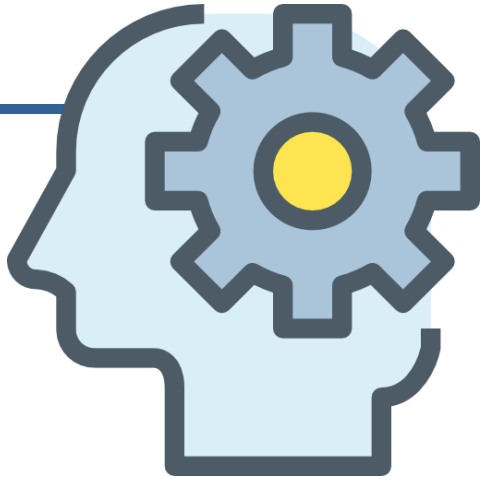
Paso 3: Explicar la clasificación.

- **Explicación:** La opinión expresa entusiasmo y satisfacción con la película, destacando aspectos positivos como la actuación y la trama, lo que justifica la clasificación como positiva.

LLM
Ingeniería de prompts (V)

- **Chain-of-thought** prompt
  - Hacer una **pregunta compleja** (puede ser un acertijo matemático).
  - Se proporciona **una respuesta**, explicando **paso a paso** cómo resolver el problema, dando la respuesta final.
  - Se expone la **pregunta definitiva** que queremos que responda.
  - El asistente va a "**razonar**" y responder de la misma forma que anteriormente se ha comentado.
    - **Tedioso**, hay que proporcionar ejemplos
- **Zero-shot** prompt
  - Se especifica: "Piensa **paso a paso**".
- Usando "**expertos**".
  - Se tienen en cuenta múltiples puntos de vista para conseguir la solución.

LLM
Opciones avanzadas



- Incorporación de **memoria**.
  - En la versión **original**, cada nuevo prompt **olvida** lo anterior.
- Enfoques
  - Añadir en cada nuevo prompt, **toda la conversación.**
    - Costoso, se alcanza el límite de prompts.
  - Añadir **X** respuestas **anteriores.**
    - Puede olvidar información más antigua.
  - Añadir un **resumen** de la conversación.
    - Costoso, pero permite mantener durante más tiempo la memoria.

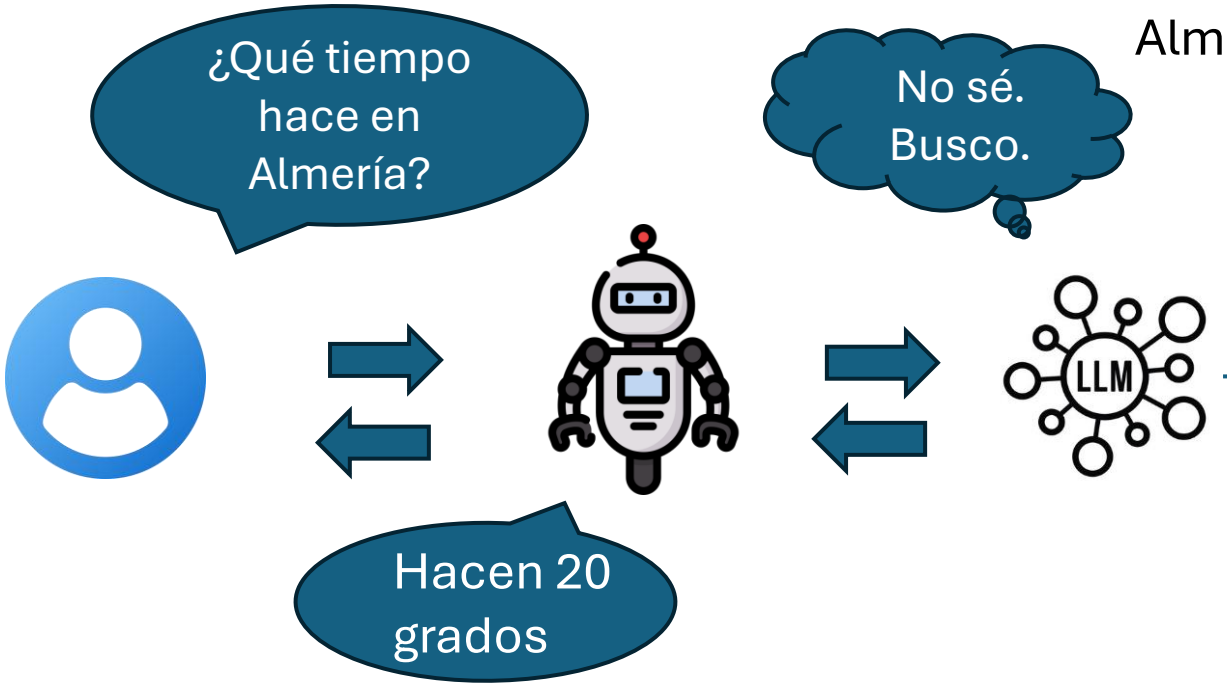
Ejercicio 10 – Creación de un chatbot con memoria.

- Usando la API de OpenAI
    - Basándote en el código del CursoGenAI_UAL_11_ejercicio_10_sin_memoria.ipynb
- Programa la siguiente funcionalidad:
    - Añadir memoria en la conversación:
        - Tener en cuenta los últimos 3 mensajes únicamente.
        - Tener en cuenta un resumen de toda la conversación anterior.

LLM
Razonar y Actuar (ReAct) (I)

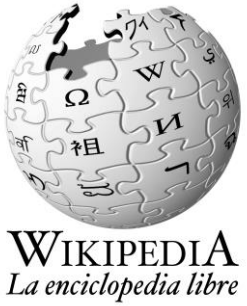
- El **LLM no** está **entrenado** en **todas** las tareas.
- Hay **herramientas específicas**:
  - Cálculo **matemático**: calculadores.
  - Búsqueda por **internet**: navegadores.
- Algunos prompts requieren usar estas herramientas para obtener **resultados precisos**.
- Se entrena un **agente** para **identificar** la necesidad de usar una **herramienta**, y usarla adecuadamente, ofreciéndole la **entrada** correspondiente y procesando su **salida**.
- El sistema por lo tanto está formado por:
  - Razonamiento: Agente/LLM
  - Actuación: Herramientas

LLM
Razonar y Actuar (ReAct) (I)



Herramientas

Búsqueda en Internet



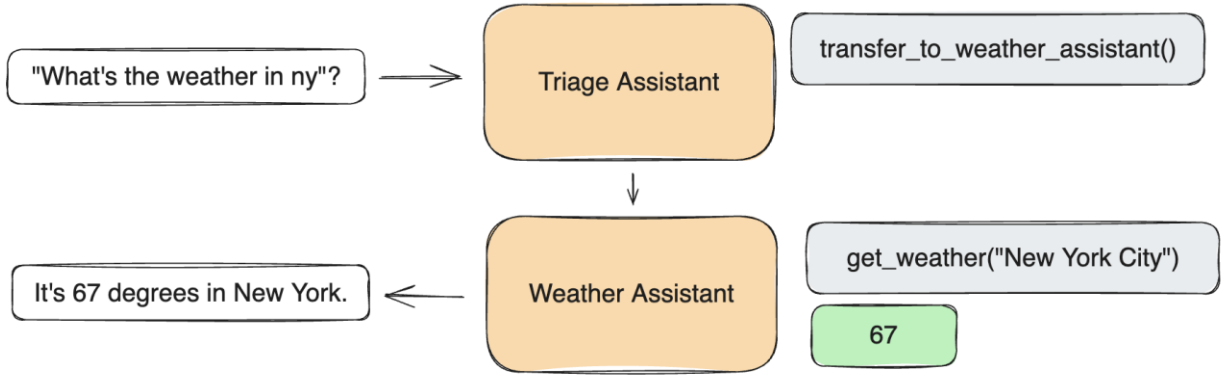
Búsqueda en wikipedia

Uso de calculadora

LLM
Razonar y Actuar (ReAct) (II)

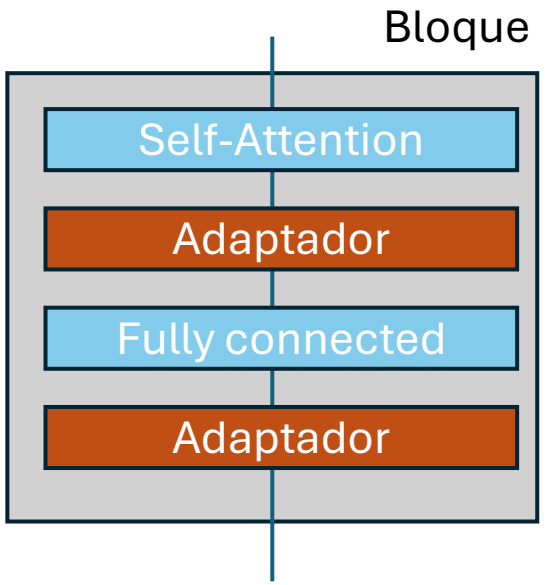
- https://github.com/openai/swarm

- Fecha de lanzamiento:
  - 12/10/2024 !!!!!



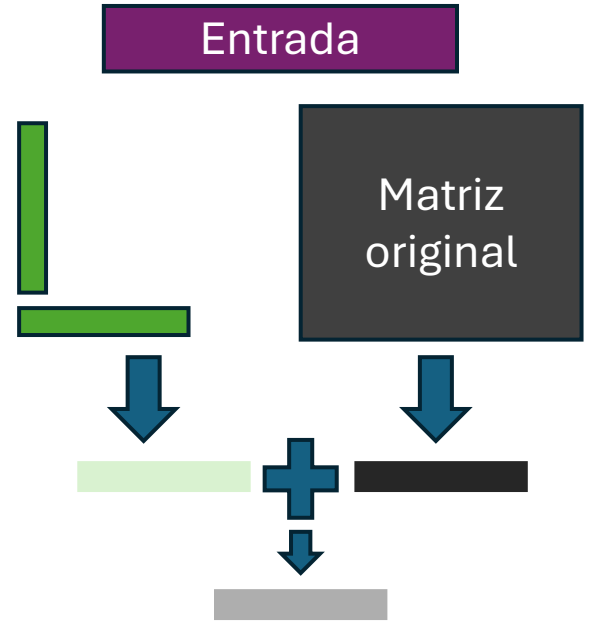
LLM
Fine-Tuning usando Parameter-Efficient (PEFT)

- Entrenamiento es **muy costoso**.
- Se añade un componente dentro de cada bloque.
  - **Adaptador** (Adapter).
- Se **congelan los pesos** del modelo. Se entrena únicamente el **adaptador**.
  - *“Fine-tuning 3.6% parameters of BERT similar to the whole”*
- Se pueden entrenar **diferentes actuadores** para **diferentes tareas**.



LLM
Fine-tuning usando LowRank

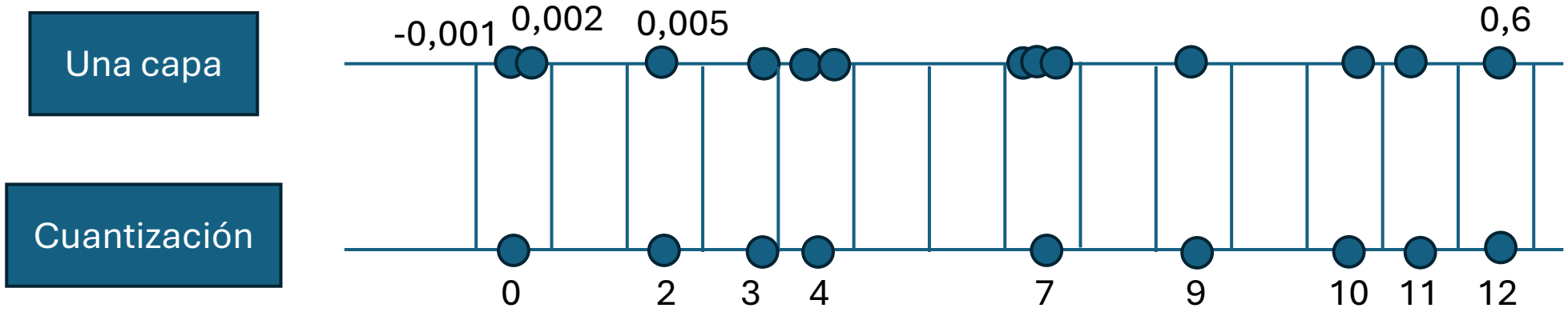
- **Aproximar** las grandes **matrices** de atención en matrices mucho más **pequeñas**.
- Se **actualizan** las matrices **pequeñas**.
- El resultado final es la **agregación** de los valores de ambas.
- **Rank**: La **dimensionalidad** de las matrices. A menor valor, mayor reducción de parámetros.



Aspecto	Sin LoRA	Con LoRA
Dimensiones de Matrices	$W : 6,000,000,000$	$A : 6,000,000,000 \times 16$
		$B : 16 \times 6,000,000,000$
Total de Parámetros	6,000,000,000	96,000
Número de Épocas	5	5
Tiempo por Época	1.5 horas	7.5 minutos
Tiempo Total	7.5 horas	37.5 minutos
Reducción de Parámetros (%)	N/A	99.9984%

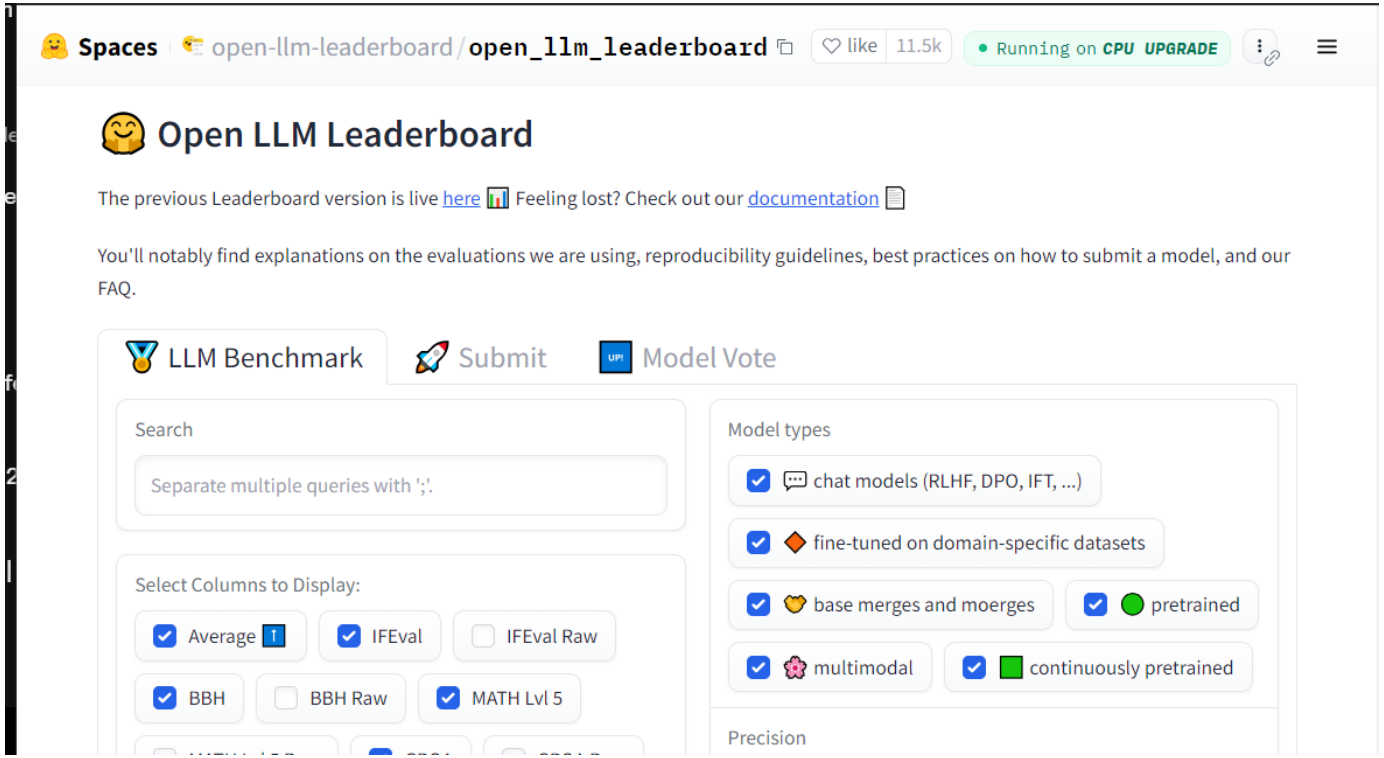
LLM
Fine-tuning usando cuantización

- La **cuantización** es una técnica para **reducir la representación numérica** de los pesos del modelo.
- En lugar de **almacenarlos** o realizar **cálculos** en **32 bits**, se usan 8 o incluso 4.
  - Reduce **memoria**.
- **TPUs** optimizadas para cálculo en 16 bits.



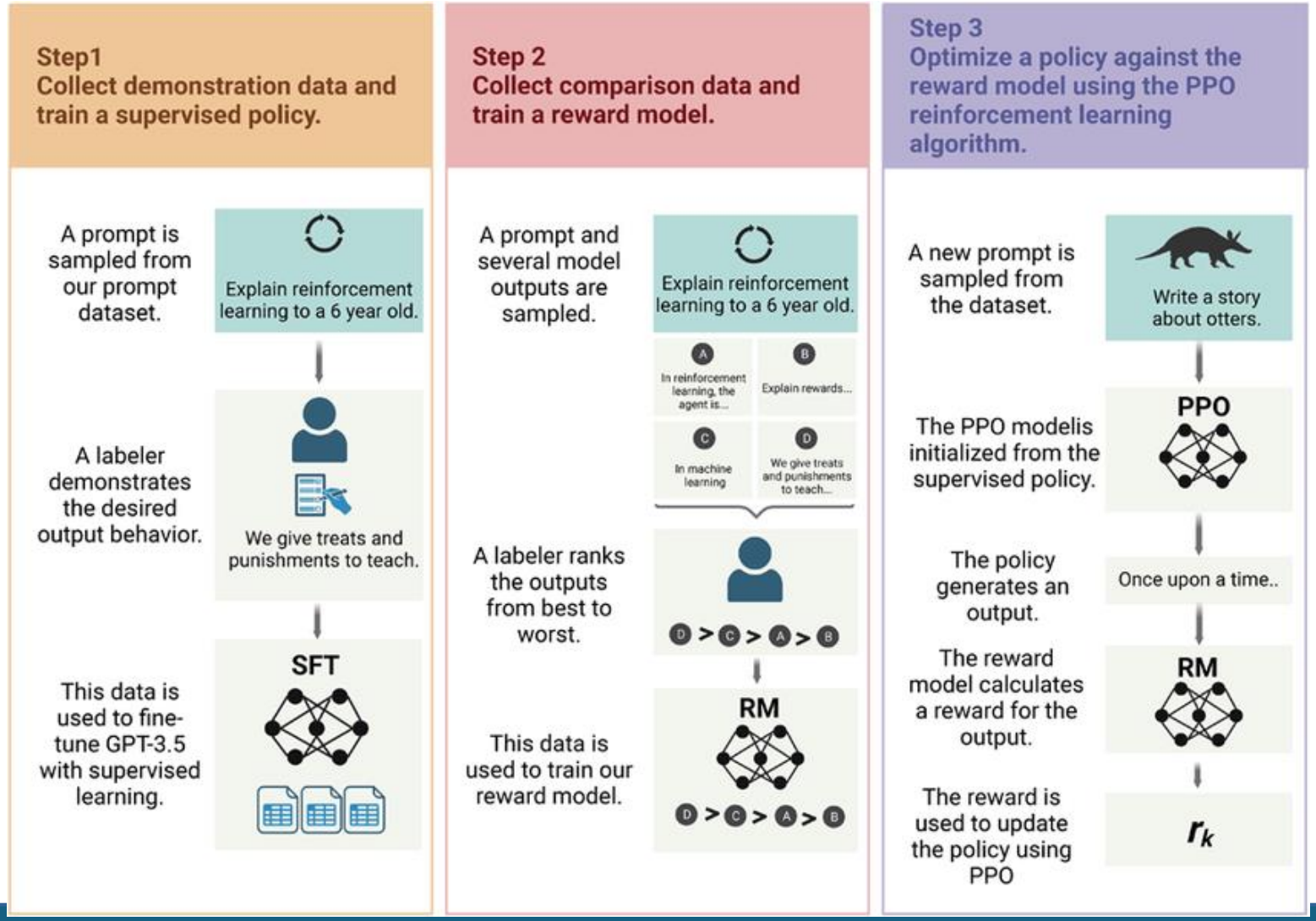
LLM
Evaluación

- Tablas de líderes
- Evaluación de los modelos
  - MMLU
  - GLUE
  - TruthfulQA
  - GSM8k
  - HellaSwag
- Evaluación humana
  - https://lmarena.ai/



https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

¿Cómo fue ChatGPT entrenado?



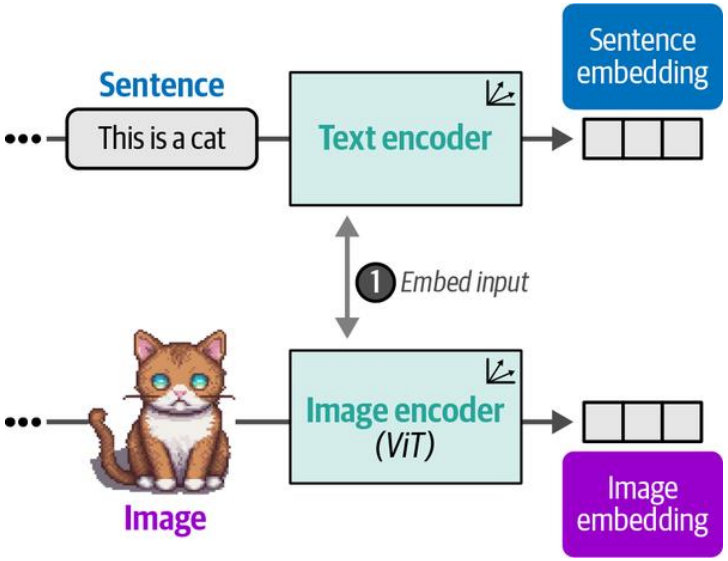
Bai, Long & Liu, Xiangfei & Su, Jia-Can. (2023). ChatGPT: The cognitive effects on learning and memory. Brain-X.

9 Modelos multimodales

UNIVERSIDAD DE ALMERÍA

Interactuando con imágenes
CLIP

- Conectar **imágenes** con **texto**: **CLIP**
- Procesar imágenes como frases:
  - **Divide la imagen** en "tokens" (grupos de píxeles).
- Entrenar un **modelo** de **embedding** que genere **embeddings** a partir de imágenes y texto en el mismo espacio.
  - Codificador de texto (SBERT).
  - Codificador de imagen (ViT).
- Comparar embeddings
  - Forzar que sean **iguales**.
  - Diferencia usando **similitud de coseno**.
- Usar **aprendizaje contrastivo**: pares iguales y diferentes.



Hands-On Large Language Models. By Jay Alammar, Maarten Grootendorst. 2024.

10 Aplicaciones

Ejercicio 11 – Uso del servicio “Azure Speech”

- Creación de un sistema para resumir audios de WhatsAPP.
  - Obtención/grabación de un mensaje de voz .mp3
  - Obtención de texto partir del audio.
  - Resumen del texto.
  - Salida de audio con el resumen del texto.
  - ¿Podrías crear un bot de Telegram/WhatsApp para que al recibir dichos mensajes, los resuma?

Ejercicio 12 – Evaluación de un TFG

- En esta aplicación, se va a usar la API de ChatGPT para evaluar un TFG dado en base a una rúbrica. Se evaluarán los diferentes aspectos contemplados en la misma, estableciendo una nota.
  - TFG: https://repositorio.ual.es/handle/10835/8008
  - Rúbrica: https://www.ual.es/application/files/5416/3211/7224/AnexoIV-RubricaEvaluacionTFG.docx

- ¿Cómo se incorporan ambos archivos?
- ¿Los acepta la API?
- ¿Cuál es su salida?
- Una posible solución se encuentra en: CursoGenAI_UAL_12_ejercicio_12_EvaluarTFG.ipynb