CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

Bases de Datos para Inteligencia Artificial

16Co2024

Trabajo Práctico N°4

Vectores

DEGANO, MYRNA LORENA N° SIU a1618 myrna.l.degano@gmail.com

abril de 2025

Índice

Trabajo Práctico Nro. 4 - Vectores

CEIA - Bases de datos para inteligencia artificial.
Myrna Degano (a1618)

Set up

[123]:
```
# Importar librerías
from sentence_transformers import SentenceTransformer
import chromadb
from chromadb.config import Settings
import os
import numpy as np
import uuid
```

Modelo de embeddings

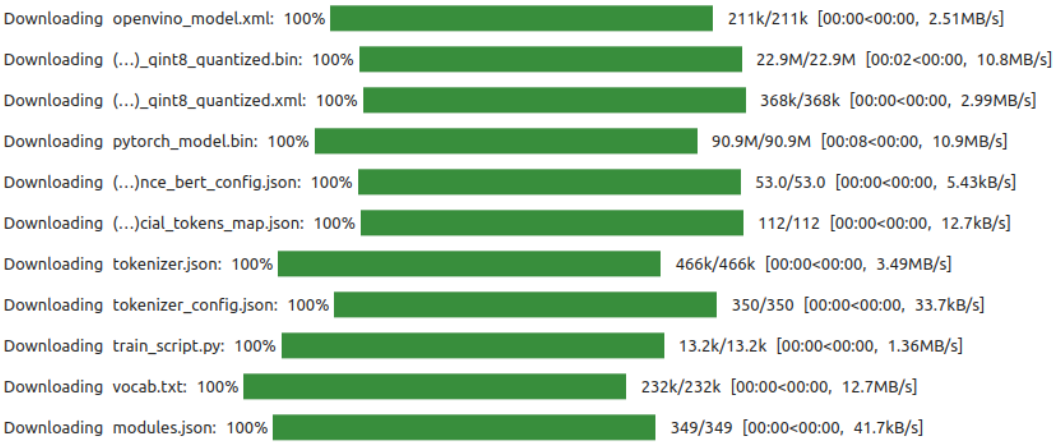
[124]:
```
# Modelo de embeddings
model_name = "sentence-transformers/all-MiniLM-L6-v2"

cache_dir = "./model_cache"  # Guardar el modelo localmente para evitar sucesivos downloads

# Cargar el modelo y guardarlo en la carpeta local
# model = SentenceTransformer(model_name)
model = SentenceTransformer(model_name, cache_folder=cache_dir)
```

```
Downloading .gitattributes: 100% 1.23k/1.23k [00:00<00:00, 113kB/s]
Downloading config.json: 100% 190/190 [00:00<00:00, 19.6kB/s]
Downloading README.md: 100% 10.5k/10.5k [00:00<00:00, 620kB/s]
Downloading config.json: 100% 612/612 [00:00<00:00, 31.5kB/s]
Downloading (…)ce_transformers.json: 100% 116/116 [00:00<00:00, 12.5kB/s]
Downloading data_config.json: 100% 39.3k/39.3k [00:00<00:00, 1.85MB/s]

Downloading model.safetensors: 100% 90.9M/90.9M [00:09<00:00, 10.6MB/s]
Downloading model.onnx: 100% 90.4M/90.4M [00:08<00:00, 11.0MB/s]
Downloading model_O1.onnx: 100% 90.4M/90.4M [00:08<00:00, 11.0MB/s]
Downloading model_O2.onnx: 100% 90.3M/90.3M [00:08<00:00, 10.1MB/s]
Downloading model_O3.onnx: 100% 90.3M/90.3M [00:08<00:00, 11.1MB/s]
Downloading model_O4.onnx: 100% 45.2M/45.2M [00:04<00:00, 11.3MB/s]
Downloading model_qint8_arm64.onnx: 100% 23.0M/23.0M [00:02<00:00, 11.4MB/s]
Downloading model_qint8_arm64.onnx: 100% 23.0M/23.0M [00:02<00:00, 11.2MB/s]
Downloading model_qint8_arm64.onnx: 100% 23.0M/23.0M [00:02<00:00, 10.8MB/s]
Downloading model_quint8_avx2.onnx: 100% 23.0M/23.0M [00:02<00:00, 10.4MB/s]
Downloading openvino_model.bin: 100% 90.3M/90.3M [00:08<00:00, 10.3MB/s]
```



sentence-transformers es una librería que se utiliza para la generación de embeddings de oraciones. Un embedding es una representación numérica (vectorial) de una oración o texto.

all-MiniLM-L6-v2 es un modelo pequeño y eficiente que genera embeddings de texto. Está basado en la arquitectura MiniLM y tiene 6 capas (L6). Es más rápido y eficiente en comparación con otros modelos más grandes, pero aún mantiene un buen rendimiento.

Se carga el modelo preentrenado desde el nombre especificado en model_name usando la clase **SentenceTransformer** de la librería sentence-transformers.

(En futuras ejecuciones, el modelo se cargará directamente desde la carpeta de caché sin necesidad de ser descargado de nuevo).

Base de datos ChromaDB

[125]:
```
# Inicializar ChromaDB con persistencia
client = chromadb.PersistentClient(path="./chroma_store")
collection = client.get_or_create_collection(name="documentos_ia")
```

chromadb es una base de datos de vectores (embeddings) diseñada para facilitar la gestión y la búsqueda eficiente de grandes volúmenes de datos vectorizados.

PersistentClient es una clase que permite interactuar con la base de datos de manera que los datos sean guardados de forma persistente en disco. Si no se usa persistencia, los datos se mantienen solo en la memoria hasta que se termine la sesión.

El parámetro **path="./chroma_store"** define la ubicación del directorio en el cual se almacenarán los vectores y los índices. En este caso, la base de datos se guardará en una carpeta llamada chroma_store del directorio actual.

Funciones auxiliares

get_embeddings

[126]:
```
def get_embeddings(text):
    return model.encode(text).tolist()
```

cargar_documentos

[127]:
```python
def cargar_documentos():
    docs_path = "docs/"
    documentos = []
    metadatos = []
    ids = []

    for filename in os.listdir(docs_path):
        if filename.endswith(".txt"):
            with open(os.path.join(docs_path, filename), 'r', encoding="utf-8") as f:
                text = f.read().strip()
                chunks = [text[i:i+500] for i in range(0, len(text), 500)]

                for i, chunk in enumerate(chunks):
                    documentos.append(chunk)
                    metadatos.append({"source": filename})
                    ids.append(str(uuid.uuid4()))  # ID único

    return documentos, metadatos, ids
```

crear_base_datos

[128]:
```python
def crear_base_datos():
    docs, metas, ids = cargar_documentos()
    embeddings = [get_embeddings(d) for d in docs]
    collection.add(documents=docs, embeddings=embeddings, metadatas=metas, ids=ids)
    print("✅ Base de datos creada con éxito.")
```

Creación de la DB

[129]:
```python
crear_base_datos()
```

```
✅ Base de datos creada con éxito.
```

[130]:
```python
results = collection.get(include=["embeddings", "documents", "metadatas"])
```

La variable **results** almacenará un diccionario o estructura de datos (dependiendo de cómo ChromaDB esté configurado) que contiene tres claves:

- embeddings: Los vectores numéricos correspondientes a cada documento en la colección.
- documents: Los documentos originales (por ejemplo, los textos) almacenados.
- metadatas: Los metadatos asociados a esos documentos.

[131]:
```python
for id, content, metadata, embedding in zip(results["ids"],
    results["documents"], results["metadatas"], results["embeddings"]):

    print(f"ID: {id}, Contenido: {content[:100]} ",
          f"Metadata: {metadata}", f"Embedding: {embedding[:5]}...")

    break
```

```
ID: 567e887f-ff0f-4396-831b-ce66659b9094, Contenido: La ciudad de Córdoba, ubicada en el corazón de Argentina, es co
nocida por su rica historia colonial   Metadata: {'source': 'doc3.txt'} Embedding: [0.07863820344209671, -0.02900855
988264084, -0.048923082649707794, 0.007204551715403795, -0.03373530134558678]...
```

Operaciones (CRUD)

Crear documento

[132]:
```python
def create_example(new_doc):

    """Añadir un documento nuevo"""
    embedding = get_embeddings(new_doc)
    doc_id = str(uuid.uuid4())
    metadata = {"source": "nuevo_documento.txt"}

    collection.add(
        documents=[new_doc],
        embeddings=[embedding],
        metadatas=[metadata],
        ids=[doc_id]
    )
    print(f"✅ Documento añadido con ID: {doc_id}")

    return doc_id
```

Consultar por tema

[133]:
```python
def read_example(query, n=1):

    """Consultar documentos similares"""
    query_emb = get_embeddings(query)
    results = collection.query(
        query_embeddings=[query_emb],
        n_results=n
    )

    if results:
        for id_list, content_list, metadata_list in zip(results["ids"], results["documents"], results["metadatas"]):
            for id, content, metadata in zip(id_list, content_list, metadata_list):
                print("-----------------------------------------------------------\n")
                print(f"ID: {id}")
                print(f"Contenido: {content}")
                print(f"Metadata: {metadata}")
    else:
        print("⚠️ No se encontraron resultados.")

    return results
```

Buscar documento por id

[134]:
```python
def get_documents_by_id(doc_id):

    """Obtener un documento por ID"""
    results = collection.get(ids=[doc_id], include=["documents", "metadatas", "embeddings"])

    if results["documents"]:
        print(f"📄 Documento: {results['documents'][0]}")
        print(f"📁 Fuente: {results['metadatas'][0]['source']}")
    else:
        print("⚠️ Documento no encontrado.")

    return results
```

Actualizar documento por id

[135]:
```python
def update_example(old_id, new_text):

    """Actualizar un documento (eliminar y reinsertar)"""
    collection.delete(ids=[old_id])

    new_id = str(uuid.uuid4())
    embedding = get_embeddings(new_text)
    metadata = {"source": "documento_actualizado.txt"}

    collection.add(
        documents=[new_text],
        embeddings=[embedding],
        metadatas=[metadata],
        ids=[new_id]
    )

    print(f"♻️ Documento actualizado. Nuevo ID: {new_id}")

    return new_id
```

Eliminar documento por id

[136]:
```python
def delete_example(doc_id):

    """Eliminar por ID"""
    collection.delete(ids=[doc_id])

    print(f"🗑️ Documento con ID '{doc_id}' eliminado.")
```

Ejercicios

Ejercicio 1: Crear y Consultar

Crea un documento nuevo con el texto: "La gravedad es una fuerza fundamental en el universo".
Realiza una consulta para encontrar documentos similares a la frase: "Fuerzas naturales que gobiernan el cosmos".

¿Qué tipo de operación CRUD usaste primero?
¿Cuál es el ID del documento creado? (Usa collection.peek()).
Explica por qué el documento aparece como coincidencia en la consulta.

Para crear el documento se utiliza la función **create_example** que usa el método **add** de la colección para añadir el documento a ChromaDB.
Esto añadirá el nuevo documento a la base de datos, junto con su embedding, metadatos e ID.

[137]:
```
doc_id = create_example("La gravedad es una fuerza fundamental en el universo")
```

```
✅ Documento añadido con ID: 703c9472-c004-4c04-ba8b-42cb131bd2c3
```

[138]:
```
similar_docs = read_example("Fuerzas naturales que gobiernan el cosmos")
```

```
------------------------------------------------------------

ID: 1c5a4c0f-071b-4f55-b2d0-56d45352cd4b
Contenido: caudal y la belleza de su entorno selvático atraen a miles de visitantes cada año.

La Antártida Argentina es un territorio reclamado por el país, donde se realizan importantes investigaciones científ
icas en diversas áreas, desde la glaciología hasta la biología marina.

La astronomía tiene un lugar destacado en Argentina, con observatorios de renombre internacional como el Observatori
o Pierre Auger, dedicado al estudio de los rayos cósmicos de ultra alta energía.

La biotecnología es un sector
Metadata: {'source': 'doc3.txt'}
```

[139]:
```
similar_docs = read_example("Fuerzas naturales que gobiernan el cosmos", 5)
```

```
------------------------------------------------------------

ID: 1c5a4c0f-071b-4f55-b2d0-56d45352cd4b
Contenido: caudal y la belleza de su entorno selvático atraen a miles de visitantes cada año.

La Antártida Argentina es un territorio reclamado por el país, donde se realizan importantes investigaciones científ
icas en diversas áreas, desde la glaciología hasta la biología marina.

La astronomía tiene un lugar destacado en Argentina, con observatorios de renombre internacional como el Observatori
o Pierre Auger, dedicado al estudio de los rayos cósmicos de ultra alta energía.

La biotecnología es un sector
Metadata: {'source': 'doc3.txt'}
------------------------------------------------------------


ID: ccf24687-28c4-4bb2-83a9-06d04a1a9524
Contenido: territorio y sus influencias culturales.

La ciencia y la tecnología también tienen un lugar importante en Argentina. Investigadores y científicos trabajan en
diversas áreas, desde la biotecnología hasta la astronomía, contribuyendo al avance del conocimiento.

La música folklórica argentina, con sus ritmos y melodías características, cuenta historias de la tierra y sus habit
antes. Instrumentos como la guitarra, el charango y el bombo legüero son protagonistas de estas expresiones cultural
es.

E
Metadata: {'source': 'doc1.txt'}
------------------------------------------------------------
```

```
ID: 7e3f0a57-56b5-4906-92fe-36cedec9a3af
Contenido: r la paz, el desarrollo y la cooperación.

El respeto por los derechos humanos es un valor fundamental en la sociedad argentina. Se han realizado importantes a
vances en la protección y promoción de los derechos civiles, políticos, económicos, sociales y culturales.

La diversidad cultural es una característica enriquecedora de Argentina, resultado de la inmigración de diferentes p
artes del mundo a lo largo de su historia. Esta mezcla de culturas se refleja en la gastronomía, la música, el arte
y
Metadata: {'source': 'doc2.txt'}
------------------------------------------------------------

ID: 703c9472-c004-4c04-ba8b-42cb131bd2c3
Contenido: La gravedad es una fuerza fundamental en el universo
Metadata: {'source': 'nuevo_documento.txt'}
------------------------------------------------------------

ID: adfd4bd3-a9c5-4ce8-b7c0-6e5956b2bbd5
Contenido: numerosos sitios históricos y culturales que son protegidos y valorados.
Metadata: {'source': 'doc2.txt'}
```

El nuevo documento está en la lista de los mejores cinco, significa que es considerado relevante para la consulta debido a que los embeddings son semánticamente similares.
En otras palabras, el documento es temáticamente cercano a lo que se ingresó en la búsqueda.
Ese es el objetivo de usar embeddings en lugar de búsquedas exactas de texto.

[140]:
```python
# Obtener un resumen de la colección
summary = collection.peek()
```

[141]:
```python
for key, value in summary.items():
    print (key, len(value))
```

```
ids 10
embeddings 10
metadatas 10
documents 10
```

[142]:
```python
# Obtener la cantidad de documentos en la colección
print(f"La colección tiene {collection.count()} documentos.")
```

```
La colección tiene 34 documentos.
```

[143]:
```python
results = collection.get(include=["documents", "embeddings", "metadatas"])
print(len(results['documents']))
```

```
34
```

[144]:
```python
print(f'Id del último documento añadido: {doc_id}')
```

```
Id del último documento añadido: 703c9472-c004-4c04-ba8b-42cb131bd2c3
```

Ejercicio 2: Actualizar y Eliminar

Crea un documento con el texto: "Redes neuronales imitan procesos cerebrales".
Encuentra su ID, luego actualízalo por: "Deep Learning es un subcampo de IA con redes neuronales profundas".

Elimina el documento original usando el mismo ID.
¿Qué ocurre si intentas actualizar un documento eliminado?
¿Cómo se garantiza la integridad de los datos en las operaciones?

---

[145]:
```
doc_id = create_example("Redes neuronales imitan procesos cerebrales")
```
```
✅ Documento añadido con ID: 3447bb5f-a79c-4ce5-a1b7-5f4200c160a3
```

[146]:
```
new_doc_id = update_example(doc_id, "Deep Learning es un subcampo de IA con redes neuronales profundas")
```
```
♻️ Documento actualizado. Nuevo ID: b75708ae-2688-433d-9446-956bea38048e
```

La función **update_example** realiza la eliminación del embedding original y la creación de uno nuevo, por lo que, se genera un nuevo id.
Lo ideal sería reutilizar el mismo id en lugar de crear uno nuevo (por ejemplo, con la operación de collection.upsert()). Esto ayuda a mantener la coherencia en las búsquedas y en las relaciones entre los documentos y sus embeddings.

[147]:
```
delete_example(doc_id)
```
```
Delete of nonexisting embedding ID: 3447bb5f-a79c-4ce5-a1b7-5f4200c160a3
Delete of nonexisting embedding ID: 3447bb5f-a79c-4ce5-a1b7-5f4200c160a3
🗑️ Documento con ID '3447bb5f-a79c-4ce5-a1b7-5f4200c160a3' eliminado.
```

[148]:
```
delete_example(new_doc_id)
```
```
🗑️ Documento con ID 'b75708ae-2688-433d-9446-956bea38048e' eliminado.
```

Las operaciones de eliminación y actualización son atómicas en ChromaDB.

Ejercicio 3: Errores y Validación

Intenta eliminar un documento con el ID "no_existe".
Crea un nuevo documento sin usar get_embeddings() (ejemplo: enviar un array vacío).

¿Qué errores ocurren en cada caso?
¿Cómo prevenir estos problemas?

---

[149]:
```
delete_example("no_existe")
```
```
Delete of nonexisting embedding ID: no_existe
Delete of nonexisting embedding ID: no_existe
🗑️ Documento con ID 'no_existe' eliminado.
```

La eliminación no genera error, la operación se completa sin afectar la base de datos.
Sin embargo, arroja el mensaje a manera de warning de que ese ID no existe.

La opción sería prevenir la eliminación de documentos que no existen, verificar previamente si el documento existe antes de intentar eliminarlo, teniendo así control explícito de la operación y dando además la posibilidad de personalizar el feedback.

Por ejemplo:

[150]:
```
def delete_example_ok (doc_id):

    # Verificar que el documento existe antes de eliminarlo
    documento = collection.get(ids=[doc_id])

    if documento['ids']:
        collection.delete(ids=[doc_id])
        print(f"🗑️ Documento con ID '{doc_id}' eliminado.")
        ok = True
    else:
        print(f"⚠️ El documento con ID '{doc_id}' no existe.")
        ok = False

    return ok
```

[151]:
```
flag = delete_example_ok("no_existe")
```
```
⚠️ El documento con ID 'no_existe' no existe.
```

[152]:
```
doc_id = create_example("")
```
```
✅ Documento añadido con ID: 826633d4-fff4-42f4-b7a5-2059628749a2
```

Para añadir un documento sin *get_embeddings*:

[153]:
```python
def create_example_without_emb(new_doc, emb=""):

    """Añadir un documento nuevo con string vacío en los embeddings"""
    embedding = emb
    doc_id = str(uuid.uuid4())
    metadata = {"source": "nuevo_documento.txt"}

    collection.add(
        documents=[new_doc],
        embeddings=[embedding],
        metadatas=[metadata],
        ids=[doc_id]
    )
    print(f"✅ Documento añadido con ID: {doc_id}")

    return doc_id
```

[155]:
```python
create_example_without_emb("nuevo documento de prueba", "")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[155], line 1
----> 1 create_example_without_emb("nuevo documento de prueba", "")

Cell In[153], line 8, in create_example_without_emb(new_doc, emb)
      5 doc_id = str(uuid.uuid4())
      6 metadata = {"source": "nuevo_documento.txt"}
----> 8 collection.add(
      9     documents=[new_doc],
     10     embeddings=[embedding],
     11     metadatas=[metadata],
     12     ids=[doc_id]
     13 )
     14 print(f"✅ Documento añadido con ID: {doc_id}")
     16 return doc_id

File ~/anaconda3/envs/BDIA_TP4/lib/python3.11/site-packages/chromadb/api/models/Collection.py:95, in Collection.add
(self, ids, embeddings, metadatas, documents)
     69 def add(
     70     self,
     71     ids: OneOrMany[ID],
   (...)     74     documents: Optional[OneOrMany[Document]] = None,
     76     """Add embeddings to the data store.
     77     Args:
     78         ids: The ids of the embeddings you wish to add
   (...)     92     """
     93     ---> 95     ids, embeddings, metadatas, documents = self._validate_embedding_set(
     96         ids, embeddings, metadatas, documents
     97     )
     99     self._client._add(ids, self.id, embeddings, metadatas, documents)

File ~/anaconda3/envs/BDIA_TP4/lib/python3.11/site-packages/chromadb/api/models/Collection.py:344, in Collection._va
lidate_embedding_set(self, ids, embeddings, metadatas, documents, require_embeddings_or_documents)
    329 def _validate_embedding_set(
    330     self,
    331     ids: OneOrMany[ID],
   (...)    340     Optional[List[Document]],
    341 ):
    342     ids = validate_ids(maybe_cast_one_to_many(ids))
    343     embeddings = (
--> 344         validate_embeddings(maybe_cast_one_to_many(embeddings))
    345         if embeddings is not None
    346         else None
    347     )
```

```
    348     metadatas = (
    349         validate_metadatas(maybe_cast_one_to_many(metadatas))
    350         if metadatas is not None
    351         else None
    352     )
    353     documents = maybe_cast_one_to_many(documents) if documents is not None else None

File ~/anaconda3/envs/BDIA_TP4/lib/python3.11/site-packages/chromadb/api/types.py:337, in validate_embeddings(embedd
ings)
    333     raise ValueError(
    334         f"Expected embeddings to be a list with at least one item, got {embeddings}"
    335     )
    336 if not all([isinstance(e, list) for e in embeddings]):
--> 337     raise ValueError(
    338         f"Expected each embedding in the embeddings to be a list, got {embeddings}"
    339     )
    340 for embedding in embeddings:
    341     if not all([isinstance(value, (int, float)) for value in embedding]):

ValueError: Expected each embedding in the embeddings to be a list, got ['']
```

Esta operación da error.
ChromaDB lanza el error **"ValueError: Expected each embedding in the embeddings to be a list, got ['']"** porque un embedding vacío no es válido.
Los embeddings deben ser vectores numéricos.

Para evitar insertar embeddings vacíos, se puede hacer una validación previa antes de intentar la inserción para asegurarse de que el vector no esté vacío y que tenga la longitud correcta.

[156]:
```python
def create_example_without_emb_ok(new_doc, emb=""):

    """Añadir un documento nuevo con string vacío en los embeddings y validación"""
    embedding = emb

    if len(embedding) > 0:
        doc_id = str(uuid.uuid4())
        metadata = {"source": "nuevo_documento.txt"}

        collection.add(
            documents=[new_doc],
            embeddings=[embedding],
            metadatas=[metadata],
            ids=[doc_id]
        )
        print(f"✅ Documento añadido con ID: {doc_id}")
    else:
        print(f"⚠️ El embedding no puede ser vacío.")
```

[157]:
```python
create_example_without_emb_ok("nuevo documento de prueba", "")
```

```
⚠️ El embedding no puede ser vacío.
```

Ejercicio 4: Consultas por Tema

Crea tres documentos:
"Python es ideal para ciencia de datos"
"JavaScript maneja interacciones web dinámicas"
"Java se usa en aplicaciones empresariales"
Realiza una consulta por la frase: "Lenguajes backend y frontend".

¿Cuáles documentos aparecen como coincidencias?
Analiza cómo las palabras clave influyen en los resultados.

---

[158]:
```
doc1_id = create_example("Python es ideal para ciencia de datos")
```
```
✅ Documento añadido con ID: 2c06af2d-069c-4142-99f8-7d7913e9968e
```

[159]:
```
doc2_id = create_example("JavaScript maneja interacciones web dinámicas")
```
```
✅ Documento añadido con ID: 19c60dc8-7bae-472c-baf9-9ca5e96e6e24
```

[160]:
```
doc3_id = create_example("Java se usa en aplicaciones empresariales")
```
```
✅ Documento añadido con ID: 1e135cd1-a229-46fd-b364-21b7816b82e8
```

[161]:
```
x1 = read_example("Lenguajes backend y frontend", n=1)
```
```
------------------------------------------------------------

ID: 02d8f891-b6ed-448c-bd87-b04e181e088d
Contenido: lenguaje natural (PNL) es un campo de la inteligencia artificial que se centra en la comprensión y genera
ción del lenguaje humano. Los embeddings son una herramienta fundamental en muchas tareas de PNL.

Los sistemas de recomendación utilizan bases de datos vectoriales para encontrar elementos similares a los que un us
uario ha mostrado interés previamente. Esto permite ofrecer sugerencias personalizadas de productos, películas o más
ica.

La detección de anomalías se basa en la identificación de
Metadata: {'source': 'doc1.txt'}
```

[162]:
```
x5 = read_example("Lenguajes backend y frontend", n=5)
```
```
------------------------------------------------------------

ID: 02d8f891-b6ed-448c-bd87-b04e181e088d
Contenido: lenguaje natural (PNL) es un campo de la inteligencia artificial que se centra en la comprensión y genera
ción del lenguaje humano. Los embeddings son una herramienta fundamental en muchas tareas de PNL.

Los sistemas de recomendación utilizan bases de datos vectoriales para encontrar elementos similares a los que un us
uario ha mostrado interés previamente. Esto permite ofrecer sugerencias personalizadas de productos, películas o más
ica.

La detección de anomalías se basa en la identificación de
Metadata: {'source': 'doc1.txt'}
------------------------------------------------------------


ID: 3c258089-09c5-4f00-be4e-42db4393e908
Contenido: ósiles.

La exploración espacial es un campo en crecimiento en Argentina, con el desarrollo de satélites y la participación e
n proyectos internacionales. La investigación y el desarrollo en este ámbito son fundamentales para el avance tecnol
ógico del país.

La educación es un pilar fundamental para el desarrollo de Argentina. Se están realizando esfuerzos para mejorar la
calidad y la equidad del sistema educativo en todos los niveles.

La salud pública es una prioridad en Argentina, con un siste
Metadata: {'source': 'doc2.txt'}
------------------------------------------------------------

ID: 19c60dc8-7bae-472c-baf9-9ca5e96e6e24
Contenido: JavaScript maneja interacciones web dinámicas
Metadata: {'source': 'nuevo_documento.txt'}
------------------------------------------------------------


ID: 2ccb0d22-0420-4f5a-b82f-642561c584de
Contenido: ales para acelerar la búsqueda de vecinos más cercanos.

La evaluación de la calidad de los embeddings es crucial para asegurar que capturen de manera efectiva el significad
o semántico de los datos. Se utilizan diversas métricas y técnicas para realizar esta evaluación.

La elección del modelo de embedding adecuado depende de la tarea específica y del tipo de datos que se estén utiliza
ndo. Existen diferentes modelos pre-entrenados y técnicas para entrenar modelos personalizados.

El ajuste fino
Metadata: {'source': 'doc1.txt'}
------------------------------------------------------------
```

```
ID: 1b9916cf-5183-440c-84c8-71324b99d2f1
Contenido:  los restaurantes servían deliciosas parrilladas.

Hacia el norte, el elegante barrio de Recoleta invitaba a la tranquilidad con sus amplias avenidas y parques cuidado
s. El Cementerio de la Recoleta, con sus imponentes mausoleos, era un laberinto de historia y arte funerario. Los ca
fés y boutiques de la zona ofrecían un ambiente sofisticado.

La literatura argentina ha dado grandes nombres como Jorge Luis Borges y Julio Cortázar, cuyas obras han trascendido
fronteras y generaciones. Sus cuentos
Metadata: {'source': 'doc1.txt'}
```

Los embeddings transforman las frases en vectores de alta dimensión que representan su significado semántico. Las palabras clave son cruciales, pero su influencia depende del contexto, y el modelo que se está usando.

En los resultados se observa que el documento "JavaScript maneja interacciones web dinámicas" apareció como resultado, pero los otros dos documentos creados no.

Por otro lado, consideró otros resultados asociados a "Lenguajes" en su significado más amplio.
Si la consulta contiene términos ambiguos o demasiado generales, puede que no coincida bien con las frases en la base de datos y el modelo puede no considerarlas como suficientemente similares debido a las diferencias contextuales.

Ejercicio 5: Eliminaciones Selectivas

Crea dos documentos con tópicos opuestos (ejemplo: uno sobre "Ecología" y otro sobre "Minería de datos").
Elimina el documento sobre ecología usando su ID.
Verifica que solo quede un documento.

¿Cómo usar collection.count() para confirmar?

---

[163]:
```
doc_eco_id = create_example("La ecología es el estudio de las interacciones entre los organismos y su entorno.")
```
```
✅ Documento añadido con ID: a54a491f-95ec-4e84-9883-e278ad2fdeee
```

[164]:
```
doc_min_id = create_example("La minería de datos es el proceso de descubrir patrones en grandes conjuntos de datos.")
```
```
✅ Documento añadido con ID: e6d5aaba-80e5-41e8-9916-056afb2641ad
```

[165]:
```
count_before_delete = collection.count()
print(f'Cantidad de documentos: {count_before_delete}')
```
```
Cantidad de documentos: 40
```

[166]:
```
delete_example(doc_eco_id)
```
```
🗑️ Documento con ID 'a54a491f-95ec-4e84-9883-e278ad2fdeee' eliminado.
```

[167]:
```
count_after_delete = collection.count()
print(f'Cantidad de documentos: {count_after_delete}')
```
```
Cantidad de documentos: 39
```

El método **collection.count()** en Chroma DB sirve para obtener el número total de documentos que actualmente están almacenados dentro de una colección específica.

En esencia, es una manera rápida y sencilla de saber cuántos elementos hay en la colección sin tener que realizar una búsqueda o iterar sobre todos los documentos.

Puede usarse entonces para el control de flujo de procesos o como en este caso para verificar el resultado de operaciones como la eliminación de documentos.

Ejercicio 6: Actualización en Cadena

Crea un documento con el texto "IA y ética son temas actuales".
Actualízalo dos veces:
Primera actualización: "Ética en IA aborda dilemas morales"
Segunda actualización: "El uso de IA debe estar regulado por principios éticos".

¿Cómo sigue evolucionando el contenido del documento con cada update?

---

[168]:
```
doc_id = create_example("IA y ética son temas actuales")
```

```
✅ Documento añadido con ID: 4b564f28-07d8-4180-8e4b-d49e52ff456c
```

Dado que la función **update_example** utiliza el enfoque de eliminar y crear un nuevo documento en lugar de actualizar el existente, utilizo entonces para este caso el método **collection.update()** para mantener así el mismo id de documento.

[169]:
```
text_1 = "Ética en IA aborda dilemas morales"
collection.update(
    ids=[doc_id],
    documents=[text_1],
    embeddings=[get_embeddings(text_1)]
)
```

[170]:
```
text_2 = "El uso de IA debe estar regulado por principios éticos"
collection.update(
    ids=[doc_id],
    documents=[text_2],
    embeddings=[get_embeddings(text_2)]
)
```

Con cada llamada al método **update()**, el contenido del documento asociado al ID "doc_id" se reemplaza completamente con el nuevo texto proporcionado.

Cada actualización representa una nueva versión del documento, donde el contenido evoluciona para reflejar una idea o perspectiva ligeramente diferente dentro del tema general de "IA y Ética".

Al llamar al método update() y proporcionar un nuevo contenido estoy volviendo a calcular y actualizando también su **embedding**. El embedding anterior asociado a ese id es reemplazado por el nuevo embedding, reflejando la semántica del texto actualizado.

Los embeddings son representaciones numéricas (vectores) del significado del texto.
Al cambiar el texto, el significado cambia y, por lo tanto, su representación vectorial también debería cambiar.
Esto significa que la posición del documento en el espacio vectorial se mueve, reflejando su nueva semántica en relación con los otros documentos de la colección.

Dado que el embedding del documento actualizado ha cambiado, su relevancia para futuras búsquedas podría variar también.

Ejercicio 7: Consultas Ambiguas

Crea dos documentos similares pero no idénticos:
"Blockchain garantiza transacciones seguras"
"Criptomonedas usan tecnología blockchain".
Realiza una consulta por "Seguridad en finanzas digitales".

¿Ambos documentos aparecen? Explica la similitud semántica.

---

[171]:
```
doc_id_1 = create_example("Blockchain garantiza transacciones seguras")
```

```
✅ Documento añadido con ID: f6133ecd-e262-4a08-bf88-b305546acd54
```

[172]:
```
doc_id_2 = create_example("Criptomonedas usan tecnología blockchain")
```

```
✅ Documento añadido con ID: f64d6cfa-e109-4c4d-b884-ddf26e4317c6
```

[173]:
```
x1 = read_example("Seguridad en finanzas digitales", n=5)
```

```
------------------------------------------------------------

ID: f6133ecd-e262-4a08-bf88-b305546acd54
Contenido: Blockchain garantiza transacciones seguras
Metadata: {'source': 'nuevo_documento.txt'}
------------------------------------------------------------

ID: 7da5439f-d7e2-455d-87ee-c62ef208b7eb
Contenido: en crecimiento en Argentina, con investigaciones y desarrollos en áreas como la agricultura, la salud y l
a industria.

La industria del software y los servicios informáticos tienen un gran potencial en Argentina, con un talento humano
calificado y un ecosistema emprendedor en expansión.

El diseño de indumentaria y la moda argentina están ganando reconocimiento a nivel internacional, con diseñadores qu
e fusionan tradición y vanguardia.

La producción audiovisual argentina, incluyendo cine y tele
Metadata: {'source': 'doc3.txt'}
------------------------------------------------------------

ID: 3c258089-09c5-4f00-be4e-42db4393e908
Contenido: ósiles.

La exploración espacial es un campo en crecimiento en Argentina, con el desarrollo de satélites y la participación e
n proyectos internacionales. La investigación y el desarrollo en este ámbito son fundamentales para el avance tecnol
ógico del país.

La educación es un pilar fundamental para el desarrollo de Argentina. Se están realizando esfuerzos para mejorar la
calidad y la equidad del sistema educativo en todos los niveles.

La salud pública es una prioridad en Argentina, con un siste
Metadata: {'source': 'doc2.txt'}
------------------------------------------------------------

ID: 2c33ab02-ef64-4f65-b1ce-356ce20a6d04
Contenido: ma que busca garantizar el acceso a la atención médica a todos los ciudadanos. Se están implementando pol
íticas para fortalecer la prevención y el tratamiento de enfermedades.

La cultura del vino es muy importante en Argentina, especialmente en la región de Mendoza, donde se producen vinos d
e alta calidad reconocidos a nivel mundial, siendo el Malbec la cepa insignia.

El turismo es una actividad económica clave en Argentina, atrayendo a visitantes interesados en la diversidad de sus
paisajes,
Metadata: {'source': 'doc2.txt'}
------------------------------------------------------------

ID: 8471308a-4d4a-4f90-b0c3-5ff73f1edd64
Contenido: su cultura y su historia. Desde las Cataratas del Iguazú hasta las playas de Mar del Plata, el país ofrec
e una amplia gama de destinos.

La innovación tecnológica es un motor de crecimiento en Argentina, con emprendedores y empresas desarrollando soluci
ones en áreas como el software, la biotecnología y la inteligencia artificial.

La colaboración internacional es fundamental para abordar los desafíos globales. Argentina participa activamente en
foros y organizaciones internacionales para promove
Metadata: {'source': 'doc2.txt'}
```

Sólo aparece el primer documento y no el segundo.

En términos de similitud semántica según el modelo de embedding, el vector de "Seguridad en finanzas digitales" está más cerca en el espacio vectorial al vector de "Blockchain garantiza transacciones seguras" que al vector de "Criptomonedas usan tecnología blockchain".

La frase menciona explícitamente la palabra "seguras", que es un componente clave del concepto de "seguridad" en la consulta. Aunque la consulta se enfoca en un contexto más amplio ("finanzas digitales"), la mención directa de la seguridad en el primer documento podría ser un factor de mayor peso para el modelo de embedding.
El verbo "garantiza" implica una fuerte promesa o seguridad inherente proporcionada por la tecnología blockchain en el contexto de las transacciones. Esto podría resonar más con la idea de "seguridad" en la consulta.

La segunda frase, en cambio, se centra en la relación de uso ("usan") entre las criptomonedas y la tecnología blockchain. Si bien la seguridad es una característica importante de blockchain que beneficia a las criptomonedas, la frase en sí no menciona la seguridad directamente.

Ejercicio 8: Reiniciar Base de Datos

Elimina todos los documentos usando delete() con el parámetro correcto.
Vuelve a cargar la base desde cero con crear_base_datos().

¿Cuántos documentos hay ahora? (Usa collection.count()).

---

[174]:
```
count_before_delete = collection.count()
print(f'Cantidad de documentos: {count_before_delete}')
```

```
Cantidad de documentos: 42
```

[175]:
```
# Borra todos los documentos de la colección
collection.delete(where={})

print(f"Se han borrado todos los documentos de la colección.")
```

```
Se han borrado todos los documentos de la colección.
```

[176]:
```
count_after_delete = collection.count()
print(f'Cantidad de documentos: {count_after_delete}')
```

```
Cantidad de documentos: 0
```

[177]:
```
# Recrear la DB
crear_base_datos()
```

```
✅ Base de datos creada con éxito.
```

[178]:
```
count_after_create = collection.count()
print(f'Cantidad de documentos: {count_after_create}')
```

```
Cantidad de documentos: 33
```

Ejercicio 9: Errores de Inserción

Intenta crear un documento sin texto (ejemplo: cadena vacía).
Corrige el error usando la función get_embeddings().

¿Qué lección aprendemos sobre los requisitos de ChromaDB?

---

[179]:
```
doc_id = create_example("")
```

```
✅ Documento añadido con ID: 411cf6ac-e0f6-4349-a8d2-a637652f2c97
```

[180]:
```
print (len(get_embeddings("")))
```

```
384
```

La creación de un documento vacío no da error, porque de todas formas se genera una representación vectorial.

Sin embargo, Chroma DB está diseñado principalmente para trabajar con texto que tenga significado semántico para poder generar embeddings útiles para la búsqueda y comparación. Intentar almacenar documentos sin contenido textual o con cadenas vacías puede llevar a un comportamiento inesperado, ya que no hay información que el modelo de embedding pueda procesar de manera efectiva.

Lo que sí arroja error, como se vio en el punto 3, es cuando el embedding es vacío.

ChromaDB no admite la inserción de un embedding no valido (**"ValueError: Expected each embedding in the embeddings to be a list, got []"**).

Para que Chroma DB pueda realizar búsquedas semánticas eficientemente, los documentos deben tener embeddings que representen su significado. Si no se puede generar un embedding válido, el documento no podría ser indexado y consultado de manera apropiada.

Ejercicio opcional: Retrieval-Augmented Generation

Intente levantar un llm en un endpoint local con ollama o LMStudio.
Utilice el llm para generar respuestas aumentadas con contexto similar a una consulta dada.

---

[181]:
```
import requests
import json
```

Utilizo **Ollama** con el modelo **llama2**:

[182]:
```
# Ollama instalación local
# curl -fsSL https://ollama.com/install.sh | sh

url = "http://localhost:11434/v1/chat/completions"  # ollama serve

model_name = "llama2" # ollama pull llama2
```

Función para consultar al modelo incluyendo contexto o no, dependiendo de los parámetros ingresados:

[183]:
```
def query_ollama(endpoint, user_input, model_id, db=0):

    """ Consulta ollama, si db > 0,
        añade el contexto de los documentos
        encontrados en la db"""

    # Headers para la solicitud
    headers = {
        "Content-Type": "application/json"
    }

    print("\n[Pregunta del usuario]:")
    print(user_input)
```

```
    if db > 0:

        # Añadir contexto
        query_emb = get_embeddings(user_input)

        results = collection.query(
            query_embeddings=[query_emb],
            n_results=db
        )

        if results["documents"]:
            context = "\n".join(results["documents"][0])

            # Juntar el contexto obtenido y la pregunta
            user_input = f"Basándote en la siguiente información:\n{context}.\nResponde a la siguiente pregunta: {use

            print("\n[Prompt enviado a Ollama con contexto de ChromaDB]:")
            print(user_input)
```

```
# Definir el cuerpo de la solicitud
payload = {
    "model": model_id,
    "messages": [
        {"role": "system", "content": "Responder en idioma español."},
        {"role": "user", "content": user_input}
    ]
}

# Enviar la solicitud POST a la API de Ollama
response = requests.post(endpoint, headers=headers, data=json.dumps(payload))

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Extraer el contenido de la respuesta
    response_data = response.json()
    # Imprimir la respuesta del modelo
    print("\n[Respuesta del modelo]:", response_data['choices'][0]['message']['content'])
else:
    # Mostrar un mensaje de error si la solicitud falla
    print(f"Error en la solicitud: {response.status_code}, {response.text}")
```

Se realiza la misma consulta sin contexto y con el contexto añadido a partir de la consulta a la DB.

[184]:
```
doc_id = create_example("El color del cielo en Saturno es violeta")
```

```
✅ Documento añadido con ID: 9172fd4b-fc95-43f2-8ad3-a56d1c4da4e9
```

[185]:
```
# Prompt del usuario para interactuar con el modelo
prompt = "¿De qué color es el cielo en Saturno?"
```

[186]:
```
# Consulta a Ollama sin contexto
query_ollama(url, prompt, model_name, 0)
```

```
[Pregunta del usuario]:
¿De qué color es el cielo en Saturno?

[Respuesta del modelo]:
En Saturno, el cielo es de un color naranja-rojobrillante debido a la presencia de partículas de polvo en la atmósfe
ra del planeta. La luz solar que incide sobre Saturno es dispersada por estas partículas, dando lugar a un efecto co
nocido como "efecto Tyndall". El color naranja-rojo es un resultado de la interacción entre la luz y las partículas
de polvo en la atmósfera de Saturno, y no es el mismo que el cielo azul que se puede observar en Tierra debido a la
ausencia de polvo en la atmósfera.
```

[187]:
```
# Consulta con contexto (1 documento)
query_ollama(url, prompt, model_name, 1)
```

```
[Pregunta del usuario]:
¿De qué color es el cielo en Saturno?

[Prompt enviado a Ollama con contexto de ChromaDB]:
Basándote en la siguiente información:
El color del cielo en Saturno es violeta.
Responde a la siguiente pregunta: ¿De qué color es el cielo en Saturno?

[Respuesta del modelo]:
 ¡Claro! Según la información proporcionada, el color del cielo en Saturno es violeta. Entonces, la respuesta a la p
regunta es:

El colors del cielo en Saturno es violeta.
```

```
[188]: # Consulta con contexto (3 documentos)
query_ollama(url, prompt, model_name, 3)
```

```
[Pregunta del usuario]:
¿De qué color es el cielo en Saturno?

[Prompt enviado a Ollama con contexto de ChromaDB]:
Basándote en la siguiente información:
El color del cielo en Saturno es violeta
ciosa continuaba, con el tráfico constante y el murmullo de las conversaciones. Los turistas se maravillaban ante la
grandiosidad del Teatro Colón, imaginando las óperas y ballets que habían resonado en su interior.

Más al sur, en el barrio de La Boca, los colores vibrantes de las casas de Caminito contaban historias de inmigrante
s y artistas. El tango resonaba en las calles, invitando a los visitantes a sentir la pasión de esta danza emblemáti
ca. Los artesanos ofrecían sus creaciones, mientras
territorio y sus influencias culturales.

La ciencia y la tecnología también tienen un lugar importante en Argentina. Investigadores y científicos trabajan en
diversas áreas, desde la biotecnología hasta la astronomía, contribuyendo al avance del conocimiento.

La música folklórica argentina, con sus ritmos y melodías características, cuenta historias de la tierra y sus habit
antes. Instrumentos como la guitarra, el charango y el bombo legüero son protagonistas de estas expresiones cultural
es.

E.
Responde a la siguiente pregunta: ¿De qué color es el cielo en Saturno?

[Respuesta del modelo]: El color del cielo en Saturno es violeta, según la información proporcionada en la narració
n.
```

✅ En conclusión, se observa que el modelo adapta su respuesta en función del contexto obtenido de los documentos en la base de datos.

Referencias

- https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/BDIA/tree/main/clase_6/VectorDatabases
- https://ollama.com/

Anexos

010011 </> IPYNB	• Notebook con el código fuente.
GitHub	• Repositorio GitHub con todos los inputs y el contenido del TP.