

INTRODUCTION À L'IA

ACTIVITÉ 1 – DÉCOUVERTE D'UNE BDD VECTORIELLE

PRÉREQUIS

Ces labos ont été réalisés avec l'environnement technique suivant :

- Machine virtuelle ou physique Linux Debian 13 (paquets git, zstd et curl) / Python 3.11.2 (paquets python3-pip et python3-venv).
- CPU : dans l'activité 2, les traitements IA sont assez longs (2x20mn en VM avec 2 CPU x 2 cœurs) donc plus c'est mieux ! – Non testé mais les traitements peuvent être optimisés avec un GPU (paramétrable dans les scripts Python).
- RAM : 8 Gio / Stockage minimum : 60 Gio minimum (les modèles sont assez volumineux et téléchargés en local).

PRÉPARATION DE L'ENVIRONNEMENT PYTHON

🔗 Initialiser l'environnement Python3 :

```
mkdir tp-ia-rag
cd tp-ia-rag
python3 -m venv env && source ./env/bin/activate (deactivate pour quitter)
pip install chromadb sentence-transformers
```

🔗 Copier le code source du premier programme Python de test (généré via ChatGPT ;-) (fichier activite1.py fourni dans l'archive) dans le dossier tp-ia-rag :

```
import chromadb
from chromadb.config import Settings
from sentence_transformers import SentenceTransformer

# 1) Modèle d'embedding
model = SentenceTransformer("all-MiniLM-L6-v2")

def embed(texts):
    return model.encode(texts).tolist()

# 2) Client Chroma en mode local (en mémoire pour le test)
client = chromadb.PersistentClient(path="db/")

# 3) Création / récupération d'une collection
collection = client.get_or_create_collection(
    name="docs_demo",
    metadata={"hnsw:space": "cosine"} # métrique de similarité
)

# 4) Jeu d'essai : quelques phrases
documents = [
    "Les chats aiment dormir au soleil.",
    "Les chiens sont des animaux très fidèles.",
    "Les voitures électriques deviennent de plus en plus populaires.",
    "Les félins sauvages chassent principalement la nuit.",
    "Les transports en commun réduisent la pollution en ville."
]

ids = [f"doc_{i}" for i in range(len(documents))]
metadatas = [
    {"type": "animal"},
    {"type": "animal"},
    {"type": "transport"},
    {"type": "animal"},
    {"type": "transport"},
]

# 5) Vectorisation + insertion dans Chroma
embeddings = embed(documents)

collection.add(
    ids=ids,
```

```

    documents=documents,
    metadatas=metadatas,
    embeddings=embeddings
)

print("Documents insérés dans la collection.")

# 6) Requête sémantique
query = "animaux qui chassent la nuit"

query_emb = embed([query])

results = collection.query(
    query_embeddings=query_emb,
    n_results=3,
    include=["documents", "metadatas", "distances"]
)

print("\nRequête :", query)
for doc, meta, dist in zip(
    results["documents"][0],
    results["metadatas"][0],
    results["distances"][0]
):
    print(f"- doc: {doc}")
    print(f" meta: {meta}, distance: {dist:.4f}")

print("Dimension des vecteurs :", len(embeddings[1]))

```

ANALYSE DU CODE SOURCE

Dans cette partie, vous pouvez vous aider d'une IA pour répondre aux questions. Cela ne signifie pas que vous devez vous contenter de simples copier-coller !

Travail à faire 1 Analyser le programme fourni. En déduire son objectif.

Travail à faire 2 Ce programme utilise une bibliothèque SentenceTransformers. Indiquer son rôle.

Travail à faire 3 Ce programme utilise en ligne 6 : all-MiniLM-L6-v2 en paramètre de SentenceTransformer . Préciser ce que c'est et son rôle.

EXÉCUTION DU PROGRAMME

- 🔗 Exécuter le programme pour observer son fonctionnement. Vous pouvez modifier la requête (ligne 51) pour voir les différences.



L'utilisation de la variable d'environnement `HF_HUB_DISABLE_XET=1` permet d'accélérer le téléchargement du modèle lors de l'exécution du script.

```
export HF_HUB_DISABLE_XET=1
python 3 activite1.py
```

Travail à faire 4 Dire si ce programme utilise une IA de type « modèle de langage » ?

Travail à faire 5 Relever la dimension des vecteurs.

Travail à faire 6 Modifier le programme afin d'afficher les *embeddings* correspondant aux 5 documents. Résultat attendu (extrait) :

```
{'ids': ['doc_0', 'doc_1', 'doc_2', 'doc_3', 'doc_4'], 'embeddings':
array([[ -0.05063749,  -0.00309375,   0.02703848, ...,   0.11366574,
        -0.04635011,  -0.01640419],
       [ -0.00529536,   0.01956804,   0.02137414, ...,   0.03058645,
         0.03381941, -0.03557273],
       [  0.01148486,   0.01540026,   0.00806793, ...,   0.00211769,
         0.05351813, -0.03133301],
       [ -0.01214059,   0.03900571,  -0.06783811, ...,   0.05137474,
         0.0211298 , -0.0579013 ],
       [  0.06523567,  -0.02686711,   0.04271925, ...,   0.13519789,
        -0.00123394, -0.05878868]], shape=(5, 384)), 'documents': None, 'uris':
None, 'included': ['embeddings'], 'data': None, 'metadatas': None}
```

Travail à faire 7 Modifier le programme pour afficher l'embedding correspondant au prompt.