Conception d'un Système de Recommandation de Films par Approche RAG

1. Description du problème et objectifs

- **Contexte :** L'abondance de contenus (Netflix, Amazon) crée une "paralysie du choix". Les moteurs de recherche classiques par mots-clés (ex: "film science fiction") manquent de nuance.
- **Objectif :** Créer un assistant conversationnel capable de comprendre une requête complexe en langage naturel (ex: "Un film avec des aliens mais qui fait peur") et de proposer une recommandation justifiée.
- **Solution :** Utilisation d'une architecture RAG (Retrieval-Augmented Generation) combinant une base de données vectorielle (ChromaDB) et un LLM (Gemma).

2. Architecture Logicielle et Pipeline (Diagrammes)

1. **Ingestion & Prétraitement (ETL) :**
    - Fusion des datasets `movies` et `credits` via `id`.
    - Parsing des colonnes JSON (`genres`, `cast`, `director`) via `ast.literal_eval`.
    - **Création de la "Soup" :** Concaténation des métadonnées clés en une seule chaîne de texte pour enrichir le contexte sémantique.
2. **Vectorisation (Embedding) :**
    - Conversion de la "Soup" en vecteurs denses .
3. **Stockage (Vector Store) :**
    - Sauvegarde locale persistante via ChromaDB.
4. **Retrieval & Generation :**
    - Recherche par similarité cosinus (Top-k).
    - Construction du prompt avec le contexte.
    - Génération de la réponse via l'API **Ollama**.

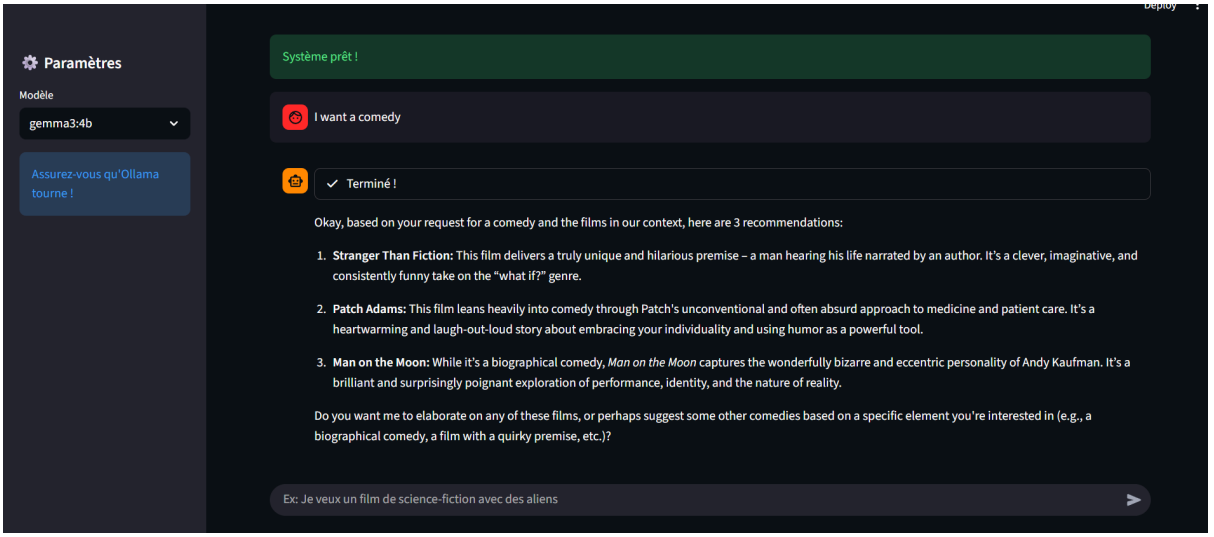
3. Justification des choix techniques

- **Modèle d'Embedding (`all-MiniLM-L6-v2`) :**
    - *Choix :* Modèle léger conçu pour le "Semantic Search".
    - *Justification :* Il offre le meilleur compromis rapidité/précision pour une exécution sur CPU (`device='cpu'`), indispensable pour faire tourner le projet sur un ordinateur portable standard sans GPU dédié.
- **Base Vectorielle (`ChromaDB`) :**
    - *Choix :* Base de données vectorielle open-source.

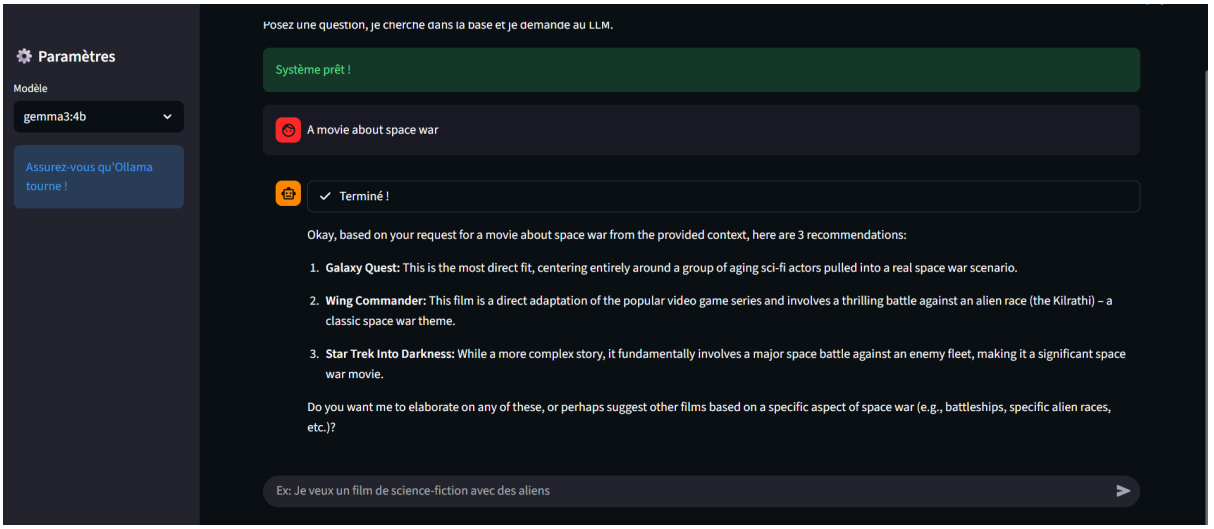
    - *Justification :* Contrairement à Pinecone (cloud), Chroma fonctionne en local (`PersistentClient`), ce qui garantit la confidentialité des données et ne nécessite pas de clé API internet.
- **LLM (`Gemma 3:270m` ou `4b`) :**
    - *Justification :* Utilisation de modèles "SLM" (Small Language Models) via Ollama. Cela permet une inférence locale rapide tout en étant suffisant pour de la tâche de synthèse et de recommandation simple, réduisant l'empreinte carbone et les coûts.
- **Interface (`Streamlit`) :**
    - *Justification :* Permet un prototypage rapide en Python pur, avec une gestion native du cache (`@st.cache_data`) pour optimiser le chargement des gros fichiers CSV (5000 lignes).

4. Résultats Expérimentaux

- **Cas 1 : Recherche par genre.**



- **Cas 2 : Recherche sémantique.**



5. Limites et Améliorations

- **Limite 1 :** Le Dataset s'arrête en 2017 (pas de films récents).
- **Limite 2 :** Risque d'hallucination du petit modèle (270m) s'il ignore le contexte.
- **Amélioration possible :** Ajouter un système de filtrage hybride (ex: filtrer par année avant de faire la recherche vectorielle).