MOVIE SEARCH APPLICATION

Big Data Capstone Project – August 2024

ARUN KUMAR SUBRAMANIAM

CARLOS MUNOZ EBRATT

KAMALDEEP KAUR

LUIS ALEJANDRO GUTIERREZ HAYEK

FELIPE FERNANDEZ

INTRODUCTION

This project is an example of building a semantic search using Python, NLP models, and Atlas Vector Search as a vector database for finding movies using natural language queries and Streamlit as the user interface to interact with the Database.

VECTOR EMBEDDINGS

A vector is a list of floating point numbers (representing a point in an n-dimensional embedding space) and captures semantic information about the text it represents. For instance, an embedding for a string using an open source LLM model called all-MiniLM-L6-v2 would consist of 384 floating point numbers and look like this.

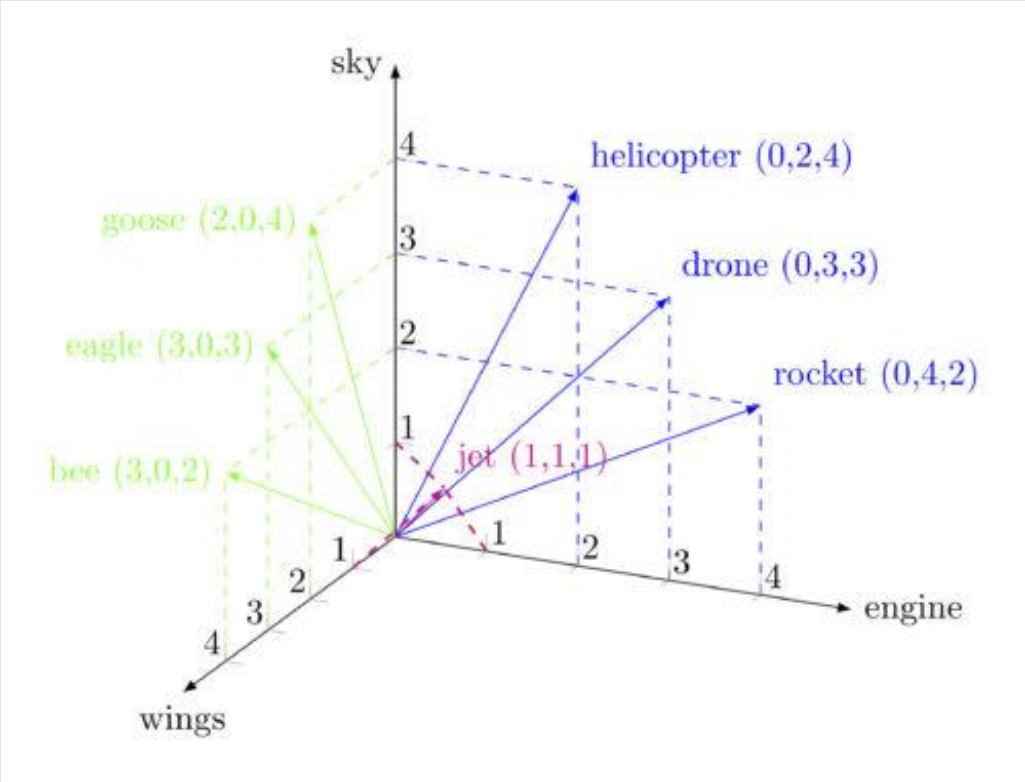
```
▾ plot_embedding_hf : Array (384)
    0: -0.03985478729009628
    1: -0.016877852380275726
    2: -0.051168572157621384
    3: 0.08319630473852158
    4: 0.04009385406970978
```

ALL-MINILM-L6-V2

- The **all-MiniLM-L6-v2** is a pre-trained language model developed by Microsoft. It's a smaller and more efficient version of the MiniLM model, specifically designed for tasks like semantic search, sentence similarity, and embedding generation. The model has 6 layers and uses a smaller architecture to provide fast and accurate performance while maintaining a lower computational cost. It is widely used in applications that require embedding text into dense vector spaces for tasks like document retrieval and clustering.
- This is a sentence-transformers model: It maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search
- https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

COSINE SIMILARITY

- Vector coordinates can be used to calculate the proximities between words. This is done with cosine similarity (cos θ), the cosine of the angle between two word vectors.



VECTOR SEARCH

- Vector search is a capability that allows you to find related objects that have a semantic similarity. This means searching for data based on meaning rather than the keywords present in the dataset.
- Then, it finds related content by comparing the distances between these vector embeddings, using approximate k nearest neighbor (approximate KNN) algorithms.
- The most commonly used method for finding the distance between these vectors involves calculating the cosine similarity between two vectors.

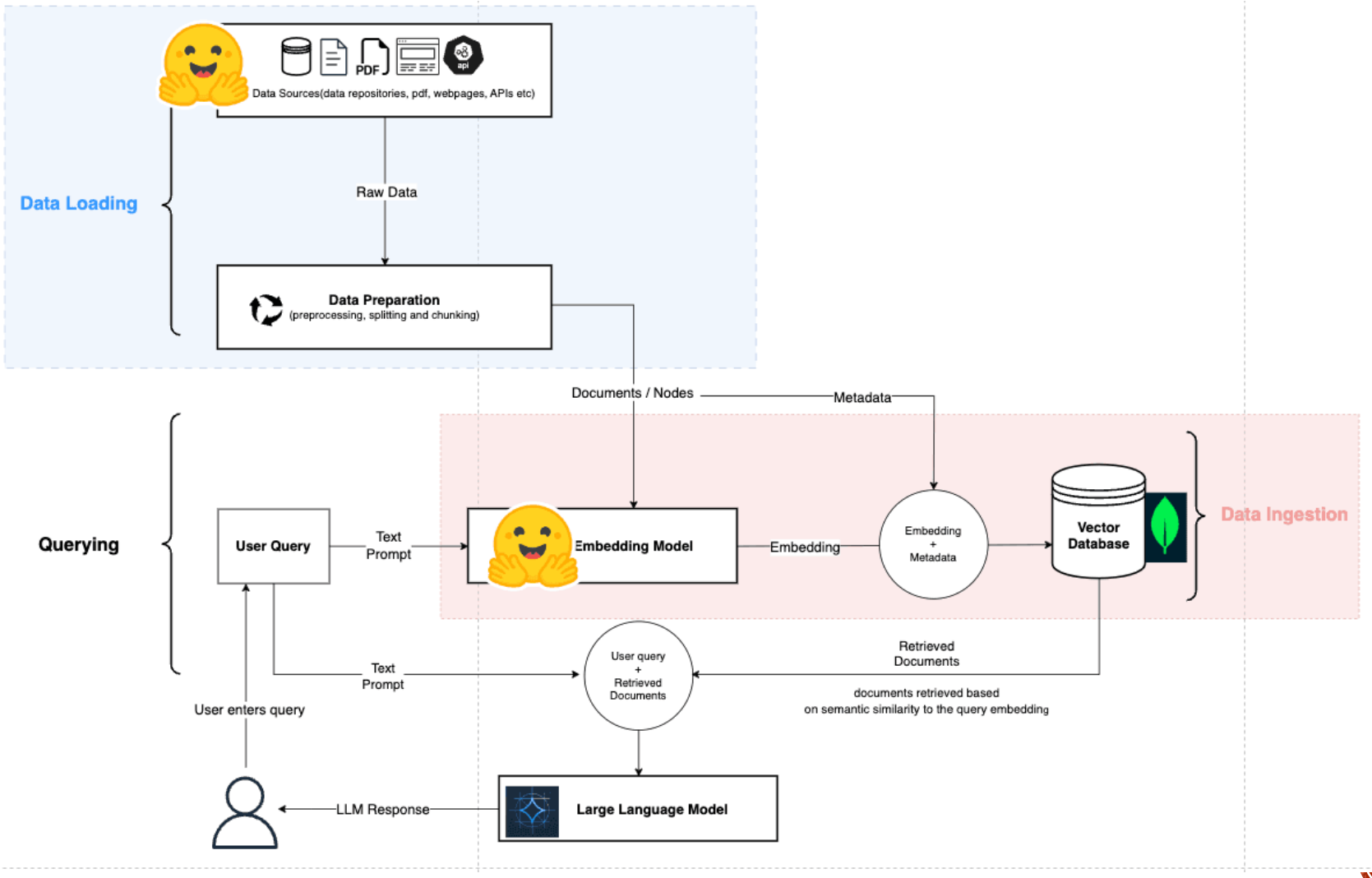
ATLAS VECTOR DATABASE

- Fully managed service that simplifies the process of effectively indexing high-dimensional vector data within MongoDB and being able to perform fast vector similarity searches.
- MongoDB as a standalone vector database for a new project or augment the existing MongoDB collections with vector search functionality.

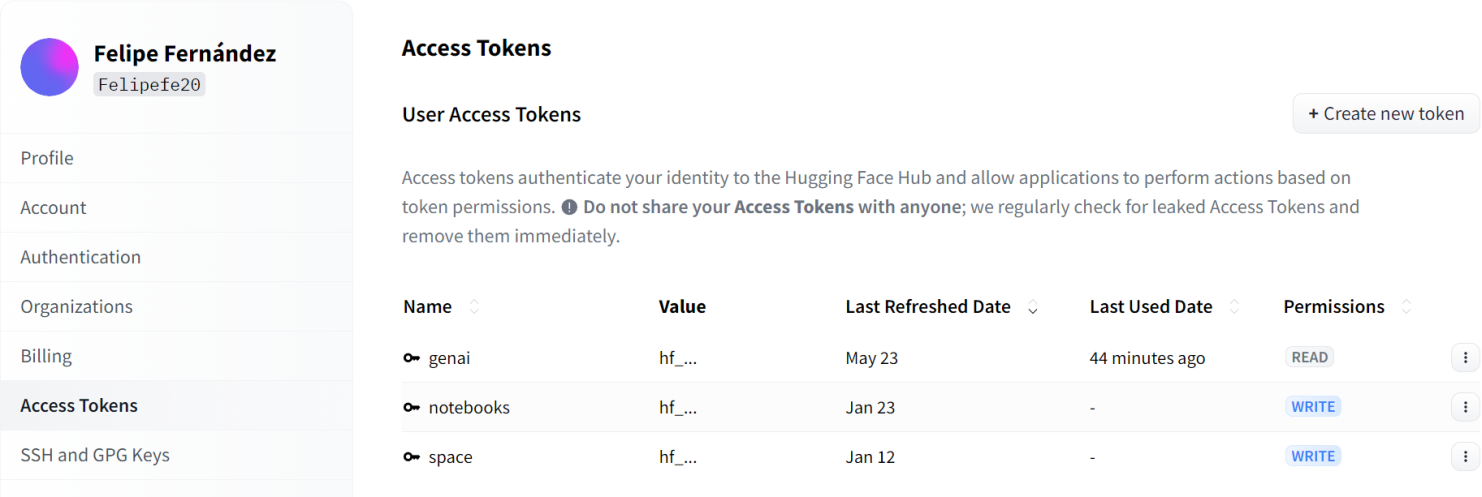
SEMANTIC SEARCH FOR MOVIE RECOMMENDATIONS

- We will be using the all-MiniLM-L6-v2 model from HuggingFace for generating the vector embedding during the index time as well as query time.
- The user query can be converted into vector embedding, and then the vector search can find the most relevant results by finding the nearest neighbors in the embedding space.

SYSTEM ARCHITECTURE



DATA LOADING: STEP 1- CREATE HUGGINGFACE ACCESS TOKENS



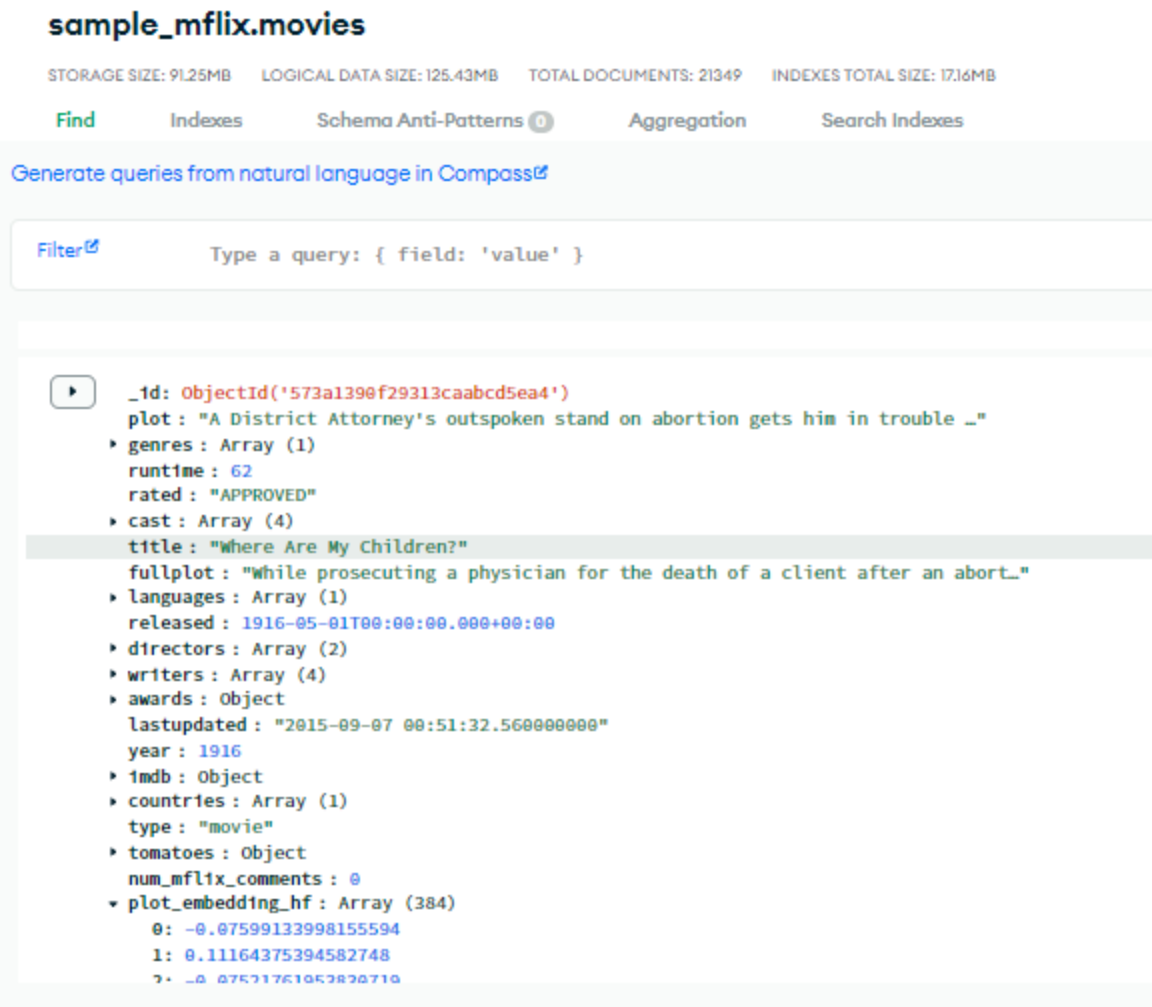
STEP 2: CONNECT TO MONGODB AND CREATE EMBEDDINGS

vector-search-projects - Generate_embeddings.py

```python

db = client.sample_mflix
collection = db.movies
#add counter to the loop
counter = 0
for doc in collection.find({'fullplot':{"$exists": True}}):
   #check if the plot embedding already exists
    if 'plot_embedding_hf' not in doc:
      try:
        doc['plot_embedding_hf'] = generate_embedding(doc['fullplot'])
        collection.replace_one({'_id': doc['_id']}, doc)
        counter=counter+1
        print(f"Updated{counter}",)
      except:
        pass

```

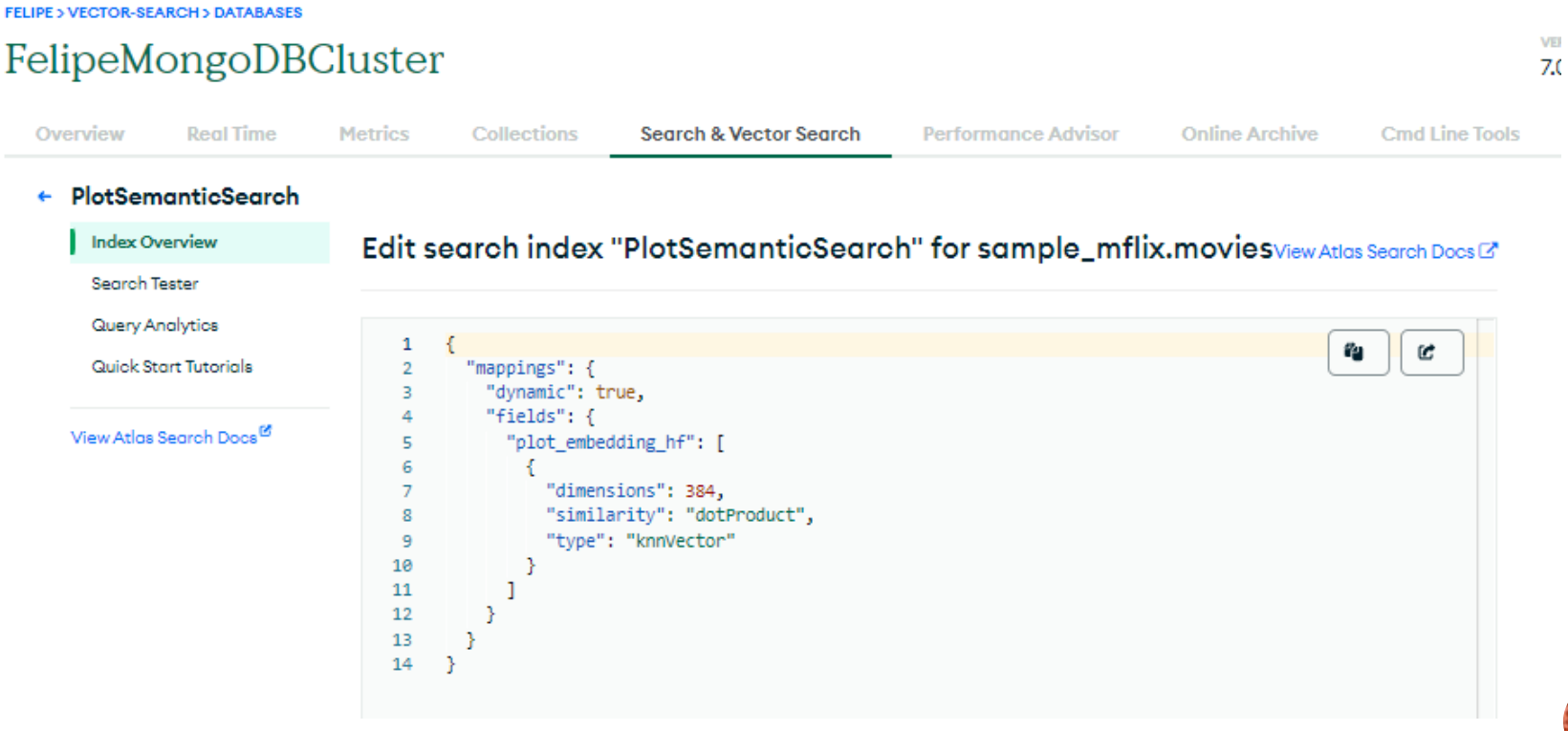
Restrictions in how many vectors per hour, we had to subscribe to HuggingFace Pro

- Once this step completes, we verify in the database that a new field "plot_embedding_hf" has been created for some collections.



STEP 3: CREATE A VECTOR SEARCH INDEX

To perform vector search on the data in Atlas, we create an Atlas Vector Search index, which are used to efficiently retrieve documents that contain vector embeddings.



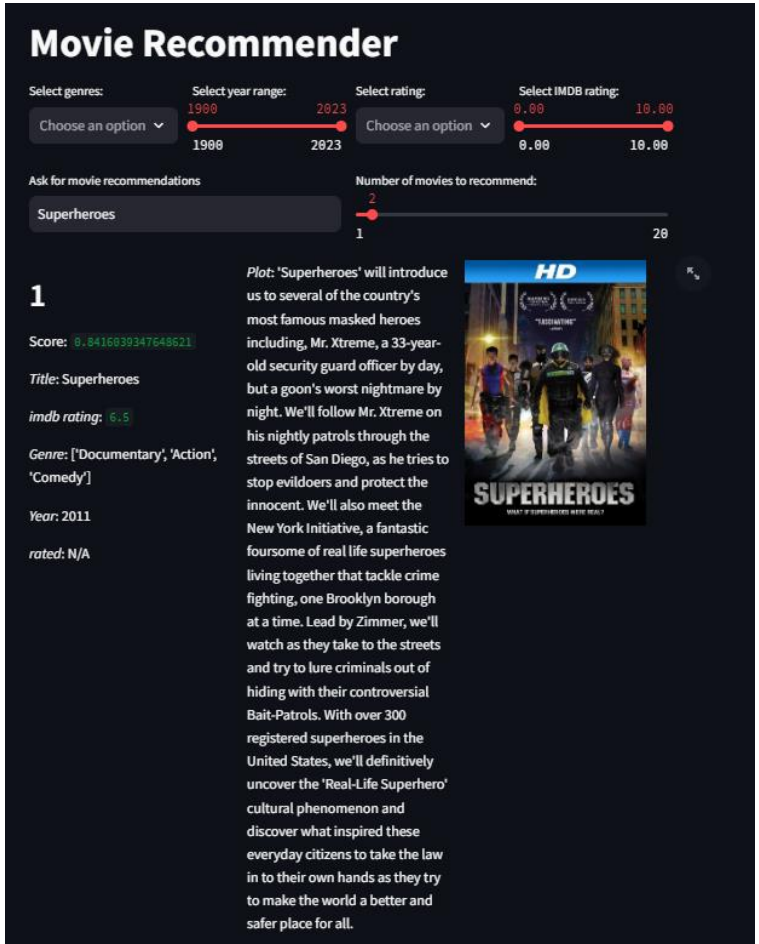
QUERYING THE DATA

vector-search-projects - user_interface.py

```python
pipeline = []

pipeline.append({
    "$vectorSearch": {
        "queryVector": query_vector,
        "path": "plot_embedding_hf",
        "numCandidates": 5000,
        "limit": num_movies,
        "index": "PlotSemanticSearch",
    }
})

# Add a match stage to limit vector search to the initially filtered results
pipeline.append({"$match": {"_id": {"$in": matching_ids}}})

pipeline.append({"$project": {"score": {"$meta": "vectorSearchScore"}}})

results = collection.aggregate(pipeline)
```

USER INTERFACE AND DEMO

- https://capstone-project-movies-vector-search.streamlit.app/



CONCLUSIONS:

- Effective Use of Transformer Models
- Cosine Similarity as a Reliable Metric
- Scalable and User-Friendly System
- Challenges with Data Quality and API Limits

RECOMMENDATIONS:

- Use a LLM to interact with the data and create a chatbot or RAG (retrieval-augmented generation) system with memory
- App depends on the data quality and size, instead of the model used.
- Fine-Tuning Models: Further fine-tuning of the transformer models on movie-specific data could improve the quality of the embeddings, leading to even more accurate similarity scores and recommendations.

QUESTIONS & ANSWERS