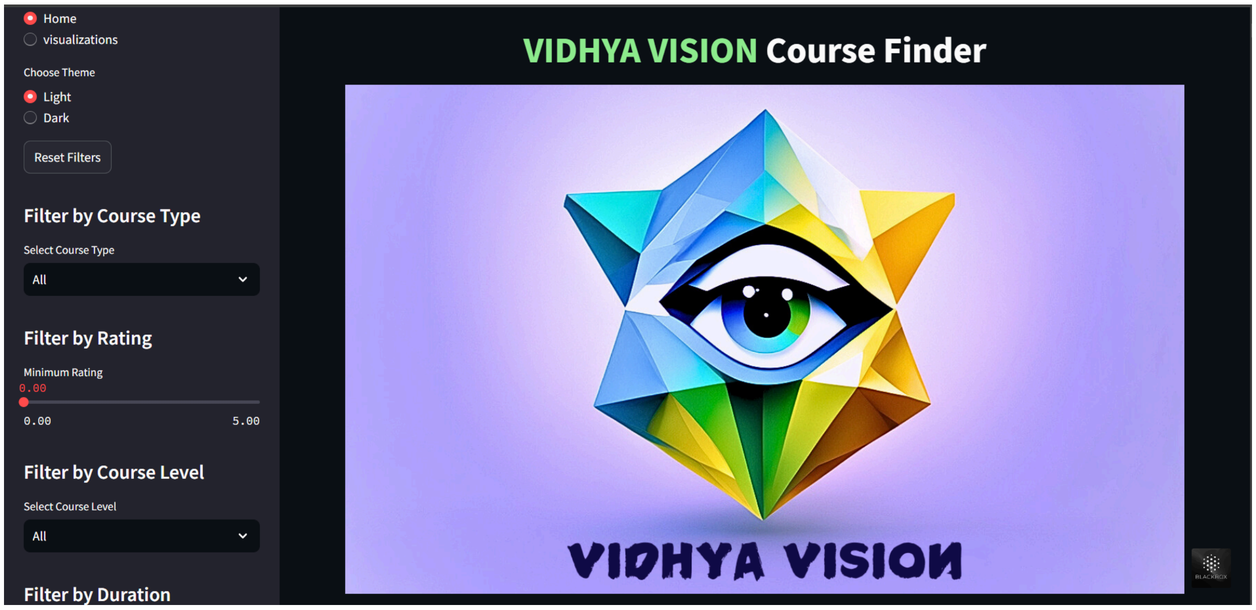
Name: Yogesh Jadhav
B.Tech E&TC 2024

Access the Vidyavision App :
https://huggingface.co/spaces/yogeshjadhav666/Vidhya_Vision



Project Overview

Purpose:
The goal of this project is to build a smart search tool to help users quickly find relevant free courses on the Analytics Vidhya platform. By leveraging embeddings, the tool allows users to input natural language queries or keywords to find courses that match their interests.

Overview:
The tool uses embeddings to search through a collection of course data, including titles, descriptions, and other details, to retrieve the most relevant results based on user input.

App Features

1. **Navigation & Page Structure**:
    - **Home Page**: Displays the course search functionality.
    - **Visualizations Page**: Provides course data insights.
2. **User Interface**:
    - Sidebar for navigation between "Home" and "Visualizations".
    - Light and Dark theme options.
    - Reset button to clear filters.
3. **Filtering & Search**:
    - Filters: Course Type, Rating, Level, Duration.
    - Search bar for keyword-based queries.
    - Results are dynamically displayed with course details, including title, description, rating, price, and duration.
4. **Data Insights**:
    - Displays total courses, average rating, and free vs paid courses statistics.
5. **Visualizations**:
    - Course Rating Distribution: Histogram and bar chart.
    - Featured Courses: Top-rated courses.
    - Course Comparison: Allows comparison based on title.

Problem Understanding

The challenge was to create a tool that efficiently searches and suggests relevant courses based on natural language input. The tool should handle varied course data and provide quick, accurate results.

Methodology

1. **Data Collection & Cleaning**:
    - Scraped course data from Analytics Vidhya, focusing on free courses.
    - Cleaned data for consistency (e.g., handling missing descriptions).
2. **Embedding Model**:
    - Used **Sentence-Transformers** to generate embeddings from course descriptions.
    - Stored embeddings in **FAISS** for efficient similarity searches.
3. **Search Tool Development**:
    - Built a Streamlit app to allow users to query courses.
    - Deployed the tool on Hugging Face Spaces.

Libraries & Tools

- **requests**: Web scraping.
- **beautifulsoup4**: Parsing HTML.
- **pandas**: Data processing.
- **streamlit**: User interface.
- **langchain**: Embedding generation.
- **faiss-cpu**: Efficient search.
- **sentence-transformers**: Embedding generation.
- **matplotlib**: Data visualizations.

Dataset Description

The dataset consists of scraped course data, including attributes like title, description, rating, instructor, and more. Key columns include:

- **course_title**: The title of the course.
- **course_url**: URL link to the course page.
- **course_description**: A brief description of what the course offers.
- **course_curriculum**: The curriculum or syllabus of the course (if available).
- **course_level**: The level of difficulty (e.g., Beginner, Intermediate, Advanced).
- **course_rating**: User ratings for the course.
- **lesson_count**: The total number of lessons in the course.
- **price**: The cost of the course (likely with a focus on free courses).
- **reviews**: Number of reviews the course has received.
- **course_duration**: The total duration of the course.
- **instructor_name**: Name of the course instructor.
- **who_should_enroll**: Target audience or intended learners for the course.

Data Cleaning & Preprocessing

1. Removed missing or irrelevant data (e.g., missing titles or descriptions).
2. Standardized numerical values for ratings, durations, and lesson counts.
3. Preprocessed course descriptions for better searchability (removing stopwords and punctuation).

Embedding Model Selection

Why Embeddings?
Embeddings capture the semantic meaning of text, enabling efficient comparison of user queries with course descriptions.

Embedding Process:
Used **Sentence-Transformers** to generate embeddings for course titles and descriptions, which were stored in a **FAISS** index.

Search System Development

1. **Search Functionality**:
    - User input is processed into an embedding and compared with course embeddings in the FAISS index.
    - Results are ranked by cosine similarity.
2. **Use of FAISS**:
    - FAISS is used for fast similarity search, indexing course embeddings for quick retrieval.
3. **Frontend**:
    - Built using **Streamlit**, allowing users to query courses and view results interactively.

Challenges & Solutions

Problem:
Inconsistent data format across courses made it difficult to parse course descriptions consistently.

Solution:

- Applied data cleaning techniques and manual corrections to standardize the data.

- Used regular expressions to handle different course content structures.

Challenges:

- Parsing issues due to inconsistent data formats.
- Optimizing retrieval speed with the growing dataset.

Learnings:

- Efficient vector storage is crucial for fast search performance.
- Integration of **LangChain**, **FAISS**, and **Streamlit** streamlined the entire process.

Conclusion

The smart search tool effectively helps users find relevant free courses on Analytics Vidhya. Future improvements include adding additional filters (e.g., course level, rating) and enhancing the search algorithm with user feedback.

References

- **ChatGPT**
- **beautifulsoup4** : https://beautiful-soup-4.readthedocs.io/en/latest/
- **Sentence-Transformers Documentation**: https://sbert.net/docs/package_reference/sentence_transformer/SentenceTransformer.html
- **FAISS Documentation**: https://python.langchain.com/docs/integrations/vectorstores/faiss/
- **Streamlit Documentation**: https://docs.streamlit.io/deploy
- **LangChain Documentation**: https://python.langchain.com/docs/introduction/