INNOMATICS® RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Enhancing Search Engine Relevance for Video Subtitles

By,
TeamID - T211068
1.Pranav Bhawsar - IN1240543
2. Pratik Patidar - IN1240381

INNOMATICS RESEARCH LABS

Objective

Develop a robust search engine algorithm that prioritizes the analysis of video subtitles to retrieve the most relevant results for user queries.

The primary goal is to leverage natural language processing and machine learning techniques to enhance the relevance and accuracy of search results.

Introduction

Search engines play a vital role in connecting users with the ever-growing volume of digital information.

Google, a magnificent in search technology, strives to deliver accurate and user-friendly search experiences.

This project seeks to refine the search relevance of video content by leveraging video subtitles. This will enhance the accessibility of video information for a wider audience.

Types of Search Engines

Keyword Based Search Engine: These search engines match your search query to web pages that contain the same keywords, without considering the deeper meaning or context of those words.

Semantic Search Engine: Semantic search engines try to grasp the intent behind your search and the meaning of the webpages, to provide more relevant results.

Workflow

There are 2 different steps to implement search engine:

1. Ingesting the Documents
    a. Data Sampling
    b. Data Preprocessing
    c. Document Chunker
    d. Text Vectorization
    e. Storing the Embeddings in ChromaDB
2. Retrieving the Documents

Ingesting the Documents

1. Data Sampling:

We started with a .db file and converted its data into a format that a powerful tool called pandas can understand (a DataFrame). Since we had limited computing resources, we randomly picked 30% of the data to work with.

Ingesting the Documents

2. Data Preprocessing:

We created a special function that performs several tasks on the text data. This includes

- Removes timestamps that track when the data was collected.
- Converts all the text to lowercase for consistency.
- Eliminates common words (stopwords) that don't hold much meaning.
- Gets rid of anything that isn't actual text (like symbols or punctuation).

Ingesting the Documents

3. Document Chunker:

- When dealing with massive amounts of text data, a technique called "embeddings" can be used to represent the text in a way that machines can understand. However, this process can sometimes lose important information, especially with very long documents.
- To avoid this information loss and improve how well the machine understands the text, we can break down large documents into smaller, more manageable pieces. This process is called "chunking.“
- We applied the document_chunker function to each of our cleaned documents, effectively chopping them up into bite-sized pieces that are better suited for creating accurate text embeddings.

Ingesting the Documents

4. Text Vectorization:

We need a way to convert chunks of text into a format that machine can understand for further analysis. This process is called "text vectorization," and it essentially turns words into numbers.

There are different techniques for text vectorization, each with its own strengths:

a. BOW/TF-IDF: This method creates a simple representation based on word counts. It's helpful for tasks like keyword-based search engines where finding exact matches is important.

b. BERT: This is a more advanced technique that considers the meaning and context of words. It's great for semantic-based search engines where understanding the overall meaning of a query is crucial.

Ingesting the Documents

5. Storing the Embeddings:

ChromaDB is a special kind of database designed specifically for handling these vector representations.

ChromaDB isn't just a storage locker for our embeddings.

It also allows us to:

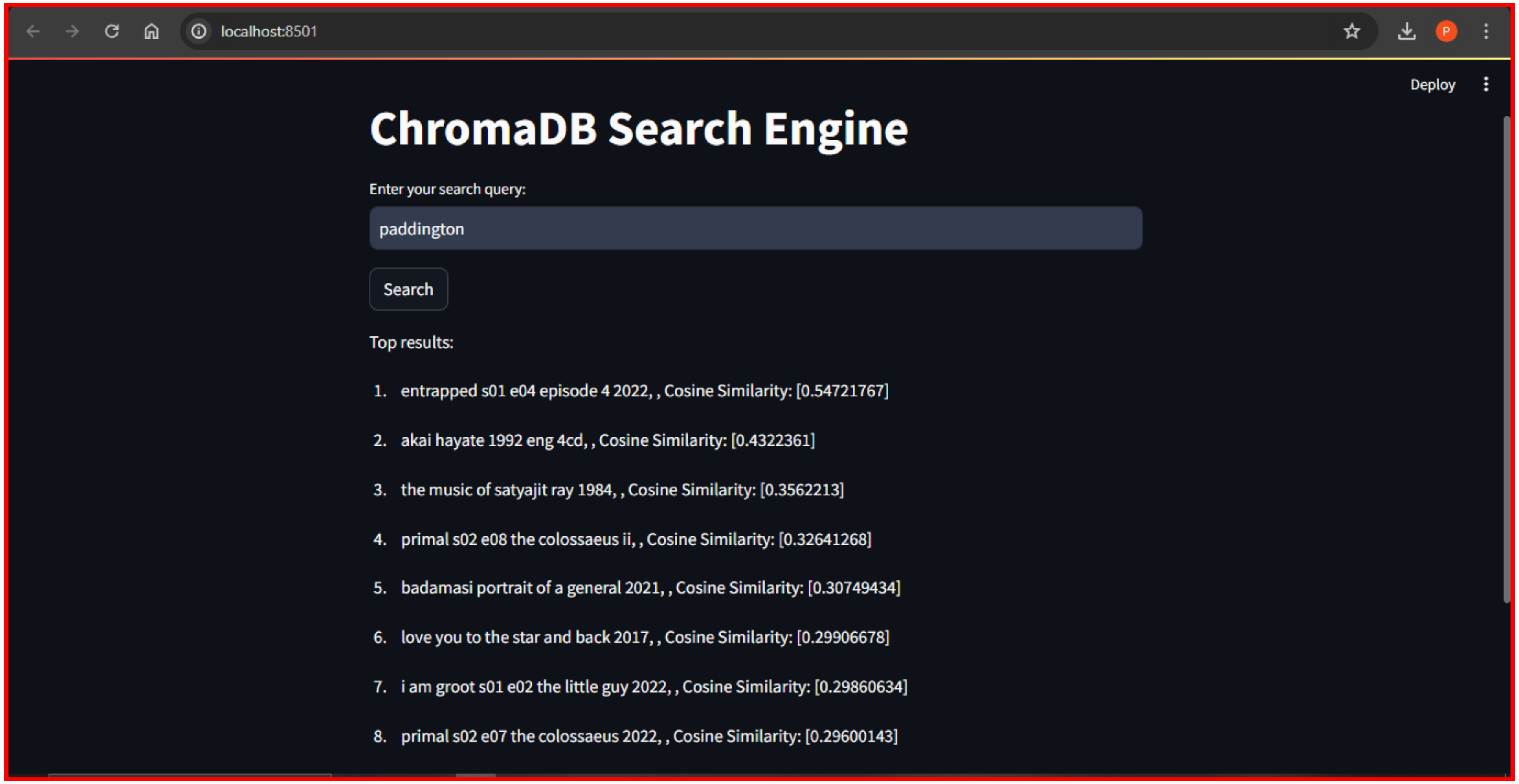
- Include extra information (metadata) alongside the embeddings, making them more informative. Filter and search through the embeddings based on this metadata during retrieval.
- We can store embeddings in the computer's memory (in-memory) for faster access, but be aware that they'll be cleared when the computer restarts.

Retrieving the Documents

1. Take the search query as input from the user.
2. Preprocess the user query i.e., clean the user input by converting it to lowercase.
3. Convert the user's query into a numerical vector representation using the same method as the subtitle documents.
4. Compute the cosine similarity between the vector representations of the subtitle documents and the user query.
5. Based on cosine similarity scores, it will return the subtitle documents with the highest cosine similarity scores, indicating their relevance to the user's query.

Results

Search Results Page



ChromaDB Search Engine

Enter your search query:

paddington

Search

Top results:

1. entrapped s01 e04 episode 4 2022, , Cosine Similarity: [0.54721767]
2. akai hayate 1992 eng 4cd, , Cosine Similarity: [0.4322361]
3. the music of satyajit ray 1984, , Cosine Similarity: [0.3562213]
4. primal s02 e08 the colossaeus ii, , Cosine Similarity: [0.32641268]
5. badamasi portrait of a general 2021, , Cosine Similarity: [0.30749434]
6. love you to the star and back 2017, , Cosine Similarity: [0.29906678]
7. i am groot s01 e02 the little guy 2022, , Cosine Similarity: [0.29860634]
8. primal s02 e07 the colossaeus 2022, , Cosine Similarity: [0.29600143]

INNOMATICS RESEARCH LABS

Conclusion

This approach aims to significantly improve the accuracy and relevance of video search results by focusing on the content of the subtitles themself.

This will lead to a more user-friendly search experience and increased accessibility of video subtitle information.

THANK YOU