**PROJECT REPORT:**

**Enhancing Search Engine Relevance for Video Subtitles**

Problem Statement:

Develop an advanced search engine algorithm that efficiently retrieves subtitles based on user queries, with a specific emphasis on subtitle content. The primary goal is to leverage natural language processing and machine learning techniques to enhance the relevance and accuracy of search results.

**Keyword based Vs Semantic Search Engines:**

**Keyword Based Search Engine:** These search engines rely heavily on exact keyword matches between the user query and the indexed documents.

**Semantic Search Engines:** Semantic search engines go beyond simple keyword matching to understand the meaning and context of user queries and documents.

**Comparison:** While keyword-based search engines focus primarily on matching exact keywords in documents, semantic-based search engines aim to understand the deeper meaning and context of user queries to deliver more relevant and meaningful search results.

Objective :

To compare a user query against a video subtitle document, and return the most similar documents.

Dataset Summary:

Database contains a sample of 82498 subtitle files from opensubtitles.org.

Most of the subtitles are of movies and tv-series which were released after 1990 and before 2024. Database File Name: eng\_subtitles\_database.db

Database contains a table called 'zipfiles' with three columns.

1. num: Unique Subtitle ID reference for www.opensubtitles.org

2. name: Subtitle File Name

3. content: Subtitle file were compressed and stored as a binary using 'latin-1' encoding.

Data Cleaning Steps:

Reading the data from the database using the sqlite3 library .Read the tables inside the database, and then reading columns of the database.

Next step is to load the database into a pandas dataframe. The subtitles text data contains data which is latin-1 encoded. The next step is to unzip the content and decode the Latin -1 data.

The next step is preprocessing of the data which involves cleaning the data, checking null values ,removing stopwords and lemmatization.

The subtitles contain timestamps which is not required, so the next step involves removing the timestamps from the data.

After preprocessing the next step is to chunk the dataset with a token-size of 500.

After the data is chunked, the data is embedded using sentence transformer model .

The next step involves storing the embedded dataset into chromaDB database.

To store the data in chromaDB we need to create a persistent client for storing data in local folder in my system.

We then create collections for storing data and retrieving data from chromaDB

I have created two collections, one on the embeddings and ids and the other on, the documents for name, random embeddings and ids.

To retrieve the data I have quered on collection 1 ,before that I have taken the user input and created embedding of it ,this is compared with the query embeddings in collection and returns top 10 ids of relevant embeddings ,internally chromaDB performs cosine similarity and gives relevant ids .

Now using the ids from collection1 query ,I have fetched the corresponding document names and printed the same as output from collection2.

At last, the project has been developed on stream lit interface .

**THANK YOU**

.