INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Enhancing Search Engine Relevance for Video Subtitles

Alekhya Vankayala (IN1240348)
Kranthi Sunkari (IN1240175)

Introduction :

➢ Significance of Search Engines:

- In the digital age, search engines play a crucial role in efficiently accessing relevant content.

➢ Focus Area:

- This project targets the enhancement of search engine relevance, particularly for video subtitles, with the goal of improving user experience and content accessibility.

➢ Methodology:

- Leveraging natural language processing (NLP) and machine learning techniques, the project aims to develop an advanced search algorithm.

INNOMATICS RESEARCH LABS

Project focus :

- The project is entirely dedicated to crafting an efficient search engine tailored to user queries, with a special emphasis on subtitle content.

Objective:

- The overarching goal is to apply advanced techniques such as natural language processing (NLP) and machine or deep learning models to refine the accuracy of search results.

Chunking :

- The dataset undergoes chunking to manage its size effectively.
- Only 30% of the data is selected for analysis to reduce computational load.
- Subtitle files are decoded after chunking.
- Seven CSV files are created post-processing.
- Each CSV contains columns for ID, title, encoded content, and decoded content of subtitle files.

Libraries Used :

- Database connector – sqlite
- Extractor – zipfile, io (for decoding)
- Directory - os
- Dataframe – pandas, numpy
- Cleaning – re
- Visualization – wordcloud
- Vectorization – torch, BertTokenizer and BertModel from transformers, scikit-learn, scipy-sparse.
- Embeddings – chromadb, Ipython.

	num	Movies&WebSeries	Subtitles
0	9251120	maybe this time 2014	it couldve been just another summer but as i ...
1	9211589	down the shore s01 e10 and justice for all 1992	oh i know that its getting late but i dont wa...
2	9380845	uncontrollably fond s01 e07 heartache 2016	timing and the uncontrollable lovebirds team ...
3	9301436	screen two s13 e04 the precious blood 1996	ethereal music api opensubtitles org is depre...
4	9408707	battlebots 2015	chris oh no not the minibots yelling oh you l...
...	...	...	...
24744	9458807	kevin can wait s01 e13 ring worm 2017	script info title default file scripttype v wr...
24745	9244890	bia s01 e29 2019	where did that come from i dont know its a ta...
24746	9345965	heroes s02 e11 chapter eleven powerless 2007	previously on heroes tell me where i can find...
24747	9417351	hot in cleveland s05 e09 bad george clooney 2014	hot in cleveland is recorded in front of a li...
24748	9460606	silk stalkings s04 e18 i know what scares you ...	api opensubtitles org is deprecated please im...

24749 rows × 3 columns

Data Processing:

- Utilize Python to extract data from database tables.
- Manage ZIP files encoded in Latin-1 format.

Data Operations:

- Perform data cleaning tasks such as eliminating timestamps, leading characters, symbols, numbers, and tags.
- Randomly select 30% of the data for analysis.

Result:

Display the cleaned and processed data, showcasing the refined dataset ready for further analysis.

INNOMATICS RESEARCH LABS

Data Preprocessing:

- Prepare text data by removing unnecessary elements like timestamps to facilitate effective vectorization.

Vectorization:

- Convert subtitle documents and user queries into vector representations for further analysis.

Cosine Similarity:

- Calculate cosine similarity scores between document and query vectors to measure their similarity.

Result Retrieval:

- Retrieve the most relevant documents based on the calculated similarity scores.

Text Vector Generation:

- Utilize Bag-of-Words (BOW) or TF-IDF for keyword-based search.Employ BERT-based SentenceTransformers for semantic search, capturing deeper contextual meaning.

Document Chunker:

- Implement a document chunking mechanism to handle large documents efficiently.Utilize overlapping windows to mitigate information loss during chunking.

Embedding Storage in ChromaDB:

- Optimize storage and retrieval efficiency by utilizing ChromaDB for storing embeddings.

Retrieving Documents :

- Accept the user's search query.
- Optionally preprocess the query for better analysis.
- Generate an embedding to represent the query's meaning.
- Calculate cosine similarity between the query embedding and document embeddings.
- Rank documents based on similarity scores to retrieve the most relevant ones for the user's query.

```python
import re
import chromadb
from sentence_transformers import SentenceTransformer
import streamlit as st

# Initialize ChromaDB client
client = chromadb.PersistentClient(path="/search_engine_db")
collection = client.get_collection(name="search_engine")
collection_name = client.get_collection(name="search_engine_FileName")
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

# Function to clean data
def clean_data(data):
    # Remove timestamps
    data = re.sub("\d{2}:\d{2}:\d{2},\d{3}\s-->\s\d{2}:\d{2}:\d{2},\d{3}", " ",  data)
    # Remove index no. of dialogues
    data = re.sub(r'\n?\d+\r', "", data)
    # Remove escape sequences like \n \r
    data = re.sub('\r|\n', "", data)
    # Remove <i> and </i>
    data = re.sub('<i>|</i>', "", data)
    # Remove links
    data = re.sub("(?:www\.)osdb\.link\/[\w\d]+|www\.OpenSubtitles\.org|osdb\.link\/ext|api\.OpenSubtitles\.org|OpenSubtitles\.com"
    # Convert to lower case
    data = data.lower()
    return data

# Function to extract IDs
def extract_id(id_list):
    new_id_list = []
    for item in id_list:
        match = re.match(r'^(\d+)', item)
        if match:
            extracted_number = match.group(1)
            new_id_list.append(extracted_number)
    return new_id_list

# Streamlit UI
st.header("Movie Subtitle Search Engine")
search_query = st.text_input("Enter a dialogue to search....")
if st.button("Search"):
    st.subheader("Relevant Subtitle Files")
    search_query = clean_data(search_query)
    query_embed = model.encode(search_query).tolist()
    search_results = collection.query(query_embeddings=query_embed, n_results=10)
    id_list = search_results['ids'][0]
    id_list = extract_id(id_list)
    for id in id_list:
        file_name = collection_name.get(ids=f"{id}")["documents"][0]
        st.markdown(f"[{file_name}](https://www.opensubtitles.org/en/subtitles/{id})")
```

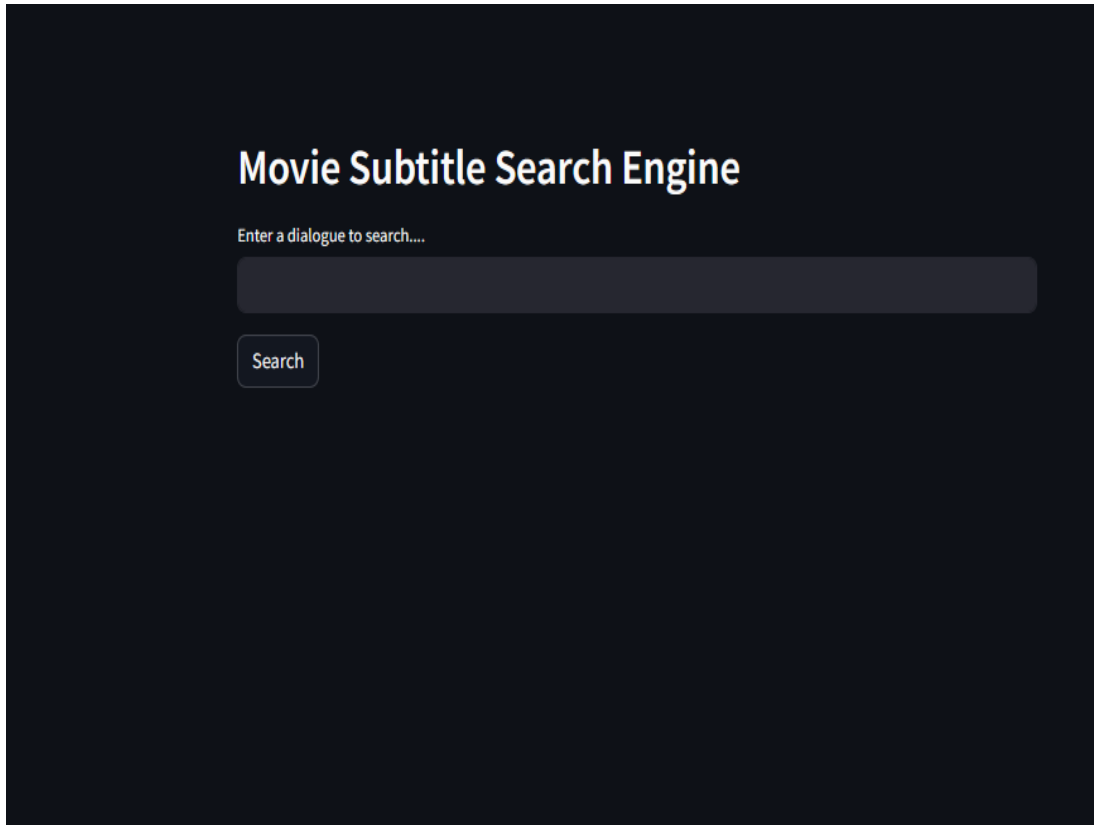
The Flask app functions as a **movie subtitle search engine**. It performs the following tasks:

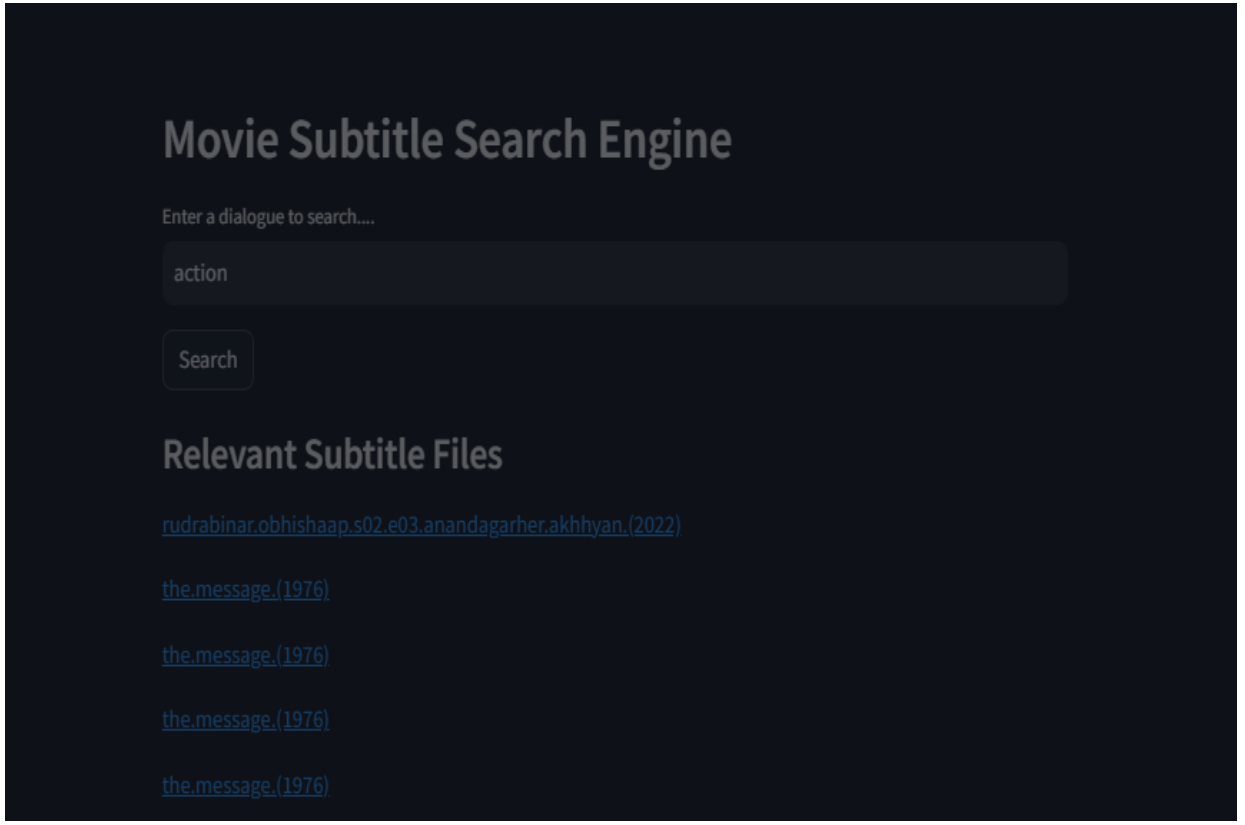
- **Reads data** from a CSV file and loads it into a Pandas DataFrame.
- **Handles user search queries** inputted through the app.
- Presents **search results** using **HTML templates** that are styled using **CSS.**

INNOMATICS RESEARCH LABS

Output Result:

- The code sets up a web application enabling users to search for movie subtitles. Upon submission, the search results are showcased on a page featuring a carefully desi





INNOMATICS RESEARCH LABS

Conclusion :

- The project's aim is to evaluate the effectiveness of keyword and semantic-based searching.
- It underscores the significance of semantic search in considering word meanings, unlike exact-match keyword search.
- Despite resource constraints, a rudimentary search engine was constructed.
- Future endeavors may involve further exploration with increased data and model training to enhance search capabilities.

INNOMATICS
RESEARCH LABS