INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Enhancing Search Engine Relevance for Video Subtitles

Ramesh

Introduction:

- **Overview:** In today's digital era, the accessibility of video content relies heavily on the effectiveness of search engines. Enhancing search engine relevance for video subtitles is crucial for connecting users with the content they seek.
- **Importance:** Search engines play a pivotal role in navigating the vast landscape of digital media. A well-designed search algorithm can significantly improve the user experience by providing accurate and relevant results.
- **Project Focus:** This project aims to develop a sophisticated search engine algorithm that prioritizes the content of video subtitles. By leveraging natural language processing (NLP) and machine learning (ML) techniques, we seek to enhance the accuracy and relevance of search results, ultimately improving the accessibility of video content for all users.

Objective :

- **Goal:** Create a sophisticated search engine algorithm for efficient subtitle retrieval based on user queries.
- **Emphasis:** Special focus on subtitle content to enhance relevance.
- **Approach:** Leverage Natural Language Processing (NLP) and Machine Learning (ML) techniques.

Project Steps Overview:

- Read the given data
- Understand the data
- Decode the files inside the database
- Data sampling (optional)
- Cleaning steps
- Text vectorization
- Document chunking
- Store embeddings

INNOMATICS RESEARCH LABS

Keyword-based vs Semantic Search Engines:

Keyword-Based Search Engine:

- Relies on exact keyword matches between user queries and indexed documents.
- Matches are based on specific keywords without considering the context or meaning of the words.
- Examples include traditional search engines like early versions of Google and Bing.

Semantic Search Engines:

- Go beyond simple keyword matching to understand the meaning and context of user queries and documents.
- Use natural language processing (NLP) and machine learning to understand user intent and deliver more relevant results.
- Examples include modern search engines like Google Search with its Knowledge Graph and BERT algorithm.

Data Preparation Steps:

Reading the Data: Read the provided database file to access the subtitle data.

- **Understanding the Data:** Review the README.txt file to understand the structure and contents of the database.
- **Decoding the Files:** Decode the encoded subtitle files to extract the text content.
- **Data Sampling (Optional):** Sample a random 30% of the data for processing if compute resources are limited.

Data Preparation - Text Preprocessing:

Apply appropriate cleaning steps to the subtitle documents, such as removing timestamps, special characters, spaces, and converting text to lowercase.

	num	name	content	file_content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...	1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...	1\r\n00:00:29,359 --> 00:00:32,048\r\nAh! Ther...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...	1\r\n00:00:53,200 --> 00:00:56,030\r\n<i>Yumi'...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...	1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...	ï»¿1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch...

+ Code + Markdown

```
df.info()
```

```
1
00:00:06,000 --> 00:00:12,074
Watch any video online with Open-SUBTITLES
Free Browser extension: osdb.link/ext

2
00:00:15,370 --> 00:00:16,506
You lose everything, my girl.

3
00:00:16,530 --> 00:00:19,360
So you've said - four times.

4
00:00:20,330 --> 00:00:22,120
I definitely had
it on yesterday.

5
```

```python
import re

# Function to clean text
def clean_text(text):
    # Remove timestamps and other patterns (assuming timestamps are like [00:00:00])
    text = re.sub(r'\[\d+:\d+:\d+\]', '', text)

    # Remove special characters, numbers, and extra spaces
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Replace dots with space
    text = re.sub(r'\.', ' ', text)

    # Convert to lowercase
    text = text.lower()

    # Remove unnecessary whitespace characters
    text = re.sub(r'\s+', ' ', text).strip()

    return text
```

```python
from tqdm import tqdm
tqdm.pandas()
```

```python
# Clean the 'name' and 'file_content' columns
df['cleaned_file_content'] = df['file_content'].progress_apply(clean_text)

# Display the cleaned columns
print(df[['cleaned_file_content']])
```

```
100%|██████████████████████████████████████████████████████████████████████| 30
000/30000 [03:25<00:00, 146.00it/s]

                                cleaned_file_content
0      riley ipreviously oni the client list derek yo...
1      support us and become vip member to remove all...
2      script info title english us original script p...
3      wind whooshing eerie synth music bats chirping...
4      watch any video online with opensubtitles free...
...                                                  ...
29995  please do not attempt to perform any of these ...
29996  crickets chirring music box music apiopensubti...
29997  i previously oni velma the last place my moms ...
29998  one two three four five numberblocks six seven...
```

INNOMATICS RESEARCH LABS

Tokenization for Word2Vec:

Tokenization is a crucial step in natural language processing (NLP) that involves breaking down text into smaller units called tokens. These tokens can be words, subwords, or characters, depending on the tokenization strategy used. For Word2Vec, tokenization is essential as it prepares text data for training word embeddings. The process involves splitting text, removing punctuation and special characters, and converting text to lowercase. By tokenizing text, Word2Vec can learn meaningful representations of words, capturing semantic relationships between them. This is useful for various NLP tasks such as sentiment analysis, named entity recognition, and machine translation. Tokenization helps improve the accuracy of word embeddings by providing a structured representation of text data.

	num	name	cleaned_file_content	tokenised_content_lemma	content_size_lemma	doc_vector
0	9487821	the.client.list.s02.e06.unanswered.prayers.(20...	riley ipreviously oni the client list derek yo...	['riley', 'ipreviously', 'oni', 'client', 'lis...	3244	[-1.09343290e-01 -7.68003017e-02 -1.80567741e-...
1	9183678	the.devil.is.a.woman.(1935).eng.1cd	support us and become vip member to remove all...	['support', 'u', 'become', 'vip', 'member', 'r...	2853	[-9.83889028e-02 -7.53047839e-02 -1.65519699e-...
2	9258855	kingdom.s04.e18.rivercrossing.battle.(2022).en...	script info title english us original script p...	['script', 'info', 'title', 'english', 'u', 'o...	1732	[-4.89264429e-02 -1.39095232e-01 -1.84526831e-...
3	9421238	sorority.babes.in.the.slimeball.bowlorama.2.(2...	wind whooshing eerie synth music bats chirping...	['wind', 'whooshing', 'eerie', 'synth', 'music...	2948	[-9.89964455e-02 -6.88564703e-02 -1.66541591e-...
4	9518525	ada.twist.scientist.s03.e03.dadbotinto.the.bra...	watch any video online with opensubtitles free...	['watch', 'video', 'online', 'opensubtitles', ...	1482	[-0.10334199 -0.06256191 -0.15584423 -0.213279...

INNOMATICS RESEARCH LABS

Pretrained BERT for Sentence Vectors:

The Sentence Transformers library provides pretrained BERT models for generating sentence embeddings, which are crucial for various natural language processing (NLP) tasks. These models allow for the efficient creation of high-quality sentence vectors that capture the semantic meanings of sentences. By using the SentenceTransformer class, you can load a pretrained BERT model, such as 'all-MiniLM-L6-v2', and then use it to encode sentences and generate corresponding sentence vectors. These pretrained BERT models have several benefits, including their ability to improve search relevance in information retrieval systems and support semantic search and similarity matching between sentences. For example, you can encode sentences using the pretrained BERT model and calculate similarity scores for a given query, thereby enhancing the performance of your search engine.

	num	name	cleaned_file_content	tokenised_content_lemma	content_size_lemma	doc_vector
0	9487821	the.client.list.s02.e06.unanswered.prayers.(20...	riley ipreviously oni the client list derek yo...	['riley', 'ipreviously', 'oni', 'client', 'lis...	3244	[-1.09343290e-01 -7.68003017e-02 -1.80567741e-...
1	9183678	the.devil.is.a.woman.(1935).eng.1cd	support us and become vip member to remove all...	['support', 'u', 'become', 'vip', 'member', 'r...	2853	[-9.83889028e-02 -7.53047839e-02 -1.65519699e-...
2	9258855	kingdom.s04.e18.rivercrossing.battle.(2022).en...	script info title english us original script p...	['script', 'info', 'title', 'english', 'u', 'o...	1732	[-4.89264429e-02 -1.39095232e-01 -1.84526831e-...
3	9421238	sorority.babes.in.the.slimeball.bowlorama.2.(2...	wind whooshing eerie synth music bats chirping...	['wind', 'whooshing', 'eerie', 'synth', 'music...	2948	[-9.89964455e-02 -6.88564703e-02 -1.66541591e-...
4	9518525	ada.twist.scientist.s03.e03.dadbotinto.the.bra...	watch any video online with opensubtitles free...	['watch', 'video', 'online', 'opensubtitles', ...	1482	[-0.10334199 -0.06256191 -0.15584423 -0.213279...

Efficiently Store and Retrieve Embeddings with ChromaDB:

ChromaDB is a specialized database designed for efficient storage and retrieval of embeddings and other high-dimensional vectors. It is particularly useful in applications that require handling large amounts of embedding data, such as natural language processing (NLP) and machine learning (ML). One of the key advantages of ChromaDB is its ability to store embeddings without needing a predefined schema, providing flexibility in data storage. Additionally, ChromaDB offers fast retrieval of embeddings based on their IDs or other query parameters, making it suitable for applications where quick access to embeddings is critical, such as similarity search. Overall, ChromaDB is a valuable tool for managing embeddings, offering efficiency, flexibility, and speed in handling high-dimensional vector data.

```python
import sqlite3
import pandas as pd
import json

# SQLite database path
db_path = 'chromadb1136.db'

# Connect to the SQLite database
conn = sqlite3.connect(db_path)

# Convert lists to strings
emb_df['content_chunks'] = emb_df['content_chunks'].apply(json.dumps)
emb_df['embeddings'] = emb_df['embeddings'].apply(json.dumps)

# Save DataFrame to SQLite
emb_df.to_sql("dd", conn, if_exists="replace", index=False)

print(f"DataFrame from {emb_df} is successfully saved to ChromaDB using SQLite: {db_path}")
```

App Deployment:

Deploying a Streamlit app is a simple process. After preparing your app, you can deploy it using the Streamlit Sharing service with a single command. This command uploads your app to Streamlit's servers and provides you with a unique URL to access your deployed app. Sharing this URL allows others to interact with your app over the web, making it easy to share your work with a broader audience.

Conclusion:

the project focuses on improving search engine relevance for video subtitles through advanced algorithms leveraging natural language processing and machine learning. By emphasizing semantic understanding over keyword matching, the project aims to deliver more accurate and meaningful search results. Implementing a document chunker addresses the challenge of embedding large documents, enhancing the overall search experience. Overall, the project seeks to enhance the accessibility and relevance of video content through innovative search engine techniques.

THANK YOU

INNOMATICS RESEARCH LABS