INNOMATICS®
RESEARCH LABS

INNOVATION. AUTOMATION. ANALYTICS

PROJECT ON

Subtitle Semantic Search Engine

By,
TeamID-T211111
1. Vishal Kumar-IN1241003
2. N Paul Gabriel Richman-IN1241338

Objective

- Develop an advanced search engine algorithm specifically designed for retrieving video subtitles based on user queries.
- Focus on prioritizing the analysis of video subtitles to deliver the most relevant results to users.
- Leverage natural language processing (NLP) and machine learning (ML) techniques to significantly enhance the relevance and accuracy of search results.
- Aim to improve the accessibility and usability of video content by providing users with more precise and contextually relevant subtitle search results.

Introduction

- The project aims to develop an advanced search engine for movie and series subtitles.
- It utilizes natural language processing (NLP) and machine learning (ML) techniques to enhance relevance and accuracy.
- Emphasis is placed on understanding the semantic context of both user queries and subtitle content, moving beyond basic keyword matching.
- Implemented in Python utilizing libraries such as pandas, scikit-learn, numpy, and Flask.
- The goal is to create a robust and efficient subtitle similarity search engine, enhancing accessibility and usability of video content.

Reading the Data from the Database

- Establish a connection to the database file 'eng_subtitles_database.db'.
- Retrieve the names of tables present in the database.
- Extract column information of the 'zipfiles' table.
- Read data from the 'zipfiles' table into a DataFrame.

Since we had limited computing resources, we randomly picked 30% of the data to work with.

new_project_notebook.ipynb

Markdown

```
cols = cursor.fetchall()
for col in cols:
    print(col[1])
```

```
num
name
content
```

The above code helps in checking the column names in the database table.

Let's now use `SELECT * FROM zipfiles` to read all the data into a `df` variable.

Step 3: *Loading the Database Table inside a Pandas DataFrame*

[ ]:
```
df = pd.read_sql_query("""SELECT * FROM zipfiles""", conn)
df.head()
```

[4]:

	num	name	content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...
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...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...

INNOMATICS RESEARCH LABS

Preprocessing Text Data:

- A special function is created to process text data.
- Tasks performed include:
  - Removing timestamps indicating data collection time.
  - Converting all text to lowercase for consistency.
  - Eliminating common words (stopwords) lacking significant meaning.

Settings Help

new_project_notebook.ipynb

Code Python 3 (ipykernel)

[ ]:
```python
import re

def clean_subtitle(subtitle):
    # Remove timestamp and special characters
    subtitle = re.sub(r'<[^>]*>', '', subtitle)  # Remove HTML tags
    subtitle = re.sub(r'\r\n', ' ', subtitle)  # Replace newlines with spaces
    subtitle = re.sub(r'[^a-zA-Z\s]', '', subtitle)  # Remove non-alphabetic characters

    # Convert to lowercase
    subtitle = subtitle.lower()

    # Remove extra spaces
    subtitle = re.sub(r'\s+', ' ', subtitle).strip()

    return subtitle
```

[ ]:
```python
clean_subtitle(data.file_content[0])
```

[21]:
```
'watch any video online with opensubtitles free browser extension osdblinkext it couldve been just another summer but as i set foot on the sand that su
mmer suddenly felt different like it was going to be the summer that would change my life the summer of freedom the summer of endless possibilities the
summer of ooh aah ooh oh oh oh ooh that was the summer of you and me youre quite the dancer why did you stop come on keep dancing whatever im kidding d
ont get mad huh what hey im just going to get my towel what stop that you thought i was gonna kiss you no excuse me i wanna kiss you but not just yet w
hat do you mean not yet only when youre my girl what do you mean your girl my girlfriend miss as if you wish and dont call me miss dont pretend to be a
gentleman when youre clearly not so what should i call you rude snob bitch and you douche handsome conceited just like you huh jerk exactly your type l
eave me alone steph aha steph ill just call you tep remove the s and the f by the way im tonio will you still be here tomorrow dont leave yet im going
to court you i chose to walk away from you but fate had a different plan councilor were teaching basic english literacy well be teaching the children h
ow to read and write in english is that so yes how long will this program run if its okay the entire summer i thought i could escape you but you someho
w found me again tep what are you doing here you couldnt resist me huh so youre courting me instead excuse me im not here to court you oh so youre here
to be courted yeah no uncle tonio shes the one who will teach us this summer dont believe that story thats just her excuse to get me to date her gramps
uncle erning aunt elma this is tep my suitor approved approved you passed mmm i dont approve excuse us well be going now all right my dear thank you si
r well go ahead excuse me please oh no tep tep are you okay i said excuse me were not yet a couple and were already fighting he has a point oh tonio se
venty snacks seven seven silvery silvery sheep sheep do you know what that is in filipino no filipino seventyseven silvery sheep why say it in english
if you can say it in filipino because they need to learn how to read write and speak properly in order to make a living why are you here anyway tep the
```

Document Chunking:

- Dealing with very large text data involves utilizing the “embeddings” technique to represent text for machine comprehension.
- Concerns arise over the potential loss of vital information, particularly with lengthy documents.
- The solution involves segmenting large documents into smaller, more manageable pieces, known as "chunks."
- Applied the document_chunker function to divide each of our cleaned documents into smaller, digestible sections, enhancing accuracy for generating precise text embeddings.

```python
def document_chunker_by_words(document, chunk_size_words, overlap_words):
    words = document.split()
    chunks = []
    start = 0
    end = chunk_size_words

    while start < len(words):
        chunk = ' '.join(words[start:end])
        chunks.append(chunk)
        start += chunk_size_words - overlap_words
        end = start + chunk_size_words

    return chunks

# Set chunking parameters
chunk_size_words = 500  # Adjust according to your requirements
overlap_words = 20  # Adjust according to your requirements

# Perform chunking on the dataset
chunked_data = []
for index, row in data.iterrows():
    chunks = document_chunker_by_words(row['clean_subtitle2'], chunk_size_words, overlap_words)
    for chunk in chunks:
        chunked_data.append({
            'num': row['num'],
            'name': row['name'],
            'chunk': chunk
        })

# Convert the chunked data to a DataFrame
chunked_df = pd.DataFrame(chunked_data)
```

Vectorization

- Employed the Sentence Transformers library to convert textual data into numerical representations, making it understandable for machines.
- Applied this conversion process to a specific portion of text data, enabling further processing.
- Created numerical representations, known as embeddings, for each segment of text, capturing its essence in a machine-readable format.
- Saved the DataFrame containing these embeddings to a CSV file, ensuring accessibility for future utilization or analysis of the encoded representations.

```
[ ]: from sentence_transformers import SentenceTransformer
     model = SentenceTransformer('all-MiniLM-L6-v2')
```

...

...

```
[ ]: def encode_text(text):
       return model.encode(text)
```

```
[ ]: chunked_df['embedding'] = chunked_df['chunk'].apply(encode_text)
```

```
[ ]: # Save DataFrame to a file (e.g., CSV)
     chunked_df.to_csv('/content/drive/MyDrive/Search Engine Dataset/embedding_clean_chunk_data.csv', index=False)
```

```
[ ]: print(chunked_df.shape)
     chunked_df
```

(262891, 4)

[12]:

	num	name	chunk	embedding
0	9251120	maybe.this.time.(2014).eng.1cd	watch any video online with opensubtitles free...	[-0.087850414, -0.1228606, 0.027850531, -0.052...
1	9251120	maybe.this.time.(2014).eng.1cd	just teach them how to swim in a sinking boat ...	[-0.07370347, -0.047148414, 0.056917146, -0.01...
2	9251120	maybe.this.time.(2014).eng.1cd	you know tep once i have enough money from wor...	[-0.032818347, -0.066111684, 0.107563965, -0.0...
3	9251120	maybe.this.time.(2014).eng.1cd	that moment i wanted to be part of your world ...	[-0.099456385, -0.10122586, 0.0766382, 0.00370...
4	9251120	maybe.this.time.(2014).eng.1cd	how will the girl feel a angry b surprised or ...	[-0.06304503, -0.013828885, 0.028796686, -0.05...
...	...	...	...	...
262886	9460606	silk.stalkings.s04.e18.i.know.what.scares.you....	this is our dream house no this is your dream ...	[-0.038717896, -0.02087232, 0.047487248, -0.03...

INNOMATICS RESEARCH LABS

Cosine Similarity

- Developed a function to find similar documents by calculating cosine similarity between a search query and document embeddings.
- Utilized the cosine similarity metric, which measures the cosine of the angle between two vectors, to quantify the similarity between text representations.
- Retrieved the top ‘n' most similar documents based on their similarity scores.
- The function returns a list of the most similar documents, allowing users to identify relevant texts related to their search query.

```
[56]: similar_text = similarity_finder([emb_query2], 10, embeddings_array)

for text in enumerate(similar_text):
    print(text)
    print()
```

```
(0, 'operation.fortune.ruse.de.guerre.(2023).eng.1cd')

(1, 'operation.fortune.ruse.de.guerre.(2023).eng.1cd')

(2, 'survivors.remorse.s01.e03.how.to.build.a.brand.(2014).eng.1cd')

(3, 'queer.eye.s06.e05.crawzaddy.(2021).eng.1cd')

(4, 'tales.of.wells.fargo.s05.e20.the.hand.that.shook.the.hand.(1961).eng.1cd')

(5, 'survivors.remorse.s01.e03.how.to.build.a.brand.(2014).eng.1cd')

(6, 'nightingales.s02.e05.reach.for.the.sky.(1993).eng.1cd')

(7, 'operation.fortune.ruse.de.guerre.(2023).eng.1cd')

(8, 'magnum.p.i.s02.e19.may.the.best.one.win.(2020).eng.1cd')

(9, 'the.voice.s22.e23.live.semifinal.top.8.eliminations.(2022).eng.1cd')
```

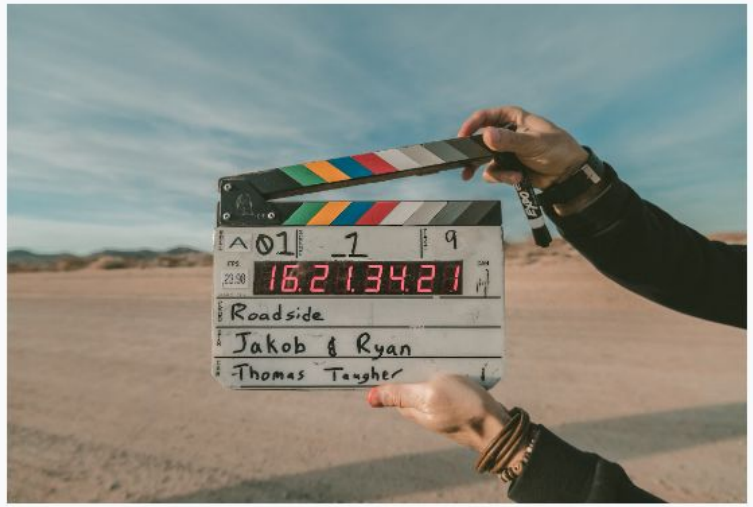
INNOMATICS RESEARCH LABS

Storing the Embeddings

Building the Flask Web App

- The application is built using Flask, a Python web framework.
- Users can input their queries, and we'll find matching subtitles from our collection.
- The preprocessed query is then transformed into a numerical vector representation, employing the same method utilized for the subtitle documents.
- The application computes the cosine similarity between the vector representations of the subtitle documents and the user query.
- The app displays top matches, scores showing how similar they are.

Results



Welcome to Video Search Engine

Enter your search query

Search

INNOMATICS RESEARCH LABS

Results

Top 10 Movies and TV-Shows 😍 😍

operation.fortune.ruse.de.guerre.(2023).eng.1cd

operation.fortune.ruse.de.guerre.(2023).eng.1cd

survivors.remorse.s01.e03.how.to.build.a.brand.(2014).eng.1cd

queer.eye.s06.e05.crawzaddy.(2021).eng.1cd

tales.of.wells.fargo.s05.e20.the.hand.that.shook.the.hand.(1961).eng.1cd

survivors.remorse.s01.e03.how.to.build.a.brand.(2014).eng.1cd

nightingales.s02.e05.reach.for.the.sky.(1993).eng.1cd

operation.fortune.ruse.de.guerre.(2023).eng.1cd

magnum.p.i.s02.e19.may.the.best.one.win.(2020).eng.1cd

the.voice.s22.e23.live.semifinal.top.8.eliminations.(2022).eng.1cd

Back to Home

Links:

Video Link:
https://www.linkedin.com/feed/update/urn:li:ugcPost:7189683442677080064/

Github: https://github.com/VishalDeoPrasad/Search-Engine-Project

LinkedIn:
https://www.linkedin.com/feed/update/urn:li:ugcPost:7189683442677080064/

INNOMATICS RESEARCH LABS

Conclusion

After implementation of our subtitle similarity search system, it's evident that our approach focusing on the content of subtitles has significantly improved the accuracy and relevance of video search results. By diligently following the steps outlined, including importing libraries, preprocessing text data, computing vectors, calculating cosine similarity, and building a user-friendly Flask web app, we've successfully created an efficient tool tailored to our specific requirements.

THANK YOU

INNOMATICS
RESEARCH LABS