Semantic search in video datasets

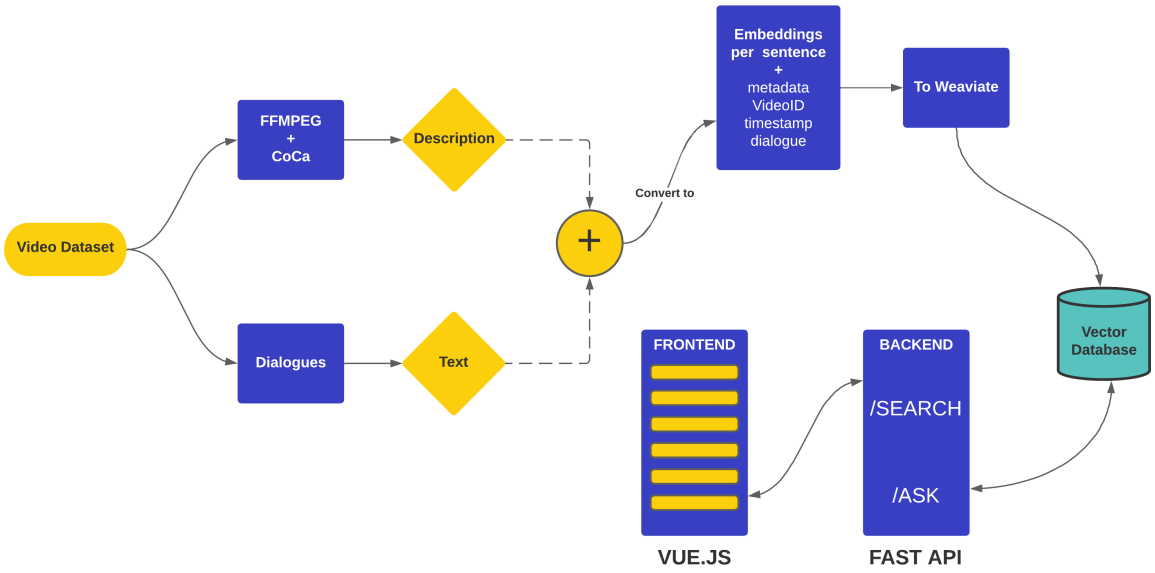
Shauray Singh

March 31, 2023

Summary of the Proposal

The proposed project is to develop a multimodal semantic search engine for a set of videos with their corresponding time-aligned transcripts. The search engine will generate descriptions of a limited number of frames per phrase/sentence using a model for image description, such as CLIP [1] or CoCa, and generate an embedding combining the phrase and the frame descriptions. The embeddings will be stored in a vector database [4], and when performing semantic searches, the engine will produce the embedding of the phrase searched and find the embeddings in the database that are closest to the embedding searched. The website will be able to list the N first moments in the videos that are most similar to the embedding searched. The project may also include a bonus question/answer platform that uses "augmented knowledge" to generate responses.

Background



An essential aspect of this project is to enable automatic video captioning using state-of-the-art deep learning models. To achieve this, we propose a multi-step approach that involves selecting relevant video frames based on the dialogue content, processing them with a subtitle model, and then merging the generated textual description with the original dialogue.

Specifically, our goal is to use natural language processing (NLP) techniques to analyze spoken language in videos and identify the most informative frames that capture the essence of the conversation. These frames can be selected based on various criteria, such as the speaker's facial expressions, gestures, or scene context. But to keep things simple we just take 3 frames from every dialogue.

Once we have identified the frames of interest, we will use a deep learning model on video captions, which will generate a textual description of each frame.

The resulting text descriptions can then be concatenated with the corresponding dialogue and stored in a vector database with metadata such as video IDs, time frames and other relevant information. The

database can be queried using a fastAPI-based backend that provides a RESTful interface to access and manipulate stored data.

To present information to users, we plan to develop an interface using Vue and Vite frameworks that can interact with the backend and display relevant information in a user-friendly way. This frontend can also include features like search, filtering, and sorting, which can help users quickly find the video content they are looking for. This project aims to provide a scalable solution for a semantic search for video data that can be used in various contexts, such as video indexing, search, and recommendation systems. By combining the power of NLP and deep learning, we can extract rich and meaningful information from videos and make it easily accessible to users.

Personal Background

I am Shauray Singh (mail: shauraysingh08@gmail.com, website: `https://shauray8.github.io/about_shauray`) a Junior at Manipal University, pursuing a Bachelor of Technology in Data Science and Engineering. I developed a passion for hardware when I was 8 and then I taught myself programming in my school days and that same passion has transferred over to the field of Machine Learning.

The project 'semantic search for videos' is particularly meaningful to me as often there are things about a video you remember but you don't exactly know what video it was, and a semantic search for the same will make things easier and faster. The experience I gained by researching and applying for this project left me with a new appreciation and interest in large language model research.

I am very eager for the opportunity to become a successful contributor to this project.

Goal and Objectives

The goal of this project is to develop better semantic search engines and here's a rough road map of the same -

- Retrieve data from .vrts files and segment frames according to dialogues for semantic segmentation and then merge the raw data with annotations and store them in a JSON file which would at the end store sentences, timestamps, videoId, description, etc.
- Generate textual descriptions of keyframes using ffmpeg, and CoCa, and store them in the JSON file along with the corresponding embeddings of the textual descriptions.
- Store the embeddings and metadata in a vector database, Weaviate, to enable semantic search and other vector-based operations.
- Implement a frontend interface using Vue3.js to provide an easy-to-use interface for users to perform semantic searches on the video dataset.
- Implement a backend API using FastAPI to take user search queries, generate the corresponding embeddings using the chosen embedding model, and
- find the K nearest embeddings in Weaviate's vector database.
- Use language generation models to enhance the search experience for users by generating relevant answers based on the search results.

Additional Goals and Suggestions

- Using Whisper can be a better idea while going through a second iteration but my only concern is that It is used to store and retrieve numeric data points over time and not text data. .vtt files may help in this condition.
- Create .vtts with Whisper of the videos and then create two databases and show the differences between both methods.
- Dividing the segment for every dialogue in 3 frames (first, middle, last) and using those to generate a description using CoCa can make things significantly faster and maybe better in terms of performance.

- Another improvement here would be to determine how likely are those 3 frames and use only the different frames. If the 3 are really similar, just simply generate the description of one of them.
- Using HuggingFace libraries as it's easy to change and experiment with different models.
- Improving data retrieval from the database can be made possible by caching the layer, optimizing database queries, or parallelizing requests. But as I was told the project contains 50-100 hours of video data this won't be a practical thing to do unless the mentors plan to scale it.
- Using semantic segmentation to enhance the quality of semantic search. Saving frames created by ffmpeg, which could then be shown in the results list. This is only possible when we don't have any memory constraints. Or we can present a list of texts with a link to the video. To make it prettier and a JS video player previews the video at the moment, which is more memory efficient than the above method.
- Using LangChain [3] for user-generated inputs. It will provide a suite of pre-trained language models and tools for building and evaluating NLP models. Consider optimizing database queries or parallelizing requests to improve the speed of information retrieval.

Methods and Benefits to the Community

Firstly, it will provide a powerful tool for searching and analyzing large video datasets with time-aligned transcripts. This will be particularly useful for researchers, journalists, and other professionals who work with video content and need to quickly find specific information within large video archives.

Secondly, the project will make use of open-source deep learning models and technologies, making them accessible to a wide range of developers and researchers. The use of popular frameworks like Python FastAPI and Vue3.js will also make it easier for developers to contribute to the project and build on top of its functionality.

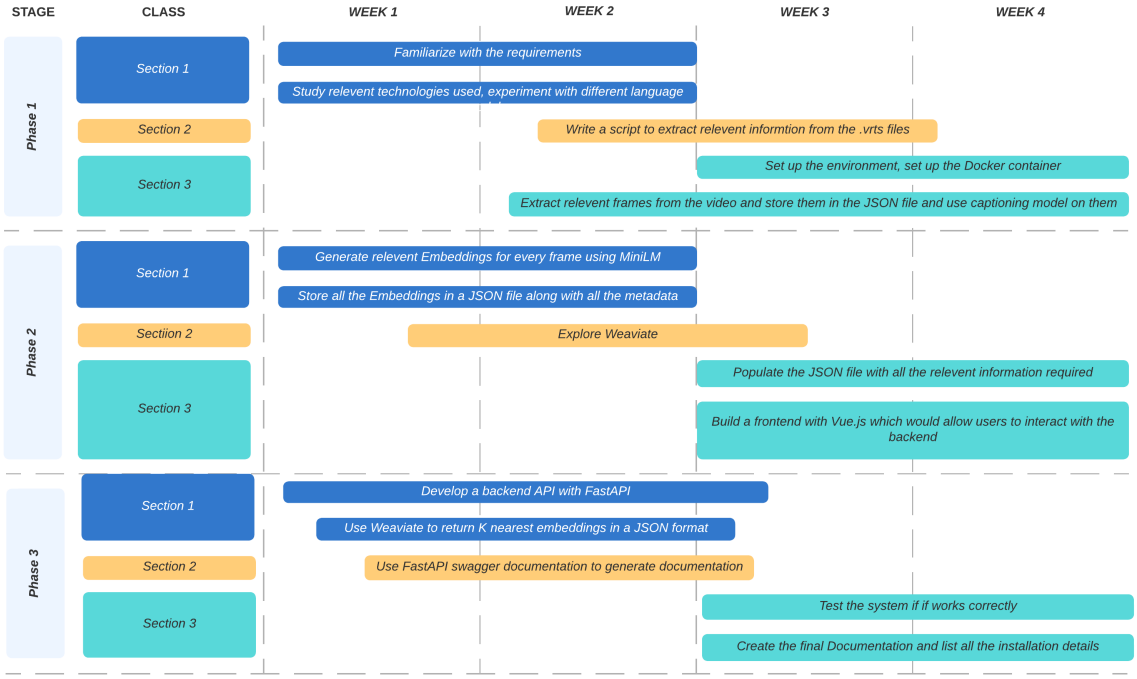
Thirdly, the project has the potential to improve the accessibility and usability of video datasets for a variety of applications, including education, entertainment, and social media. By providing a more effective way to search and analyze video content, this project can help unlock new insights and possibilities in these fields.

Fourthly, This could be an addition, the project can scale by using Whisper to generate timestamped transcripts so anyone can create their own semantic search video engine without having any transcripts for the video.

Overall, this project has the potential to make a significant contribution to the field of video analysis and search, and to the wider community of developers, researchers, and professionals who work with video content.

Tentative Timeline

The project utilizes a time line of 12 weeks and here's an overview of those 12 weeks.



Week 1-2:

- Familiarize with the project requirements and objectives.
- Study the technologies and tools that will be used in the project such as ffmpeg, CLIP, miniLM [2], Weaviate, Vue3.js, and fastAPI.
- Discuss with mentors any questions or concerns about the project.

Week 3-4:

- Set up the development environment and install the required tools.
- Set up a Docker container to containerise applications, which can help simplify deployment and ensure consistency across different environments.
- Develop a script to retrieve relevant information from the .vrts files.
- Write code to extract 1-3 frames from each video segment and generate descriptions using CoCa.
- Store the frames and descriptions in a JSON file.

Week 5-6:

- Implement the code to generate embeddings for the frames using miniLM.
- Store the embeddings in the JSON file alongside the frames and descriptions.
- Explore Weaviate and its alternatives to better store and retrieve information when and where needed.

Week 7-8:

- Populate the JSON file with relevant information required for semantic search.
- Build a frontend interface using Vue3.js that allows users to interact with the dataset and perform semantic searches on the embeddings.
- Use Vite to develop a server for the same. It provides a fast and lightweight development experience.
- Maybe use Pinia as well to manage the state of a Vue application.

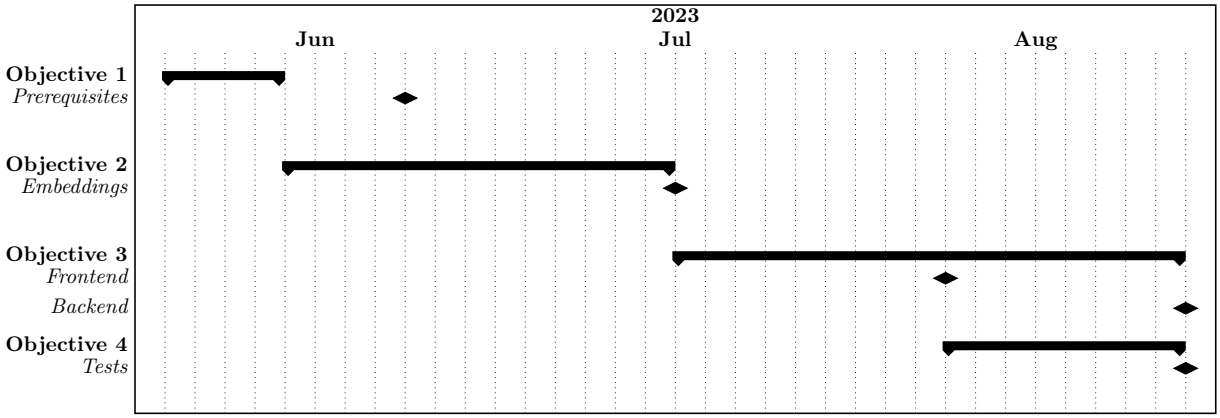
Week 9-10:

- Develop the backend API using fastAPI to take user search queries and generate embeddings for them.

- Use FastAPI swagger documentation to create documentation for the API.
- Use Weaviate's vector database to find the K nearest embeddings to the user's query.
- Return a JSON response containing the search results in a format that can be easily displayed to the user.

Week 11-12:

- Test the system to ensure it is functioning correctly.
- Create documentation for the project, including how to install and use the system.
- Submit the final project proposal and documentation to the GSOC program.



Improvements

- Write Unit Tests, This project has a relatively low complexity so PyTest works fine.
- Optimize the database queries, parallelize requests, and set up caching to speed up information retrieval this might not be very useful as the database is very optimized.
- Explore the possibility of improving the model by Using semantic segmentation to enhance the quality of semantic search. This could improve the semantic search based on experimentation.

Skill Set

- Deep learning and natural language processing, with experience in developing and evaluating deep learning models for image and text processing.
- Experience with image and text processing.
- Have worked before with building Efficient and accurate semantic and hybrid searches.
- Python FastAPI and Vue3.js
- Knowledge of vector databases such as Weaviate
- Proficiency in using open-source models such as MiniLM-L6, T5, CLIP, CoCa, and other SOTA LLMs

Previous Work

As a developer, I have extensive experience working with various natural language processing (NLP) technologies and building systems that require advanced search capabilities. One project that stands out is my work on a semantic search engine for podcasts `https://github.com/podcast-ai/pod-search`. This project utilized the FAISS vector storage library to index podcast episodes. We used Elastic Search for the first iteration to better handle storing and indexing the podcast data. However, we found FAISS

to be a better alternative and we used CLIP, a state-of-the-art language model, to perform the semantic search. By leveraging these technologies, we were able to build a powerful search engine that was capable of understanding the context of spoken words and matching them with user queries. The Backend was written in Flask. The project was a great success and received positive feedback from users. With my experience in NLP and search technologies, I believe I am well-equipped to contribute to the proposed project for this year's Google Summer of Code.

References

[1] Clip - learning transferable visual models from natural language supervision.

[2] Embedding generation: all datasets v4 minilm l6.

[3] Langchain - https://github.com/hwchase17/langchain.

[4] vector databases explained - https://zilliz.com/learn/what-is-vector-database.