AI Based Lecture Summarization Tool

Team 9

IIIT Hyderabad

April 27, 2020

Overview

Problem Statement

Despite the many advantages of online learning, it faces some severe disadvantages such as:

- Lack of accessibility due to internet connectivity issues
- Low attention span of students as compared to physical learning in classroom settings
- Uploaded videos are difficult to revise as compared to class notes

Problem Statement

As part of our end semester project in Software Engineering, we aim to address a few of the above problems by using AI to automatically summarize lecture videos. Our solution would leverage the latest advancements in machine learning and computer vision to make it very easy for users to revise using the uploaded videos.

Our Solution

We have developed our software using a client-server architecture and the user interacts with our product using a web application.

- The client side provide the user with a UI where they are able to login, and add, search and summarize the videos
- The server deal with all the client side requests such as:
  - Provide authentication methods for login
  - Summarize a given uploaded video
  - Return the most relevant clips for a given search query
- Our solution uses a combination of video processing, machine learning and summarization techniques to solve the problem in a fast and efficient manner.

High Level Design

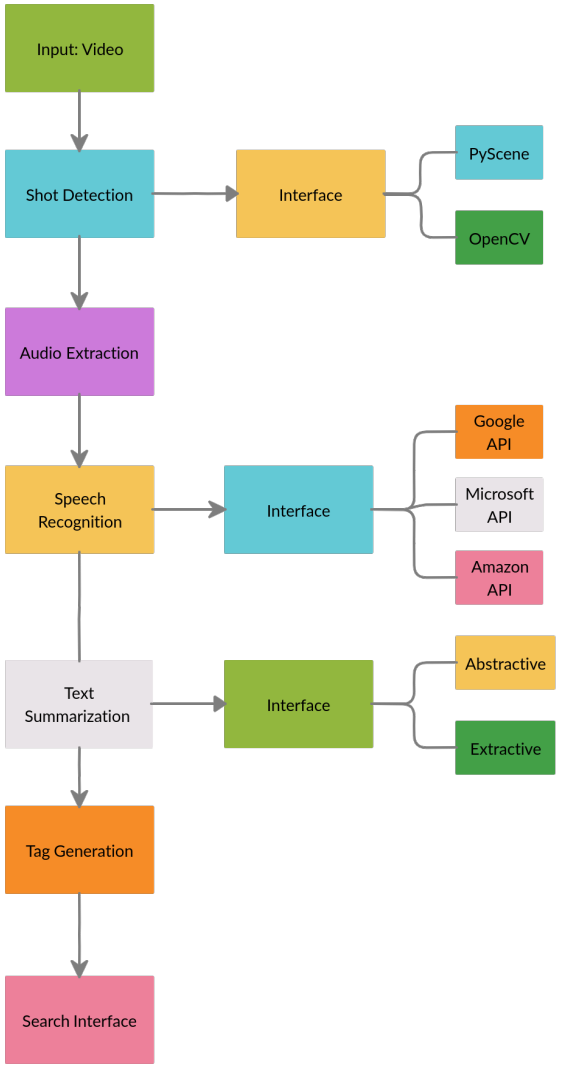

Figure: High Level Design of our algorithm

Coding Principles and Design Patterns

- We have followed Model View Template **Design Patterns**. The Model is the logical data structure behind the entire application and is represented by a database. The View is the user interface and template consists of static parts of the HTML output as well as describes how dynamic content will be inserted.
- Follows object oriented paradigm, which is a sub category of imperative programming paradigm.
- Interfaces and objects are created for each of the sub modules.
- Functions names are self-explanatory and small in size (7-10) lines.
- Exceptions are handled by using try except blocks.
- The code of different sub modules is easily **extendable** to other APIs and libraries.
- Followed **SOLID** design principles.

Software Development Model

- Agile SDLC Model
- Feature Driven Development (FDD) Agile methodologies
- Great solution to maintain control for incremental Agile project management

Integration and Testing Process

- Unit Testing
  - Each module has its test files in its respective folder.
- After Unit Testing we have performed **Incremental Integration Testing**.
  - Integrate the modules one by one using stubs or drivers to uncover the defects.
  - We have used **Functional** incremental testing methodologies as integration and testing takes place on the basis of the functionalities as per the requirement specifications in design document.
- We have used this because the defects are found early in a smaller assembly when it is relatively easy to detect the root cause of the same.