Unsupervised Multi-Modal Video Summarization Documentation

Papagrigoriou Vasileios Savvas
Auth

Overview

Unsupervised Multi-Modal Video Summarization is a cutting-edge project at the intersection of computer vision and machine learning. Its primary aim is to automate the process of condensing lengthy videos into succinct summaries without the need for pre-labeled training data, leveraging advancements in object detection and unsupervised learning algorithms.

Objectives

- Provide an efficient means of summarizing long videos to facilitate quicker content consumption and better information retention.
- Utilize state-of-the-art object detection techniques to enhance the understanding of video content and improve the accuracy of summaries.
- Employ unsupervised learning methods to enable automatic summarization that adapts to diverse video types and content without manual intervention.
- Ensure the summarization process is scalable and performant, capable of handling large datasets and high-resolution videos.

Feature Extraction and Integration

Visual Features

Feature Extraction: **Tool Used:** VGG16 is used to predict the features of the frames.
Process: Analyzes video frames to identify and categorize various objects.
Tokenization: RoBERTa used to tokenize the objects **Output:** A vector of features for each frame.

Histogram: **Tool Used:** OpenCV is used to create the histogram.
Process: Calculate the histogram and normalize
Output: A vector of features for each frame.

Optical Flow: **Tool Used:** OpenCV is used to calculate the optical flow.
Process: From the previous and next frame calculate the optical flow based on the motion and the angle of the substraction.
Output: A vector of features for each frame.

Integration of Visual Features:

- Integration: All the extracted features (visual features, histogram, optical flow) are concatenated.
- Dimensionality Reduction: The concatenated feature vector is then fed into an AutoEncoder model to reduce its dimensionality to 1024.
- Integration: The concatenated and reduced feature vector is used for further analysis and processing.

Audio Features

Feature Extraction: **Tool Used:** Librosa is used to extract the audio features.
Process: Extract the audio features using the MFCC algorithm.
Output: A vector of features for each frame.

Integration of Audio Features:

- Integration: All the extracted features are concatenated.
- Dimensionality Upscale: The feature vector is then fed into an Neural Network model to upscale its dimensionality from 128 to 1024.
- Integration: The feature vector is used for further analysis and processing.

Object Detection

Tool Used: YoloV5, renowned for its accuracy and speed in real-time object detection.
Process: Analyzes video frames to identify and categorize various objects.
Tokenization: RoBERTa used to tokenize the objects **Output:** A comprehensive list of detected objects, including their types and coordinates within each frame.

Title

Tool Used: RoBERTa is used to tokenize the title.
Process: Tokenize the title and create a vector of features.
Output: A vector of (1024,) for the title.

A. Video Summarization

Tool Used: knapsack is used to create the trailer.
Process: Calculate the knapsack based on the importance of each frame.
Output: A trailer of 15 seconds.

I. GitHub Repository Structure

The GitHub repository contains several directories and files, each with a specific purpose in the Unsupervised Multi-Modal Video Summarization project:

AutoEncoder

- `autoEncoder.py`: Trains an autoencoder model for reducing the dimensionality of feature vectors extracted from the video.

DataExtraction

- `audio.py`: Extracts audio features from the video for analysis.
- `visual.py`: Processes video frames for visual feature extraction, utilizing color features and optical flow.
- `objects.py`: Detects and encodes objects within video frames.
- `title.py`: Extracts and processes title features from the video's metadata.
- `getData.py`: Retrieves data for the other extraction modules.
- `frames.py`: Manages individual video frames for processing.
- `save.py`: Saves processed data for further use in the pipeline.

Evaluation

- `fscoreEval.py`: Calculates F-scores to evaluate the quality of video summaries.

knapsack

- `knapsack.py`: Implements the 0/1 knapsack algorithm to select the most important video segments for the summary.

vgg16

- `vgg16_weights.h5`: Holds the pre-trained weights for the VGG16 model used for image recognition tasks.

yolo

- `coco.names`: Contains the names of the objects that the YOLOv5 model can detect.
- `objectDetection.py`: Implements the YOLOv5 object detection algorithm to identify objects within video frames.

datasets

- `extractedAudio`: Contains the audio features extracted from the video.
- `summary_videos`: Contains the summarized videos.
- `ydata-tvsum50-v1_1`: Contains the TVSum50 dataset.

datasets/ydata-tvsum50-v1_1

- `data`: Contains the dataset's annotations and information.
- `video.zip`: Contains the videos in the dataset **(Extract the folder)**.
- `ground_truth`: Contains the ground truth data for the dataset.

videoSum

- `clipImportance.py`: Determines the importance of each video clip for the summary.
- `importances.py`: Defines functions for calculating the importance scores for frames.
- `mapping.py`: Manages the correspondence between data representations and their importance scores.
- `videoCreator.py`: Assembles the summarized video from selected frames based on their importance.
- `deletePkl.py`: Deletes the 'frame' pkl file created during the summarization process.
- `__init__.py`: Initializes the video summarization pipeline.

Other Files

- `.gitignore`: Lists files for Git to ignore.
- `Presentation.pptx`: A presentation summarizing the project's objectives and findings.
- `README.md`: Provides an overview of the project and instructions for use.
- `results.json`: Contains the output data from the summarization process in JSON format.
- `requirements.txt`: Lists the required Python packages for the project.

A. Directory Tree

```
+-- AutoEncoder
|   `-- autoEncoder.py
+-- DataExtraction
|   |-- audio.py
|   |-- data.py
|   |-- frames.py
|   |-- getData.py
|   |-- objects.py
|   |-- save.py
|   |-- title.py
|   `-- visual.py
+-- Evaluation
|   `-- fscoreEval.py
+-- datasets
|   |-- extractedAudio
|   |   `-- extracted_audio.wav
|   |-- summary_videos
|   |-- ydata-tvsum50-v1_1
|   |   |-- data
|   |   |   |-- ydata-tvsum50-anno.tsv
|   |   |   `-- ydata-tvsum50-info.tsv
|   |   |-- video.zip
|   |   `-- ground_truth
|   |       `-- ydata-tvsum50.map
+-- yolo
|   |-- coco.names
|   `-- objectDetection.py
+-- knapsack
|   `-- knapsack.py
+-- vgg16
|   `-- vgg16_weights.h5
+-- videoSum
```

```
|   |-- __init__.py
|   |-- clipImportance.py
|   |-- deletePkl.py
|   |-- importances.py
|   |-- mapping.py
|   `-- videoCreator.py
+-- .gitignore
+-- Presentation.pptx
+-- README.md
+-- results.json
`-- requirements.txt
```