International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Generation and Presentation of Metadata for Movies using Computer Vision and Machine Learning

Shashank C[1], Veena Gadad[2]

[1]*Dept. of Computer Science and Engineering, R.V College of Engineering, Bangalore India*
[2]*Dept. of Computer Science and Engineering, R.V College of Engineering, Bangalore India*

Abstract - *Entertainment and media are a large part of our lives these days, we watch many of our favorite TV shows and Movies on streaming sites such as Netflix, Hulu, Amazon prime and Sling TV. User engagement and retention is very important on these platforms. To do this features like content recommendation systems and auto skip the shows intro and outro have been implemented. We propose another such feature that will improve the audience user experience with the platform. It is a method of generating and presentation of metadata for movies using computer vision and machine learning.*

Each movie or TV show have a wide range of characters and scenes in them, ranging from 10 to a 100 actors, and multiple shots. The feature allows the user to pause the movie or TV show to display the actors in the current shot, or search the entire movie for all occurrences of an actor. The principle behind this system is, shot detection and face detection based on computer vision.

Key Words: shot detection, face detection, computer vision, API's, TMdb, ANN, Histogram oriented gradients, REST API's, JSON, BSON, Database.

1. INTRODUCTION

Staying on top of your competition is important for any business, this is even more important in the fast moving present age, where platforms are being updated regularly to cater to users requirements. This feature aims to improve the user experience of a viewer by involving them more into the people involved in making the movie and other information based on the content of the movie. Moreover indexing of movies based on shots and actors in the movie may prove useful in other ways in the future. Metadata here means the extra data that stores information about the movie, such as the start and end timestamps and frame number for each shot along with the actors in that shot.

Since there exists a huge volume of movies already on these platforms, we require an automated way to generate these metadata and store them in a database. The client (viewer's browser or TV) can then make API calls to a server to obtain the metadata when a movie is being watched. To achieve this the entire process is divided into two separate stages, a preprocessing stage and playback stage. The preprocessing stage is performed only once for each movie, and its purpose is to cache the metadata in a database. This stage need not be completely automated and can involve some manual addition of data. The main reason to automate it is to generate metadata for already existing movies. New movies can have an option to add this metadata manually by the movie producers. The playback stage is what occurs when the user presses play on a certain movie. This paper is organized by the steps involved in the entire process. The next two sections will go into each of these steps in overview, and the rest of the paper goes into each step in detail.

1.1 Preprocessing stage

During the preprocessing stage (fig-1), the system acquires the cast of the movie from TMdb (The movie database), performs shot segmentation and matches the faces to the cast of the movie. TMdb contains a vast database of movies of various genre, cast, languages, and details of each cast members and so on. More importantly it also lets you get the images of the cast members, which is required for the face identification system to compare against. Each shot of the movie is put through the face matching process to identify all actors present in that shot, this obviously comes with many issues that we will see later. For any given movie any detected face needs to be only compared against the faces of the cast of that movie.

1.2 Playback stage

The playback stage (fig-2) involves the client querying the backend server when required, the information is delivered via a REST API. When the user clicks to view a movie, it automatically fetches the metadata from the backend server. The client application will have a video player with widgets to show the list of actors and other related data like the images of the actors.

2. METHODOLOGY

This section describes the entire process in more detail, with all the steps involved to go from the movie to the movie being played by a user on the platform.

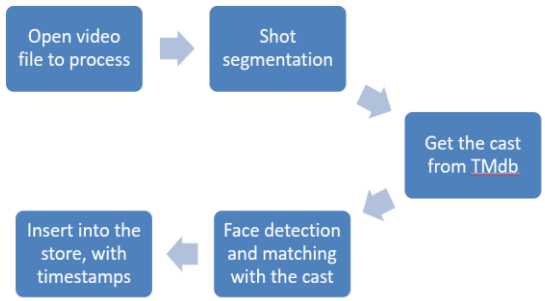

Fig -1: Preprocessing stage

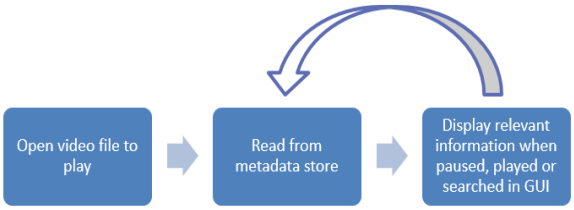

Fig -2: Playback stage

1.1 Shot Segmentation

A shot is an uninterrupted recording from a camera. Not to be confused with a scene, which is a series of shots that are part of the same logical act in the movie or TV show. Two shots are always separated by a cut or transition. A few common cuts seen in movies and shows are soft cuts and hard cuts. A Soft cut is a gradual transition from one shot to another whereas a hard cut is an abrupt transition from one shot to another.

Shot transition detection or cut detection involves finding the cuts from a continuous video (also called video segmentation, shot boundary detection, video parsing).

A general algorithm to pinpoint the timestamps of the cuts is what is required. From the review of current methodologies it is seen that soft cuts are generally harder to detect when compared to hard cuts as it involves the change taking place over a larger temporal space of around 10-20 frames, whereas hard cuts take maybe 2-3 frames to occur. Common issues in current methodologies are wrongly detecting a cut where the camera pans fast (true negative) and missing a soft cut because it was too slow (false positive) etc.

Since about 95 % of cuts in movies are hard cuts we use a simple method that uses pixel intensity histograms between successive frames with adaptive thresholds. This method is slightly more complex when compared to the absolute pixel to pixel difference method but performs a lot better owing to using histograms to take into consideration the color distribution over the entire frame and the fact that it uses adaptive thresholds. It is capable of detecting hard cuts effortlessly and detecting soft cuts with moderate certainty.

Advantages of using this method are that it is very fast and can be parallelized easily. It also performs very well compared to other methods which contain many parameters to tweak. It also detects mood shifts in the movie depending on the histogram shape.

1.2 Getting Cast from TMdb

TMdb is an online database of movies, TV shows, commonly referred to as HOGs, along with an artificial neural network (ANN).

Fig -3: Cast of Lord of the rings from TMdb

1.3 Face Detection and Identification

Face detection involves only saying if a face is present in an image or sequence of images, on the other hand face identification is identifying a face as a specific person. From the state of the art we see that there are many ways that faces are detected and identified. The method we chose was the Histogram oriented gradients, commonly referred to as HOGs, along with an artificial neural network (ANN).

An overview of the face identification pipeline involves Generating gradients for each 16 x 16 cell of pixels. The gradients by definition always point in the direction of the largest change in brightness. A Face landmark estimation is done using an ANN using 68 landmarks. An affine transformation is applied to get rid of the variations in position, scale, and orientation of the face. The face is encoded into a set of features having 128 measurements. After a classifier can be used to get the name of person from the encoding.

The advantage of using a HOG is that it detects spatial structures in images very well, and runs very quickly, as the aim is to pre-process a huge amount of media quickly, HOG fits the use case very well.

1.4 Caching in MongoDB

MongoDB is a NoSQL, document oriented database. It is perfect to store soft real-time data such as this. It also has very well documented drivers for multiple languages including python. The generated metadata is stored into a collection, to be queried when viewers watch the movie. The metadata for each movie needs to be stored separately to search based on the movie. The schema needs to allow for fast searching of all timestamps related to a particular movie, and have the timestamps sorted so the client doesn't have to.

Along with shot segmentation and face identification the preprocessing stage discussed in the previous sections the preprocessing stage also involves caching the generated metadata into the database. The python driver for MongoDB is used to do this. The generated data is converted into BSON formatted data and then inserted into a MongoDB collection. The schema of the collections would look like the schema shown in fig1.

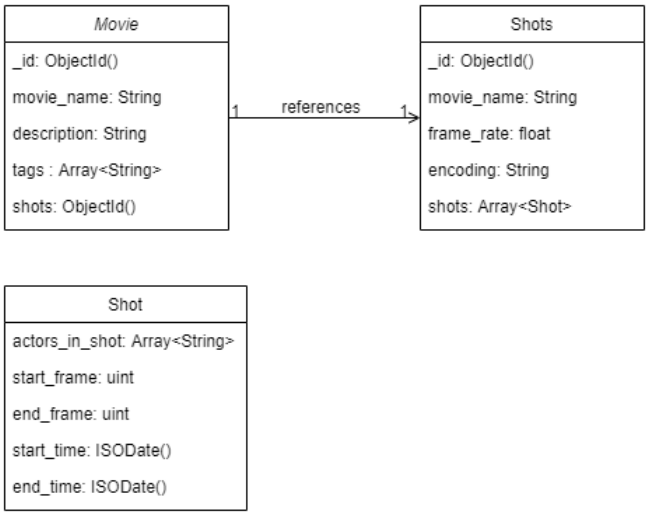

Fig -4: database schema for movies and shots

The reason to group them this way is because the viewer might decide to view the genre and description of the story before watching, and getting the entire shots data would not be required at this point. When the user clicks to watch the movie is when the shots data can be requested by the client. And the entire shots document comes in presorted, making the process efficient.

1.5 Building the REST API

REST API's forms the backbone of modern internet services, it is a software architectural style that allows access to resources over the network in a uniform and consistent way. It basically allows the client to get access to the metadata without having explicit connection to the server database, instead it goes through a server process that acts as an adapter to the database. Python provides a framework called Flask to handle these RESTful requests from the client and return JSON formatted data back to the client.

Pymongo is a MongoDB driver for python, as discussed previously. It is also used during the playback stage of the process in the server side. Flask is a python framework to handle http requests, and to create RESTful API's.

Any API has endpoints that need to be defined, this method has two endpoints, one for movies and another for shots for a particular movie. The movie endpoint serves the client for the general overview, plot description, genre of the movie. The shots endpoint provides the shots metadata when the movie begins playing.

1.6 Building the Frontend

The frontend is the final component of the entire system, it is a basic web application that makes requests to this API and plays the movie, with displaying the metadata in a presentable form. It consists of a video player that has all of the basic requirements of a video player like pause, resume, full screen, timeline etc. A widget shows the current actors on screen in the current shot. To implement these we have chosen JavaScript and an accompanying library Video.js.

JavaScript is a just in time compiled language that is mainly used in browsers, and is used to make webpages responsive, make API calls, read and write to the webpage during runtime etc.

Video.js is a simple JavaScript based web player built from the ground up for a HTML5 video, it runs on any modern browser and gives a simple interface to the playback variables.

The frontend makes API requests to the backend when the video needs to be played. And selects the appropriate metadata to display. The WebApp is made using HTML, CSS and JavaScript to make requests to the backend through API calls.

Just before the movie is loaded to begin playing, the API call is made to fetch all the shot and actor metadata. When playing, the current shots are found by the timestamps of the current play head and the relevant HTML is modified to show the images and names of the actors in the shot.

Fig -5: Frontend showing the video player and widget to show actors in current shot.

3. CONCLUSION

This paper discusses a method to improve user viewership and retention. It increases the sense of community among the viewers, bridging the gap between movie producers and viewers. It also provides a base for future work with this metadata, like searching for all the shots a particular actor appears, or all shots that a set of actors are present in the same shot.

REFERENCES

[1] Zeeshan Rasheed and Mubarak Shah 'Scene Detection In Hollywood Movies and TV Shows' School of Electrical Engineering and Computer Science University of Central Florida Orlando, Fl 32816.

[2] Muhammad Haroon, Junaid Baber, Ihsan Ullah, Sher Muhammad Daudpota, Maheen Bakhtyar, and Varsha Devi, 'Video Scene Detection Using Compact Bag of Visual Word Models' Department of Computer Science & IT, University of Balochistan, Pakistan, Department of Computer Science, Sukkur IBA University, Pakistan, Department of Computer Science, Sardar Bahadur Khan Women's University, Pakistan.

[3] Michael Gygli, 'Ridiculously Fast Shot Boundary Detection with Fully Convolutional Neural Networks' Zurich, Switzerland.

[4] Rainer Lienhart, 'Comparison of Automatic Shot Boundary Detection Algorithms' Microcomputer Research Labs, Intel Corporation, Santa Clara, CA 95052-8819.

[5] Ramin Zabih, Justin Miller, Kevin Mai, 'A Feature Based Algorithm for Detecting and Classifying Scene Breaks' Computer Science Department Cornell University Ithaca NY.

[6] Jihua WANG, Tat-Seng CHUA and Liping CHEN, 'CINEMATIC-BASED MODEL FOR SCENE BOUNDARY DETECTION' National University of Singapore, Singapore 117543.

[7] S. T. Gandhe, K. T. Talele, and A.G.Keskar. 'Face Recognition Using Contour Matching', IAENG International Journal of Computer Science, 35:2, IJCS_35_2_06.

[8] Vahid Kazemi and Josephine Sullivan, 'One Millisecond Face Alignment with an Ensemble of Regression Trees' KTH, Royal Institute of Technology Computer Vision and Active Perception Lab Teknikringen 14, Stockholm, Sweden.

[9] Tat-Seng CHUA, Mohan Kankanhalli and Yi Lin, 'A GENERAL FRAMEWORK FOR VIDEO SEGMENTATION BASED ON TEMPORAL MULTI-RESOLUTION ANALYSIS' School of Computing, National University of Singapore.

[10] Histograms of Oriented Gradients for Human Detection Navneet Dalaland Bill Triggs INRIARh one-Alps, 655 avenuedel' Europe, Montbonnot 38334, France {Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr