Globo Face Stream: A System for Video Meta-data Generation in an Entertainment Industry Setting

Rafael Pena[1], Felipe A. Ferreira[1], Frederico Caroli[2], Luiz Schirmer[1] and Hélio Lopes[1]
[1]*Pontifícia Universidade Católica do Rio de Janeiro, Brazil*
[2]*Globo.com, Rio de Janeiro, Brazil*

Keywords: Computer Vision, Face Recognition, Video Meta-data.

Abstract: The amount of recorded video in the world is increasing a lot due not only to the humans interests and habits regarding this kind of media, but also the diversity of devices used to create them. However, there is a lack of information about video content because generating video meta-data is complex. It requires too much time to be performed by humans, and from the technology perspective, it is not easy to overcome obstacles regarding the huge amount and diversity of video frames. The manuscript proposes an automated face recognition system to detect soap opera characters within videos, called Globo Face Stream. It was developed to recognize characters, in order to increase video meta-data. It combines standard computer vision techniques to improved accuracy by processing existing models output data in a complementary manner. The model performed accurately using a real life dataset from a large media company.

1 INTRODUCTION

A key goal of the entertainment and media industry is to understand users preferences in order to get content production insights and provide relevant content. To this end, more information about the content potentially generates more information about the users. This work will focus on video content.

Many computer vision technologies emerged in recent years and good results on specific tasks have been achieved. Face recognition (Schroff et al., 2015a), scene detection (Castellano, 2012), optical flow (Kroeger et al., 2016) and pose estimation (Cao et al., 2018) are examples of existing technologies created to better understand humans appearance within videos.

The ultimate goal of this work is to create a high accuracy model to detect soap opera characters within videos in order to enrich video meta-data. On this case study an archive from Grupo Globo, the largest media group in Latin America entertainment industry, and the leader in the production and broadcasting soap operas in Brazil.

Our main contribution is the proposal of an unified system for face verification (is this the same person), recognition (who is this person), re-identification and clustering (find common people among these faces) (Schroff et al., 2015a). It combines different methodologies to improve soap opera characters recognition.

In Brazil, tens of millions of people watch soap operas almost every day. Broadcasting tv has popularized this type of content but part of its consumption is through cable TV channels, and in the last decade online consumption, whether live or on demand, has grown sharply.

The Globo Group occupies a prominent position in this scenario, being the audience leader for this type of content on broadcasting tv, cable tv or via the Internet. Its production started in 1965, surpasses 300 soap operas and tens of thousands of chapters. In recent years, online channels are used by approximately 2 million users every day.

This work has been applied to the context of soap operas but is a generic solution that can be used in various other types of content such as series, movies, auditorium programs, humor shows and reality shows. The existence of a content ontology that was used in an integrated way to the developed solution contributed positively to the work results, both in the solution assertiveness and the process optimization. This ontology that describes all kinds of content produced by the company was very important to an easier generalization of the solution.

The task that this paper is proposing to solve has several difficulties, as for example: the calculation of image similarity requires a high computational cost; entity match with a proprietary database is hard when using online face recognition services; and the accuracy on face recognition varies and depends on video frame aspects to get a good result (for instance, the

Pena, R., Ferreira, F., Caroli, F., Schirmer, L. and Lopes, H.
Globo Face Stream: A System for Video Meta-data Generation in an Entertainment Industry Setting.
DOI: 10.5220/0009380203500358
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 1*, pages 350-358
ISBN: 978-989-758-423-7

character face angle).

From these difficulties raises many challenging issues to create an efficient automated character detection system. Identifying the correct character in a scene, from the very first appearance until the last frame is not a trivial task. In an attempt to identify characters presence in scenes it was necessary to test a hybrid solution based on complementary, specific techniques.

Here, the advantages, disadvantages and results achieved with a developed system based on existing and complementary methods will be presented.

This paper is organized as follow. Section 2 describes some previous and related work. Section 3 discusses the proposed method. Section 4 shows the results. Section 5 discusses the results. Finally, Section 6 concludes the work pointing some future directions.

2 RELATED WORK

Face recognition is a challenging task that has been attacked for more than 2 decades. Classical approaches for videos consider video sequences where each frame are analyzed individually. More accurate approaches use the temporal cues in addition to the appearance of the faces in videos (Zhou et al., 2018; Wang et al., 2008; Zhou et al., 2004).

The image-based face recognition system normally involves task that includes face detection, face alignment, and face matching between a detected face in a photo or video and a dataset of *N* faces (Zhou et al., 2018). However considering the real world situations, the illumination condition, facial expression, and occlusion represents challenging problems.

Google (Schroff et al., 2015a) presented a system, called FaceNet, that map face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. These feature vectors can be used in tasks such as face recognition, verification and clustering combined with classical machine learning techniques. Their method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer in other architectures.

To deal with real video situations, Huang et al. (Huang et al., 2015) propose a Hybrid Euclidean-and-Riemannian Metric Learning method to fuse multiple statistics of image set. They represent each image set simultaneously by mean, covariance matrix and Gaussian distribution. To test they approach, they use use a public large-scale video face datasets: YouTube Celebrities, containing 1910 video clips of 47 subjects collected from the web site. Their results are impressive, although it face some problems considering our context, as for examples, it do not perform person re-identification. There is no association between images frames of the same person taken from different cameras or from the same camera in different occasions.

Masi et al. propose a (Masi et al., 2018) method that is designed to explicitly tackle pose variations. Their Pose-Aware Models (PAM) process a face image using several pose-specific and deep convolutional neural networks (CNN). Also, in their application, a 3D rendering is used to synthesize multiple face poses from input images to both train these models and to provide additional robustness to pose variations.

Zhou et al. (Zhou et al., 2018) address the problem of recognition and tracking of multiple faces in videos involving pose variation and occlusion. They introduce constraints of inter-frame temporal smoothness and coherence on multiple faces in videos, and model the tasks of multiple face recognition and multiple face tracking jointly in an optimization framework. Also, they focus in practical problems involving surveillance cameras.

Ranajam et al. (Ranjan et al., 2017) propose an algorithm for simultaneous face detection, landmarks localization, pose estimation and gender recognition using deep convolutional neural networks (CNN). Their proposal consider the intermediate layers of a CNN using a separate CNN followed by a multi-task learning algorithm that operates on the fused features. Their method can detect face, localize land-marks (re related to the face points), estimate the pose, which is the order of roll, pitch and yaw, and recognize the gender.They use specialized CNN layers to get each output prediction. Besides the outstanding results, this approach still seems to be very computational intensive.

Gou et al. (Gou et al., 2017) use the collaborative representation method to get the outputs of different sizes of sub image sets, and obtain the final result by optimally combining these outputs

While outstanding results by these methods presents advances to tackle different problems for face recognition, they still present some limitations for real world problems and industry applications.

A framework that combine the robustness of each proposal can be used to tackle the optimization problems as well illumination condition and facial expression problems. In the next section, we will present the main aspects of our system.

3 METHOD

The approach of this work is to combine different methodologies to improve people recognition tasks using inferences based on established models.

Figure 1 presents a high level schema that explains how the system works. From the input video file to the final frame set containing actor annotations frame by frame. The first step is to split the video into multiple clips, each one corresponds to a different scene. PySceneDetect is a command-line application and a Python library for detecting scene changes in videos, and automatically splitting the video into separate clips (Castellano, 2012). This application was used to eliminate errors observed during the propagation of actors annotations through the frames. Another advantage is that it enables parallel processing. A ten minutes video, that represents one of the four episode blocks, has on average a hundred scenes. Once the separate clips are available, all the further steps are taken for all different scenes, simultaneously. The next step is to identify, for each different actor that appears in the scene, the actor face stream. The face stream is the entire set of frames where a specific actor appears during a scene. It is obtained by a module developed during this work, named FaceTracker and detailed in section 3.2. An existing system was used to overcome facial recognition challenges. FaceNet uses a deep convolutional network (Schroff et al., 2015b) and this model will be discussed in section 3.1. All the outputs from these different technologies are processed by a logical model that combines information and infers the actors for the entire frame set. This logical model will be detailed in section 3.3. In order to scale in real life situations the system was integrated with two existing components, the computer vision architecture detailed in section 3.4 and the content ontology discussed in section 3.5.

3.1 Facenet

As described in (Schroff et al., 2015b), FaceNet is an end-to-end learning technique that uses deep neural network to learn an embedding $f(x)$ from an image x in a feature space $\mathbb{R}^d$, such that the square of the distance between all faces, regardless the conditions of the image such as illumination and pose variation, of the same identity is small, whereas the square of the distances between pairs of faces of different identities images is large. To this end, a triple loss is implemented to ensure that a specific person's x_i^a *(anchor)* image is close to all x_i^p *(positive)* images of a single person is for any other image x_i^n *(negative)* of any other person. Thus, FaceNet minimizes the following loss function L for a set of N faces where α is a margin that is enforced between positive and negative pairs:

$$L = \sum_i^N [\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha]_+ \tag{1}$$

As the resulting output, it generates a compact d-dimensional Euclidean space as feature vectors where distances directly corresponds to a measure of face similarity that can be used in tasks like face recognition, verification and clustering.

In this work, a pre-trained model that outputs 128-dimensional vectors was used for the task of recognize actors face. Another important aspect from this system that fits perfectly to the use case reported in this paper is that the system depends on an initial set of faces, and from this set, for each frame processed, the system will calculate the similarity between faces from the frame and each face from the initial set. In this work, the initial set of faces is the soap opera actors faces. It is possible to assume that for each face detected in a frame, it will be one of the actors informed in the initial set (the cast). In order to obtain the cast actors faces images, a semantic query endpoint was used to query the company's content ontology. The only necessary filter is the name or the identification code of the content. Once this information is known, it is possible to use that endpoint to find out the actors that participates in a soap opera. Section 3.5 describes the existing ontology model and how it allows generic queries for various content types.

Once a face is detected in a frame, before use FaceNet in order to generate the face embeddings, it is necessary to align the face. The alignment preprocess faces for input into a neural network. Faces are resized to the same size (such as 96x96) and transformed to make landmarks (such as the eyes and nose) appear at the same location on every image(Amos et al., 2016). Figure 2 details the embedding generation pipeline before face recognition step.

After all face embeddings have been generated, each of them is compared against all embeddings of all labeled faces dataset in order to recognize who is the actor that appear in the frame. Basically, the actor's facial recognition task is accomplished through simple computation of the Euclidean distance between the embedding of the frame and the embeddings of the dataset. If the distance is smaller than a certain *Threshold*, the face of the actor is recognized as being the one corresponding to the labeled embedding face from the dataset. In this work was used a *Threshold=0.6*. Figure 3 details how this step works.

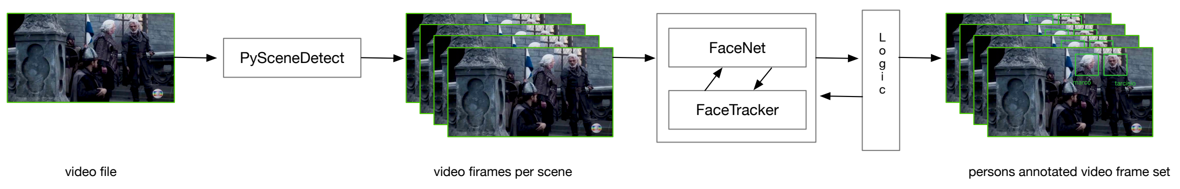

Figure 1: High level schema that explains the Globo Face Stream system.

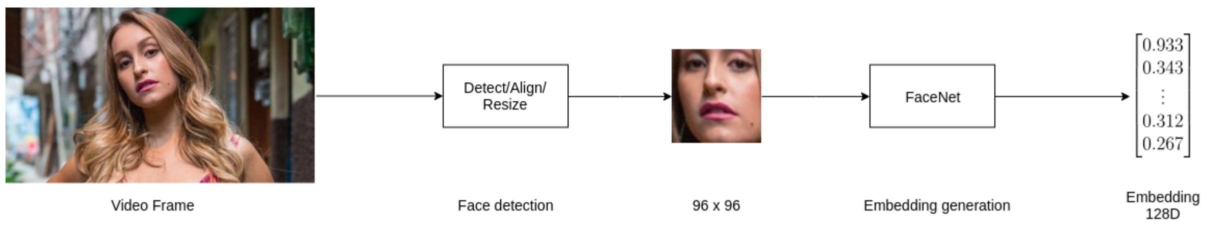

Figure 2: Example of 128-dimenisonal embedding generation for each face found in a frame.

3.2 Facetracker

This module was created to track actor faces through scene frames. During the tracking process, the module interacts with the face recognition system and uses a face detector and an object tracker (*get-frontal-face-detector* and *correlation-tracker* from the *dlib Python* library).

This face detector is made using the now classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme. This type of object detector is fairly general and capable of detecting many types of semi-rigid objects in addition to human faces(King, 2009).

To track a face among the frames the dlib Python library provides a specific object. This object lets you track the position of an object as it moves from frame to frame in a video sequence(King, 2009). Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge(King, 2009).

The structural SVM based training algorithm behind the easy to use object detector provided by Dlib is called Max-Margin Object Detection (MMOD). This method does not perform any sub-sampling, but instead optimizes over all sub-windows. MMOD can be used to improve any object detection method which is linear in the learned parameters, such as HOG or bag-of-visual-word models (King, 2015).

The system developed during this research attempts to create a face stream for each different character appearance within the scene. After the face detection, the face tracker step was implemented using dlib's implementation of the correlation tracking algorithm. To use it, you give the correlation-tracker the bounding box of the face you want to track in the current video frame. Then it will identify the location of the object in subsequent frames. The dlib correlation tracker implementation is based on paper (Danelljan et al., 2014), Accurate Scale Estimation for Robust Visual Tracking.

Our baseline is closely related to the MOSSE tracker (Bolme et al., 2010). The tracker learns a discriminative correlation filter used to localize the target in a new frame (Danelljan et al., 2014).

The face tracking process is started every time a face is detected by the *get-frontal-face-detector*. This function is very efficient to detect a face from the frontal view. But sometimes a character first appearance in a scene is a side view. Those cases didn't find perfect results because the tracker only starts to track after the frontal detection, genarating a lack of meta-data for all the frames that preceeded the first frame that has a frontal face detection.

Figure 4 shows that one of the actors wasn't tracked since his first appearance because *get-frontal-face-detector* couldn't detect his face in the first frame. On the second frame the same actor face was detected and after that the actor will be tracked even in frames with a side view.

To avoid this problem we created a two-step tracking process. The forward pass tracks the actor in the frames after the frame in which the frontal face detection occurred. The order is from the first frame to the last. Analogously, the backward pass tracks the actor in the frames before the frame in which the frontal face detection happened and the process starts from the last frame. At this phase the goal is to link the ac-

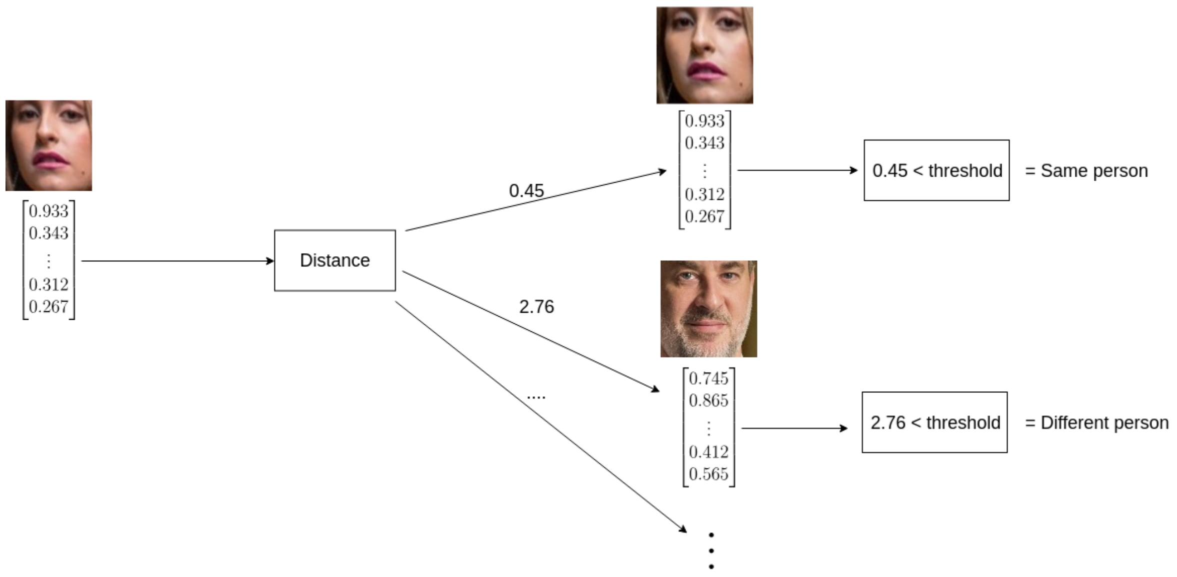

Figure 3: Example of face recognition step. Based on a threshold, each face embedding found in frame is compared against all labeled face embeddings dataset in order to recognize the actor face.

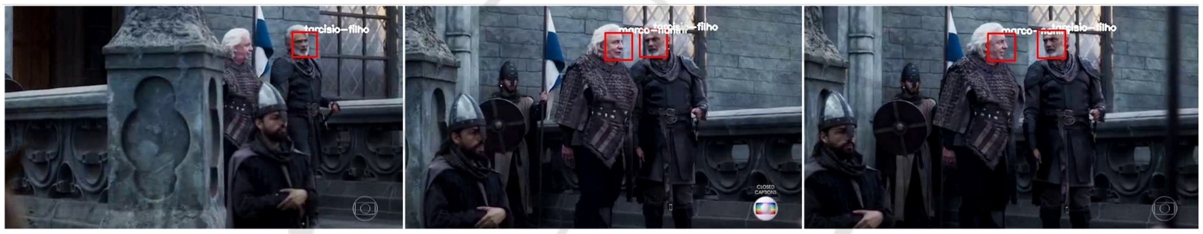

Figure 4: Example images from the TV Globo dataset. The red boxes show the face tracking after frontal face detection.

tor face positions among the scene frames. The output will be as many actor streams as the number of different actors appearances that occurred in the scene.

During the tracking process, every time a face is detected, FaceNet is used in order to identify that face. Different results for the same actor in different frames were observed. To eliminate that noise, the entire actor stream is submitted to a logical model that decides who is the actor frame by frame. The actor stream is created by merging forward and backward passes outputs.

3.3 The Logical Model

The logical model optimizes the frames annotations. It eliminates annotations errors and also annotate frames with no frontal face detection.

Figures 5 shows an example of a frame sequence containing both annotations issues. Either non annotated or wrong annotated frames can be observed.

Once the actor stream is created, the logic model attempts to assign the same annotation to all actor stream's frames. The model processes each different actor stream separately. For each frame containing a FaceNet annotation, there is a distance metric calculated by FaceNet. The logic model uses the smallest distance actor suggested by FaceNet, all the suggestions from different frames are considered.

Both figures 5 and 6 shows the same frame sequence. The first one shows the initial results using *get-frontal-face-detector* from the *dlib Python* library and *FaceNet*, the second one shows results using the entire proposed framework that includes *correlation-tracker*, also from the *dlib Python* library and the Logical Model implemented during this research.

3.4 Computer Vision Architecture

In Globo.com, there is an in-house computer vision architecture to support different applications. This architecture provides an asynchronous process to integrate different components from the company's video platform.

Figure 7 shows a high level architecture to explain how the proposed solution is integrated to the existing systems. On the top of the figure there is a com-

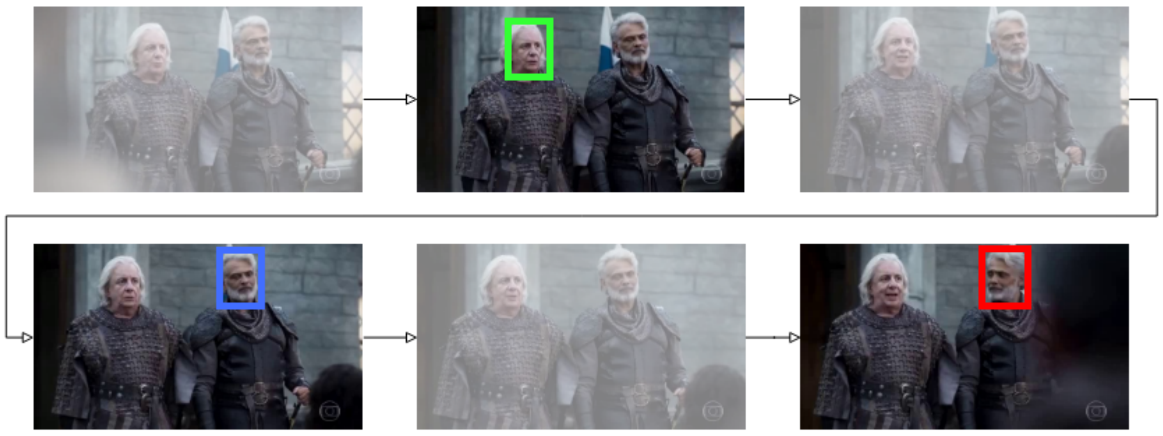
Figure 5: Frames sequence inconsistently annotated.

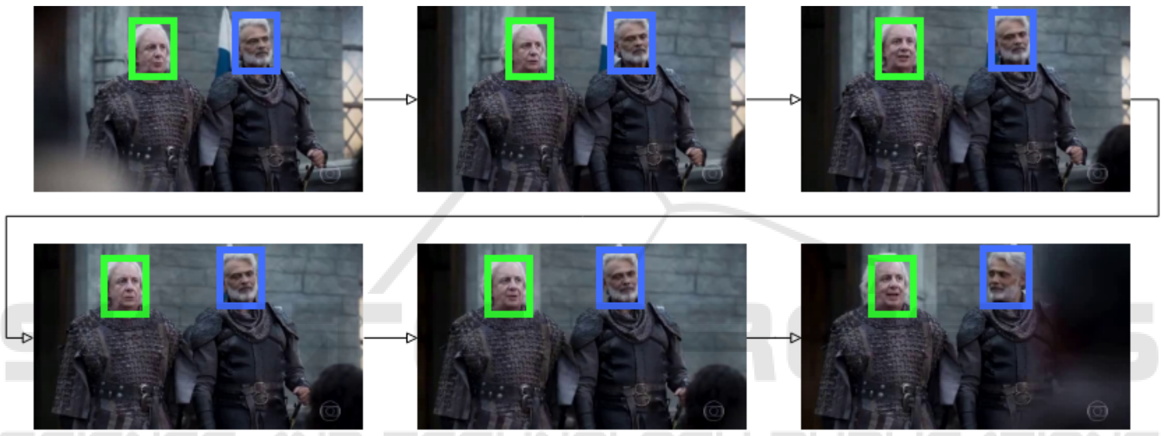
Figure 6: Frames sequence consistently annotated.

ponent called *Scheduler* that is connected to the *Webmedia API* using the *recents* service that informs every new media published on the video platform. The *Scheduler* identifies if the media must be processed by a computer vision job and enqueues the media to be processed as soon as there is available computer resources to.

3.5 The Content Ontology

(Guizzardi and Wagner, 2010) emphasizes the difference between the meanings of the term ontology in computing. On the one hand, by the Conceptual Modeling community, the term has been used according to its definition in Philosophy: A domain-independent and philosophically well-grounded formal categorization system, which can be used to spell out specific models of reality. domain. On the other hand, by the Artificial Intelligence, Software Engineering, and Semantic Web communities, the term is used as a concrete engineering artifact designed for a specific purpose without paying much attention to grounding issues.

Globo.com has previously invested in the creation of an content ontology that describes all kinds of content produced or offered by the company. During the last 10 years this ontology has been evolved by many journalists and content producers with the focus on articles categorization to improve content search, organization of content offerings in digital products and online navigation. Nowadays this ontology has hundreds of concepts and hundreds of thousands instances that helps to describe content types, events, roles, locations, persons and objects.

The Unified Foundational Ontology (UFO) has been used to obtain a high-level abstraction that helps on generalize and organize concepts from different domains. From the technical aspect, the implementation of the ontology using Resource Description Framework (RDF - https://www.w3.org/RDF/), a W3C pattern, enabled the implementation of a semantic query endpoint that integrates concepts from the company vocabulary and other public vocabularies, like DBpedia, to enrich concepts.

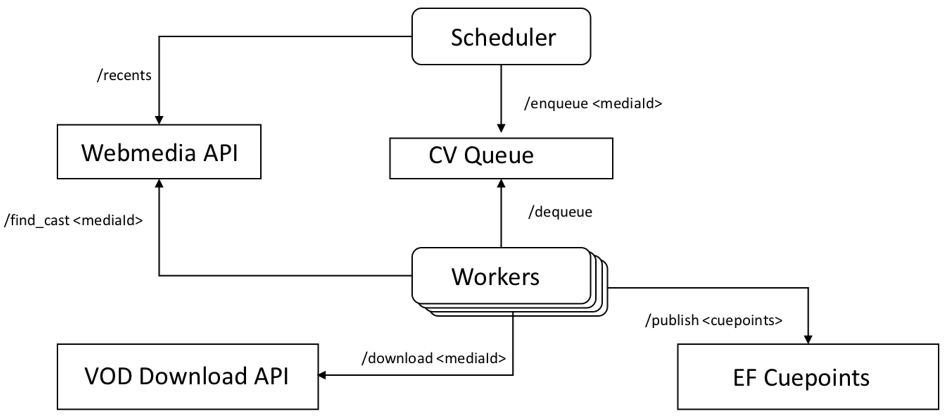

Figure 7: In-house architecture for computer vision applications.

Using this controlled vocabulary described by the company ontology is possible to find out all cast and crew from any creative work, just like finding any celebrity from sports, entertainment and journalism TV shows.

4 RESULTS

Exact matches is the main metric that was used to evaluate the system. It corresponds to the percentage of frames for which the system correctly indicates all the actors that appear. Regarding the evaluation task, a ground truth dataset was created using a customized tool developed to help humans to annotate actors appearances in video frames. The dataset contains 7000 frames, randomly selected from 20 different scenes (3 different soap operas). The evaluated accuracy was 92%. Two different approaches was tested in order to evaluate the proposed method: the approach 1 was based on the FaceNet recognition without any additional logic. The approach 2 was based on the FaceNet recognition together with the logic model developed during this research and detailed in section 3.3.

Table 1 shows the enormous advantage of approach 2 over approach 1 when the goal is to maximize exact matches.

From the application performance perspective, the implementation used during this survey achieved results that enable the real life usage in the company. A parallel scene processing approach was proposed to reduce the total time spent to annotate soap opera episodes.

Table 2: time metrics from the entire annotation process. Five different scenes selected from the ground truth dataset.

On average, the time spent to annotate is 5.5 times the scene duration. For instance, a 1 minute long scene will take 5 minute and 30 seconds to have all the frames annotated. Considering that each episode of a soap opera is 40 minute, divided into 4 blocks of 10 minute, the total time to annotate the entire block is 5 minutes and 30 seconds. That's because in general a 10 minute block has over 100 different scenes, and each scene is less than a minute. So, its feasible to process all scenes in parallel and spend around 5 minutes to process a 10 minutes block. This way, if there is enough computer resource in order to process all the scenes, from all the blocks in parallel, the total time is directly related to the longest scene duration. The computer used to process the videos was an Intel Xeon 2.7GHZ CPU with 128 GB RAM memory and a GPU computing processor NVIDIA Tesla P40 with 22 GB memory.

5 DISCUSSION

After the evaluation task it was clear that the method generated significant results regarding the amount of video frames, correctly, annotated by the model. One of the proposed method's benefit is that it enables automatic actors and actresses annotation even on frames that does not have a frontal face detection. Also, the model proved to be very efficient for disambiguate different annotations regarding the same person. The integration with the content ontology query endpoint facilitated the automatic photos retrieval from different cast. These photos were used as a reference to identify the actors in the video frames.

Table 1: Exact matches comparison between two survey's approaches.

Scene	# frames	approach 1	approach 2
Scene 1	730	300	698
Scene 2	810	507	810
Scene 3	1200	780	1180

Table 2: Total time to annotate scenes.

	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8
Duration	30	27	32	61	54
# Frames	925	810	985	1830	1628
Time loading videos	0.8610	0.8100	0.9910	1.6390	1.6090
Time loading models	10.6060	9.9700	9.8650	10.2210	10.4710
Time Forward Pass	122.4060	71.9740	71.6120	123.0910	170.2540
Time backward pass	63.8630	99.7530	88.4320	94.6830	167.3830
Time Merging passes	0.0072	0.0083	0.0028	0.0037	0.0157
Time Deciding Names	0.0166	0.0161	0.0042	0.0070	0.0108
Total Time	197.7598	182.5314	170.9070	229.6447	349.7475

6 CONCLUSIONS

When it comes to video metadata generation, the proposed method achieved important goals. The use of a content ontology was essential in automating actor photo capture. In addition, it has made it possible to use the tool in a generic and extensible way for various types of content besides soap operas, for example, TV series, movies, auditorium shows, humor shows and reality shows.The proposed method improved the original FaceNet accuracy because it iterates the video frames in order to eliminates annotations errors and annotate frames with no frontal face detection. The accuracy obtained with the new method is much higher when the observed metric is exact matches because it analyses the sequence of video frames that characterizes an appearance of a character in a given video part instead of analyse a single frame observation. A huge archive of soap opera videos will be enriched with metadata enabling different applications improvements. New features for recommender systems and users preferences information to help advertising targeting are the first applications that will benefit from this metadata. Recently, the company developed a new pipeline for computer vision that supports different kinds of application. Nowadays, a model designed to detect intro and credits parts from TV series episodes is using this pipeline in production. The same pipeline will be used to integrate the Facestream solution to the company's video processing platform.

REFERENCES

Amos, B., Ludwiczuk, B., and Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science.

Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2018). Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*.

Castellano, B. (2012). Pyscenedetect. *Journal Title*, 13(52):123–456.

Danelljan, M., Häger, G., Khan, F. S., and Felsberg, M. (2014). Accurate scale estimation for robust visual tracking. In *BMVC*.

Gou, G., Li, Z., Xiong, G., Guan, Y., and Shi, J. (2017). Video face recognition through multi-scale and optimization of margin distributions. *Procedia Computer Science*, 108:2458–2462.

Guizzardi, G. and Wagner, G. (2010). Using the unified foundational ontology(ufo) as a foundation for general conceptual modeling languages.

Huang, Z., Wang, R., Shan, S., and Chen, X. (2015). Face recognition on large-scale video in the wild with hybrid euclidean-and-riemannian metric learning. *Pattern Recognition*, 48(10):3113–3124.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

King, D. E. (2015). Max-Margin Object Detection. *arXiv e-prints*, page arXiv:1502.00046.

Kroeger, T., Timofte, R., Dai, D., and Van Gool, L. (2016). Fast optical flow using dense inverse search. In *Euro-*

pean Conference on Computer Vision, pages 471–488. Springer.

Masi, I., Chang, F.-J., Choi, J., Harel, S., Kim, J., Kim, K., Leksut, J., Rawls, S., Wu, Y., Hassner, T., et al. (2018). Learning pose-aware models for pose-invariant face recognition in the wild. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):379–393.

Ranjan, R., Patel, V. M., and Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):121–135.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015a). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015b). Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832.

Wang, R., Shan, S., Chen, X., and Gao, W. (2008). Manifold-manifold distance with application to face recognition based on image set. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Zhou, S. K., Chellappa, R., and Moghaddam, B. (2004). Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13(11):1491–1506.

Zhou, X., Jin, K., Chen, Q., Xu, M., and Shang, Y. (2018). Multiple face tracking and recognition with identity-specific localized metric learning. *Pattern Recognition*, 75:41–50.