Robust Actor Recognition in Entertainment Multimedia at Scale

Abhinav Aggarwal
aggarw@amazon.com
Amazon, India

Yash Pandya
yaspan@amazon.com
Amazon, India

Lokesh A. Ravindranathan*
lravindranathan@twitter.com
Twitter, India

Laxmi S. Ahire
ahilaxmi@amazon.com
Amazon, India

Manivel Sethu
mssethu@amazon.com
Amazon, India

Kaustav Nandy
kaustn@amazon.com
Amazon, India

ABSTRACT

Actor identification and localization in movies and TV series seasons can enable deeper engagement with the content. Manual actor identification and tagging at every time-instance in a video is error prone as it is a highly repetitive, decision intensive and time-consuming task. The goal of this paper is to accurately label as many faces as possible in the video with actor names. We solve this problem using a multi-step clustering process followed by a selection of face-instances that are: (a) representative of their member clusters and (b) aesthetically pleasing for visual identification. These face-instances can be matched with the actor names by automated or manual techniques to complete actor tagging. This solution is further optimized for seasons with repeating cast members which constitutes majority of the entertainment multimedia content. In such titles, the face labels from the previous episodes are efficiently used to pre-label faces in the subsequent episode. We guarantee the same level of accuracy even after scaling the solution to TV series seasons. This novel solution works in a completely realistic setup where the input to the solution is just the raw video. This is the first known work which has proved its robustness on more than 5000 TV episodes and movies across different genres, languages and runtimes with actors of diverse ethnicity, race, gender identity, age, etc. The proposed solution establishes a new state-of-the-art for cluster purity in both movies and TV series seasons by achieving near-perfect cluster homogeneity.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision tasks**; **Cluster analysis** .

KEYWORDS

face identification, clustering, face quality, actor tagging, movies, TV shows, prime video, netflix, disney+

ACM Reference Format:
Abhinav Aggarwal, Yash Pandya, Lokesh A. Ravindranathan, Laxmi S. Ahire, Manivel Sethu, and Kaustav Nandy. 2022. Robust Actor Recognition in Entertainment Multimedia at Scale. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548408

* Work done while at Amazon.

MM '22, October 10–14, 2022, Lisbon, Portugal

ACM ISBN 978-1-4503-9203-7/22/10.
https://doi.org/10.1145/3503161.3548408

1 INTRODUCTION

In 2019, the global entertainment market reached $100 billion (+1400 bps year-over-year) [1] in revenues for the first time. With growing revenue, there is a continued investment to enhance customer experience. Actor identification and localization are essential in enabling immersive experiences. For example, in visual question answering such as "who is the actor to the left of Jim Parksons?". It can enable automatic in-content quiz generation like answering "In which other movie did Edward Norton play a bald character?". In addition, video captioning needs actor names to generate name entity tags. We can also enable shopping within the content by answering questions such as "Find necklaces similar to the one worn by Angelina Jolie". This could be used by OTT platforms like Amazon Prime Video, Netflix, etc.

With the advancements in deep learning, face identification and tagging in images is a well studied problem with near perfect performance [25] [5] [32] [33] [16] [28] [18] [3]. However, scaling that to videos is non-trivial. Entertainment multimedia contains a variety of actor face poses with different illumination and focus along with post-editing and after effects to enhance visual appeal. This makes working with raw videos a challenging problem.

The current set of research [20] [7] [13] [26] [30] [29] majorly uses manually curated video tracks as an input which simplifies the problem to a great extent. These tracks are curated for primary actors and do not include the secondary and supporting cast. This also makes it challenging to extend the problem to real-world use cases where the cost of generating near-perfect face tracks, is similar to or more than that required for solving the problem manually.

With this limitation of using face tracks, the evaluation datasets are also limited. Typically a few genres are more difficult to address than others. For example, action movies with fast-moving scenes are typically more challenging. Most of the works typically use The Big Bang Theory, F.R.I.E.N.D.S and Sherlock [20] [14] [7] [13] [26] [30] for evaluation. These are Hollywood content with an evaluation done for majorly white actors with limited exposure to different genres and diversity of content. With the increased reach of global content and concerns around bias, it is important to build unbiased solutions.

Most of the methods based on video face clustering [20] [7] [13] [26] use additional training on the same content (i.e same TV show or even the same episode) as the evaluation. In practice, this is a bottleneck as it requires annotated data for every title which is not feasible. This greatly simplifies the problem and also makes title processing costly.

Along with these, approaches [27] use the cast list as an input. This is again a practical limitation as the cast list providers (IMDb,

Rotten Tomatoes, etc.) provide this information for only a few specific regions. For example. IMDb has rich actor profile data and images for primarily English-speaking countries, but not globally. For titles unavailable on such platforms, it is non-trivial to know the cast list beforehand. This causes practical limitations for non-English content and also for content produced by small independent filmmakers.

Manual generation of this information is a tedious process. To generate the metadata, operators need to watch the video data which may not be suitable to their taste because it can contain gore, horror, or nudity. The operators need to identify and tag actors whenever a new face appears on the screen and increasing the operator pool size is impractical since the number of titles is exponentially growing.

We want to address the problem of identifying credited actors in entertainment multimedia such as movies and TV shows. We want a practical and scalable solution which can be used by various multimedia platforms to answer questions such as "who is the actor on screen", "where have I seen this actor before?", etc.

In this paper, we propose a scalable fully automated actor tagging solution in entertainment multimedia. The solution takes the input video (movie/ TV episode) and generates a small set of unique faces. These faces are then mapped to actor names by using automated or manual steps. The goal of this solution is to accurately label as many faces as possible in the video with a minimum number of actor name mapping.

The major contributions of this paper are 1) Definition of the problem of actor recognition from videos in a practical setup. Only the raw video is provided as input. The face tracks or cast list is not available which makes the solution generic enough to solve the problem for real-world applications. 2) Novel SnapFilm clustering algorithm for actor recognition in video streams. This is a hierarchical algorithm that exploits video choreography techniques. 3) The proposed solution establishes a new state-of-the-art for cluster purity in both movies and TV series seasons by achieving near-perfect cluster homogeneity. The evaluation is performed across a variety of genres, ethnicity, race, gender identity, age, languages, etc. 4) We have optimized to make the actor face tagging fast and simple. In TV series seasons which have a high overlap of actors, we do not have to tag the same actors again for every episode.

To the best of our knowledge, this is the first known work which optimizes the recognition of actors in entertainment with the given constraints and evaluates on a large dataset.

2 RELATED WORK

Most studies have focused on face grouping and labelling for images. These approaches group the faces into clusters and interactively label the clusters. EasyAlbum [4] is a semi-automatic tool for labelling faces. [31], [37], [19] demonstrate that clustering-based solutions seem to be most effective for such tasks.

TV episodes and movies have their own set of challenges given its unconstrained scenarios such as occlusions, blurs, after effects, shot changes, face masks, etc. There is significant redundancy and spatial coherence between frames of the video. [23] used 22 episodes (2 per year for 11 years) of Friends to generate a face dataset. [2] also focussed on finding face groups in news by filtering in high confidence clusters. Both these approaches have focused on high precision with low recall.

Our problem is different from dataset generation. We want to identify the maximum number of people and label them, rather than identifying a few face clusters. We prefer a high precision system with a high value of recall. Along with the work discussed above, [12] labelled all the faces in news videos. They used MTCNN [36] and FaceNet model [25] with K-Means and (DBSCAN) clustering [8]. [30] used a supervised approach to carve the embedding space into balls of equal size, one for each cluster. It can predict clusters with 85% accuracy.

3 PROCESSING INDIVIDUAL TITLES

This section describes our solution to address the problem of actor identification in a movie or a TV episode. The system takes a video as input and generates clusters of faces along with a representative headshot or face for each cluster. It then invokes a workflow for mapping generated headshots to actor names. To simplify, we discuss the operator-based manual mapping in the below sections. The system then uses the identified faces to propagate the actor labels from tens of faces to thousands of faces. We introduce a metric *Coverage* based on the size of the cluster represented by a headshot. The coverage of a headshot is the proportion of the corresponding cluster size with respect to all faces detected in the video. This explains the screen presence covered by labelling an actor's face in the video.

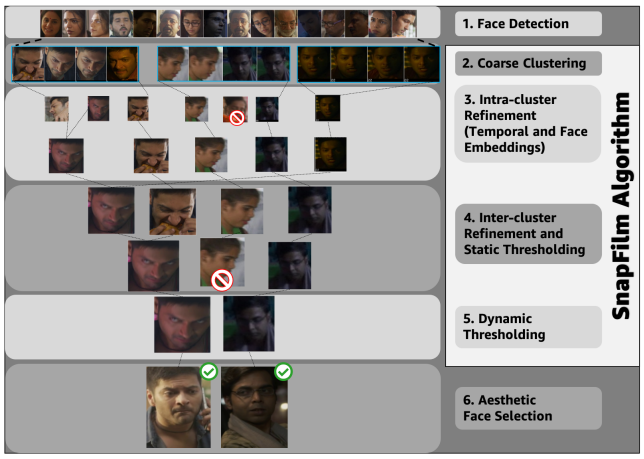

Figure 1: The various steps followed by our solution to cluster faces from a video and to generate an aesthetically pleasing and easy to identify headshot for each cluster.

We start by detecting faces from the video and generating facial embeddings for each detected face. We need to cluster the faces based on their facial embeddings. We tried using standard clustering algorithms but we saw numerous errors. The metrics for this are covered in the baseline subsection. This was due to the inherent nature of the problem. We see erratic camera movement, occlusions, inconsistent lighting conditions, after effects, and a lack of visual continuity in entertainment multimedia. We present SnapFilm algorithm to enable efficient clustering of actor faces in video.

Other related works rely on manually curated face tracks for only the credited actors which typically remove side actors and other noise present in the video. We operate on the direct raw video as opposed to manually curated face tracks. The algorithm is designed to exploit choreography techniques by orchestrating the clustering steps and tuning the thresholds. We also crop and choose aesthetically pleasing faces from the video to be shown to the customers to represent the actors. We use a multi-step filtering algorithm for face selection followed by cropping the image using professional photography techniques.

Fig 1 shows the flow chart of the various steps in this algorithm. We start by detecting faces from the video and generating facial embeddings in step 1. Our novel SnapFilm algorithm starts with **Coarse Clustering** (3.2.2) of the faces obtained from the video in step 2. This step aims to get homogenous clusters, at the cost of having multiple clusters of the same person. In step 3, we perform **Intra-clustering Refinement** (3.2.3) to create sub-clusters for each coarse cluster to reduce non-homogeneity. We break the clusters into sub-groups based on their temporal proximity. This step emulates generating face tracks from the video. We now select a face to represent each temporal group. We then match different temporal sub-groups based on the facial embedding of their representative faces within the same cluster to reduce redundancy. We also select a good quality representative face for representing these face groups created by combining temporal groups. In step 4, **Inter-cluster Refinement and Static Thresholding** (3.2.4) we cluster the representative faces from each group from all the clusters. We then remove some of these newly formed clusters which have a sparse screen presence to remove non-actor faces. We finally select the faces which are likely to be credited actors in step 5, **Dynamic Thresholding** (3.2.5). We use a metric *Coverage* based on screen presence of the face to decide if the cluster is worth the effort of getting manually labelled. We then generate aesthetically pleasing headshots to represent each cluster from the video in step 6 (3.3). These headshots are then manually mapped to actor names in a tool called Album. This multi-step SnapFilm algorithm helps in handling the complexities of clustering faces from a video and removing faces which are not relevant for identification as they might belong to background actors.

3.1 Face Detection and Embeddings Generation

We use high-resolution video and sample it at a constant interval (usually 1 frame sampled every 2-4 frames). The first step is to localize all faces in each of the sampled frames. We begin by detecting the faces from a video using a pre-trained face detection model [36] [6]. For embeddings for face recognition, we use a model based on the ResNeSt-50 [35] architecture which is trained on our custom internal dataset.

3.2 SnapFilm Clustering Algorithm

We have developed a novel solution which includes multi-step clustering, screen-presence based cluster selection and an algorithm to select a good quality face for each cluster. We define two relevant metrics which we use to judge the precision and recall of our solution. WCP (Equation 1) refers to weighted cluster purity and is used to measure the precision of the solution. TC% (Equation 2) refers to total coverage and is used to measure recall for the solution. These are measured as follows:

$$\begin{aligned} &Weighted\ Cluster\ Purity(WCP) \\ &= \frac{\Sigma(Cluster\ size\ *Cluster\ Purity)}{\Sigma(Cluster\ size)} \end{aligned} \tag{1}$$

$$\begin{aligned} Total\ Coverage(TC) &= \frac{Number\ of\ faces\ labelled\ in\ the\ video}{Total\ Number\ of\ faces\ in\ the\ video} \\ &= \Sigma\ (Coverage\ of\ all\ clusters\ labelled) \end{aligned} \tag{2}$$

Before we describe the individual steps in the SnapFilm Clustering Algorithm in detail, we want to establish the need for a multi-step clustering algorithm designed specifically for videos. We do this by evaluating if standard clustering algorithms can effectively cluster actor faces in entertainment multimedia.

3.2.1 Baseline. In this section, we evaluate the performance of standard clustering algorithms for a sample title (Mirzapur S1E4). We use the face embeddings (Section 3.1) generated for all detected faces and directly apply the clustering algorithms with the embeddings as input features.

These standard algorithms assign a cluster for every detected face which results in a TC of 1. But coverage of all faces is undesirable in the practical scenarios because we want to generate clusters for only credited actors even without a-priori information about the cast list. The SnapFilm algorithm is designed to optimally choose clusters of potentially credited actors while discarding the faces of non-credited actors. To have a like-for-like comparison between the standard clustering algorithms and our SnapFilm algorithm, we perform an additional benchmark by covering a TC of 1. For this comparison, we remove all filtering steps from the SnapFilm algorithm. This comparison is covered in Table 1. Though this is not the direct focus of the SnapFilm algorithm, we still outperform all the baseline solutions in the tradeoff between WCP and cluster count. The baseline methods include the K-Means algorithm with 50 clusters, Agglomerative clustering (with single linkage and complete linkage) using a distance threshold on the cosine distance as the stopping criteria, DBSCAN clustering, FINCH [24] algorithm with the number of partitions selected based on their silhouette score, and FINCH algorithm with 2 partitions.

We also want to compare the SnapFilm clustering algorithm in a practical scenario where we have removed background actors and other faces in the final clustering step. The SnapFilm algorithm selects clusters with a TC of 0.81 for our sample title. To compare other solutions, we remove the smallest clusters formed by the clustering algorithms such that the TC is approximately 0.81. For the baseline, we evaluate the FINCH algorithm with 2 partitions, the FINCH algorithm with the number of partitions selected based on their silhouette score, Agglomerative Clustering with single linkage and a cosine distance threshold, and the K-means algorithm with 50 clusters. After removing the smallest clusters, they all have their TC between 0.79 to 0.82. This comparison is covered in Table 2.

From Table 1 and Table 2, we can see that the SnapFilm algorithm is the best choice for both scenarios. The algorithm consistently achieves a near-perfect WCP score but it should be noted that it comes at the expense of a large number of clusters if we need

Clustering Algorithm	Clusters	WCP	TC
K-Means	50	0.897	1
Agglomerative (Single Linkage)	90	0.756	1
Agglomerative (Complete Linkage)	657	0.984	1
DBSCAN	241	0.840	1
FINCH [24]	11	0.832	1
FINCH (partitions = 2) [24]	204	0.963	1
SnapFilm algorithm (no filtering)	178	0.982	1
SnapFilm algorithm	21	0.991	0.81

Table 1: Comparison between standard clustering algorithms and the proposed SnapFilm algorithm for the title of Mirzapur S1E4 by labelling all faces in the video

Clustering Algorithm	Clusters	WCP	TC
FINCH (partitions = 2) [24]	121	0.967	0.81
FINCH [24]	6	0.868	0.79
Agglomerative (Complete Linkage)	155	0.990	0.81
K-Means	28	0.932	0.82
SnapFilm algorithm	21	0.991	0.81

Table 2: Comparison between standard clustering algorithms and the proposed SnapFilm algorithm for the title of Mirzapur S1E4 by labelling clusters to cover around 81% of screen presence in the video.

Coarse Clustering Algorithm	Coarse Clusters	Final Clusters	WCP	TC
K-Means (50 Clusters)	50	14	0.967	0.85
DBSCAN	241	23	0.953	0.85
FINCH [24]	11	13	0.907	0.85
Agglomerative (Single Linkage - Distance Threshold)	90	14	0.792	0.85
Agglomerative (Complete Linkage - Distance Threshold)	657	21	0.99	0.8
Agglomerative (Complete Linkage - 256 clusters)	256	21	0.991	0.81

Table 3: Comparison between different clustering algorithms for the coarse clustering step. We use different clustering algorithms for the coarse clustering step while keeping the other steps unchanged. The stopping criteria for the clustering algorithms is mentioned in brackets.

complete coverage(TC=1). This is unavoidable because when we work with all faces detected in a video, there are a large number of identities but most are usually not credited actors. The SnapFilm algorithm is thus designed to be able to filter them out and generate a small number of faces for identification.

3.2.2 Coarse Clustering. In the first step, we cluster all the detected faces based on facial embeddings (3.1). The aim is to get homogenous clusters, at the cost of having multiple clusters of the same person. We do not use any a-priori input on the number of actor faces present in the video. We have chosen the number of clusters to be 256, which is around 6x-12x the size of most cast lists. This helps us to operate in a high-precision setting. We use agglomerative clustering [21] with complete linkage. We evaluated various other clustering algorithms for this step, and the comparison of the result is covered in Table 3.

3.2.3 Intra-Cluster Refinement. We have coarsely clustered faces using their face embeddings in the previous step. In this step, we ensure a second layer of verification to reduce non-homogeneity. We break the clusters into sub-groups such that each sub-group is homogeneous. We exploit the video choreography techniques of shot patterns [17], [15], [9], [10] and rich temporal information for finding similar faces within the cluster. We start by creating temporal sub-groups within each cluster, these are similar to the kind of results we can get from a face tracking algorithm. We create these temporal groups by placing faces within each cluster which are from temporally close frames into the same temporal group. This is a simpler yet effective alternative compared to regular face tracking because shots in entertainment multimedia are often short and feature many cuts. For example, we see shot changes every 2.54 seconds in a scene where few people are talking.

We then represent each temporal group by a single representative face. We call it optimal face representation (OFR). We use FSANet [34] for face pose, along with the face image size and combine these metrics using a simple heuristic to determine the OFR.

We filter out a small subset of these temporal groups having a very small number of faces and a bad quality OFR. Most of these faces are blurred due to the bokeh effect of the camera which puts focus on the characters relevant to the scene. This is also helpful in removing the faces of non-credited background actors from crowded scenes. We do not remove more than 5% of all faces in this step. This ensures we do not remove too many faces at a very early stage which can lead to potentially missing out on the faces of credited actors.

Next, we group these temporal groups by using the facial embeddings of their OFR. We combine two temporal groups if their OFR match with very high confidence. Entertainment multimedia has a significantly high number of temporal groups because the camera cuts to move from one speaker to another [22]. The temporal groups we have at this stage are still large in number (300-1000). We need to further combine high confidence matches within a coarse cluster. In the case of homogenous coarse clusters, we end up with one face group at this stage. For non-homogeneous coarse clusters, we can end with multiple face groups. We represent the obtained face groups by their OFR. We generally end with 100-300 face groups at this stage because many coarse clusters completely consisting of blurred low-quality faces are removed.

3.2.4 Inter-Cluster Refinement. We have obtained high-quality OFR from the previous step. We now cluster face sub-groups obtained from the previous step. We start to cluster the OFR of these face sub-groups across all course clusters using agglomerative clustering with complete linkage. With the input having high-quality OFR, we can generate highly homogeneous clusters. We are still working with a large number of faces and we don't want to introduce any impurities in the face groups, thus we stick to a very conservative clustering algorithm which relies on complete linkage. All other clustering algorithms can create a smaller number of groups but introduce some errors at this stage.

We then remove low-size clusters from the previous step such that we are covering at-least 85% of *Coverage*. This helps to further remove non-actor faces from the clusters. We represent these face clusters with their OFR.

We then use agglomerative clustering with single linkage on these small number of high-quality face clusters to get the final set of faces. Single linkage adds faces to the clusters more aggressively but we can do it because we have very few high-quality faces at this stage. This is because we only have clusters with a large number of faces at this stage. The number of clusters remaining is typically 1.2x-1.5x the size of the cast list.

3.2.5 Dynamic Thresholding. After the previous step which is the last clustering step, we get a set of faces to be labelled along with their *Coverage*. We start with a threshold of 85% *Coverage* and then recursively remove clusters of the smallest sizes until we no longer have any small clusters. We define small clusters as clusters with a potential screen presence of less than 5 seconds. These are generally clusters of side actors which are not credited. We ensure that we cover at least 70% of the screen presence. Based on our experiments, we generally cover around 80% of screen presence. These faces are then labelled by the operators efficiently.

3.3 Face Selection

For each of the face clusters to be labelled, we generate a face to be shown to human operators for actor labelling. We cannot use the faces identified in OFR as they are efficient in differentiating identities from an embeddings perspective but lag aesthetic appeal. For example, OFR overprioritizes the faces having a frontal pose which is not ideal from an aesthetic perspective.

The selection of headshot comprises two major problems. The first is to identify the frame of the representative face for each credited actor. The challenge is that we do not have influence over frame conditions (such as illumination, location, background homogeneity, focus, and sharpness in the input). The faces themselves can have diverse poses, presence of occlusions, and different expressions. We overcome these factors by using our robust algorithm to select a frame where the actor is easy to recognize.

To select the frame for the representative face in each group, we filter out faces which have characteristics of a low-quality face in a series of steps. Each step has been tuned based on observations from 1800 different actor faces. We filter out faces from the groups in each step by rejecting low-quality faces based on different criteria such as face size, FaceQNet score (a deep learning model [11] for identifying face quality with stress on face sharpness), face pose (inferred using a capsule-based deep learning model [34]), and face brightness. The best ranked (based on FaceQNet score) remaining face is called the representative face for the group.

The second problem is to extract the optimal headshot from the selected frame. Based on an in-depth study of various photography blogs and professional photography courses, we have identified some key techniques to yield automatic professional quality headshots. This is the first known work to automatically extract professional quality headshot from a video frame. This also helps to ensure a standardized experience for the customers.

To select the professional quality headshot from the frame, we localize the actor in the frame using a deep learning based face detection algorithm [36] [6]. The actor's faces are cropped to be appealing and easy to recognize for humans. We maintain the aspect ratio of these images and ensure that the headshots have a standardized composition with their shoulders and hair being partially visible.

We can look at some representative headshots curated by our algorithm in Figure 2. Our user acceptance testing shows that our standardized aesthetically pleasing headshots consistently outperform manual choice by 80%. The results were of similar quality in the remaining 20% of faces.

Stage	Clusters	WCP	TC
Coarse Clustering	256	0.891	1
Temporal Grouping	1183	0.989	1
Temporal Groups Filtered	877	0.991	0.951
Intra-Cluster Grouping	203	0.986	0.951
First Inter-Cluster Grouping	182	0.983	0.951
Removal of small clusters	56	0.990	0.85
Single Linkage Clustering	34	0.986	0.85
Dynamic Thresholding	21	0.991	0.81

Table 4: Effect of each step of SnapFilm Algorithm on WCP and TC of clusters on a sample title (Mirzapur S1E4).

Figure 2: The visually clear and aesthetically pleasing faces curated by the face selection component without any manual intervention.

3.4 Sample Title

In this sub-section, we look at the effect of each step in the SnapFilm algorithm on a sample title (Mirzapur S1E4). We start with 11295 detected faces and finally generate 21 representative headshots which need to be labelled. The WCP and TC of the clusters after each step are covered in Table 4. We can see that each step of the algorithm has a positive contribution in either reducing the number of clusters or increasing the WCP.

4 TV SEASONS

We extend the approach for individual titles when processing seasons. A TV series season generally has a high overlap of the cast list in different episodes. We generate a dynamic number of labelling jobs based on content and optimally combine the labelling from multiple episodes to generate labelling jobs. We also use the labels from previous labelled jobs of the same season reducing labelling efforts. We make this optimization without adding any additional errors. The cast information, when available is used in the labelling jobs to simplify the labelling workflows for the operators but not in 3.2.

4.1 Processing seasons

We have two major choices to consider when processing a season. The first option is to process the season at once and generate a single labelling job. The second option is to process the labelling jobs independently (Section 3) and then optimize the labelling effort.

The first option poses the challenge of scaling the system to seasons with 30-40 or more episodes. It is more scalable to process the titles individually. The SnapFilm algorithm is a cascade of steps, it is more likely to introduce impurities in the clustering process. In the second option, we do not need to change the algorithm. We have seen that the labelling effort is proportional to the cast list size. Generally, each episode has a few unique actors. In the first approach, we would have to label the union of cast list as output. It is generally easier to label individual episodes than to label a season as a whole. There are lesser chances of missing a cameo appearance in the second approach. We use the second approach.

4.2 First Labelling Job

We use the faces obtained from the previous step to identify the first job to be labelled for a given season. The first job does not have any pre-labelled faces available and has an experience similar to a movie labelling job. We want to select an episode which has a high overlap with other episodes. For this, we compute the embeddings distance for all pairs of faces across different episodes. Since the first episode is unfamiliar for operators, it is better to reduce the number of faces and cast list to be labelled. This reduces the cognitive load on the operators and makes job more efficient. We use the faces to be labelled to identify the optimal episode. We use a rule-based heuristic based on the number of faces to be labelled, and face similarity with other episodes to choose the optimal episode to be labelled from the season.

Figure 3: The faces selected from the algorithm are labelled in a tool called Album.

4.3 Remaining labelling jobs

We process the remaining labelling jobs after processing the first episode. We use the labels from the already labelled jobs to automatically label faces in the other episodes. At each step, we send one labelling job to the operators. This process is repeated till the complete season has been labelled.

4.3.1 Grouping episodes. Instead of labelling all the episodes of the season individually, we intelligently group the episodes to create a single annotation task. Let us assume that the cast list of episode E_i is denoted by C_i. When the episodes $\{E_{k_1}, ...E_{k_n}\}$ are merged to create a single job J, the cast list for the job is represented by $C_{k_1} \cup \cdots C_{k_n}$. Our experiments have identified that the cast list length is the most prominent feature when merging episodes to create a single job. The human cognitive load increases sharply with the increase in the cast list. In the case of a season with 20 episodes, if all the episodes are combined and have exactly the same cast list, we end up having the merged cast list the same as for any episode. In case all the episodes are combined together, and there are 90% same people and just 10% different cast in every episode, we end up with $\tilde{3}$x extra cast during cast merge.

There can still be cases where we are not able to label all the faces in an episode with previous labels. We get these faces manually labelled. We use a simple heuristic based on the increase in the cast list length, the number of unlabelled faces, and the maximum faces present in a job to arrive at a group for the episodes. We then group the other episodes and generate a labelling job. This labelling job might include anywhere from one episode (no overlap of cast list) to one job (complete overlap of cast list). The labelling job is dynamically chosen based on the input.

4.3.2 Verification Process. Once the episodes are grouped, they are ready to be labelled (Fig 3). We weighed multiple UX options based on cognitive load, the number of clicks, and time taken. The final UX is a list on the top with actor headshots generated by the solution. The lower half of the page has the cast list (if available). The labelling task is to drag and drop faces from the top list to the correct actor column. For non-available actors, the cast list can be added by the operators during labelling. The verification faces are pre-labelled as we can see in the second row of the actor column.

We do not skip the pre-labelled faces. We give them as a verification task to the operators. The verification task is 15-30x faster than the face to actor name mapping task. This helps to ensure that we are not introducing any incorrect labels. This is especially important because each face to be labelled represents a large number of faces given by *Coverage*. Incorrect labelling of one face could have an impact not on just one face but the complete title. The only challenge to consider is the number of incorrect verification tasks. Our experiments show lesser than 1%.

Title Name	Clusters	Actors	Impurity	WCP	TC%
The Big Bang Theory S1E1	6	6	0	1	84.5
Pequeñas Coincidencias S1E5	28	21	0	1	85
Andy Griffith S4E8	5	5	0	1	85
Atlanta S1E2	13	13	0	1	80
The Goes Wrong Show S1E3	16	11	0	1	85
Mirzapur S1E4	24	19	1	0.98	80
Guava Island	32	-	1	0.97	70.5
Titanic	78	-	3	0.99	73

Table 5: Individual Titles Results

5 RESULTS

We present the results on a few diverse titles and benchmark the solution performance. We perform an extensive evaluation of the solution on 5000 titles comprising TV series seasons and Movies in the production system. We were not able to generate ground truth for all these titles because of extensive annotation effort. To cater to this, we have tested these titles through visual inspection by the manual operators and discussed the quality metrics on these titles.

5.1 Individual Titles

We generate the faces to be labelled for each of the titles in the ground truth. We use the ground truth labels to automatically label

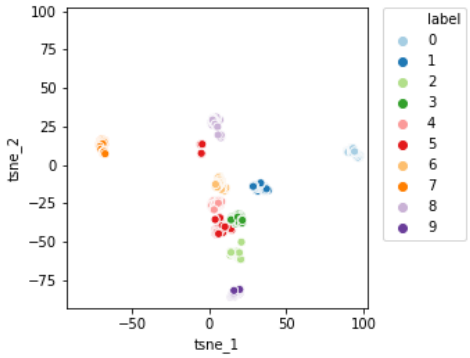

Figure 4: Qualitative results being shown by a typical t-SNE plot showing for an episode with embeddings (best viewed in color) distance and actor label predicted. The color of each dot represents different actors in a title.

the generated faces. We then propagate the labels to the cluster of faces obtained from the SnapFilm algorithm (3.2).

We compare the predictions of the faces to the ground truth labels to generate the technical metrics in Table 5. Clusters represent the total number of faces to be labelled generated by the solution. Actors represent the unique actors in the clusters generated. Impurity represents the number of non-homogenous clusters. WCP (Equation 1) refers to weighted cluster purity. TC% (Equation 2) refers to total coverage. We see that we almost saturate the results with near-perfect WCP. One may argue that we reach this benchmark at the cost of TC. But this comparison is incorrect. Most existing works operate on an actor face track level and manually remove background actors and other faces. We do not use this information since it is infeasible in a practical scenario. We do not remove these non-useful actor faces from TC calculation, hence we see a drop in TC. We have instead solved the problem of segregating actor faces from non-actor faces in the video by using dynamic thresholding (3.2.5). This solution automatically identifies the faces to be considered based on the type of content. For example, in a sports genre content, we expect to encounter a considerably higher number of background faces compared to a sitcom like Andy Griffith. Table 1 also showcases that we outperform baseline solutions with lesser clusters and better WCP even while covering TC of 1.

We can see how the SnapFilm algorithm can capture similarities not obvious in the embeddings space by looking at a t-distributed Stochastic Neighbour Embedding (t-SNE) 2D map of the embeddings space in Fig 4. We can see here that actor 5 is present at multiple locations in the title which is separated by a huge difference from facial embeddings. The algorithm has been able to separate that well. Also, the distance between some dots of actor 3 dots seems to be more than that between actor 3 and actor 5. We can see this similar for actor 4 and actor 5, actor 6 and actor 4 etc. We can see how this solution can separate these seemingly similar embeddings. To validate, we also manually compared the actor predicted labels to confirm that actor labelling was correct. We can see complexity in image quality within a cluster in Fig 5. The solution is able to handle such diversity in ethnicity, race, gender and age

Figure 5: Diversity of faces within an actor cluster. We show the clustering algorithm is agnostic to changes in focus, pose, illumination, brightness, sharpness and across actor ethnicity, race, gender and age groups.

Title	Faces	#Verification	TC
An Unknown Enemy - S1E7	21	65	75.4%
An Unknown Enemy - S1E6	39	54	59.5%
Andy Griffith Show - S4E4	28	37	77.0%
Andy Griffith Show - S4E5	8	24	38.0%
F.R.I.E.N.D.S. S5E20	10	165	86.2%
F.R.I.E.N.D.S. S5E1	33	171	87.2%
The Big Bang Theory: S1E2	5	78	87.1%
The Big Bang Theory: S1E6	21	83	88.4%

Table 6: Metrics for first episode selection

groups because of the custom dataset which is used to train the face embeddings model.

5.2 Optimization for TV Seasons

We give focus to the labelling order of episodes in a season. Table 6 shows the selected first episode and compares it with a non-optimal first episode candidate in the two rows for the selected seasons. Faces represent the number of faces to be labelled in the first episode. The verification count is the number of verification faces for the remaining episodes. Coverage represents the screen presence which is already labelled in other episodes because of verification jobs. We see that even with a lot fewer faces, we can label a similar or more coverage compared to the worst episode. This shows that selecting the first episode is an important problem which can impact the total annotation time for the complete season.

$$Precision = \frac{\#faces\ correctly\ labelled}{\#Total\ faces\ predicted} \tag{3}$$

$$Recall = \frac{\#faces\ correctly\ labelled}{\#Total\ faces\ present} \tag{4}$$

Actor name	Total faces	Correct	Precision	Recall
Johnny Gakecki	19	14	1	0.74
Jim Parsons	16	12	1	0.75
Kaley Cuoco	18	12	1	0.67
Simon Helberg	15	12	1	0.80
Kunal Nayyar	16	14	1	0.88
Sara Gilbert	3	2	1	0.67
Others	42	-	-	-

Table 7: Metrics from labelling BBT Season using one episode

Season	#Episodes	Album Time	Video Length
Guzman	52	1.5	46
Macgyver	34	2.4	42
Backdoors S1	20	1.1	27
Backdoors S2	20	1.2	26
Friends S3	25	1.8	22

Table 8: Average time (minutes) to label season

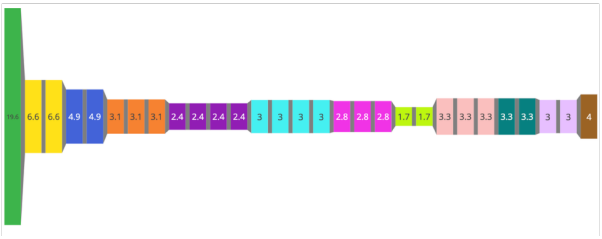

Figure 6: Typical funnel for time spent in labelling job. The example represents labelling all episodes of ICesaroni Season. The grouped episodes as represented with the same colour. The numbers show the labelling time per title.

Table 7 uses labels from just episode 5 (6 actors) and propagates the labels to the remaining 16 episodes of The Big Bang Theory Season 1 (BBT). Our predictions for this season have a precision of 1 and a recall of 0.758. But when we calculate the recall in terms of coverage, the 6 labels accurately label 87.1% of all faces for the rest of the season.

Table 8 shows that with a minimal manual effort of just 2 minutes per episode, we can label the complete season. Most of the labelling time for a season is spent completing the first labelling job. The verification jobs can be done in negligible time and thus the time taken for the rest of the episodes is very low when there is a significant overlap in the cast list of all episodes in the season. We visualize this in Fig 6. We see that for Iceseroni, the first labelling job took 19.6 minutes, and the next was a combination of 2 episodes and took 6.6 minutes each. Then this flattens out to 3-4 minutes per title. We needed to label only 12 jobs from 29 episodes. This makes season labelling 5x faster. In the production setup, we see less than 0.1% of false positives generated by the system across 5000 titles. We see an average of 3-4 minutes of labelling time in the Album tool per title.

5.3 Benchmark against State of the Art

Most of the existing works are tested only on a few popular English titles (The Big Bang Theory, Buffy the vampire slayer, Friends, Sherlock) due to a lack of available ground truth datasets. We show our results on a diverse set of titles in Table 5 with different formats (Andy Griffith - Black and White), and longer runtime (Titanic - 3h 14m). These titles also have cast with more racial diversity (Atlanta, Guava Island, Mirzapur, Pequenas Coincidências).

Title Name	C1C	FINCH	BCL
Friends S3 E1-25	0.77	0.697	-
Buffy S5 E1-E6	0.88	0.829	0.865

Table 9: WCP Benchmarks [13] [30]

Title Name	Clusters	FPA	Impurity	WCP	TC%	AMC
Friends S3 E1-25	504	1.14	3	0.999	79.41	98
Buffy S5 E1-E6	98	1.10	2	0.998	84.71	32

Table 10: Benchmarked Titles Results

We report the state-of-the-art on Friends and Buffy the Vampire Slayer in Table 9. The table shows the average WCP over the complete season for Constrained 1NN hierarchical Clustering method (C1C) [13], FINCH clustering algorithm [26], and Ball Cluster Learning(BCL) [30]. Our problem is slightly different from these benchmarks. These solutions use manually crafted face/head tracks taken from the videos. This curation removes a lot of noise from short tracks and low-quality faces and focuses on only the credited actors by removing uncredited actor face tracks. Our solution operates directly on the raw video, it also needs to automatically discard faces which might not be relevant. Our solution can lose some credited actor faces but it ensures that at least 70% of the screen presence is covered.

Unfortunately, we could not access the benchmark datasets used by other papers as they aren't available online. We compute metrics for our algorithm, based on the problem it is designed for. We can see detailed metrics for our solution for Friends and Buffy in Table 10. Clusters are the total number of faces which need to be labelled across all episodes of the season. Faces per actor (FPA) is the average number of faces which need to be labelled per actor per episode. Impurity is the count of clusters which have faces from more than one actor in a cluster. WCP is the average WCP over all episodes. TC% is the total coverage % of the selected clusters from our solution cover. Actor Mapping Count (AMC) is the total number of faces which need to be tag by the annotator to complete actor tagging for all episodes. This number is much smaller than the total number of clusters because verification jobs don't need to be tagged with an actor name. Our WCP metrics show how all our clusters are very pure.

6 CONCLUSION

The problem of actor tagging in entertainment multimedia requires the identification of unique actors and all their occurrences with high accuracy. Our solution relies only on the video as an input to generate faces to be labelled with both high precision and high recall. We are automatically able to filter out non-credited actor faces when generating faces to be labelled. The solution is further optimized for seasons by efficient labelling job selection, grouping and automatic labelling of the faces. This solution has achieved state-of-the-art results on both existing datasets and generalized on more than 5000 titles. In future work, we would be exploring automatic labelling of titles without human effort by leveraging the internet for actor name mapping.

REFERENCES

[1] 2021. https://variety.com/2020/film/news/global-entertainment-industry-surpasses-100-billion-for-the-first-time-ever-1203529990/

[2] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee-Whye Teh, Erik Learned-Miller, and D.A. Forsyth. 2004. Names and Faces in the News. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2004).

[3] Kai Chen, Qi Lv, Taihe Yi, and Zhengming Yi. 2021. Reliable Probabilistic Face Embeddings in the Wild. *ArXiv* abs/2102.04075 (2021).

[4] Jingyu Cui, Fang Wen, Rong Xiao, Yuandong Tian, and Xiaoou Tang. 2007. EasyAlbum: An Interactive Photo Annotation System Based on Face Clustering and Re-ranking. *Conference on Human Factors in Computing Systems* (2007), 367–376.

[5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2019). https://doi.org/10.1109/cvpr.2019.00482

[6] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. 2019. RetinaFace: Single-stage Dense Face Localisation in the Wild. *CoRR* abs/1905.00641 (2019). arXiv:1905.00641 http://arxiv.org/abs/1905.00641

[7] Mark Everingham, Josef Sivic, and Andrew Zisserman. 2006. Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video. In *BMVC*.

[8] Michael Hahsler, Matthew Piekenbrock, and Derek Doran. 2019. dbscan: Fast Density-Based Clustering with R. *Journal of Statistical Software* 91, 1 (2019), 1–30. https://doi.org/10.18637/jss.v091.i01

[9] Alex Hall, Amy Pavel, Alyosha Efros, and Maneesh Agrawala. [n.d.]. A Tool for Computational Analysis of Narrative Film. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-102.pdf. [Online].

[10] Timothy Heiderich. [n.d.]. *Cinematography techniques: The different types of shots in film.*

[11] Javier Hernandez-Ortega, Javier Galbally, Julian Fierrez, Rudolf Haraksim, and Laurent Beslay. 2019. FaceQnet: Quality Assessment for Face Recognition based on Deep Learning. arXiv:1904.01740 [cs.CV]

[12] Anirudh Jain, Matthew Sun, and Cherry Zou. 2017. Unsupervised Face Recognition in Television News Media. http://cs229.stanford.edu/proj2017/final-reports/5244380.pdf. [Online].

[13] Vicky S. Kalogeiton and Andrew Zisserman. 2020. Constrained Video Face Clustering using1NN Relations. In *BMVC*.

[14] Prakhar Kulshreshtha and Tanaya Guha. 2018. An Online Algorithm for Constrained Face Clustering in Videos. *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), 2670–2674.

[15] P. Lehman and W. Luhr. 2018. *Thinking about Movies: Watching, Questioning, Enjoying*. Wiley. https://books.google.co.in/books?id=ZClxDwAAQBAJ

[16] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2018. SphereFace: Deep Hypersphere Embedding for Face Recognition. arXiv:1704.08063 [cs.CV]

[17] George Lowery. [n.d.]. Pattern in movies mimics that found in our brain. https://news.cornell.edu/stories/2010/03/study-pattern-movies-mimics-found-our-brain. [Online].

[18] Qiang Meng, Shichao Zhao, Zhida Huang, and F. Zhou. 2021. MagFace: A Universal Representation for Face Recognition and Quality Assessment. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 14220–14229.

[19] Charles Otto, Dayong Wang, and Anil K. Jain. 2018. Clustering Millions of Faces by Identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 2 (Feb 2018), 289–303. https://doi.org/10.1109/tpami.2017.2679100

[20] Omkar M. Parkhi, Esa Rahtu, Qiong Cao, and Andrew Zisserman. 2020. Automated Video Face Labelling for Films and TV Material. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 780–792. https://doi.org/10.1109/TPAMI.2018.2889831

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Stephen Prince and Wayne E. Hensley. 1992. The Kuleshov Effect: Recreating the Classic Experiment. *Cinema Journal* 31, 2 (1992), 59–75. http://www.jstor.org/stable/1225144

[23] Deva Ramanan, Simon Baker, and Sham Kakade. 2007. Leveraging archival video for building face datasets. *International Conference on Computer Vision* (2007).

[24] M. Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. 2019. Efficient Parameter-free Clustering Using First Neighbor Relations. https://doi.org/10.48550/ARXIV.1902.11266

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jun 2015). https://doi.org/10.1109/cvpr.2015.7298682

[26] Vivek Sharma, Makarand Tapaswi, M. Saquib Sarfraz, and Rainer Stiefelhagen. 2020. Clustering based Contrastive Learning for Improving Face Representations. arXiv:2004.02195 [cs.CV]

[27] Vivek Sharma, Makarand Tapaswi, M. Saquib Sarfraz, and Rainer Stiefelhagen. 2020. Video Face Clustering With Self-Supervised Representation Learning. *IEEE Transactions on Biometrics, Behavior, and Identity Science* 2 (2020), 145–157.

[28] Yichun Shi and Anil K. Jain. 2019. Probabilistic Face Embeddings. arXiv:1904.09658 [cs.CV]

[29] Krishna Somandepalli, Rajat Hebbar, and Shrikanth S. Narayanan. 2021. Robust Character Labeling in Movie Videos: Data Resources and Self-supervised Feature Adaptation.

[30] Makarand Tapaswi, Marc T. Law, and Sanja Fidler. 2019. Video Face Clustering with Unknown Number of Clusters. arXiv:1908.03381 [cs.CV]

[31] Yuandong Tian and Wei Liu. 2007. A Face Annotation Framework with Partial Clustering and Interactive Labeling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2007).

[32] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. CosFace: Large Margin Cosine Loss for Deep Face Recognition. arXiv:1801.09414 [cs.CV]

[33] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*. 529–534. https://doi.org/10.1109/CVPR.2011.5995566

[34] Tsun-Yi Yang, Yi-Ting Chen, Yen-Yu Lin, and Yung-Yu Chuang. 2019. FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation From a Single Image. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2019).

[35] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. 2020. ResNeSt: Split-Attention Networks. arXiv:2004.08955 [cs.CV]

[36] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (Oct 2016), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342

[37] Chunhui Zhu, Fang Wen, and Jian Sun. 2011. A Rank-Order Distance based Clustering Algorithm for Face Tagging. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2011).