Unsupervised Face Recognition in Television News Media

Anirudh Jain * **Matthew Sun** * **Cherry Zou** *

Abstract

Recently, the impact of mass media has become a contentious issue in American society. It has become increasingly clear that the media does not only communicate stories; instead, it shapes, fuels, and participates in the news itself. Some of the central issues when discussing mass media are identity and representation: who gets to communicate the news, and who is typically on display in the media, particularly on TV? To answer these questions, we apply unsupervised learning methods to television news broadcast video from the Internet Archive to identify and cluster faces based on unique personal identity. In particular, we evaluated the effectiveness of k-means, Gaussian mixture models, rank-order clustering, and DBSCAN using both qualitative and quantitative metrics, including silhouette score and inertia.

1. Introduction

Recently, the role of the mass media in shaping public perceptions at scale has come under increasing scrutiny. Notably, many of the questions researchers wish to ask about mass media revolve around the question of who, or making conclusions based on the identifying unique individuals on screen. Some examples: identifying the person with the most screentime in a given month or analyzing representation in media by gender and race. These are tasks for which distinguishing between individuals who appear in the media is prerequisite.

In September 2012, the Internet Archive launched the TV News Archive, a repository of 500,000+ TV news broadcasts aired since 2009 (2017). Closed captions are also provided alongside the video, allowing for richer interactions with the video data. Researchers hope to enable users to ask identity-based queries about this data, such as who appeared on Fox News and discussed immigration the greatest number of times in January.

The goal of our project is to automatically recognize and cluster unique faces in television news media. Unsupervised learning is the best particularly well-suited for this task, since the input to our algorithm is a TV news broadcast video, potentially supplemented by a closed captions text file, and our output a clustering of all faces which appear in the video by identity. To do so, we first run the input through a preprocessing pipeline, which consists of first extracting frames from the video and detecting and aligning faces using a neural network. To generate the vectorized face embeddings, we use FaceNet, and additionally attempted to use n-gram features from closed captions to augment the vector. For clustering, we used unsupervised learning techniques, including k-means, Gaussian mixture models, rank ordering, and DBSCAN.

It should also be noted that portions of this final project reported in this paper were performed with respect to another class, CS221. In particular, the text feature augmentation and baseline analysis portions of the project were part of our project for CS229. Though some of these results are noted for comparative analysis in this paper, we believe the nature of those two aspects are more in line with CS221.

2. Related Work

Publication	Features	Clustering method	# Face images	# Subjects
Ho et al. [3]	Gradient and pixel intensity features	Spectral clustering	1,386	66
Zhao et al. [4]	2DHMM + contextual	Hierarchical clustering	1,500	8
Cui et al. [5]	LBP, clothing color + texture	Spectral	400	5
Tian et al. [6]	Image + contextual	Partial clustering	1,147	34
Zhu et al. [7]	Learning-based descriptor [8]	Rank-order	1,322	53
Vidal and Favaro [9]	Joint subspace learning and clustering	-	2,432	38

The above table lists prior work on face clustering with the face representation and clustering algorithm used as well as the largest dataset size used in terms of face images and the number of subjects. Ho et al. (2003) developed variations on spectral clustering by computing the probability that two face images are of the same object. Zha et al. (2006) clustered personal photograph collections by combining a variety of contextual information including time based clustering, and the probability of faces of certain people to appear together in images. They also used identity estimates from a Hidden Markov model, and hierarchical clustering results based on body detection. Cui et al. (2007) developed a semi-automatic tool for annotating photographs, which employs clustering as an initial method for organizing photographs. They extracted features from detected faces, and color and texture features were extracted, and then spectral clustering was performed and evaluated on a dataset consisting of 400 photographs of 5 subjects.

Tian et al. (2007) refined this approach by incorporating a probabilistic clustering model which used a junk class that

allowed the algorithm to discard clusters that do not have tightly distributed samples. Zhu et al. (2011) developed a dissimilarity measure based on two faces being in each others nearest neighbor lists, and perform hierarchical clustering based on the resulting rank-order distance. However, this clustering method was evaluated on small datasets (approximately 1, 300 face images).

To the best of our knowledge, our study is the first to attempt to do unsupervised identity clustering on a large-scale (80,000 images) and the first to actually run it on TV and news footage. From the literature review, it is clear that most studies used pre-extracted faces rather than using videos as our work does. Moreover, the 128-byte feature extractors we use are significantly smaller than the features employed in other studies allowing for higher efficiency and scalability.

3. Dataset and Features

We take our data from the TV News Archive (Archive, 2017). Specifically, we use 30 hours of color video (translating to 99,990 frames) with closed captions from Fox News, CNN, and NBC during the six months leading up to the 2012 and 2016 presidential elections. We apply unsupervised learning, so there is no need to separate our data into training vs. dev datasets. Instead, we designate 3 randomly-selected hours of video as training examples and designate the remaining 27 hours of video as testing examples. We choose this ratio of training-testing examples because our limited computing resources prohibit a larger training dataset.

To discretize our data, we use FFmpeg to extract one frame per second of video. The resulting frames are 640 x 480 pixels, stored together as .png files. We take the frames and run them through our feature extraction pipeline (detailed in methods) which consists of a multi-task cascading convolutional network (Zhang et al., 2016) that detects and aligns faces and a deep convolutional network called FaceNet (Schroff et al., 2015) which takes the aligned faces and generates a 128-byte feature vector for each face. These final feature vectors are used in the clustering algorithms.

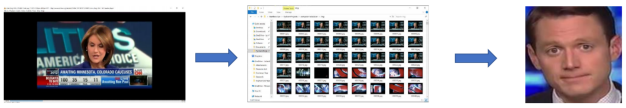

Figure 1. Face Extraction Pipeline

Closed captions are already discretized in our input data, in which each text phrase is mapped to one range of timestamps. We create a new mapping: for each second s in a time range, we map s to the text phrase associated with that entire range. We then extract character bigram frequencies from each phrase, using our own Python scripts with scikit-learn libraries. We further apply term frequency-inverse document frequency (tf-idf) weighting to reweight count features. A high tf-idf value indicates high term frequency and low document frequency in the larger corpus.

4. Pipeline Methods

This study utilized two methods to transform frames into unique feature vectors for the clustering algorithms.

4.1. Multi-task Cascaded Convolutional Network

We used a pre-trained multi-task cascaded convolutional neural network (MTCNN) to detect and align the faces (Zhang et al., 2016). The MTCNN framework uses a cascading structure with three stages of deep convolutional networks that predict face and landmark location. Given an image, it is initially resized to different scales to build an image pyramid, which is given as the input for the three-stage cascaded framework. In stage 1, a convolutional network called Proposal Network (P-Net) obtains the candidate windows and their bounding box regression vectors. After that, estimated bounding box regression vectors are used to calibrate the candidates. Finally, a non-maximum suppression (NMS) is used to merge highly overlapped candidates. In stage 2, all candidates are inputted into a CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and NMS candidate merge. In stage 3, the network describes the face in more details and outputs five facial landmarks positions along with the aligned and cropped image.

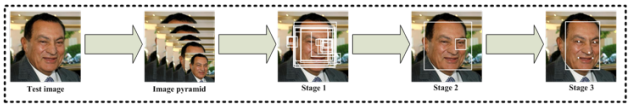

Figure 2. MTCNN Pipeline

4.2. FaceNet Embedding Generation

Taking the aligned images generated by the MTCNN, we then sought to generate feature vectors corresponding to each face where vectors would be close in distance if two faces were similar/same and far apart if two faces were different. To do this, we used FaceNet, a pre-trained convolutional neural network that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity (Schroff et al., 2015). The method uses a deep convolutional network which is trained using triplets of roughly aligned matching / non-matching face patches generated using a triplet mining method. The generated feature vectors were used in all clustering algorithms.

5. Clustering Methods

We experiment with four clustering algorithms: (1) k-means, (2) GMMs, (3) rank-order, (4) DBSCAN, and (5) DBSCAN + k-means.

5.1. K-Means

The k-means clustering algorithm alternates between (1) assigning training examples to the nearest centroid and (2) setting centroids to the average of all assigned examples. Formally,

Repeat until convergence:

For every i, set

$$c^{(i)} := arg\min_j ||x^{(i)} - \mu_j||^2$$

For every j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

Centroids are typically initialized randomly, but we instead use k-means++ (Bahmani et al., 2012) to choose initial centroids.

5.2. Gaussian Mixture Models (GMM)

We model each training example as originating from one of k Gaussian distributions. Formally, we specify the joint distribution $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$, where $z^{(i)} \sim Multinomial(\phi)$ and $x^{(i)}|z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$. This latent random variable $z^{(i)}$ can take one of k values and identifies the Gaussian from which $x^{(i)}$ was drawn.

We use the Expectation-Maximization algorithm to estimate our parameters ϕ, μ_j, Σ_j. The EM algorithm alternates between (E-Step) estimating the values of $z^{(i)}$ and (M-Step) updating the parameters of our model based off those estimates. Formally,

Repeat until convergence:

(E-Step) For each i, j, set

$$w_j^{(i)} := p(z^{(i)} = j \mid x^{(i)}; \phi, \mu, \Sigma)$$

(M-Step) Update the parameters:

$$\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)}$$

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}}$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

We initialize parameters using k-means, as random initialization performed poorly in comparison.

5.3. Rank-Order

Rank-Order is a form of agglomerative hierarchical clustering, meaning all training examples initially represent different clusters and are iteratively merged together. In our specific implementation, we use the following distance metric:

$$D(a, b) = \frac{d(a, b) + d(b, a)}{\min(O_a(b), O_b(a))},$$

where

$$d(a, b) = \sum_{i=1}^{\min(O_a(b), k)} I_b(O_b(f_a(i)), k),$$

$f_a(i)$ is the i-th nearest face to a, $O_b(f_a(i))$ gives the rank of face $f_a(i)$ in face b's neighbor list, and $I_b(x, k)$ is 0 if face x is in face b's top k nearest neighbors and 1 otherwise.

The algorithm then (1) finds the k-nearest neighbors for each face a, (2) computes pairwise distances $D(a, b)$ between a and its top-k neighbors, and (3) transitively merges all (a, b) with distances below a given threshold. It repeats these steps until no further pairwise distances fall below this threshold.

5.4. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

The objective of DBSCAN is to cluster together high-density regions and mark low-density regions as noise. The algorithm takes two parameters: ϵ, a distance parameter, and $minPts$, the minimum number of points required to form a dense region. For each training example x, the algorithm then (1) retrieves all n points within an ϵ radius of x, (2) either adds all n points and their respective clusters to a new cluster if $n \geq minPts$ or marks x as noise if $n < minPts$. Note that a point marked as noise can still be assigned to a future cluster.

As a modification, we initialize each training example as being its own unique assignment. Then instead of labeling a point as noise, we simply leave its original assignment unchanged. This ensures that every face eventually gets assigned to some unique identity. Moreover, we iteratively test out various epsilon values for optimization and choose the epsilon value which yields the highest silhouette coefficient as a parameter (i.e. tuning our hyperparameters).

5.5. DBSCAN + K-Means

In this novel algorithm, we run DBSCAN on top of clusters already produced by k-means. In other words, we initialize each training example using the assignments given by

k-means. We undershoot k when running k-means, then choose an optimal ϵ to break apart too-large clusters. For each individual cluster generated by k-means, we find the optimal epsilon (based on silhouette coefficient) to run DBSCAN on the cluster. Thus, each DBSCAN run is optimized for each individual cluster and we achieve highly refined and distinct clusters. This provides an automated method to approximate the number of identities or clusters (i.e. k) in contrast to k-means.

6. Results and Discussion

We ran all clustering algorithms over the training data and the highest-performing algorithm over the testing data.

We use the silhouette coefficient as our primary evaluation metric, where a higher silhouette coefficient is more preferred. This metric is defined for each sample x as

$$s = \frac{b - a}{\max(a, b)},$$

where a is the mean distance between x and all other points assigned to the same centroid, and b is the mean distance between x and all other points of the next nearest cluster.

6.1. Training Results

For both k-means and GMMs, we used maxIters = 10,000 and defined convergence as a difference in total loss of 0.0001. For every value of k from 0 to 1000 (increments of 50), we perform 10 runs with different centroid seeds and record the best run in terms of inertia. Using FaceNet embeddings as feature vectors, we produce Figure 3.

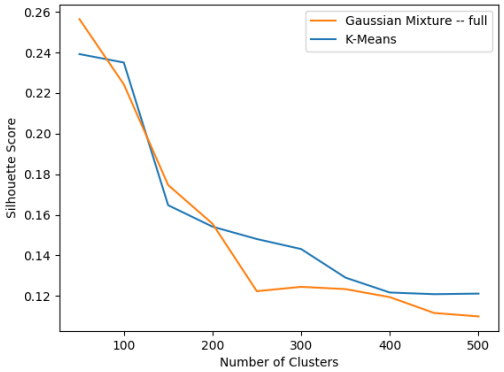

Figure 3. Silhouette Coefficients for k-Means and GMMs with FaceNet Features

A manual count over our training data yielded 411 unique faces. We thus compare results at $k = 411$ and find that k-means yielded higher silhouette coefficients not only at $k = 411$, but in the entire surrounding range of $300 \leq k \leq 500$ (Figure 3). We conclude that k-means performs better on this task, likely indicating that our face images were not well-modeled by Gaussian distributions.

Using the stronger k-means algorithm, we now augment our feature vectors with text features. See Figure 4.

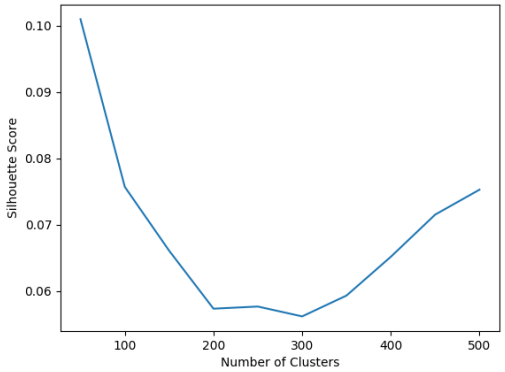

Figure 4. Silhouette Coefficients for k-Means with Combined FaceNet and Text Features

Surprisingly, the addition of text features resulted in much lower silhouette coefficients and higher inertia scores across all k, showing that performance suffered. We believe this is because our specific text features were very sparse and not diverse enough to represent a unique identity. We use only FaceNet embeddings as features for the remainder of our results.

In a large-scale application, such as over our testing dataset, it is impractical to manually count the correct parameter k. This led us to pursue rank-ordering and DBSCAN algorithms, which do not require a given k.

Rank-order does not lend itself well to quantitative evaluation metrics, as there are no centroids. Qualitatively, the algorithm generated one extremely large cluster of 3551 faces and hundreds of clusters with no more than 10 faces. Upon manual observation of the 3551-large cluster, we observed more than 15 unique identities, a sampling of which are shown in Figure 5.

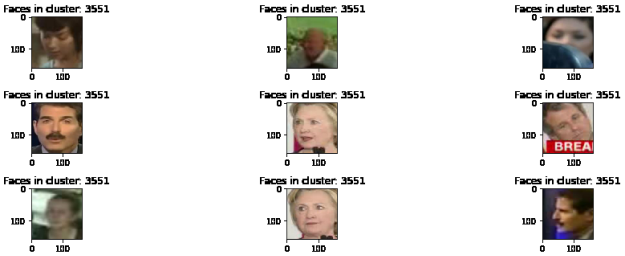

Figure 5. Faces in 3551-Large Cluster Generated by Rank-Order

Results from DBSCAN are shown in Figures 6.1, 6.1. Rather than varying k, we vary ϵ for DBSCAN, testing all values from 0.0 to 1.0 in increments of 0.05. Our highest silhouette coefficient is generated at $\epsilon = 0.6$, while $\epsilon = 0.45$ generates $k = 350$, the closest number of clusters to our true $k = 411$. As opposed to k-means, DBSCAN allows us to find an optimal neighborhood distance (ϵ) which

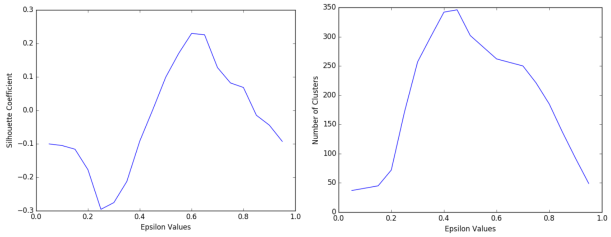

Figure 6. Left: Silhouette Coefficients for DBSCAN; Right: Number of Clusters outputted by same

can approximate the number of clusters. Also, the silhouette coefficients generated by DBSCAN were comparable to those generated by k-means (slightly lower) and manual inspection confirmed the authenticity of the clusters. However, it appears that optimizing for epsilon with DBSCAN caused the algorithm to under-predict the number of clusters and cause some clusters to contain multiple identities.

As a final experiment, we run DBSCAN + k-means to achieve the high performance of k-means without having to specify k by utilizing DBSCAN. Results shown in Figures 6.1, 6.1 demonstrate the significant improvement that this algorithm achieves in silhouette coefficient over just k-means. Manual inspection of the clusters (largest, medium and smallest) confirmed that DBSCAN + k-means was indeed separating the photos by identity.

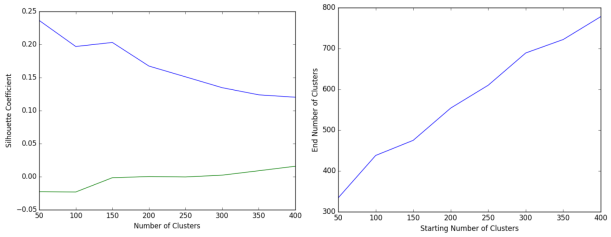

Figure 7. Left: Silhouette Coefficients for DBSCAN + k-Means (Green = k-means, Blue = DBSCAN + k-means); Right: Number of Clusters outputted by same

6.2. Testing Results

We run only our highest-performing algorithm, DBSCAN, on the testing dataset. To quicken the process, we use maxIters = 5,000 but otherwise leave all parameters from our training trials unchanged. Results and sample clusters are shown in Figures 8, 9. By manual inspection, we noted that one cluster achieved 91.3% precision.

7. Conclusions and Future Work

Through our work, we have shown the feasibility of clustering a large collection of unlabeled face images (methods tested with up to 81,000 faces or 30 hours of video) into

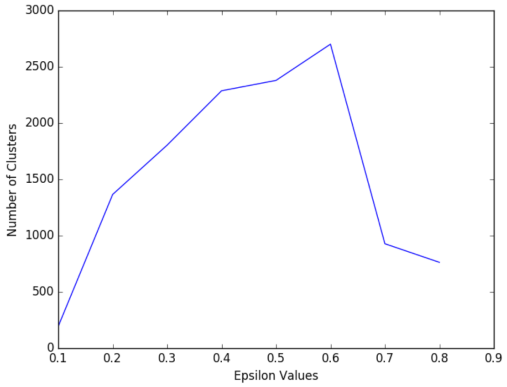

Figure 8. Number of Clusters outputted by DBSCAN over Testing Dataset (81,000 faces)

Figure 9. Sample Clusters Generated by DBSCAN

an unknown number of identities. This problem is critical as a first step in organizing a large collection of unlabeled face images in order to allow researchers to ask relevant and pressing identity-based queries of television and news footage. We acknowledge that clustering accuracy is difficult to measure given large quantities of unlabeled data, but manual inspection revealed high level of precision and recall for DBSCAN. Indeed, multiple clusters reached a level of 100% precision.

The best clusters are generated by the DBSCAN clustering algorithm. Upon manual inspection, the clusters produced by the algorithm corresponded reasonably well to true identities. To evaluate the quality of the clustering, we used silhouette coefficient as a metric of clustering quality as well as manual examination of the largest, median and smallest clusters to determine if the silhouette coefficient accurately indicated cluster quality. Experimental results showed that DBSCAN and k-means were extremely effective for both smaller scale data (10,000 faces) and large-scale data (81,000 data). From both datasets, the algorithms yielded high-quality, homogeneous clusters.

As future work, we would first experiment further with feature selection, perhaps including more detailed representations of the closed captions. We would also investigate temporal-based features to automatically cluster images in adjacent frames. Other areas for further investigation include optimizing automatic selection of the number of clusters (choosing the correct k) by further refining DBSCAN and k-means, especially in concert, as well as improving clustering accuracy by further tuning hyperparameters.

8. Contributions

We believe that all members contributed valuably and equally. In terms of coding, Anirudh implemented k-means and DBSCAN, Cherry implemented GMMs and ran FaceNet feature extractions, and Matthew implemented rank-order and text feature extractions. Together, we also researched, met advisors, designed trials, pre-processed data, talked through implementations, interpreted results, and wrote this paper.

References

Archive, Internet. Tv news [us tv news shows], 2017.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

Cui, J., Wen, F., Xiao, R., Tian, Y., and Tang, X. Easyalbum: an interactive photo annotation system based on face clustering and re-ranking. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 367–376, 2007.

Ho, J., Yang, M.-H., Lim, J., Lee, K.-C., and Kriegman, D. Clustering appearances of objects under varying illumination conditions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Tian, Y., Liu, W., Xiao, R., Wen, F., and Tang, X. A face annotation framework with partial clustering and interactive labeling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

Zha, M., Teo, Y., Liu, S., Chua, T., and Jain, R. Automatic person annotation of family photo album. *Image and Video Retrieval*, 2(3):163–172, 2006.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23 (10):1499–1503, 2016.

Zhu, C., Wen, F., and Sun, J. A rank-order distance based clustering algorithm for face tagging. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 481–488, 2011.