Talking Pictures: Temporal Grouping and Dialog-Supervised Person Recognition

Timothee Cour
INRIA - WILLOW Project
timothee.cour@gmail.com

Benjamin Sapp Akash Nagle Ben Taskar
University of Pennsylvania
{bensapp,akashn,taskar}@seas.upenn.edu

Abstract

We address the character identification problem in movies and television videos: assigning names to faces on the screen. Most prior work on person recognition in video assumes some supervised data such as screenplay or hand-labeled faces. In this paper, our only source of 'supervision' are the dialog cues: first, second and third person references (such as "I'm Jack", "Hey, Jack!" and "Jack left"). While this kind of supervision is sparse and indirect, we exploit multiple modalities and their interactions (appearance, dialog, mouth movement, synchrony, continuity-editing cues) to effectively resolve identities through local temporal grouping followed by global weakly supervised recognition. We propose a novel temporal grouping model that partitions face tracks across multiple shots while respecting appearance, geometric and film-editing cues and constraints. In this model, states represent partitions of the k most recent face tracks, and transitions represent compatibility of consecutive partitions. We present dynamic programming inference and discriminative learning for the model. The individual face tracks are subsequently assigned a name by learning a classifier from partial label constraints. The weakly supervised classifier incorporates multiple-instance constraints from dialog cues as well as soft grouping constraints from our temporal grouping. We evaluate both the temporal grouping and final character naming on several hours of TV and movies.

1. Introduction

We address the problem of learning to name characters in movies and television with minimal supervision. The ability to accurately determine who is on the screen for the vast amounts of unannotated videos opens up the possibility of a number of applications such as large-scale indexing, retrieval and summarization. In several recent papers [5, 8, 2, 3], a screenplay and closed captions are used to name characters by essentially using the knowledge of who is speaking and when to associate names to faces using temporal overlap and mouth motion.

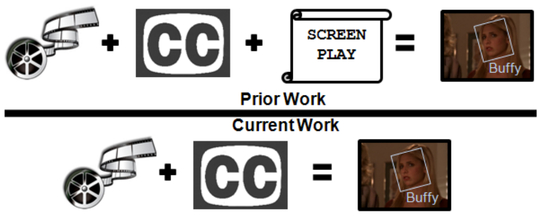

Figure 1. Diagram of prior and current work on character naming. When screenplay is not available, we know *what* is being said but not *who* said it. The only source of name information is from dialog cues: first, second and third person references (See text).

When you watch a movie, consider how you are able to infer characters' names. Without ever being given direct supervision (e.g., a cast list with head shots) or weaker annotation (e.g., a screenplay), you can infer identities based on rare occurrences of first, second and third person references (e.g., "I'm Jack","Hey, Jack", and "Jack left.", respectively), and implicitly apply these identities to other occurrences throughout the movie based on visual appearance and speech (e.g., "the woman with brown hair" or "the man with a raspy voice"). In addition, you may rely on movie structure to aid in resolution: character occurrences on screen often alternate during dialogs, and when a new scene begins, a new set of characters is usually present.

In this work, we do not assume that a screenplay or any special annotation is available, but instead integrate natural dialog and movie structure cues to automatically name the people present in a video. The only input we assume is what is typically available to the movie-goer: the video and subtitles or closed captions[1]. We also use an automatically extracted cast list from IMDB website to extract unique identifiers for characters in the video, although this is not crucial to our approach[2]. Eliminating dependence on screenplays (which are not necessarily available) allows broad application of our method to video collections fully automatically. This contrasts to prior work that requires either manual annotation or name references extracted from a screenplay in order to provide training examples (see figure 1).

[1]The need for subtitles could be avoided with reliable speech recognition. Fortunately, such transcriptions are frequently available.

[2]Name references in dialog are mapped to the closest name in the cast list using string edit distance.

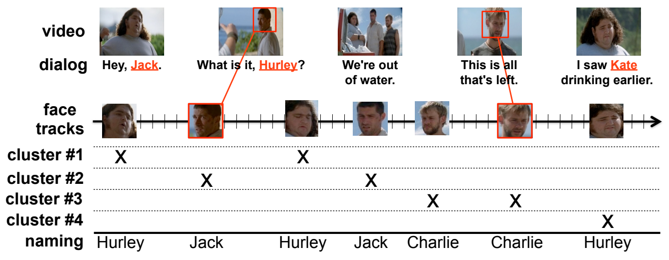

Figure 2. Overview of our approach: we model a movie or TV show as a sequence of face tracks ordered temporally, which we group into a smaller number of clusters. We then combine those grouping cues with gender cues and weak supervision from name references in the dialog to resolve identities globally.

There are several difficult challenges caused by the lack of screenplays. Essentially, we know *what* is being said, but not *who* said it. In addition, first, second, and third person references in the dialog are extremely sparse and often ambiguous. To overcome these challenges, we propose a two phase approach for naming characters. In the first phase, we detect and cluster a sequence of face tracks into a small number of groups. In the second phase, we resolve identities of individual face tracks using 1$^{\text{st}}$, 2$^{\text{nd}}$ and 3$^{\text{rd}}$ person references as weak supervision, as well as gender cues and grouping cues obtained in the first phase, see figure 2.

To help us in our task, we exploit the rich structured information present in video arising from film-editing. Movies and TV series are typically edited according to a set of conventions, aimed at creating the perception of "continuity" across a shot cut (see [12] for an excellent introduction to the topic). A number of these rules can be used as (soft) constraints for grouping people by identity, see table 1. For example, the 180° rule implies that a character will consistently appear on the left (or right) side of the screen throughout a scene whenever two characters are visible. Likewise, two consecutive shots typically show a different character before and after the cut, especially when there are only two people in the scene.

Intra-Scene Cues	**Description**
# actors per scene	Average scene has few characters
180° rule	Relative position of actors stays constant
shot alternation	Consecutive shots show different actors
2 faces per frame	Two faces in a frame are different people

Table 1. Grouping cues within a scene, see text for details.

In this paper, we propose a novel *temporal grouping model* (sections 3 - 5) that groups faces based on not only appearance but also on local film-editing cues. In this model, states represent partitions of the k most recent face tracks, and transitions represent compatibility of consecutive partitions. We present dynamic programming inference and discriminative learning for the model. The face tracks are subsequently assigned a name by *learning a classifier from partial label constraints* derived from dialog (sections 6 - 7). The weakly supervised classifier incorporates multiple-instance constraints from dialog cues as well as grouping constraints from our temporal grouping. We evaluate both the temporal grouping and final character naming on several hours of TV and movies.

To summarize, our main contributions are: (1) the first fully automatic system for character naming in video; it uses dialog in the common case when screenplay is not available, and (2) a temporal grouping model that may be of independent interest, which incorporates arbitrary non-pairwise cues including novel film-editing cues.

2. Related work

Identity resolution. Most prior work on character naming in video assumes additional supervised or weakly supervised data. A screenplay is used in [5, 3] to align faces to dialogues using mouth motion cues, from which we can deduce the identity of the speaker by aligning screenplay and closed captions. In [10], the authors use grouping cues at different time scales to form increasingly large clusters of faces. However, they use groundtruth annotations to assign names to clusters. The authors of [1] automatically cluster images aided by text coming from associated captions. Video and closed-captions are used in [11] to name people in televised news broadcasts via a simple co-occurrence score with extracted labels.

Grouping. Regarding our temporal grouping approach, there is a large body of related work on metric learning and learning to cluster in different applications; a few applicable techniques include [9, 7]. In general, these approaches are limited in that they only model pairwise interactions, rather than arbitrary interactions in a larger local neighborhood. Furthermore, global distance functions are forced to generalize to a wide variety of situations. We believe our approach of learning how to cluster locally is more effective. Related recent work [4] proposes a graphical model for simultaneous partitioning and labeling of graphs with low tree-width, and models labeled partitions. The setting is significantly different, however, relying on pairwise interactions for a binary labeling task.

Face representation. A number of visual cues can be used for grouping faces together. In [10], the authors distinguish cues at different time scales. At the shortest time scale, within a shot, they cluster faces based on tracking. At the medium time scale, within a scene, they use hair, torso and face color histograms. Across episodes, they rely more on facial appearance as the other cues are no longer reliable.

3. Temporal grouping

One of the main challenges for naming individual face tracks is the inherent sparsity of name references in dialog. Our goal in this section is to group face tracks so as to ultimately 1) propagate name references to all faces in a group, and 2) encourage a labeling of individual faces that is consistent with the grouping. Ideally we would like to find a minimal number of groups such that face tracks within a group correspond to a single person. Our character naming classifier learned in section 6 uses the clustering as *soft* constraints, and therefore can recover from grouping errors such as a group containing two distinct characters. By incorporating appearance cues and continuity editing cues into our model, we are able to improve the performance of both the grouping and the final character naming.

We model a movie as a linear sequence of face tracks $\{x_i\}_{i=1}^n$, ordered temporally using the first face in each track (as shown in figure 2, ties are broken based on the position of such faces). As an example, a typical 45-minute TV show contains approximately 500 face tracks in the sequence. Track i is associated with a character name label z_i out of a set of L possible characters in the episode (known *a priori* using a cast list).

A plausible approach to the grouping problem would be to learn a sequence model over the set of labels $\{1..L\}$, for example using a k^{th}-order Conditional Random Field (CRF) to model local interactions within k consecutive face tracks. The number of states at each position during inference would be L^k, and the number of transitions L^{k+1}—prohibitively expensive even for moderate values of k. In practice, we would like to capture long-range interactions within a scene (typically spanning 10 consecutive shots), thus be able to incorporate constraints arising form shot alternation and scene structure. Such approach is an overkill for our grouping task, where we only care about whether two given faces are the same, regardless of their label.

3.1. Local partitions

In our approach, we address this issue by representing *local partitions* (denoted as $\mathcal{P}_k$) instead of local labelings for each k consecutive elements. The number of partitions of k elements is much smaller than the number of possible labelings. It is given by the k^{th} Bell number $B_k = \sum_{i=0}^{k-1} \binom{k-1}{i} B_i$, with $B_1 = 1$ (see table 2). Even with only $L = 4$ labels (compared to 20 characters in a typical TV show), we have $B_k/L^k \leq 0.1$ for any $k \in \{2...10\}$. This allows us to capture rich long range interactions.

k	3	4	5	6	7	8	9	10
B_k	5	15	52	203	877	4140	21147	10^5
4^k	64	256	1024	4096	16384	65536	$3 \cdot 10^5$	10^6

Table 2. B_k: number of partitions of k elements, representing our state space. This is much more compact than a standard sequence labeling state space over a neighborhood of k elements, even with a very small label set ($L = 4$).

We propose to learn a model which enumerates the set of possible partitions for a local window of face tracks and predicts a partition based on appearance and video editing cues and constraints. In particular we are able to capture scene-level interactions such as shot alternation in dialogues or co-occurence of two faces, see section 4.

There are many ways to represent partitions. For $k = 3$, we have 5 possible partitions: using set notation these are $\{(1,2,3),(1,2)(3),(1)(2,3),(1,3)(2),(1)(2)(3)\}$. The order inside and outside the parenthesis is irrelevant, for example $(2,3)(1)$, $(3,2)(1)$, and $(1)(3,2)$ represent the same partition. A more convenient notation is to associate to a partition $y \in \mathcal{Y}$ a matrix $Y \in \{0,1\}^{k \times k}$ such that $Y_{ij} = 1$ if i, j are in the same cluster, which represents an equivalence relationship, a illustrated in figure 3. We will use y and Y interchangeably.

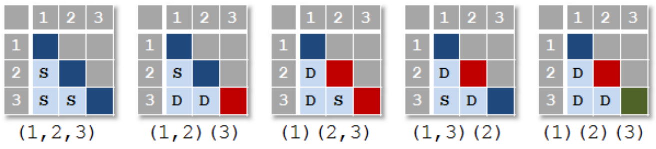

Figure 3. Matrix representation of all 5 partitions of size $k = 3$. Each pair of elements (representing face tracks) can either be same (S) or different (D), yielding different clusterings. Cluster labels are represented by colors along the diagonal.

3.2. Global partitioning inference

The partition classifier we learn in section 3.3 can incorporate rich, *non-additive*, local cues for windows of size up to 10 face tracks in practice. We are interested here in producing a consistent grouping of face tracks across the entire sequence by integrating the predictions of the partition classifier on local windows. We define a sequence-consistent partition to be collection of partitions on overlapping windows that agree on their grouping decisions on the overlap. Figure 4 illustrates this concept.

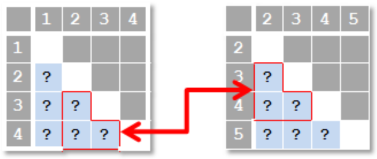

Figure 4. Consistency between partitions of successive windows of size $k = 4$ are enforced. Same/different relationships for elements 2,3,4 must be the same as elements 1,2,3 in the next window.

More formally, we say that y and y' are **consistent**, denoted as $y \sim y'$, if their matrix representations Y, Y' verify:

$$\forall u, v \in \{1..k-1\}, Y_{u+1,v+1} = Y'_{uv} \tag{1}$$

Note, this relation is transitive but not symmetric. A sequence-consistent partition of elements $\mathbf{x} = \{x_1, \dots x_n\}$ is a set of $n - k + 1$ partitions y^i of corresponding sets

$\mathbf{x}^i = \{x_i, \ldots, x_{i+k-1}\}$, with consistency between consecutive partitions: $\forall i, y^i \sim y^{i+1}$. Given a scoring function on local partitions $g^{y^i}(\mathbf{x}^i)$ (learned in section 3.3), we seek the optimal sequence-consistent partition:

$$y^* = \underset{y^1 \sim y^2 \ldots \sim y^{n-k+1}}{\arg\max} \sum_i g^{y^i}(\mathbf{x}^i).$$

y^* can be found via a standard Viterbi-like dynamic program whose running time is $O(n \cdot B_{k+1})$, as it is easily shown that the number of consistent consecutive partitions is $\leq B_{k+1}$ (see **supplement** for details). Note that y itself defines a valid partition on the set $x_1, \ldots, x_n$, as a consequence of the running-intersection property.

Cardinality-constrained partition. The dynamic program defined above finds a single solution. In practice it is useful to have control over the number of clusters in the partition, which allows us to compare two different partitioning algorithms (or sets of parameters) for a fixed number of clusters. We extend our framework to compute the optimal C-way partition, for any $C \in \{1..n\}$:

$$y_C^* = \underset{y^1 \sim y^2 \ldots \sim y^{n-k+1}}{\arg\max} \sum_i g^{y^i}(\mathbf{x}^i) \quad \text{s.t.} \quad |y| = C \quad (2)$$

where $|y|$ denotes the cardinality of partition y, *i.e.* its number of clusters. To this end we augment the state-space in the dynamic program to (y, c), where $c \in \{1..n\}$ represents the number of clusters seen so far. We define (y, c) and (y', c') to be consistent if the following holds:

$$(y, c) \sim (y', c') \quad \text{if} \begin{cases} y \sim y' \\ c' - c = \mathbf{1}(\sum_{u=1}^{k-1} Y_{uk} = 0) \end{cases}$$

The second condition states that $c' = c$ when y' groups its last element k with at least one of elements $\{1..k-1\}$, and $c' = c + 1$ if the last element falls in its own cluster, thereby incrementing the total count of clusters seen so far. We compute the optimal C-way partition with dynamic programming, using the augmented states (y, c) and compatible transitions. A single dynamic programming table can be used to retrieve optimal C-way partitions for *all* values of C. Each one requires a decoding pass, starting from the best scoring state (y^*, C^*) in the last table column that satisfies $C^* = C$.

3.3. Learning to partition

We cast the problem of learning to partition as a structured multiclass classification problem. The input is a set of instances $\mathbf{x} = \{x_1, \ldots, x_k\}$ and the output is a partition $y \in \mathcal{Y} = \mathcal{P}_k$. The number of classes $|\mathcal{P}_k| = B_k$ is large but structured. We assume that each input output pair is described by a d-dimensional feature mapping $\mathbf{f}(\mathbf{x}, y) \in R^d$. The features we use are based on pairwise relations between items in the partition, such as the maximal RGB distance between images grouped together, or global, such as the number of clusters in the partition, see section 4. We assume a linear classifier of the form

$$g^*(\mathbf{x}) = \arg\max_y g^y(\mathbf{x}), \quad \text{where} \quad g^y(\mathbf{x}) \quad = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y)$$

where $\mathbf{w} \in R^d$ is the set of parameters which we will estimate from a set of n labeled partitions $\{(\mathbf{x}^i, y^i)\}$. The corresponding loss function is

$$\ell(g) = \sum_i \ell^i(g) \quad \text{with} \quad \ell^i(g) = \ell(y^i, g^*(\mathbf{x}^i)), \quad (3)$$

where the individual loss $\ell(y^i, y) : \mathcal{Y} \times \mathcal{Y} \mapsto R^+$ might depend on the structure of the partition. For example, if the true partition is $\{(1, 2)(3, 4)\}$, then a predicted partition $\{(1, 2)(3)(4)\}$ might be penalized less than $\{(1)(2, 3)(4)\}$ since it is closer to the truth. There are several natural metrics for measuring the quality of a predicted partition with respect to the true partition. For example, precision and recall with respect to same/different relationships is often used. Another metric involves mapping true clusters to predicted clusters by a majority vote in each cluster, and computing the number of mis-matches. We choose the following expression, which allows to penalize differently splits and merges compared against groundtruth:

$$\begin{aligned} \ell(y^i, y) =& \alpha_{merge} \sum_{u<v} \mathbf{1}(Y_{uv}^i = 0 \text{ and } Y_{uv} = 1) \\ &+ (1 - \alpha_{merge}) \sum_{u<v} \mathbf{1}(Y_{uv}^i = 1 \text{ and } Y_{uv} = 0) \end{aligned}$$

where the binary matrices Y, Y^i refer to partitions y, y^i and $\mathbf{1}(\cdot)$ is the indicator function. We set $\alpha_{merge} = 0.1$ in our experiments based on validation data.

A convex learning formulation. We propose to estimate $\mathbf{w}$ by minimizing a convex upperbound on the loss function (3). Let $\psi(z)$ be a standard convex binary loss such as exponential, logistic or hinge, which is decreasing and upperbounds the step function $\mathbf{1}(z \leq 0)$. We define a convex loss function as $\mathcal{L}(g) = \sum_i \mathcal{L}^i(g)$, where

$$\mathcal{L}^i(g) = \sum_{y \in \mathcal{Y}} \ell(y^i, y) \psi\left(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)\right).$$

The following proposition (proved in the **supplement**) states that $\mathcal{L}(g)$ is a (convex) upperbound on $\ell(g)$.

Proposition 3.1 $\ell^i(g) \leq \mathcal{L}^i(g)$.

Optimization with boosting. In our experiments, we use the exponential loss $\psi(z) = \exp(-z)$ and minimize the objective with boosting using decision stumps as weak classifiers, minimizing a second order Taylor expansion along the optimal coordinate at each boosting round. We show in the **supplement** how to select the optimal stump across all feature dimensions and thresholds in only $O(d \cdot n \cdot B_k)$ per boosting round, which is not obvious.

4. Grouping cues

Our representation can encode any type of local cues or constraints defined in a neighborhood of size $\leq k$, and can depend on both the input and a proposed labeling. We define base features by computing several types of pairwise distances $d(x_i, x_j)$ for each pair of face tracks x_i, x_j in a window of size k, and computing statistics over those. We condition on the proposed labeling, distinguishing whether the pair of face tracks lies in the same cluster ($Y_{ij} = 1$):

$$f_{\text{same}}(x, y) = s(\{d(x_i, x_j)\}_{i,j:Y_{ij}=1}) \quad (4)$$
$$f_{\text{different}}(x, y) = s(\{d(x_i, x_j)\}_{i,j:Y_{ij}=0}) \quad (5)$$

where $s(\cdot)$ aggregates statistics over its inputs: we compute the mean, the sum, the max, and the min. The latter two statistics are non-additive and thus cannot be reduced to a sum of pairwise interactions. We present below the different types of **distance functions** $d(x_i, x_j)$ used, based on standard appearance cues and novel video editing cues. These are tuned automatically using section 3.3. We could define many other features on y or $(x, \mathbf{y})$ but decided to keep our feature set relatively simple to compare to previous work.

4.1. Appearance cues

Best registered face. In a preprocessing stage, each face is registered using 4 control points (eyes, nose and mouth), using a method similar to [5]. For each face track we take the best face as determined by registration score of the 4 control points, and represent it in the following color spaces: RGB, LAB, and the separate channels R,G,B,L,A,B. We use L_1 norm as a distance for these color spaces f_c, $d_{registered}(x_i, x_j) = ||f_c(x_i) - f_c(x_j)||_1$.
Color histograms. Following [10], we also use color histograms for face, hair, torso. Figure 5 displays a set of frames from the TV show Buffy the Vampire Slayer with face detections and torso/hair rectangles we use to compute the color histograms. We define $d_{hair}(\cdot,\cdot)$, $d_{torso}(\cdot,\cdot)$ and $d_{face}(\cdot,\cdot)$ using χ^2-distance.
Exemplars. Again following [10], we also represent each track with 5 exemplars using Principal Components Analysis (PCA)—estimate PCA components from all our data and project our examples onto the first 50 components. A difference between our approach and [10] is that we first register faces. The distance between 2 tracks is the minimum distance between all pairs of exemplars across the tracks: $d_{exemplar}(x_i, x_j) = \min_{kl} ||e_i^k - e_j^l||_2$, where e_i^k is exemplar k for track i (likewise for e_j^l).

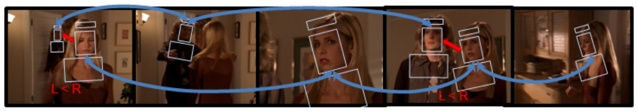

Figure 5. Example frames from the TV show *Buffy*. Face/hair/torso rectangles are shown for appearance cues. Red arrows mark do-not-group cues between face tracks appearing in the same frame. Blue arrows and "L < R" labels indicate connections between face tracks resulting from the 180° rule which dictates that faces remain in the same left to right ordering in a scene and more weakly, they often remain in the same horizontal portion of the screen.

Gender. We trained a gender classifier by collecting 200,000 images via Google Image search, querying for common male and female names. These were registered as before and used to train a gentleboost classifier with Haar features, whose accuracy on a hold-out set was 83% (**code** will be made available). For each track we compute the resulting mean, max, and median gender score over the track, before defining distances $d_{gender}(\cdot,\cdot)$ between tracks.

4.2. Video editing cues

A key contribution of our model is the ability to encode interesting local editing cues. We define the following additional distances based on such cues, illustrated in figure 5.
Relative positioning of faces. Based on the 180° rule, we compute distances and signed distances based on the x and y coordinates of the mean face position in each track, $d_{position}(\cdot,\cdot)$. We also compute a corresponding distance in log-scale for the mean face size in a track.
Relative difference in pose. People tend to look in opposite directions in typical dialogue scenes. Hence, for each facial part of the best-registered face, we compute relative distances in x and y coordinates (after registration), providing information about the difference in pose.
Shot distance. Shot alternation is a strong cue for grouping. We provide 2 features based on shot distance. One is absolute difference in shot id, determined beforehand using a conservative shot-segmenter. The second is L_1 distance in color histograms in LAB space for a whole frame, accumulated over every frame in each face track. Eight bins per channel (8^3 total bins) were used.
Track overlap. Another simple but effective cue is whether two tracks ever overlap in time (*i.e.* there is a frame for which faces from both tracks appear at the same time). These tracks should clearly not be merged.

5. Results for temporal grouping

We train our temporal grouping model using The Office (US) Season 2 Episode 5, a 1.5 hour episode. This provides us with 1273 (overlapping) windows of 7 consecutive face tracks for training data. On a validation episode, we observe that clustering accuracy increases with the window size, and choose $k = 7$ for computational reasons (see figure 6, right). We evaluate our temporal grouping on 6 episodes of the TV show Lost, 1 episode of Buffy the Vampire Slayer and the movie Misery. We compare to the following baselines:

Baseline 1 is a global agglomerative method proposed by Ramanan et al. [10]. Briefly, this uses the same features as we employ, each associated with its own distance function (d_{hair}, d_{torso}, d_{face} and $d_{exemplar}$ described in Section 4). Logistic regression is used to learn the best weighting of these distances, and we feed the weighted distance function into an agglomerative clustering algorithm.

Baseline 2 is a simple agglomerative method: each track i is modeled with one representative face exemplar in LAB color-space ℓ_i, and we use an L_2-norm distance function between 2 tracks i and j: $d(i,j) = ||\ell_i - \ell_j||_2$.

In all results, we measure performance as a tradeoff between the number of clusters (x-axis) versus the purity of the clustering (y-axis). Purity for a clustering is defined as prediction accuracy of names assigned by using the most frequent true name in a cluster.

In figure 7 we show our performance against the 2 baselines. There is an inherent tradeoff between global methods (such as the baselines) and our method. While we can model rich local interactions of a label sequence, it is difficult to merge things that are temporally distant, even if they are perceptually very similar. Global clustering, on the other hand, is unconstrained by temporal distance but cannot perform as well when local clustering is challenging. Our method performs best in a high precision regime in which the number of clusters is relatively high, and the precision is close to perfect. This type of behavior will aid in the naming task to come.

We also analyze the efficacy of features via ablative analysis in figure 6. Starting with our full feature set, we sequentially remove sets of features and note the decay in performance. The features in [10] clearly help, but are significantly improved by adding the gender and editing cues.

6. Dialog-supervised person recognition

We now go back to our original motivation of naming characters using references in dialogue as only source of weak supervision. Our task is to predict the label $z \in \{1..L\}$ corresponding to the name of a character in a face track x. In order to simplify our evaluation, we assume the label set known using automatically extracted cast list from IMDB website. Note, we only use the list of names and their associated gender from this website. We build our formulation on a simple and general one-against-all multiclass scheme of the form:

$$z = \arg\max_a g^a(x), \tag{6}$$

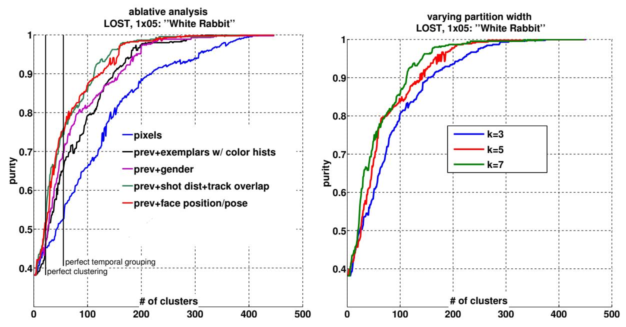

Figure 6. Clustering purity (y-axis) vs number of clusters (x-axis). **Left:** ablative analysis of the features in our model, on Lost Season 1 Episode 5. The biggest gains are obtained from adding gender and face location and scale. **Right:** performance as we vary the local partition window size k. We chose $k = 7$ in our experiments.

where $g^a(x) = \mathbf{w}^a \cdot \mathbf{f}(x)$ is a linear function parameterized by weight vector $\mathbf{w}^a \in \Re^d$ for each class label $a \in \{1..L\}$, $\mathbf{f}(x) \in \Re^d$ is a feature vector for input face track x, described in section 7. *If we had* access to labeled examples (x, z), with z^a a {+1,-1} indicator of the class label, one common solution would be to encourage $z^a g^a(x) > 0$ by minimizing the following convex upperbound on the corresponding 01-loss (where $\psi(\cdot)$ denotes some convex loss):

$$\mathcal{L}_{\text{supervised}}(g) = \sum_{i,a} \psi(z_i^a g^a(x_i)) \tag{7}$$

In our partial supervision setting however, we have **no labeled examples** and therefore *cannot* rely on such supervised learning schemes. Instead we only have *constraints* on the possible labels for each face track, summarized in table 3. The constraints are of 3 kinds: grouping constraints, 1st and 2nd reference constraints, and label exclusion constraints from 3rd person references and gender predictions.

A) Grouping. This constraint has the form: $z_i = z_j$ for consecutive face tracks x_i, x_j that are in a same cluster, as found by our previous temporal grouping[3].

B) 1st and 2nd reference. If we observe the utterance "Hey Jack" (classified as 2nd reference, see section 7) at time t, we assume that some face track x_i in the temporal neighborhood of t will have label $z = Jack$[4]. The neighborhood is defined as all clusters of face tracks which contain a face within 10 seconds of time t (converted to frame number). This is a type of multiple instance (MI) constraint of the form $\exists i \in S : z_i = a$ for a label a and a set of face tracks S associated to a reference.

C) Exclusion. This is a negative label constraint of the form: $z_i \neq a$, and comes from either a 3rd person reference or from gender. Firstly, for an utterance classified as

[3]The number of must-link constraints is $n - C$, where $n = \#$face tracks, $C = \#$ clusters which we set to $C = 0.3 \cdot n$ using section 3.2

[4]One could refine this rule using mouth motion cue[5], at the cost of introducing additional errors

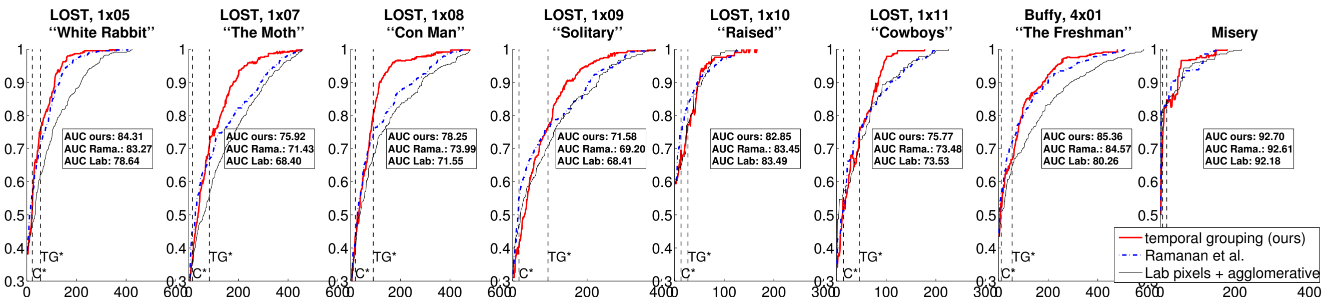

Figure 7. Grouping results versus global methods. $C^\star$ refers to the number of correct labels. $TG^\star$ denotes the optimal number of clusters temporal grouping can achieve via inference with a scope of size $k = 7$. AUC denotes the area under each curve from $TG^\star$ to the total number of face tracks, normalized so that the area for each episode sums to 1.

3rd person reference such as "Jack left" at time t, we assume that neighboring face tracks *cannot* have the label a corresponding to Jack. Secondly, if the average score of a face track x_i according to our gender classifier exceeds a threshold θ_M (M for Male), we add a label constraint $z_i \neq a$ for each label a corresponding to a female (F) name, and a similar rule applies in the opposite case with some $\theta_F < \theta_M$[5].

We encode these label constraints with a unified framework, based on combining convex losses on linear combinations of the score $g^a(x)$. Our combined loss function is:

$$\mathcal{L}(g) = \sum_{i,a} \mathcal{L}^{i,a}_{\text{excl.}}(g) + \sum_{i,j} \mathcal{L}^{ij}_{\text{link}}(g) + \sum_{S,a} \mathcal{L}^{S,a}_{\text{MI}}(g)$$

$$\mathcal{L}^{i,a}_{\text{excl.}}(g) = \psi(-g^a(x_i)) \tag{8}$$

$$\mathcal{L}^{ij}_{\text{link}}(g) = \sum_{a \in \{1..L\}} \bar{\psi}(g^a(x_i) - g^a(x_j)) \tag{9}$$

$$\mathcal{L}^{S,a}_{\text{MI}}(g) = \psi\Big(\frac{1}{|S|} \sum_{i \in S} g^a(x_i)\Big), \tag{10}$$

where each term expresses a constraint: (8) for $z_i \neq a$, (9) for $z_i = z_j$, and (10) for $\exists i \in S : z_i = a$. We use $\bar{\psi}(u) = \psi(u) + \psi(-u)$ for some convex loss $\psi(\cdot)$. In our experiments, we use the square hinge loss, which is differentiable. Note, (8) and (9) are standard formulations[13], but we believe (10) is a **novel way** to express a multiple instance constraint in a convex formulation. Intuitively, this term encourages *at least one* of $g^a(x_i)$ ($i \in S$) to be positive while allowing for the others being negative. In contrast, the simpler term $\sum_{i \in S} \psi(g^a(x_i))$ would encourage *all* of $g^a(x_i)$ ($i \in S$) to be positive.

Optimization. We convert minimization of our convex loss $\mathcal{L}(g)$ into a standard L_2-loss binary Support Vector Machine, which we solve using liblinear [6]. In our naming experiments, running time is about 30 seconds for 3,000 face tracks, 300 features, and 55 labels (names from the cast list that are mentioned by *at least one* dialogue reference).

[5] We set θ_F and θ_M to achieve 90% precision on a validation set

constraint	example	#	constraint
screenplay line	JACK: "Hey".	750	$z_i = a$
1st person ref	"I'm Jack"	20	$\exists i \in S : z_i = a$
2nd person ref	"Hey, Jack"	60	$\exists i \in S : z_i = a$
3rd person ref	"Jack left"	30	$z_i \neq a$
gender(M/F)	M score$> \theta_M$	400	$z_i \neq a, \forall a \in F$
grouping	track clusters	400	$z_i = z_j$

Table 3. Without screenplay supervision (top), we use constraints on possible labels for our weakly supervised character naming based on dialogue, gender and grouping cues. We show their occurrences per episode averaged over 16 episodes of Lost, and the resulting type of constraint. a is a label (e.g. Jack), z_i the predicted label of a face track, S a set of face tracks in the temporal neighborhood of the reference.

7. Character naming results

We run our grouping and naming system on 8 episodes of the TV show Lost. MI constraints described in the previous section are propagated throughout clusters. Thus, for any tuple of face tracks with a MI constraint (for example, the constraint "$z_i = a \vee z_j = a$" for a pair i, j and label a), we also add other tuples of face tracks from the corresponding clusters (in our example, $z_{i'} = a \vee z_{j'} = a$ with (i, i') in one cluster and (j, j') in another cluster), subsampling at most 200 out of all possible combinations. The final decision for each face track x_i in some cluster S is determined using majority vote ($\arg\max_a \sum_{j \in S} g^a(x_j)$). We fix the number of clusters output by our grouping algorithm to 30% of the total number of face tracks in each episode, using section 3.2. We extract references by matching words in the subtitle to the cast list, and determine reference type (1st, 2nd, 3rd) via a discriminative classifier, described in the supplement.
Features. We use the following features $\mathbf{f}(x)$ for character naming: a face track x is described by its best face (c.f. registration score), using 100 PCA components for the whole

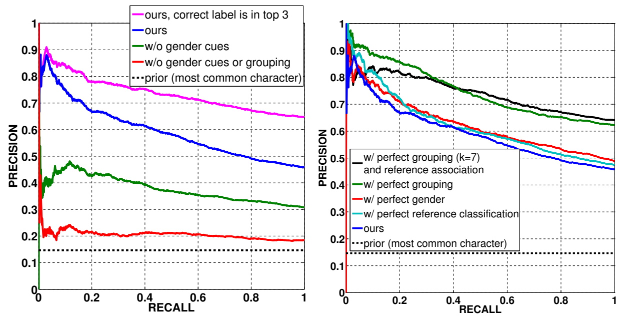

Figure 8. **Left:** Results for naming across 8 episodes of Lost, containing a total of 55 possible labels. "prior" is the most common character. "ours" uses gender, grouping, and reference constraints propagated through grouping. **Right:** Each curve is obtained by replacing *only* the mentioned component with its perfect counterpart: "perfect temporal grouping" uses the best clustering possible with $k = 7$. "perfect gender" and "perfect reference classification" use groundtruth information. "perfect temporal grouping and reference association" applies each reference to its closest correct neighbor, and adds exclusion constraints for negative neighbors.

pre-registered face, and 50 PCA components per part (eyes, nose and mouth) using 15×15 patches.

Evaluation. We measure performance of our system in a "refusal to predict" scheme inspired by [5]. At a given confidence threshold, we measure how accurately the system names characters in examples above the threshold (precision) versus how many examples pass the threshold (recall). Confidence for an example is measured as the difference between the best and second best scores over all classifiers g^a. Results are shown in figure 8, left, evaluated on the 10 most frequent characters in 8 episodes of Lost.

We see that the cues provided by grouping, references, and gender help tremendously in the performance of our system. For the top 10% (resp. 50%) most confident scores, our accuracy is 80% (resp. 60%), compared to the baseline of 15% described in figure 8. Our results are quite encouraging and suitable for retrieval applications, which only require high precision at low recall. Given the wide applicability of our method, this could be used for large-scale content-based video search. Those results obviously are not as good as the ones reported in [5, 3], since we are not using a screenplay. In our case, supervision provided by dialog is *much* lower and noisier, see table 3.

In figure 8, right, we also compare the quality of our naming system by replacing various components with their equivalent "perfect" counterparts: assuming perfect clustering (TG^* in figure 7), gender classification, reference classification or association of references to face tracks.

8. Conclusion

To our knowledge, this work is first to address the problem of character identification in video *without* the use of a screenplay, which opens the possibility of name-based retrieval in general video collections without human intervention. We propose a novel temporal grouping model to group face tracks locally with high precision and we then learn to identify characters using weak supervision provided by cues from references in the dialog, gender, and grouping. We present a novel way to integrate these cues as multiple instance constraints in a convex formulation. Our end-to-end system provides the first quantative results and analysis on naming without screenplay. We are releasing[6] code and data for gender classification, inference/learning in the partition model and for the optimization problem in section 6.

Acknowledgements

This work was supported by NSF grant 0803538. The authors would like to thank Jean Ponce for helpful comments.

References

[1] T. Berg, A. Berg, J.Edwards, M.Maire, R.White, Y. Teh, E. Learned-Miller, and D. Forsyth. Names and faces in the news. In *CVPR*, pages 848–854, 2004.

[2] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *ECCV*, 2008.

[3] T. Cour, B. Sapp, C. Jordan, and B. Taskar. Learning from Ambiguously Labeled Images. In *CVPR*, 2009.

[4] P. J. Cowans and M. Szummer. A graphical model for simultaneous partitioning and labeling. In *AISTATS*, 2005.

[5] M. Everingham, J. Sivic, and A. Zisserman. Hello! My name is... Buffy – automatic naming of characters in tv video. In *BMVC*, 2006.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[7] A. Ferencz, E. Learned-Miller, and J. Malik. Learning to locate informative features for visual identification. *IJCV*, 2006.

[8] A. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. *Lecture Notes In Computer Science*, pages 304–320, 2002.

[9] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007.

[10] D. Ramanan, S. Baker, and S. Kakade. Leveraging archival video for building face datasets. In *ICCV*, 2007.

[11] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in news videos. *IEEE MultiMedia 1999*.

[12] T. J. Smith. *An Attentional Theory of Continuity Editing*. PhD thesis, University of Edinburgh, 2005.

[13] R. Yan, J. Zhang, J. Yang, and A. Hauptmann. A discriminative learning framework with pairwise constraints for video object classification. In *CVPR*, pages 284–291, 2004.

[6]`http://www.vision.grasp.upenn.edu/video/`