Talking Pictures: Temporal Grouping and Dialog-Supervised Person Recognition
Supplemental material

This supplement presents:

- Additional details on the Viterbi decoding.
- Proof of proposition 3.1.
- Description of the reference classifier.
- Additional qualitative results for temporal grouping.
- Proof of linear time complexity for stump selection.

1. Viterbi algorithm implementation details

We explain how to implement the dynamic programs (DP) on lines 315 and 332. The optimal score $s^j(y^j)$ of a sequence-consistent partition up to j, ending with y^j can be computed with the following recursion:

$$
\begin{aligned}
s^j(y^j) := & \max_{y^1..y^{j-1}:y^1\sim y^2...\sim y^j} \sum_{i=1..j} g^{y^i}(\mathbf{x}^i) \\
= & \max_{y^{j-1}:y^{j-1}\sim y^j} s^{j-1}(y^{j-1}) + g^{y^j}(\mathbf{x}^j), \quad \forall j > 1
\end{aligned}
$$

with $s^1(y^1) = g^{y^1}(\mathbf{x}^1)$. The DP table is computed in $O(n \cdot B_{k+1})$, with B_k states and $\leq B_{k+1}$ transitions per column. The backward pass takes $O(n)$ time.
For the cardinality constrained partition, we use the augmented states and transitions described in the paper and initialize the DP table with $s^1(y^1, c) = g^{y^1}(\mathbf{x}^1)$ for $c = |y^1|$ and $-\infty$ otherwise. The DP table is computed in $O(n^2 \cdot B_{k+1})$, with $n \cdot B_k$ states and $\leq n \cdot B_{k+1}$ transitions per column. The backward pass takes $O(n)$ time *per requested partition size*.

2. Proof of proposition 3.1

Given that $\ell(y^i, y) \geq 0$ and $\psi(z) \geq \mathbf{1}(z \leq 0)$,

$$
\begin{aligned}
\ell^i(g) &= \ell(y^i, \arg\max_y g^y(\mathbf{x}^i)) \\
&\leq \sum_{y\in\mathcal{Y}} \ell(y^i, y)\mathbf{1}(g^{y^i}(\mathbf{x}^i) \leq g^y(\mathbf{x}^i)) \\
&\leq \sum_{y\in\mathcal{Y}} \ell(y^i, y)\psi(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i))) \quad \square
\end{aligned}
$$

3. Reference classification in dialogue

We trained a classifier to determine whether a given reference (*i.e.* mention of a character name) corresponds to a 1st, 2nd or 3rd person reference, or none of the above. We matched proper nouns in the subtitles to the cast list and extract features centered around the name occurrence. For training the classifier we use 500 downloaded screenplays from TV shows and movies (excluding the ones we test on) as follows: we automatically label groundtruth by aligning subtitles with the screenplay, which indicates who is speaking, and who else is present in the scene, from which we can infer the type of reference. For features, we calculated the 1,000 most frequent unigrams/bigrams in the vicinity of 2 tokens from the reference, including punctuation. We used 6,000 features, each corresponding to an occurrence of a unigram or bigram in a particular position w.r.t. the reference. We also hand-labeled 300 examples to validate our system and achieved 79.3% accuracy.

4. Additional qualitative results

Figure 1 shows the tracks extracted from a *test* episode, Lost Season 1 episode 5 (reads columnwise). There are typically very few false positive face tracks (just one track in this episode, appearing in its own cluster as cluster #3 in figure 2).

Figure 2 shows the result of our temporal grouping on this episode. The main failure modes are for poorly lit scenes, where appearance cues are less reliable. Face registration errors also lead to errors in grouping, for example cluster 54 in figure 2 should be merged with the previous face track which is ill-registered. In our model, two consecutive elements i, j in a cluster are within a distance at most $k - 1$ (if their distance was $\geq k$, linking them would have no impact on the score function defined on consecutive windows of size k, so we don't by convention). So our model can potentially link face tracks (of, say, Jack) that are very far apart, so long as *at least* one face track of Jack appears every $k - 1$ track in between them. For long dialog scenes or scenes involving few but recurring people (less than k), our model will perform best as tracks of a given person can be merged across long periods of time. And if tracks do get split apart over time, the subsequent name assignment can still merge them based on the learned face appearance model.

5. Proof of linear complexity for stump selection

We show the complexity mentioned on line 422, *i.e.* that at each round of boosting we can select the optimal decision stump across all feature dimensions and thresholds in **linear time** $O(d \cdot n \cdot B_k)$, and in fact, requires a *single pass* over the data[1]. In comparison a naive approach would require $O(d \cdot (n \cdot B_k)^2)$ time (in the case of standard boosting, a common trick based on sorting would bring it to $O(d \cdot n \cdot B_k \log(n \cdot B_k))$, but our non-standard boosting form requires more care; and we even bring it down to linear). We adapt gentle Adaboost [1] (Algorithm 4, page 353) to our setting: at each boosting round we update the linear classifier $g^y(\mathbf{x})$ with the weak learner $h^y(\mathbf{x})$ that minimizes the second order Taylor expansion of $\mathcal{L}(g)$, leading to:

$$\min_h E_{i,y}[h^{y^i}(\mathbf{x}^i) - h^y(\mathbf{x}^i) - 1]^2, \tag{1}$$

where $E_{i,y}[\cdot]$ denotes expectation over $(i, y) \in \{1..n\} \times \mathcal{Y}$ weighted by $\ell(y^i, y)\psi\left(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)\right)$. An apparent difficulty compared to [1] is the **coupling introduced by the difference** term $h(\mathbf{x}^i, y^i) - h(\mathbf{x}^i, y)$, which introduces a non-linear cross-term in equation (3) below.

Using decision stumps of the form $h(\mathbf{x}, y) = a \cdot \mathbf{1}(f_j(\mathbf{x}, y) \leq \theta) + b \cdot \mathbf{1}(f_j(\mathbf{x}, y) > \theta)$ we seek the best feature dimension j, threshold θ, and parameters a, b. Notice we can set $b = 0$ since adding a constant term to $h(\mathbf{x}, y)$ doesn't change the objective. For a fixed j this leads to:

$$\min_{a,\theta} E\left[a(\bar{F}^\theta - F^\theta) - 1\right]^2 = \min_\theta 1 - \frac{E^2[\bar{F}^\theta - F^\theta]}{E[\bar{F}^\theta - F^\theta]^2} \tag{2}$$

with $F^\theta_{i,y} = \mathbf{1}(f_j(\mathbf{x}^i, y) \leq \theta)$, $\bar{F}^\theta_{i,y} = \mathbf{1}(f_j(\mathbf{x}^i, y^i) \leq \theta)$.

The ratio can be computed **without reevaluating** the expectations for all values of θ using that $\bar{F}^\theta, F^\theta$ are binary, and the **trick** $\bar{\bar{F}}^\theta_{i,y} = \mathbf{1}(\max(f_j(\mathbf{x}^i, y^i), f_j(\mathbf{x}^i, y)) \leq \theta)$:

$$\begin{aligned} E[\bar{F}^\theta - F^\theta]^2 &= E[\bar{F}^\theta]^2 + E[\bar{F}^\theta]^2 - 2E[\bar{F}^\theta F^\theta] & (3)\\ &= E[\bar{F}^\theta] + E[\bar{F}^\theta] - 2E[\bar{\bar{F}}^\theta], & (4)\end{aligned}$$

Now that we have linearized the denominator (the numerator is easier), the final **trick** for computing the optimal θ efficiently is to use a form of bucket sort, discretizing θ into a set of regularly spaced discrete values $\theta_1 ... \theta_T$ (with $T < nB_k$). In a single pass over the data of size $n \cdot B_k$ per feature dimension j, we compute the weighted counts $E_{i,y}[\mathbf{1}(f_j(\mathbf{x}^i, y) \in [\theta_t, \theta_{t+1}))]$ and similar terms, and then we compute in $O(T)$ the corresponding cumulative sums. From this we can deduce the optimal (discretized) θ from equation (2) in $O(T)$. The overall running time is just $O(d \cdot n \cdot B_k)$, a huge speedup over the naive approach. □

[1] Our proof in particular applies with little modification to the easier setting of standard boosting with d feature dimensions and n training samples: we replace a $O(dn \log n)$ complexity with a $O(dn)$ complexity, at the expense of discretizing the possible thresholds over which we optimize.

References

[1] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Statist.*, 28:337–407, 2000. 2

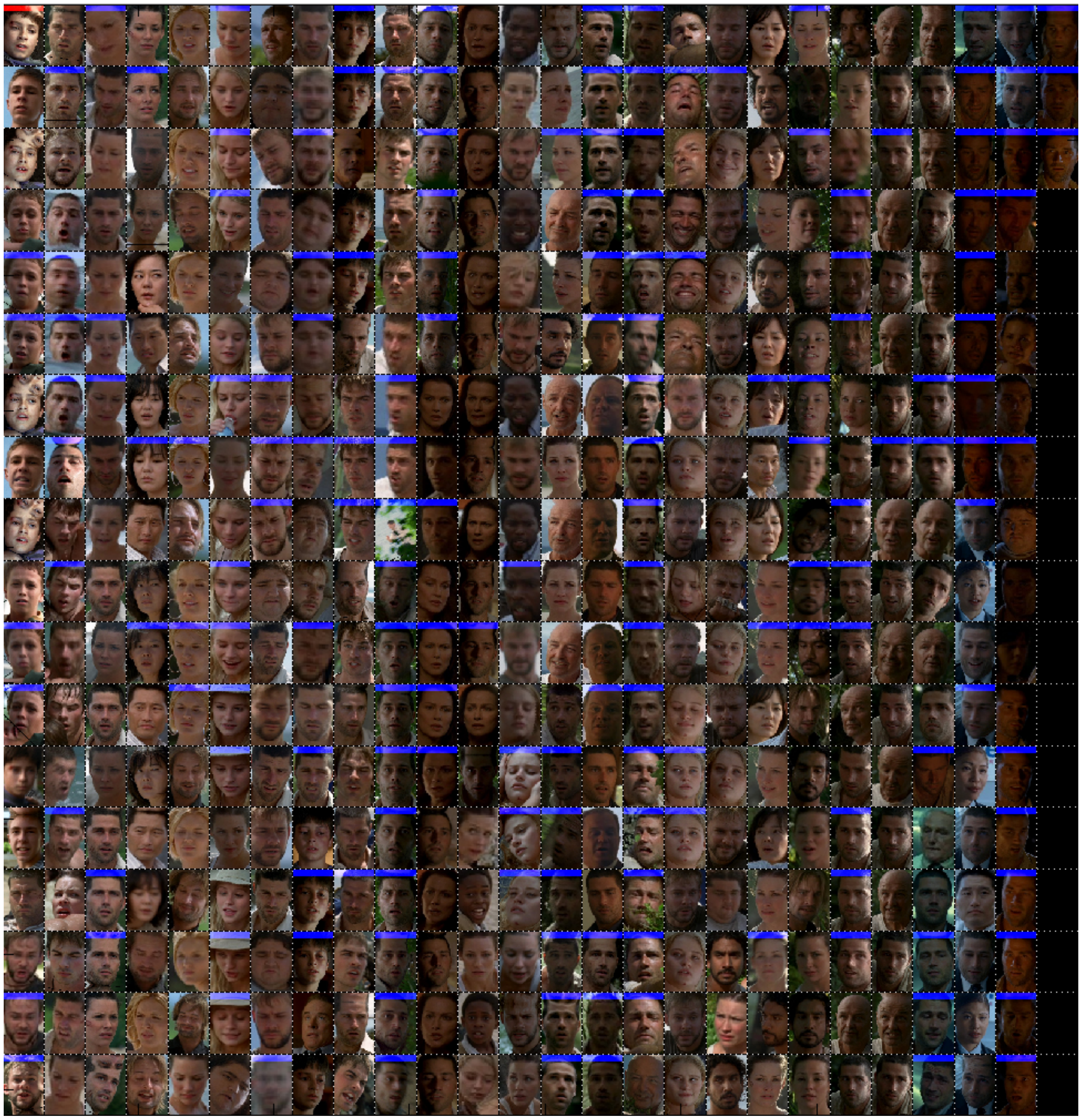

Figure 1. Original ordering of face tracks on Lost Season 1 episode 5 (reads columnwise, and shows the central face in each face track). Blue shows two consecutive face tracks with same identity (either because of two consecutive shots showing a same person, or because of face track was interrupted).

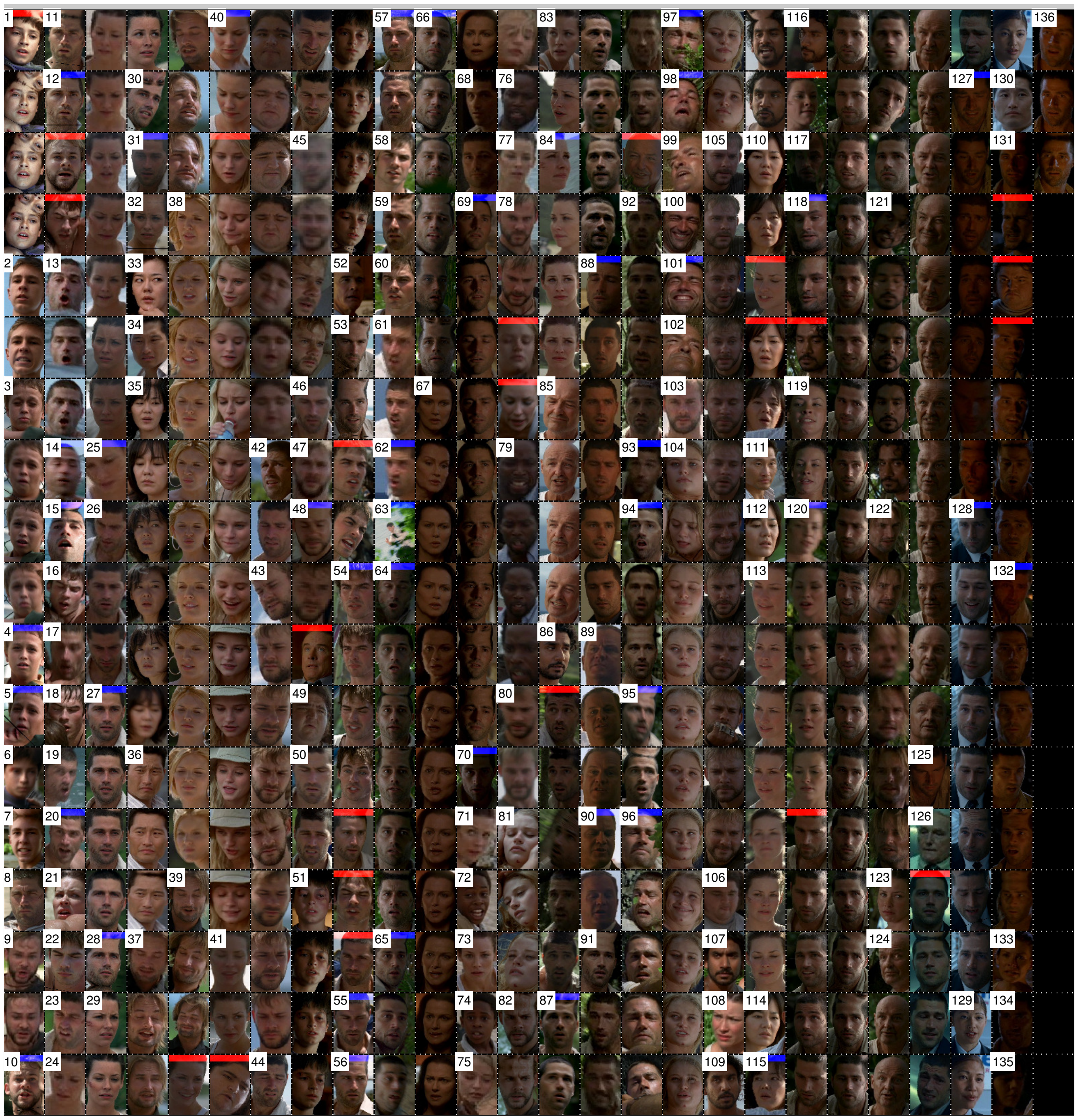

Figure 2. Grouping results for the face tracks on figure 1 (reads columnwise). Blue represents bad splits, red represents bad merges (compared to groundtruth). We have reordered faces according to their cluster id (shown as a white rectangle). The number of clusters is fixed to be 30% of the total number of face tracks, as in the paper.