# WEAKLY SUPERVISED LEARNING FROM MULTIPLE MODALITIES: EXPLOITING VIDEO, AUDIO AND TEXT FOR VIDEO UNDERSTANDING

Timothee Cour

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial

Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2009

Ben Taskar
Supervisor of Dissertation

Rajeev Alur
Graduate Group Chairperson

# Acknowledgements

I would like to acknowledge first and foremost my advisor, Ben Taskar, who represents in my eyes what a perfect advisor should be. As his first Ph.D. student I benefited from a lot of his guidance, his knowledge in machine learning and his inspiration for tackling difficult problems. I am particularly grateful to him for helping me select a very interesting thesis topic, and for his constant push for quality, original and ambitious work. Ben was and still is a mentor for me.

I would like to thank my thesis committee, Kostas Daniilidis, Fernando Pereira, CJ Taylor and Andrew Zisserman, for their excellent feedback, questions and suggestions which shaped this thesis to its current form. I have found in their work a source of inspiration and knowledge. The work of Andrew on character naming in video using screenplay was particularly influential for me, as was the work of Fernando on Conditional Random Fields. As my committee chair, Kostas gave me invaluable advice to improve my thesis, as well as academic guidance throughout my Ph.D. I would also like to thank my written preliminary exam committee, Ali Jadbabaie, Kostas Daniilidis and Lyle Ungar. I had the chance to work with Eleni Miltsakaki, Michael Shilman, Paul Viola, and especially Jianbo Shi, who taught me a lot about computer vision, how to think hard about a problem and how to ask the right questions before proposing a solution. Jianbo has greatly influenced my research, presentation skills, and approach to problem solving.

As a Grasp lab alumni, I have had the privilege to collaborate, work with, or interact with a number of outstanding people. My special thanks go to Ben Sapp, who was an amazing colleague. I would also like to thank Akash Nagle, Chris Jordan, Praveen

Srinivasan, Florence Benezit, Nicolas Gogin, Qihui Zhu, Alex Toshev, Sandy Patterson, Katerina Fragiadakis, Mirko Visontai and all those that I forgot to mention. Collaborators or friends, they all contributed to making my stay at Upenn a very enjoyable one. Special thanks to Surabhi, who helped me in some of the difficult moments.

Last but not least, I would like to express deep gratitude to my parents, Caroline and Jean-marie, for their never-ending support throughout my studies and their help to make difficult decisions. I would like to thank finally my brother Bijan, my sisters Marjolaine and Philippine, for being such a united and supportive family.

ABSTRACT

WEAKLY SUPERVISED LEARNING FROM MULTIPLE MODALITIES: EXPLOITING VIDEO, AUDIO AND TEXT FOR VIDEO UNDERSTANDING

Timothee Cour

Ben Taskar

As web and personal content become ever more enriched by videos, there is increasing need for semantic video search and indexing. A main challenge for this task is lack of supervised data for learning models. In this dissertation we propose weakly supervised algorithms for video content analysis, focusing on recovering video structure, retrieving actions and identifying people. Key components of the algorithms we present are (1) alignment between multiple modalities: video, audio and text, and (2) unified convex formulation for learning under weak supervision from easily accessible data.

At a coarse level, we focus on the task of recovering scene structure in movies and TV series. We present a weakly supervised algorithm that parses a movie into a hierarchy of scenes, threads and shots. Movie scene boundaries are aligned with screenplay scenes and shots are reordered into threads. We present a unified generative model and novel hierarchical dynamic program inference.

At a finer level, we aim at resolving person identity in video using images, screenplay and closed captions. We consider a partially-supervised multiclass classification setting where each instance is labeled ambiguously with more than one label. The set of potential labels for each face is the characters' names mentioned in the corresponding screenplay scene. We propose a novel convex formulation based on minimization of a surrogate loss. We show theoretical analysis and strong empirical proof that effective learning is possible even when all examples are ambiguously labeled.

We also investigate the challenging scenario of naming people in video without screenplay. Our only source of (indirect) supervision are person references mentioned in dialog, such as "'Hey, Jack!". We resolve identities by learning a classifier from partial label constraints, incorporating multiple-instance constraints from dialog, gender and local grouping constraints, in a unified convex learning formulation. Grouping constraints are provided by a novel temporal grouping model that integrates appearance, synchrony and film-editing cues to partition faces across multiple shots. We present dynamic programming inference and discriminative learning for this partitioning model.

We have deployed our framework on hundreds of hours of movies and TV, and present quantitative and qualitative results for each component.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

With the ever increasing number of videos available on the web and personal collections, there is a growing need for content-based retrieval, semantic indexing and smart browsing devices. Current video indexing solutions still rely on text annotations, which are often sparse and incomplete. While there is active research on the topic of video understanding, a key challenge is the lack of supervision for training predictive models. In this dissertation we propose weakly supervised algorithms for video content analysis, specifically focusing on identifying people and retrieving actions, see figure 1.1. A key component of the learning algorithms we present is alignment and interaction between multiple modalities: video, audio and text. In contrast to previous work, we do not use any annotation or supervision other than readily available data accompanying the video.

## 1.1 Motivation

One important motivation for studying videos instead of image datasets is that there is much interesting structure in videos to be revealed that can be helpful for understanding its content. If a picture is worth a thousand words, than certainly a video can reveal a lot more. Besides the addition of a third dimension that can be used for tracking, videos are usually associated with an audio stream, as well as text in the form of closed captions,

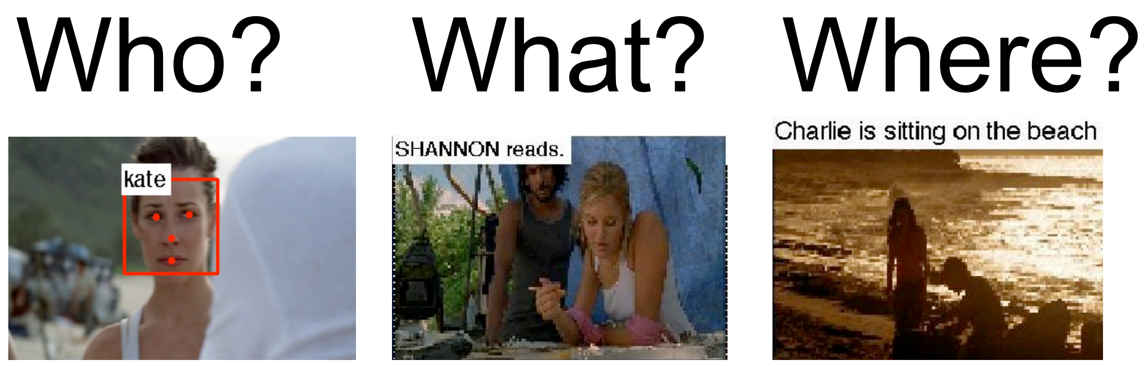


Figure 1.1: Basic tasks for video understanding include: (1) *who* is in the picture, (2) *what* are they doing, (3) *where* is the scene taking place, as well as other questions such as when is the action happening, what are the objects, etc. In this thesis we attempt to answer some of these questions in a weakly supervised manner with the help of easily accessible data: video, audio, screenplay and closed captions.

subtitles or screenplay. Understanding the interaction between the visual, audio, and text modalities provides independent verifications for understanding the underlying action. For example, in [Everingham et al., 2006] the authors exploit the synchronous motion of the mouth with speech utterances to localize and name actors in TV series. In this thesis we will identify and exploit many other cross-modality features in video.

A second motivation for this thesis is to use videos as a virtually infinite source of weakly labeled data. We have collected, decompiled and automatically parsed a dataset of over 200 TV series and movies, and extracted audio files corresponding to 1,000 different speakers annotated with speaker name and text. We have assembled a dataset of 10 million registered faces (the largest we know of at this time) and automatically assigned names to a large number of those. See figure 1.2 for a sample of faces classified by our weakly supervised algorithm using video and screenplay in TV serie LOST. We have gathered and automatically parsed over 2000 screenplays and subtitle files and demonstrated their use for retrieving a wide diversity of actions in video, see figure 1.6 for an example. Such data, when correctly aligned, provides invaluable help for various supervised task such as gender recognition from speech or face, or to bootstrap non-frontal face detectors or detect actions “in the wild”. We argue that large quantities of cheaply available, weakly labeled

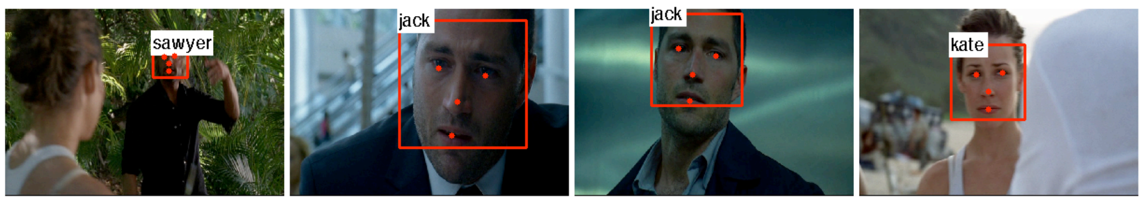


Figure 1.2: Examples of registered faces and named characters using our weakly supervised algorithm from chapters 4-6.

data can compensate the lack of manually annotated groundtruth.

Finally, a third motivation for this thesis is to use videos as a realistic testbed to develop and evaluate algorithms in interesting learning scenarios, such as: ambiguously labeled images, multiple instance learning, learning from multiple modalities, and inference and learning with rich, semi-local constraints.

## 1.2 Problem Statements

We analyze videos at multiple granularities. At a coarse level, we focus on recovering scene structure in movies and TV series for object tracking and action retrieval. We present a weakly supervised algorithm that uses the screenplay and closed captions to parse a movie into a hierarchy of scenes, threads and shots, as illustrated in figure 1.3. Scene boundaries in the movie are aligned with screenplay scene labels and shots are reordered into a sequence of long continuous threads which allow for more accurate tracking of people, actions and objects. Scene segmentation, alignment, and shot threading are formulated as inference in a unified generative model, and we present a novel hierarchical dynamic programming algorithm that can handle alignment and jump-limited reorderings.

At a finer level, we tackle the problem of resolving person identity in video using images, screenplay and closed captions. To this end we consider a partially-supervised multiclass classification setting where each instance is labeled ambiguously with more than one label. In our setting, the set of potential labels for each face is the characters'

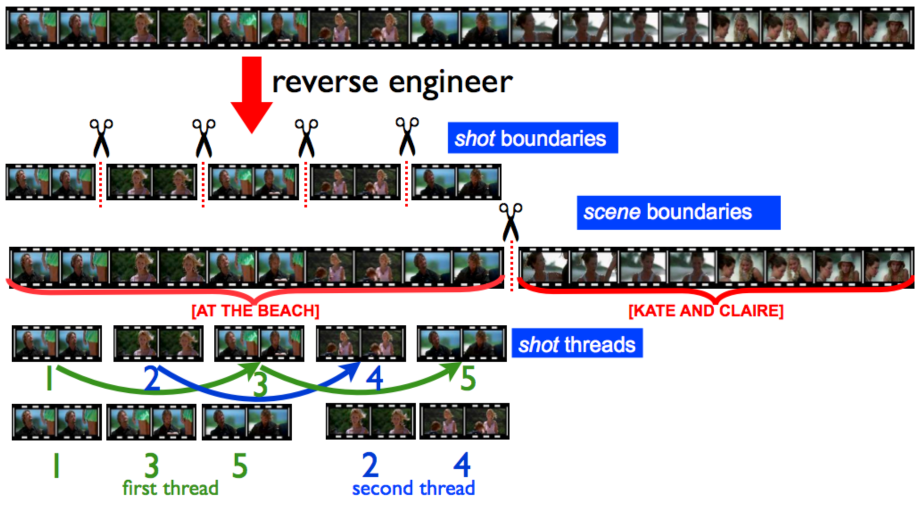


Figure 1.3: Deconstruction pipeline.

names mentioned in the screenplay in the corresponding scene, see figure 1.4. Another typical example arises in photograph collections containing several faces per image and a caption that only specifies who is in the picture but not which name matches which face. We propose a novel convex optimization formulation based on minimization of a surrogate loss appropriate for the ambiguous label setting. We show theoretical analysis and strong empirical proof that effective learning is possible under reasonable assumptions even when all examples are ambiguously labeled.

We also investigate the much more difficult scenario where no screenplay is available, see figure 8.1. We address the identity resolution problem in that case: assigning a name to each encountered face. Our only source of 'supervision' are the naturally occurring linguistic 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 (face appearance, voice characteristics, mouth movement and speech synchrony, video-editing cues) to effectively resolve identities globally. We propose a novel sequence partitioning model that groups faces and utterances (denoted

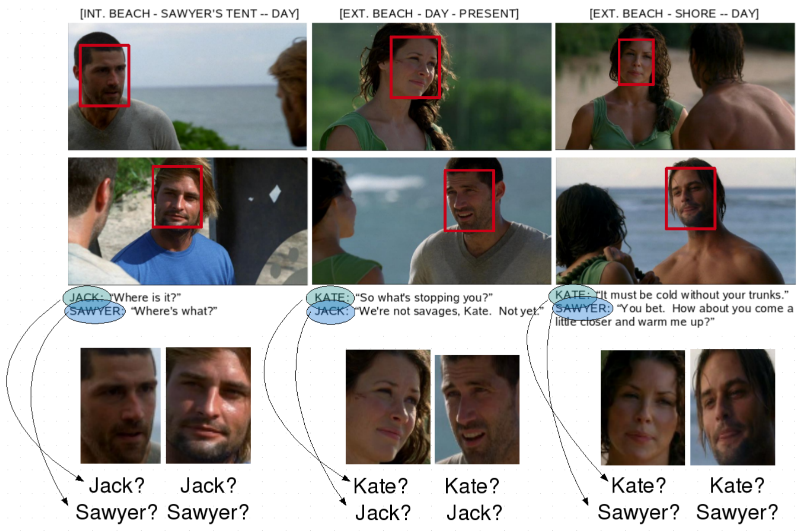

[INT. BEACH - SAWYER'S TENT -- DAY]
[EXT. BEACH - DAY - PRESENT]
[EXT. BEACH - SHORE -- DAY]
JACK: "Where is it?"
SAWYER: "Where's what?"
KATE: "So what's stopping you?"
JACK: "We're not savages, Kate. Not yet."
KATE: "It must be cold without your trunks."
SAWYER: "You bet. How about you come a little closer and warm me up?"
Jack? Sawyer?
Jack? Sawyer?
Kate? Jack?
Kate? Jack?
Kate? Sawyer?
Kate? Sawyer?

Figure 1.4: We use the aligned screenplay to generate a set of potential labels for each face. Our ambiguous learning algorithm correctly recovers their identity.

as units) while respecting local constraints. In this model, states represent partitions of the k most recent units (such as ABAB, representing a bipartition of units A and B), and transitions represent compatibility of such partitions. We present an efficient dynamic program for inference and a large margin parameter estimation method. The resulting clusters of faces are subsequently assigned a name by learning a classifier from partial label constraints. The weakly supervised classifier incorporates multiple-instance-type constraints from dialog cues as well as local grouping constraints.

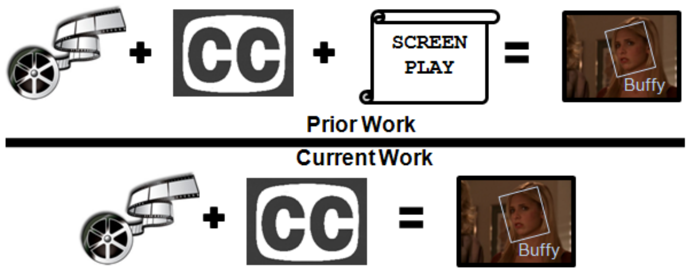


Figure 1.5: We present a new, challenging, problem: identity resolution in video in the absence of screenplay. When screenplay is not available, we know *what* is being said but not *who* said it. The only source of name information is from dialog cues: first, second and third person references.

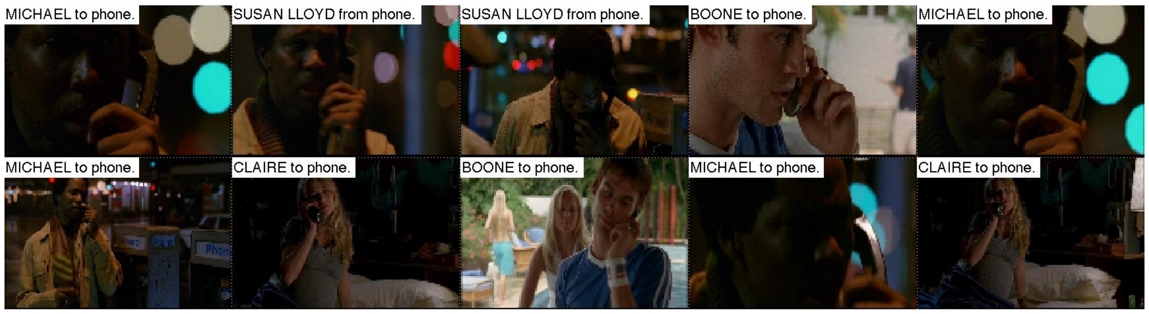


Figure 1.6: Action retrieval for “phone” using alignment between video and screenplay.

| chapter | Input | Output | Supervision |
|---|---|---|---|
| 3 | video, closed captions, screenplay | coarse alignment | weak |
| 5 | image,audio | classification | ambiguous |
| 6 | video, closed captions, screenplay | naming | ambiguous |
| 8 | video, closed captions | naming | multiple-instance |
| 7 | video | clustering | unsupervised |

Figure 1.7: Summary of chapters: input-output relations, and type of supervision

We have deployed our framework on hundreds of hours of video from movies and TV. We present quantitative and qualitative results for the different components of our system, including movie alignment and parsing, character naming and retrieval of actions.

## 1.3 Contributions

We make several contributions in this thesis.

- We show how screenplays can be used to recover the detailed structure of movies and demonstrate the use of the aligned screenplay for action retrieval and character naming applications

- We propose an effective convex optimization algorithm for the ambiguous learning scenario, and show that accurate learning is generally possible even when all examples are ambiguously labeled. We show our proposed convex formulation gives a tighter approximation to the discrete ambiguous loss, compared to a related naive approach. We provide extensive theoretical analysis connecting the true loss with the ambiguous loss, including transductive disambiguation bounds. We show how those bounds are affected by the degree of ambiguity of the underlying distribution, and provide label specific recall bounds to distinguish hard to disambiguate classes from easier ones. We analyze the finite sample case, providing concrete bounds on the disambiguation error rate. We propose several flavors of our ambiguous learning method, based on Linear Programming, Support Vector Machines, Boosting, as well as a Kernelized variant.

- We validate our ambiguous learning method with extensive controlled experimentation, including image, audio, and standard machine learning datasets. We also present state of the art results on automatic character naming in TV video, using a screenplay as weak supervision. We introduce novel cues targeted for the ambiguous label setting and achieve $6\%$ error rate (resp. $13\%$) for character naming, across 16 episodes of Lost and 8 characters (resp. 32 characters). We consider the setting where the ambiguous label set contains occasional errors, ommitting the true label (due to aggressive pruning heuristics for example). To this end we propose an improved, hybrid confidence measure to boost precision recall curves.

- We consider the unsupervised setting where no screenplay or closed captions are available. We address the problem of learning how to cluster faces, and present a novel *temporal grouping model* that groups faces across an episode based on arbitrary local constraints, such as appearance, gender constraints, shot alternation and other local film structure cues. We present a novel dynamic programming inference and discriminative large margin learning for the model, which operates on local partitions of the data. We evaluate on several hours of TV and movies, achieving significantly higher accuracy than several strong baselines.

- Finally we present a new, challenging, problem: identity resolution in video with closed captions but no screenplay. We learn a classifier from partial label constraints, with a convex formulation that incorporates constraints from multiple modalities including dialog cues for naming, grouping cues, and gender cues. The weakly supervised classifier incorporates multiple-instance-type constraints from dialog cues, as well as exclusion constraints from gender and must-link constraints from local grouping. We present the first quantitative results on dialog supervised naming in TV, achieving $42\%$ accuracy across all 55 labels, computed over the 10 most frequent characters in 8 episodes of Lost.

- Along the way, we also create large datasets of faces, annotated speech samples, parsed and aligned screenplays and closed captions, and localized action snippets. As sub-components of our system we also design classifiers for a number of basic tasks, such as efficient part localization, face tracking, face gender classification, audio gender classification, pronoun resolution, first, second and third reference classification to name a few.

## 1.4 Thesis outline

We refer the reader to table 1.7 for a summary of input-output relations and type of supervision considered in each chapter. The rest of the thesis can be summarized as follows:

- **Chapter 2. From Supervised to Weakly Supervised Learning:** We start with an overview of different basic learning scenarios, from supervised learning to weakly supervised learning. This will help putting our contributions in perspective.

- **Chapter 3. Coarse-Level Alignment of Video and Text Transcriptions:** We focus on the task of recovering scene structure in movies and TV series for object tracking and action retrieval. We present a weakly supervised algorithm that uses the screenplay and closed captions to parse a movie into a hierarchy of shots and scenes. Scene boundaries in the movie are aligned with screenplay scene labels and shots are reordered into a sequence of long continuous tracks or threads which allow for more accurate tracking of people, actions and objects. Scene segmentation, alignment, and shot threading are formulated as inference in a unified generative model and a novel hierarchical dynamic programming algorithm that can handle alignment and jump-limited reorderings in linear time is presented. We present quantitative and qualitative results on movie alignment and parsing, and use the recovered structure to improve character naming and retrieval of common actions in several episodes of popular TV series.

- **Chapter 4. Learning from Ambiguously Labeled Images:** we consider a partially-supervised multiclass classification setting where each instance is labeled ambiguously with more than one label. A typical example arises in photograph collections containing several faces per image and a caption that only specifies who is in the picture but not which name matches which face. We propose a general convex optimization algorithm based on minimization of a surrogate loss appropriate for the ambiguous label setting. We show theoretically and empirically that effective learning is possible under reasonable assumptions even when all the data is weakly labeled.

- **Chapter 5 . Experiments for Learning with Ambiguously Labeled Data:** We apply our framework to identifying faces culled from web news sources, as well as speaker recognition from audio and classification in standard machine learning datasets. We provide extensive experimental results comparing our proposed approach with 7 baselines.

- **Chapter 6. Learning with Ambiguously Labeled Faces in Videos:** We focus on the task of automatic character naming in TV video, using a screenplay as weak supervision. We show how to design ambiguous labels from coarse screenplay alignment, and apply our method which we compare to several baselines. We also introduce novel cues targeted for the ambiguous label setting, and perform an ablative analysis of performance. We experiment on a very large dataset consisting of 100 hours of video, and in particular achieve $13\%$ error for character naming across 32 labels on 16 episodes of Lost.

- **Chapter 7. Temporal Grouping:** 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. We evaluate on several hours of TV and movies, achieving significantly higher accuracy than several strong baselines. We perform ablative analysis.

- **Chapter 8. Identity Resolution Without Screenplay:** We present a new, challenging, problem: identity resolution in video with closed captions but no screenplay. In this chapter, 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 with weakly supervised recognition. We build upon the previous chapter, and cluster the face tracks with local temporal grouping cues. The resulting clusters of faces are subsequently assigned a name by learning a classifier from partial label constraints. The weakly supervised classifier incorporates multiple-instance-type constraints from dialog cues as well as local grouping constraints . We perform ablative analysis as well as consider the cases where certain cues are perfect.

- **Chapter 9. Conclusion and Directions for Future Work:** We review the main contributions presented in this dissertation, and present the applicability and limitations of the methods presented. We discuss future work we plan to work on.

## 1.5 Previously published work

Some of the material presented in this dissertation has been published in conference proceedings. The coarse-level alignment of video and text presented in chapter 3 is based on [Cour et al., 2008]. Chapters 4, 5 and 6 extend the original framework and experiments presented in [Cour et al., 2009a]. Finally, chapters 7 and 8 describe and extend the results presented in paper currently under review, [Cour et al., 2009b]. Additional previous work [Cour and Shi, 2007b, Cour and Shi, 2007a, Cour et al., 2005a, Cour et al., 2005b, Cour et al., 2007] is not described in this thesis.

# Chapter 2

# From Supervised to Weakly Supervised Learning

We review in this chapter basic definitions and methods for several learning scenarios, which will be used in the next chapters. We want to classify an input $x_i$ in some input space $\mathcal{X}$ (for example an image of a face) as some label $y_i$ in a label space $\mathcal{Y}$ (for example a name, see figure 2.1). The learner seeks to select the best function $g : \mathcal{X} \rightarrow \mathcal{Y}$ in some **hypothesis class** (or model) $\mathcal{G}$ given some form of supervision $\mathcal{D}_{\text{train}}$ (**training data**), some **loss function** or criterion to optimize, as well as some **optimization procedure**. The main dimension we consider is the **type of supervision** available to the learner, which we summarize in figure 2.2.

- In **supervised** learning, the learner has access to a set of $m$ training examples, all of

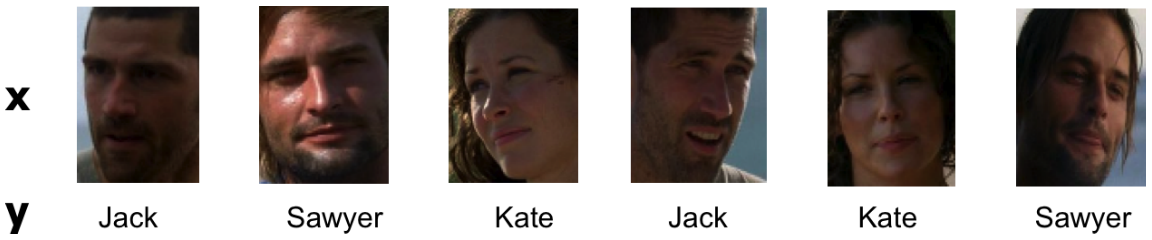


Figure 2.1: A supervised task: classifying an input face $x$ as a name $y$.

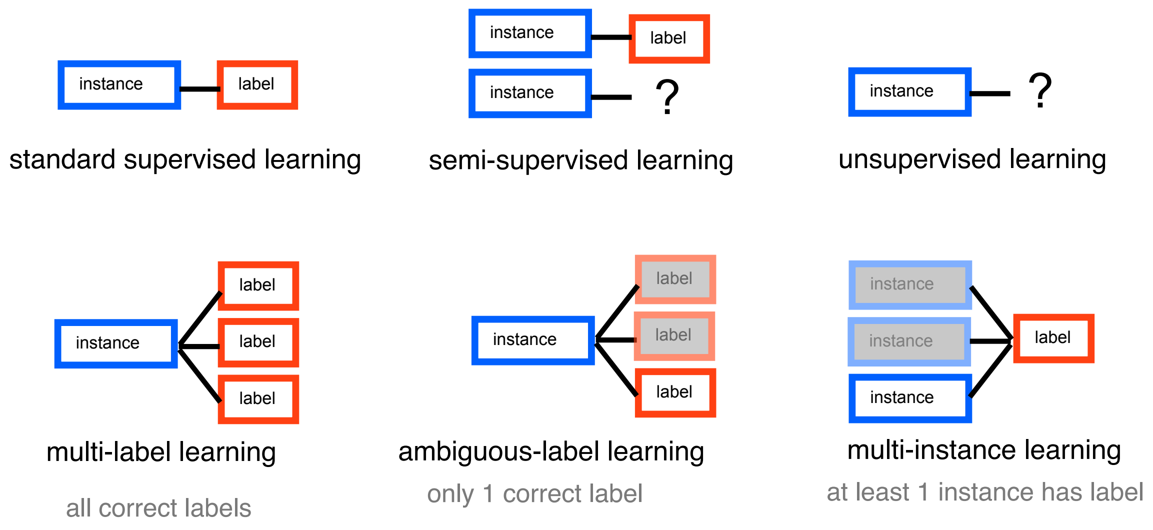


Figure 2.2: Diagram of different learning scenarios.

which are labeled: $\mathcal{D}_{\text{train}} = (x_i, y_i)_{i=1}^m$. This is a fully supervised setting.

- In **semi-supervised** learning, the learner has access to a set of $m$ labeled examples as well as a set of $m'$ unlabeled examples: $\mathcal{D}_{\text{train}} = (x_i, y_i)_{i=1}^m\} \cup \{(x_i)_{i=m+1}^{m+m'}$. This is a form of *partial* supervision.

- In **multiple-instance** learning, examples are not individually labeled but grouped into sets $S_i$ which either contain at least 1 positive example, or only negative examples: $\mathcal{D}_{\text{train}} = ((x_j)_{j\in S_i}, y_i)_{i=1}^m$. This is a form of *weak* supervision. We will address such a setting in chapter 8, and propose a novel convex formulation for multiple instance constraints.

- In **multi-label** learning, each example $x_i$ is assigned multiple labels, all of which can be true: $\mathcal{D}_{\text{train}} = (x_i, (y_j)_{j\in S_i})_{i=1}^m$. This can be considered as a fully supervised setting.

- In **ambiguous learning**, which is the topic of chapter 4, each example again is supplied with multiple potential labels, *only one of which is correct*. This is a form of *weak* supervision.

- Finally, **unsupervised** learning is a degenerate case in which no example is labeled: $\mathcal{D}_{\text{train}} = (x_i)_{i=1}^m$. We will address such a setting in chapter 7, with a temporal grouping algorithm.

Note that many other learning scenarios are possible, such as Multi-Instance Multi-Label Learning [Zhou and Zhang, 2007], Multi-Task Learning[Caruana, 1997], ranking etc. The prime reason for so many alternatives to standard supervised learning is the lack of supervised data for practical applications, where human labeling effort is expensive or unavailable. In many tasks such as the ones we consider in this thesis, only some form of *partial* or *weak* supervision is available, sometimes at little or no cost in terms of human labeling. In contrast to semisupervised learning, weak supervision includes the setting where none of the training examples are labeled; instead the learner has access to a weaker form of supervision such as grouping constraints, multiple instance learning constraints or ambiguous label constraints. We review below the main concepts and algorithms referenced in this thesis and refer the reader to [Zhu, , Chapelle et al., 2006, Maron, 1998, Ghahramani, 2004] for more complete surveys.

## 2.1 Supervised Learning

In supervised learning, the training data $\mathcal{D}_{\text{train}} = (x_i, y_i)_{i=1}^m$ is drawn i.i.d. (independent and identically distributed) from a (unknown) distribution $P(x, y)$. The goal of the learner is to compute a hypothesis $g : \mathcal{X} \to \mathcal{Y}$ such that $g(x)$ approximates $y$ for new samples $(x, y) \sim P$. Even though the training data is fully supervised, the problem remains fundamentally ill-posed as the data is insufficient to find a unique solution that will give the best **generalization**. The first decision to make pertains to the hypothesis class $\mathcal{G}$ from which to select $g$. A wide range of options are available and the best choice often depends on assumptions about the data (linear separability, presence of outliers, amount of data, prior model). If $\mathcal{G}$ is too large, the model might **overfit** the training data, and if $\mathcal{G}$ is too small the model will underfit and also result in poor generalization. [Maruyama et al., 2002] studies the relationships in terms of relative approximation ratios between linear discriminant functions and certain boolean functions (decision lists, decision trees, and conjunctive or disjunctive normal form boolean formulas). One can distinguish **instance-based** methods (k-Nearest Neighbor, radial basis functions) which memorize the training data, from **parametric** methods (generalized log-linear models, neural networks, Kernel methods, decision trees) which summarize the data using parameters. The distinction is subtle however, as certain methods can be considered as both parametric (primal version of Support Vector Machine or Perceptron) and instance-based (dual version, expressing the classifier as linear combination of training inputs). A related problem is **model choice**, or how to select the best meta-parameters given the training data. For example, the learner needs to choose the depth of a decision tree, the number of hidden units in a neural network, or width for a Gaussian Radial Basis Function. Another example is the type and amount of regularization. Typically cross-validation [Kohavi, 1995] or one of its variants is used for that purpose, where one trains the parameters on subsets or folds of the training data and evaluates on the remaining fold or validation set.

The second decision to make is the loss function or criterion to optimize. We are given a loss function $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^+$ such that $\mathcal{L}(x, y, \hat{y})$ measures the penalty incurred by a prediction $g(x) = \hat{y}$ for an example $(x, y)$. For binary classification, $\mathcal{Y} = \Re, y \in \{+1, -1\}$ and the 0-1 loss is typically used: $\mathcal{L}(x, y, g(x)) = \mathbb{1}(y \neq sign(g(x))) = \mathbb{1}(yg(x) \leq 0)$. More generally, a margin-based loss is of the form $\mathcal{L}(x, y, g(x)) = \mathcal{L}(yg(x))$. We will introduce new interesting losses in the next chapters for ambiguous learning and sequence partitioning. We define the **risk** as $R(g) = E_{(x,y)\sim P}[\mathcal{L}(x, y, g(x))]$, which the learner estimates using the **empirical risk** defined as $\hat{R}(g) = E_{(x,y)\in\mathcal{D}_{\text{train}}}[\mathcal{L}(x, y, g(x))]$, also known as training error for 01 loss. Direct minimization of the empirical risk, however, is intractable in many situations [Arora et al., 1993, Bishop, 1995, Quinlan, 1986] for the 01-loss, and may lead to multiple solutions. A common approach is to minimize instead a convex surrogate $\psi(z)$

of the 01-loss, leading to a convex optimization problem for a wide class of models such as generalized linear models. Figure 2.3 shows examples of surrogate losses commonly used in algorithms such as Support Vector Machines (hinge loss), Adaboost (exponential loss), Logistic Regression (log loss). The associated approximation error in replacing the 01-loss with the surrogate loss is the topic of [Steinwart, 2007, Bartlett et al., 2006]. These studies also suggest that using a smoother loss than the 01-loss can provide **regularization** benefits, in order to prevent overfitting the training data, *i.e.*achieving a very small empirical risk at the expense of actual risk. For example, a nearest-neighbor classifier achieves zero empirical risk by definition, but may have high variance, which can be reduced by averaging using K-nearest neighbor classifier. Another common regularization approach is to augment the loss function with an additive regularization term that penalizes complex models, in effect trading off model complexity with low fitting error, see [Devroye et al., 1996, Hastie et al., 2001] for more in-depth treatment:

$$\min_{g \in \mathcal{G}} \mathcal{R}eg(g) + C\hat{R}(g) \tag{2.1}$$

The coefficient $C \geq 0$ is set using cross-validation. Early stopping (for example during boosting or when training Neural Networks) is another example of regularization commonly used.

Finally, the choice of the loss function and regularization often dictates the type of optimization procedure to use, and we can distinguish between tractable methods (such as the ones arising from convex optimization or certain polynomial time search algorithms) from intractable ones which involve sampling or some form of search heuristic. We will focus on methods arising from convex optimization as they are well understood, guarantee optimality and often give rise to efficient algorithms. However even for convex optimization problems with guaranteed polynomial time solutions, the particular type of problem/solver used can greatly impact the accuracy or running time, a typical example being different primal/dual formulations of linear Support Vector Machines.

In summary the incurred risk $R(g)$ accumulates four types of errors:

- the estimation error due to finite sample size; this depends on the training data

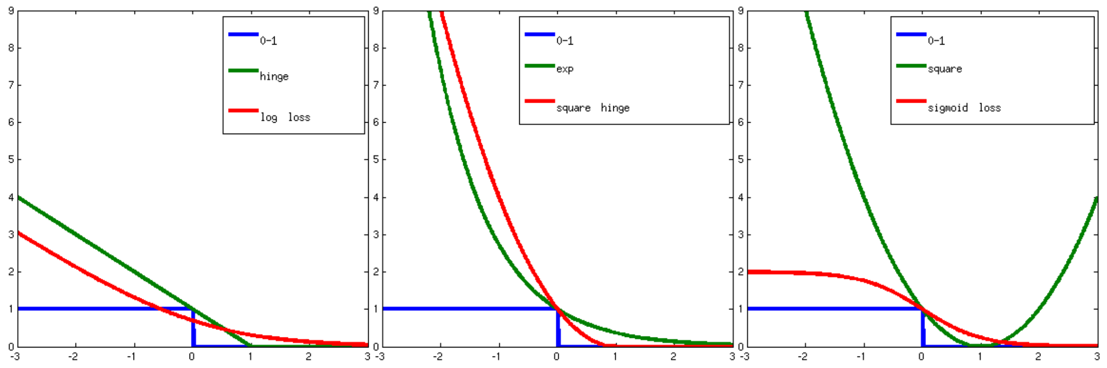


Figure 2.3: Commonly used binary loss functions as continuous upper-bounds on the 01-loss $\mathbb{1}(u \leq 0)$: hinge loss $\max(0, 1-u)$, log-loss $\log(1+\exp(-u))$, exponential loss $\exp(-u)$, square hinge loss $\max(0, 1-u)^2$, square loss $(1-u)^2$ and sigmoid loss $1-\tanh(u)$. Note the log-loss is a proper upper bound when scaled by a factor $1/\log(2)$. Also note, the square loss is not monotonous, the sigmoid loss is not convex, and the hinge loss is not differentiable.

- the Bayes Risk $\min_{g \in \mathcal{G}} R(g)$; this depends on the expressive power of the function class $\mathcal{G}$
- the approximation error that depends on how well the surrogate loss $\psi(z)$ approximates the 01-loss
- the optimization error that depends on the accuracy of the optimization procedure (typically small for convex optimization problems)

## 2.2 Basic Supervised Classification Models

We review below some important supervised classification models that can be extended for other types of supervision and that we will use later on.

### 2.2.1 Generalized Linear Family

We are given a basis of feature functions $f : \mathcal{X} \times \mathcal{Y} \to \Re^d$. Note, the dependency of $f$ on $(x, y)$ can be arbitrary, including non-linear and even infinite-dimensional via the use of Kernels. A wide class of supervised classification models can be derived from the generalized linear family, in which a hypothesis is parametrized linearly according to $f$:

$$g_w(x) = \arg\max_y \mathbf{w} \cdot f(x, y) \tag{2.2}$$

### 2.2.2 Support Vector Machines

Support Vector Machines [Vapnik, 1995] aim at constructing a separating hyperplane that maximizes the margin between the positive and negative class, resulting in good generalization ability. We assume here the feature mapping can be written as $f(x, y) = f(x) \in \Re^d$. In the binary linear separable case, we seek $w \in \Re^d$ such that $y(\mathbf{w} \cdot f(x) - b) \geq 0$ for each training example $(x, y)$, while maximizing the margin $1/||\mathbf{w}||$. In the non-separable case, we introduce slack variables $\xi_i \geq 0$ for each constraint, and arrive at the following convex quadratic program:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{2.3}$$

$$\text{s.t.} \quad \xi_i \geq 0, \quad y_i(\mathbf{w} \cdot f(x_i) - b) \geq 1 - \xi_i \tag{2.4}$$

At the optimum, $\xi_i = \psi(y_i(\mathbf{w} \cdot f(x_i) - b))$ where $\psi$ is the hinge loss upper bounding the 01-loss. The dual of (2.3) has the following form:

$$\max_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j f(x_i) \cdot f(x_j) \tag{2.5}$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0 \tag{2.6}$$

The dual solution $\alpha$ gives the solution of the primal solution $w$ as a nonnegative (sparse) linear combination of a subset of the inputs (the support vectors): $w = \sum_i \alpha_i y_i f(x_i)$. Since $f(x_i)$ only appears as a dot product in (2.9), the dual can be solved without reference to the (possibly infinite) dimension of $f(x_i)$. We can rewrite the dual more compactly by introducing the kernel function $K(x, x') \stackrel{\text{def}}{=} f(x) \cdot f(x')$, the (positive semidefinite) matrix $K = (K(x_i, x_j))_{ij}$, label matrix $Y = (y_i y_j)_{ij}$, and Hadamard product

$\odot$:

$$\max_{\alpha} \quad 2\mathbf{1}^{\mathsf{T}}\alpha - \alpha^{\mathsf{T}} K \odot Y \alpha \tag{2.7}$$

$$\text{s.t.} \quad \alpha \geq 0, \quad \alpha \leq C, \quad y^{\mathsf{T}}\alpha = 0 \tag{2.8}$$

Using the representer theorem, the classifier can be used at run-time on a new input $x$ without reference to $f(x)$:

$$\mathbf{w} \cdot f(x) = \sum_i \alpha_i K(x, x_i) \tag{2.9}$$

This is commonly referred to as the Kernel trick, and can be used in several other algorithms such as Perceptron, Logistic Regression, Principal Components Analysis, Clustering. Note, other variations of the regularization term and loss are also possible, but they may not produce a sparse set of support vectors.

Several multi-class extensions have been proposed to Support Vector Machines [Crammer et al., 2001, Weston, 1998], for example reducing the multiclass problem to multiple binary classification problems (one-vs-all or one-vs-one), followed by a winner-take-all or max-voting strategy. The approach proposed by [Crammer et al., 2001] differs in that a single binary classification problem is formed: they maximize the minimal (real valued) margin between $f(x_i, y_i)$ and all the other labels $f(x_i, y)$, $\forall i, y \neq y_i$:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{2.10}$$

$$\text{s.t.} \quad \mathbf{w} \cdot (f(x_i, y_i) - f(x_i, y)) \geq 1 - \xi_i \quad \forall i, y \neq y_i \tag{2.11}$$

In this setting, the offset $b$ is redundant, and the feature functions $f(x, y)$ can depend on the label $y \in \mathcal{Y} = \{1, ..., L\}$.

### 2.2.3 Boosting

The general idea behind boosting is to linearly combine a set of **weak learners** (whose error rate over "any" distribution is $< 0.5 - \epsilon$, for some fixed $\epsilon > 0$) into a **strong**

**classifier** (whose error rate is arbitrarily close to 0). At each round of boosting, a weak learner is trained with respect to a distribution and added (with some weight) to the ensemble model, typically depending on the weak learner's accuracy. The resulting strong classifier can reduce both the variance and the bias of the collection of weak learners, resulting in superior performance compared to other model averaging methods such as bagging[Breiman, 1996]. A number of different boosting algorithms have been proposed[Freund and Schapire, 1999, Schapire and Singer, 1999], differing mainly on the choice of the distribution at each round of boosting. For example, FloatBoost[Li et al., 2003] allows removal of weak learners previously selected. In the popular Adaboost[Freund and Schapire, 1999] algorithm, a distribution $z_i$ is maintained over the examples $(x_i, y_i)$ in the training set and updated after each boosting stage, increasing the weight of misclassified inputs. One statistical interpretation is that it fits an additive logistic regression model by stagewise minimization of $E[\exp(-yg(x))]$. We present below the Gentle Adaboost[Friedman et al., 2000] variant or Gentleboost, which minimizes the same functional using Newton steps rather than exact optimization. This prevents any given weak learner from having too much weight, and hence acts as a regularization. The justification of the algorithm, summarized below, is given by forming the $2^{nd}$ order Taylor expansion of

$$\mathcal{L}(g) = E[\exp(-y(g(x) + u(x)))]$$

about $u(x) = 0$: $\forall x$,

$$\begin{aligned} \exp(-y(g(x)+u(x))) &\approx \exp(-yg(x))(1 - yu(x) + y^2u(x)^2/2) && (2.12)\\ &= \frac{1}{2}\exp(-yg(x))[y-u(x)]^2 + \text{ constant} && (2.13)\\ &\propto z\cdot(y-u(x))^2 + \text{ constant} && (2.14)\end{aligned}$$

Other base loss functions have been proposed, such as the log loss in logitboost[Friedman et al., 2000] or even a non-convex loss adapted for noisy datasets,

**Algorithm 1** Gentleboost

```
1: Initialize the weights $z_i = 1/m$ $(i = 1..m)$, and the ensemble model $g(x) = 0$
2: for $t = 1 \dots T$ do
3:     compute weighted least-squares fit $g^t = \arg\min_{u \in \mathcal{G}} E_z[(y - u(x))^2]$
4:     update the ensemble $g(x) := g(x) + g^t(x)$
5:     update the weights $z := z \exp(-yg^t(x))$ and renormalize
6: end for
7: Output the strong classifier $sign(g(x)) = sign(\sum_t g^t(x))$
```

Brownboost[Freund, 2001]. Similarly to the SVM case, a number of multiclass variants have been proposed, such as [Schapire and Freund, 1997, Schapire and Singer, 1999, Zhu et al., , Allwein et al., 2000].

Of course, the quality of the final classifier is limited by the expressive power of the weak learner class of functions. For example, to achieve zero error on a non-linearly separable training set (such as the truth table of the boolean XOR function), one must use non-linear weak learners. The simplest, yet effective, example is a decision stump of the form $u(x) = f(x) \leq \theta$ for some feature function $f(x)$ and some threshold $\theta \in \Re$. In general the accuracy of classifier owes a lot to the design of good features that generalize well by incorporating domain knowledge. For example, for face detection, [Viola and Jones, 2004] uses an overcomplete basis of Haar features $f(x)$ defined on a $24 \times 24$ gray-level image, which are more robust than single pixel estimates. Other more complex weak learners can be used, such as Classification and Regression Trees (CART)[Lewis, 2000], but may lead to a non closed form solution when searching for the optimal newton update.

## 2.3 Semi-Supervised Learning

So far all the models discussed assume each training example has an associated label. In many applications, however, labeled data is hard to obtain, requiring costly human or expert annotation. On the other hand, the common assumption is that **unlabeled data is plentiful** and the central question is whether or not it can be used along with labeled data to build better classifiers. We will denote as $(x_i, y_i)$ the $m$ labeled examples and $x'_i$ the $m'$ unlabeled examples whose label $y'_i$ is to be determined. In generative models such as mixture models, the joint distribution $P(x, y)$ explicitly models $P(x|y)$ as a function of the conditional model $P(x|y)$ and the prior $P(x)$ that models the unlabeled (and labeled) data. In discriminative models (such as Transductive Support Vector Machines and graph-based methods) which do not model $P(x, y)$ explicitly, the prior $P(x)$ is integrated into the objective to influence the classifier.

In absence of any other information, unlabeled data can therefore be used to push the decision boundary of the classifier towards **low density regions** of $P(x)$. Consider for example the task of naming faces in a movie. If the set of (automatically extracted) faces are organized into $k$ well-separated clusters, a single label per cluster could in theory be used to label the set of all faces. In many situations however, weak labeling information is available in the form of **pairwise constraints** of the form $(x_i, x_j, y_{ij})$ with $y_{ij} = +1$ if $x_i, x_j$ must have the same (unknown) label (must-link constraint), and $y_{ij} = -1$ if $x_i, x_j$ must have a different label (must-not-link constraint). For example, two faces in the same face track should have the same label, while two faces in the same image should have different labels. Another form of weak supervision is a prior on the **proportion of each class** (or relative proportions), of the form $P(y_i)$ or $P(y_i, y_j)$. For example, one could expect the relative proportions of names of faces in a movie to match the number of occurrences of each character in the screenplay, or the order given by the cast list.

In order for semisupervised learning to be successful, the model assumptions need to be adapted to the data, for example by building good features or kernels, such that distances within a class tend to be smaller than distances across classes. This can be problematic however since generally only few labeled examples are provided to begin with. A natural solution is **self training**, in which a classifier is initialized based on the labeled data points only, and then iteratively re-trained based on the high-confidence predictions of the classifier at the previous iteration. Such approach has been used successfully in [Everingham et al., 2006] to label characters in TV series, with initial labels coming from alignment of the screenplay to faces classified as "speaking". The performance of such algorithms is in general hard to analyze and depends largely on the quality of the initial labeled data. In particular, the predictions can become over-confident and even drift away from the initial labels if no special provision is taken. **Co-training** [Blum and Mitchell, 1998], or more generally **Multiview learning**, provides such a guard by making sure the predictions agree among two independent classifiers. Each classifier is trained using one split of the feature set. Good performance can be achieved if the two sets of features are independent, and are sufficient to train a classifier.

Mixture models trained with **Expectation Maximization** (EM) provide a softer solution to solve this chicken-and-egg problem, avoiding to make hard decisions at each step. The model iteratively estimates the posterior $P(x|y)$ and the parameters of the mixture model, using coordinate ascent on a lower bound of the data log-likelihood.

In Transductive Support Vector Machines (TSVM) [Vapnik, 1995, Joachims, 1999], the SVM formulation is augmented with a search over the unknown labels $y' \in \{+1, -1\}$ for the unlabeled examples $x'$:

$$\min_{\mathbf{w}, b, \xi, \xi', y'} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i + C'\sum_i \xi'_i \tag{2.15}$$

$$\text{s.t.} \quad \xi_i \geq 0, \quad y_i(\mathbf{w} \cdot f(x_i) - b) \geq 1 - \xi_i \tag{2.16}$$

$$\xi'_i \geq 0, \quad y'_i(\mathbf{w} \cdot f(x'_i) - b) \geq 1 - \xi'_i \tag{2.17}$$

This can be interpreted by removing the labels $y'$ from the optimization and adding the following non convex regularization term on the objective, which drives the decision hyperplane away from dense region:

$$C' \sum_i \hat{\psi}(\mathbf{w} \cdot f(x'_i) - b) \tag{2.18}$$

$$\text{with } \hat{\psi}(u) \overset{\text{def}}{=} \max(0, 1 - |u|) \tag{2.19}$$

Both formulations as well as the approaches presented thus far, however, are **non-convex** in nature and suffer from existence of multiple local minima. A consequence is that the quality of the solution depends crucially on the particular (approximate) optimization method used, except for small problems where exact search is possible[Chapelle et al., 2007]. In contrast, for exact methods such as the ones arising from convex optimization, the algorithmic component can be decoupled from the cost function itself as one can efficiently find the optimal solution to arbitrary precision. Such methods are easier to analyze and compare against each other.

One way to obtain a convex formulation is by **convex relaxation**, either of the objective, or the constraint set. Chief among those methods are the ones based on Semidefinite Programming (SDP). For example, [Bie and Cristianini, 2004] starts with the dual of (2.15), which adds to (2.7) a constraint of the form $Y_{ij} \in \{+1, -1\}$ on the label matrix $Y = (y_i y_j)_{ij}$, and relax it into a semidefinite constraint $Y \succeq 0$ plus other linear constraints.

In **graph-based** semisupervised algorithms, nodes represent labeled and unlabeled examples and edges $w_{ij}$ represent weighted must-link constraints between examples. In the mincut formulation of [Blum and Chawla, 2001], they optimize an objective equivalent to

$$\min_{y' \in \{+1,-1\}^{m'}} \sum_{i'j'} w_{ij}(y'_{i'} - y'_{j'})^2 + \sum_{ij'} w_{ij'}(y_i - y'_{j'})^2 \tag{2.20}$$

which can be solved with graph mincut. The Gaussian random fields method of [Zhu et al., 2003] solves a related “softer” problem allowing $y_{ij} \in \Re$, which can be solved in closed-form with a linear system, while the manifold regularization of [Belkin et al., 2006] allows for more general losses and out-of-sample predictions. Those methods can be extended to multiclass setting, using multiway graphcuts [Boykov et al., 2001], multi-label random walker segmentation [Grady and Funka-Lea, 2004], or approaches based on spectral clustering[De Bie et al., 2004].

A straightforward way to encode a **must-link** constraint between $x_i$ and $x_j$ is a penalty of the form

$$w_{ij} \sum_a \psi(g^a(x_i) - g^a(x_j)) + \psi(g^a(x_j) - g^a(x_i)) \tag{2.21}$$

where $\psi(\cdot)$ is some convex loss, with a one-versus-all representation $(g^a)_a$ of the classifier. However, while incorporating **must-not-link** constraints is relatively straightforward in the binary setting (replacing the penalty by $w_{ij}(\psi(g(x_i) + g(x_j)) + \psi(-g(x_j) - g(x_i)))$), it gets tricky in the multiclass setting. The root of the difficulty can be summarized by the following misconception: *"The enemy of my enemy is my friend"*, which does not always hold in a multipolar society! More precisely, at least one of those equalities for $a \in \mathcal{Y}$ will hold: $g^a(x_i)g^a(x_j)) \geq 0$ unless $g^a(x_i)$ or $g^a(x_j)$ is positive for more than one label $a \in \mathcal{Y}$. In either case we will incur some penalty in the loss function. [Yan et al., 2004] proposes to address this problem with a *one-sided penalty*, of the form

$$w_{ij} \sum_a (\psi(-g^a(x_i) - g^a(x_j)), \tag{2.22}$$

penalizing the only case when *both* terms $g^a(x_i), g^a(x_j)$ are large. They demonstrate the use of such constraints in a surveillance video person classification task with simple temporal constraints. Another paper [Goldberg et al., 2007] uses the same form of must-not-link constraint to determine political affiliation of users in a discussion board, with quoting another user's post as a potential source of dissimilarity constraints.

## 2.4 Multiple instance Learning

The semisupervised learning framework still requires at least a few examples to be labeled. Multiple Instance Learning (MIL) goes a step further by assuming examples are not individually labeled but grouped into sets $S_i$ (called bags) which either contain at least 1 positive example, or only negative examples. The first instances of MIL was introduced in [Dietterich et al., 1997] for a drug activity prediction, where positive molecules contain at least one binding shape, and negative molecules contain no binding shape. It has since then been applied to a variety of problems including Content Based Image Retrieval (CBIR), object detection[Maron and Lozano-Pérez, 1998, Viola et al., 2006] and to some extent [Fergus et al., 2007, Crandall and Huttenlocher, 2006], text classification to name a few. Several approaches have been proposed, which [Gehler and Chapelle, 2007] classifies into 3 categories: the first category treats the problem directly at the bag-level instead of the instance level, using Hausdorff type distances to compare bags [Wang and Zucker, 2000]. The second category assumes that all instances in a positive bag are either positive or negative instances, and optimizes over their labels. For example, mi-SVM[Andrews et al., 2003] attempts to solve the following problem:

$$\min_{\mathbf{w},b,\xi,y'} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{2.23}$$

$$\text{s.t.} \quad \xi_i \geq 0, \quad y_i(\mathbf{w}\cdot f(x_i) - b) \geq 1 - \xi_i \tag{2.24}$$

$$\sum_i \frac{y_i + 1}{2} \geq 1 \text{ for a positive bag, } y_i = -1 \text{ for a negative bag} \tag{2.25}$$

$$y_i \in \{+1, -1\} \tag{2.26}$$

The problem is combinatorial in nature, as was TSVM, and heuristics need to be employed. The assumption about labeling each instance may be invalid for certain object detection tasks where a given instance (patch) in a positive bag (image) near the object of interest does not exactly fit either the foreground or background category. A third category seeks to identify which one of the instances in a positive bag is positive, without forcing the other ones to be negative. This is usually carried out through a combinatorial formulation, or a continuous relaxation that searches over the convex hull of the instances. For example, the authors of [Fung et al., 2007] claim they are guaranteed to find the global optimum of a related Convex Hull MIL problem, but it seems the optimization problem they solve in equation (3), page 3 is non-convex, with a *quadric* term hidden in the objective. In any case, such MIL algorithms are essentially intractable and suffer from existence of local minima.

## 2.5 Multi-Label Learning

Multi-Label learning breaks the assumption that labels are mutually exclusive, and in fact allows for correlation between them. Such problems can generally be considered as fully supervised. Originally motivated by text classification and medical diagnosis, where a document could be classified according to multiple interrelated criteria, it has since then been applied to a variety of problems such as protein function classification, music categorization and scene classification. There are multiple ways to convert a Multi-Label problem to a supervised problem, for example the power set representation may be desirable if the set of labels is not too large. A more straightforward method is to learn independent binary classifiers for each label, which has been used for scene classification[Boutell et al., 2004], document classification [Goncalves and Quaresma, 2003] and music categorization[Li and Ogihara, 2003]. A number of boosting variants address Multi-Label learning, for example Adaboost.MH, Adaboost.M2, SAMME[Schapire and Singer, 1999, Schapire and Freund, 1997, Zhu et al., ] propose different encodings of supervisory information.

In contrast several approaches[Qi et al., 2007, Wang et al., 2008, Godbole and Sarawagi, 2004, Schapire, 1997] explicitly use correlation between labels to improve classification accuracy. [Godbole and Sarawagi, 2004] uses a form of stacking[Wolpert, 1992] to exploit potential correlation between class labels, by augmenting the input features with class predictions of single-label classifiers. Those classifiers are initially trained on each label separately, and iteratively updated. Label correlation is also exploited in a boosting variant[Schapire, 1997], where output codes are used to ease the task of each weak learner.

## 2.6 Ambiguous Learning

In ambiguous learning, each example again is supplied with a bag of multiple potential labels, but *only one of them is correct*. In contrast with Multi-Label learning, this is a form of *weak* supervision, where potentially no single example is fully labeled. This ambiguous form of supervision is the topic of chapter 4, in which we show both theoretically and in practice that learning is possible even when all the training examples are ambiguously labeled. This problem is loosely related to another weakly supervised task, Multiple Instance Learning with a few notable differences: 1) because of mutual exclusion of labels, there is *exactly one* correct label in each bag (instead of *at least one example in each multi instance bag*), 2) the problem is inherently multi-class instead of binary, 3) some examples may have a bag containing all labels, which means it has no label information as in semisupervised learning.

## 2.7 Unsupervised Learning

The general goal of unsupervised learning is to discover hidden structures underlying observed data. The precise definition differs according to the modeling assumptions, for example generative models seek to optimize the data likelihood, while an information theoretic criterion is redundancy reduction via efficient coding, such as sparse coding, minimum entropy, minimum description length [Barlow, 1989]. Two common problematics are dimensionality reduction and clustering, both of which are well studied problems. Instead of giving an overview of the huge amount of literature, we will mention some of our previous work in image segmentation and two applications related to sequence clustering for video analysis which we discuss in this thesis. We propose in [Cour et al., 2005a] a segmentation algorithm that operates on multiple scales *simultaneously*. The algorithm, motivated by ecological statistics on natural images, compresses long-range interactions between image pixels, thereby allowing for large-scale image segmentation in *linear time* without loss of fine-grain accuracy. In [Cour and Shi, 2007a] we address the problem of object specific segmentation, combining bottom-up grouping information in the form of superpixels, and top-down modeling via an *articulated* object model. Our algorithm simultaneously searches for the *optimum combination of superpixels* composing the foreground object, jointly with the space of model configurations. We show how exhaustive search can be carried out efficiently for this seemingly combinatorial problem, and produces a short-list of accurate segmentations that can be used by a reranker. We also propose in [Cour et al., 2005b] a learning algorithm for spectral graph clustering, that allows for direct supervised learning of graph structures using training examples of segmentations. In contrast to previous work, the algorithm works directly on the segmentation eigenvectors, for which we provide *closed form derivatives* using the implicit function theorem, as well as a convergence analysis.

In chapter 3 we introduce a novel hierarchical generative model for video deconstruction, revealing the coarse-level structure composed of a hierarchy of scenes, threads, shots and images. We show how a detailed analysis of movie structure helps for video retrieval tasks, person tracking and recognition as well as activity recognition. In chapter 8 we seek a finer-grain representation of movies, with the task of clustering (and subsequently naming) people's faces across a movie. We propose a novel temporal grouping model that partitions face tracks across multiple shots while respecting appearance, geometric and film-editing cues and constraints. We present dynamic programming inference and discriminative learning for the model, where states represent local partitions.

# Chapter 3

# Coarse-Level Alignment of Video and Text Transcriptions

## 3.1 Introduction

Hand-labeling images of people and objects is a laborious task that is difficult to scale up. Several recent papers [Huang et al., 2007a, Ramanan et al., 2007] have successfully collected very large-scale, diverse datasets of faces "in the wild" using weakly supervised techniques. These datasets contain a wide variation in subject, pose, lighting, expression, and occlusions which is not matched by any previous hand-built dataset. Labeling and segmenting actions is perhaps an even more painstaking endeavor, where curated datasets are more limited. Automatically extracting large collections of actions is of paramount importance. In this chapter, we argue that using movies and TV shows *precisely* aligned with easily obtainable screenplays can pave a way to building such large-scale collections. Figure 3.1 illustrates this goal, showing the top 6 retrieved video snippets for 2 actions (walk, turn) in TV series LOST using our system. The screenplay is parsed into a temporally aligned sequence of action frames (subject verb object), and matched to detected and named characters in the video sequence. Simultaneous work[Laptev et al., 2008] explores similar goals in a more supervised fashion. In order to enable accurately localized action retrieval, we propose a much deeper analysis of the structure and syntax of both movies and transcriptions.

Movies, TV series, news clips, and nowadays plentiful amateur videos, are designed to effectively communicate events and stories. A visual narrative is conveyed from multiple camera angles that are carefully composed and interleaved to create seamless action. Strong coherence cues and continuity editing rules are (typically) used to orient the viewer, guide attention and help follow the action and geometry of the scene. Video shots, much like words in sentences and paragraphs, must fit together to minimize perceptual discontinuity across cuts and produce a meaningful scene. We attempt to uncover elements of the inherent structure of scenes and shots in video narratives. This uncovered structure can be used to analyze the content of the video for tracking objects across cuts, action retrieval, as well as enriching browsing and editing interfaces.

We present a framework for automatic parsing of a movie or video into a hierarchy of shots and scenes and recovery of the shot interconnection structure. Our algorithm makes use of both the input image sequence, closed captions and the screenplay of the movie. We assume a hierarchical organization of movies into shots, threads and scenes, where each scene is composed of a set of interlaced threads of shots with smooth transitions of camera viewpoint inside each thread (see figure 3.2). To model the scene structure, we propose a unified generative model for joint scene segmentation and shot threading. We show that inference in the model to recover latent structure amounts to finding a Hamiltonian path in the sequence of shots that maximizes the “head to tail” shot similarity along the path, given the scene boundaries. Finding the maximum weight Hamiltonian path (reducible to the Traveling Salesman Problem or TSP) is intractable in general, but in our case, limited memory constraints on the paths make it tractable. In fact we show how to jointly optimize scene boundaries and shot threading in *linear time* in the number of shots using a novel hierarchical dynamic program.

We introduce textual features to inform the model with scene segmentation, via temporal alignment with screenplay and closed captions, see figure 3.3. Such text data

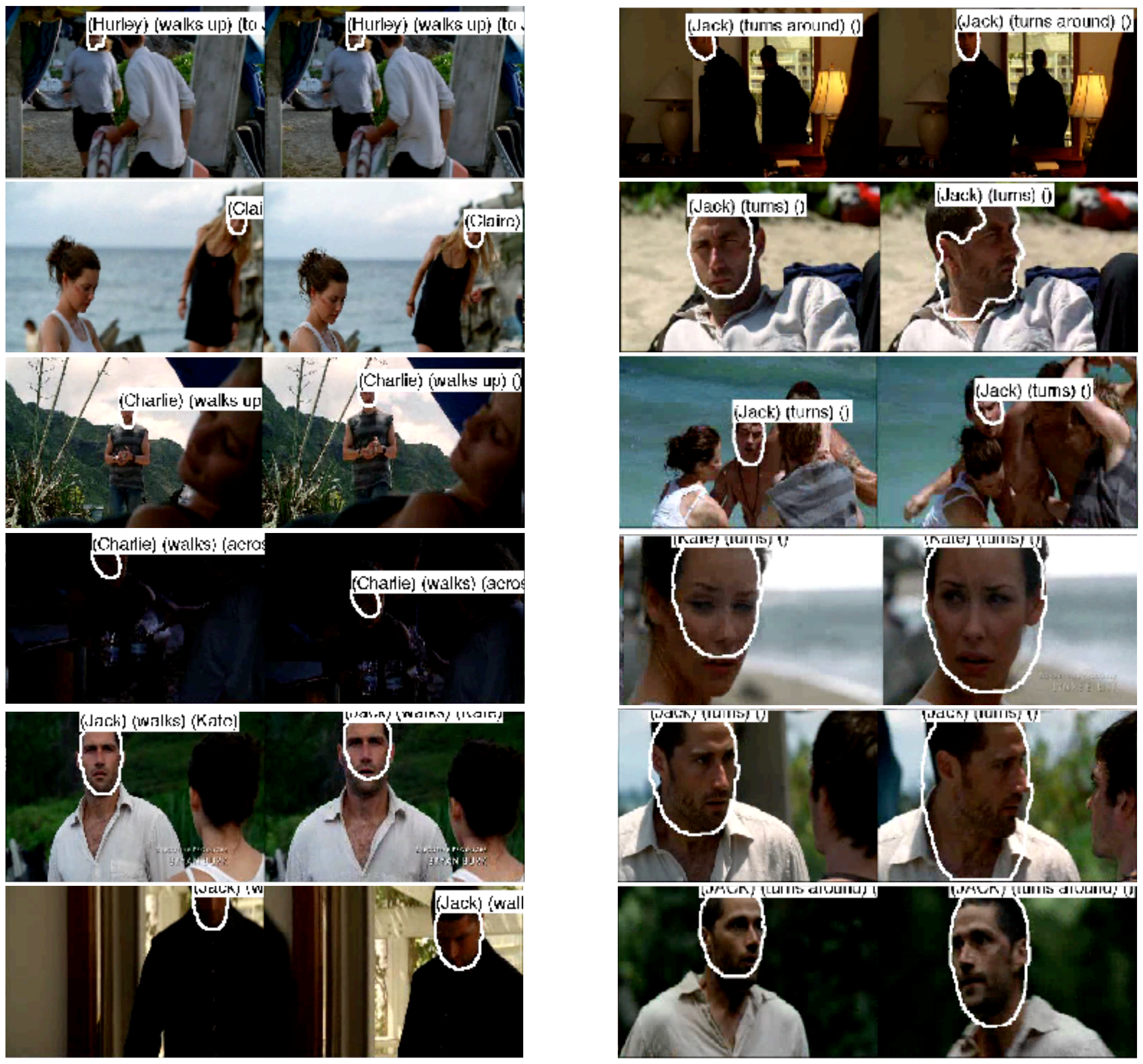


Figure 3.1: Action retrieval using alignment between video and parsed screenplay. For each action verb (left: walk, right: turn), we display the top 6 retrieved video snippets in TV series LOST using our system. The screenplay and closed captions are parsed into a temporally aligned sequence of verb frames (subject-verb-object), and then matched to detected and named characters in the video sequence. The third retrieval, second column ("Jack turns") is counted as an error, since the face shows Boone instead of Jack. Additional results appear under `www.seas.upenn.edu/~timothee`.

has been used for character naming [Sivic et al., 2005, Everingham et al., 2006] and is widely available, which makes our approach applicable to a large number of movies and TV series. In order to retrieve temporally-aligned actions, we delve deeper into resolving textual ambiguities with pronoun resolution (determining whom or what ‘he’, ‘she’, ‘it’, etc. refer to in the screenplay) and extraction of verb frames. By detecting and naming characters, and resolving pronouns, we show promising results for more accurate action retrieval for several common verbs. We present quantitative and qualitative results for scene segmentation/alignment, shot segmentation/threading, tracking and character naming across shots and action retrieval in numerous episodes of popular TV series, and illustrate that shot reordering provides much improved character naming.

The main contributions of the chapter are: 1) novel probabilistic model and inference procedure for shot threading and scene alignment driven by text, 2) extraction of verb frames and pronoun resolution from screenplay, and 3) retrieval of the corresponding actions informed by scene structure and character naming.

The chapter is organized as follows. Section 3.2 proposes a hierarchical organization of movies into shots, threads and scenes. Sections 3.3 and 3.4 introduce a generative model for joint scene segmentation and shot threading, and a hierarchical dynamic program to solve it as a restricted TSP variant. Section 3.5 addresses the textual features used in our model. We report results in section 3.6 and conclude in section 3.7.

## 3.2 Movie elements: shots, threads, scenes

Movies and TV series are organized in distinctive hierarchical and continuity structures consisting of elements such as scenes, threads and shots. Detecting and recovering these elements is needed for uninterrupted tracking of objects and people in a scene across multiple cameras, recovering geometric relationships of objects in a scene, intelligent video browsing, search and summarization.

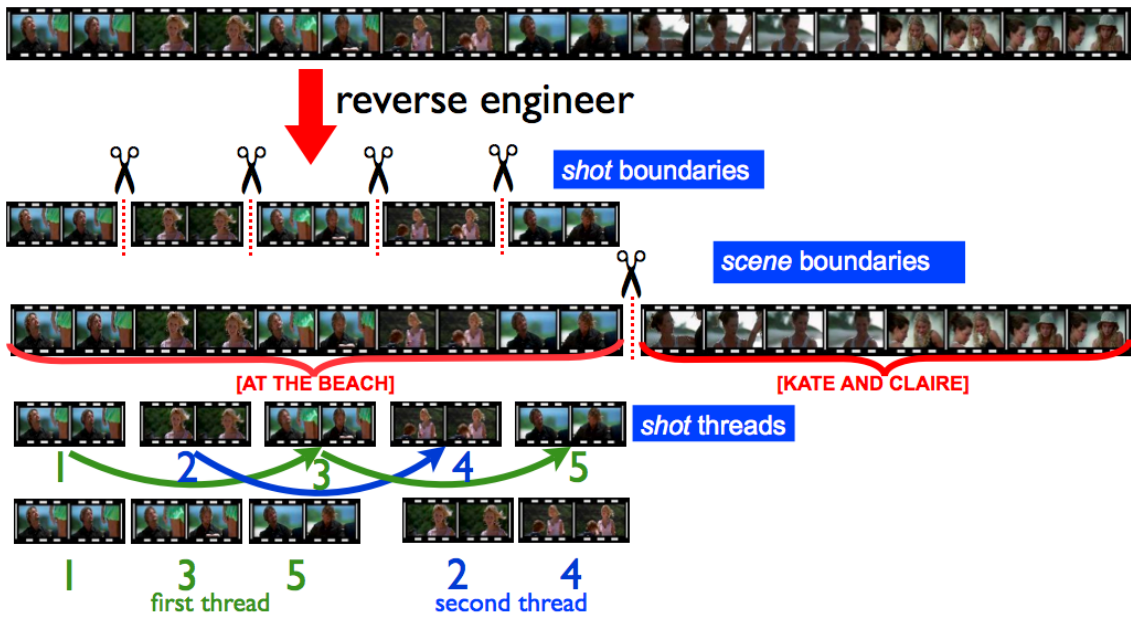


Figure 3.2: Deconstruction pipeline.

**Shot boundaries.** The aim of shot segmentation is to segment the input frames into a sequence of shots (single unbroken video recordings) by detecting camera viewpoint discontinuities. A popular technique is to compute a set of localized color histograms for each image and use a histogram distance function to detect boundaries [Lienhart, 2001, Ngo et al., 2001].

**Shot threads.** Scenes are often modeled as a sequence of shots represented as letters: **ABABAB** represents a typical dialog scene alternating between two camera points of view A and B. More complex patterns are usually observed and in practice, the clustering of the shots into letters (camera angles/poses) is not always a very well defined problem, as smooth transitions between shots occur. Nevertheless we assume in our case that each shot in a scene is either a novel camera viewpoint or is generated from (similar to) a previous shot in the scene. This makes weaker assumptions about the scene construction and doesn't require reasoning about the number of clusters. In the example above, the

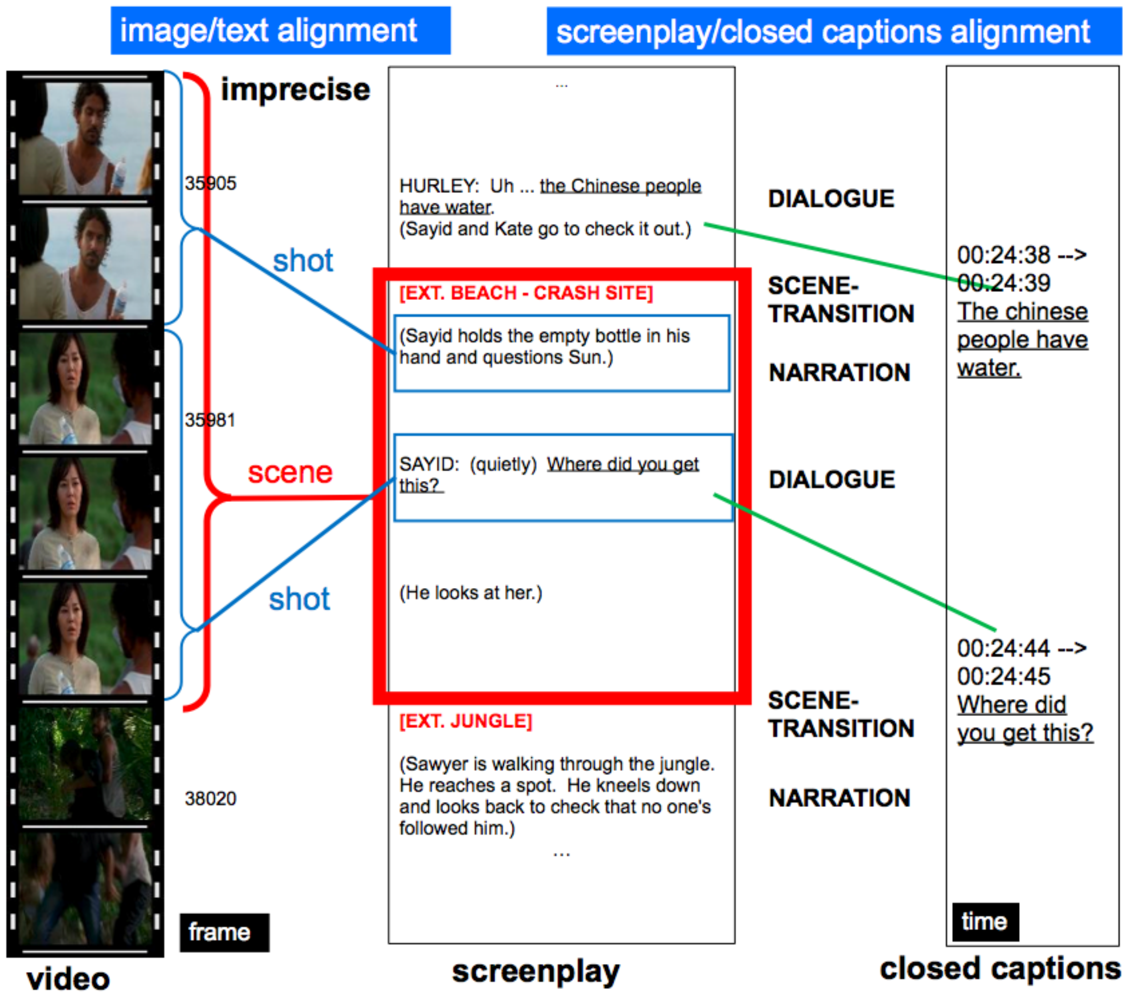

image/text alignment
screenplay/closed captions alignment
imprecise
35905
shot
35981
scene
shot
38020
frame
video
...
HURLEY: Uh ... the Chinese people have water.
(Sayid and Kate go to check it out.)
[EXT. BEACH - CRASH SITE]
(Sayid holds the empty bottle in his hand and questions Sun.)
SAYID: (quietly) Where did you get this?
(He looks at her.)
[EXT. JUNGLE]
(Sawyer is walking through the jungle. He reaches a spot. He kneels down and looks back to check that no one's followed him.)
...
screenplay
DIALOGUE
SCENE-TRANSITION
NARRATION
DIALOGUE
SCENE-TRANSITION
NARRATION
00:24:38 --> 00:24:39
The chinese people have water.
00:24:44 --> 00:24:45
Where did you get this?
time
closed captions

Figure 3.3: Alignment between video, screenplay and closed captions

first A and B are novel viewpoints, and each subsequent A and B is generated by the previous A or B. Figure 3.6 shows a more complex structure.

**Scene boundaries.** A scene consists of a set of consecutive semantically related shots (coherence in action, location and group of actors is typical). The process of segmenting a video sequence into scenes has received some attention in the video analysis literature [Ngo et al., 2001]. An MCMC based clustering framework is used in [Zhai and Shah, 2006]. Hierarchical clustering on a shot connectivity graph is proposed in [Yeung et al., 1998]. In [Kender and Yeo, 1998], the authors detect scene boundaries as local minima of a backward shot coherence measure. As opposed to shot boundaries, which correspond to strong visual discontinuity in consecutive frames, scene boundaries are not detectable from purely local cues: the entire sequence of preceding and following shots must be considered. For example, **ABCBABDEFEABD** shot sequence is one scene, while **ABCBAB DEFEDEF** can be two.

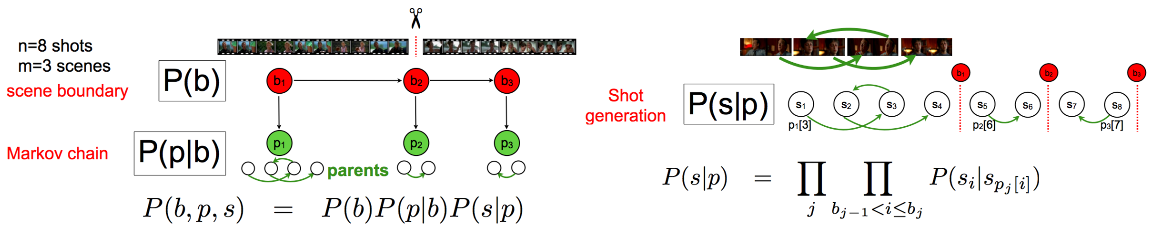
## 3.3 A (simple) generative model of movies

To capture the hierarchical and continuity structure, we propose a simple generative model, where scenes are constructed independently of other scenes, while shots within a scene are produced via an interleaved Markov (first order) structure.

We begin with some notation to define our model, assuming the video sequence has already been segmented into shots:

- $s_i$: $i^{th}$ shot (interval of frames), with $i \in [1, n]$
- $b_j$: $j^{th}$ scene boundary (index of its *last* shot), with $j \leq m;\ 1 \leq b_1 < ... < b_m = n$
- $p_j[i]$: parent generating shot $i$ in scene $j$ (could be NULL), with $j \leq m, i \leq n$.

We assume the shots in a video sequence are generated as follows: first generate the sequence of scene boundaries $(b_j)$, then generate for each scene $j$ a dependency structure



$$P(b,p,s) = P(b)P(p|b)P(s|p)$$

$$P(s|p) = \prod_j \prod_{b_{j-1}<i\leq b_j} P(s_i|s_{p_j[i]})$$

Figure 3.4: Graphical model for joint scene segmentation and shot reordering, see text for details.

$p_j$ defining a Markov chain on shots, and finally generate each shot $i$ given its parent $p_j[i]$. The model is conditioned upon $m$ and $n$, assumed to be known in advance. This can be represented using the generative model in figure 3.4. For the **scene boundary model** $P(b)$, we investigate both a uniform model and an improved model, where scene boundaries are informed by the screenplay (see section 3.5). The **shot threading model** $P(p|b)$ is uniformly distributed over valid Markov chains (shot orderings) on each scene. The **shot appearance model** $P(s_i|s_{p_j[i]})$ is treated next (we set it to uniform for the root of scene $j$ where $p_j[i] = \text{NULL}$). This model encourages (1) smooth shot transitions within a scene and (2) scene breaks between shots with low similarity, since the model doesn't penalize transitions across scenes.

**Shot appearance model ($P(s_{i'}|s_i)$).** In order to obtain smooth transitions and allow tracking of objects throughout reordered shots, we require that $P(s_{i'}|s_i)$ depends on the similarity between the *last* frame of shot $s_i$ ($I = s_i^{\text{last}}$) and the *first* frame of shot $s_{i'}$ ($I' = s_{i'}^{\text{first}}$). Treating each shot as a word in a finite set, we parametrize the **shot similarity term** as $P(s_{i'}|s_i) = \exp(-d_{\text{shot}}(s_i, s_{i'}))/\sum_{i''}\exp(-d_{\text{shot}}(s_i, s_{i''}))$ where $d_{\text{shot}}(s_i, s_{i'}) = d_{\text{frame}}(I, I')$ is the chi-squared distance in color histogram between frames $I, I'$. Note, $d_{\text{shot}}(s_i, s_{i'})$ is *not symmetric*, even though $d_{\text{frame}}(I, I')$ is.

## 3.4 Inference in the model

In this section we attempt to solve the Maximum a Posteriori (MAP) problem in figure 3.4. Let us first consider the simplified case without scene transitions (when $m = 1$). In this case, maximizing the log becomes:

$$\max_{p:\text{Markov Chain}} \sum_i W_{i,p[i]} = \max_{\pi \in \mathcal{P}_{[1,n]}} \sum_t W_{\pi_{t-1},\pi_t} \tag{3.1}$$

where $W_{ii'} = \log P(s_{i'}|s_i)$ and $\pi \in \mathcal{P}_{[1,n]}$ denotes a permutation of $[1,n]$ defined recursively from the parent variable $p$ as follows: $p[\pi_t] = \pi_{t-1}$, with $\pi_1$ indicating the root. This amounts to finding a maximum weight Hamiltonian Path or **Traveling Salesman Problem** (TSP), with $\pi_t$ indicating which shot is visited at time $t$ on a virtual tour. TSPs are *intractable in general*, so we make one additional assumption restricting the set of feasible permutations.

### 3.4.1 Memory-limited TSPs

Given an integer $k > 0$ (memory width), and an initial ordering of shots (or cities by analogy to TSP) $1, ..., n$, we introduce the following limited memory constraint on our hamiltonian path $\pi = (\pi_t)$:

$$\mathcal{P}^k_{[1,n]} = \{\pi \in \mathcal{P}_{[1,n]} : \forall (i,i') i' \geq i + k \Rightarrow \pi_{i'} > \pi_i\} \tag{3.2}$$

This is illustrated in figure 3.5 for $k = 2$ ($k = 1$ means $\pi$ is the identity, and $k = n$ is fully unconstrained). There are two important consequences: (1) the MAP becomes tractable (*linear complexity in* $n$), and (2) the problem becomes sparse, *i.e.*, we can restrict W.L.O.G. $W$ to be sparse (banded):

$$\pi_t \in [t - (k-1), t + (k-1)] \tag{3.3}$$

$$W_{ii'} = -\infty \text{ except for } i - (2k-3) \leq i' \leq i + 2k - 1 \tag{3.4}$$

The first line comes from the pigeonhole principle, and the second one uses the first line: $-(2k-3) \leq \pi_{t+1} - \pi_t \leq 2k - 1$. Note, this constraint is natural in a video sequence, as

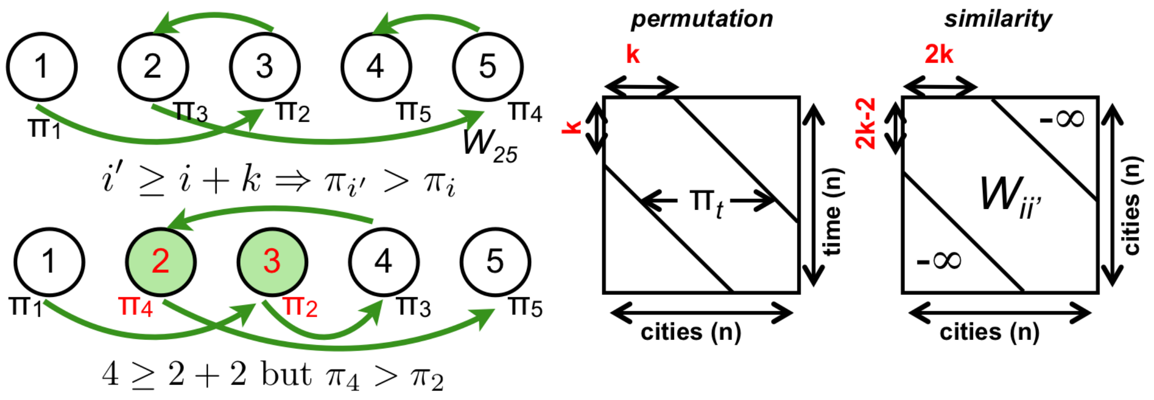


Figure 3.5: Top left: a feasible solution for the restricted TSP with $k = 2$. Bottom left: an infeasible solution, violating the precedence constraint (shaded cities). Middle: the constraint limits the range of the permutation: $\pi_t \in [t - (k-1), t + (k-1)]$. Right: the constraint implies a banded structure on the similarity matrix $W = (W_{ii'})$: $i - (2k-3) \leq i' \leq i + 2k - 1$.

video editing takes into account the limited memory span of humans consisting of a few consecutive shots.

### 3.4.2 Dynamic Programming solution without scene breaks ($P(p, s)$)

The solution to the simplified problem without scene breaks (3.1) under constraint (3.2) has been addressed in [Balas and Simonetti, 2001] (it dealt with a hamiltonian *cycle* with $\pi_1(1) = 1$, but this is easily adaptable to our case). We summarize the main points below. Let $C_t(S, i')$ be the optimal cost of the paths $\pi \in \mathcal{P}^k_{[1,n]}$ satisfying $\pi_t = i'$ and $\{\pi_1, ..., \pi_{t-1}\} = S$ (set of cities visited before time t). The dynamic programming solution uses the relation:

$$C_t(S, i') = \min_{i \in S} C_{t-1}(S - \{i\}, i) + W_{ii'} \tag{3.5}$$

Because of the precedence constraint, the pair $(S, i')$ can take at most $(k+1)2^{k-2}$ possible values at any given time $t$ (instead of $\binom{n-1}{t-1}n$ without the constraint). The idea is to construct a directed weighted graph $G_n^k$ with $n$ layers of nodes, one layer per position in the path, with paths in the graph joining layer 1 to layer $n$ corresponding to feasible hamiltonian paths, and shortest paths joining layer 1 to $n$ corresponding to optimal hamiltonian paths. Since there are at most $k$ incoming edges per node (corresponding to valid transitions $\pi_{t-1} \to \pi_t$), the total complexity of the dynamic program is $O(k(k+1)2^{k-2} \cdot n)$, exponential in $k$ (fixed) but *linear in* $n$, see [Balas and Simonetti, 2001] for details.

### 3.4.3 Dynamic Programming solution with scene breaks $(\boldsymbol{P(b,p,s)})$

The general problem can be rewritten as:

$$\max_b \sum_j \max_{\pi \in \mathcal{P}^k_{(b_{j-1}, b_j]}} \sum_t W_{\pi_{t-1}, \pi_t} \tag{3.6}$$

**Naive solution.** One can solve (3.6) as follows: for each interval $\mathcal{I} \subset [1, n]$, precompute the optimal path $\pi^*_{\mathcal{I}} \in \mathcal{P}^k_{\mathcal{I}}$ using 3.4, and then use a straightforward dynamic programming algorithm to compute the optimal concatenation of $m$ such paths to form the optimal solution. Letting $f(k) = k(k+1)2^{k-2}$, the complexity of this algorithm is $O(\sum_{1 \le i \le i' \le n} f(k) \cdot (i' - i + 1)) = O(f(k)n(n+1)(n+2)/6)$ for the precomputation and $O(mn(n+1)/2)$ for the dynamic program, which totals to $O(f(k)n^3/6)$. The next paragraph introduces our joint dynamic programming over scene segmentation and shot threading, which reduces computational complexity by a factor $n$ (number of shots).

**Joint dynamic program over scene breaks and shot threading.** We exploit the presence of overlapping subproblems. We construct a *single* tour $\pi$, walking over the joint space of shots and scene labels. Our approach is based on the (categorical) **product graph** $G_n^k \times C_m$ where $G_n^k$ is the graph from 3.4.2 and $C_m$ is the chain graph of order $m$.

A node $(u, j) \in G_n^k \times C_m$ represents the node $u \in G_n^k$ in the $j^{th}$ scene. Given two *connected* nodes $u = (S, i, t)$ and $u' = (S', i', t+1)$ in $G_n^k$, there are two types of connections in the product graph. The first connections correspond to shots $i, i'$ both being in the $j^{th}$ scene:

$$(u, j) \to (u', j), \text{ with weight } W_{ii'} \tag{3.7}$$

The second connections correspond to a scene transition:

$$(u, j) \to (u', j+1), \text{ with weight } 0, \tag{3.8}$$

and only happen when $u = (S, i, t)$ satisfies $\max(i, \max(S)) = t$, to make sure the tour decomposes into a tour of each scene (we can switch to the next scene when the set of shots visited up to time $t$ is exactly $\{1, ..., t\}$).

The solution to (3.6) similarly uses a dynamic program to find the shortest path in $G_n^k \times C_m$ (and backtracking to recover the $\arg\max$). Since there are $m$ times as many nodes in the graph as in $G_n^k$ and at most twice as many incoming connections per node (nodes from the previous scene or from the same scene), the total complexity is: $O(2k(k+1)2^{k-2}mn) = O(2f(k)mn)$.

**Comparison.** We manually labeled shot and scene breaks for a number of movies and TV series and found that a typical scene contains on average about 11 shots, *i.e.*$m \approx n/11$. So the reduction in complexity between the naive algorithm and our joint dynamic program is: $O(\frac{f(k)n^3/6}{2f(k)mn}) = O(n^2/(12m)) \approx n$, which is a huge gain, especially given typical values of $n = 600$. The resulting complexity is linear in $n$ and $m$ and in practice takes about 1 minute as opposed to 11 hours for an entire episode, given pre-computed shot similarity.

## 3.5 Scene segmentation via coarse image to text alignment ($P(b)$)

We now assume we have some text data corresponding to the movie sequence, and we focus on simultaneously segmenting/threading the video into scenes and aligning the text with the video. The extra text media removes a lot of ambiguity for the scene segmentation

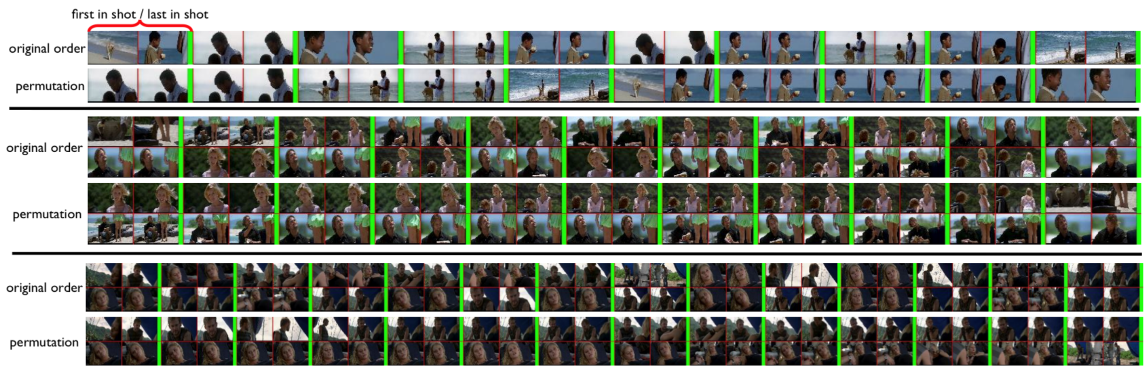


Figure 3.6: Shot reordering to recover continuity in 3 scenes of LOST.

and, combined with our model, leads to improved scene segmentation results as we shall see in section (3.6).

### 3.5.1 Text data: screenplay and closed captions

We use two sources of text for our segmentation-alignment problem: the screenplay, which narrates the actions and provides a transcript of the dialogs, and the closed captions, which provide time-stamped dialogs, as in figure 3.3. Both sources are essential since the screenplay reveals speaker identity, dialogs and scene transitions but no time-stamps, and closed captions reveal dialogs with time-stamps but nothing else. See the appendix for details on how to obtain screenplays and closed captions.

### 3.5.2 Screenplay/closed captions alignment

The alignment between the screenplay and the closed captions is non-trivial since the closed captions only contain the dialogs (without speaker) mentioned in the screenplay, often with wide discrepancies between both versions. We extend the dynamic time warping[Myers and Rabiner, 1981] approach in a straightforward way to time-stamp each element of the screenplay (as opposed to just the dialogs as in [Everingham et al., 2006]).

The screenplay is first parsed into a sequence of elements (either NARRATION, DIALOG, or SCENE-TRANSITION) using a simple grammar, and the dynamic programming alignment of the words in the screenplay and the closed captions provides a time interval $[T^{\text{start}}(i), T^{\text{end}}(i)]$ for each DIALOG element $E_i$. A NARRATION or SCENE-TRANSITION element $E_j$ enclosed between two DIALOG elements $E_{i_1}, E_{i_2}$ is assigned the following conservative time interval: $[T^{\text{start}}(i_1), T^{\text{end}}(i_2)]$.

### 3.5.3 Scene segmentation via alignment

We determine the scene boundary term $P(b)$ from section 3.3 by aligning each SCENE-TRANSITION element mentioned in the screenplay to a scene start. $P(b)$ is uniform among the set of $b$ satisfying the temporal alignment constraints:

$$1 \leq b_1 < ... < b_m = n \tag{3.9}$$

$$t^{\text{start}}(j) \leq b_{j-1} + 1 \leq t^{\text{end}}(j) \tag{3.10}$$

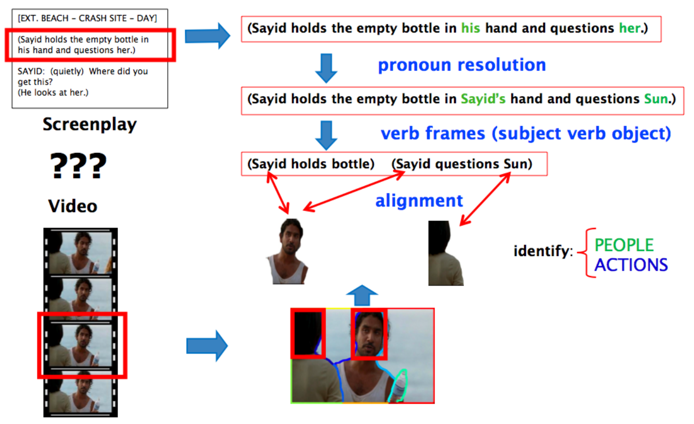
where $[t^{\text{start}}(j), [t^{\text{end}}(j)]$ is the time interval of the $j^{th}$ SCENE-TRANSITION element, converted into frame numbers, then to shot indexes.

**Additional alignment constraints.** Close inspection of a large number of screenplays collected for movies and TV series revealed a fairly regular vocabulary used to describe shots and scenes. One such example is FADE IN and FADE OUT corresponding to a transition between a black shot (where each frame is totally black) and a normal shot, and vice versa. Such black shots are easy to detect, leading to additional constraints in the alignment problem, and a performance boost.

### 3.5.4 Pronoun resolution and verb frames

Alignment of the screenplay to dialog in closed captions and scene boundaries in the video helps to narrow down the scope of reference for other parts of the screenplay that are interspersed – the narration or scene descriptions, which contain mentions of actions and objects on the screen. In addition to temporal scope uncertainty for these descriptions,

there is also ambiguity with respect to the subject of the verb, since personal pronouns (he, she) are commonly used. In fact, our analysis of common screenplays reveals there are more pronouns than occurrences of character names in the narrations, and so resolving those pronouns is an important task. We employed a simple, deterministic scheme for pronoun resolution that uses a standard probabilistic context-free parser to analyze sentences and determine verb frames (subject-verb-object) and then scans the sentence for possible antecedents of each pronoun that agree in number and gender, see figure 3.7. The details of the algorithm are given in *supplemental materials*. Here is an example output of our implementation on a sentence extracted from screenplay narration (pronoun resolution shown in parenthesis): *On the side, Sun watches them. Jin reaches out and touches Sun 's chin, his (Jin's) thumb brushes her (Sun's) lips. She (Sun) looks at him (Jin) and pulls away a little. He (Jin) puts his (Jin's) hand down.*
Output verb frames: *(Sun - watches - something) (Jin - reaches out - ) (Jin - touches - chin) (Sun - looks - at Jin) . (Sun - pulls away - ) (Jin - puts down - hand).*
We report pronoun resolution accuracy on screenplay narrations of 3 different TV series (about half a screenplay for each), see table 3.1.


[EXT. BEACH – CRASH SITE – DAY]
(Sayid holds the empty bottle in his hand and questions her.)
SAYID: (quietly) Where did you get this?
(He looks at her.)
Screenplay
???
Video
(Sayid holds the empty bottle in his hand and questions her.)
pronoun resolution
(Sayid holds the empty bottle in Sayid's hand and questions Sun.)
verb frames (subject verb object)
(Sayid holds bottle)
(Sayid questions Sun)
alignment
identify:
PEOPLE
ACTIONS

| | |
|---|---|
| **total verbs** | 25,000 |
| distinct verbs | 1,000 |
| looks (most common) | 2,000 |
| turns | 1,100 |
| walks | 800 |
| takes | 550 |
| climbs | 40 |
| kisses | 40 |
| **total dialog lines** | 16,000 |
| distinct speaker names | 190 |
| Jack (most common) | 2,100 |

Figure 3.7: Left: pronoun resolution and verb frames obtained from the parsed screenplay narrations. Right: statistics collected from 24 parsed screenplays (1 season of LOST).

| TV series screenplay | pronoun resolution accuracy | # pronouns | # sentences |
|---|---|---|---|
| LOST | 75% | 93 | 100 |
| CSI | 76 % | 118 | 250 |
| ALIAS | 78% | 178 | 250 |

Table 3.1: Pronoun resolution accuracy on screenplay narrations of 3 different TV series.

## 3.6 Results

We experimented with our framework on a significant amount of data, composed of TV series (19 episodes from one season of LOST, several episodes of CSI), one feature length movie "The Fifth Element", and one animation movie "Aladdin", representing about 20 hours of video at DVD resolution. We report results on scene segmentation/alignment, character naming and tracking.

### 3.6.1 Shot segmentation

We obtain 97% F-score (harmonic mean of precision and recall) for shot segmentation, using standard color histogram based methods.

### 3.6.2 Scene segmentation and alignment

We hand labeled scene boundaries in one episode of LOST and one episode of CSI based on manual alignment of the frames with the screenplay. The accuracy for predicting the scene label of each shot was 97% for LOST and 91% for CSI. The F-score for scene boundary detection was 86% for LOST and 75% for CSI, see figure 3.8. We used $k = 9$ for the memory width, a value similar to the buffer size used in [Kender and Yeo, 1998] for computing shot coherence. We also analyzed the effect on performance of the memory width $k$, and report results with and without alignment to screenplay in table 3.2. In comparison, we obtained an F-score of 43% for scene boundary detection using a model based on backward shot coherence [Kender and Yeo, 1998] uninformed by screenplay, but optimized over buffer size and non-maximum suppression window size.

### 3.6.3 Scene content analysis

We manually labeled the scene layout in the same episodes of LOST and CSI, providing for each shot in a scene its generating shot (including the special case when this is a new viewpoint). We obtain a precision/recall of $75\%$ for predicting the generating parent shot. See figure 3.6 for a sample of the results on 3 scenes. Note, to obtain longer tracks in figure 3.6, we recursively applied the memory limited TSP until convergence (typically a few iterations).

### 3.6.4 Character identification on reordered shots

We illustrate a simple speaker identification based on screenplay alignment and shot threading, see figure 3.9 (more elaborate models will be proposed in later chapters). We use a Viola-Jones[Viola and Jones, 2004] based face detector and tracking with normalized cross-correlation to obtain face tracks in each shot. We build a Hidden Markov Model (HMM) with states corresponding to assignments of face tracks to character names. The face tracks are ordered according to the shot threading permutation, and as a result there are much fewer changes of character name along this ordering. Following [Everingham et al., 2006], we detect on-screen speakers as follows: 1) locate mouth for each face track using a mouth detector based on Viola-Jones, 2) compute a mouth motion score based on the normalized cross correlation between consecutive windows of the mouth track, averaged over temporal segments corresponding to speech portions of the screenplay. Finally we label the face tracks using Viterbi decoding for the Maximum a Posteriori (MAP) assignment (see website for more details). We computed groundtruth face names for one episode of LOST and compared our method against the following baseline that does not use shot reordering: each unlabeled face track (without a detected speaking character on screen) is labeled using the closest labeled face track in feature

| $P(b)$ | $k=1$ | $k=2$ | $k=3$ | $k=9$ | $k=12$ |
|---|---|---|---|---|---|
| aligned | 73/90 | 77/91 | 82/96 | 86/97 | 88/97 |
| uniform | 25/0 | 45/14 | 55/0 | 52/1 | -/- |
| total time (s) | $< 0.1$ | $< 0.1$ | 0.1 | 5 | 68 |

Table 3.2: $\%$ F-score (first number) for scene boundary detection and $\%$ accuracy (second number) for predicting scene label of shots (on 1 episode of LOST) as a function of the memory width $k$ used in the TSP, and the prior $P(b)$. The case $k = 1$ corresponds to no reordering at all. Line 1: $P(b)$ informed by screenplay; line 2: $P(b)$ uniform; line 3: total computation time.

space (position of face track and color histogram). The accuracy over an episode of LOST is $76\%$ for mainly dialogue scenes and $66\%$ for the entire episode, as evaluated against groundtruth. The baseline model based using nearest neighbor performs at resp. $43\%$ and $39\%$. The next chapters will present more elaborate and accurate models for character naming.

### 3.6.5 Retrieval of actions in videos

We consider a query-by-action verb retrieval task for 15 query verbs across 10 episodes of LOST, see figure 3.10. The screenplay is parsed into verb frames (subject-verb-object) with pronoun resolution, as discussed earlier. Each verb frame is assigned a temporal interval based on time-stamped intervening dialogues and tightened with nearby shot/scene boundaries. Queries are further refined to match the subject of the verb frame with a named character face. We report retrieval results as follows: for each of the following action verbs, we measure the number of times (out of 10) the retrieved video snippet correctly shows the actor on screen performing the action (we penalize for wrong naming): close eyes (9/10), grab (9/10), kiss (8/10), kneel (9/10), open (9/10), stand (9/10), cry (9/10), open door (10/10), phone (10/10), point (10/10), shout (7/10), sit (10/10), sleep (8/10), smile (9/10), take breath (9/10). The average is 90/100. Two additional queries are shown in figure 3.1 along with the detected and identified characters. We are creating a

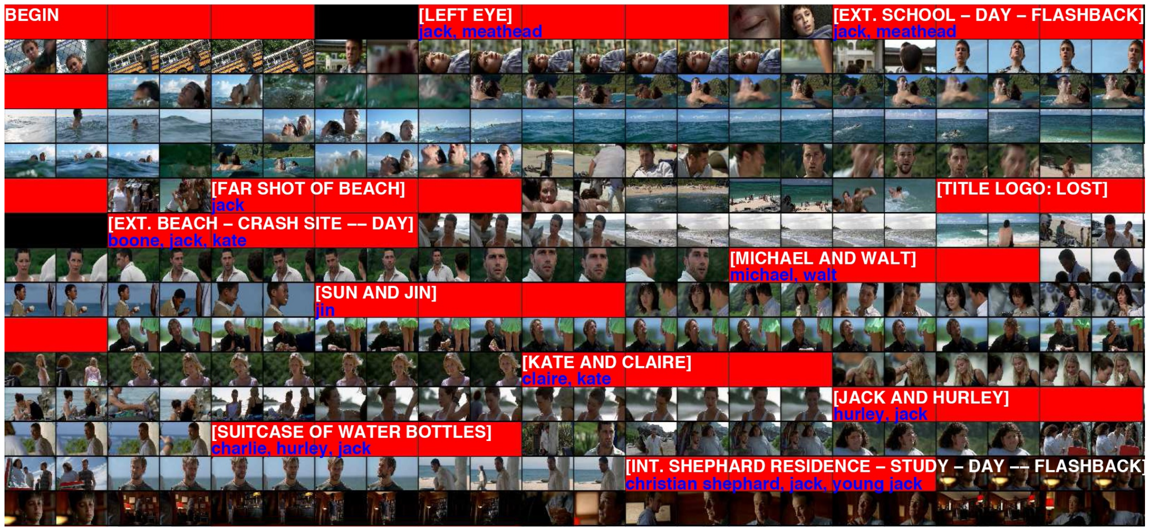


Figure 3.8: Movie at a glance: scene segmentation-alignment and shot reordering for an episode of LOST (only a portion shown for readability). Scene boundaries are in red, together with the set of characters appearing in each scene, in blue.

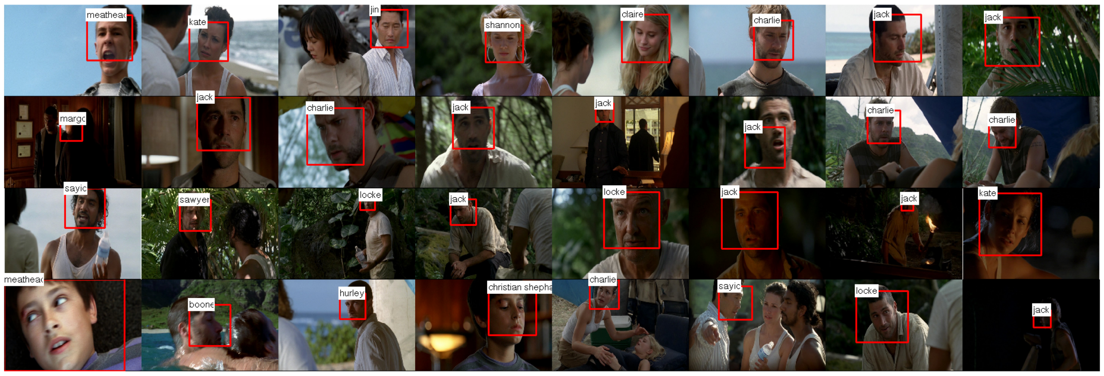


Figure 3.9: Character naming using screenplay alignment and shot threading. Top 3 rows: correctly named faces; bottom row: incorrectly named faces. We detect face tracks in each shot and reorder them according to the shot threading permutation. Some face tracks are assigned a name prior based on the alignment between dialogs and mouth motion. We compute a joint assignment of names to face tracks using an HMM on the reordered face tracks.

large dataset of retrieved action sequences combined with character naming for improved temporal and spatial localization, see `www.seas.upenn.edu/~timothee`.

## 3.7 Conclusion

In this work we have addressed basic elements of movie structure: hierarchy of scenes and shots and continuity of shot threads. We believe that this structure can be useful for many intelligent movie manipulation tasks, such as semantic retrieval and indexing, browsing by character or object, re-editing and many more. We plan to extend our work to provide more fine-grained alignment of movies and screenplay, using coarse scene geometry, gaze and pose estimation.

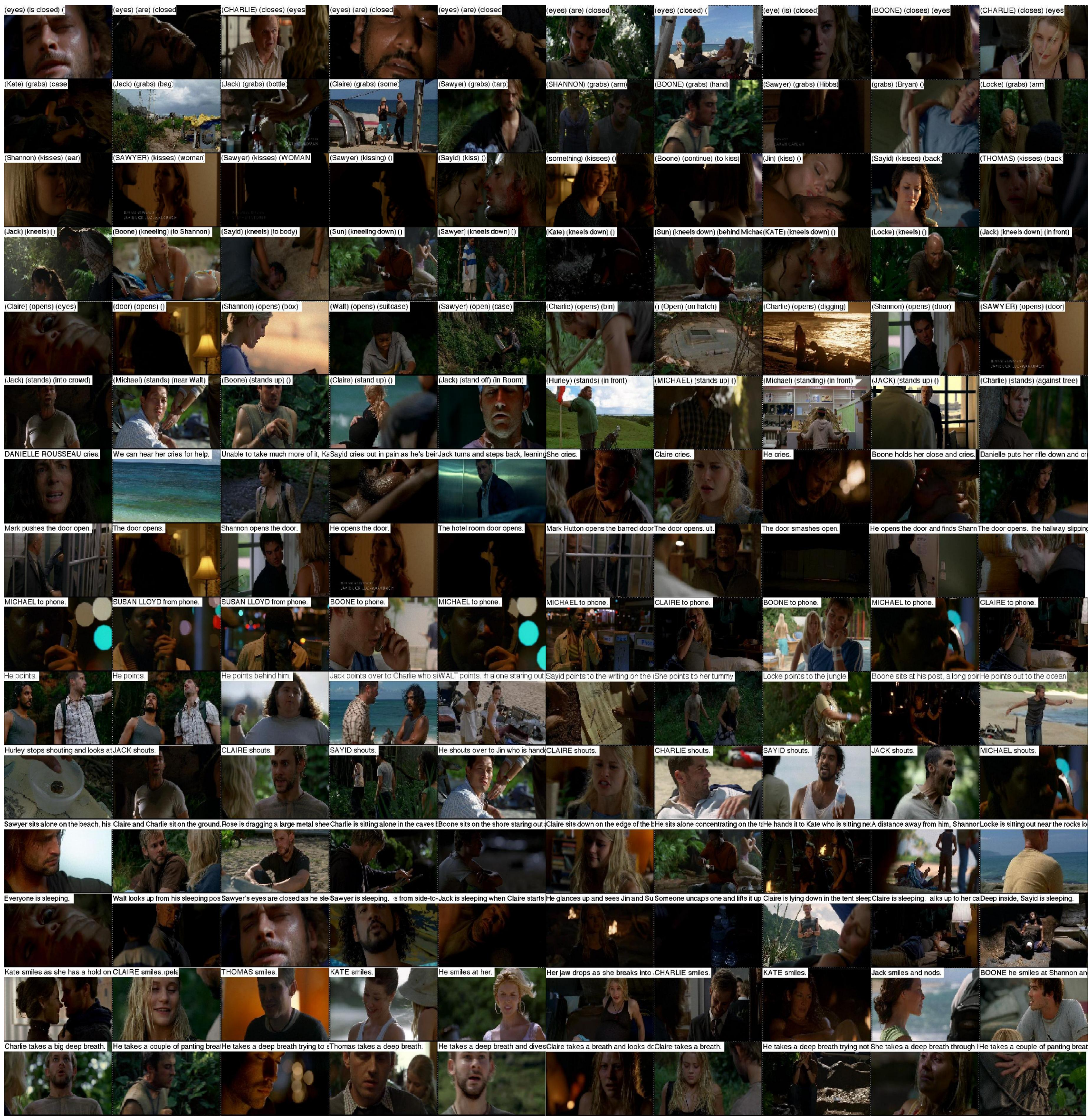


Figure 3.10: Top 10 retrieved video snippets for 15 query action verbs: close eyes, grab, kiss, kneel, open, stand, cry, open door, phone, point, shout, sit, sleep, smile, take breath. Please zoom in to see screenplay annotation (and its parsing into verb frames for the first 6 verbs).

# Chapter 4

# Learning from Ambiguously Labeled Images

## 4.1 Introduction

We consider a weakly-supervised multiclass classification setting where each instance is labeled ambiguously with more than one potential labels. A typical example arises in photograph collections containing several faces per image and a caption that only specifies who is in the picture but not which name matches which face, see Figure 4.1. As a further motivation, consider Figure 4.2, which shows another common setting where we can obtain plentiful but ambiguously labeled data: videos and screenplays. Using a screenplay, we can tell who is in the scene, but for every face in the images, the person's identity is ambiguous. The social network of characters sharing a scene in a season of LOST is shown in Figure 4.3.

Learning accurate face and object recognition models from such imprecisely annotated images and videos can improve many applications, including image retrieval and summarization. In this and the next two chapters, we investigate theoretically and empirically when effective learning from this weak supervision is possible.

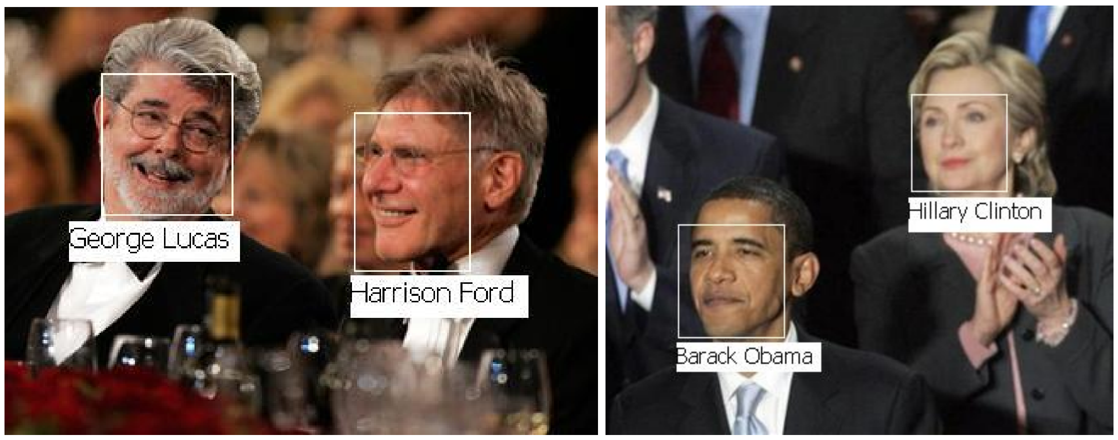


Figure 4.1: Weak supervision from photograph collections: examples of web images with multiple people present. Left caption: "George Lucas and Harrison Ford". Right caption: "Hillary Clinton, Secretary of State of Barack Obama ?"

## Partially Supervised Learning

To put the ambiguous labels learning problem into perspective, it is useful to lay out several related learning scenarios, see figure 2.2:

- In **semi-supervised** learning, the learner has access to a set of labeled examples as well as a set of unlabeled examples.
- In **multiple-instance** learning, examples are not individually labeled but grouped into sets which either contain at least 1 positive example, or only negative examples.
- In **multi-label** learning, each example is assigned multiple labels (for example attributes), all of which are correct.
- Finally, in our setting of **ambiguous labeling**, each example again is supplied with multiple potential labels, *only one of which is correct.* A formal definition is given in Sec. 4.3.

There have been several papers that addressed the ambiguous label framework. [Hullermeier and Beringer, 2006] proposes several non-parametric, instance-based algorithms for ambiguous learning based on greedy heuristics.

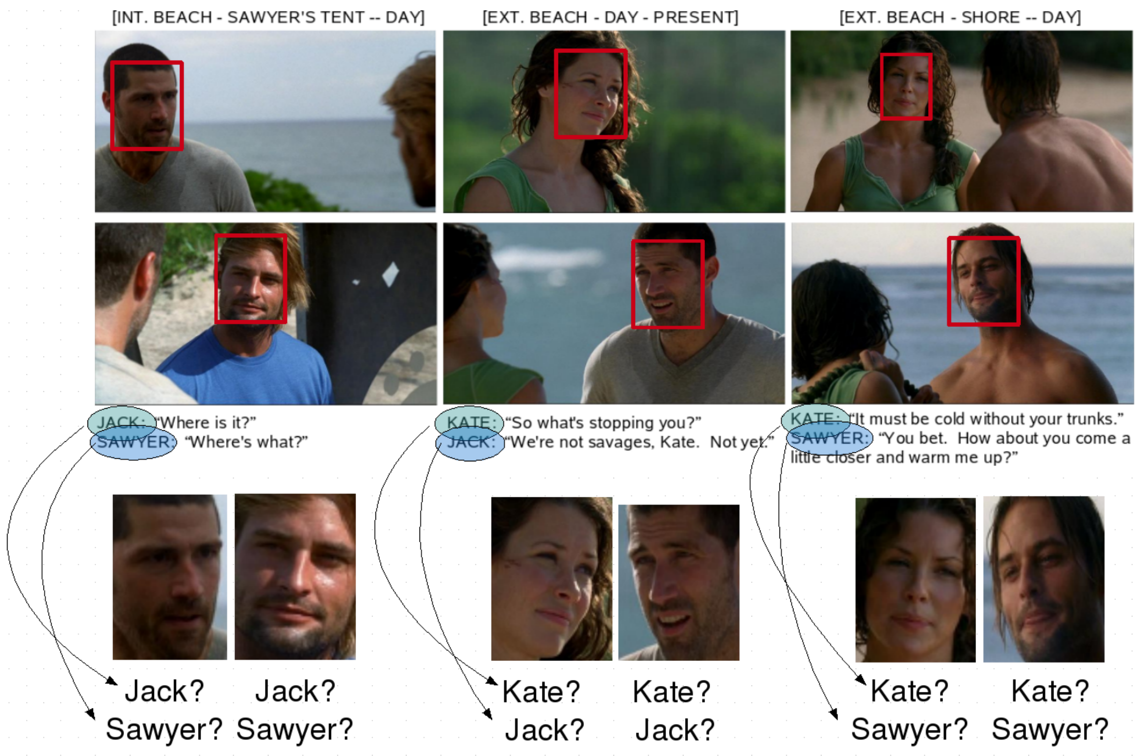

[INT. BEACH - SAWYER'S TENT -- DAY]
[EXT. BEACH - DAY - PRESENT]
[EXT. BEACH - SHORE -- DAY]
JACK: "Where is it?"
SAWYER: "Where's what?"
KATE: "So what's stopping you?"
JACK: "We're not savages, Kate. Not yet."
KATE: "It must be cold without your trunks."
SAWYER: "You bet. How about you come a little closer and warm me up?"
Jack? Sawyer?
Jack? Sawyer?
Kate? Jack?
Kate? Jack?
Kate? Sawyer?
Kate? Sawyer?

Figure 4.2: Examples of frames and corresponding parts of the script from the TV series LOST. From aligning the script to the video, we have 2 ambiguous labels for each person in the 3 different scenes.

[Jin and Ghahramani, 2002] uses expectation-maximization (EM) algorithm with a discriminative log-linear model to disambiguate correct labels from incorrect. Additionally, these papers only report results on synthetically-created ambiguous labels and rely on iterative non-convex optimization.

In this work, we provide intuitive assumptions under which we can expect learning to succeed in identifying the correct label for each instance. We identify conditions under which ambiguously labeled data is sufficient to compute a useful upper bound on the true labeling error. We propose a simple, convex formulation based on this analysis and show how to extend general multi-class loss functions to handle ambiguity. We show that our method significantly outperforms several strong baselines on a large dataset of pictures from newswire and a large video collection.

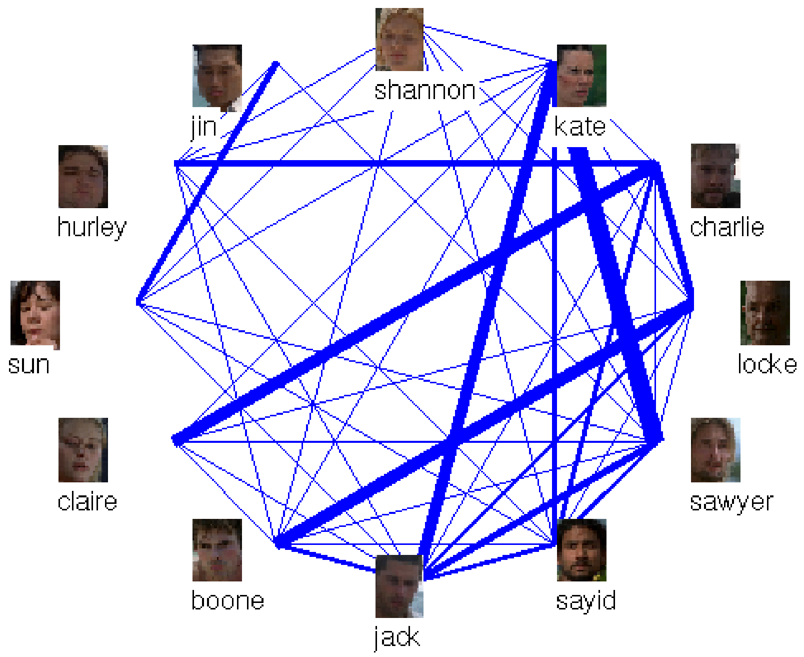


Figure 4.3: Co-occurrence graph of the top characters across 16 episodes of LOST. Larger edges correspond to a pair of characters appearing together more frequently.

## 4.2 Related Work

A more general multi-class setting is common for images with captions (for example, a photograph of a beach with a palm and a boat, where object locations are not specified). [Duygulu et al., 2002, Barnard et al., 2003] show that such partial supervision can be sufficient to learn to identify the object locations. The key observation is that while text and images are separately ambiguous, jointly they complement each other. The text, for instance, does not mention obvious appearance properties, but the frequent co-occurrence of a word with a visual element could be an indication of association between the word and a region in the image. Of course, words in the text without correspondences in the image and parts of the image not described in the text are virtually inevitable. The problem of naming image regions can be posed as translation from one language to another. Barnard et al. [Barnard et al., 2003] address it using a multi-modal extension to mixture of latent Dirichlet allocation.

**Names and Faces**

The specific problem of naming faces in images and videos using text sources has been addressed in several works [Satoh et al., 1999, Berg et al., 2004, Gallagher and Chen, 2007, Everingham et al., 2006]. There is vast literature on fully supervised face recognition, which is out of the scope of this thesis. Approaches relevant to ours include [Berg et al., 2004] which aims at clustering face images obtained by detecting faces from images with captions. Since the name of the depicted people typically appears in the caption, the resulting set of images is ambiguously labeled, if more than one name appears in the caption. Moreover, in some cases the correct name may not be included in the set of potential labels for a face. The problem can be solved by using unambiguous images to estimate discriminant coordinates for the entire dataset. The images are clustered in this space and the process is iterated. Gallagher and Chen [Gallagher and Chen, 2007] address the similar problem of retrieval from consumer photo collections, in which several people appear in each image which is labeled with their names. Instead of estimating a prior probability for each individual, the algorithm estimates a prior for groups using the ambiguous labels. Unlike [Berg et al., 2004], the method of [Gallagher and Chen, 2007] does not handle erroneous names in the captions.

In work on video, a wide range of cues was used to help supervise the data, including: using captions or transcripts [Everingham et al., 2006], using sound [Satoh et al., 1999] to obtain the transcript, using clustering based on clothing within scenes to group instances [Ramanan et al., 2007]. Most of the methods involve either procedural, iterative reassignment schemes or non-convex optimization.

## 4.3 Formulation

In the standard supervised multiclass setting, we have labeled examples $S = \{(x_i, y_i)_{i=1}^m\}$ from an unknown distribution $P(x, y)$ where $x \in \mathcal{X}$ is the input and $y \in \{1, \ldots, L\}$ is the class label. In the partially supervised setting we investigate, instead of an unambiguous single label per instance we have a set of labels, one of which is the correct label for the instance. We will denote the sample as $S = \{(x_i, y_i, Z_i)_{i=1}^m\}$ from an unknown distribution $P(x, y, Z) = P(x, y)P(Z \mid x, y)$ where $Z_i \subseteq \{1, \dots, L\} \setminus y_i$ is a set of additional labels. We will denote $Y_i = y_i \cup Z_i$ as the ambiguity set actually observed by the learning algorithm. Clearly, our setup generalizes the standard semi-supervised setting where some examples are labeled and some are unlabeled: if the ambiguity set $Y_i$ includes all the labels, the example is unlabeled and if the ambiguity set contains one label, we have a labeled example. We consider the middle-ground, where all examples are partially labeled as described in our motivating examples and analyze assumptions under which learning can be guaranteed to succeed.

Consider a very simple ambiguity pattern that makes learning impossible: $L = 3$, $|Z_i| = 1$ and label $1$ is present in every set $Y_i$. Then we cannot distinguish between the case where $1$ is the true label of every example or the case where it is not a label of any example. More generally, if two labels always co-occur when present in $Y$, we cannot tell them apart. In order to learn from ambiguous data, we need to make some assumptions about the joint distribution of $P(Z \mid x, y)$. Below we will make an assumption that ensures some diversity in the ambiguity set. Looking at Figure 4.3, we can see that the distribution of ambiguous pairs is diverse enough to prevent the aforementioned scenario.

## 4.3.1 The model and loss functions

We assume a mapping $\mathbf{f}(x) : \mathcal{X} \mapsto \Re^d$ from inputs to $d$ real-valued features and a multi-linear classifier $g(x) : \mathcal{X} \mapsto \Re^L$ with $L$ components,

$$g^a(x) = \mathbf{w}^a \cdot \mathbf{f}(x),$$

one for each label $a \in \{1, \dots, L\}$, to which we will refer to as class scores. The prediction of the classifier is determined by:

$$g^*(x) = \arg\max_a g^a(x),$$

the highest scoring label according to $g^a$ (we assume that ties are broken arbitrarily, for example, by selecting the label with smallest index $a$). Hence the classifier is parametrized by $d \times L$ weights $w_i^a$, one for each feature-and-class pair.

Many formulations of fully-supervised multiclass learning have been proposed based on minimization of convex upper bounds on risk, usually, the $0/1$ loss [Zhang, 2004]:

$$\mathcal{L}_{01}(g(x), y) = \mathbb{1}(g^*(x) \neq y).$$

In addition to the standard $0/1$ loss, we define the ambiguous $0/1$ loss:

$$\mathcal{L}_{01}(g(x), Y) = \mathbb{1}(g^*(x) \notin Y).$$

### 4.3.2 Connection between ambiguous loss and standard $0/1$ loss

An obvious observation is that the ambiguous loss is an underestimate of the true loss. However in the ambiguous learning setting we would like to minimize the $0/1$, with access only to the ambiguous loss. Therefore we need a way to bound the $0/1$ loss with the ambiguous loss. The following definition defines a measure of the hardness of learning under ambiguous supervision.

**Definition : ambiguity degree $\epsilon(P)$ of a distribution** We define the ambiguity degree $\epsilon(P)$ of a distribution $P(x, y, Z)$ as:

$$\epsilon(P) = \sup_{x \in \mathcal{X}; y, a \in \{1,\dots,L\}} P(a \in Z \mid x, y). \tag{4.1}$$

In words, $\epsilon(P)$ corresponds to the maximum probability of an extra label co-occurring with a true label $y$, over all labels and examples. Let us consider several extreme cases: when $\epsilon(P) = 0$, $Z = \emptyset$ with probability one, and we are back to standard supervised learning case, with no ambiguity. When $\epsilon(P) = 1$, some extra label consistently co-occurs with a true label $y$ on an example $x$ and we cannot tell them apart: no learning is possible for this example. For a fixed ambiguity set size $|Z|$, the smallest possible ambiguity degree is achieved for the uniform case: $\epsilon(P) = |Z|/(L - 1)$. Intuitively, the best case scenario for ambiguous learning corresponds to a distribution with high conditional entropy for $P(Z|x,y)$.

The following proposition shows we can bound the (unobserved) $0/1$ loss by the (observed) ambiguous loss, allowing us to approximately minimize the standard loss with only access to the ambiguous one. The tightness of the approximation directly relates to the ambiguity degree.

**Proposition 4.3.1** *For any classifier $g$ and distribution $P$ with $\epsilon(P) < 1$,*

$$\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] \leq \mathbf{E}_P[\mathcal{L}_{01}(g(x), y)] \leq \frac{1}{1-\epsilon(P)}\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] \tag{4.2}$$

Note, the second bound is tight, as can be shown by considering the uniform case with a fixed ambiguity size $Z$ and $P(a \in Z \mid x, y) = |Z|/(L-1)$.

**Proof** The first inequality comes from the fact that $g^*(x) \notin Y \implies g^*(x) \neq y$. For the second inequality, fix an $x \in \mathcal{X}$ and define $\mathbf{E}_P[\cdot \mid x]$ as the expectation with respect to $P(Y \mid x) = P(y, Z \mid x)$.

$$\begin{aligned}
\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)|x] &= P(g^*(x) \notin Y \mid x) \\
&= P(y \neq g^*(x), g^*(x) \notin Z \mid x) \\
&= \sum_{a \neq g^*(x)} P(y = a \mid x) \underbrace{(1 - P(g^*(x) \in Z \mid x, y = a))}_{\geq 1-\epsilon(P)} \\
&\geq \sum_{a \neq g^*(x)} P(y = a \mid x)(1 - \epsilon(P)) \\
&= (1 - \epsilon(P))\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)|x]
\end{aligned}$$

Hence, $\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)|x] \leq \frac{1}{1-\epsilon(P)}\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)|x]$ for any $x$. We conclude by taking expectation over $x$. □

### 4.3.3 Robustness to outliers

One potential issue with proposition 4.3.1 is that unlikely pairs $x, y$ might force $\epsilon$ to be large, making the bound very loose. We show we can refine the notion of ambiguity degree $\epsilon(P)$ by excluding such pairs.

**Definition** $(\epsilon, \delta)$**-ambiguous distribution.** Define a distribution P to be $(\epsilon, \delta)$-ambiguous if there is a subset of the space $A \subseteq \mathcal{X} \times \{1, \ldots, L\}$ with probability mass at least $1 - \delta$, (i.e. $P((x, y) \in A) \geq 1 - \delta$), where

$$\sup_{(x,y) \in A, a \in \{1,\ldots,L\}} P(a \in Z \mid x, y) \leq \epsilon$$

Note, in the extreme case $\epsilon = 0$, this corresponds to standard semi-supervised learning, where $\delta$-proportion of examples are unambiguously labeled, and $1 - \delta$ are (potentially) fully unlabeled.

This definition allows us to bound the $0/1$ loss even in the case when some unlikely pair $x, y$ with probability $\leq \delta$ would make the ambiguity degree arbitrarily large. Suppose we mix an initial distribution with small ambiguity degree, with an outlier distribution with large overall ambiguity degree. The following proposition shows that the bound degrades only by an additive amount, which can be interpreted as a form of robustness to outliers.

**Proposition 4.3.2** *For any classifier $g$ and $(\epsilon, \delta)$-ambiguous $P(Z \mid x, y)$,*

$$\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)] \leq \frac{1}{1-\epsilon} \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] + \delta.$$

**Proof** We split up the expectation in two parts:

$$\begin{aligned}
\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)] &= \mathbf{E}_P[\mathcal{L}_{01}(g(x), y) | (x, y) \in A](1 - \delta) + \mathbf{E}_P[\mathcal{L}_{01}(g(x), y) | (x, y) \notin A]\delta \\
&\leq \mathbf{E}_P[\mathcal{L}_{01}(g(x), y) | (x, y) \in A](1 - \delta) + \delta \\
&\leq \frac{1}{1-\epsilon} \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) | (x, y) \in A](1 - \delta) + \delta
\end{aligned}$$

We applied proposition 4.3.1 in the last step. Using a symmetric argument,

$$\begin{aligned}
\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] &= \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) | (x, y) \in A](1 - \delta) + \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) | (x, y) \notin A]\delta \\
&\geq \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) | (x, y) \in A](1 - \delta)
\end{aligned}$$

Finally we obtain $\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)] \leq \frac{1}{1-\epsilon} \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] + \delta$ $\square$

### 4.3.4 Label-specific recall bounds

In real settings such as in our movie experiments we observe that certain subsets of labels are harder to disambiguate than others. We can further tighten our bounds between ambiguous loss and standard $0/1$ loss if we consider label specific information. We define the *label-specific ambiguity degree* $\epsilon^a(P)$ of a distribution (with $a \in \{1, \dots, L\}$) as:

$$\epsilon^a(P) = \sup_{x \in \mathcal{X}; a' \in \{1,\dots,L\}} P(a' \in Z \mid x, y = a)$$

We can show the label-specific analog of proposition 4.3.1:

**Proposition 4.3.3** *For any classifier $g$ and distribution $P$ with $\epsilon^a(P) < 1$,*

$$\mathbf{E}_P[\mathcal{L}_{01}(g(x), y) \mid y = a] \leq \frac{1}{1 - \epsilon^a} \mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid y = a].$$

where we see that $\epsilon^a$ bounds per-class recall.

**Proof** Fix an $x \in \mathcal{X}$ (such that $P(y = a|x) > 0$) and define $\mathbf{E}_P[\cdot \mid x, y = a]$ as the expectation with respect to $P(Z \mid x, y = a)$. We consider two cases:

a) if $g^*(x) = a$,

$$\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid x, y = a] = P(g^*(x) \neq y, g^*(x) \notin Z \mid x, y = a) = 0$$

b) if $g^*(x) \neq a$,

$$\begin{aligned}\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid x, y = a] &= P(g^*(x) \notin Z \mid x, y = a) \\ &= 1 - P(g^*(x) \in Z \mid x, y = a) \geq 1 - \epsilon^a\end{aligned}$$

We conclude by taking expectation over $x$:

$$\begin{aligned}\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid y = a] &= P(g^*(x) = a|y = a)\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid g^*(x) = a, y = a] \\ &\quad + P(g^*(x) \neq a|y = a)\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y) \mid g^*(x) \neq a, y = a] \\ &\geq 0 + P(g^*(x) \neq a \mid y = a) \cdot (1 - \epsilon^a) \\ &= (1 - \epsilon^a) \cdot \mathbf{E}_P[\mathcal{L}_{01}(g(x), y) \mid y = a] \quad \square\end{aligned}$$

These bounds give a strong give a strong connection between ambiguous loss and real loss, which allows us to approximately minimize the expected real loss by minimizing (an upper bound on) the ambiguous loss.

## 4.4 A convex learning formulation

We build our formulation on a simple and general multiclass scheme that combines convex binary losses $\psi(\cdot) : \Re \mapsto \Re_+$ on individual components of $g$ to create a multiclass loss. For example, we can use hinge $\psi(u) = \max(0, 1 - u)$, exponential $\psi(u) = \exp(-u)$ or logistic loss $\psi(u) = \log(1 + \exp(-u))$, see figure 2.3. In particular, we assume a type of one-against-all scheme for the supervised case:

$$\mathcal{L}_\psi(g(x), y) = \psi(g^y(x)) + \sum_{a \neq y} \psi(-g^a(x)). \tag{4.3}$$

A classifier $g$ is selected by minimizing the empirical loss on the sample in some function class for $g$ (or by adding a regularization term) to penalize complex models. This form of the multiclass loss is infinite-sample consistent. Informally, this means that empirical loss minimizer achieves Bayes risk if as the number of samples $m$ grows to infinity, the function class for $g$ grows appropriately fast to be able to approximate any function from $\mathcal{X}$ to $\Re^L$ and $\psi(u)$ satisfies the following conditions: $(1)$ $\psi(u)$ is convex, $(2)$ bounded below, $(3)$ differentiable and $(4)$ $\psi(u) < \psi(-u)$ when $u > 0$ (Theorem 9 in [Zhang, 2004]). These conditions are satisfied, for example, for the exponential, logistic and squared hinge loss $\max(0, 1 - u)^2$.

### 4.4.1 Our convex loss function for ambiguously labeled data

In the partial supervision setting, instead of an unambiguous single label $y$ per instance we have a set of labels $Y$, one of which is the correct label for the instance. We propose the following loss function:

$$\boxed{\mathcal{L}_\psi(g(x), Y) = \psi\left(\frac{1}{|Y|}\sum_{a\in Y} g^a(x)\right) + \sum_{a\notin Y}\psi(-g^a(x)) \qquad (4.4)}$$

Note that if the set $Y$ contains a single label $y$, then the loss function reduces to the regular multiclass loss. When $Y$ is not a singleton, then the loss function will drive up the *average* of the scores of the labels in $Y$. If the score of the correct label is large enough, the other labels in the set do not need to be positive. This tendency alone does not guarantee that the correct label has the *highest* score. However, we show in (4.8) that $\mathcal{L}_\psi(g(x), Y)$ upperbounds $\mathcal{L}_{01}(g(x), Y)$ whenever $\psi(\cdot)$ is an upper bound on the $0/1$ loss.

Of course, minimizing an upperbound on the loss does not always lead to sensible algorithms. We show next that our convex relaxation offers a tighter upperbound to the ambiguous loss compared to a more straightforward multi-label approach.

### 4.4.2 Comparison vs naive, multi-label approach

The "naive" model treats each example as taking on multiple correct labels, which implies the following loss function

$$\mathcal{L}_\psi^{naive}(g(x), Y) = \sum_{a\in Y}\psi\left(g^a(x)\right) + \sum_{a\notin Y}\psi(-g^a(x)) \qquad (4.5)$$

One reason we expect our loss function to outperform the naive approach is that we obtain a tighter convex upper bound on $\mathcal{L}_{01}$. Recall our loss function, in comparison:

$$\mathcal{L}_\psi(g(x), Y) = \psi\left(\frac{1}{|Y|}\sum_{a\in Y} g^a(x)\right) + \sum_{a\notin Y}\psi(-g^a(x)) \qquad (4.6)$$

Let us also define

$$\mathcal{L}_\psi^{max}(g(x), Y) = \psi\left(\max_{a\in Y} g^a(x)\right) + \sum_{a\notin Y}\psi(-g^a(x)) \qquad (4.7)$$

which is not convex. Under the usual conditions that $\psi$ is a convex, decreasing upper bound of the step function (e.g., square hinge loss, exponential loss, and log loss with proper scaling), the following inequalities hold:

**Proposition 4.4.1 (comparison between ambiguous losses)**

$$\mathcal{L}_{01} \leq \mathcal{L}_{\psi}^{max} \leq \mathcal{L}_{\psi} \leq \mathcal{L}_{\psi}^{naive} \tag{4.8}$$

**Proof** For the first inequality, if $g^*(x) \in Y$, $\mathcal{L}_{\psi}^{max}(g(x), Y) \geq 0 = \mathcal{L}_{01}(g(x), Y)$. Otherwise we have two cases with $a^* = g^*(x) \notin Y$:

a) if $g^{a^*}(x) \leq 0$, $\max_{a \in Y} g^a(x) \leq 0$ by definition of $g^*(x)$ so $\psi(\max_{a \in Y} g^a(x)) \geq 1$

b) if $g^{a^*}(x) > 0$, $\psi(-g^{a^*}(x)) \geq 1$

In both cases, $\mathcal{L}_{\psi}^{max}(g(x), Y) \geq 1 = \mathcal{L}_{01}(g(x), Y)$.
The second inequality comes from the fact that

$$\max_{a \in Y} g^a(x) \geq \frac{1}{|Y|} \sum_{a \in Y} g^a(x)$$

For the third inequality, using the convexity of $\psi$,

$$\psi\left(\frac{1}{|Y|} \sum_{a \in Y} g^a(x)\right) \leq \frac{1}{|Y|} \sum_{a \in Y} \psi(g^a(x)) \leq \sum_{a \in Y} \psi(g^a(x)) \quad \square$$

This shows that our loss $\mathcal{L}_{\psi}$ is a tighter approximation to $\mathcal{L}_{01}$ than $\mathcal{L}_{\psi}^{naive}$, as illustrated in figures 4.4 and 4.5. What's more, the bound is non-trivial: when $g^a(x) = constant$ over $a \in Y$, we have

$$\psi\left(\max_{a \in Y} g^a(x)\right) = \psi\left(\frac{1}{|Y|} \sum_{a \in Y} g^a(x)\right) = \frac{1}{|Y|} \sum_{a \in Y} \psi\left(g^a(x)\right)$$

To gain additional intuition on why our proposed loss (4.4) is better than the naive loss (4.5): For an input $x$ with ambiguous label set $(a, b)$, our model only encourages the *sum* $g^a(x) + g^b(x)$ to be large, allowing the correct score to be positive and the extraneous score to be negative (e.g., $g^a(x) = 2, g^b(x) = -1$). In contrast, the naive model encourages both $g^a(x)$ and $g^b(x)$ to be large.

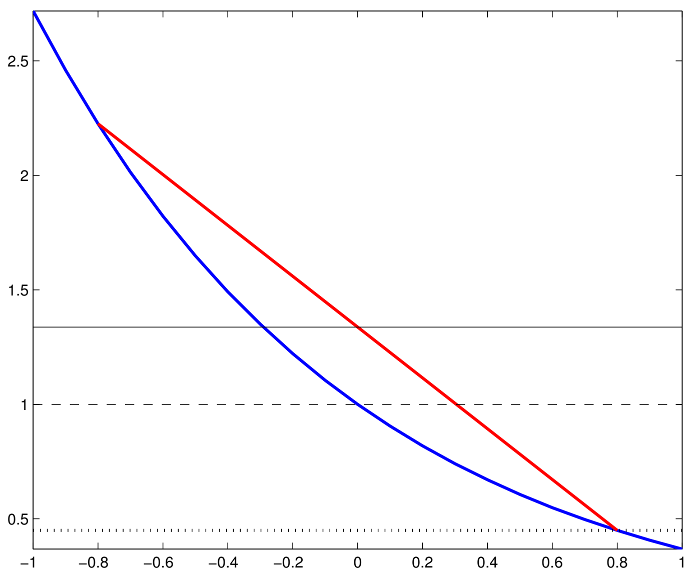


Figure 4.4: For a strictly convex loss such as the exp-loss (blue curve), the ambiguous loss provides a better approximation to the max-loss than the naive loss. The red segment corresponds to the chord with points $g^1, g^2$, the dashed line corresponds to $\psi(\frac{1}{2}(g^1 + g^2))$, the dotted line corresponds to $\psi(\max(g^1, g^2))$, and the black line corresponds to $\frac{1}{2}(\psi(g^1) + \psi(g^2))$.

## 4.4.3 Asymptotic bounds on the convex relaxation

It is possible to identify reasonable conditions on $P(x, y, Z)$ under which our loss will be infinite-sample consistent, for example the case where $P(y \mid x)$ is deterministic.

To derive concrete generalization bounds on multiclass error for our case we define our function class for $g$ as $\mathcal{G} = \{g : \forall a, ||\mathbf{w}^a||_2 \leq 1\}$ and use squared hinge loss $\psi(u) = \max(0, 1 - u)^2$. The corresponding margin-based loss is defined via a truncated, rescaled version of $\psi$:

$$\phi_\gamma(u) = \begin{cases} 1 & \text{if } u \leq 0, \\ (1 - u/\gamma)^2 & \text{if } 0 < u \leq \gamma, \\ 0 & \text{if } u > \gamma. \end{cases} \cdot$$

Using Corollary 15 from [Bartlett and Mendelson, 2002], we can show:

**Proposition 4.4.2** *Assume $||\mathbf{f}(x)||_2 \leq \infty, \ \forall \mathcal{X}$. For any sample S, with probability of at*

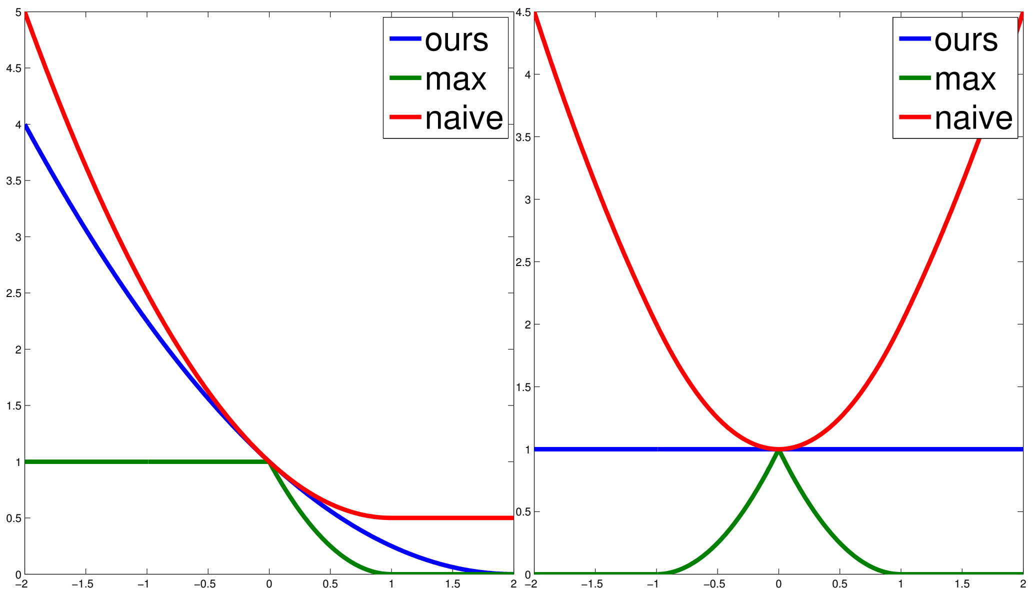


Figure 4.5: Our loss, (4.4) provides a tighter upperbound than the naive loss (4.5) on the non-convex function (4.7). **Left:** plots of $\psi(\frac{1}{2}(g^1 + g^2))$ (ours), $\psi(\max(g^1, g^2))$ (max), $\frac{1}{2}(\psi(g^1) + \psi(g^2))$ (naive), as a function of $g^1 \in [-2, 2]$ (with $g^2 = 0$ fixed). **Right:** same, with $g^2 = -g^1$. In each case we use the square hinge loss for $\psi$, assume $Y = \{1, 2\}$, and drop the negative terms.

*least* $1 - \eta$*, every* $g \in \mathcal{G}$*:*

$$\mathbf{E}_P[\mathcal{L}_{01}(g(x), Y)] \leq \mathbf{E}_S[\mathcal{L}_{\phi_\gamma}(g(x), Y)] + \frac{c_1 L^2}{\gamma m} \cdot \sqrt{\sum_{x \in S} ||\mathbf{f}(x)||_2^2} + c_2 \sqrt{\frac{L^2 \ln(2/\eta)}{m}},$$

*where where* $c_1$ *and* $c_2$ *are constants.*

Using proposition 4.4.2 and proposition 4.3.1, we can derive bounds on the true loss $\mathbf{E}_P[\mathcal{L}_{01}(g(x), y)]$ from purely ambiguous data.

## 4.5 Algorithm and implementation details

Our ambiguous learning formulation is flexible and we can derive many alternative algorithms depending on the choice of the binary loss $\psi(u)$, the regularization, and the optimization method. We describe below a linear programming with hinge loss and $L_1$

regularization, a Support Vector Machine variant with square hinge loss and $L_2$ regularization, and a boosting algorithm with exponential loss. We compare those alternatives with several baselines in our experiments with in the newswire image naming task. First we show how to reduce ambiguous multiclass learning to standard binary classification.

## 4.6 Reduction of ambiguous learning to standard binary classification

In order to minimize (4.4) using off-the-shelf optimization software packages, we first reformulate it as a standard binary classification problem. We can unify the two types of terms in (4.4) as linear combinations of $m \cdot L$ vectors $\bar{\bar{\mathbf{x}}}_{ia} \in \Re^{d \cdot L}$:

$$\bar{\bar{\mathbf{x}}}_{ia} = \delta_a \otimes \mathbf{f}(x) = \left[ \begin{array}{ccccccccc} 0_d & ; & \cdots & ; & \mathbf{f}(x) & ; & \cdots & ; & 0_d \end{array} \right] \tag{4.9}$$

with $\delta_{a,a'} = \mathbb{1}(a = a')$ the kronecker symbol and $\otimes$ the kronecker product: $\bar{\bar{\mathbf{x}}}_{ia}$ has $d \cdot (L-1)$ zero elements (we used matlab notation ";" to denote vertical concatenation). We now replace each input $(x, Y)$ by the following $1+L-|Y|$ fictitious inputs $\{(\bar{\mathbf{x}}_j, \bar{\mathbf{y}}_j)\}_j$:

$$\{(\bar{\mathbf{x}}_j, \bar{\mathbf{y}}_j)\}_j = \{(\frac{1}{|Y|} \sum_{a \in Y} \bar{\bar{\mathbf{x}}}_{ia}, +1)\} \cup \{(\bar{\bar{\mathbf{x}}}_{ia}, -1)\}_{a \notin Y} \tag{4.10}$$

For shortcut, we can write $\bar{\mathbf{x}}_j = \sum u^j_{ia} \bar{\bar{\mathbf{x}}}_{ia}$ the linear decomposition of $\bar{\mathbf{x}}$ ($u^j$ contains either one or $Y$ non-zero elements).

The convex ambiguous loss (4.4) simplifies into a standard binary classification problem:

$$\mathcal{L}_\psi(g(x), Y) = \sum_j \psi(\mathbf{w} \cdot \bar{\mathbf{x}}_j \bar{\mathbf{y}}_j) \tag{4.11}$$

where $\mathbf{w} \in \Re^{d \cdot L}$ is the combined weight vector to learn:

$$\mathbf{w} = \left[ \begin{array}{ccccc} \cdots & ; & \mathbf{w}^a & ; & \cdots \end{array} \right] \tag{4.12}$$

### 4.6.1 Ambiguous loss with Linear or Quadratic Programming

Using $L_p$ regularization ($p = 1$ or $p = 2$), and hinge ($p' = 1$) or square hinge loss ($p' = 2$) as binary loss $\psi(u) = \max(0, 1 - u)^{p'}$ (the square loss is also possible), the optimization problem for ambiguous learning becomes:

$$\min_{\mathbf{w}} ||\mathbf{w}||_p + C \sum_j \max(0, 1 - \mathbf{w} \cdot \bar{\mathbf{x}}_j \bar{\mathbf{y}}_j)^{p'} \tag{4.13}$$

which can be converted into a Linear Program ($p = p' = 1$) or a Quadratic Program. In canonical form, the program has $O(d \cdot L)$ variables, $O(m \cdot L)$ constraints, and $O(mdL)$ non-zero elements (it is sparse). The regularization parameter $C$ can be set by K-fold cross-validation: an interesting point to notice is that even though the original program is *not* fully supervised, the transformed one can be considered as fully supervised, allowing the use of standard supervised learning techniques such as cross-validation. The resulting Linear or Quadratic Program can be solved with general optimization tools such as MOSEK [MOSEK ApS, ]. However, one can achieve better scalability by using more specialized solvers as we describe next.

### 4.6.2 Large scale Ambiguous learning with $L_2$ loss linear Support Vector Machine

The use use of square hinge loss and $L_2$ regularization ($p = p' = 2$) allows us to have a differentiable objective in the primal, which can be rewritten using slack variables $\xi_j$ as:

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||_2^2 + C ||\xi||_2^2 \tag{4.14}$$

$$\text{s.t.} \quad \bar{\mathbf{y}}_j \mathbf{w} \cdot \bar{\mathbf{x}}_j \geq 1 - \xi_j \tag{4.15}$$

We solve the primal using a trust region Newton method described in[Lin et al., 2008], and use the off-the-shelf implementation [Fan et al., 2008] in our experiments. Note that the matrix $(\bar{\mathbf{x}}_j)_j$ is sparse, containing a total of $O(mdL)$ non-zero elements. This allows us to tackle large scale problems with thousands of instances and features, and hundreds of labels.

### 4.6.3 Kernelized Ambiguous loss using Support Vector Machine

We can extend the ambiguous learning framework to the case where the (nonlinear) mapping $\mathbf{f}(x) : \mathcal{X} \mapsto \Re^d$ maps to a high, possibly infinite dimensional space. Note, the mapping is on the original variable $x$, not the fictitious variable $\bar{\mathbf{x}}$ which requires a vector input, and so we cannot use directly the standard SVM formulation on $\bar{\mathbf{x}}$. We suppose we are given a kernel $K : \mathcal{X} \times \mathcal{X} \mapsto \Re$ such that $x_i \cdot x_{i'} = K(x_i, x_i')$. We use here the hinge loss, $L_2$ regularization and an optional additional bias term $b$ for similarity with the SVM framework. The optimization problem for ambiguous learning becomes:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_j \xi_j \tag{4.16}$$

$$\text{s.t.} \quad \bar{\mathbf{y}}_j(\mathbf{w} \cdot \bar{\mathbf{x}}_j - b) \geq 1 - \xi_j, \quad \xi_j \geq 0 \tag{4.17}$$

whose dual is:

$$\max_{\mathbf{w}} \sum_j \alpha_j - \frac{1}{2}\sum_{j,j'} \alpha_j \alpha_{j'} \bar{\mathbf{y}}_j \bar{\mathbf{y}}_{j'} \bar{\mathbf{x}}_j \cdot \bar{\mathbf{x}}_{j'} \tag{4.18}$$

$$\text{s.t.} \quad 0 \leq \alpha_j \leq C, \sum_j \alpha_j \bar{\mathbf{y}}_j = 0 \tag{4.19}$$

Note, without the extra bias term, the dual would be replaced by removing the equality constraint. We can compute the inner product in the dual as:

$$\bar{\mathbf{x}}_j \cdot \bar{\mathbf{x}}'_{j'} = \sum_{i,i',a,a'} u^j_{ia} u^{j'}_{i'a'} \bar{\bar{\mathbf{x}}}_{ia} \cdot \bar{\bar{\mathbf{x}}}_{i'a'} \tag{4.20}$$

$$= \sum_{i,i',a,a'} u^j_{ia} u^{j'}_{i'a'} \delta_{a,a'} f(x_i) \cdot f(x_{i'}) \tag{4.21}$$

$$= \sum_{i,i',a,a'} u^j_{ia} u^{j'}_{i'a'} \delta_{a,a'} K(x_i, x_i') \tag{4.22}$$

where $\bar{\mathbf{x}}_j = \sum u^j_{ia} \bar{\bar{\mathbf{x}}}_{ia}$ is the linear decomposition of $\bar{\mathbf{x}}_j$, and likewise for $\bar{\mathbf{x}}_{j'}$. Note that, given the form of the coefficients $u^j$, the resulting Gram matrix $(\bar{\mathbf{x}}_j \cdot \bar{\mathbf{x}}'_{j'})_{j,j'}$, defining a new kernel $\bar{\mathbf{K}}(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_{j'})$, is sparse, and we can show it has at most $m^2 \cdot L$ elements. When each $Y$ contains only 1 element, the Gram matrix is bloc diagonal with one bloc corresponding to one label, and we can solve the optimization for *each label independently* which amounts to standard one-versus-all multi-class SVM. Otherwise, in the general case we can plug in $\bar{\mathbf{K}}(\bar{\mathbf{x}}_j, \bar{\mathbf{x}}_{j'})$ (or a functional version of it) directly into an off-the-shelf dual based SVM solver.

### 4.6.4 Ambiguous loss with feature selection using boosting

We also present (and experiment with, later) a boosting variant of ambiguous learning, allowing us to incorporate feature selection as part of learning. We consider the case $\psi(x) = \exp(-x)$, although this can be extended to other losses. We take a second order Taylor expansion of the loss $\mathcal{L}_\psi(g(x), Y)$. See Algorithm 2 for details, which is a variant of GentleBoost. The updates of the algorithm are similar to the standard multiclass boosting, but keeps a combined weight for all the ambiguous labels ($z_i$ in Algorithm 2 ).

**Algorithm 2** Ambiguous boosting

```
1: Initialize weights: $z_i = 1 \quad \forall i, \qquad z_{i,a} = 1 \quad \forall i, a \notin Y_i$
2: for $t = 1 \dots T$ do
3:     for $a = 1 \dots L$ do
4:         Fit the parameters of each weak classifier $u$ to minimize the second-order
           Taylor approximation of the cost function with respect to the $a^{th}$ classifier:
           $\frac{1}{2}\sum_i \left[z_i \cdot \mathbb{1}(a \in Y_i)(u(x_i) - 1)^2 + z_{i,a} \cdot \mathbb{1}(a \notin Y_i)(u(x_i) + 1)^2\right] + constant$
5:     end for
6:     Choose the combination of $u, a$ with lowest residual error.
7:     Update $g^a(x) = g^a(x) + u$
8:     for $i = 1 \dots m$ do
9:         if $a \in Y_i$ then
10:            $z_i = z_i \cdot \exp(-u(x_i))$
11:        else
12:            $z_{i,a} = z_{i,a} \cdot \exp(u(x_i))$
13:        end if
14:    end for
15:    Normalize $z$ to sum to 1.
16: end for
```

## 4.7 Transductive analysis

We analyze the ambiguous loss from the point of view a a finite sample. We show we can have guarantees on disambiguating two instances under fairly reasonable assumptions, and generalize this result to obtain a number of bounds on the disambiguation error rate, each making some assumption on the distribution of ambiguous labels.

**Example** Consider a dataset of two points, $x_i, x_j$, with label sets $\{1,2\}, \{1,3\}$, respectively and suppose that the total number of labels is 3. The objective function is given by:

$$\psi(\frac{1}{2}(g^1(x_i) + g^2(x_i))) + \psi(-g^3(x_i)) + \psi(\frac{1}{2}(g^1(x_j) + g^3(x_j))) + \psi(-g^2(x_j))$$

Suppose the correct labels are $1, 1$. It is clear that without further assumptions about $x_i$ and $x_j$ we cannot assume that the optimal solution will predict the right label. However, if $\mathbf{f}(x_i)$ and $\mathbf{f}(x_j)$ are close, it should be intuitively clear that we should be able to deduce that the label of the two examples is $1$.

A natural question is under what conditions on the data will this loss function produce a labeling that is consistent with groundtruth. We provide an analysis under several (fairly restrictive but plausible) assumptions.

**Good Neighbor Assumption** We assume that each instance has a neighbor with the same correct label at most $\epsilon$ away in feature space:

$$\forall x_i \in S \quad \exists x_j \in S : ||\mathbf{f}(x_i) - \mathbf{f}(x_j)|| \leq \epsilon \ , \ y_i^* = y_j^* \ , \ i \neq j$$

where $y_i^*, y_j^*$ are the correct labels for the $x_i, x_j$, and $|| \cdot ||$ is an arbitrary norm.

Note that this does not mean that the *nearest* neighbor of each point has the same label.

**Proposition 4.7.1** *Suppose the objective is bounded by $\psi(\frac{\epsilon}{2})$ and $||\mathbf{w}^a||^* \leq 1$ for every class $a$, where $|| \cdot ||^*$ is the dual norm of $|| \cdot ||$ and $\psi$ is a decreasing upperbound on the step function. Consider $x_i$, $x_j$ that are good neighbors and have different label sets of size 2: $Y_i \neq Y_j$, $|Y_i| = |Y_j| = 2$ . Then $g(x)$ predicts the correct label for $x_i, x_j$.*

**Proof** Without loss of generality, let $Y_i = \{1, 2\}$ and $Y_j = \{1, 3\}$, so the correct label is 1. Note that the number of classes could be larger than 3. Since the objective is bounded by $\psi(\frac{\epsilon}{2})$, and each term is positive, each term is also bounded by $\psi(\frac{\epsilon}{2})$. Consider two of the terms: $\psi(\frac{1}{2}(g^1(x_i) + g^2(x_i))) < \psi(\frac{\epsilon}{2})$ and $\psi(-g^2(x_j)) < \psi(\frac{\epsilon}{2})$, which implies:

$$\frac{1}{2}(\mathbf{w}^1 \cdot \mathbf{f}(x_i) + \mathbf{w}^2 \cdot \mathbf{f}(x_i)) > \frac{\epsilon}{2}$$
$$-\mathbf{w}^2 \cdot \mathbf{f}(x_j) > \frac{\epsilon}{2}$$

Combining the constraints, we obtain $\mathbf{w}^1 \cdot \mathbf{f}(x_i) > \frac{3\epsilon}{2} + \mathbf{w}^2 \cdot (\mathbf{f}(x_j) - \mathbf{f}(x_i))$, which implies

$$\mathbf{w}^1 \cdot \mathbf{f}(x_i) > \frac{3\epsilon}{2} - \epsilon = \frac{\epsilon}{2},$$

since $x_i, x_j$ are $\epsilon$-neighbors and $||\mathbf{w}^2||^* \leq 1$. Similarly,

$$\mathbf{w}^2 \cdot \mathbf{f}(x_i) = \mathbf{w}^2 \cdot \mathbf{f}(x_j) + \mathbf{w}^2 \cdot (\mathbf{f}(x_i) - \mathbf{f}(x_j)) < -\frac{\epsilon}{2} + \epsilon = \frac{\epsilon}{2} \tag{4.23}$$

Hence, $\mathbf{w}^1 \cdot \mathbf{f}(x_i) > \frac{\epsilon}{2} > \mathbf{w}^2 \cdot \mathbf{f}(x_i)$. The proof for $\mathbf{w}^1 \cdot \mathbf{f}(x_j) > \mathbf{w}^3 \cdot \mathbf{f}(x_j)$ is the same $\square$

The assumption about ambiguity set size can be relaxed but would require a stronger assumption about the good neighbors and their sets. We have also assumed that we were lucky and the good neighbors had different ambiguous label sets, which allowed us to deduce the correct label for both. In order to ensure that this happens with high probability, we need to assume that the ambiguity sets for each instance are randomly generated and uncorrelated. Learning under adversarial ambiguity is much harder. In practice, ambiguity sets may be correlated, as we observe in our experiments with naming characters in movies.

We generalize this result and obtain the following bounds on the probability of error (again in the case where $|Y|$=2).

**Proposition 4.7.2 (P(error))** *Suppose* $||w^a||^* \leq 1, \forall a$, *and, for* $\epsilon > 0$, *define:*

$$\alpha_\epsilon = \psi(\epsilon/2) \tag{4.24}$$

$$B_\epsilon(x) = \{x' \neq x : ||x' - x|| \leq \epsilon, y[x] = y[x'], Y[x] \neq Y[x']\} \tag{4.25}$$

$$Z_\epsilon = \{x : \forall x' \in B_\epsilon(x), \mathcal{L}_\psi(x', Y[x']) \geq \alpha_\epsilon\} \tag{4.26}$$

*with $y[x]$ (resp. $Y[x]$) the true label (resp. ambiguous label set) of $x$. We show that:*

$$P(error) \leq \min_{\epsilon} \frac{1}{\alpha_\epsilon} \left( E[\mathcal{L}_\psi] + \frac{E[|B_\epsilon|\mathcal{L}_\psi]}{E_{Z_\epsilon}[|B_\epsilon|]} \right) \tag{4.27}$$

*Introducing $p_\epsilon = P(B_\epsilon(x) = \emptyset)$, we also show:*

$$P(error) \leq \min_{\epsilon} p_\epsilon + E[\mathcal{L}_\psi] \frac{1 + ||B_\epsilon||_\infty}{\alpha_\epsilon} \tag{4.28}$$

*In the dense sampling approximation, when the density $|B_\epsilon(x)| = b_\epsilon > 0$ is constant (over a range $\epsilon \in \mathcal{E}$) we also show:*

$$P(error) \leq \min_{\epsilon \in \mathcal{E}} \frac{2}{\alpha_\epsilon} E[\mathcal{L}_\psi] \tag{4.29}$$

**Proof** If $\mathcal{L}_\psi(x, Y[x]) < \alpha_\epsilon$ and $\exists x' \in B_\epsilon(x) : \mathcal{L}_\psi(x', Y[x']) < \alpha_\epsilon$, then $\mathcal{L}_{01}(x, y[x]) = 0$ (from proposition 4.7.1). Conversely, if $\mathcal{L}_{01}(x, y[x]) = 1$ then $\mathcal{L}_\psi(x, Y[x]) \geq \alpha_\epsilon$ or $\forall x' \in B_\epsilon(x) : \mathcal{L}_\psi(x', Y[x']) \geq \alpha_\epsilon$. It follows that:

$$P(error) \leq P(\mathcal{L}_\psi(x, Y[x]) \geq \alpha_\epsilon) + P(x \in Z_\epsilon)$$

By Markov's inequality, $P(\mathcal{L}_\psi(x, Y[x]) \geq \alpha_\epsilon) \leq E[\mathcal{L}_\psi]/\alpha_\epsilon$ (since $\mathcal{L}_\psi \geq 0$). Let $m$ be the number of instances $x$:

$$\begin{aligned}
m \cdot E[|B_\epsilon|\mathcal{L}_\psi] &= \sum_x |B_\epsilon(x)| \mathcal{L}_\psi(x, Y[x]) \\
&= \sum_x \sum_{x' \in B_\epsilon(x)} \mathcal{L}_\psi(x, Y[x]) \\
&= \sum_x \sum_{x' \in B_\epsilon(x)} \mathcal{L}_\psi(x', Y[x']) \quad \text{(by symmetry)} \\
&\geq \sum_{x \in Z_\epsilon} \sum_{x' \in B_\epsilon(x)} \mathcal{L}_\psi(x', Y[x']) \\
&\geq \sum_{x \in Z_\epsilon} \sum_{x' \in B_\epsilon(x)} \alpha_\epsilon \\
&= \alpha_\epsilon |Z_\epsilon| E_{Z_\epsilon}[|B_\epsilon|]
\end{aligned}$$

Finally $P(x \in Z_\epsilon) \leq E[|B_\epsilon|\mathcal{L}_\psi]/(\alpha_\epsilon E_{Z_\epsilon}[|B_\epsilon|])$.

For the second statement we just write: if $\mathcal{L}_{01}(x, y[x]) = 1$ then $\mathcal{L}_\psi(x, Y[x]) \geq \alpha_\epsilon$ or $B_\epsilon(x) = \emptyset$ or $x \in Z'_\epsilon = \{x \in Z_\epsilon : B_\epsilon(x) \neq \emptyset\}$, yielding:

$$\begin{aligned} P(error) &\leq \min_\epsilon p_\epsilon + \frac{1}{\alpha_\epsilon}\left(E[\mathcal{L}_\psi] + \frac{E[|B_\epsilon|\mathcal{L}_\psi]}{E_{Z'_\epsilon}[|B_\epsilon|]}\right) \\ &\leq \min_\epsilon p_\epsilon + \frac{1}{\alpha_\epsilon}\left(E[\mathcal{L}_\psi] + E[|B_\epsilon|\mathcal{L}_\psi]\right) \\ &\leq \min_\epsilon p_\epsilon + E[\mathcal{L}_\psi]\frac{1 + ||B_\epsilon||_\infty}{\alpha_\epsilon}. \end{aligned}$$

For the third statement, $E[|B_\epsilon|\mathcal{L}_\psi] = b_\epsilon E[\mathcal{L}_\psi]$ and $E_{Z'_\epsilon}[|B_\epsilon|] = b_\epsilon$ □

# Chapter 5

# Experiments for Learning with Ambiguously Labeled Data

We apply our algorithm for ambiguous learning to several datasets, including standard benchmarks from the **UCI repository** [Asuncion and Newman, 2007], a **speaker identification** task from audio extracted from movies, and a **face naming task** from Labeled Faces in the Wild [Huang et al., 2007c]. In chapter 6 we also consider the challenging task of **naming characters in TV shows** throughout an entire season. In each case the goal is to correctly label faces/speech segments/instances from examples that have multiple potential labels, as well as learn a model that can generalize to other unlabeled examples.

We first perform a series of **controlled experiments** to analyze the effect of the the distribution of ambiguous labels on learning. In particular we are interested in the following factors: size of ambiguous bags, proportion of instances which contain an ambiguous bag, entropy of the ambiguity, distribution of true labels and number of true labels. We also consider two learning scenarios: 1) transductive learning, where the task is to disambiguate the true label given an observed bag of labels, and 2) inductive learning (out-of-sample prediction), where we learn a model on an ambiguously labeled dataset, and test the model on a new set of examples which are completely unlabeled.

We compare our approach against a number of **baselines**, including a generative mode, a discriminative maximum-entropy model, a naive model, two K-nearest neighbor models, as well as models that ignore the ambiguous bags. We also propose and compare several **variations** on our cost function.

We conclude with a **comparative summary**, analyzing our approach and the baselines according to several criteria: accuracy, applicability, space/time complexity and running time.

## 5.1 Baselines: description and implementation details

In the experiments, we compare our approach with the following baselines.

### 5.1.1 Random model

We define *chance* as randomly guessing between the possible ambiguous labels only. Defining the (empirical) average ambiguous size to be $E[|Y|] = \frac{1}{m}\sum_{i=1}^{m}|Y_i|$, then the error from the *chance* baseline is given by $\text{error}_{\text{chance}} = 1 - \frac{1}{E[|Y|]}$.

### 5.1.2 IBM Model 1

This generative model was originally proposed in [Brown et al., 1993] for machine translation, but we can adapt it to the ambiguous label case. In our setting, the conditional probability of an example $x \in \Re^d$ belonging to one of its ambiguous labels $a \in Y$ is normally distributed. We use the expectation-maximization (EM) algorithm to learn the parameters of the Gaussians. There are $2d \cdot L$ parameters in total, which are mean $\mu_a \in \Re^d$ and diagonal covariance matrix $\Sigma_a = diag(\sigma_a), \quad \sigma_a \in \Re^d$ for each label (we avoid learning the full covariance matrix, which overfits, is numerically unstable and makes the likelihood unbounded). Compared to EM for standard Gaussian Mixture Models (GMM), each example contributes to the sufficient statistics of only its set of possible labels.

$$P(x|\theta) = \sum_{a \in Y} P(x|a)P(a) \tag{5.1}$$

$$\theta = (\mu_a, \sigma_a)_a, \quad P(x|a) \sim \mathcal{N}(\mu_a, diag(\sigma_a)), \quad P(a) \sim Unif(Y) \tag{5.2}$$

Since the prior $P(a)$ is fixed, the log-likelihood $L(x|\theta) = \sum_i \log P(x_i|\theta)$ is strictly concave and EM will converge to the global optimum. As an implementation detail, we initially experienced numerical instability in the E-step, which involves computing:

$$P(a|x) = \frac{P(x|a)P(a)}{\sum_{a' \in Y} P(x|a')P(a')} \tag{5.3}$$

We fixed this by using the following formula to compute the denominator, with $p_a \overset{\text{def}}{=} \log(P(x|a)) + \log(P(a))$ and $p^* = \max_{a \in Y} p_a$:

$$\log(\sum_{a \in Y} P(x|a)P(a)) = p^* + \log(\sum \exp(p_a - p^*)) \tag{5.4}$$

This reduces the dynamic range in the exponent and avoids numerical overflow. Note, for the same reasons, we compute $\log(P(x|a))$ directly from $\mu_a, \sigma_a$ and avoid forming $P(x|a)$ explicitly.

### 5.1.3 Discriminative EM model

In [Jin and Ghahramani, 2002] the authors train a discriminative model with an EM procedure adapted for the ambiguous label setting. The discriminative model they use is a maximum entropy model [Della Pietra et al., 1997]:

$$P(y|x, \theta) \propto \exp(f(x, y) \cdot \theta) \tag{5.5}$$

The model parameters $\theta$ are found by minimizing the KL divergence between $P(y|x, \theta)$ and some (unknown) distribution over class labels $\hat{P}(y|x)$ that reflects the ambiguous labels ($i.e. P(y|x, \theta) = 0 \forall y \notin Y$):

$$\theta^* = \arg\min \sum_i \sum_{y \in Y} \hat{P}(y|x_i) \log\left(\frac{\hat{P}(y|x_i)}{P(y|x_i, \theta)}\right) \tag{5.6}$$

They propose two approaches:

- in the naive model (we refer to it as **naive-KL**, since we also introduce a different naive model), $\hat{P}(y|x)$ is uniform over ambiguous labels: $\hat{P}(y|x) = \frac{1}{|Y|}$ for $y \in Y$. This gives rise to the following:

$$\theta^* = \arg\min \sum_i \frac{1}{|Y_i|} \sum_{y \in Y} \log\left(\frac{1}{|Y_i| P(y|x_i,\theta)}\right) \tag{5.7}$$

$$= \arg\max \sum_i \frac{1}{|Y_i|} \sum_{y \in Y} \log(P(y|x_i,\theta)) \tag{5.8}$$

$$= \arg\max \sum_i \frac{1}{|Y_i|} \sum_{y \in Y} \frac{\exp(f(x,y)\cdot\theta)}{\sum_{y'} \exp(f(x,y')\cdot\theta)} \tag{5.9}$$

- in the **discriminative EM-model**, the label distribution $\hat{P}(y|x)$ is not fixed, but instead is iteratively estimated using an EM procedure, initialized with $\hat{P}(y|x)$ uniform over the ambiguous labels. In the E-step, minimizing the KL divergence in (5.10) w.r.t. $\hat{P}(y|x)$ gives the following:

$$\hat{P}(y|x) = \begin{cases} \frac{P(y|x,\theta)}{\sum_{y' \in Y} P(y'|x,\theta)} & \forall y \in Y, \\ 0 & \text{else} \end{cases} \tag{5.10}$$

In the M-step, (5.10) is minimized w.r.t. $\theta$, fixing $\hat{P}(y|x)$. As an implementation detail, we experienced slow convergence rate when optimizing for the M-step, which has a form similar to (5.9). We implemented a conjugate gradient ascent to improve convergence.

### 5.1.4 k-Nearest Neighbor

In the supervised case, the weighted k-Nearest Neighbor Classifier[Cover and Hart, 1967] outputs the following function:

$$g_k(x) = \arg\max_y \sum_{i=1}^{k} w_i \mathbb{1}(y = y_i) \tag{5.11}$$

where $x_i$ is the $i^{th}$ nearest-neighbor of $x \in \Re^d$ using Euclidian distance in $\Re^d$, and $w_i$ are a set of weights. Following [Hullermeier and Beringer, 2006], we adapt the k-Nearest

Neighbor Classifier to the ambiguous label setting as follows:

$$g_k(x) = \arg\max_{y \in Y} \sum_{i=1}^{k} w_i \mathbb{1}(y \in Y_i) \tag{5.12}$$

We use two kNN baselines: **kNN** assumes uniform weights $w_i = 1$ (model used in [Hullermeier and Beringer, 2006]), and **weighted kNN** uses linearly decreasing weights $w_i = k - i + 1$, which suppresses dynamic range issues in the distances (compared to alternatives in which the weights depend on the distances). As implementation details, ties are broken randomly, and unless specified otherwise, we use $k = 5$ as was the case in [Hullermeier and Beringer, 2006]. We also report some experiments searching over all possible values of $k$, and number of PCA components when using principal components as features.

### 5.1.5 Naive model

We introduce the following naive model (denoted **naive**) as another baseline, where we train separate binary models for each label (using the same binary loss $\psi(\cdot)$ as in our model) and treat ambiguous label sets as true binary labels for each class (i.e., the model assumes that an exemplar can have two labels during training). The loss function on an example $(x, Y)$ is:

$$\mathcal{L}_{\psi}^{\text{naive}}(g(x), Y) = \sum_{a \in Y} \psi(g^a(x)) + \sum_{a \notin Y} \psi(-g^a(x)), \tag{5.13}$$

After training, we predict the label with the highest score (in the transductive setting):

$$y = \arg\max_{a \in Y} g^a(x) \tag{5.14}$$

As a comparison, our loss function is:

$$\mathcal{L}_{\psi}(g(x), Y) = \psi\left(\frac{1}{Y} \sum_{a \in Y} g^a(x)\right) + \sum_{a \notin Y} \psi(-g^a(x)) \tag{5.15}$$

The parameters shared between our model and the naive model (choice of binary loss $\psi(\cdot)$, regularization term and coefficient of regularization) are set in the same way for the two models in our experiments, so as to have a fair comparison.

### 5.1.6 Supervised models

Finally we also consider two baselines that *ignore* the ambiguous label setting: these model only use the examples that are unambiguously labeled. and discard the other ones. The first one, denoted as **supervised model**, uses the following loss for an example $(x, Y)$:

$$\mathcal{L}_\psi^{\text{supervised}}(g(x), Y) = \begin{cases} \psi(g^y(x)) + \sum_{a \neq y} \psi(-g^a(x)), & Y = \{y\} \\ 0 & |Y| > 1 \end{cases} \tag{5.16}$$

Note, this collapses to the naive model *and* to our ambiguous learning model in the case $|Y| = 1$ (no ambiguity). The second model, denoted as **supervised kNN**, is essentially the same as (5.11) where ambiguously labeled examples have been discarded.

## 5.2 Variants of our approach

Our ambiguous learning model optimizes the following loss during training:

$$\mathcal{L}_\psi(g(x), Y) = \psi\left(\frac{1}{Y}\sum_{a \in Y} g^a(x)\right) + \sum_{a \notin Y} \psi(-g^a(x)) \tag{5.17}$$

This is denoted as the **mean** model in our experiments. In order to get some intuition on our approach, we also investigate the following **sum** and **contrastive** alternatives:

$$\mathcal{L}_\psi^{\text{sum}}(g(x), Y) = \psi\left(\sum_{a \in Y} g^a(x)\right) + \sum_{a \notin Y} \psi(-g^a(x)) \tag{5.18}$$

$$\mathcal{L}_\psi^{\text{contrastive}}(g(x), Y) = \sum_{a' \notin Y} \psi\left(\frac{1}{Y}\sum_{a \in Y} g^a(x) - g^{a'}(x)\right) \tag{5.19}$$

Note, in each case the term $\sum_{a \in Y} g^a(x)$ appears *inside* the binary loss. When $\psi(\cdot)$ is the hinge loss, the mean and sum model are very similar, but this is not the case for strictly convex binary losses. The regularization coefficient $C$ can be set using cross-validation on the bag-level as we have explained. In our experiments, we observed that accuracy was relatively insensitive to even large variations (several orders of magnitude) of $C$, and unless specified otherwise we fixed $C = 10^3$ in all experiments.

| **Dataset** | # instances ($m$) | # features ($d$) | # labels ($L$) | prediction task |
|---|---|---|---|---|
| UCI: dermatology | 366 | 34 | 6 | disease diagnostic |
| UCI: ecoli | 336 | 8 | 8 | site prediction |
| UCI: abalone | 4177 | 8 | 29 | age determination |
| FIW(10b) | 500 | 50 | 10 (balanced) | face recognition |
| FIW(10) | 1456 | 50 | 10 | face recognition |
| FIW(100) | 3011 | 50 | 100 | face recognition |
| LOST-audio | 522 | 50 | 19 | speaker id |
| TV+movies | 10,000 | 50 | 100 | face recognition |

Table 5.1: Summary of datasets used in our experiments. The TV+movies experiments are treated in chapter 6. Faces in the Wild (1) uses a balanced distribution of labels (first 50 images for the top 10 most frequent people).

## 5.3 Datasets and feature description

We describe below the different datasets used to report our experiments. The experiments for automatic naming of characters in TV shows can be found in chapter 6. A concise summary is given in table 5.1.

### 5.3.1 UCI Datasets

We selected three biology related datasets from the publicly available UCI repository [Asuncion and Newman, 2007]: dermatology, ecoli, abalone. As a preprocessing step, each feature was independently scaled to have zero mean and unit variance.

### 5.3.2 Faces in the Wild (FIW)

We experiment with different subsets of the publicly available Labeled Faces in the Wild [Huang et al., 2007c] dataset. We use the images registered with funneling [Huang et al., 2007b] (the registration is not perfect), and crop out the central part corresponding to the approximate face location, which we resize to 60x90, see figure 5.1. We compute the corresponding eigenfaces[Turk and Pentland, 1991] by projecting the 60x90

Figure 5.1: Examples from Faces in the Wild dataset used in our experiments (one face is shown for each of the top 10 most frequent labels).

grayscale images (treated as 5400x1 vectors) onto a 50 dimensional space using Principal Components Analysis. We kept the features simple by design; more sophisticated part based registration and representation and would further improve results as we will see in chapter 6. In table 5.1, FIW(10b) extracts the first 50 images for each of the top 10 most frequent people (balanced label distribution); FIW(10) extracts *all* images for each of the top 10 most frequent people (heavily unbalanced label distribution, with 530 hits for George Bush and 53 hits for John Ashcroft); FIW(100) extracts up to 100 faces for each of the top 100 most frequent people (again, heavily unbalanced label distribution).

### 5.3.3 Speaker Identification from Audio

We also investigate a speaker identification task based on audio in an **uncontrolled** environment. The audio is extracted from an episode of LOST (season 1, episode 5) and is initially **completely unaligned**. Compared to recorded conversation in a controlled environment, this task is more realistic and very challenging due a number of factors: background noise, strong variability in tone of voice due to emotions, people shouting or talking at the same time. After alignment (described next), our dataset is composed of 522 utterances (each one corresponding to a closed caption line), with 19 different speakers. We describe below the details of the alignment, which produces a set speech segments along with the corresponding spoken text and speaker id.

**Alignment between audio and text.** We used mplayer (`http://www.mplayerhq.hu/`) to extract audio from the DVD, representing 45 minutes sampled at 48kHz. The closed captions extracted from the DVD provide an initial guess to segment the audio into speech segments. There are two main difficulties: (1) closed caption time stamps often overlap temporally, resulting in multiple potential speakers for an audio segment, and (2) the time stamp intervals are approximate, often covering a significant portion of non-speech audio signal along with the actual utterance.

We address both those issues using **forced alignment** [Moreno et al., 1998, Sjlander, 2003], which aligns each phoneme in the text to an audio interval. This requires establishing one-to-one correspondence between transcription and speech, which we carry out by merging closed caption lines that overlap in time, and extracting the audio from the union of the corresponding time stamp intervals. We use the Hidden Markov Model Toolkit (HTK) (`http://htk.eng.cam.ac.uk/`) to compute forced alignment. Accuracy of the alignment at the phoneme level was rather poor, but in our case we mostly care about coarse, sentence level alignment so as to segment speech from non-speech and prevent overlap between different speakers. We measured accuracy of the forced alignment by manually groundtruthing audio boundaries for an entire episode of Lost. Results in figure 5.2 show that we are $95\%$ precise if we tolerate a $\pm 2$ seconds incertitude over the interval. Most of the errors come from strong background music or noise.

**Alignment between audio, text and speaker.** To obtain groundtruth, we use our aligned screenplay and closed captions from chapter 3 to attach to each speech segment a speaker id.

**Feature representation.** For each speech segment (typically between 1 and 4 seconds) we extract standard voice processing audio features: pitch[Talkin, 1995], Mel-Frequency Cepstral Coefficients (MFCC)[Mermelstein, 1976], Linear predictive coding (LPC). For more details on the topic we refer the reader to [Proakis and Manolakis, 1996, Schroeder, 2004]. This results in a total of 4,000 features, which we normalize to the range $[-1, 1]$ and then project onto 50 dimensions using PCA.

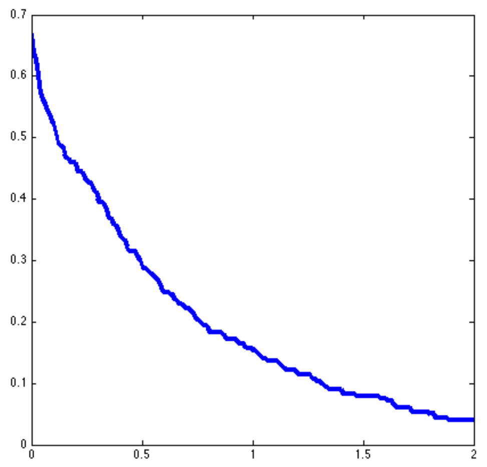


Figure 5.2: Forced alignment error rate at the sentence level. To evaluate the accuracy of our forced alignment pipeline, we manually groundtruthed audio boundaries at the utterance level (one line of closed captions) for an entire episode of Lost (season 1, episode 5). The plot shows the frequency of automatically recovered speech boundaries that fall outside of the confidence interval (x-axis, in seconds) around the groundtruth boundary.

## 5.4 Controlled experiments

In order to assess the performance of our proposed approach for ambiguous learning, we perform a series of controlled experiments in which follow the same protocol. In each experiment, we vary a single parameter (size of ambiguous bags, number of ambiguous bags, number and distribution of true labels, etc.) over a range of **10 values**.

For the **inductive experiments**, we split randomly in half the instances into (1) **ambiguously labeled training set**, and (2) **unlabeled testing set**. The ambiguous labels in the training set are generated randomly according to different noise models which we specify in each case. For each method and parameter setting, we report the **average test error rate** over **20 trials** after training the model on the ambiguous train set. We also report the corresponding **standard deviation** as error bar in the plots. Note, in the inductive setting we consider the test set is unlabeled, and so the classifier votes among *all* possible labels:

$$y = \arg \max_{a \in \{1..L\}} g^a(x) \tag{5.20}$$

For the **transductive experiments**, there is no test set; we report the error rate for disambiguating the ambiguous labels (also averaged over 20 trials corresponding to random settings of ambiguous labels). The main differences with the inductive setting are: (1) the model is trained on all instances and tested on the same instances; and (2) the classifier votes only among the ambiguous labels, which is easier:

$$y = \arg \max_{a \in Y} g^a(x) \tag{5.21}$$

We compare our approach (denoted as **mean**) against the **baselines** presented in section 5.1: Chance, Model 1, Discriminative EM model, k-Nearest Neighbor, weighted k-Nearest Neighbor, Naive model, supervised model, and supervised kNN. Note, in our experiments the Discriminative EM model was much slower to converge than all the other methods, and we only report the first series of experiments with this baseline (in figure 5.3).

Table 5.2 summarizes the different settings used in each experiment. We experiment with 3 different noise models for ambiguous bags, parametrized by $p, q, \epsilon$. $p$ represents the proportion of examples that are ambiguously labeled. $q$ represents the number of *extra* labels for each ambiguous example. $\epsilon$ represents the degree of ambiguity for each ambiguous example (see definition 4.1).

**Experiments with a boosting version of the ambiguous learning.** We also experiment with a boosting version of our algorithm, as presented in section 4.6.4. Results are shown in figure 5.15, comparing our method with kNN and the naive method (also using boosting). Despite the change in learning algorithm and loss function, the trends remain the same.

| **Experiment** | **fig** | **induct.** | **dataset** | **parameter** |
|---|---|---|---|---|
| ambiguity size | 5.3 | yes | FIW(10b) | $p = 1, q \in [0, 0.9(L-1)]$ |
| ambiguity size | 5.4 | yes | Lost audio | $p = 1, q \in [0, 0.9(L-1)]$ |
| ambiguity size | 5.5 | yes | ecoli | $p = 1, q \in [0, 0.9(L-1)]$ |
| ambiguity size | 5.6 | yes | derma | $p = 1, q \in [0, 0.9(L-1)]$ |
| ambiguity size | 5.7 | yes | abalone | $p = 1, q \in [0, 0.9(L-1)]$ |
| 10 labels (unbalanced) | 5.8 | yes | FIW(10) | $p = 1, q \in [0, 0.9(L-1)]$ |
| 100 labels (unbalanced) | 5.9 | yes | FIW(100) | $p = 1, q \in [0, 0.9(L-1)]$ |
| ambiguity size(trans.) | 5.10 | **no** | FIW(10b) | $p = 1, q \in [0, 0.9(L-1)]$ |
| # of ambiguous bags | 5.11 | yes | FIW(10b) | $p \in [0, 0.95], q = 2$ |
| degree of ambiguity | 5.12 | yes | FIW(10b) | $p = 1, q = 2, \epsilon \in [1/L, 1]$ |
| degree of ambiguity | 5.13 | **no** | FIW(10b) | $p = 1, q = 2, \epsilon \in [1/L, 1]$ |
| dimension | 5.14 | yes | FIW(10b) | $p = 1, q = \frac{L-1}{2}, d \in [1, .., 200]$ |

Table 5.2: Summary of controlled experiments. We experiment with 3 different noise models for ambiguous bags, parametrized by $p, q, \epsilon$. $p$ represents the proportion of examples that are ambiguously labeled. $q$ represents the number of *extra* labels for each ambiguous example. $\epsilon$ represents the degree of ambiguity for each ambiguous example (see definition 4.1). $L$ is the total number of labels. We study the effects of: (1) label noise model and amount of noise, (2) dataset choice, (3) number and distribution of true labels, (4) inductive vs transductive learning, (5) feature dimensionality

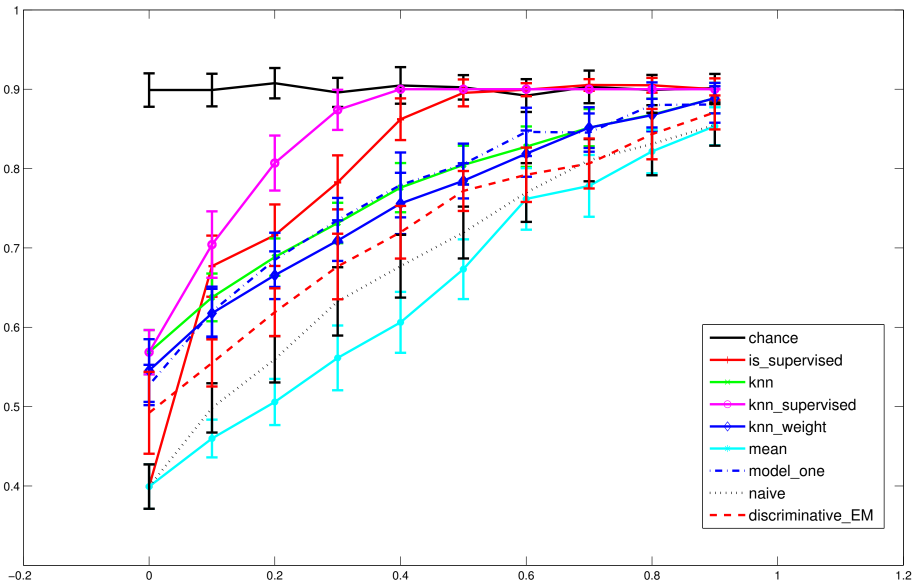


Figure 5.3: Please refer to table 5.2. Comparison on FIW(10b).

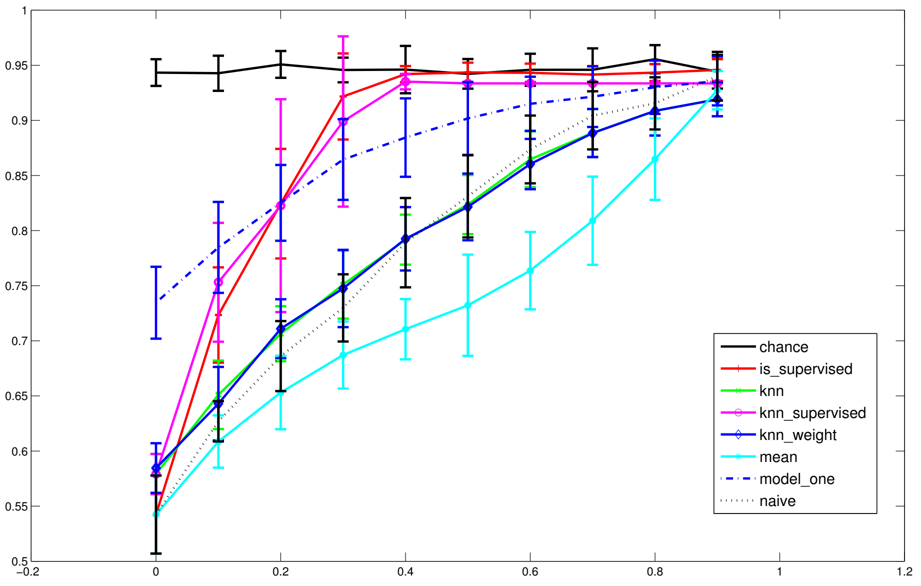


Figure 5.4: Please refer to table 5.2. Comparison on audio from Lost.

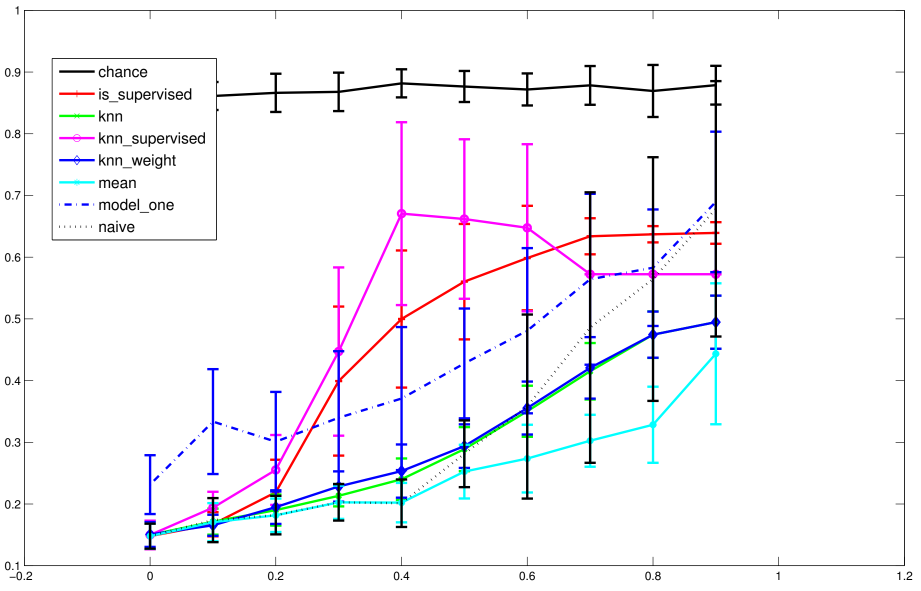


Figure 5.5: Please refer to table 5.2. Comparison on ecoli.

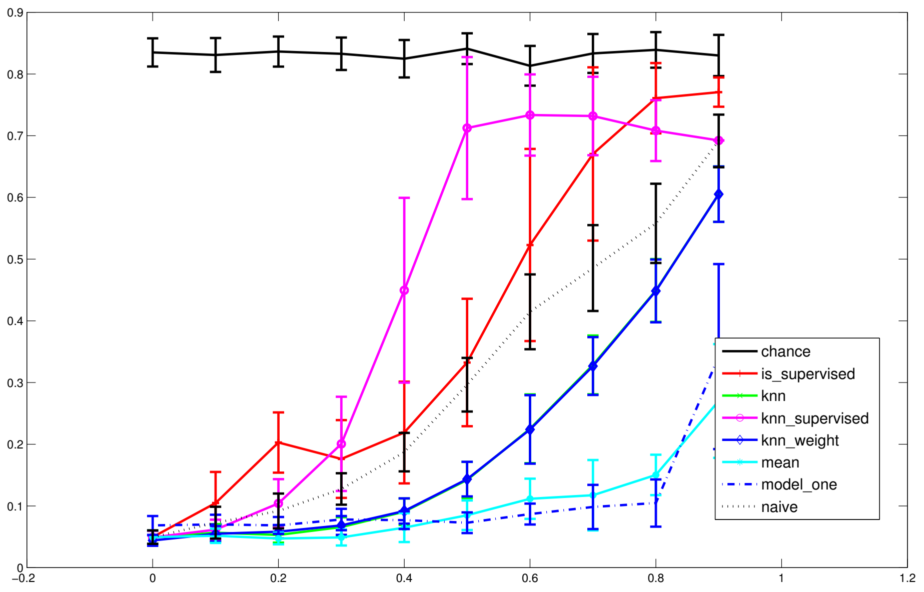


Figure 5.6: Please refer to table 5.2. Comparison on dermatology.

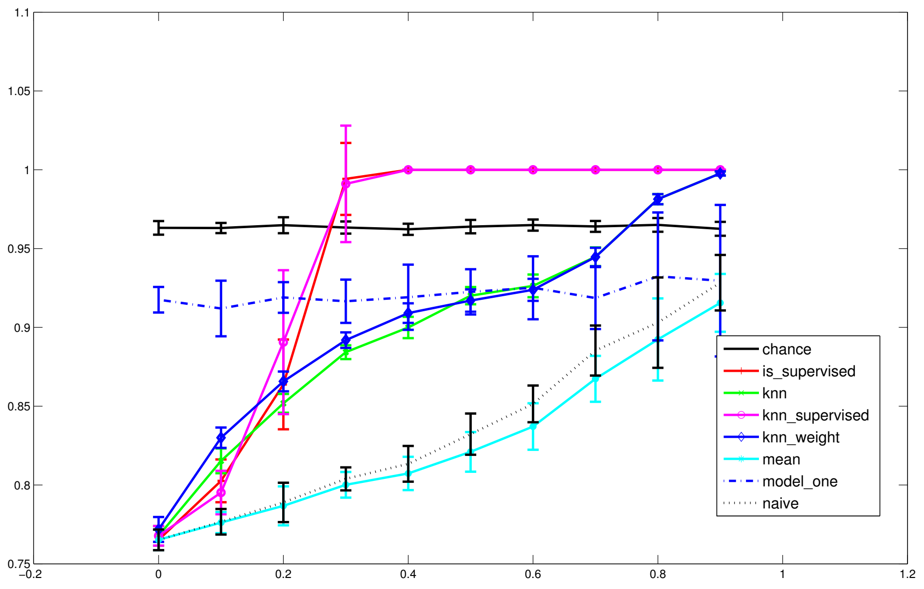


Figure 5.7: Please refer to table 5.2. Comparison on abalone.

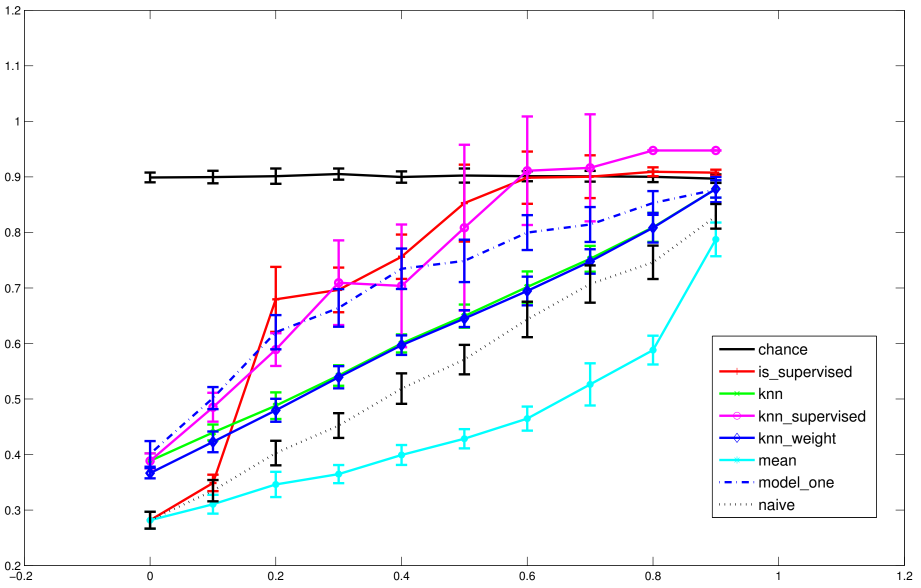


Figure 5.8: Please refer to table 5.2. Using 10 labels (unbalanced).

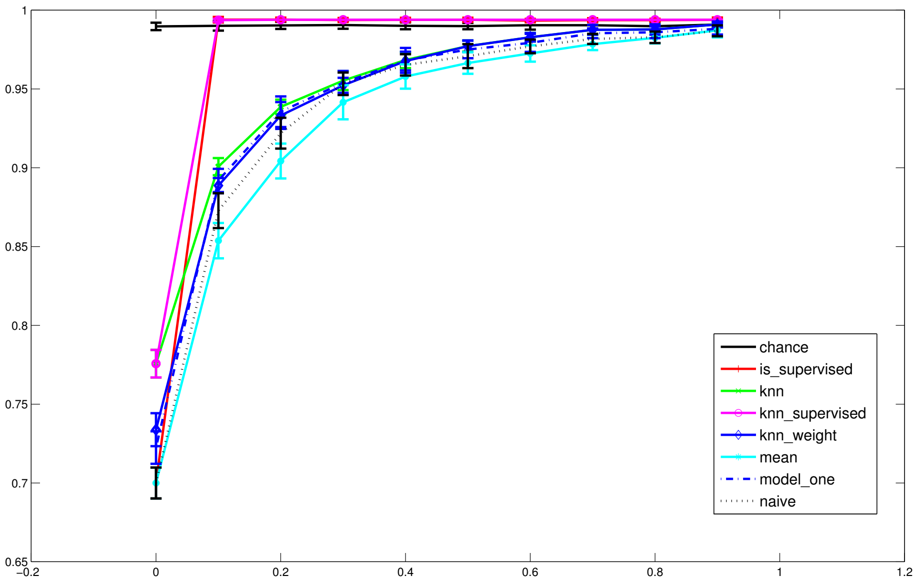


Figure 5.9: Please refer to table 5.2. Using 100 labels (unbalanced).

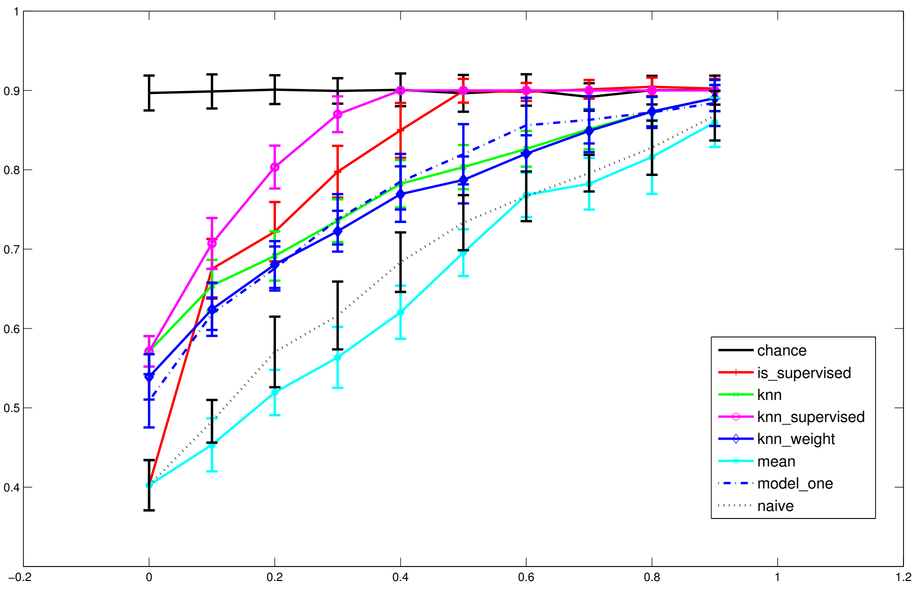


Figure 5.10: Please refer to table 5.2. Using transductive learning.

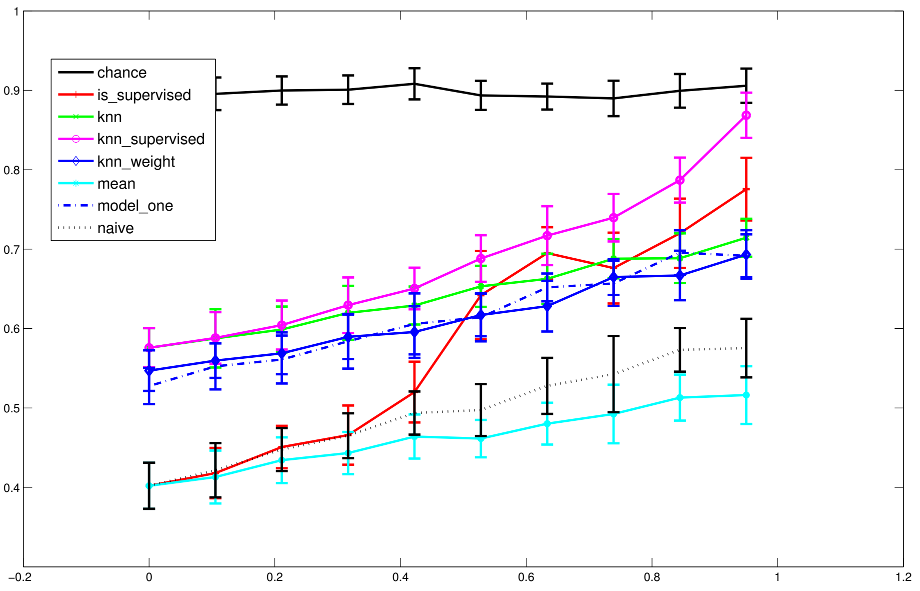


Figure 5.11: Please refer to table 5.2. Effect of # of ambiguous bags.

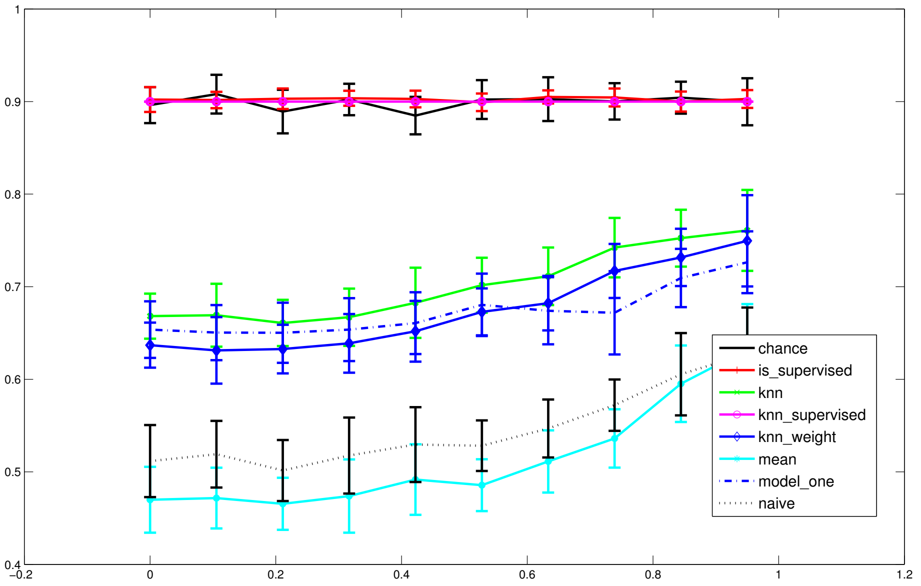


Figure 5.12: Please refer to table 5.2. Effect of ambiguity degree.

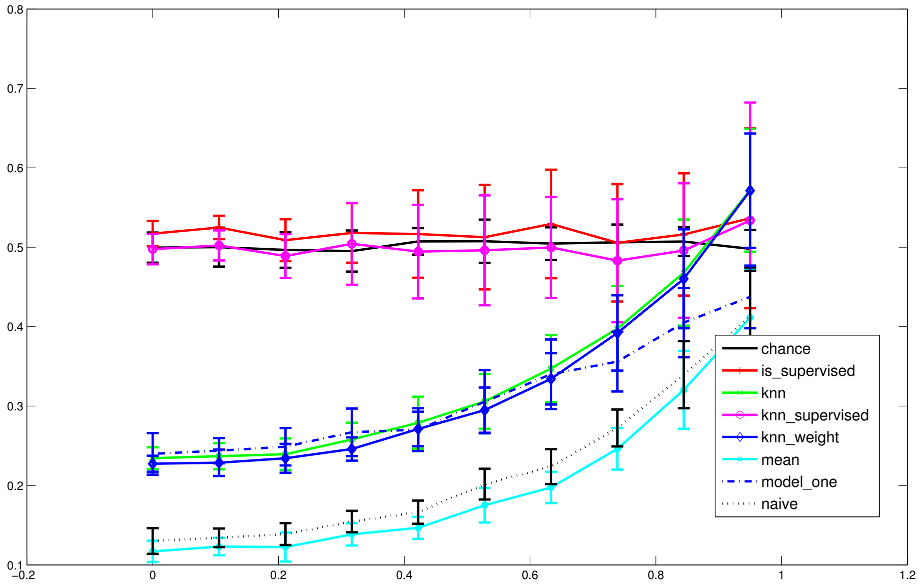


Figure 5.13: Please refer to table 5.2. Effect of ambiguity degree (with transductive learning).

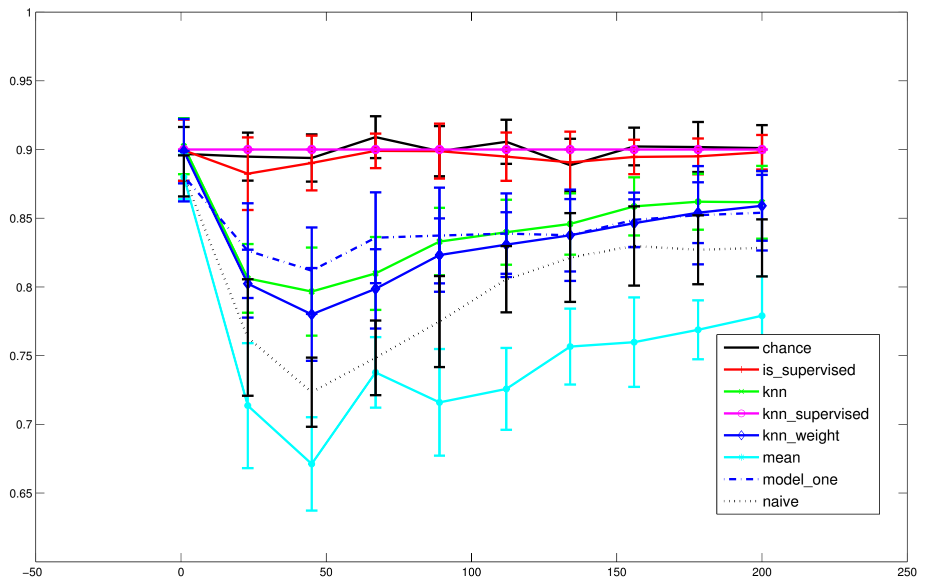


Figure 5.14: Please refer to table 5.2. Effect of dimensionality

## 5.5 Comparative Summary

**How useful are ambiguously labeled examples compared to labeled examples?.** One of the most prominent observations is that the supervised models defined in section 5.1.6 (which ignore the ambiguously labeled examples) consistently perform worse than their counterparts adapted for the ambiguous setting. For example, consider figure 5.11. A model trained with nearly all examples ambiguously labeled ("mean" curve", $p = 95\%$) performs as good as a model which uses $60\%$ of *fully labeled* examples ("supervised" curve, $p = 40\%$). The same holds between the "kNN" curve at $p = 95\%$ and the "supervised kNN" curve at $p = 40\%$.

**Comparison between different approaches.** We make the following observations:

- The naive model uniformly outperformed the *other* baselines in all but 3 experiments (out of 12)

- Our proposed model **uniformly outperformed** all baselines in all but one experiment (figure 5.6, with UCI: dermatology dataset), where it ranked second closely

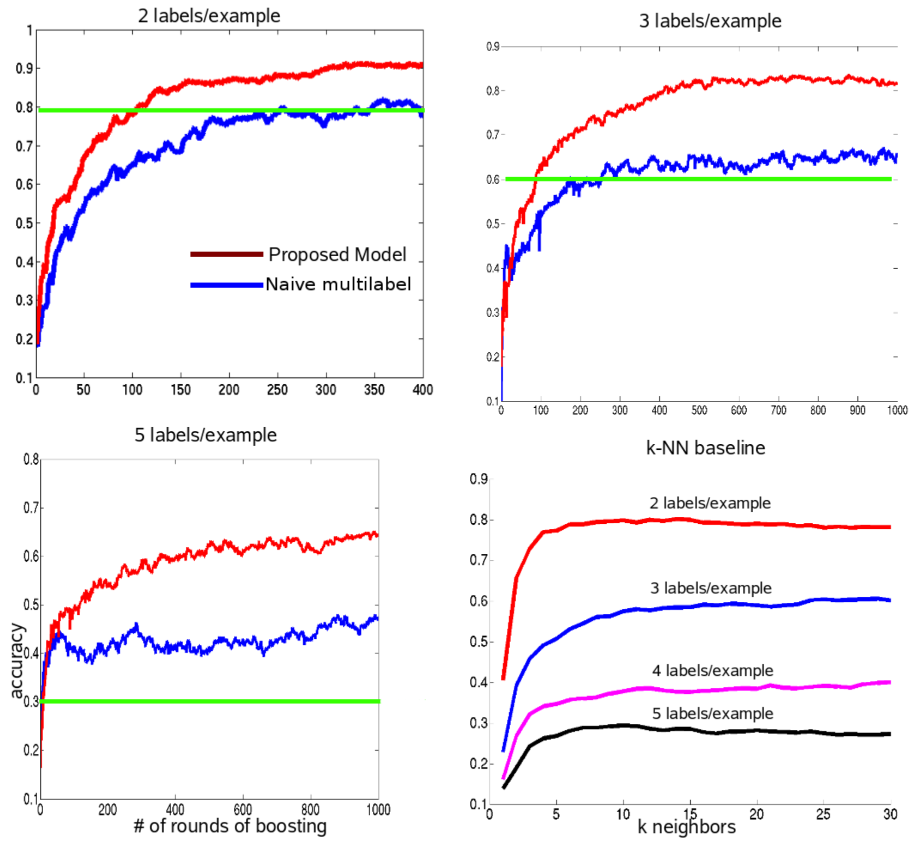


Figure 5.15: We experiment with a boosting version of the ambiguous learning, and compare to a boosting version of the naive baseline, as well as the kNN (searching over multiple values of $k$). We vary the *total* size of ambiguous bags, and plot *accuracy* as the number of boosting rounds increases. The green horizontal lines correspond to the best performance of the nearest neighbor method.

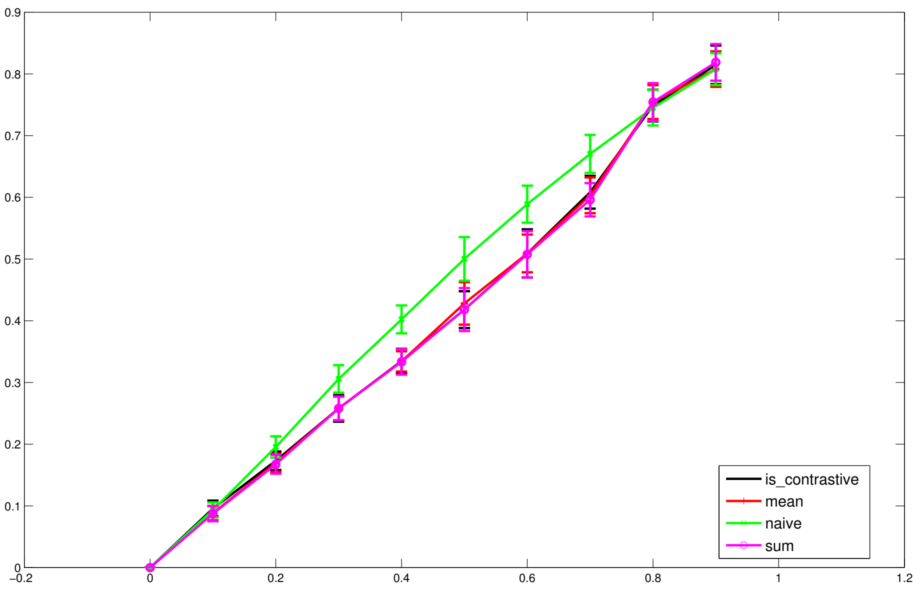


Figure 5.16: Variations on our loss function: we compare our mean model proposed in chapter 4 with variations on the model defined in section 5.2. We also show the baseline naive model. The setting is the same as for figure 5.11.

behind Model 1. In particular it always uniformly outperformed the naive model.

- Weighted kNN has a slight edge over kNN
- other than these, there is no clear additional ranking among methods that is consistent across experiments

**Comparison between variations of our approach.** Figure 5.16 shows that variations on our cost function have *little effect* in the transductive setting. In the inductive setting, other experiments we performed show that the mean and sum version are still very similar, but the contrastive version is worse.

In general it seems that models based on minimization of a convex loss function (naive and different versions of our model) usually outperform the other models.

**Factors that influence the error rate.** As could be expected, $p$ and $q$ monotonously affect the error rate: more ambiguity or larger ambiguous bags consistently increase error rate.

We also see that the ambiguity degree $\epsilon$ significantly affects error rate (even though the total amount of ambiguity remains constant). Lastly, transductive setting is more affected by the amount of ambiguity than the inductive setting (since final voting is among all ambiguous labels in the transductive case).

**Dataset considerations.** Less understood is the effect of dataset parameters (number of dimensions, number and distribution of labels, number of examples, type of features) on the ambiguous learning. The relative ranking and gap in performance between the different methods are affected by the choice of dataset.

# Chapter 6

# Learning with Ambiguously Labeled Faces in Videos

We now return to our introductory motivating example, naming people in TV shows (see figure 4.2). Our goal is to identify characters given ambiguous labels derived from the screenplay. Our data consists of 100 episodes ( 75 hours) of *LOST* and *CSI*, from which we extract ambiguously labeled faces to learn models of common characters. We used the same features, learning algorithm and loss function as in section 5.3.2.

In our video naming task, we explain how we collected our dataset, and how to obtain the ambiguous labels from automatically aligned screenplays. We explore several combinations of existing and novel features and constraints to improve accuracy of naming, and propose several metrics for analyzing performance. Finally we explain how we generate videos containing episode-wide and series-wide labeled face tracks.

## 6.1 Data Collection

We adopt the following filtering pipeline, illustrated in figure 6.2, to extract face tracks, inspired by [Everingham et al., 2006]:

1. Run the off-the-shelf OpenCV face detector over all frames, searching over in-plane

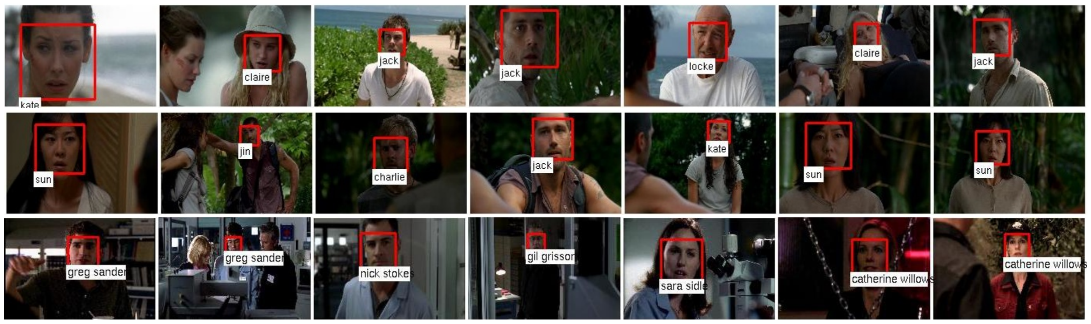


Figure 6.1: Results of our predictions for LOST and CSI. The misclassified examples are: row 1, column 3 (truth: Boone) and row 2, column 2 (truth: Jack).

rotations and scales.

2. Run face part detectors[1] over the face candidates, and perform a 2D rigid transform of the parts in each face to a fixed template. The face image is registered accordingly.

3. Compute the score of a candidate face $s(x)$ as the sum of part detector scores plus rigid fit error, normalizing each to weight them equally, and filtering out faces with low score.

4. Assign faces to tracks by associating face detections within a shot using normalized cross correlation on RGB, and using dynamic programming to group them together into tracks. We subsample the tracks to avoid repetitive examples.

Concretely, for a particular TV episode, step (1) finds approximately 100,000 faces, step (3) keeps approximately 10,000 of those, and after subsampling tracks in step (4) we are left with 1000 face detections.

## 6.2 Ambiguous Label Selection

Screenplays for popular tv series and movies are readily available on the web. Given an alignment of the screenplay to frames, we have ambiguous labels for characters in each

[1]Boosted cascade classifiers of Haar features for the eyes, nose and mouth

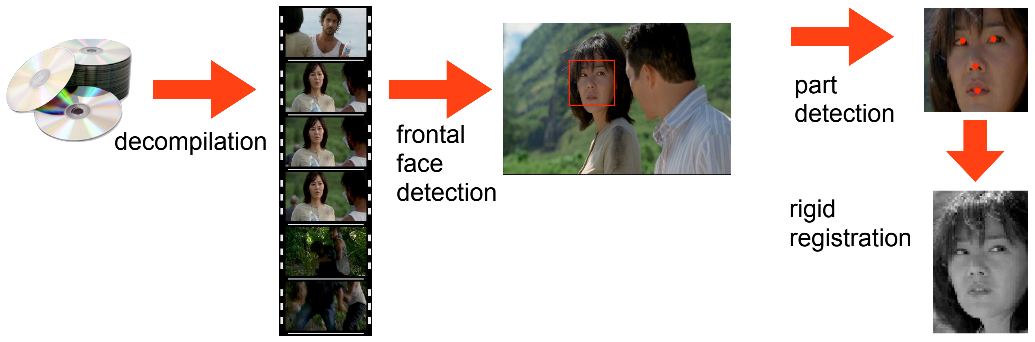


Figure 6.2: Face pipeline

scene: the set of speakers mentioned at some point in the scene, as illustrated in Figure 4.2. We align the screenplay to the video using the methods presented in chapter3.

We use the scripts to select faces filtered by our pipeline and their associated ambiguous label set based on two criteria:

- We prune out scenes that contain other characters than the ones we focus on (top 8, 16, or 32 characters across the episodes we consider).
- We prune out scenes that contain 4 or more characters in the scene (those are rare cases anyway)

Each face track $x$ in a scene is then represented by an ambiguous bag of size $|Y| \in \{1, 2, 3\}$, corresponding to the set of characters mentioned at any point during the scene: set of speakers, and set of characters mentioned in the narrative elements of the screenplay. All names are aligned to a common first name / last name reference using automatically extracted cast list from IMDB, see section 7.8.1). The average bag size we obtain is $2.13$ for LOST, and $2.17$ for CSI.

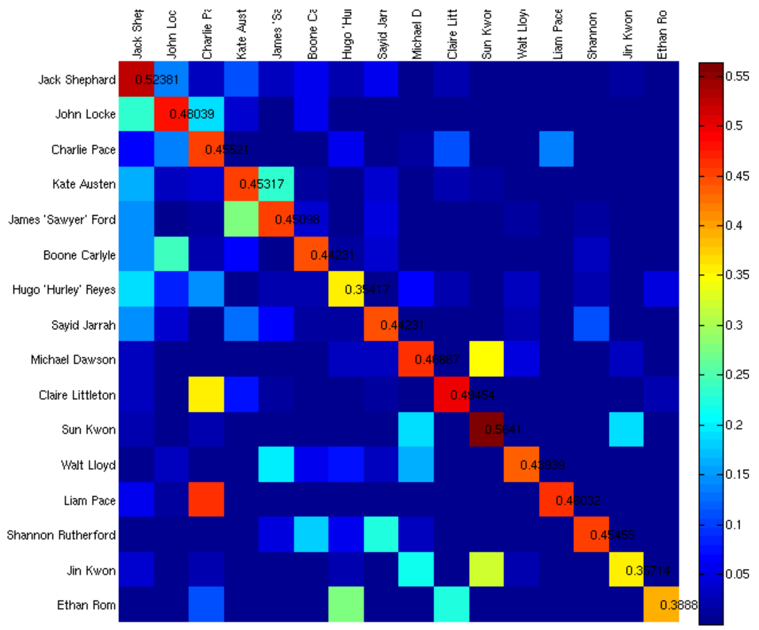
**Errors in ambiguous label selection.** In the TV episodes we considered, we observed that approximately $1\%$ of ambiguous label sets were wrong, in that they didn't contain the ground truth label of the face track. This came from several reasons: presence of a non-english speaking character (Jin Kwon in Lost, who speaks Korean) whose dialogue is not transcribed in the closed captions; sudden occurence of an unknown, uncredited character on screen, and finally alignment problems due to large discrepencies between screenplay and closed captions. While this is not a major problem, it becomes so when we consider additional cues (mouth motion, gender) that restrict the ambiguous label set. We will see how we tackle this issue with a robust confidence measure for obtaining good precision recall curves.

## 6.3 Results with the basic system

Now that we have a set of instances (face tracks), feature descriptors for the face track and ambiguous label set for each face track, we can apply the same method as described in the previous chapter. We use a transductive setting: we test our method on our ambiguously labeled training set.

The confusion matrix displaying the distribution of ambiguous labels for the top 16 characters in LOST is shown in figure 6.3 (top). The confusion matrix of our predictions after applying our ambiguous learning algorithm is shown in figure 6.3 (bottom). Our method had the most trouble disambiguating Ethan Rom from Claire Littleton (Ethan Rom only appears in $0.7\%$ of the ambiguous bags, 3 times less then the second least common character) and Liam Pace from Charlie Pace (they are brothers and co-occur frequently, as can be seen in the top figure). The case of Sun Kwon and Jin Kwon is a bit special, as Jin does not speak english in the serie and is almost never mentioned in the closed-captions, which creates alignment errors between screenplay and closed captions. These difficulties illustrate some of the interesting challenges in ambiguously labeled datasets. As we can see, the most difficult classes are the ones with which another class is strongly correlated in the ambiguous label confusion matrix. This is consistent with the theoretical bounds we obtained in section 4.3.4, which establish a relation between the class specific error rate and class specific degree of ambiguity $\epsilon$.

Quantitative results are shown in Table 6.1. We measure error according to average 0-1 loss with respect to hand-labeled groundtruth labeled in 8 entire episodes of LOST. Our



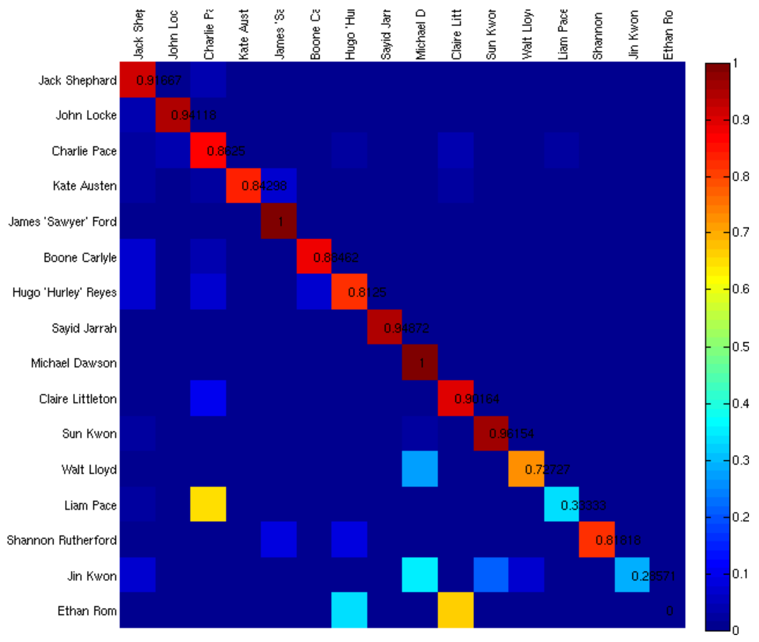


Figure 6.3: **Top:** Label distribution of top 16 characters in Lost. Element $D_{ij}$ represents the proportion of times true class $i$ was seen with class $j$ in the ambiguous bags, and $\sum_j D_{ij} = 1$. **Bottom:** Confusion matrix of predictions (without the additional cues). Element $A_{ij}$ represents the proportion of times true class $i$ was classified as class $j$, and $\sum_j A_{ij} = 1$. Class priors for the most frequent, median, and the least frequent characters in LOST are Jack Shephard, $14\%$; Hugo Reyes, $6\%$; Ethan Rom, $0.7\%$.

model does significantly better than all baseline methods, and we will further improve results in section 6.4. We now compare several methods to obtain the best possible precision at a given recall, and propose a confidence measure to this end.

### 6.3.1 Improved confidence measure for precision-recall evaluation

We contribute a **confidence measure** that significantly improves precision-recall in a **refusal to predict** scheme. A refusal to predict scheme, as used by [Everingham et al., 2006] for a related character naming application, is a way to obtain a precision recall curve for a multiclass classification problem: for a given recall rate $r$, and total number of examples $m$, we extract the $r \cdot m$ most confident predictions (according to some confidence measure we need to define) and compute precision $p$ on those examples. The curve is obtained by varying $r \in [0, 1]$.

Figure 6.4 shows several natural choices for defining a confidence score $confidence(x)$ for an example $x$:

1. the **max** score is the simplest:

$$confidence(x) = \max_a g^a(x)$$

2. the **ratio** score is an improvement, used by [Everingham et al., 2006]. It is defined as:

$$confidence(x) = \max_a \frac{\exp(g^a(x))}{\sum_b \exp(g^b(x))}$$

3. the **relative** score can be defined as the difference between the best and second best scores over all classifiers $(g^a)_{a\in\{1..L\}}$:

$$confidence(x) = g^{y^*}(x) - \max_{a\in\{1..L\}-\{y^*\}} g^a(x),$$

   where $y^* = \arg\max_{a\in\{1..L\}} g^a(x)$.

4. we can define the **relative-constrained** score as an adaptation to the ambiguous setting; we only consider votes among ambiguous labels $Y$:

   $$confidence(x) = g^{y^*}(x) - \max_{a \in Y - \{y^*\}} g^a(x),$$

   where $y^* = \arg\max_{a \in Y} g^a(x)$.

There are some limitations with all of those measures, in the case where we have some errors in ambiguous labels ($z \notin Y$ for the true label $z$). This can occur for example if we restrict them with some heuristics to prune down the amount of ambiguity, such as the ones we consider in section 6.4 (mouth motion cue, gender, etc). At **low recall**, we want maximum precision, therefore we cannot trust too much the heuristic used in relative-constrained confidence. At **high recall**, the errors in the classifier dominate the errors in ambiguous labels, and relative-constrained confidence gives better precision because of the restriction it introduces.

We introduce a **hybrid** confidence measure that gets the best of both worlds, performing well in the two regimes. We provide experimental support for this in figure 6.4. It is defined as an intermediate solution between relative confidence and relative-constrained confidences: let $h_r^a(x)$ (where $r$ is the given recall) be defined as:

$$h_r^a(x) = \begin{cases} g^a(x) & \text{if } a \in Y \\ (1-r)g^a(x) + r \min_b g^b(x) & \text{else} \end{cases} \tag{6.1}$$

Our confidence $confidence_r(x)$ is then simply defined by applying confidence confidence on the new classifier $h_r^a(x)$. By design, in the limit $r \to 0$, $confidence_r(x)$ behaves like relative confidence. In the limit $r \to 1$, $h_r^a(x)$ is small for $a \notin Y$ and so $confidence_r(x)$ behaves like relative confidence.

### 6.3.2 Precision-recall

We compute precision-recall curves, plotted in figure 6.4, using the different confidence measures introduced. We will now further improve those results by considering additional cues for naming.

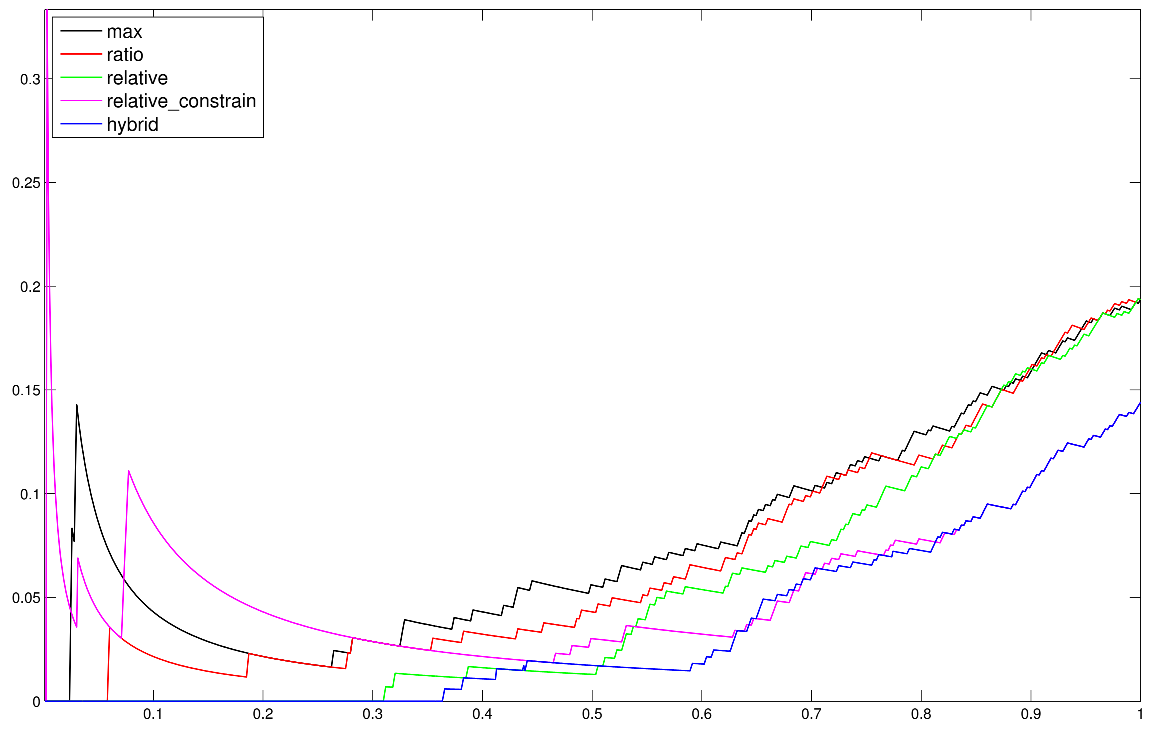


Figure 6.4: Our **hybrid** confidence score versus other confidence scores. $x$ axis: recall; $y$ axis: error rate for our ambiguous learning method on 16 episodes of Lost (top 16 characters). The simplest is the **max** confidence score, and performs rather poorly as it ignores other labels. **relative-constrain** improves the high precision/low recall region by considering the margin instead. The **relative-constrain** improves the high-recall/low-precision region by only voting among the ambiguous bags, but it suffers in high-precision/low recall region because some ambiguous bags may be erroneous (not containing the groundtruth). Our **hybrid** confidence score gets the best of both worlds, and is **asymptotically the same** as the other 2 methods in their best regimes, dominating all the other schemes almost everywhere.

| LOST (#labels, #episodes) | (8,16) | (16,16) | (32,16) |
|---|---|---|---|
| Naive | 14% | 16.5% | 18.5% |
| ours ("mean") | 10% | 14% | 17% |
| ours + mouth motion + gender | **6%** | **11%** | **13%** |

Table 6.1: Misclassification rates of different methods for naming in TV show LOST: naive model, our model, and our model improved with additional constraints of mouth motion and gender, on the top $\{8,16,32\}$ characters across 16 episodes. For comparison, other baseline methods' performances for (#labels, #episodes) = $(16,16)$ are *knn*: 30%; *Model 1*: 44%; *chance*: 53%.

## 6.4 Additional cues

We investigate with additional features to further improve the performance of our system: mouth motion, grouping constraints, gender. Final misclassification results are reported in table 6.1.

### 6.4.1 Mouth motion

The first cue is the **mouth motion** cue, introduced in [Everingham et al., 2006]. It uses lip motion during dialog to detect presence or absence of a speaking character on screen. After aligning screenplay and closed captions, we know *who* is supposed to speak in a certain time interval $[t_1, t_2]$. Absence of lip motion in this interval for a given face track is an evidence that this person is *not* the speaker. If on the other hand that face has significant lip motion, it is likely to be the speaker. We use a very similar approach to [Everingham et al., 2006] to detect lip motion: for each face in a face track $x$ that intersects with $[t_1, t_2]$, we compute $s(x)$, the maximum normalized cross correlation between a patch centered around the detected mouth (after registration) and neighboring patches in the next frame, allowing for translations of $\pm 20$ pixels in each direction. The aggregate score for the face track covering an interval $[t'_1, t'_2]$ is:

$$\bar{s} = E_{[t_1,t_2]\cap[t'_1,t'_2]}[s(x)] \tag{6.2}$$

We use three thresholds $t_{\min}, s_{\min}, s_{\max}$ defined as follows: we refuse to predict lip motion for face tracks with $|[t_1, t_2] \cap [t'_1, t'_2]| < t_{\min}$ or $\bar{s} \in [s_{\min}, s_{\max}]$. Otherwise, we detect lip motion if $\bar{s} > s_{\max}$ and absence of lip motion if $\bar{s} < s_{\max}$. Those thresholds are set to achieve $90\%$ precision for detecting lip-motion and absence of lip motion on a validation set that we groundtruthed (containing 100 tracks with extracted and registerd lip regions and the corresponding motion label for the whole track). In our experience the cue is more accurate for detecting absence of lip motion (a person talking cannot be completely still). Lip motion can be the artifact of a sudden head rotation or lighting change, as observed in [Everingham et al., 2006], and is harder to classify correctly. We suspect more advanced classifiers and feature representations would improve performance of the lip motion detector, which plays a significant role in [Everingham et al., 2006].

**Contribution.** Our main contribution to this cue is to reduce the temporal scope of each utterance (initially a very coarse estimate based on closed captions time stamps) using the **forced alignment** described in section 5.3.3. This gives a tighter alignment of potential lip motion and dialog, thereby reducing false positive / false negative motions in non-speech intervals. Quantitative results for precision of the alignment are reported in figure 5.2.

**Usage.** We use the cue as follows in our ambiguous label setting: suppose we have a face track $x$ with ambiguous label set $Y$ and a temporally overlapping utterance from a speaker corresponding to $a \in \{1..L\}$. We modify $Y$ as follows:

$$Y := \begin{cases} \{a\} & \text{if lip motion} \\ Y & \text{if refuse to predict or } |Y| = \{a\} \\ Y - \{a\} & \text{if absence of lip motion} \end{cases} \tag{6.3}$$

Note, this could result in multiple faces in one frame being classified as $a$ in the case where our lip motion detected fired for two overlapping face tracks. Our temporal grouping model in chapter 7 addresses this issue and many related ones.

### 6.4.2 Gender constraints

We discuss gender classification in section , which gives a score $\gamma(x)$ for each face track $x$. We assume known the gender of names mentioned in the screenplay (using automatically extracted cast list from IMDB, see section ). We use gender by filtering out the labels that do not match by gender the predicted gender of a face track, if the confidence is greater than a threshold (one threshold for females, one for males, are set on a validation data to achieve $90\%$ precision for each direction of the gender prediction). Thus, we modify ambiguous label set $Y$ as follows:

$$Y := \begin{cases} Y & \text{if gender uncertain} \\ Y - \{a : a \text{ is male}\} & \text{if gender predicts female} \\ Y - \{a : a \text{ is female}\} & \text{if gender predicts male} \end{cases} \tag{6.4}$$

### 6.4.3 Grouping constraints

We also experiment with **grouping constraints**, which provide only indirect information on labels. Assuming with have a clustering of face tracks, grouping constraints state that face tracks grouped together should be labeled in the same way. We propose a very simple must-not-link constraint, which states $y_i \neq y_j$ if face tracks $x_i, x_j$ are in two consecutive shots (expressing alternation of shot/reaction shots that are common in dialogs for example). The constraint is only set for the cases where there are two characters locally: the ambiguous label sets are $Y_i = Y_j = \{a, b\}$. This is therefore equivalent to a grouping constraint with inverted labels (the multiclass case is more tricky, as discussed in our related work chapter).

We also propose *groundtruth* grouping cues for comparison, and compare the use of both real and groundtruth grouping cues in the experiments. The groundtruth constraints are: $y_i = y_j$ for each pair of face tracks $x_i, x_j$ of the same label, and that are separated by at most one shot. Thus, the number of constraints remains linear with the number of tracks. Details on how to incorporate such constraints as convex terms in our ambiguous loss function are provided in section 8.4.

We describe more elaborate constraints in detail in section 8.3.1 for a related naming application.

## 6.5 Ablative analysis

Figure 6.5 shows the ablative, showing precision-recall curves for several methods: our method (mean), the naive baseline, and our method augmented with the additional cues we discussed:

- **mean**: our method
- **naive**: the naive model from the previous chapter
- **link**: simple must-not-link constraints from shot alternation,
- **gender**: gender cue for simplification of ambiguous bags;
- **mouth**: mouth motion cue for detecting the speaker with synchronous mouth motion
- **mouth+gender**: a combination
- **groundtruth grouping**: link constraints with perfect grouping
- **groundtruth mouth**: mouth motion cue with perfect lip motion detection

We see that the constraints provided by mouth motion help most, followed by gender and link constraints. The best setting (without using groundtruth) combines the former two cues. Also, we notice once again a significant performance improvement of our method over the naive method.

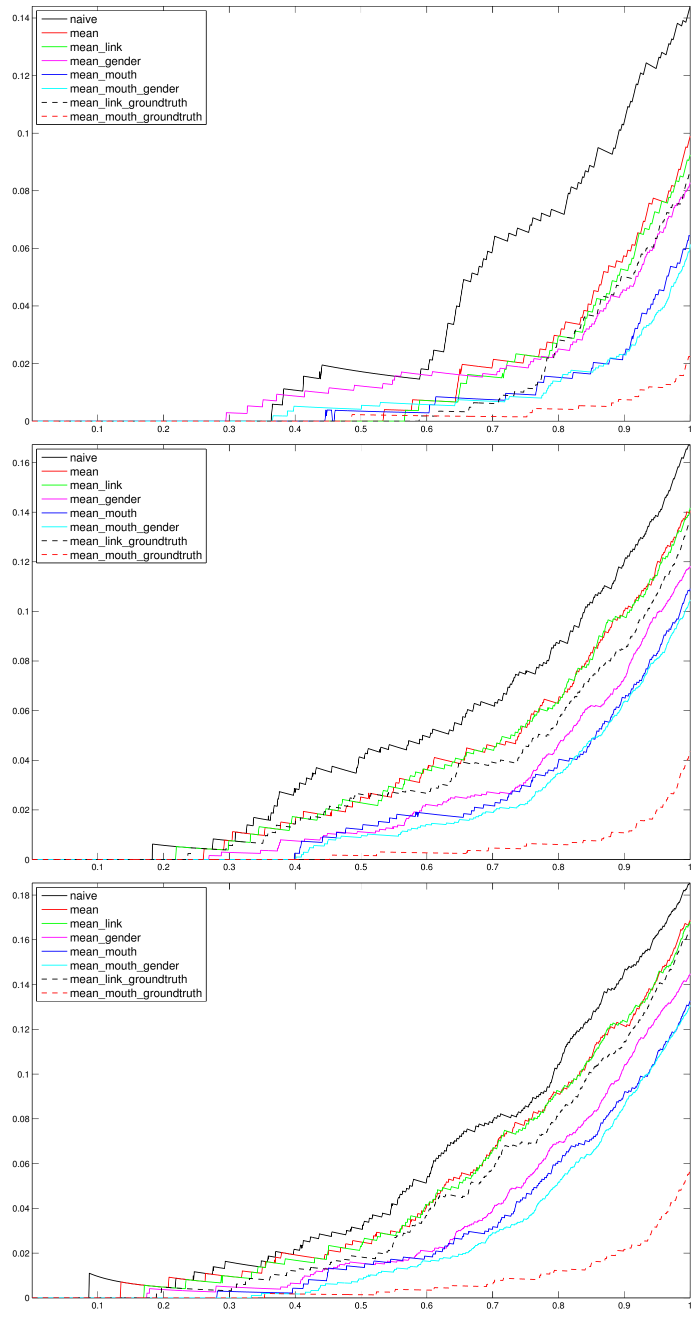


Figure 6.5: Ablative analysis. $x$-axis: recall; $y$-axis: error rate for character naming across 16 episodes of Lost, and the 8,16,32 most common labels (respectively for the top, middle, bottom plots). We compare our method, **mean**, to the **naive** model and show the effect of adding several cues to our system. **link**: simple must-not-link constraints from shot alternation, **gender**: gender cue to prune ambiguous bags; **mouth**: mouth motion cue to detect the speaker with synchronous mouth motion; we also consider the combination **mouth+gender**, as well as how well we perform with perfect sub components such as **groundtruth grouping** constraint and **groundtruth mouth** motion cue.

## 6.6 Qualitative results and Video demonstration

We show examples with predicted labels and corresponding accuracy, for various characters in figures 6.6, 6.7, 6.8, 6.9 for LOST and figures 6.10, 6.11, 6.12, 6.13 for CSI. Those results were obtained with the basic system of section 6.3. Full-frame detections can be seen in figure 6.1.

### 6.6.1 Video demonstration

We propagate the predicted labels of our model to all faces in the same face track throughout an episode. Video results of several episodes can be found at the following website `http://www.youtube.com/user/AmbiguousNaming`. In the videos we show the resolved label below each face detection. As a reference, images of the most common characters corresponding to those episodes are in Figure 6.14.

## 6.7 Conclusion

We have presented an effective approach for learning from ambiguously labeled data, where each instance is tagged with more than one potential labels. We show bounds on the classification error, even when all examples are ambiguously labeled. We compared our approach to strong competing algorithms on two naming tasks and demonstrated that our algorithm achieves superior performance. We attribute the success of our approach to better modeling of the mutual exclusion between labels, compared to the naive multi-label approach. Moreover, unlike recently published techniques that address similar ambiguously labeled problems, our method does not rely on heuristics and does not suffer from local optima of non-convex methods.

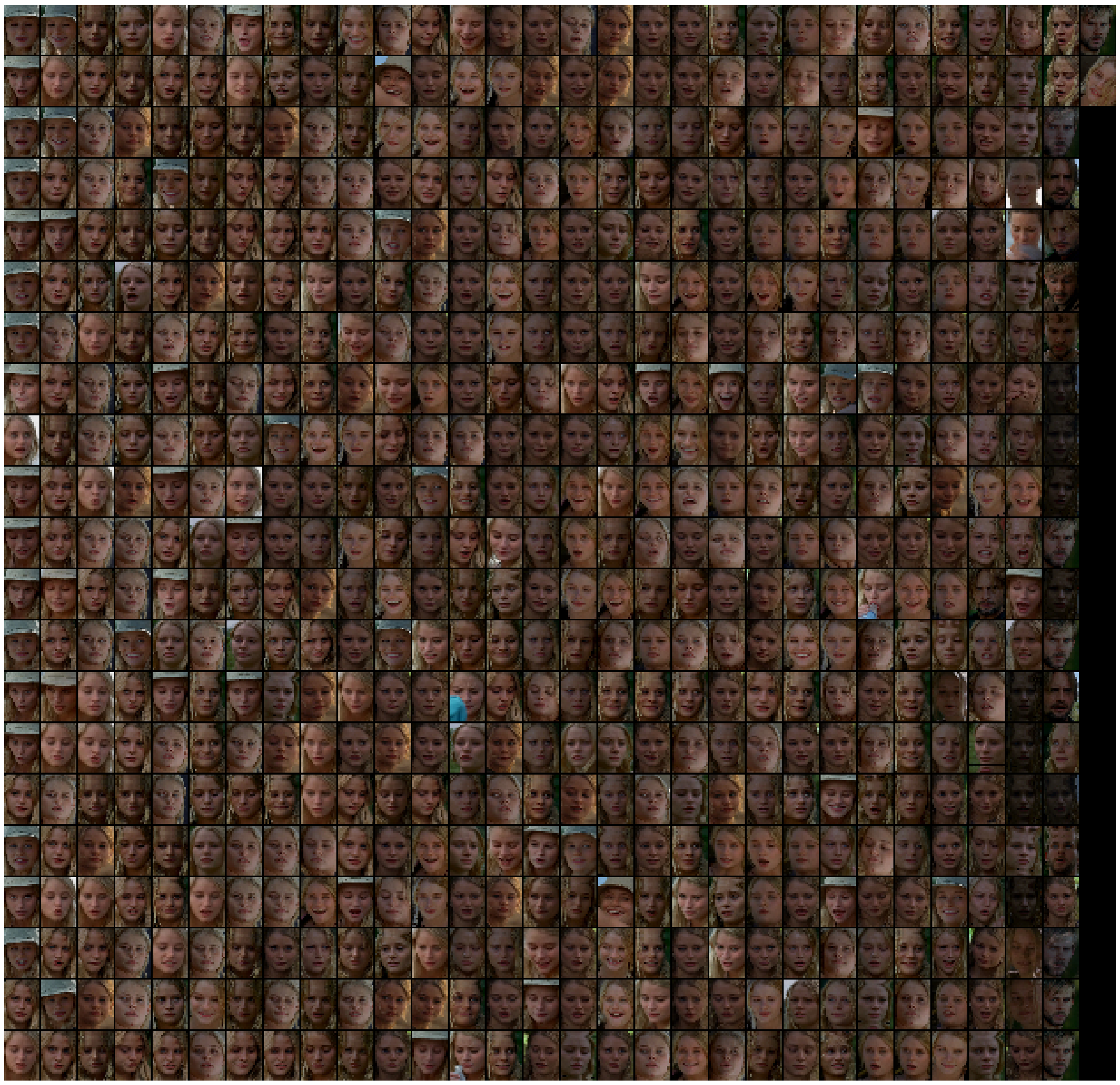


Figure 6.6: Examples classified as Claire in the LOST data set using our method. Results are sorted by classifier score, in column major format; this explains why most of the errors occur in the last columns. The precision is $97.4\%$.

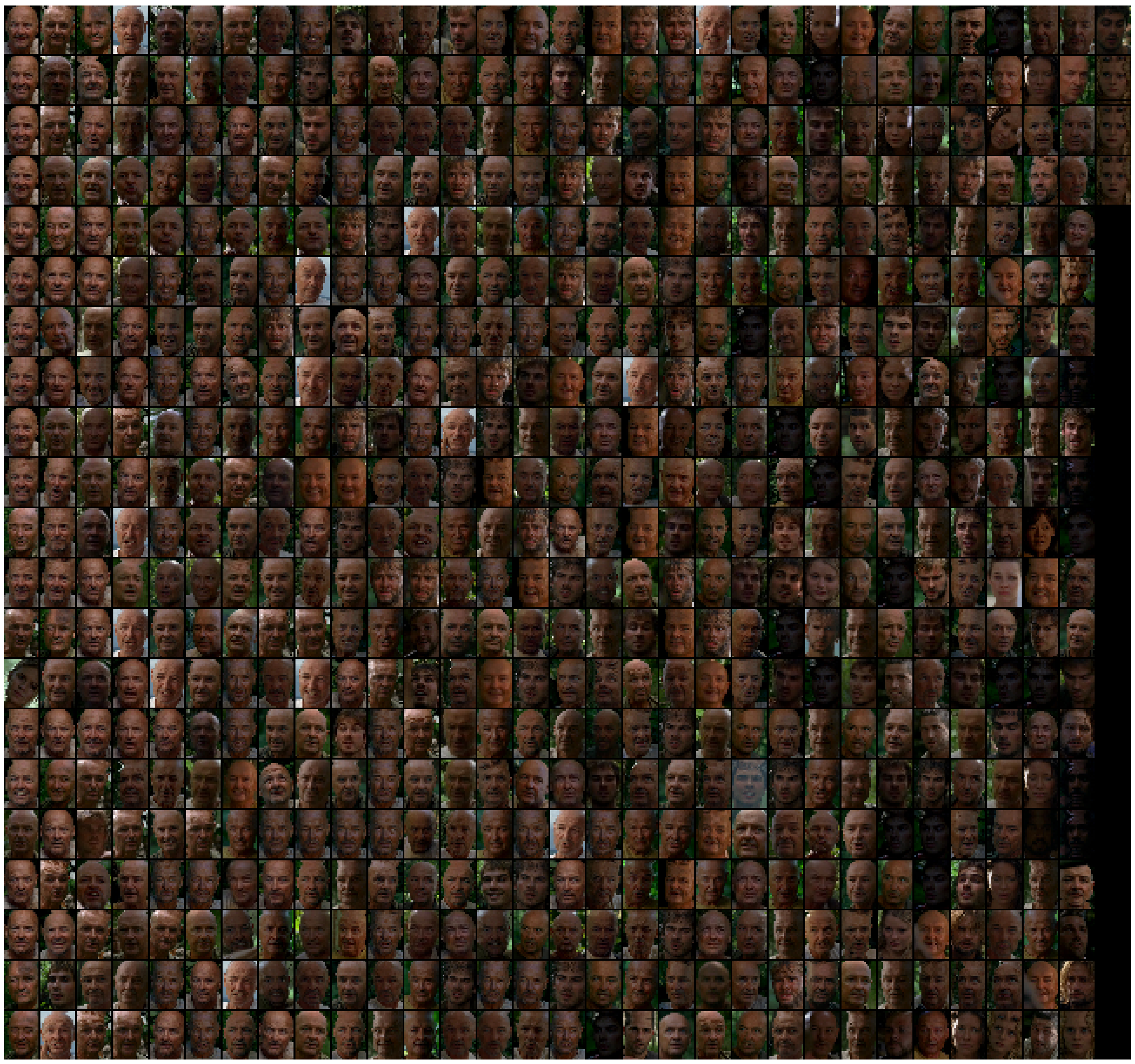


Figure 6.7: Examples classified as Locke in LOST. The precision is $78.7\%$.

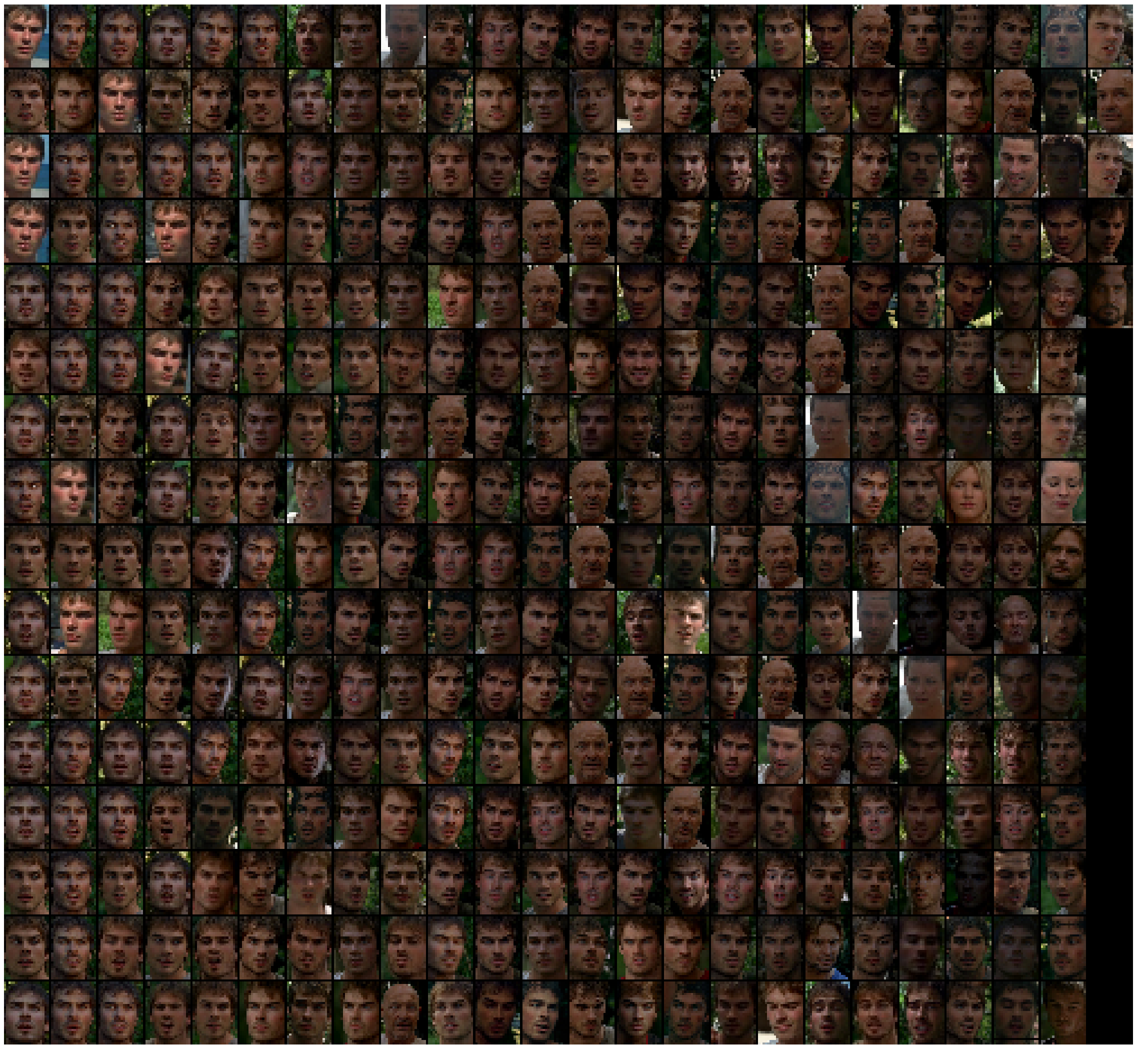


Figure 6.8: Examples classified as Boone in LOST. The precision is $90.1\%$.

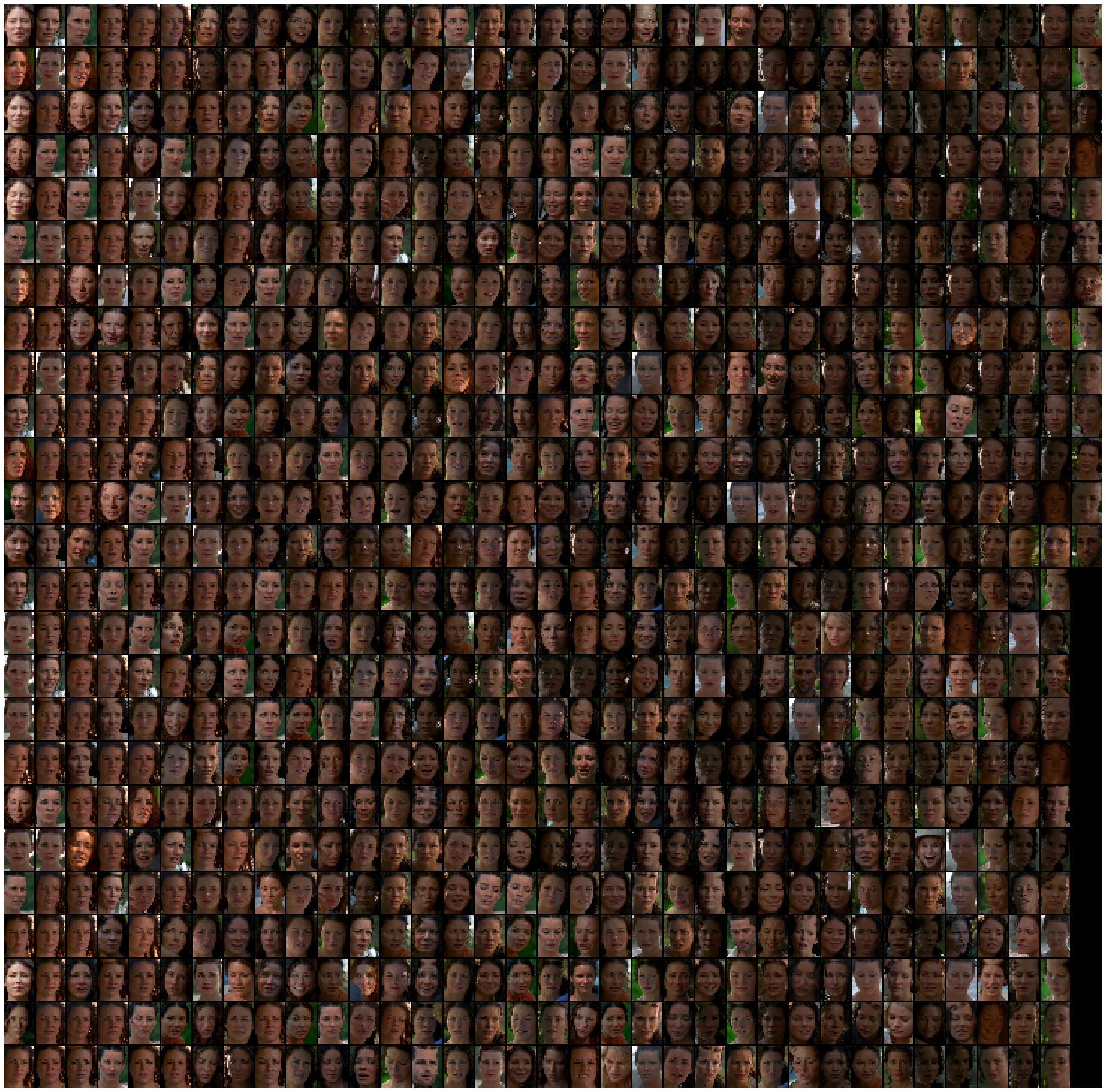


Figure 6.9: Examples classified as Kate in LOST. The precision is $97.5\%$.

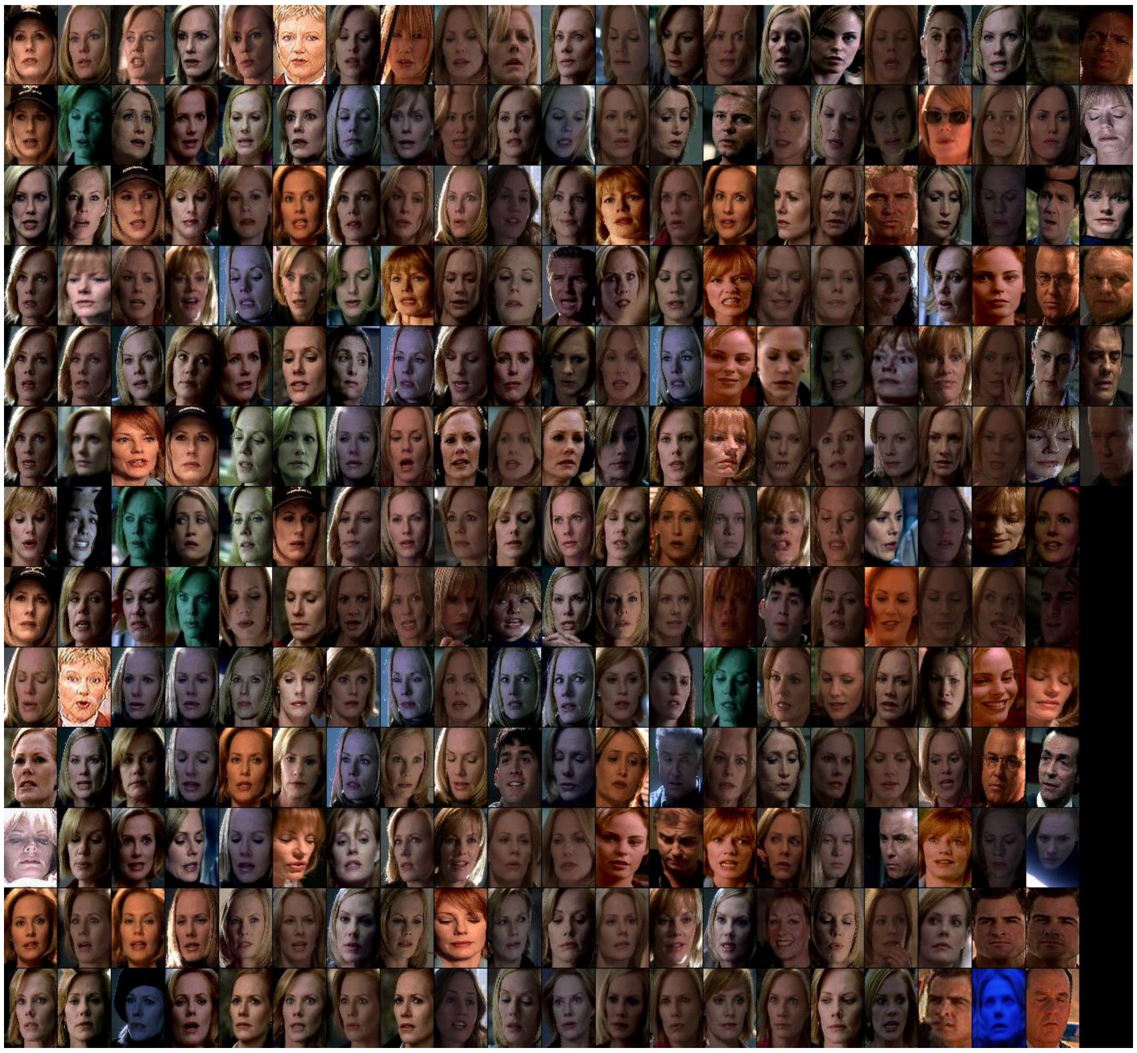


Figure 6.10: Examples classified as Catherine Willows in CSI. The precision is 85.3%.

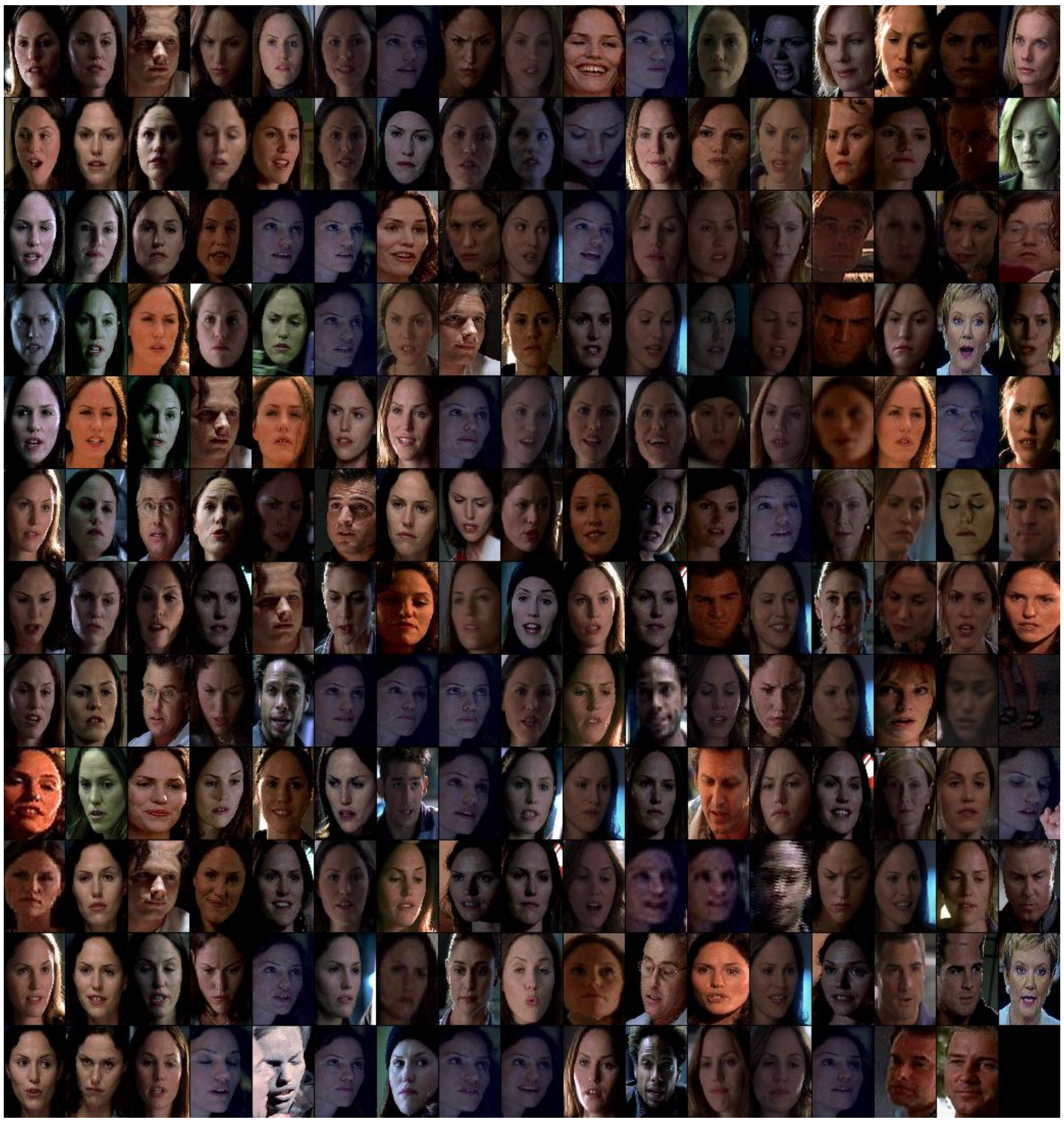


Figure 6.11: Examples classified as Sara Sidle in CSI. The precision is 78.3%.

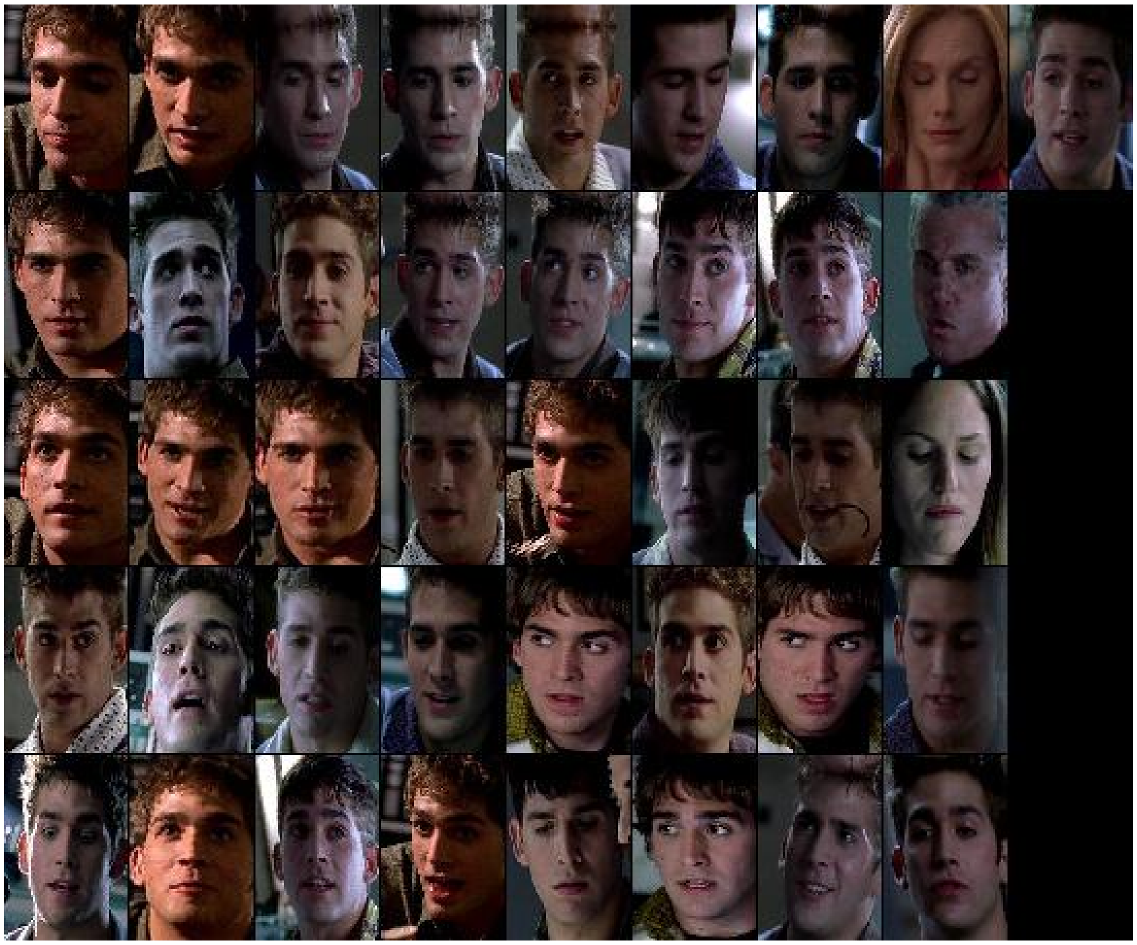


Figure 6.12: Examples classified as Greg Sanders in CSI. The precision is 92.7%.

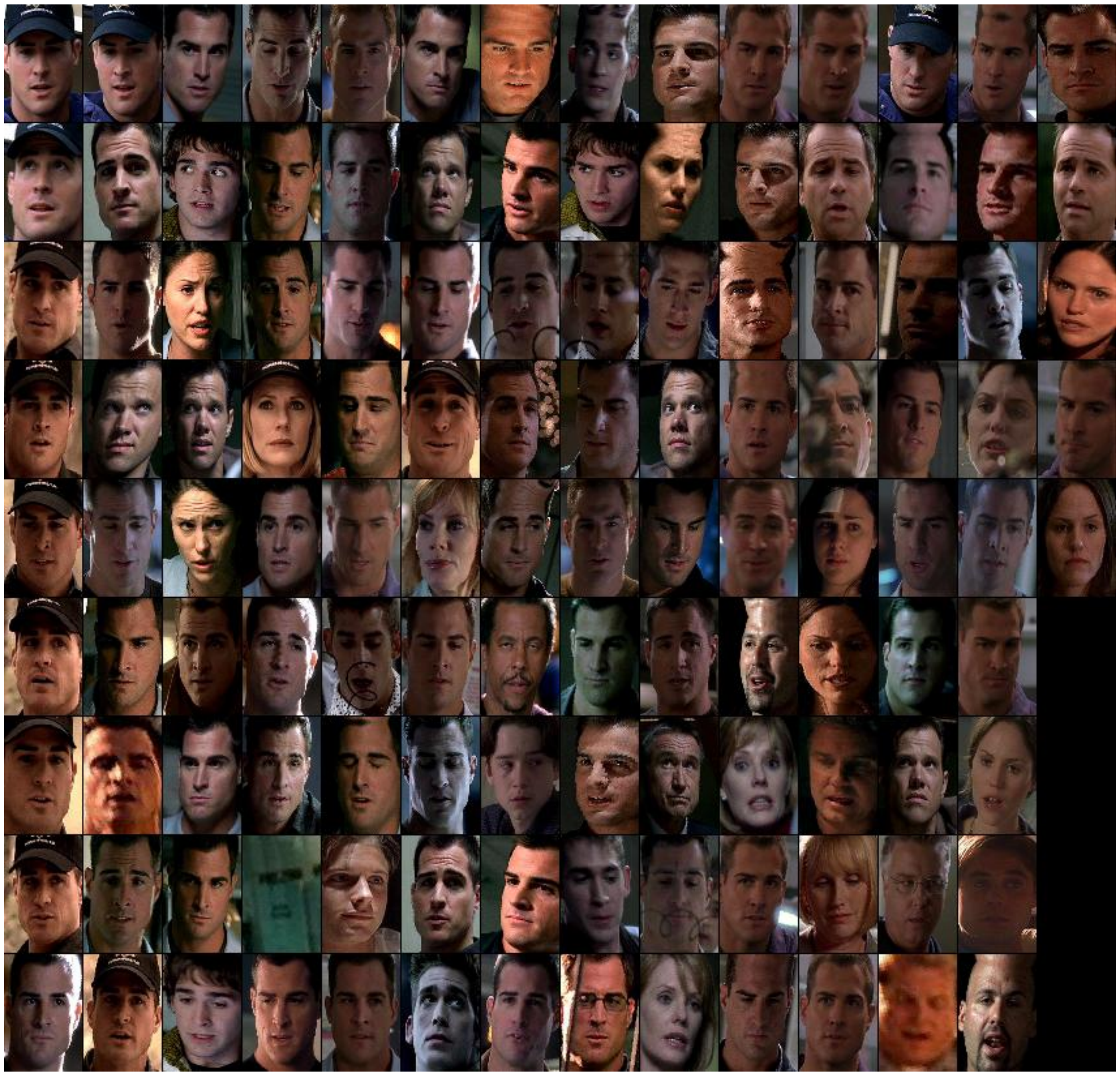


Figure 6.13: Examples classified as Nick Stokes in CSI. The precision is 66.4%.

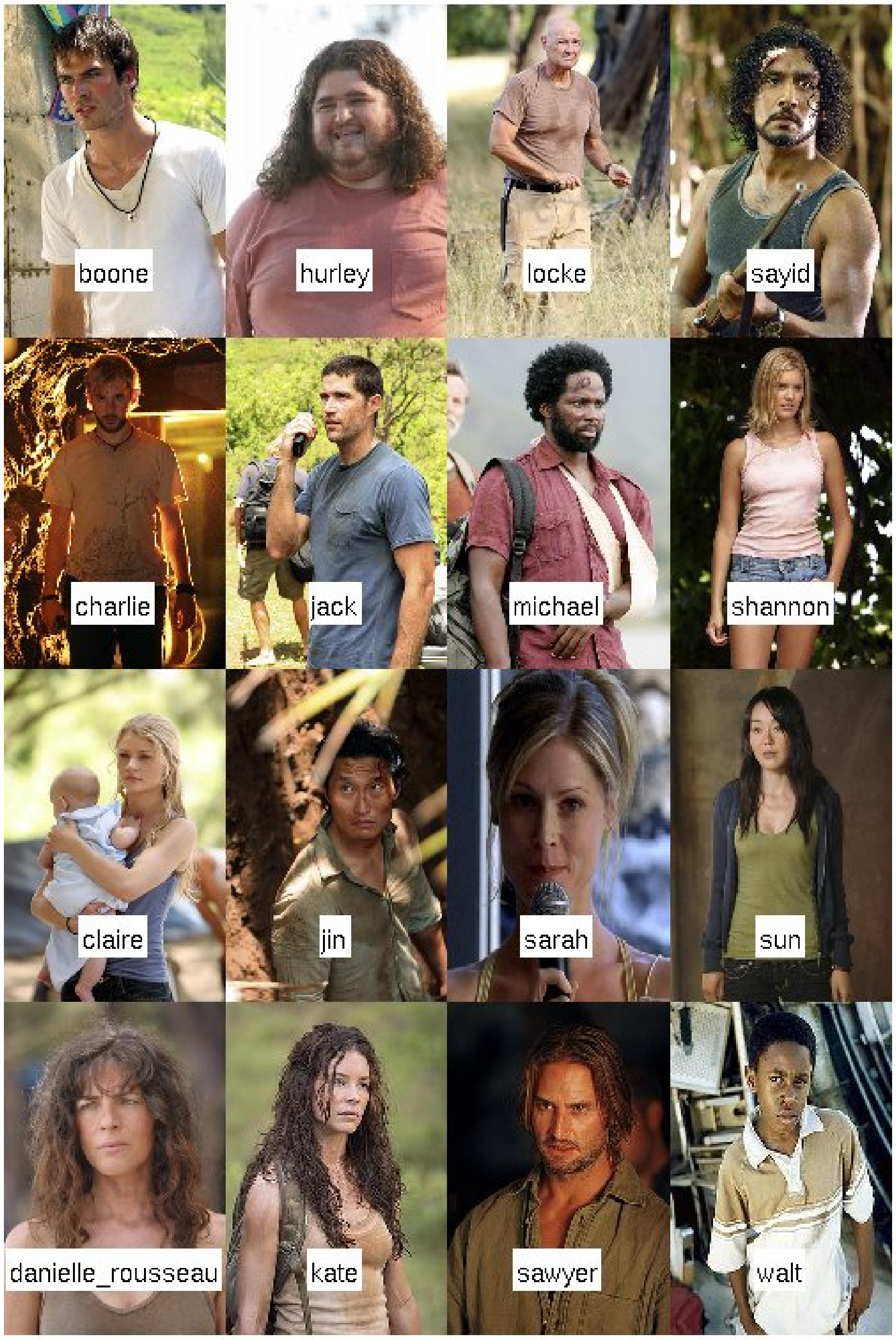


Figure 6.14: Groundtruth examples for 16 common characters in LOST.

# Chapter 7

# Temporal Grouping

## 7.1 Introduction

Up till now we have considered various weakly supervised tasks: alignment of shots, threads ands scenes to a screenplay, action retrieval, character naming. All of those involve access to a screenplay and closed captions. In this chapter and the next one, we relax this assumption to various degrees. This chapter in particular addresses the totally unsupervised case where the video is the only input. Our focus in this chapter shifts from *naming* characters to *grouping* characters, which, in the ideal case, amounts to attaching a unique identifier to each encountered person. If this task could be done reliably, a small amount of supervision or even a single name reference per person could in theory be sufficient to name *all* the characters. This is in fact the strategy we employ in the next chapter, where we show that character naming can still be done relatively accurately based on dialog only, exploiting the *sparse, indirect, and noisy* cues provided by first, second and third person name references. The related work of [Ramanan et al., 2007] exploits similar ideas with a hierarchical clustering and a small amount of manual supervision.

The ability to perform unsupervised character grouping in a video sequence can give rise to other applications beyond naming, such as face retrieval[Sivic et al., 2005], action

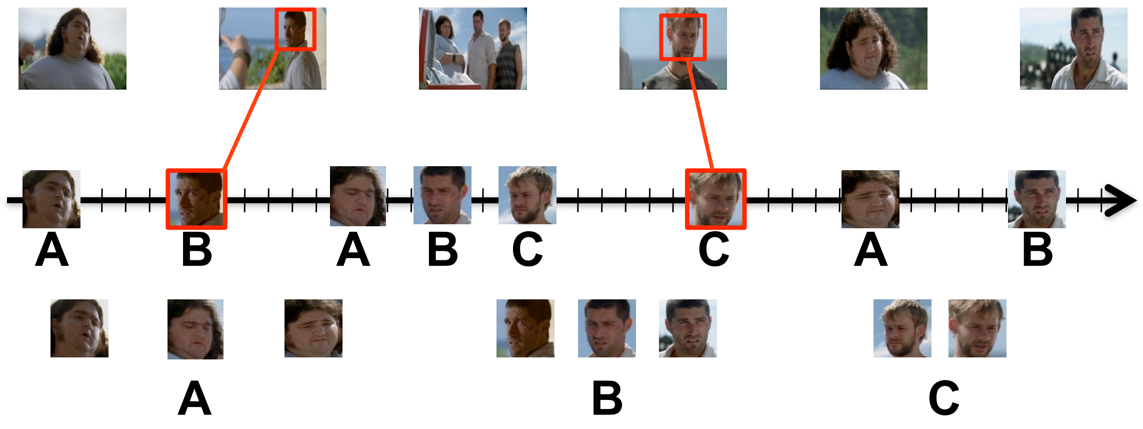


Figure 7.1: We model a movie or TV show as a sequence of face tracks ordered temporally. We propose a temporal grouping model that groups faces based on not only appearance but also on local film structure cues. The resulting clusters of faces are subsequently assigned a name based on dialog references in the next chapter.

recognition, cast list generation[Fitzgibbon and Zisserman, 2002], and video summarization. The lack of supervision makes the task significantly harder than in previous chapters, however, as we lack the oft-occurring textual cues that "glue" together otherwise very different facial appearances of a given character. To overcome these challenges, we exploit the rich structured information present in video which arises from 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 [Smith, 2005] for an excellent introduction to the topic). A number of those rules can be used as constraints for grouping, see table 7.2. 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. In this chapter, we propose a novel *temporal grouping model* that groups faces across an episode based on not only appearance but also on local film structure 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. In the next chapter we will see how the resulting clusters of faces are subsequently assigned a name by learning a classifier from partial label constraints derived from dialog. We evaluate our temporal grouping model on several hours of TV and movies, achieving significantly higher accuracy than several strong baselines.

## 7.2 Related Work

**Grouping.** Related to several of the approaches above is general learning of global distance metrics or similarity functions to cluster globally. There is a large body of work on metric learning and learning to cluster in different applications; a few applicable techniques include [Nowak and Jurie, 2007, Ferencz et al., 2006]. 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.

A related recent work [Cowans and Szummer, 2005] proposes a graphical model for simultaneous partitioning and labeling of graphs with low tree-width, and modeled effectively labeled partitions. Their setting was significantly different, however, being restricted to pairwise interactions and focusing on a binary labeling task.

**Face representation.** A number of visual cues can be used for grouping faces together. In [Ramanan et al., 2007], 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 and clothing to group faces across shot boundaries. At longer time scales (across episodes), they use facial appearance as the other cues are no longer reliable.

## 7.3 Approach

We model a movie as a linear sequence of face tracks, ordered temporally into a chain (see Figure 8.2). 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 possible characters in the episode (known *a priori*). In a typical TV show, the number of labels $L \approx 20$.

A straightforward approach to this problem would be to learn a sequence model over the set of labels $\{1..L\}$. Then, assuming a local scope/clique size $k$ (i.e., a $k^{th}$-order Conditional Random Field or CRF), the number of states considered at any time 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 in a scene involving up to 10 face tracks, thus be able to express shot alternation, scene structure etc.

An underlying problem with this representation is its redundancy. In our temporal grouping task, we only wish to learn patterns of the label sequence, not anything specific about particular labels, which are arbitrary and whose values change in each example. For example, assuming a local scope of size 3, there is no difference between the sequence of labels $z_1, z_1, z_2$ and the sequence $z_4, z_4, z_3$, although in a CRF over labels, both sequences are considered during inference.

Our approach addresses these problems by representing the state of a size-$k$ scope as one of the set of possible partitions of $k$ elements. The number of partitions of size $k$ grows much slower than the number of possible label sequences of size $k$ (see Table 7.1, and next section), giving us a much more compact representation. As a concrete example, even limiting the label set to $L = 4$ labels (compared to 20 for the characters in a typical TV show), the state space over all partitions is at least 10 times smaller than the state space over all labelings for sequences of size $k \leq 10$. This allows us to significantly increase the size of the neighborhood relations captured by our model.

This motivates our 2 phase approach. In the first phase, we learn a large-scope partition classifier, and use it to locally cluster a sequence of faces and utterances. In the second phase, we use first, second and third person references to propagate labels to our small set of clusters.

**Local Partitioning.** We propose to learn a model which explicitly enumerates the set of possible partitions for a local window of face tracks and predicts a partition based on local constraints, similarities and interactions across modalities. In particular we are able to capture scene-level interactions such alternations, presence in same image, etc.

## 7.4 How to represent a partition

Let $\mathcal{P}_n$ denote the set of partitions of the set $\{1, ..., n\}$, $\forall n \geq 1$. The size of $\mathcal{P}_n$ is given by the $n^{th}$ Bell number, defined recursively as $B_n = \sum_{i=0}^{n-1} \binom{n-1}{i} B_i$, with $B_1 = 1$. We denote $|y|$ as the **cardinality** (number of sets or clusters) of a partition $y$.

There are many ways to represent partitions. For $n = 3$, using set notation we have exactly 5 possible partitions: $\{(1,2,3), (1,2)(3), (1)(2,3), (1,3)(2), (1)(2)(3)\}$. Note that order within the parenthesis and outside the parenthesis does not matter, so that, for example, $(2,3)(1) \simeq (3,2)(1) \simeq (1)(3,2)$. A more convenient notation is to associate to a partition $y \in \mathcal{P}_n$ a **partition matrix** $Y \in \{0,1\}^{n \times n}$ representing the equivalence class relationship defined by $y$: $Y_{ij} = 1$ if $i, j$ are in the same cluster (*i.e.* equivalent), as illustrated in Figure 7.2. We will use $y$ and $Y$ interchangeably.

We define the **restriction** $y_I$ of a partition $y \in \mathcal{P}_n$ (with partition matrix $Y$) to a subset $I \subset \{1, ..., n\}$ as the partition $y \in \mathcal{P}_{|I|}$ whose partition matrix is $Y(I, I)$, obtained by extracting rows and columns of $Y$ indexed by $I$.

Inversely, for $m \in \mathbb{Z}$ we define an **extension** operation $\tau^m[y]$ on a partition $y \in \mathcal{P}_n$ as the unique partition $y' \in \mathcal{P}_{n+|m|}$ such that $|y'| = |y| + |m|$, and $y'_{[m+1,m+n]} = y$ (for $m > 0$), or $y'_{[1,n]} = y$ (for $m \leq 0$).

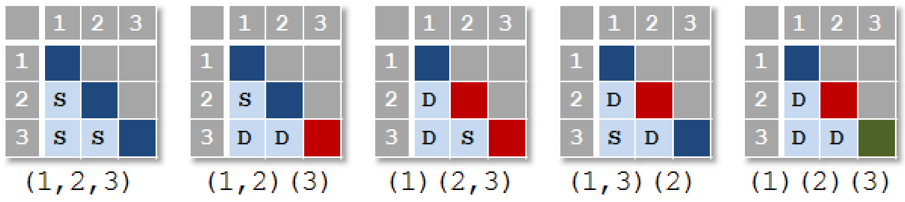


Figure 7.2: Matrix representation of all 5 partitions of size $n = 3$. Each pair of elements (representing face tracks) can either be same (S) or different (D), yielding different clusterings. Cluster labels are represented by colors along the diagonal.

| **n** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|
| $B_n$ | 5 | 15 | 52 | 203 | 877 | 4140 | 21147 | $10^5$ |
| $4^n$ | 64 | 256 | 1024 | 4096 | 16384 | 65536 | $3 \cdot 10^5$ | $10^6$ |

Table 7.1: $B_n$: number of partitions of $n$ elements, representing our state space. This is much more compact than a standard sequence labeling state space over a neighborhood of $n$ elements, even with a very small label set ($L = 4$).

## 7.5 Partition CRF

Conditional Random Fields (CRF), have been introduced in [Lafferty et al., 2001] to model labeling of sequence data. We introduce here a new model for *partitioning* of sequence data. Our partition CRF can handle general high-order clique potentials defined on local partitions, and in particular allows for complex, interleaved partitions. Thus, it is well adapted to model the alternation of shots and reaction shots that are typical in movie sequences. We define a **partition CRF** of order $k$ a graphical model over *partitions* $y$ of a sequence of observations $\mathbf{x} = (x_1, ..., x_n) \in \mathcal{X}^n$ with arbitrary potential function $g : \mathcal{P}_k \times \mathcal{X}^n \times \{1..n\} \to \Re$:

$$P(y|x) \propto \prod_{i=1}^{n-k+1} \exp(g(y_{[i,i+k-1]}, x, i)) \tag{7.1}$$

Note, without any additional constraint on $y$, the above expression is degenerate for $n > k$ as it gives equal probability to certain partitions regardless of the potential function: for example, take $k = 2, n = 3$ and the partitions $\{(1,3)(2)\}$ and $\{(1)(2)(3)\}$. To remedy this we introduce the concept of **memory-limited partition**, which is the partition analog of memory-limited TSP, introduced in section 3.4.1. We say that $y \in \mathcal{P}_n$ is $k$-limited, denoted as $y \in \mathcal{P}_n^k$ if:

$$\forall i, \forall j \geq i+k, Y_{ij} = 1 \implies \exists i' \in \{i+1.., j-1\} : Y_{i'j} = 1 \tag{7.2}$$

In other words, two consecutive elements $i, j$ in a cluster are within a distance at most $k-1$. In our example, this rules out $\{(1,3)(2)\}$. A partition CRF is thus defined for $y \in \mathcal{P}_n^k$ to avoid this degeneracy.

## 7.6 Inference

We describe here how to perform exact MAP (Maximum Aposteriori) inference in a partition CRF, which is the following problem:

$$y^* = \arg\max_{y \in \mathcal{P}_n^k} P(y|x) = \arg\max_{y \in \mathcal{P}_n^k} \sum_{i=1}^{n-k+1} g(y_{[i,i+k-1]}, x, i) \tag{7.3}$$

The main difficulty is that partition $y$ is defined globally, introducing dependencies between consecutive windows. We break this chain of dependencies by a decomposition of $y$ into a collection of sequence-consistent partitions, defined next.

### 7.6.1 Sequence-consistent decomposition

Informally, a sequence-consistent partition is a collection of partitions on overlapping windows that agree on the overlap. Given a partition $y \in \mathcal{P}_n^k$, we define its sequence-consistent decomposition as:

$$dec(y) = (y_{[i,i+k-1]})_{i=\{1..n-k+1\}} \in (\mathcal{P}_k)^{n-k+1} \tag{7.4}$$

In order to solve the MAP, we are interested in the inverse operation, which is not always well defined. In particular, two consecutive partitions in the decomposition must consistent with each other: we say that $y', y'' \in \mathcal{P}_k$ are **consistent**, denoted as $y' \sim y''$, if:

$$y'_{[2,k]} = y''_{[1,k-1]} \tag{7.5}$$

Figure 7.3 illustrates the concept of consistency between consecutive partitions. Note, this relation is transitive but not symmetric. Reciprocally, a sequence $y^1 \sim y^2 \sim \ldots \sim y^{n-k+1}$ with consistent consecutive partitions in $\mathcal{P}_k$ defines a unique partition $y \in \mathcal{P}_n^k$ such that $dec(y) = (y^i)_{i=\{1..n-k+1\}}$:

$$y = y^1 \odot y^2 \odot ... \odot y^{n-k+1} \tag{7.6}$$

where $y' \odot y''$ denotes the concatenation of two partitions $y' \in \mathcal{P}_{k'}$, $y'' \in \mathcal{P}_{k''}$ (with $k' \geq k''$), with corresponding matrix defined as:

$$\mathbb{1}(\tau^{-1}[Y']\tau^{k'-k''+1}[Y''] + \tau^{k'-k''+1}[Y'']\tau^{-1}[Y']) \tag{7.7}$$

which is of order $k' + 1$. Note, this expression simply expresses the symmetric transitive closure of the equivalence relationship of the partition.

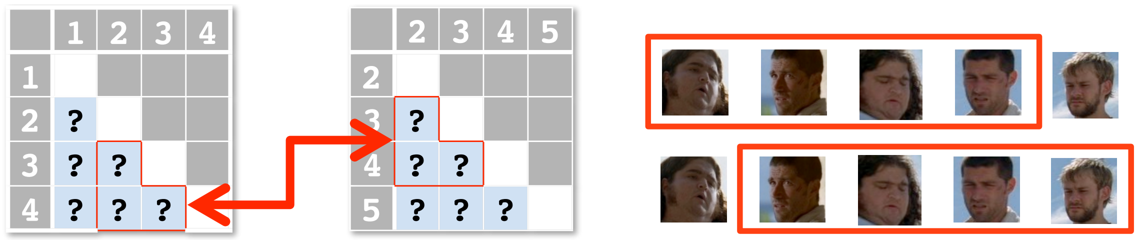


Figure 7.3: Left: Consistency between partitions of successive windows of size $k = 4$ (Right) are enforced. Same/different relationships for elements 2,3,4 must be the same as elements 1,2,3 in the next window.

## 7.6.2 Dynamic Programming Solution

With this decomposition, the MAP reduces to:

$$y^* = y^{1*} \odot y^{2*} \odot ... \odot y^{n-k+1*} \tag{7.8}$$

$$(y^{i*})_{i=1..n-k+1} = \operatorname*{arg\,max}_{y^1 \sim y^2 \ldots \sim y^{n-k+1}} \sum_i g(y^i, x, i). \tag{7.9}$$

which can be solved with a standard Viterbi-like dynamic program: the following score

$$s^j(y^j) = \max_{y^1 \sim y^2 \ldots \sim y^j} \sum_i g(y^i, x, i). \tag{7.10}$$

can be computed with the recursion:

$$s^1(y^1) = g(y^1, x, 1) \tag{7.11}$$

$$s^i(y^i) = \max_{y^{i-1} \sim y^i} s^{i-1}(y^{i-1}) + g(y^i, x, i), \quad \forall i > 1 \tag{7.12}$$

The running time for computing the dynamic programming table is $O(n \cdot B_{k+1})$, as it is easily shown that the number of consistent consecutive partitions is $\leq B_{k+1}$. Decoding time is $O(n)$, using a standard book-keeping and starting from the end of the table.

### 7.6.3 Cardinality-Constrained MAP

The MAP assignment finds a single partition. In practice it is useful to have control over the number of clusters in the partition, which allows 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 $C \in \{1..n\}$:

$$y_C^* = \arg\max_{y \in \mathcal{P}_n^k} P(y|x, |y| = C) \tag{7.13}$$

The **cardinality-constrained MAP** can be solved with another dynamic program, by augmenting the state-space from $y \in \mathcal{P}_k$ to $(y, c) \in \mathcal{P}_k \times \{1..n\}$: $c$ represents the number of clusters seen so far. We define $(y', c')$ and $(y'', c'')$ to be consistent if the following holds:

$$(y', c') \sim (y'', c'') \quad \text{if} \begin{cases} y' \sim y'' \\ c'' = c' + \mathbb{1}(|y''| = |y''_{[1,k-1]}| + 1) \end{cases}$$

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 computed in $O(nkB_{k+1})$ to retrieve optimal C-way partitions for *all* values of $C$. Each one requires an $O(n)$ decoding pass, starting from the best scoring state $(y^*, C^*)$ in the last table column that satisfies $C^* = C$.

## 7.7 Learning

The temporal grouping model we consider has a lot of parameters, and tuning them by hand would be overly complicated. We propose to learn our model as a structured multi-class classification, predicting for each window $\mathbf{x} = \{x_1, \ldots, x_k\}$ its associated partition $y \in \mathcal{P}_k$. The number of classes $|\mathcal{P}_k| = B_k$ is large (see table 7.1) but structured, which makes learning easier via parameter sharing. We treat each window independently in a sequence at training time, which is justified since the large window we use already captures a lot of context information. Note, other structured learning schemes are possible, for example treating the entire sequence as input. We also investigated with a perceptron rule, which performed worse.

We assume that the potential function is linearly parametrized via a $d$-dimensional feature mapping $\mathbf{f} : \mathcal{P}_k \times \mathcal{X}^k \rightarrow \Re^d$, taking the following form:

$$g(y, x, i) = g^y(x_i, ..., x_{i+k-1}) = g^y(\mathbf{x}^i) = \mathbf{w} \cdot \mathbf{f}(\mathbf{x}^i, y)$$

where $\mathbf{w} \in \Re^d$ is the set of parameters which we will estimate from a set of $n$ labeled partitions $\{(\mathbf{x}^i, y^i)\}$. The window groundtruth partition $y^i \in \mathcal{P}_k$ is obtained from the sequence groundtruth partition $y \in \mathcal{P}_n^k$ via the decomposition introduced in section 7.6.1.

### 7.7.1 Discrete loss

We follow a standard approach in structured prediction, which aims at minimizing over all hypothesis $g$ the loss incurred by the best prediction $g^*(\mathbf{x}) = \arg\max_y g^y(\mathbf{x})$. This leads to minimizing the following discrete loss:

$$\mathcal{L}_{01}(g) = \sum_i \mathcal{L}_{01}^i(g) \quad \text{with} \quad \mathcal{L}_{01}^i(g) = \ell(y^i, g^*(\mathbf{x}^i)), \tag{7.14}$$

where the individual loss $\ell(y^i, y) : \mathcal{Y} \times \mathcal{Y} \mapsto R^+$ penalizes partitions based on how much they differ from the gold standard, as discussed in section 7.7.4.

### 7.7.2 A convex large-margin formulation

We propose to estimate $g$ by minimizing a convex upperbound on the discrete loss function (7.14). Let $\psi(z)$ be a standard convex binary loss such as exponential, logistic or hinge, which is decreasing and upperbounds the step function $\mathbb{1}(z \le 0)$. We define a convex loss function as $\mathcal{L}(g) = \mathcal{R}(g) + \sum_i \mathcal{L}^i(g)$, where

$$\mathcal{L}^i(g) = \sum_{y \in \mathcal{Y}} \ell(y^i, y)\psi\left(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)\right). \tag{7.15}$$

and $\mathcal{R}(g)$ is a regularization term. We show in proposition 7.7.1 that $\mathcal{L}(g)$ is indeed a (convex) upperbound on $\mathcal{L}_{01}(g)$.

Note that other choices of convex upperbounds are possible; the one we chose has the advantage of not assigning any special meaning to $g^y(\mathbf{x}) = 0$ since only the difference $g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)$ matters. In contrast, the following loss

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

performed worse. We believe the main reason is that the latter formulation tries to simultaneously drive $g^{y^i}$ up and $g^y$ down even when $y$ is very close to $y^i$, which makes the problem harder to fit.

**Proposition 7.7.1** $\mathcal{L}^i_{01}(g) \le \mathcal{L}^i(g)$.

**Proof** Given that $\ell(y^i, y) \ge 0 \quad$ and $\quad \psi(z) \ge \mathbb{1}(z \le 0)$,

$$\begin{aligned}
\mathcal{L}^i_{01}(g) &= \ell(y^i, \arg\max_y g^y(\mathbf{x}^i)) \\
&\le \sum_{y \in \mathcal{Y}} \ell(y^i, y)\mathbb{1}(g^{y^i}(\mathbf{x}^i) \le g^y(\mathbf{x}^i)) \\
&\le \sum_{y \in \mathcal{Y}} \ell(y^i, y)\psi(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i))).
\end{aligned}$$

### 7.7.3 Optimization

Using the same technique as in section 4.6, we can convert the loss (7.15) into a standard supervised binary classification and solve the resulting optimization using a Quadratic Program, Support Vector Machine, or boosting for example. The last option performed best in our experiments, as it allows us to perform feature selection during learning. We use the exponential loss $\psi(z) = \exp(-z)$ and iteratively optimize the objective with boosting using decision stumps as weak classifiers. We show in appendix C.1 that we can select the optimal stump across all feature dimensions at each round of boosting in **linear time** $O(d \cdot n \cdot B_k)$, and present the corresponding **algorithm**. Note, this requires some care since the stumps are defined on the scores $g^y(\mathbf{x})$, *not* on the differences $g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)$.

### 7.7.4 Hamming loss for partitioning

Our individual loss $\ell(y^i, y) : \mathcal{Y} \times \mathcal{Y} \mapsto R^+$ should penalize partitions based on how much they differ from the true partition $y^i$. 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. Other choices include Normalized Mutual Information, and Clustering Accuracy which involves finding the optimal mapping between true and predicted clusters. We choose the following expression, which allows to penalize differently splits and merges compared against groundtruth:

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

where the binary matrices $Y, Y^i$ refer to partitions $y, y^i$ and $\mathbb{1}(\cdot)$ is the indicator function. We observed that smaller values of $\alpha_{merge}$ performed best on a validation set, and set $\alpha_{merge} = 0.1$ in all our experiments.

## 7.8 Grouping Cues

Our representation can encode any type of local cues or constraints defined over a neighborhood of size $\leq k$, and can depend on both the input and proposed partition. Our base features are defined 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 multiple statistics over those. We **condition on y**, the proposed partition, distinguishing whether the pair of face tracks lies in the same cluster ($Y_{ij} = 1$):

$$f_{\text{same}}(x, y) = s(\{d(x_i, x_j)\}_{i,j:Y_{ij}=1}) \tag{7.16}$$

$$f_{\text{different}}(x, y) = s(\{d(x_i, x_j)\}_{i,j:Y_{ij}=0}) \tag{7.17}$$

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 also considered (but didn't use) features that only depend on the partition $y$, representing local partition patterns. We could define many other features on $y$ or $(x, y)$ but decided to keep our feature set relatively simple to compare to previous work. We describe below the set of pairwise distances $d(x_i, x_j)$ used. Each feature is tuned automatically using max-margin estimation described earlier.

### 7.8.1 Appearance cues

Appearance cues are illustrated in figure 7.4.

**Best registered face.** In a preprocessing stage, each face is registered using 4 control points (eyes, nose and mouth), using a method similar to [Everingham et al., 2006]. 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 each color spaces $f_{\text{c}}$: $d_{registered}(x_i, x_j) = ||f_{\text{c}}(x_i) - f_{\text{c}}(x_j)||_1$.

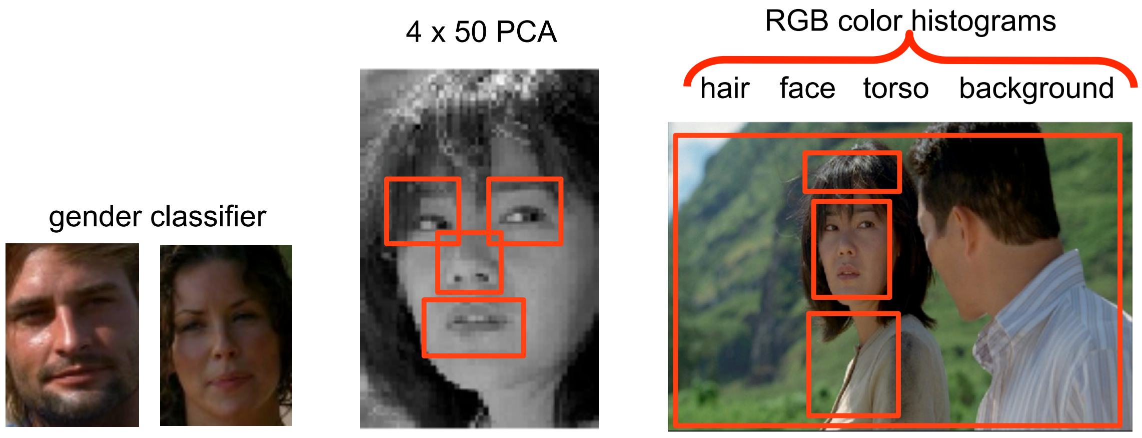


Figure 7.4: Appearance cues for grouping.

**Color histograms.** Following [Ramanan et al., 2007], we also use color histograms for face, hair, torso. Figure 7.5 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 $\chi^2$-distance.

**Exemplars.** Again following [Ramanan et al., 2007], 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 [Ramanan et al., 2007] 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 $e_j^l$).

**Gender.** We collected $200,000$ images via Google Image search for common male and female names. These were registered as before and used to train a gender classifier, described in appendix A.1. Accuracy on a hold-out set was $83\%$. Each face track has a vector of gender classification scores $\gamma(x)$, which we aggregate in the following ways before defining distances: mean, max, median over the track, as well as their signs (predicted gender).

### 7.8.2 Video editing cues

A key contribution of our model is the ability to encode interesting local editing cues, some of which are summarized in table 7.2. We define the following additional distances based on such cues (and invite the reader to consider Figure 7.5, which illustrates many of the cues, while reading):

**Relative positioning of faces.** We compute distances and signed distances based on the $x$, and $y$ coordinates of the mean face position in each track. We also compute difference in log-space scale of each face.

**Relative difference in pose.** For each face part, after registering to a fixed scale, we compute relative distances in $x$,$y$ and $(x, y)$ coordinates, providing information about the orientation of the parts of the face relative to the face detection. The track is again represented by the best-registered face.

**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. We found that 8 bins per channel (amounting to $8^3$ total bins) performed best.

**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.

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

Table 7.2: Grouping cues with a scene, see text for details.

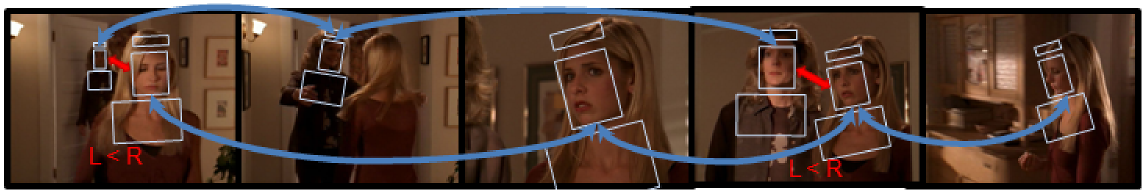


Figure 7.5: Example frames from the TV show *Buffy*. Face/hair/torso rectangles are shown for appearance cues. Red arrows mark do-not-group cues between face tracks appearing in the same frame. Blue arrows and "L $<$ R" labels indicate connections between face tracks resulting from the $180°$ degree 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.

## 7.9 Results for Temporal Grouping

### 7.9.1 Training and testing dataset

We trained our temporal grouping model using The Office (US) Season 2 Episode 5, a 1.5 hour episode. This provided us 1273 windows of 7 consecutive face tracks for training data. Based on cross-validation, we chose to use a partition size $k = 7$ because it performed the best out of the computationally feasible possibilities (see Figure 7.8). We evaluate our grouping on eight episodes of the TV show Lost (Season 1, Episodes 5-12), one episode of Buffy the Vampire Slayer (Season 4 Episode 1) and the movie Misery. We compare to 2 baselines, which are described next.

### 7.9.2 Baselines

**Baseline 1** is a global agglomerative method proposed by Ramanan et al. [Ramanan et al., 2007]. 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 7.8). Logistic regression is used to learn the best weighting of these 4 distance functions, and we feed this 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 colorspace $\ell_i$, and we use an $L_2$-norm distance function between 2 tracks $i$ and $j$: $d(i, j) = ||\ell_i - \ell_j||_2$.

### 7.9.3 Comparison

In all results, we measure performance as a trade off 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

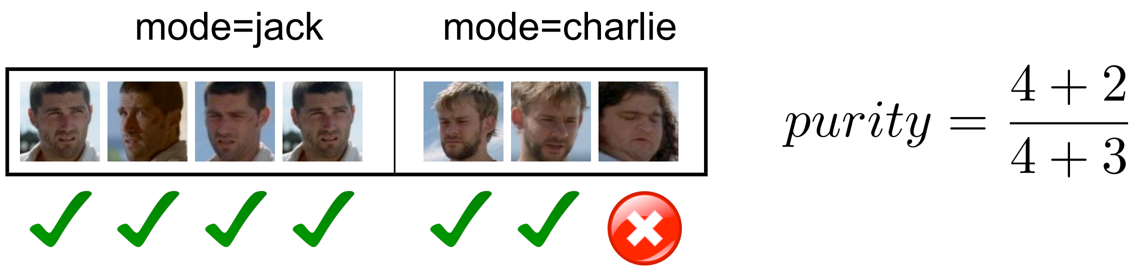


Figure 7.6: Illustration of purity of a clustering (see text for details).

for all face tracks in the cluster:

$$purity(y) = \frac{1}{n} \sum_{i=1}^{|y|} \sum_{j \in y_i} \mathbb{1}(z_{y_i[j]} = mode(z_{y_i})),$$

where $z$ denotes the true labels, $y_i[j]$ is the $j^{\text{th}}$ face track in the $i^{\text{th}}$ cluster of partition $y \in \mathcal{P}_n$, and $n$ is the number of face tracks (see figure 7.6).

In figure 7.7 we show our performance against the 2 baselines. There is an inherent trade off 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. In general, we outperform the baseline methods significantly when the number of clusters is greater than the optimal number of perfect clusters achievable by temporal grouping with limited scope $k$. 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.

### 7.9.4 Ablative analysis

We also analyze the efficacy of features via ablative analysis (see Figure 7.8, left). Starting with our full feature set, we sequentially remove sets of features and note the decrease in performance. The features in [Ramanan et al., 2007] clearly help, but are further significantly improved by the addition of gender and editing cues. Note that ablative analysis is

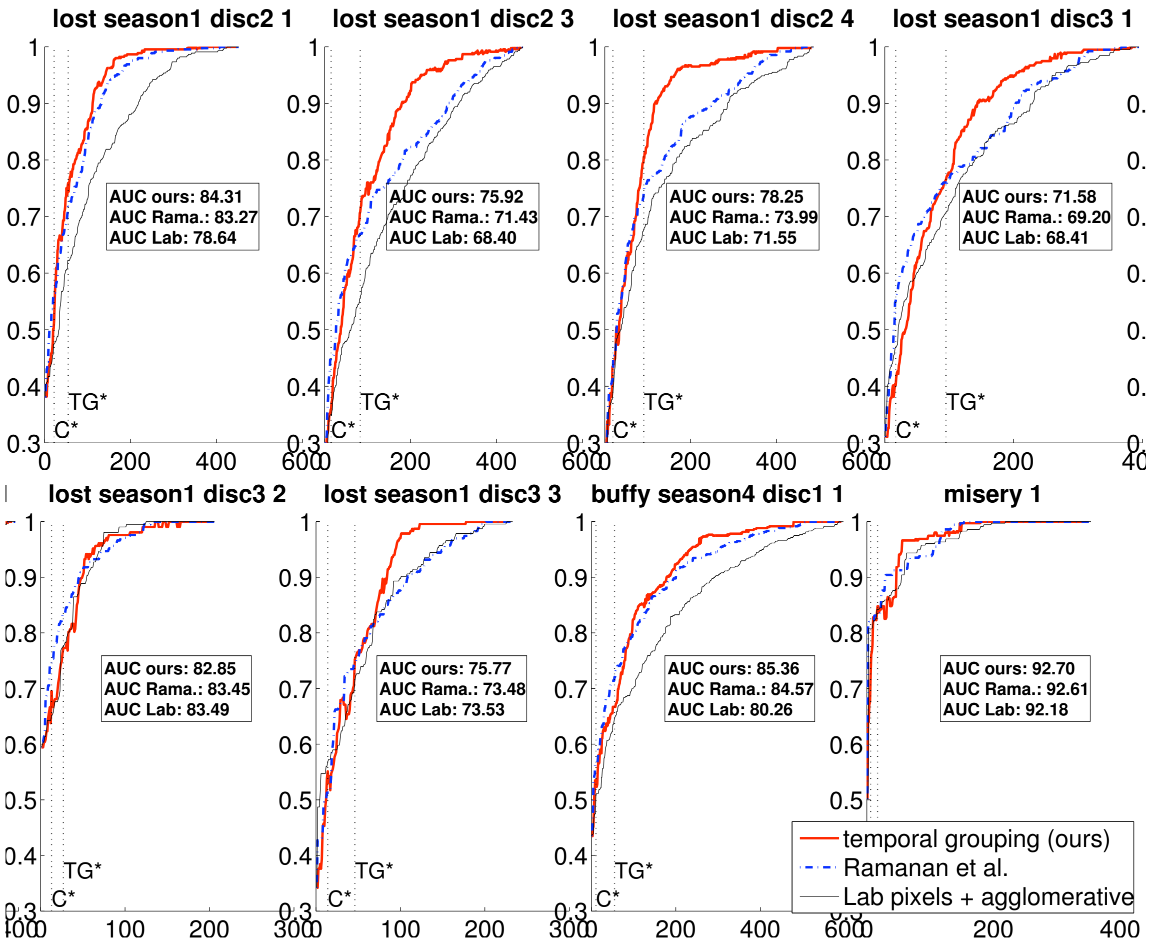


Figure 7.7: Grouping results comparing our temporal grouping against a state of the art method and an agglomerative baseline. $C^{\star}$ refers to the number of correct labels. $TG^{\star}$ denotes the optimal number of clusters temporal grouping can achieve via inference with a scope of size $k = 7$. 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. In all results, we measure performance as a trade off between the number of clusters (x-axis) versus the **purity** of the clustering (y-axis).

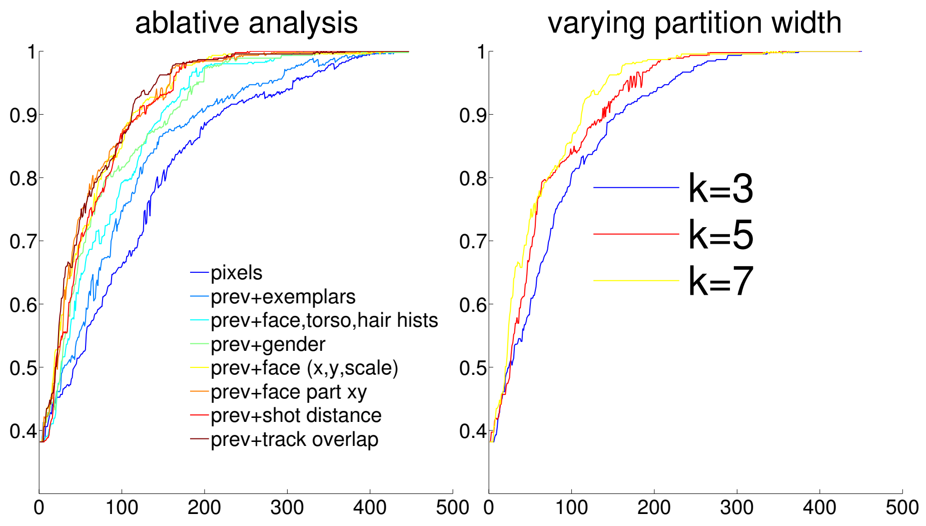


Figure 7.8: **Left:** Ablative analysis of the features present in our sequence model, on Lost Season 1 Episode 5 ("lost season1 disc2 1" in Figure 7.7). The biggest gains are obtained from adding gender and face location and scale. **Right:** Performance as we vary the local partition width $k$. We chose to use $k = 7$ in our experiments based on these results.

sensitive to the ablation ordering.

### 7.9.5 Effect of partition scope

In Figure 7.8, right, we measure performance as we vary the scope size $k$.

### 7.9.6 Qualitative results

Figure 7.9 shows qualitative results of our clustering on a test episode; we also show the original face tracks before clustering in figure 7.10.

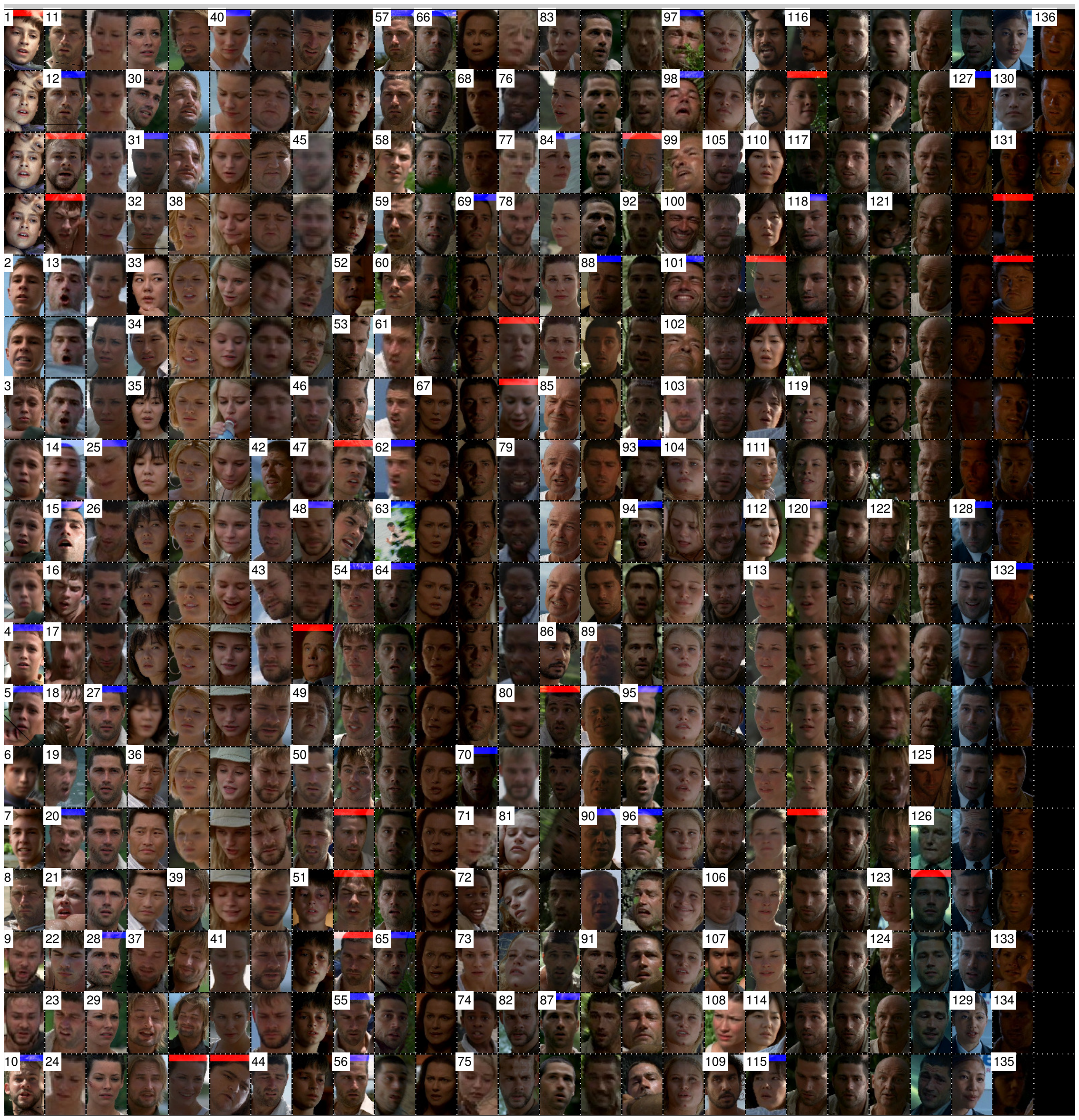


Figure 7.9: Grouping results on **Lost Season 1 episode 5**. The number of clusters was fixed to be 30% of the total number of face tracks. Blue represents bad splits, red represents bad merges (compared to groundtruth). For clarity, we have reordered faces according to their cluster id (shown as a white rectangle), and display the central face in each face track.

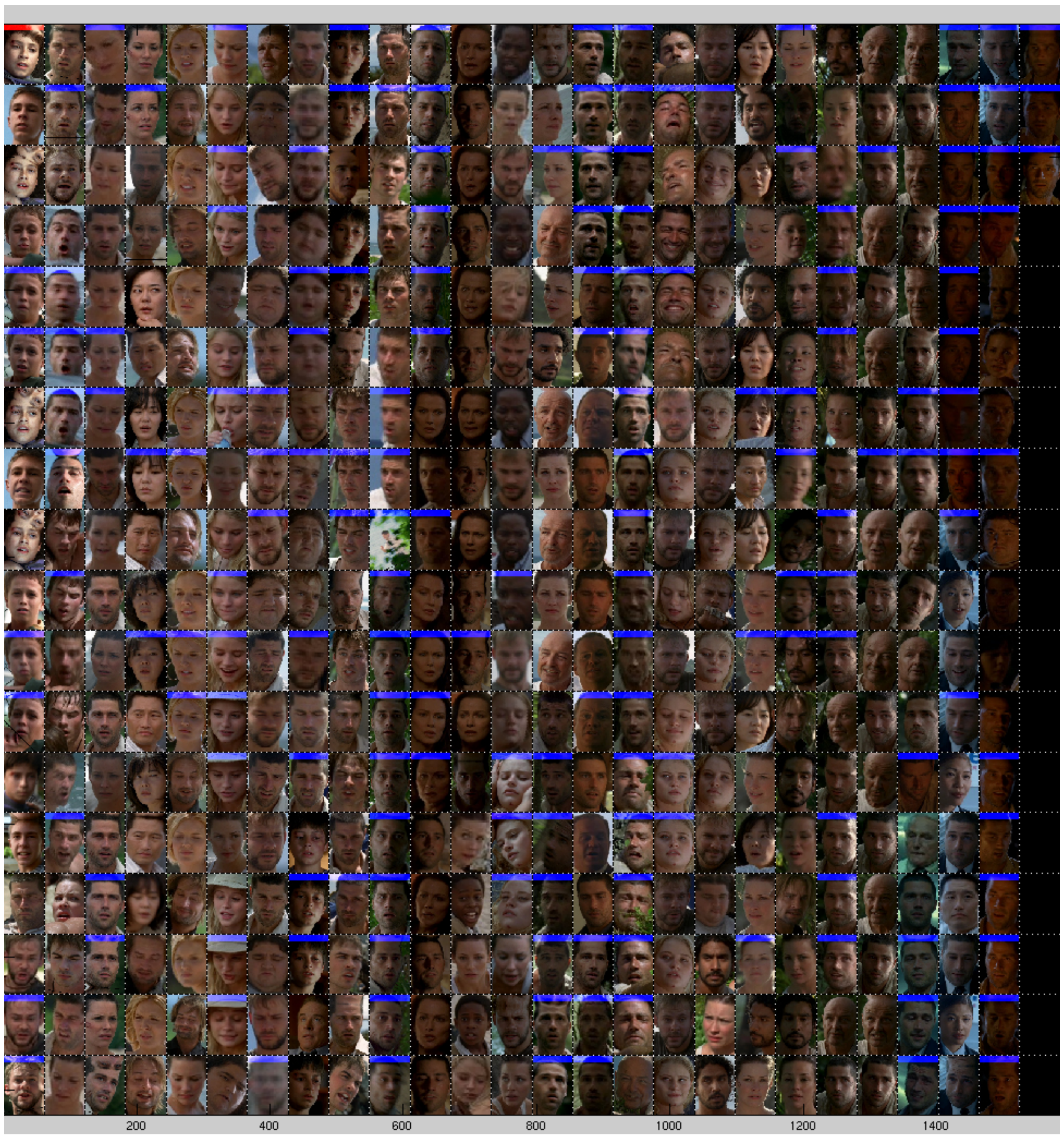


Figure 7.10: **Original ordering** of the faces clustered in figure 7.9. We show in blue the less common case when two consecutive face tracks show the same character

# Chapter 8

# Identity Resolution Without Screenplay

## 8.1 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 [Everingham et al., 2006, Cour et al., 2009a], 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 other cues such as mouth motion.

When you watch a movie, consider how you 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 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 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.

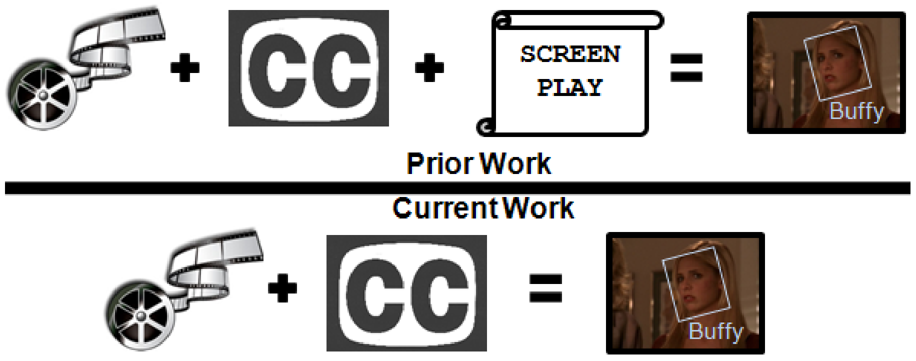


Figure 8.1: Diagram of prior and current work on character naming. When screenplay is not available, we know *what* is being said but not *who* said it. The only source of name information is dialog cues: first, second and third person references. (See text for details.)

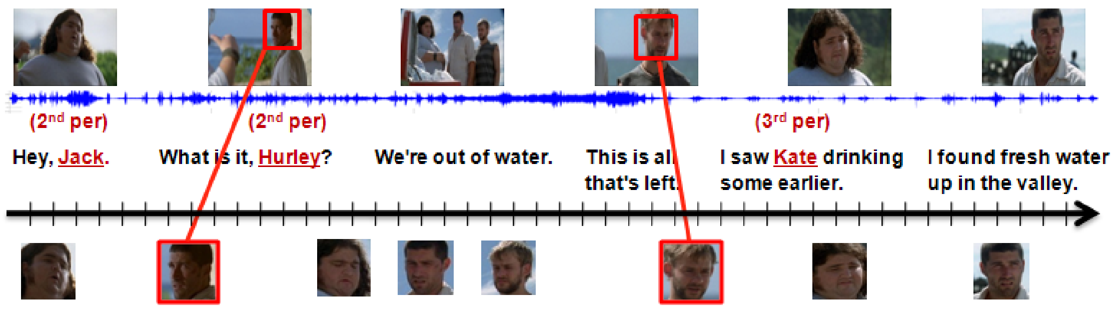


Figure 8.2: We present a weakly supervised classifier that incorporates multiple-instance-type constraints from dialog cues as well as local grouping constraints, introduced in the previous chapter.

In this work, we do not assume that the screenplay or any other 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 available to the typical movie-goer: the video and subtitle text/closed captions[1]. Eliminating dependence on screenplays allows broad application of our method to video collections with minimal human intervention. This is in contrast to prior work that requires either manual annotation or names extracted from a screenplay in order to provide training examples (see

[1]The need for subtitles could be avoided with reliable speech recognition. Fortunately, much of modern video content contains some form of closed-captions or subtitles, including YouTube videos.

figure 8.1).

There are several difficult challenges caused by lack of screenplays. Essentially, we know *what* is being said, but not *who* said it. First, second, and third person references in the speech are extremely sparse and often ambiguous. Changes in visual appearance over time make global grouping difficult. In our proposed approach, we first group faces based on local appearance cues and film structure, as presented in chapter 7. The resulting clusters of faces are subsequently assigned a name by learning a classifier from partial label constraints. The weakly supervised classifier incorporates multiple-instance-type constraints from dialog cues as well as local grouping constraints and gender constraints in a unified convex formulation. We evaluate on several hours of TV and present the first quantitative results on naming without screenplay.

## 8.2 Related Work

**Identity resolution.** Most prior work on character naming in video assumes additional supervised or weakly supervised data. A screenplay is used in [Everingham et al., 2006] to align faces to dialogs using mouth motion cues, from which we can deduce the identity of the speaker by aligning screenplay and closed captions. In [Ramanan et al., 2007], 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. In "Names and Faces" [Berg et al., 2004] the authors automatically cluster images aided by text coming from associated captions. In "Name-it" [Satoh et al., 1999], the authors use video and closed caption information to name people in televised news broadcasts via a simple co-occurrence score with extracted labels. Also related to our approach is the work of [Berg et al., 2005], in which a generative model of names in photograph captions and faces in images is proposed, with a language model that uses caption context to predict face-name assignments. Face-name correspondences, language model and face clustering are updated using Expectation Maximization.

## 8.3 Label constraints for person recognition

We explain here which cues can be exploited for naming characters with weak supervision. Our task is to predict the label $y \in \{1..L\}$ corresponding to the name of a character in a face track $x$, out of $L$ possible names. In order to simplify our evaluation, we assume the label set known using automatically extracted cast list from IMDB website. For example, the cast list of TV series Lost can be found at the following url: `http://www.imdb.com/title/tt0411008/fullcredits#cast`. Note, we only use the list of names and their associated gender, but *not* the images from this website. We build our formulation on a simple and general one-against-all multiclass scheme of the form:

$$y = \arg\max_a g^a(x), \tag{8.1}$$

where $g^a(x) = \mathbf{w}^a \cdot \mathbf{f}(x)$ is a linear function parametrized 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 8.6. *If we had* access to labeled examples $(x, y)$, with $y \in \{1..L\}$, one common approach would be to encourage $g^a(x) > 0$ for $a = y$ and $g^a(x) < 0$ for $a \neq y$ by minimizing the following convex upperbound on the corresponding 01-loss (where $\psi(\cdot)$ denotes some convex binary loss):

$$\mathcal{L}_{\text{supervised}}(g) = \sum_i \psi(g^{y_i}(x_i)) + \sum_i \sum_{a \neq y_i} \psi(-g^a(x_i)) \tag{8.2}$$

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 8.1. The constraints are of 3 kinds: grouping constraints, $1^{\text{st}}$ and $2^{\text{nd}}$ reference constraints, and label exclusion constraints from $3^{\text{rd}}$ person references and gender predictions.

## 8.3.1 Grouping constraints

Our grouping constraint states that face tracks grouped together should be labeled in the same way. One possible solution inspired by manifold regularization in semi-supervised learning would be to simply rely on appearance similarity to group faces together as a constraint for naming. However, our person classifier learns a decision boundary based on appearance cues, and so appearance based constraints are to some extent already captured. Instead, we use the **temporal grouping** framework described in chapter 7 to obtain grouping constraints, which uses additional local cues such as film structure, continuity editing and physical constraints to make grouping decisions. By introducing **independent grouping cues**, we are able to bias the weakly supervised classifier in a more effective way.

There are two possible ways to enforce grouping constraints of the form $y_i = y_j$ between face tracks $x_i, x_j$: 1) as a hard constraint, and 2) as a soft constraint. A **hard constraint** of this form can be expressed in a convex formulation by adding linear constraints to the objective, of the form: $\forall a, g^a(x_i) - g^a(x_j) = 0$, but the problem might become over constrained or even infeasible if there are too many such constraints. Alternatively, one can treat a cluster of face tracks as a single instance, for example by kernelizing the objective and using a set kernel [Wof and Shashua, 2003, Kondor and Jebara, 2003, Watkins, 1999]. Such approaches are feasible but make it hard to recover from errors in the clustering. As in [Yan et al., 2004], we choose the latter approach involving a **soft constraint** instead, the precise form of which is described in section 8.4. We introduce a soft constraint representing $y_i = y_j$ for all consecutive face tracks $x_i, x_j$ that are in a same cluster, as found by our previous temporal grouping. In the case where our temporal clustering mistakenly merges two different characters together from two different scenes (say, with corresponding true labels AAABBB), this will result in a *single* erroneous constraint ($y_3 = y_4$ in our example). In contrast, constraining all pairs of instances in a cluster would be over constraining and less robust to clustering errors (introducing 9 erroneous constraints out of 15 constraints in our example). Thus, the number of so-called must-link constraints that

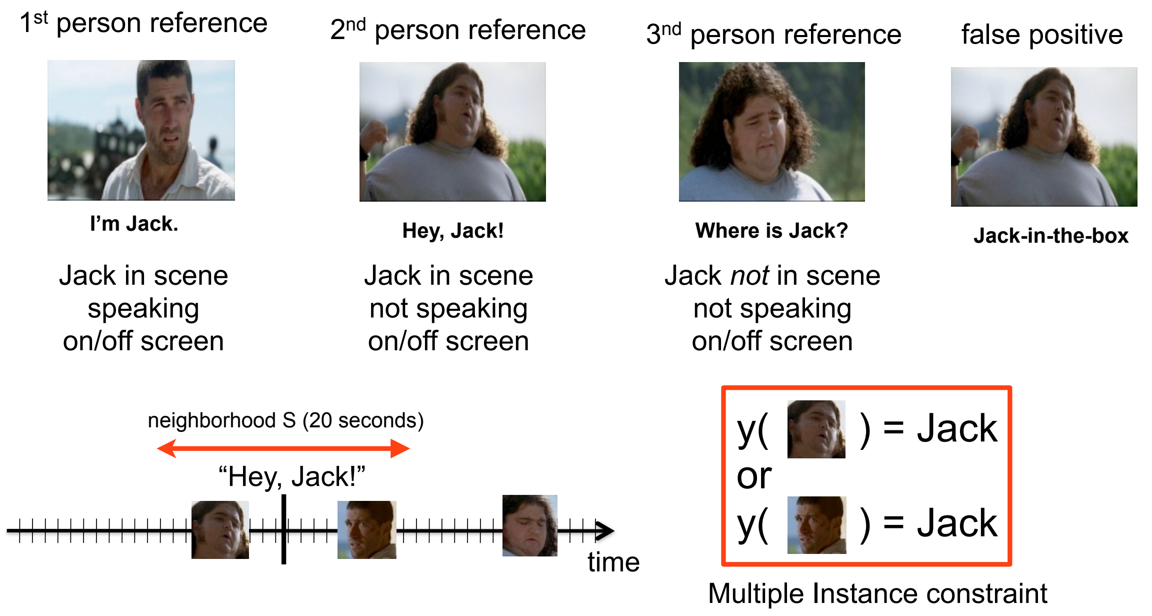

1st person reference
2nd person reference
3rd person reference
false positive
I'm Jack.
Hey, Jack!
Where is Jack?
Jack-in-the-box
Jack in scene speaking on/off screen
Jack in scene not speaking on/off screen
Jack not in scene not speaking on/off screen
neighborhood S (20 seconds)
"Hey, Jack!"
time
y( ) = Jack
or
y( ) = Jack
Multiple Instance constraint

Figure 8.3: First, second and third person references from dialogue (top) provide multiple instance type supervision on the desired labeling $y(\cdot)$ (bottom).

we use is $n - C$, where $n = \#$face tracks, $C = \#$ clusters which can be set using section 7.6.3. In our experiments we set $C = 0.3 \cdot n$ for each episode.

### 8.3.2 1st and 2nd reference constraints

When we observe the utterance "Hey Jack" (classified as 2nd reference, see section 8.6) at time $t$, we assume that some face track $x_i$ in the temporal neighborhood of $t$ will have label $y = Jack$[2]. 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 $\exists i \in S : y_i = a$ for a label $a$ and a set of face tracks $S$ associated to a reference. This is illustrated in figure 8.3.

[2]One could refine this rule using mouth motion cue[Everingham et al., 2006], at the cost of introducing additional errors

### 8.3.3 Exclusion constraints

This is a negative label constraint of the form: $y_i \neq a$, and comes from either (a) 3rd person reference or (b) gender. (a) When we observe an utterance classified as 3rd person reference (e.g. "Jack left") at time $t$, we assume that neighboring face tracks (at $t \pm 30 seconds$) *cannot* have the label $a$ corresponding to Jack. (b) When the average score of a face track $x_i$ according to our gender classifier exceeds a threshold $\theta_M$ (M for Male), we add label constraint $y_i \neq a$ for each label $a$ corresponding to a female (F) name, and a similar rule applies in the opposite case with some $\theta_F < \theta_M$[3].

### 8.3.4 Propagation of constraints through clustering

Multiple instance constraints described in the previous section are propagated throughout clusters. Thus, for any tuple of face tracks with a multiple instance constraint (for example, the constraint "$y_i = a \vee y_j = a$" for a pair $i, j$ and label $a$), we also add other tuples of face tracks from the corresponding clusters (in our example, $y_{i'} = a \vee y_{j'} = a$ with $(i, i')$ in one cluster and $(j, j')$ in another cluster), subsampling at most 200 out of all possible combinations. We also propagate in a similar way the exclusion constraints.

## 8.4 Convex formulation

We encode these label constraints (summarized in table 8.1) with a unified framework, based on combining convex losses on linear combinations of the scores $g^a(x_i)$. Our combined loss function is:

[3]In practice, we set $\theta_F$ and $\theta_M$ to achieve $90\%$ precision on a validation set

$$
\begin{aligned}
\mathcal{L}(g) &= \sum_{i,a} \mathcal{L}^{i,a}_{\text{excl.}}(g) + \sum_{i,j} \mathcal{L}^{ij}_{\text{link}}(g) + \sum_{S,a} \mathcal{L}^{S,a}_{\text{MI}}(g) \\
\mathcal{L}^{i,a}_{\text{excl.}}(g) &= \psi(-g^a(x_i)) && (8.3) \\
\mathcal{L}^{ij}_{\text{link}}(g) &= \sum_{a \in \{1..L\}} \bar{\psi}(g^a(x_i) - g^a(x_j)) && (8.4) \\
\mathcal{L}^{S,a}_{\text{MI}}(g) &= \psi(\frac{1}{|S|} \sum_{i \in S} g^a(x_i)), && (8.5)
\end{aligned}
$$

where each term expresses a constraint: (8.3) for $y_i \neq a$, (8.4) for $y_i = y_j$, and (8.5) for $\exists i \in S : y_i = a$. We use $\bar{\psi}(u) = \psi(u) + \psi(-u)$ for some convex binary loss function $\psi(\cdot) : \Re \mapsto \Re_+$. In our experiments, we use the square hinge loss $\psi(u) = \max(0, 1-u)^2$. Note, (8.3) and (8.4) have been used in a related formulation [Yan et al., 2004], but we believe (8.5) 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 with $L_2$ regularization on weight vectors $w = (w^a)_a \in \{1..L\}$. We solve it using the open-source library liblinear [Fan et al., 2008] (solving it with boosting would have also been possible). 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* dialog reference).

## 8.5 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 face, and 50 PCA components per part (two eyes, nose and mouth) using $15 \times 15$ patches around a

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

Table 8.1: Constraints on possible labels for our weakly supervised character naming based on dialog, gender and grouping cues. We show their occurrences ($\#$) per episode averaged over 16 episodes of Lost, and the resulting type of constraint. $a$ represents a label (e.g. Jack), $y_i$ the predicted label of a face track, $S$ a set of face tracks in the temporal neighborhood of the reference.

fixed location.

## 8.6 Character Naming Results

We run our grouping and naming system jointly on the 8 episodes of the TV show Lost on which we ran our clustering tests in chapter 7. 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 7.6.3. We extract references by matching words in the subtitle to the cast list, and determine reference type (1st, 2nd, 3rd) with a discriminative classifier described in section 8.6.4. The final decision for each face track $x_i$ in some cluster $S$ is determined using max-voting over the cluster ($\arg\max_a \sum_{j \in S} g^a(x_j)$).

### 8.6.1 Precision-recall evaluation

We measure performance of our system in a **refusal to predict** scheme inspired by [Everingham et al., 2006]. At a given confidence threshold, we measure how accurately the system names characters for examples whose confidence exceeds 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

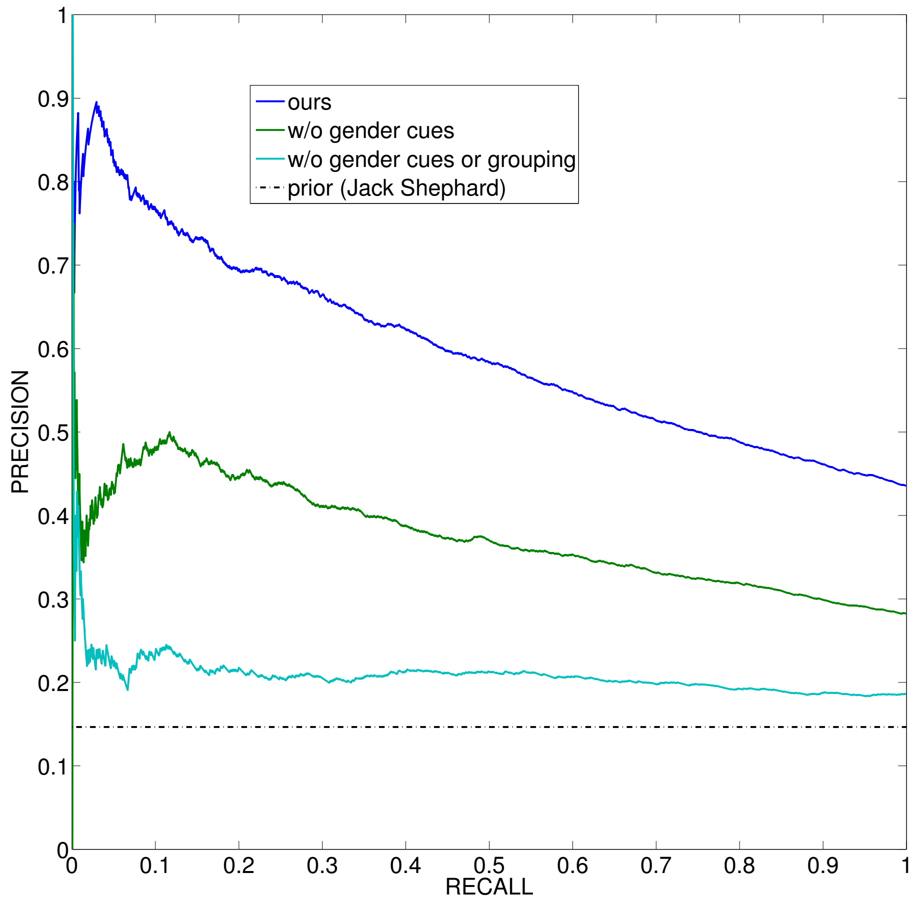


Figure 8.4: Naming results on 8 episodes of Lost. “prior” is the most common character. “ours” uses gender, grouping, and 1st, 2nd,3rd reference constraints propagated through grouping.

classifiers $(g^a)_{a\in\{1..L\}}$:

$$confidence(x_i) = g^{y_i^*}(x_i) - \max_{a\in\{1..L\}-\{y_i^*\}} g^a(x_i),$$

where $y^* = \arg\max_{a\in\{1..L\}} g^a(x)$. The precision and recall at a given confidence threshold $\theta$ are then defined as:

$$precision(\theta) = \frac{\sum_i \mathbb{1}(confidence(x_i) > \theta)\cdot \mathbb{1}(y_i^* = z_i)}{\sum_i \mathbb{1}(confidence(x_i) > \theta)} \tag{8.6}$$

$$recall(\theta) = \frac{\sum_i \mathbb{1}(confidence(x_i) > \theta)}{\sum_i 1}, \tag{8.7}$$

where $z$ denotes the true labels. By varying $\theta$ across all confidence levels, we obtain a precision-recall curve. Across the 8 testing episodes of Lost, we achieve an accuracy of $43.5\%$ for the 10 most frequent characters (each one voting among all of the $L = 55$ possible labels). The accuracy goes up to $76.7\%$ for the $10\%$ most confident scores. The precision-recall curve is shown in Figure 8.4.

### 8.6.2 Ablative analysis

Figure 8.4 shows the ablative analysis, with the exact same evaluation. We see that the cues provided by grouping, references, and gender improve significantly the performance of the system.

### 8.6.3 Analysis with combinations of perfect cues

We compare in figure 8.5 the quality of our naming system by replacing various components with their equivalent "perfect" counterparts—assuming perfect clustering ($TG^*$ in figure 7.7), perfect gender classification, perfect reference classification or perfect association of references to face tracks.

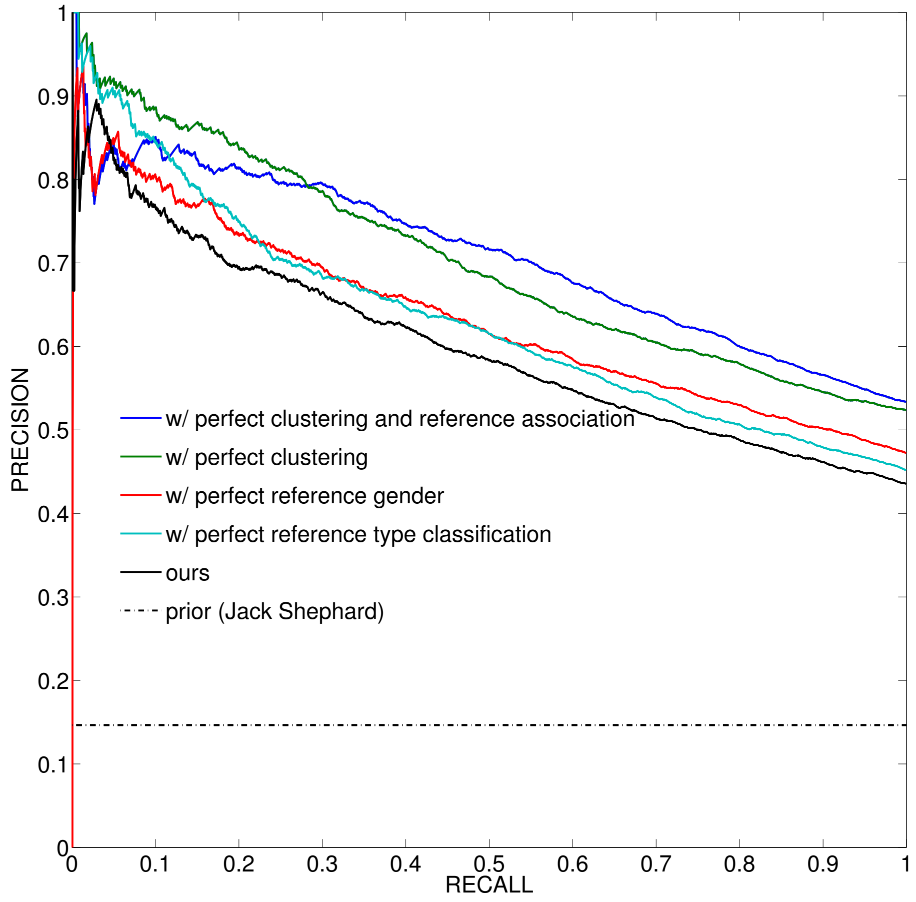


Figure 8.5: Naming results on 8 episodes of Lost, swapping in **perfect components**. “ours” is as before (see Figure 8.4). Each curve is obtained by replacing *only* the mentioned component of our system with it’s perfect counterpart: “perfect clustering” uses the optimal clustering possible using sequence partitioning with scope $k = 7$. “perfect gender” uses groundtruth gender information. “perfect reference type classification” uses the groundtruth type of reference ($1^{st}$, $2^{nd}$,$3^{rd}$) instead of our text classifier. “perfect clustering and reference association” directly applies each reference to its closest correct neighbor, and adds exclusion constraints for negative local neighbors. We see that perfect clustering and association of references adds the most improvement, and perfect gender and reference type classification add smaller but still significant improvements.

### 8.6.4 Dialog reference classification

Finally, we analyze an important component of the naming system, the ability to detect which type of reference occurs in order to constrain examples correctly. We train a discriminative classifier to determine whether a given reference (*i.e.* mention of a character name) corresponds to a 1$^{st}$, 2$^{nd}$ or 3$^{rd}$ person reference, or none of the above. We match 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 calculate the 1,000 most frequent unigrams/bigrams in the vicinity of 2 tokens from the reference, including punctuation. We use 6,000 features, each corresponding to an occurrence of a unigram or bigram in a particular position w.r.t. the reference. Our reference classifier achieves $79.3\%$ accuracy on a set of 300 hand-labeled examples.

## 8.7 Conclusion

Our end-to-end system is first to address the problem of character identification in TV video *without* the use of a screenplay, which opens the possibility of name-based retrieval in general video collections with minimal human intervention. We proposed a novel temporal grouping model to group face tracks locally with high precision and then learned to identify characters using weak supervision provided by cues from references in the dialog, gender, and temporal grouping. We presented a novel way to integrate these cues as multiple instance constraints in a convex formulation.

# Chapter 9

# Conclusion and Directions for Future Work

In this thesis we addressed weakly supervised problems that arise in computer vision, in particular movie structure parsing and character naming. We showed it is possible to learn under progressively less and less supervised settings, and explored the following three settings: (1) video, closed captions and screenplay, (2) video and closed captions, and (3) video only, where we learn to assign unique ids rather than names. We elaborated some theory on some of the intermediate cases, notably providing some answers to the following question: under which conditions can we learn from ambiguously labeled examples? We have identified cases where we can link the weak supervised objective to the fully supervised objective. Another interesting question to raise, is: for what other types of weak supervision can we get the same type of guarantees? Our last chapter has shown that we can essentially treat the ambiguous learning scenario and the multiple instance learning scenario with the same type of convex formulation, as illustrated in figure 9.1. As future work we would like to present a more unified theory of weakly labeled data to answer those questions.

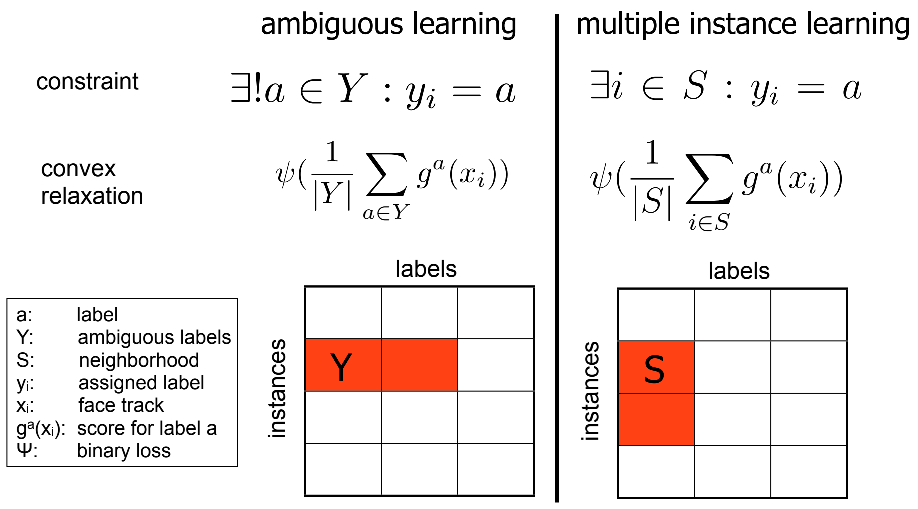

ambiguous learning
multiple instance learning
constraint
$\exists! a \in Y : y_i = a$
$\exists i \in S : y_i = a$
convex relaxation
$\psi(\frac{1}{|Y|}\sum_{a \in Y} g^a(x_i))$
$\psi(\frac{1}{|S|}\sum_{i \in S} g^a(x_i))$
labels
instances
Y
S
a: label
Y: ambiguous labels
S: neighborhood
$y_i$: assigned label
$x_i$: face track
$g^a(x_i)$: score for label a
Ψ: binary loss

Figure 9.1: Ambiguous learning scenario from chapters 4-6 and multiple instance learning scenario from chapter 8. There is a strong parallel between the two types of weak supervision on the desired labeling $y(\cdot)$.

## 9.1 Directions for future work

### 9.1.1 Closed captions and plot summary

We considered the cases of closed captions + screenplay, closed captions without screenplay. Another case of interest is to have closed captions together with some weak form of screenplay: given a one or two paragraph summary of a movie plot such as the ones we can find on Wikipedia or imdb, the goal is to find a precise alignment between plot elements and the video, including character naming and recognition of actions.

### 9.1.2 Character naming without closed captions

As future work, we would also like to consider the extreme case where only the video and audio are available. While much of the techniques presented in chapters 7 and 8 could in theory apply by replacing the closed captions with automatic speech recognition, in practice there are a number of additional audio cues that could be used for character association such as audio signature. This would allow one to process the larger mass of less structured video content such as Youtube videos.

### 9.1.3 Action recognition

We plan on significantly extending the action retrieval results to leverage the large amount of data we have collected. Relevancy can be improved by two sources of improvements: 1) using the ambiguous naming algorithm to resolve names more precisely (and filter actions by the corresponding actor performing the action), and 2) by learning models of action based with a technique similar to the ambiguous learning case: for example, a typical narration line such as “Jack turns away and runs” can give rise to two ambiguous action labels, “turn away” and “run”. The main difficulty is to segment the video into potential action snippets (the analog of faces).

# Appendix A

# Image Features

## A.1 Gender classifier

We collected an initial set of $1.5$ million images via Google Image search, searching for $1,000$ male and $1,000$ female names and downloading the top 800 images for each query. We used the gender of each name as noisy groundtruth labels for our gender classifier. For each image we ran a face detector based on Viola-Jones[Viola and Jones, 2004], removing images containing more than one face. Each image was registered using 4 facial features (using a part detector for two eyes, nose, mouth), and resized to a $60 \times 90$ grayscale image containing the face. We subsampled 200,000 images for training a two-level classifier: the first level is composed of a $8 \times 8$ grid of gentleboost classifiers based on decision stumps on Haar features restricted to regularly spaced overlapping grid locations. The second level uses the first level 64 classifiers as weak learners, and trains a linear model based on decision stumps. Accuracy on a hold-out set was $83\%$.

# Appendix B

# Text Features

## B.1 Automatic Screenplay Parsing

Our goal is to automatically parse an input screenplay into a computer-readable format consisting of narrations, scene transitions and dialogues (with identified speaker). The difficulty comes from the wide diversity of formatting rules when considering screenplay downloaded from the web. There are also inconsistencies within a screenplay, which is unavoidable since those are written by people. To further complicate matters, there is quite often a variable-length portion of text before (prologue), and after (epilogue) the actual screenplay we care about. Writing a specific parser for each type of screenplay is infeasible; instead we propose a two stage approach. In the first stage, we learn a bag of words model for the different constituents of screenplay: narrations, scene transitions, dialogues, and prologue/epilogue. The features we use are actual words, which we expect to generalize well across different screenplays (”I” is more frequent in a dialogue, whereas ”camera” occurs more frequently in narrations, and ”INT.” would be often found in scene transitions). Of course, we cannot expect such a generic classifer to perfectly classify each sentence, however the bulk of it is correct. In a second stage, we train another bag-of-words classifier which only uses formatting features (indentation, capitalization, *etc*let@tokeneonedot) which do *not* generalize across screenplays. The training labels come from the previous stage, which we expect to correct in this second stage. This approach works remarkably well, due to the fact that within a screenplay, a few formatting rules can be used to parse the entire text, and that we only need to be roughly correct in the first stage.

## B.2 Alignment of Screenplay and Closed Captions

The screenplay and the closed captions are readily available for a majority of movies and TV series produced in the US. A similar approach was used in [Everingham et al., 2006] to align faces with character names, with 2 differences: 1) they used the screenplay to reveal the speaker identity as opposed to scene transitions, and 2) subtitles were used instead of closed captions. Subtitles are encoded as bitmaps, thus require additional steps of OCR and spell-checking to convert them to text[Everingham et al., 2006], whereas closed captions are encoded as ASCII text in DVDs, making our approach simpler and more reliable, requiring a simple modification of mplayer (`http://www.mplayerhq.hu/`).

## B.3 Pronoun Resolution

Algorithm we used for pronoun resolution (adapted from Hobbs algorithm for our needs):

1. identify names of speaking characters
2. provide gender label for speaking characters (done by hand but could be automated)
3. process entities: for first sentence,
    (a) provide value for SPEAKER variable (name of a person, 'director' or null for stage guidelines)
    (b) create an ordered list of ENTITIES (nouns and pronouns, read off from preprocessed input, order of appearance from left to right).

   (c) create an ordered list of ANAPHORS (identify pronouns in ENTITIES by string match -he, she, etc- start with last element in ENTITIES and process backwards to first)

4. find antecedents for anaphors

   (a) for first element in ANAPHORS, check elements in ENTITIES until you find a gender-compatible antecedent (none found when there is no match)

   (b) output updated list of ENTITIES

   (c) proceed to next sentence and repeat a-c above except 'if none found', check ENTITIES of previous sentence.

# Appendix C

# Miscellanous proofs

## C.1 Linear time feature selection during boosting

We describe below how efficiently to adapt a multiclass variant of gentleboost[Friedman et al., 2000, Torralba et al., 2006] to the case considered in section 7.7.3. At each round of boosting, we update the linear classifier with a weak learner $h(\mathbf{x}, y)$, chosen so as to minimize a second order Taylor approximation of $\mathcal{L}(g)$. For the the exponential loss this is equivalent to minimizing (over $h$):

$$\sum_i \sum_{y \in \mathcal{Y}} z^i(y)(h(x^i, y^i) - h(x^i, y) - 1)^2, \tag{C.1}$$

where we introduced the weights $z^i(y) \propto \ell(y^i, y)\psi\left(g^{y^i}(\mathbf{x}^i) - g^y(\mathbf{x}^i)\right)$, normalized to sum to 1 over $(i, y)$. Notice, a salient contrast with the objective in [Torralba et al., 2006] (equation 3) is the coupling introduced by the difference term $h(x^i, y^i) - h(x^i, y)$, potentially complicating the optimization. We show below that in fact, we can still efficiently compute the optimal weak learner, parameterized as a decision stump, and furthermore this can be done in **linear** time with a single pass over the training examples. We use decision stumps as weak learners, of the form $h(\mathbf{x}, y) = a \cdot \mathbb{1}(f_j(\mathbf{x}, y) \leq \theta) + b \cdot \mathbb{1}(f_j(\mathbf{x}, y) > \theta)$ and seek the optimal stump across feature dimension $j$, threshold $\theta$, and parameters $a, b$. Notice we can set $b = 0$ WLOG since adding a constant term to $h(\mathbf{x}, y)$ doesn't change the objective. We further simplify the notation, rewriting (C.1) as an expectation w.r.tlet@tokeneonedotthe weights $\mathbf{z}$:

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

with $u = (i, y), z_u = z^i(y), \bar{F}_u^\theta = \mathbb{1}(h(x^i, y^i) \leq \theta), F_u^\theta = \mathbb{1}(h(x^i, y) \leq \theta)$. The optimal $a$ is given by $a = 1/E[\bar{F}^\theta - F^\theta]$ and the corresponding objective is:

$$1 - \frac{E^2[\bar{F}^\theta - F^\theta]}{E[\bar{F}^\theta - F^\theta]^2} \tag{C.3}$$

The ratio can be computed without evaluating the expectations across all values of $\theta$ using the fact that $\bar{F}^\theta, F^\theta$ are binary:

$$E[\bar{F}^\theta - F^\theta]^2 = E[\bar{F}^\theta]^2 + E[\bar{F}^\theta]^2 - 2E[\bar{F}^\theta F^\theta] \tag{C.4}$$

$$= E[\bar{F}^\theta] + E[\bar{F}^\theta] - 2E[\bar{\bar{F}}^\theta], \tag{C.5}$$

where $\bar{\bar{F}}_u^\theta = \mathbb{1}(\max(h(x^i, y^i), h(x^i, y)) \leq \theta)$. The optimal $\theta$ (discretized into a set of regularly spaced discrete values $\theta_1 ... \theta_T$) can now be computed via a single pass over the data of size $N = n \cdot k$ for a given feature dimension $j$ using the following algorithm:

**Algorithm 3** Contrastive Gentleboost

```
1: for $u$ = 1 to $N$ do
2:     compute the bins $\theta, \bar{\theta}, \bar{\bar{\theta}}$ in which fall $h(x^i, y), h(x^i, y^i), \max(h(x^i, y^i), h(x^i, y))$
3:     update the corresponding counts: $\Sigma[F]_\theta += z_u$ likewise for $\Sigma[\bar{F}], \Sigma[\bar{\bar{F}}]$
4: end for
5: replace $\Sigma[F], \Sigma[\bar{F}], \Sigma[\bar{\bar{F}}]$ by their cumulative sums
6: for $\theta$ = $\theta_1$ to $\theta_T$ do
7:     compute the objective value $val(\theta)$ using (C.3) and (C.5), using $\Sigma[\cdot]$ in place of
       expectations.
8: end for
```

The final running time to select the optimal stump across all feature dimensions is $d \cdot N = d \cdot n \cdot B_k$.

# Bibliography

[Allwein et al., 2000] Allwein, E. L., Schapire, R. E., Singer, Y., and Kaelbling, P. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141. 21

[Andrews et al., 2003] Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, pages 577–584. 26

[Arora et al., 1993] Arora, S., Babai, L., Stern, J., and Sweedyk, Z. (1993). The hardness of approximate optimia in lattices, codes, and systems of linear equations. In *IEEE Symposium on Foundations of Computer Science*, pages 724–733. 15

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). UCI machine learning repository. 74, 80

[Balas and Simonetti, 2001] Balas, E. and Simonetti, N. (2001). Linear time dynamic programming algorithms for new classes of restricted tsps: A computational study. *INFORMS Journal on Computing*, 13:56–75. 39, 40

[Barlow, 1989] Barlow, H. (1989). Unsupervised learning. *Neural Computation*, 1(3):295–311. 28

[Barnard et al., 2003] Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D., and Jordan, M. (2003). Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135. 54

[Bartlett et al., 2006] Bartlett, P. L., Jordan, M. I., and Mcauliffe, J. D. (2006). Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156. 16

[Bartlett and Mendelson, 2002] Bartlett, P. L. and Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482. 64

[Belkin et al., 2006] Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434. 24

[Berg et al., 2005] Berg, T., Berg, A., Edwards, J., and Forsyth, D. (2005). Whos in the Picture? In *Advances in Neural Information Processing Systems*, page 137. MIT Press. 142

[Berg et al., 2004] Berg, T., Berg, A., J.Edwards, M.Maire, R.White, Teh, Y., Learned-Miller, E., and Forsyth, D. (2004). Names and faces in the news. In *CVPR*, pages 848–854. 55, 142

[Bie and Cristianini, 2004] Bie, T. D. and Cristianini, N. (2004). Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*, pages 73–80. MIT Press. 24

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press. 15

[Blum and Chawla, 2001] Blum, A. and Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *In International Conference on Machine Learning (ICML*. 24

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM New York, NY, USA. 23

[Boutell et al., 2004] Boutell, M., Luo, J., Shen, X., and Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771. 27

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239. 24

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140. 20

[Brown et al., 1993] Brown, P. E., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311. 75

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75. 14

[Chapelle et al., 2006] Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-supervised learning*. MIT press. 14

[Chapelle et al., 2007] Chapelle, O., Tubingen, G., Sindhwani, V., and Keerthi, S. (2007). Branch and bound for semi-supervised support vector machines. In *Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference*, page 217. MIT Press. 24

[Cour et al., 2005a] Cour, T., Benezit, F., and Shi, J. (2005a). Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 2. 11, 28

[Cour et al., 2005b] Cour, T., Gogin, N., and Shi, J. (2005b). Learning spectral graph segmentation. In *Workshop on Artificial Intelligence and Statistics (AISTATS)*. 11, 29

[Cour et al., 2008] Cour, T., Jordan, C., Miltsakaki, E., and Taskar, B. (2008). Movie/script: Alignment and parsing of video and text transcription. In *Proceedings of 10th European Conference on Computer Vision, Marseille, France*. 11

[Cour et al., 2009a] Cour, T., Sapp, B., Jordan, C., and Taskar, B. (2009a). Learning from ambiguously labeled images. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 11, 140

[Cour et al., 2009b] Cour, T., Sapp, B., Nagle, A., and Taskar, B. (2009b). Talking pictures: Temporal grouping and dialog-supervised person recognition in video. In *Submitted to ICCV 2009*. 11

[Cour and Shi, 2007a] Cour, T. and Shi, J. (2007a). Recognizing objects by piecing together the segmentation puzzle. In *Proceedings of the 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 11, 28

[Cour and Shi, 2007b] Cour, T. and Shi, J. (2007b). Solving markov random fields with spectral relaxation. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 11. 11

[Cour et al., 2007] Cour, T., Srinivasan, P., and Shi, J. (2007). Balanced graph matching. In *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA. 11

[Cover and Hart, 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27. 77

[Cowans and Szummer, 2005] Cowans, P. J. and Szummer, M. (2005). A graphical model for simultaneous partitioning and labeling. In *Proc. AI & Statistics*. 120

[Crammer et al., 2001] Crammer, K., Singer, Y., Cristianini, N., Shawe-taylor, J., and Williamson, B. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:2001. 19

[Crandall and Huttenlocher, 2006] Crandall, D. and Huttenlocher, D. (2006). Weakly supervised learning of part-based spatial models for visual object recognition. *Lecture Notes in Computer Science*, 3951:16. 26

[De Bie et al., 2004] De Bie, T., Suykens, J., and De Moor, B. (2004). Learning from general label constraints. *Lecture notes in computer science*, pages 671–679. 24

[Della Pietra et al., 1997] Della Pietra, S., Della Pietra, V., Lafferty, J., Technol, R., and Brook, S. (1997). Inducing features of random fields. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):380–393. 76

[Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A probabilistic theory of pattern recognition*. springer. 16

[Dietterich et al., 1997] Dietterich, T., Lathrop, R., and Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71. 25

[Duygulu et al., 2002] Duygulu, P., Barnard, K., de Freitas, J., and Forsyth, D. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, pages 97–112. 54

[Everingham et al., 2006] Everingham, M., Sivic, J., and Zisserman, A. (2006). Hello! My name is... Buffy – automatic naming of characters in tv video. In *BMVC*. 2, 22, 33, 42, 46, 55, 95, 100, 103, 104, 130, 140, 142, 145, 148, 158

[Fan et al., 2008] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874. 67, 147

[Ferencz et al., 2006] Ferencz, A., Learned-Miller, E., and Malik, J. (2006). Learning to locate informative features for visual identification. *International Journal of Computer Vision*. 120

[Fergus et al., 2007] Fergus, R., Perona, P., and Zisserman, A. (2007). Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71(3):273–303. 26

[Fitzgibbon and Zisserman, 2002] Fitzgibbon, A. and Zisserman, A. (2002). On affine invariant clustering and automatic cast listing in movies. *Lecture Notes In Computer Science*, pages 304–320. 119

[Freund, 2001] Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine learning*, 43(3):293–318. 21

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. (1999). A short introduction to boosting. *Japonese Society for Artificial Intelligence*, 14(5):771–780. 20

[Friedman et al., 2000] Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Ann. Statist.*, 28:337–407. 20, 160

[Fung et al., 2007] Fung, G., Dundar, M., Krishnapuram, B., and Rao, R. (2007). Multiple instance learning for computer aided diagnosis. In *Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference*, page 425. MIT Press. 26

[Gallagher and Chen, 2007] Gallagher, A. and Chen, T. (2007). Using group prior to identify people in consumer images. In *CVPR Workshop on Semantic Learning Applications in Multimedia*. 55

[Gehler and Chapelle, 2007] Gehler, P. and Chapelle, O. (2007). Deterministic Annealing for Multiple-Instance Learning. *AISTATS07*. 26

[Ghahramani, 2004] Ghahramani, Z. (2004). Unsupervised learning. *Lecture Notes in Computer Science*, 3176:72–112. 14

[Godbole and Sarawagi, 2004] Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. *Lecture Notes in Computer Science*, pages 22–30. 27

[Goldberg et al., 2007] Goldberg, A., Zhu, X., and Wright, S. (2007). Dissimilarity in graph-based semi-supervised classification. *Artificial Intelligence and Statistics (AISTATS)*. 25

[Goncalves and Quaresma, 2003] Goncalves, T. and Quaresma, P. (2003). A preliminary approach to the multilabel classification problem of Portuguese juridical documents. *Lecture notes in computer science*, pages 435–444. 27

[Grady and Funka-Lea, 2004] Grady, L. and Funka-Lea, G. (2004). Multi-label image segmentation for medical applications based on graph-theoretic electrical potentials. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 230–245. 24

[Hastie et al., 2001] Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R. (2001). *The elements of statistical learning*. Springer New York. 16

[Huang et al., 2007a] Huang, G., Jain, V., and Learned-Miller, E. (2007a). Unsupervised joint alignment of complex images. In *International Conference on Computer Vision*, pages 1–8. 30

[Huang et al., 2007b] Huang, G., Jain, V., and Learned-Miller, E. (2007b). Unsupervised joint alignment of complex images. In *IEEE International Conference on Computer Vision*. 80

[Huang et al., 2007c] Huang, G., Ramesh, M., Berg, T., and Learned-Miller, E. (2007c). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst. 74, 80

[Hullermeier and Beringer, 2006] Hullermeier, E. and Beringer, J. (2006). Learning from ambiguously labeled examples. *Intell. Data Analysis*, 10(5):419–439. 52, 77, 78

[Jin and Ghahramani, 2002] Jin, R. and Ghahramani, Z. (2002). Learning with multiple labels. In *NIPS*, pages 897–904. 53, 76

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Sixteenth International Conference on Machine Learning*. 23

[Kender and Yeo, 1998] Kender, J. and Yeo, B. (1998). Video scene segmentation via continuous video coherence. In *IEEE Conference on Computer Vision and Pattern Recognition*. 36, 45

[Kohavi, 1995] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143. Morgan Kaufmann. 15

[Kondor and Jebara, 2003] Kondor, R. and Jebara, T. (2003). A kernel between sets of vectors. In *International Conference on Machine Learning*. 144

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*. 123

[Laptev et al., 2008] Laptev, I., Marszałek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*. 30

[Lewis, 2000] Lewis, R. (2000). An introduction to classification and regression tree (CART) analysis. In *Annual Meeting of the Society for Academy Emergency Medicine. San Francisco, California. USA*. 21

[Li et al., 2003] Li, S., Zhang, Z., Shum, H., and Zhang, H. (2003). FloatBoost learning for classification. *Advances in Neural Information Processing Systems*, pages 1017–1024. 20

[Li and Ogihara, 2003] Li, T. and Ogihara, M. (2003). Detecting emotion in music. In *Proceedings of the Fifth International Symposium on Music Information Retrieval*, pages 239–240. 27

[Lienhart, 2001] Lienhart, R. (2001). Reliable transition detection in videos: A survey and practitioner's guide. *Int. Journal of Image and Graphics*. 34

[Lin et al., 2008] Lin, C.-J., Weng, R. C., and Keerthi, S. S. (2008). Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9:627–650. 67

[Maron, 1998] Maron, O. (1998). *Learning from Ambiguity*. PhD thesis, Massachusetts Institute of Technology. 14

[Maron and Lozano-Pérez, 1998] Maron, O. and Lozano-Pérez, T. (1998). A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576. 26

[Maruyama et al., 2002] Maruyama, O., Shoudai, T., and Miyano, S. (2002). Toward drawing an atlas of hypothesis classes: Approximating a hypothesis via another hypothesis model. In *DS '02: Proceedings of the 5th International Conference on Discovery Science*, pages 220–232, London, UK. Springer-Verlag. 15

[Mermelstein, 1976] Mermelstein, P. (1976). Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence, RCH Chen, ed., Academic Press: New York*, pages 374–388. 82

[Moreno et al., 1998] Moreno, P., Joerg, C., Thong, J., and Glickman, O. (1998). A recursive algorithm for the forced alignment of very long audio segments. In *Fifth International Conference on Spoken Language Processing*. ISCA. 82

[MOSEK ApS, ] MOSEK ApS, D. The mosek optimization tools manual version 5.0 (revision 60). 67

[Myers and Rabiner, 1981] Myers, C. S. and Rabiner, L. R. (1981). A comparative study of several dynamic time-warping algorithms for connected word recognition. In *The Bell System Technical Journal*. 42

[Ngo et al., 2001] Ngo, C.-W., Pong, T.-C., and Zhang, H.-J. (2001). Recent advances in content-based video analysis. *International Journal of Image and Graphics*, 1(3):445–468. 34, 36

[Nowak and Jurie, 2007] Nowak, E. and Jurie, F. (2007). Learning visual similarity measures for comparing never seen objects. In *CVPR*. 120

[Proakis and Manolakis, 1996] Proakis, J. and Manolakis, D. (1996). *Digital signal processing: principles, algorithms, and applications*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA. 82

[Qi et al., 2007] Qi, G., Hua, X., Rui, Y., Tang, J., Mei, T., and Zhang, H. (2007). Correlative multi-label video annotation. In *Proceedings of the 15th international conference on Multimedia*, pages 17–26. ACM New York, NY, USA. 27

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106. 15

[Ramanan et al., 2007] Ramanan, D., Baker, S., and Kakade, S. (2007). Leveraging archival video for building face datasets. In *International Conference on Computer Vision*. 30, 55, 118, 120, 131, 134, 135, 142

[Satoh et al., 1999] Satoh, S., Nakamura, Y., and Kanade, T. (1999). Name-it: Naming and detecting faces in news videos. *IEEE MultiMedia*, 6(1):22–35. 55, 142

[Schapire, 1997] Schapire, R. (1997). Using output codes to boost multiclass learning problems. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 313–321. MORGAN KAUFMANN PUBLISHERS, INC. 27

[Schapire and Freund, 1997] Schapire, R. and Freund, Y. (1997). A decision theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci*, 55:119–139. 21, 27

[Schapire and Singer, 1999] Schapire, R. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336. 20, 21, 27

[Schroeder, 2004] Schroeder, M. (2004). *Computer speech: recognition, compression, synthesis*. Springer. 82

[Sivic et al., 2005] Sivic, J., Everingham, M., and Zisserman, A. (2005). Person spotting: video shot retrieval for face sets. In *International Conference on Image and Video Retrieval (CIVR 2005), Singapore*. 33, 118

[Sjlander, 2003] Sjlander, K. (2003). An hmm-based system for automatic segmentation and alignment of speech. In *In Proceedings of Fonetik 2003*, pages 93–96. 82

[Smith, 2005] Smith, T. J. (2005). *An Attentional Theory of Continuity Editing*. PhD thesis, University of Edinburgh. 119

[Steinwart, 2007] Steinwart, I. (2007). How to compare different loss functions and their risks. *Constructive Approximation*, 26(2):225–287. 16

[Talkin, 1995] Talkin, D. (1995). A robust algorithm for pitch tracking (RAPT). *Speech coding and synthesis*, pages 495–518. 82

[Torralba et al., 2006] Torralba, A., Murphy, K., and Freeman, W. (2006). Shared features for multiclass object detection. *Lecture Notes in Computer Science*, 4170:345. 160

[Turk and Pentland, 1991] Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91.*, pages 586–591. 80

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. 18, 23

[Viola and Jones, 2004] Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154. 21, 46, 156

[Viola et al., 2006] Viola, P., Platt, J., and Zhang, C. (2006). Multiple instance boosting for object detection. *Advances in neural information processing systems*, 18:1417. 26

[Wang et al., 2008] Wang, J., Zhao, Y., Wu, X., and Hua, X. (2008). Transductive multi-label learning for video concept detection. 27

[Wang and Zucker, 2000] Wang, J. and Zucker, J. (2000). Solving the multiple-instance problem: A lazy learning approach. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 1119–1126. 26

[Watkins, 1999] Watkins, C. (1999). Dynamic alignment kernels. *Advances in large margin classifiers*. 144

[Weston, 1998] Weston, J. (1998). Multi-class support vector machines. Technical report. 19

[Wof and Shashua, 2003] Wof, L. and Shashua, A. (2003). Kernel principal angles for classification machines with applications to image sequence interpretation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1. 144

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural networks*, 5(2):241–259. 27

[Yan et al., 2004] Yan, R., Zhang, J., Yang, J., and Hauptmann, A. (2004). A discriminative learning framework with pairwise constraints for video object classification. In *CVPR*, pages 284–291. 25, 144, 147

[Yeung et al., 1998] Yeung, M., Yeo, B.-L., and Liu, B. (1998). Segmentation of video by clustering and graph analysis. *Comp. Vision Image Understanding*. 36

[Zhai and Shah, 2006] Zhai, Y. and Shah, M. (2006). Video scene segmentation using markov chain monte carlo. *IEEE Transactions on Multimedia*, 8(4):686–697. 36

[Zhang, 2004] Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *J. Mach. Learn. Res.*, 5:1225–1251. 57, 61

[Zhou and Zhang, 2007] Zhou, Z. and Zhang, M. (2007). Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems*, 19:1609. 14

[Zhu et al., ] Zhu, J., Rosset, S., Zou, H., and Hastie, T. Multi-class adaboost. *Ann Arbor*, 1001:48109. 21, 27

[Zhu, ] Zhu, X. Semi-supervised learning literature survey. *world*, 10:10. 14

[Zhu et al., 2003] Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 20, page 912. 24