Learning to annotate dynamic video scenes

Piotr Bojanowski

THÈSE DE DOCTORAT
de l'Université de recherche
Paris Sciences Lettres –
PSL Research University

préparée à
l'École normale supérieure

Annotation automatique de scenes vidéo

Learning to annotate dynamic video scenes

par Piotr Bojanowski

École doctorale n°386
Spécialité: Informatique
Soutenue le 22.06.2016

Composition du Jury :

M Andrew Zisserman
University of Oxford
Rapporteur

M Greg Mori
Simon Fraser University
Rapporteur

M Jean Ponce
École normale supérieure
Directeur de thèse

Mme Cordelia Schmid
Inria
Directeur de thèse

M Ivan Laptev
Inria
Directeur de thèse

M Josef Sivic
Inria
Directeur de thèse

M Francis Bach
Inria
Membre du Jury

M Florent Perronnin
Facebook AI Research
Membre du Jury

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

The most exciting phrase to hear in science, the one that heralds new discoveries, is not “Eureka!”, but “That’s funny ...”

Isaac Asimov

Résumé

Les vidéos représentent des scènes complexes, comprenant des humains et des objets, illustrant les interactions entre ces derniers et leur environnement. Les relations entre agents sont susceptibles d'évoluer dans le temps et les agents peuvent effectuer des "actions". La compréhension automatique des vidéos nécessite de correctement localiser les agents à la fois dans l'espace et dans le temps. De plus, il faut décrire les relations entre ces agents et leur évolution temporelle.

La vision par ordinateur repose souvent sur l'apprentissage supervisé, où des échantillons étiquetés permettent d'apprendre les paramètres d'un modèle. Cependant, pour des données aussi riches que la vidéo, l'étiquetage est coûteux et compliqué. Les étiquettes symboliques ne sont pas suffisamment riches pour encoder les interactions entre personnes, objets et scènes. Le langage naturel offre une puissance descriptive qui en fait un modalité pratique pour annoter des données vidéo. Nous proposons de mettre l'accent sur la modélisation conjointe de vidéo et de texte. Nous explorons des modèles joints dans le contexte de films avec leurs scripts de tournage. Le principal défi auquel nous sommes confrontés est que les scripts de films ne fournissent pas de localisation spatiale et temporelle des objets et des actions.

Nous présentons d'abord un modèle permettant d'associer automatiquement des étiquettes de personne et d'action aux détections de personnes dans les films. Le modèle utilise une fonction de coût de clustering discriminatif, et une supervision faible sous la forme de contraintes que nous obtenons à partir de scripts. Cette approche nous permet de localiser spatialement et temporelement les agents et les actions qu'ils effectuent dans la vidéo, tel que décrit dans le script. Cependant, la localisation temporelle et spatiale est principalement due à l' utilisation de détections de personnes. Nous décrivons ensuite un modèle permettant d'aligner des phrases avec les images de la vidéo. La correspondance temporelle est obtenue en utilisant un modèle discriminatif sous contraintes d'ordre temporel. Ce modèle d'alignement est appliqué à deux ensembles de données : un composé de vidéos associées à un flux d'étiquettes ; un autre composé de vidéos et descriptions sous la forme d'étapes).

Abstract

Videos often depict complex scenes including people, objects and interactions between these and the environment. Relations between agents are likely to evolve in time and agents can perform actions. The automatic understanding of video data is complicated as it requires to properly localize the agents both in space and time. Moreover, one need to automatically describe the relations between agents and how these evolve in time.

Modern approaches to computer vision heavily rely on supervised learning, where annotated samples are provided to the algorithm to learn parametric models. However, for rich data such as video, the labelling process starts to be costly and complicated. Also, symbolic labels are not sufficient to encode the complex interactions between people, objects and scenes. Natural language offers much richer descriptive power and is thus a practical modality to annotated video data. Therefore, in this thesis we propose to focus on jointly modelling video and text. We explore such joint models in the context of movies with associated movie scripts, which provide accurate descriptions of the pictured events. The main challenge that we face is that movie scripts do not provide precise temporal and spatial localization of objects and actions.

We first present a model for automatically annotating person tracks in movies with person and action labels. The model uses a discriminative clustering cost function, and weak supervision in the form of constraints that we obtain from scripts. This approach allows us to spatially and temporally localize agents and the actions they perform, as described in the script, in the video. However, the temporal and spatial localization is due to the use of person detection tracks. Then, in a second contribution, we describe a model for aligning sentences with frames of the video. The optimal temporal correspondence is again obtained using a discriminative model under temporal ordering constraints. This alignment model is applied on two datasets: one composed of videos associated with a stream of symbolic labels; a second one composed of videos with textual descriptions in the form of key steps towards a goal (cooking recipes for instance).

Acknowledgments

I would like to thank my supervisors Jean Ponce, Cordelia Schmid, Ivan Laptev and Josef Sivic. They have been of greatest help, and have shown a lot of patience and understanding during these four years. A special thank goes to Francis Bach who was a great mentor and guide despite the fact that he wasn't a direct supervisor.

I would also like to express my gratitude to the people I collaborated with: Rémi Lajugie, Guillaume Seguin, Edouard Grave, Jean-Baptiste Alayrac, Sylvain Arlot and Phillippe Cuvillier. The countless discussions and tinkering have been at the core of our research activities and I hardly imagine completing this work without them.

I want to thank all the people at WILLOW and SIERRA for creating an amazing work environment. An exceptional thank goes to my officemates Senanayak Sesh Kumar Karri, Aymeric Dieuleveut and Augustin Lefebvre. The relaxed yet serious environment and casual discussions we had contributed a lot to this thesis.

Obvious credits got to my close friends and family, for their great informal help and understanding. Eventually, I would like to thank Rachel Fonck for the amazing amount of support and comprehension during these studious years.

This work was supported in part by the ERC grant VideoWorld.

Contents

Chapter 1

Introduction

1.1 Goal

The amount of visual digital data grows rapidly year by year. Much of this image and video data is generated by users and shared online. For example, Facebook users upload more than 350 million pictures every day and these numbers are very likely to grow. Facing such big amounts of data calls for algorithms capable of understanding and automatically sorting these huge data collections. There are many reasons for that, including the ability to make these collections searchable, abusive content detection, or efficient indexing. Nowadays things start to be even more complex as people also start sharing very large amounts of video data. Compared to images, video requires additional methods and resources for understanding dynamic scenes and events.

Automatic video understanding is a complicated problem that leads to many interesting applications. Videos typically depict complex relationships between people and their environment that evolve in time. Understanding what is going on in a video implies recognizing the context (scenes), objects, people, how they interact and keeping track of all this information along the video. However, designing good models for these concepts requires some kind of common knowledge or annotated data. Indeed, modern approaches to visual recognition are based on supervised machine learning techniques, which in most cases require precisely annotated data. For instance to rec-

Figure 1-1 – Illustration of readily available video data with associated textual descriptions. These include from left to right: feature movies and associated scripts, YouTube videos and their descriptions, and sport video commentaries.

ognize cats, these models are trained by presenting the algorithm with a set of images that picture a cat and other ones that do not. As we will discuss it later, completely annotating a video is very hard and it is not clear what kind of information should be annotated.

However, in many cases, video data comes with textual descriptions that can be meta-data, subtitles or scripts. Some examples of such paired video-text data are illustrated in Fig. 1-1. This data is highly correlated with the visual content of the video, event though there is no precise alignment between the two. For instance, if the subtitles mention a character named "James", it is the viewer who infers to which face it corresponds to. The viewer usually needs some time to actually recognize who is who, based on correlations between the dialogues and images.

The goal of this work is to develop models that enable us to work with videos that are associated with such textual data. We would like to build algorithms that use the two modalities and connect them in some ways. As most modern computer vision advances are based on supervised machine learning techniques, the simplest thing that one can imagine, is using textual data as a form of supervision. The first problem with this kind of approach is that such raw text cannot be directly used as annotations. It is often very noisy and imprecise, only partially describes the scene, and there is a high variability in the expressions. Moreover, this line of work leads to an asymmetric approach which can be criticized. Indeed, both text and video understanding can benefit from one each other and one of these modalities should not be used as a supervision for the other one.

Also, as mentioned earlier, textual descriptions that are associated with videos are usually only coarsely aligned. The exact correspondence in time and space is not given *a priori* and we would like to build models capable of recovering this alignment. We want our method to be able to exploit some textual and visual representation and align the two modalities based on data correlations. To this end we need to design a system that can discover these correlations while dealing with the uncertainty in the alignment.

This idea of aligning textual and visual information will be at the core of this work. Similar problems arise when working with images with captions, since the correspondence between nouns in the caption and objects in the image is unknown. In video, however, the alignment must be found in space but also in time, depending on how precisely the textual description is located in time. This spatio-temporal alignment problem is illustrated in Fig. 1-2, for a scene description found in the movie Fargo. The script mentions character names and describes what they are doing but we don't know how they look, and what these actions look like. If one knows that Marge gropes in the dark while Norm keeps on sleeping, a single glance at the frame allows us to identify who is Norm and who is Marge. However, we have a precise idea of what a sleeping person looks like while this is not so obvious for an algorithm.

The better a textual source is aligned, the simpler the association task would be. For instance, movie subtitles are precisely aligned in order to match the speaker's lip movements, which makes character identification simple. On the other hand, a YouTube video description is much more imprecise, and potentially describes a long piece of video (several minutes). Moreover, the alignment can be carried out at different scales, either on the level of words or entire sentences, parts of images or whole video frames. In the work described in this thesis, we explore spatial and temporal alignments, both at the level of words and sentences.

The bedroom is dark. Norm is snoring. The phone rings. Marge gropes in the dark. Marge props herself up next to the still-sleeping Norm.

Figure 1-2 – Illustration of a video frame along with the corresponding scene description found in the movie script. The movie script precisely describes what is happening in the video but no correspondence to regions in the image are given. The goal of joint video-text models would be to answer the following questions: Who is Norm? Who is Marge? What does a phone look like? How does one grope something? What does it look like to prop oneself up?

1.2 Motivation

The line of work presented in this thesis is motivated by several observations. The first one is the fact that the kind of data we want to exploit is often freely available and its quantity will keep on growing. Movies are often provided with subtitles and shooting scripts. More and more television platforms propose movies with Descriptive Video Service (DVS) descriptions for the visually impaired. These descriptions provide a precise audio description of what appears on the screen. Even though they are in the form of speech signal, these can be automatically transcribed into text. Sport events are usually provided with a commentary describing the action happening on the field. It precisely relates players actions and gives a higher level interpretation of the players behavior, for example whether or not specific actions led to the victory. Recently, it has become quite popular to post on-line "how-to" videos, providing step by step guides for a given task. These are usually associated with text or voiceover describing each step (as a description of the video).

Figure 1-3 – Examples of annotated videos found in the Hollywood 2 action recognition dataset [Marszalek et al., 2009]. Every video is associated with a label, telling what action is performed in the given clip. We show examples for six out of the twelve manually defined classes in this dataset. Clips are taken from 69 Hollywood feature movies, providing a very challenging setup.

All these correspond to a rich source of information, describing the key elements of the video. Most importantly, this kind of information was not generated on purpose for a computer vision experiment. Therefore, it is not biased towards potential computer vision applications and describes the content that really matters to the observer. Also, in most cases, it has been written (or spoken) by an expert in the field, trying to best describe the relevant events. This kind of data should be seen as much richer, more complete and faithful to the video stream than crowd-sourced descriptions. Video descriptions obtained on platforms such as Amazon Mechanical Turk are written by people who received precise orders and that are usually paid by the sentence they write. Moreover, such textual descriptions describe the video at different levels of detail and target very different tasks. This provides precise expert knowledge for very fine-grained action recognition. For instance "how-to" videos allow to learn models for very specific tasks, for which it would be very painful and time consuming to hire annotators.

Most recent advances in computer vision are due to fully-supervised machine learning techniques. These correspond to learning algorithms that are capable of building a model of an action (for instance "Standing Up") provided they are shown examples of this action. A typical learning scenario for action recognition would involve hav-

Figure 1-4 – Three frames from a scene in the movie Moonstruck that picture Loretta and Cosmo having champagne. The action "eat" as defined in standard action recognition dataset also include drinking. However, how is the "eating" action really defined? Are the temporal borders really clear? Does it start when one pours the drink into a glass (frame 1)? Does it correspond to holding the drink in hand (frame 2) or is it really the fact of pouring the drink in his mouth that matters (frame 3)?

ing a set of examples (videos) associated with a label. The label would be telling us whether or not the given video pictures the action of interest. We show in Fig. 1-3 several examples from a fully-supervised action recognition dataset [Marszalek et al., 2009]. The second observation that motivates this work is that manually defining such labels for video understanding is very hard. Indeed, scenes shown on video are very complex, usually portraying the interaction between multiple agents and objects, and go far beyond a single concept.

In the last decade, a substantial amount of research in video understanding has been dedicated to human action recognition. However, the definition of "action recognition" is still fuzzy, and providing a proper description of an action is a problem in itself. This is not only a problem faced in computer vision, and corresponds to a whole branch of philosophy. Most recent computer vision advances in the field define an action by a symbolic label that is shared by several videos. For example, in the context of sports video classification, labels correspond to what sport is practiced: Hockey, Soccer, Tennis *etc...* In the context of simpler, every day actions, the definitions are much more fuzzy. In previous work, given a fixed vocabulary of actions, the precise definition of an action is usually annotator dependent. All annotators will not have the same idea as to how an action looks like: for instance, is opening a fridge door an instance of the class "Open Door"? Also, people may often disagree on the temporal extent of the action: when does "drinking" start, and when does it end? We

illustrate how hard it is sometimes to properly set the temporal bounds of an action in Fig. 1-4. In our work, we propose to avoid defining action classes manually and exploit readily available data instead.

This work is also motivated by the fact that manual annotation of video data is highly tedious. This is due to many reasons, the first of them, being the lack of precise definition of what an event or action is. Therefore, annotating actions in movies requires the annotator to take many decisions, some of which can be quite arbitrary. Unlike objects, which have a precise spatio-temporal localization, an action's borders are much more fuzzy. Annotating a whole movie with action labels implies watching the movie a few times and setting *start* and *end* markers for every observed instance. This is a very time consuming process that requires a lot of attention and patience, and we believe that automated alignment can greatly lighten the annotation work. This way, the annotator would only provide rough localization, and the developed methods could take this imprecision into account and still come up with good models. Moreover, actually finding instances of a given action is hard because of their infrequent occurrence. For instance, on a set of 69 Hollywood movies, "answering a phone" happens on average 3.2 times per movie while "shaking hands" only happens 2.0 times.

Eventually, this line of work also allows us to move away from the traditional clip classification paradigm. Indeed, most work on action recognition and video understanding takes as input trimmed video clips that illustrate a specific action. This trimming, which is usually part of the annotation introduces a bias which is not present in freely available data (such as private video collections, TV, movies *etc.*). We propose to exploit boundaries as provided by a rough alignment of textual data, and refine this alignment if needed.

Based on the motivations described in this section, we want to design models that could use readily available meta data as a source of supervision. As explained before, such data offers information about a large set of tasks, for instance in the context of "how-to" videos. However, using it to learn models is complicated because this supervisory information is often imprecise, not localized in space nor time. Therefore

As the headwaiter takes them to a table they pass by the piano, and the woman looks at Sam. Sam, with a conscious effort, keeps his eyes on the keyboard as they go past. He appears to know this woman. After she has gone by Sam steals a look in her direction. BERGER, a slight, middle-aged man, observes the couple from a distance. The headwaiter seats Ilsa. Laszlo takes the chair opposite and surveys the room. Strasser and Renault look up at them from their table.

Figure 1-5 – Temporal alignment problem illustrated using a scene from Casablanca. The scene description as found in the movie script is shown above. We manually picked the corresponding frames and used a color code to align parts of sentences to them. This shows that even though the description from the script is rich in details, it is very imprecisely localized.

we need to be able to deal with noise and uncertainty in this very weak form of annotations. We will describe what are the main challenges related to our goal in the following section.

1.3 Challenges

We will try to sketch out the main challenges related to our goal. These are related to the encountered machine learning problems as well as to how the data should be represented. We discuss these two in detail below.

The main challenge addressed in this work is the lack of precise correspondence between the two streams of data (video and text). As mentioned before, one of the aims of this thesis was to design joint text and video models. However, such models are usually trained using pairs of data points in correspondence. As illustrated in

Fig. 1-5, when using movies with roughly aligned scripts, this correspondence is not given a priori. We only know that parts of the video data are related to parts of the textual description. In the figure, a long block of text is roughly aligned to a long video sequence. Even though all events described in the text do happen in the video, we don't know precisely when and where in the scene. We illustrate the correspondence that we would like to recover using a color scheme. As we will describe it more formally in the next chapter, we will refer to this kind of scenario as weakly-supervised learning. This concept is at the core of this thesis and is one of the main challenges that we faced while working with this kind of data.

The second challenge is related to the data representations that we have to use. The goal, as described above, would require a high-level video understanding, going further than simple classes or labels. Indeed, the represented scenes can only be understood by recognizing the agents, their relations and how they interact. It is not clear what would be a suitable representation to generate such high level interpretations. How should we represent the video? How should we represent the textual description? Should we embed both modalities to a common space? Should we use some kind of symbolic representation? This is an open problem both from the computer vision and the natural language understanding perspectives. Building symbolic representations from natural language is hard and would again provide a discrepancy between video and text. On the other hand, continuous text representations such as word2vec [Mikolov et al., 2013] or sentence embeddings discard the complex underlying structure.

1.4 Contributions and outline

This manuscript is organized into seven chapters, including this introduction. In Chapter 2 we present an overview of previously published work that is related to our goal and to the proposed models. After this review of the relevant literature, we provide in Chapter 3 a brief overview of models and methods used in our work. This will allow us to precisely define the terminology used in subsequent chapters.

Chapter 4 describes the first contribution, a model for jointly identifying the actors and actions they perform in movies. We propose to process textual descriptions and extract <PERSON, ACTION> pairs that we will use as a form of weak supervision. The idea is that character identities act as a strong cue to localize the action, allowing us to learn a model on the level of characters and not the whole frame. We learn a model for every character and models for actions using a discriminative clustering criterion [Bach and Harchaoui, 2007] and constraints that we derive from the aforementioned supervision. Using a simple convex relaxation, the cost function is optimized by solving a sequence of convex quadratic programs. We show experiments conducted on two feature movies: Casablanca and American Beauty. The developed model allowed to cope with uncertainty and imprecision in the data and outperformed the state-of-the-art algorithms at that time. This work was presented at ICCV 2013.

In Chapter 5, we describe the second contribution, a model for aligning sentences to video frames. Given the two streams, one composed of sentences and the other one of video frames, we want to find the temporal correspondence between the two. We do this under the assumption that the temporal ordering is maintained. As in the first contribution, we do so by using a discriminative clustering criterion and a structured optimization domain. We take advantage of the structure of the underlying space and optimize our cost function using the Frank-Wolfe algorithm [Frank and Wolfe, 1956]. The model is evaluated on two datasets: one composed of videos with sequences of labels, the other one composed of cooking videos along with textual descriptions in the form of steps. This work corresponds to publications presented at ECCV 2014 and ICCV 2015.

Finally, we conclude and discuss potential future work in Chapter 6.

Chapter 2

Related Work

In this section we review work that is closely related to this thesis. We first discuss previous methods for modeling text and images. This includes image captioning, image retrieval based on textual queries, and weakly-supervised methods using text to supervise visual models. Next, we describe approaches using textual descriptions of video data, and discuss some works on modeling the temporal aspect of videos. We conclude the chapter by a brief overview of the literature related to the models and optimization techniques that we will present in this thesis: discriminative clustering and the Frank-Wolfe optimization algorithm.

2.1 Images and text

2.1.1 Image captioning

During the last decade much work has been dedicated to the joint modeling of images and their textual descriptions. The ultimate goal is to represent both types of media in a shared space where their relations can be modeled and analyzed. Many approaches have been proposed and we describe some of them here.

Captioning as machine translation. One of the early attempts towards modeling images and text was presented in [Duygulu et al., 2002]. The idea was to use a machine

translation model to align parts of images to words from image captions. The authors propose to decompose images into regions and to represent each region by an entry in a dictionary. Given symbolic representations for words and image regions, the alignment was phrased as an IBM 2 model [Brown et al., 1990]. This model contains two distributions: the probability of an image region given a word and the probability of aligning a word to a given image region. As in the classical translation model, the optimal assignment is recovered using an expectation maximization (EM) algorithm. A broader set of such translation-inspired models is presented in [Barnard et al., 2003]. In addition to the aforementioned IBM model, the authors describe a multimodal extension of latent dirichlet allocation (LDA) and an extension of a hierarchical clustering model, using text and visual appearance.

The work of Duygulu et al. [2002] was extended a few years later in [Gupta and Davis, 2008] taking into account relationships between nouns. The previously used dataset is extended so as to contain not only a set of nouns for each image but also the spatial relations between them. A more complex probabilistic model that includes relationships between pairs of nouns is introduced and optimized using the EM algorithm. Quantitative results are shown on the extended dataset, and demonstrate significant improvements in image labeling performance for image labeling.

Fixed templates. Another line of work proposes to match images and sentences in a common semantic space. Farhadi et al. [2010] consider a semantic space composed of fixed template: a triplet of labels of the form `<o,a,s>` (object, action and scene). They model these triplets as a conditional random field (CRF) with unary potentials defined as functions of object detectors (for images) or words. The CRF is trained using a dataset composed of images with an associated textual description and a ground truth `<o,a,s>` triplet. In the experimental section, authors show that the common semantic representation can be used to annotate images by picking sentences that correspond to the same inferred triplet.

The template representation is further extended by Kulkarni et al. [2011]. Instead of including a single `<o,a,s>` triplet per image, the authors propose to create a node

per object detection. Each detection is further associated with an attribute node, and each pair of objects is associated with a preposition node. Unary potentials for this model come from standard object detectors, attribute and prepositional classifiers. The pairwise features are computed on a held-out set of data. Once the CRF has been trained, for a given test image one can perform inference and obtain the most likely pattern. The authors propose to generate captions using the words from this pattern and filling in "function words" using a language model. That way, a template `<<white,cloud>,in,<blue,sky>>` would be transformed into "There is a white cloud in the blue sky".

Retrieval and CCA. Much of the work on joint representations of text and images makes use of canonical correlation analysis (CCA) [Hotelling, 1936]. CCA consists in finding linear transformations of two variables such that their correlation is maximized. Performing CCA on two modalities finds two projections that allow to bring the two data sources into a common space. This has been explored for images and text by Hardoon et al. [2004], who propose to use a kernelized version of CCA. Sentences are described using term frequencies and standard visual features are computed on images. After learning the two projections, the authors propose to retrieve the closest image given a textual description.

Ordonez et al. [2011] propose to return the caption of an image that is most similar to the query image. The proposed model does not use CCA but relies on the construction of a very large captioned image database. Given a query image, a set of similar images in the database is found based on scene features, such as GIST, of heavily sub-sampled thumbnails. The set of retrieved images is re ranked based on various scores, based on object detections, attributes, people and scene features. The caption of the highest ranked image is returned as a result.

CCA-based image captioning methods are discussed in general by Hodosh et al. [2013]. The authors introduce a large benchmark dataset and compare different variants of the model, using various feature representations. Also, the evaluation metrics for image captioning are discussed, and are compared to human judgment. The top-

ics discussed in this paper are particularly interesting given how hard this task is to evaluate.

Extensions to the kernel CCA-based models are proposed in [Gong et al., 2014a,b]. Gong et al. [2014a] propose a multi-view variant where the CCA is computed between three modalities. In that work, the modalities include images, tags that one can typically find in meta data on the web, and image classes. The empirical evaluation includes experiments on several datasets with various retrieval schemes: from tag to image, from image to tag. Gong et al. [2014b] propose to help the image-text embedding using a large but weakly annotated set of images. The main dataset is Flickr30K, where each image is associated with a precise description in natural language. This work investigates whether using the much larger but imprecise Flickr1M and SBU1M datasets can enhance the quality of the embeddings. Indeed, images from these datasets contain titles, tags and very imprecise textual descriptions. The authors demonstrate empirically a moderate improvement in performance when additional data is used.

Not directly related to image captioning (with complete sentences), but also formulated as a retrieval problem, Weston et al. [2011] propose to learn a model for annotation retrieval given a query image. Images are annotated by a single tag, but the datasets used to train the model are "web scale": they contain 10M training images. The model is trained using a ranking loss, learning an embedding of visual features and annotations such that the correct label is ranked as high as possible. The embeddings learnt that way for the annotations exhibit some semantic information, as nearest neighbor queries amongst annotation embeddings show interesting similarities.

Deep models. Recently, there has also been an important amount of work on joint models for images and text using neural networks. Many caption generation (or retrieval) variants have been proposed in 2014 and 2015 some of which we will describe here. A significant part of these still rely on a ranking loss and formulate captioning as retrieval, but some propose to learn a caption generation model. Image

captioning can either be evaluated by a retrieval metric or by comparing the single proposed caption with the ground-truth one. The advantages of the two approaches are discussed in Hodosh et al. [2013], with an indication that the ranking measures provided more reliable numbers. We will now describe in detail some of the proposed deep captioning models.

Closely related to [Weston et al., 2011], the work of Frome et al. [2013] introduces a ranking-based image annotation model. It has the advantage of considering non linear transformations of the image representation and being able to generalize to annotations that have never been seen in the training set. Annotation embeddings are initialized using word2vec representations [Mikolov et al., 2013], and images are represented using a deep convolutional neural network. The similarity between annotations and images is measured using a trainable bilinear form M. At first, only the bilinear form M is optimized, while in the second step of the optimization, the loss is also propagated to the image representation.

While the previously described work computes embeddings for single words of phrases, similar models have been developed for whole sentences [Socher et al., 2014]. In this article, the authors propose to compute a sentence representation and use a ranking loss between images and sentences. A global image representation is used and the sentence is described using a dependency tree recurrent neural network (DT-RNN). In a classical RNN, the nodes of the network correspond to words that are connected as a chain. In this work, the authors propose to create one node per word and connect them following an automatically computed dependency tree. The sentence representation is computed at the root. As for other works, the whole model is trained using a ranking loss and compared to previous work, including CCA-based models.

The previously described model measured the similarity between a global image representation and a whole sentence. Another ranking-based model has been proposed in [Karpathy et al., 2014], where fragments of an image (object detections) are aligned to fragments of a sentence (dependency relations). The similarity between the image and a given sentence is measured using a latent alignment cost. All parameters of

the model, including the alignment and fragment embeddings are trained as before using a ranking loss. The great advantage of this model is that the alignment was modeled explicitly which allows interesting interpretations. This model is extended and simplified in [Karpathy and Fei-Fei, 2014], where sentence fragments correspond to words. Words are embedded using a bidirectional recurrent neural network, which allows at test time not only to retrieve the closest caption but to actually generate one.

A very simple method, using global image representations and capable of generating new captions is proposed in [Vinyals et al., 2015]. The work described in the previous section makes use of object detections and tries to resolve the alignment of words to parts of the image. In this article, a global image representation obtained from a very large CNN is fed as the first input of a recurrent neural network for text generation. The RNN takes as first input a linear transformation of the image representation and then subsequently the successive words of the ground-truth caption. At test time, a novel sentence can be sampled by feeding the output of the image representation and then performing a beam search on the RNN outputs. The proposed model provides good performance while only having access to a global representation of images. Authors evaluate this approach using the BLEU score, but also report some ranking experiments. Captions of the training set are ranked by assigning them a probability given the query image. While the ranking results are encouraging, the authors argue that ranking is not the correct way to evaluate image captioning, hence, opposing the conclusions of Hodosh et al. [2013].

An alternative, phrase-based approach has been proposed by Lebret et al. [2015]. Phrases are defined as a group of words that express a specific idea, and are here classified into noun phrases, prepositional phrases and verb phrases. The authors propose to learn a joint model of images and phrases using a maximum likelihood criterion. The model assumes that the conditional probabilities of phrases from the same sentence, conditioned on the image, are independent. Then, the conditional probability of a given phrase is defined using a soft max of a bilinear product between a phrase embedding and an image representation. Images are represented using the

output of a CNN while phrases are represented by the average of precomputed word embeddings. After optimizing the negative log-likelihood, at test time, a caption is generated by fitting a language model to the most likely phrases. The last re-ranking step, exploiting visual information, is used to choose the closest sentence. Please note that this method is generating novel captions and does not make use of a ranking loss.

Image captioning and text/image retrieval is a very interesting area of research that is currently lacking a proper task definition and suitable evaluation metrics. In the following section we will discuss how text and image captions can be used as weak supervision, for instance for face recognition.

2.1.2 Text as supervision

Textual information has extensively been used as a form of weak supervision to train visual models. This has especially been exploited in the context of person recognition in news photos. Indeed, news pictures are usually associated with captions which potentially describes the people pictured in the image. Berg et al. [2004] were among the first to cope with this rich, free, yet imprecise data. The authors propose to detect faces and rectify them based on facial landmark locations to match a canonical pose. Pixel values of the image are used as features and are projected to a lower dimensional space using kernel principal component analysis (kPCA). Using images with unambiguous labels (portraits with a single name) the authors propose to perform linear discriminant analysis. In this lower dimensional space, a modified k-means clustering procedure is carried out to obtain face clusters associated with names. This work has been extended to a probabilistic formulation in [Berg et al., 2005], using a generative model of faces. The authors have also proposed to model how likely a name is to appear in the picture, conditioned on textual context information. The proposed model is optimized using expectation maximization and provides better results as compared to [Berg et al., 2004].

Related to the work described above, Luo et al. [2009] propose to extend person identification to action recognition. The goal is now not only to recognize celebrities,

but also what they are doing. This article has relations to the previously described work on modeling relations between objects in images and text [Gupta and Davis, 2008]. In a similar spirit, the authors proposed a complete probabilistic model of person identities and related actions. The parameters of the appearance model and the text to image assignments are obtained using the EM algorithm. Empirical evaluation shows that modeling identities and actions jointly works better that only modeling identities. This work is quite similar in spirit to what we will present in Chapter 4.

Wang and Mori [2010] consider a model that allows to classify objects in images using latent attributes. The model does not take natural language as input but works on a "structured" label set, and makes use of richer labels. The model is cast in the latent support vector machine (L-SVM) framework with a potential function composed of several terms: an object class model, an attribute model, a class conditioned attribute model, an attribute coocurence term and a class-attribute coocurence term. When the latent variables are unknown in the training phase, the resulting optimization problem is non-convex. On the datasets used in the experiments, the attributes are available at training time but the authors show that using observed latent variables during training degrades the performance. The authors propose a non-convex cutting plane optimization algorithm and show experiments on two object-attribute datasets, demonstrating significant improvement over contemporary baselines.

2.2 Video and text

Text has also been extensively used with video data. First, movie scripts provide a rich source of annotations for action recognition and person identification in movies. This kind of supervision is weak as it does not provide precise sample-label correspondences. Several models have been proposed to cope with this kind of uncertainty, often yielding good models with only light supervisory burden. Following the attempts at image captioning, many approaches have been proposed for video captioning. These rely on well annotated video corpora, some of which we will describe here. The closest to our work are methods focusing on the task of aligning videos with corresponding

textual descriptions. Some of these methods will be described below.

2.2.1 Weak supervision

Similar to the work on captions and faces in news, subtitles and movie scripts have been used together with the video signal to train action and character models. Movie scripts provide a precise description of what is happening on screen. Just like the movie, they are divided into scenes, specifying the setting that will take place. They include two kinds of textual information: dialogues and scene descriptions. Dialogues contain the name of the character that is speaking as well as the pronounced words. In between these dialogues, the script contains scene descriptions which describe how the characters behave and the enviornment in which they evolve. These scripts however do not contain any precise timing indications and are therefore unaligned with the actual movie. Most of them are shooting scripts, written before the shooting, and the final movie can differ (because of actor's performance, editing *etc.*).

Thanks to the expansion of collaborative subtitle sharing, textual transcripts of the monologues are freely available on the web. These precisely temporally aligned transcriptions provide a reliable description of the character's words. One can match the text found in the subtitles with dialogues from the scripts in order to roughly align the script in time. Exploiting this data as a form of supervision was first proposed in Everingham et al. [2006] and gave rise to many subsequent models. We will describe some of the most important ones here.

Face recognition. Everingham et al. [2006] propose to link subtitles to scripts in order to recover the identities of the speakers. Time-stamped subtitles with speaker identities can be then used as reliable labels to recognize people. In this early work, frontal faces are detected and tracked. Then facial landmarks are localized and described using either SIFT or raw pixel values. The link between subtitles and video is made by detecting lip motion to decide which character is speaking at that given moment. Using conservative thresholds, a reliable set of face tracks is assigned the correct character identity. The other tracks in the video are assigned to an identity

using a simple probabilistic model.

This work is further extended in [Sivic et al., 2009] by introducing several improvements. First of all, a profile face detector is added, improving the face detection coverage. Better face descriptors are used, and a model is learned using a kernel SVM instead of simply relying on a distance in the feature space. The kernel is a mixture of different kernels, one per facial landmark, and between landmarks the closest descriptors in both tracks is used to compute the kernel entry. The weights of the mixture are learned and the whole pipeline provides much better performance. However, the use of textual information is identical to the previous work, and relies on heuristic matching of monologue speakers to lip motion.

An interesting alternative is proposed by Cour et al. [2009]. Instead of relying on the explicit modeling of speakers to assign labels to face tracks, the authors propose to formulate the problem as an ambiguously labeled one. In the proposed model a data sample can be assigned to multiple labels, as subtitles often correspond to a dialogue. The authors propose a loss that is suitable for this kind of ambiguous setting together with a convex surrogate that leads to a tractable learning algorithm. The method is evaluated on images with captions and on the TV series "Lost". This way of handling ambiguous script information in [Cour et al., 2009] is related to our contribution described in Chapter 4.

A method very similar in spirit to the contribution described in Chapter 4 is presented in [Ramanathan et al., 2014]. While building upon our weakly-supervised character recognition model, the authors add interesting extensions. This work not only uses character mentions in the script to generate constraints, but also includes a model for co-reference resolution in the text. Indeed, characters are not always named using their full name and the joint text-and-video model allows to both improve face recognition and the co-reference. The authors demonstrate experimentally that both problems actually benefit from this joint model and evaluate this improvement along the iterations of the algorithm.

The work that we describe here only use scripts to recognize characters. In our work, we also investigate scripts as a source of supervision for training action models.

In the following section we will review some related publications that used movies and script data as a source of supervision of some sort.

Action recognition. Movies with associated scripts were first used in [Laptev et al., 2008] to automatically build an action recognition dataset. The authors propose to train a text classifier that predicts whether the script mentions an action or not. Eight different actions are considered and the classifier is trained on an annotated set of 12 movie scripts. Retrieving relevant scene descriptions using this classifier works much better than when using simple regular expression matching. The video corresponding to the classified scene description is used as training samples for learning an action classifier. Using the raw piece of video is compared with using a cleaned-up version where the temporal boundaries have been corrected by an annotator.

This work is extended in [Marszalek et al., 2009], adding several actions and better exploiting textual data. Apart from learning action models, this paper shows how to train scene classifiers using script-based supervision and evaluates both. A richer model of actions given a scene context is proposed where the influence of scenes on actions is either obtained from scripts (by counting) or trained. The action recognition dataset obtained in this paper constitutes the Hollywood2 dataset, which is a well-known action recognition benchmark.

The previously described methods provide accurately labeled video clips but the temporal bounds are often imprecise. Duchenne et al. [2009] address this problem by trying to automatically adjust these bounds based on a discriminative cost. Given the imprecise temporal bounds (extended by a given amount), the model selects the temporal window inside that is most discriminative. The problem is formulated as a discriminative clustering, based on the hinge loss. The optimization is carried out by a coordinate descent approach, iterating between learning the optimal model and picking the most discriminative window. This cleanup process is evaluated for the end task of action detection in movies for two classes of interest. The authors show that the cleaned-up windows provide better training samples than the raw ones but still worse than the ground-truth ones. This work is related to what we will describe in

Chapter 5, where a list of actions is aligned to a video sequence.

2.2.2 Datasets

Several datasets with video descriptions have been released and used for building joint video and text models. In this section, we describe those that contain curated textual descriptions, written on purpose or manually corrected in some way. Regneri et al. [2013] present a video dataset designed to work on grounding textual descriptions. The point is to discover the "meaning" of sentences by looking at the corresponding visual data. The main contribution of this paper is the dataset, composed of 212 HD videos, each described by 20 different annotators. The videos correspond to a restricted cooking setup, where simple recipes are prepared in front of a static camera. All videos come from a larger cooking dataset that is described in details in [Rohrbach et al., 2012a]. Descriptions are obtained using Amazon Mechanical Turk. Annotators were asked to describe the content of the video in 5 to 15 steps, each step being one sentence. Additional annotations were then added to measure the similarity between descriptions. Overall the dataset is interesting but of limited size, if one wants to train language models on it.

Another dataset of videos with associated textual descriptions was introduced in [Rohrbach et al., 2015]. The dataset is composed of 94 movies, out of which 55 are provided with Audio Descriptions and 50 with movie scripts. A total of 11 movies are provided with both the audio description and the script which allows the authors to compare the quality of the two kinds of textual descriptions. Scripts are obtained from the web, aligned using subtitles as explained earlier, and then the alignment is corrected manually. Audio descriptions are descriptions that are provided for the visually impaired and describe precisely what is happening on screen. The authors propose to transcribe them to text using a crowd-sourcing audio transcription service. The dataset is composed of 68337 video clips with a textual description, yielding in total 74 hours of video with more than 650k words. This dataset is much larger and contains very challenging visual content as compared to the restricted cooking setup. Some of the works described in the following section make use of the datasets that

we have presented here.

2.2.3 Captioning

Another important line of work, fostered by the development of the datasets mentioned above, is automatic video captioning. Following the successes of image captioning, many variants of captioning models have been proposed. They all rely on the same key idea which is related to machine translation models. The video signal is encoded in some way into a latent representation, which in turn is transformed into a natural language sentence. In this section we describe some recent publications that propose such models.

One of the first attempts at automatic video captioning was proposed in [Rohrbach et al., 2013]. In the spirit of [Farhadi et al., 2010], based on video features, a CRF is trained for a 5-tuple of classes: activity, tool, object, source, target. Using this fixed template representation, a translation model is trained to generate sentences. The parameters are optimized on the dataset presented in [Regneri et al., 2013] and the performance is evaluated using the BLEU score. During training, because of the way the dataset is made, a single sentence can be assigned to multiple templates. The authors explore different ways to merge or select the relevant template for training.

Donahue et al. [2014] propose to generate video captions using a long short-term memory network-based (LSTM) language model. Two other applications are also shown in this paper, including video activity recognition and image captioning but let us focus on the video description model here. The same CRF as described in the previous paragraph is trained on the video signal. The output of the CRF is then fed to the LSTM, which is trained to output the actual video descriptions. Three different strategies of feeding the CRF output to the LSTM are studied in the article. The authors tried using the most likely 5-tuple as an input to and encoder-decoder LSTM, or using the output of the CRF as a constant input to a simple decoder model.

A very similar model is then presented in [Venugopalan et al., 2015b]. Instead of using a fixed template representation, a global video representation is used as a constant input to an LSTM decoder. The global representation is obtained by

taking the average activation of a CNN along the frames of the video. This can be seen as simplification of the previously described model, where no intermediate CRF representation is used. Also, the authors evaluate their model on a different video dataset.

A third variant of this model is introduced in [Venugopalan et al., 2015a]. In a fashion similar to [Donahue et al., 2014], the authors propose an encoder-decoder approach. However, instead of taking as input the 5-tuple, the decoder directly takes as input the sequence of CNN outputs on frames. As compared to [Venugopalan et al., 2015b], no average operation is performed over time, keeping the dynamic aspect of the video. The three works are different variants of the same model, always generating a sentence using an LSTM, with different ways of feeding the video as an input.

In a concurrent work, Yao et al. [2015] describe an encoder-decoder video captioning model with an attention model. The decoder LSTM is conditioned on a visual feature $\phi_t(V)$ which is computed for every position t in the output sentence. This feature uses an attention mechanism which weighs the frames of the video based on previously generated words and features of the given frame. This approach experimentally outperforms the method described in [Venugopalan et al., 2015b], showing the usefulness of the attention mechanism.

All these works produce natural language captions for videos which are inherently hard to evaluate quantitatively. The same problems as the ones discussed in the context of image captioning arise and still remain unanswered. In the following section we will describe models that instead focus on the problem of aligning videos with textual descriptions of some kind.

2.2.4 Alignment

Most related to our work, some previous methods focus on the alignment of textual data with video. The methodology described in [Everingham et al., 2006] allows to align whole movie scripts with the associated video data. However, it makes use of subtitles which are redundant with the dialogues in the script and have precise temporal boundaries. Sankar et al. [2009] discuss a model for aligning a script with

a movie when no such subtitles are available. The alignment then needs to rely on visual data, in order to match descriptions in the script to shots in the movie. The authors propose to find the optimal alignment using a criterion composed of three terms: One based on the scene location, another one based on character recognition and one on automatic speech transcription. The problem is solved using dynamic programming and evaluated provided ground-truth script to video alignment. The components of this method rely on several heuristics, which make strong assumptions on the structure of the movie (or show). An interesting application is shown, where scripts from silent films are aligned to the video.

More recently, Tapaswi et al. [2015] propose to align books to their movie (or TV show) adaptations. The alignment is carried out at the level of chapters for books and scenes in the movie. Scenes in the movie are found using the method described in [Sankar et al., 2009], while the segmentations into chapters is straightforward. Similarity between scenes and chapters is based on textual features like character occurrences and dialogue matching. The whole optimization is framed as a shortest path problem in a properly defined graph. Aside from text-based features, the article describes two priors that encourage a linear alignment (it is costly to jump forward and backward in time). However, visual information is only used to segment the movie into scenes and to generate the list of characters that appear in a scene.

A related and contemporary work [Zhu et al., 2015] proposes to align books to their movie adaptation on the level of sentences and shots. The main difference with the previous work is that this alignment is recovered using visual information. The authors propose to learn a bilinear form between the video and sentence representations using a ranking loss. They use sentence representations pre-trained of a very large book dataset, using a model they published at the same time in [Kiros et al., 2015]. Videos are simply represented using an average CNN activation along time. The bilinear form is trained on the Movie Description dataset described above. The alignment score is a linear combination of several features, including the aforementioned bilinear form, priors and similarities between dialogues. Eventually, the alignment is recovered using a CRF model with well tailored pairwise potentials.

Another line of work consist in aligning web videos with corresponding steps of a cooking recipe [Malmaud et al., 2015]. In this work, videos are aligned to steps of the recipe based on the transcription of the speech signal. A hand-tuned HMM (transition parameters set by hand) is used to align the words of the transcript to steps of the recipe. This alignment is then further refined based on visual clues: the authors propose to train more than 2800 food detectors (including ingredients as well as full dishes) to refine the correspondence. The proposed method is used on a very large set of more than 180k videos automatically harvested from the web. As opposed to what we will present in Chapter 5, this does not involve training joint models. The described model relies on well suited heuristics and pre-trained representations (word embeddings and visual detectors).

2.3 Temporal models for video

2.3.1 Action models

Action recognition in videos, and more generally video understanding has an inherent temporal aspect. In the past decade, state-of-the-art video descriptors have been based on local variations of pixel values in space but also time. Different feature point and descriptors have been proposed [Laptev, 2005, Wang et al., 2011]. However, the global temporal structure of the video has not always been modeled. In this section, we will describe some action recognition and video understanding papers that try to include a global temporal model.

A first attempt at including a video-level temporal structure for action recognition is presented in [Laptev et al., 2008]. The idea consists in splitting the video volume into bins, and compute bag-of-words representations inside these. This is very similar in spirit to the spatial pyramids for image classification [Lazebnik et al., 2006]. The paper proposes an extensive evaluation of this scheme on the KTH and movie action datasets. Different spatio-temporal tilings are tried and experiments show that different splits are optimal for different actions. For instance, the action "Kiss" does not

require any tiles, while "AnswerPhone" works best when splitting the video into 3.

Most action recognition datasets are composed of video clips that are associated with an action label. The actual action is happening somewhere in the video but often the temporal boundaries of the video clip are arbitrary. Several papers try to cope with the problem of automatically finding the best temporal bounds to learn a better model [Duchenne et al., 2009, Satkin and Hebert, 2010]. In Duchenne et al. [2009], as explained before, the goal is to clean up the temporal bounds that were obtained from a movie script. The authors propose a discriminative clustering cost function based on the hinge loss and optimize it using a coordinate descent approach. The algorithm alternates between finding the optimal model and selecting the most discriminative part of the video. The work described in Satkin and Hebert [2010] is very similar in spirit but comes from another motivation. In order to improve test-time performance of classification models, the authors propose to improve the temporal bounds of training video clips. A heuristic optimization algorithm based on a leave-on-out scheme is proposed. Supposing that there are N video clips, classifiers are trained on $N-1$ videos and all cropping of the remaining video are evaluated. Best performing cropping of a given length are selected for each video and the model is retrained on them. The authors show improved performance on several action classification dataset, showing that the proper localization of the action boundaries plays an important role.

A more complex temporal model for actions is proposed in Niebles et al. [2010]. In order to better perform action classification, the authors propose a structured classification model. It is composed of K different classifiers that have access to different portions of the video, at different scales. Their exact position is latent and the displacement with respect to an anchor position is penalized using a quadratic cost. The proposed formulation allows to train the parameters of each of the K classifiers as well as the displacement penalties. Using this model allows to exploit temporal structure as these classifiers focus on different parts of the video. However, this publication remains unclear as to how to properly set the anchor points of these parts which seems like a critical part of the model.

A somewhat similar idea is presented in a work published around the same time [Gaidon et al., 2011]. The authors propose a structured model for action recognition based on a sequence of "actoms". Each actom is a key step in realizing an action; for instance, for the action "drinking" it could correspond to pouring the drink, lifting the glass and then actually drinking. In order to describe an action, features are pooled around an actom's location, using a weighting scheme which depends on the distance to the actom's location. A generative model of the actom's location and a discriminative model of each actom are learned on training data. The authors propose a dataset of videos annotated with temporally annotated actom's locations. The model is used to perform action detection and performance is compared to the detection results found in Duchenne et al. [2009].

Tang et al. [2012] propose to use a temporal model with latent classes to represent videos. The video is subdivided into short segments, and each such segment has an associated feature vector and a latent class. In order to cope with variable duration of latent classes, each segment is also associated with a duration variable. An energy for the whole model is defined using three potentials, depending on the features, latent and duration variables. The inference of hidden states can be carried out using dynamic programming. All model parameters are learnt on an action classification dataset, where each video is assigned to a binary action label. The optimization alternates between inferring the most likely hidden states and learning model. In the experimental section, the authors compare to Niebles et al. [2010] on two datasets, showing significantly better performance.

A temporal model for action segmentation is proposed in Nguyen et al. [2011]. The task the authors aim at solving is video segmentation: the video is cut into segments and each segment is associated with a label. An action classifier is trained on a set of videos with provided segmentation. Then at test time, the optimal temporal segmentation is obtained by solving a well suited optimization problem. The cost function encourages selecting a segmentation such that the margin between the best class and the second best is maximized. This problem can is then solved using dynamic programming. Overall, this approach can be seen as a way to perform non-maximum

suppression in time for action detection. An empirical evaluation is conducted on two datasets, one composed of bee motion trajectories and on a dataset composed of concatenated video clips from an action classification set.

An interesting structured approach to action recognition was proposed in [Lan et al., 2011]. The authors propose a joint spatio-temporal action recognition and localization in videos. Instead of using the whole video, including background, to prediction action classes, motion features are used arround a person detection. The model is formulated as a latent SVM model, with the location of "action bounding boxes" being latent. The model incorporate a unary term which measures if the given location is discriminative for the given class and a binary potential for smoothness of these locations in time. An efficient inference and learning algorithm is proposed but the model requires supervised bounding box locations at training time. The model is evaluated on the UCF-Sports dataset. This was further extended by Shapovalova et al. [2012], including a weakly-supervised model that could train without groud-truth bounding box locations for training.

2.3.2 Composite activities

Apart from proposing temporally structured model for recognition of individual actions, some works focus on composite activities. These are composed of a sequence of actions, with complex interactions between individual agents and objects. This can be seen as a "macro" scale approach to action recognition, where a whole scene is composed of shorter events. A perfect illustrative example is sport videos, where an event is composed of multiple actions performed by many players.

A rich temporal model of actions is proposed in Gupta et al. [2009]. The authors describe a complex video understanding model based on AND-OR graphs. Nodes of the graph represent elementary actions, for instance in the context of baseball games: "Run", "Throw", "Miss"... The graph describes causal relations between these actions, in the form of AND-OR relations. For example, after pitching, either the batter can "Hit" OR "Miss". The work described in this paper aims at simultaneously building this relational graph and parsing the corresponding videos accordingly. This implies

associating a valid storyline in the graph with tracks of people in the video. The optimal model is obtained using an alternate optimization scheme, iterating between modifying the graph and fitting the graph to a video. Fitting the graph to the video is itself an alternate optimization, switching between the association of nodes to tracks and finding optimal parameters for each action (node). The tackled problem is very hard and the proposed model very ambitious, and it relies on a good heuristic initialization.

Kwak et al. [2011] propose a composite activity recognition based on a scenario constrained model. A scenario describes how atomic events (called primitives) are temporally related by predicates. These relational predicates allow to encode that some primitives happens before others, byt also the exclusion of two primitives and so on. The authors propose to annotate a video by considering the scenario as a set of constraints on the occurrence of primitives in time. An algorithm for transforming a scenario into what they call a constraint flow is proposed. Then, composite events can be recognized using a constrained optimization by dynamic programing. The authors demonstrate the performance of their algorithm on two datasets, one composed of surveillance videos and one on a Tennis video.

Vahdat et al. [2013] propose a compositional model for event detection with latent temporal segment positions. The authors formulate the problem as a multiple kernel learning latent support vector machine, allowing to select discriminative portions of the video and to weight different feature channels in a principled manner. This allows to cope with temporal clutter, effectively ignoring noisy parts of the video (background). The authors demonstrated the performance of the proposed model on the TRECVID MED 11 dataset.

Another approach at composite activity recognition is proposed in [Rohrbach et al., 2012b]. In this work, composite activities correspond to cooking recipes while the low-level, atomic attributes are activities (peeling or washing) and participants (knife, cucumber). The authors propose to use a pre-trained attribute scoring function at each frame of the video. Several features are computed using these scores, by taking the maximal score along time, or a coocurence of scores. This allows to build

the final feature representation for the video sequence. Composite activities are recognized using standard classification algorithms (SVM or nearest neighbors) using this representation as input. An interesting improvement is proposed, where features of this representation (corresponding to low level attributes) are weighted using outside script data. This corresponds to weighting participants and activities using cooking recipes available on-line. In the experimental evaluation, best performance is obtained when using nearest neighbors along with this pre-computed feature weighting. The proposed approach had the great advantage of working for videos picturing composite activities that were not seen in the training videos (but seen in the cooking recipes).

2.4 Learning and optimization

2.4.1 Discriminative clustering

Discriminative clustering is an unsupervised method that partitions data by minimizing a discriminative objective. The models correspond to learning a classifier and optimizing over both classifiers and labels [Bach and Harchaoui, 2007, Xu et al., 2004]. They partition the data into clusters, such that these clusters are easily modeled using a classifier of the same class. This kind of models, and their extentions have been used in computer vision and natural language processing. We will review some of the applications from the litterature in this section.

Some convex formulations and relaxations of discriminative clustering have been explored in the past. In particular, Bach and Harchaoui [2007] propose a clustering objective based on linear models along with the squared loss. The resulting optimization problem is NP hard and a relaxation of a lifted, equivalent problem is proposed. We will review this model in detail in Chapter 3 and all our contributions are based on this framework. Related to the limitations of discriminative clustering models and their trivial solutions, Guo and Schuurmans [2007] describe a convex relaxation for latent variable expectation maximisation and discuss the trivial solutions. They show that these solutions can be avoided as long as one works with the equivalence re-

lations between hidden variables instead (corresponding to the lifting employed in DIFFRAC).

In computer vision these discriminative clustering models have been successfully applied to co-segmentation. Joulin et al. [2010] formulate the co-segmentation problem as a clustering one and use a cost function composed of two terms. One of them, responsible of spatial coherence of the segmentation is based on normalized cuts while the second one is similar to DIFFRAC. The optimization is performed in the lifted equivalence matrix space, using an algorithm for optimization on matrix manifolds [Journée et al., 2010]. This work is then extended in [Joulin et al., 2012] to a multiclass model, using the softmax loss instead of the squared one. The resulting optimization problem is not convex and is initialized using the model from [Joulin et al., 2010], which can be interpreted as a relaxation.

As mentioned before, an extension of the contribution presented in Chapter 4 is presented in [Ramanathan et al., 2014]. A discriminative clustering cost function based on the squared loss is used along with linear constraints to recognize characters in movies. The proposed improvements are two fold: first, additional equality constraints allowed to rule out a whole set of characters for large portions of the movie. Indeed, for a given scene in the movie, if a character is never mentionned in the script, the authors propose to set its latent class variable to zero. Moreover, the proposed approach includes a coreference resolution model, which allows to resolve who does the pronouns correspond to.

Joulin et al. [2014] propose a model for object co-localization, where occurences of the same object in different frames are localized. The cost function is again based on a squared loss and the authors propose some temporal constraints. The optimization is carried out using the Frank-Wolfe algorithm as in the contribution presented in Chapter 5. This allows the authors to exploit the structure of the temporal constraints that they propose, instead of using a classical quadratic problem solver.

In the natural language processing (NLP) litterature, this kind of clustering based on the squared loss has been used for weakly-supervised relation extraction. Grave [2014] propose to use a squared-loss objective along with linear constraints to detect

relations between entities in text. For instance, detecting the relation `BornIn` in the sentence "Ernest Hemingway was born in Oak Park". A model for dependency parsing, based on this discriminative clustering criterion, is proposed in [Grave and Elhadad, 2015]. Similar in spirit to the contribution presented in Chapter 5, the authors propose to use a structured set of constraints. Since a valid dependecy parse of a sentence is a tree, the authors propose to carry out the optimization over the set of spanning trees. As in our contribution, the optimization is performed using the Frank-Wolfe algorihtm which allows to efficiently exploit the structure of the problem. Indeed, the linear minimization oracle over the set of depdency trees corresponds to a minimum spanning tree algorithm.

2.4.2 Frank-Wolfe

In our work we use the Frank-Wolfe algorithm (a.k.a conditional gradient) to minimize our cost function while exploiting the alignment structure of our problem. The Frank-Wolfe algorithm [Frank and Wolfe, 1956, Jaggi, 2013] is a classical convex optimization procedure that permits optimizing a continuously differentiable convex function over a convex compact domain only by optimizing linear functions over that domain. It is particularly well suited for problems where the explicit representation of the domain is complicated but efficient algorithms are available to minimize linear functions over it. As explained in the previous section, this typically include problems with structures such as alignments and trees.

We describe the classical version of this algorithm in Chapter 5. Several other variants exist, including Frank-Wolfe with away steps, pairwise away steps and the fully corrective version. All these variants are described in [Jaggi, 2013].

Chapter 3

Background

In this chapter we provide some background on the type of models that we use in this work. In the context of discriminative models, we go through the different levels of supervision that can be used. This will allow us to define the terminology that will be used in the rest of this thesis and clarify the distinctions. In the second part of this chapter we discuss clustering algorithms, in particular the DIFFRAC [Bach and Harchaoui, 2007] framework. Finally we describe how we exploit this discriminative clustering framework, incorporate supervision as constraints and how this changes the problem from clustering to classification.

3.1 Learning and supervision

The models that we have designed and that are described in this thesis are weakly supervised. In this section we provide a precise description of what we mean by weak supervision and compare this to other classical supervisory setups. All this is described in the context of learning a discriminative model, for instance a multiclass classifier. In what follows, let us assume that we have data points that we denote by x, that live in feature space $\mathcal{X}$. Data points can be associated with what we call a label y that lives in the label space $\mathcal{Y}$. For instance, in the case of binary classification in a d dimensional space, we have $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$.

Our goal is to learn a discriminative model of our data, capable of predicting

a label given a feature. That is, to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that we refer to as *classifier*. We denote the set of classifiers that we consider by $\mathcal{F}$. We call *learning* when we find the function f in $\mathcal{F}$ such that some criterion is minimized. The criterion that is used depends on the nature of the supervision that we are given and we describe some scenarios in the next sections.

3.1.1 Fully-supervised learning

The most common setup is the fully supervised one, in which one is provided with an annotated dataset, *i.e.*, a set of N pairs indexed by n of the form (x_n, y_n). The goal then is to find the function f such that the prediction for each x_n is as close as possible to y_n. The notion of proximity is measured by a *loss function* that we will denote by $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. The problem of finding the optimal function is refered to as the *empirical risk minimization* and corresponds to the following optimization problem:

$$\min_{f \in \mathcal{F}} \ \frac{1}{N} \sum_{n=1}^{N} \ell(f(x_n), y_n). \tag{3.1}$$

In order to avoid *overfitting* issues, there are some aditional terms in the cost function that act as a *regularization* over the class of functions. This is however out of the scope of this work and we will not consider regularization for the sake of simplicity.

In many cases, obtaining a label y_n for every data point can be quite costly. One can think of applications to biology or medicine, where the annotation has to be done by an expert and requires a lot of time. Many computer vision tasks require costly and complex annotations, image segmentation or action recognition are good examples. In that case, it is of great practical interest to design methods that can recover a classifier f from *imprecise* data. For instance, in the image segmentation task, one can provide very rough contours of objects, and ignoring the details. In the following section we formalize the notion of imprecise annotation.

3.1.2 Weakly-supervised learning

We call weak supervision the setup in which for each sample x_n we don't have a precise label y_n but a set of samples $\mathcal{Y}_n$, typically such that $\mathcal{Y}_n \subseteq \mathcal{Y}$. If for each sample this subset only contains one element, we recover the fully-supervised setup. Otherwise, one can exploit this imprecise information by simply minimizing the best-case discrepancy between the prediction $f(x_n)$ and elements of $\mathcal{Y}_n$. That would correspond to the following class of optimization problems:

$$\min_{f \in \mathcal{F}} \ \frac{1}{N} \sum_{n=1}^{N} \left[\min_{y \in \mathcal{Y}_n} \ \ell(f(x_n), y) \right]. \tag{3.2}$$

A simple illustration of such a scenario is trying to learn celebrity classifiers given newspaper pictures [Berg et al., 2004]. Every picture contains several samples, one per face in the image, and the label set $\mathcal{Y}_n$ is the set of people mentionned in the caption. Every face is therefore associated with several names and learning a model per identity requires to solve the kind of problem mentionned before.

The label can be seen as a latent variable, where the supervision constrains the space in which the labels lives. All the models described in this thesis rely on this setup, with different label spaces $\mathcal{Y}_n$.

3.1.3 Unsupervised learning

When no labeling information is provided about the data points, we call that the *unsupervised* setup. In the context of learning a discriminative model, this leads to what we refer to as *discriminative clustering* that will be described in detail in Sec. 3.2.2. Using the previously described notation, this would correspond to setting for every sample n, the constrained label set to $\mathcal{Y}_n = \mathcal{Y}$. Please note that this is on the other side of the spectrum than the fully supervised scenario discussed in Sec. 3.1.1. The weakly-supervised setup is a middle ground which formally include the two other ones, and solely depends on the cardinality of the sets $\mathcal{Y}_n$.

3.1.4 Semi-supervised learning

The last scenario that we want to describe is the *semi-supervised* one. This one is a special case of the weakly-supervised setup in which for a given subset of the dataset, the sets $\mathcal{Y}_n$ are singletons. That means, on one part of the dataset we have access to full supervision while on the rest we don't. Let us suppose that we have full supervision for the first S elements of the dataset, so for $n = 1, \ldots, S$, $\mathcal{Y}_n = \{y_n\}$. In this setup, the optimization problem we try to solve is the following:

$$\min_{f \in \mathcal{F}} \frac{\alpha}{S} \sum_{n=1}^{S} \ell(f(x_n), y_n) + \frac{1-\alpha}{N-S} \sum_{n=S+1}^{N} \left[\min_{y \in \mathcal{Y}_n} \ell(f(x_n), y) \right], \tag{3.3}$$

where α is a weighting parameter allowing us to control the influence of labeled versus unlabeled data. In practice properly setting this weight is important and has a big impact on the final performance.

In all these scenarios there are mutliple choices for the set of functions $\mathcal{F}$, the loss ℓ and the label set $\mathcal{Y}_n$.

3.2 Clustering

In the previous section we have discussed different levels of supervision that one can have when learning discriminative models. In the unsupervised case however, the concept of learning class models becomes what we refer to as *clustering*. When clustering data points, the goal is to group them into clusters such that some criterion is minimized. Grouping points does not link them to any particular element of $\mathcal{Y}$ and the same objective value could be obtained by permuting the labels. In that case, labels only act as an equivalence class between data points: two points share the same label if they are in the same cluster.

In this section we will discuss clustering models and how we adapt them to suit our needs. We will start by discussing generative clustering models such as k-means, followed by a generic description of discriminative clustering. Most of our work relies on the DIFFRAC framework [Bach and Harchaoui, 2007] that we will describe

in detail in Sec. 3.2.3. eventually, we will describe the shift between clustering and classification and how do we impose constraints on a clustering method.

3.2.1 Generative clustering

We will first briefly discuss generative clustering, in particular the k-means clustering model. The goal is to partition data points into K clusters, and represent each cluster by the average of its members. The criterion that is optimized is the distance between the members of the cluster and the average representant. This problem is known to be NP-hard and approximate alternate optimization schemes are used, typically Lloyd's algorithm [Lloyd, 1982]. In the following we will try to give this model a formal description, that will allow us to draw parallels with the kind of models that we use in this thesis.

Suppose that we have N data points, indexed by n and that each point x_n lives in $\mathbb{R}^d$. We want to assign each point to one of the K clusters. For each point n, we define an assignment variable y_n in $\{0,1\}^K$, such that the $k-th$ entry of y_n is equal to one if the point n is assigned to the k-th cluster. For each cluster k we define a representant μ_k also living in $\mathbb{R}^d$. If we denote by M the matrix in $\mathbb{R}^{d\times K}$, whose columns are the μ_k, the product My_n gives us the representant of the cluster to which the n-th point is assigned to. Using this notation, the criterion formulated in the previous paragraph can be simply written as:

$$\sum_{n=1}^{N} \|x_n - My_n\|_2^2. \tag{3.4}$$

This criterion has to be minimized both in M and in all the y_n.

The optimization domain for the y_n is discrete, and even with a continuous domain, the objective is not jointly convex in the two variables. The classical approximate algorithm consists in doing an alternate optimization. For each n, we obtain the y_n by selecting the column in M which is the closest to x_n. The k-th column of M can in turn be updated by computing the average of all the x_n that are assigned to cluster k. In contrast, in the following section we will describe the discriminative clustering approach.

3.2.2 Discriminative clustering

The goal of discriminative clustering is to group data points into clusters that are easy to discriminate from each other. That means that each cluster is a class for which it is easy to have a discriminative model for. Using the notation from Sec. 3.1, the general formulation of a discriminative clustering problem can be written as follows:

$$\min_{f \in \mathcal{F}} \sum_{n=1}^{N} \left[\min_{y \in \mathcal{Y}} \ell(f(x_n), y) \right]. \tag{3.5}$$

For each data point n, we select a label in $\mathcal{Y}$ such that it will be easy to find a suitable classifier function f. Various choices for ℓ and $\mathcal{F}$ lead to different algorithms and results, some of which were discussed in Chapter 2. In the following section, we will describe in detail this model in the case where $\mathcal{F}$ is the set of linear (or affine) functions and ℓ is the square loss.

3.2.3 DIFFRAC

The DIFFRAC framework introduced in Bach and Harchaoui [2007], is a discriminative clustering model where the classifier is linear and the loss is the square loss. We will describe here in detal this framework as much of the work presented in this thesis is based upon it. As before, suppose that we are given N data points x_n in $\mathbb{R}^d$ indexed by n. Suppose for now that every point is assigned a label, that we represent by an indicator vector y_n in $\{0,1\}^K$. The label vector is binary and sums up to one: $y_n^\top \mathbf{1}_K = 1$, where $\mathbf{1}_K$ is the vector composed of ones in dimension K. We choose the classification function as follows:

$$f(x) = \operatorname*{arg\,max}_{k \in \{1,\dots,K\}} \; h_k(x), \tag{3.6}$$

where h_k is the linear scoring function for class k. For each class k, we define it as follows:

$$h_k(x) = w_k^\top x + b_k. \tag{3.7}$$

where w_k is a vector in $\mathbb{R}^d$ and b_k is a real-valued scalar. If we stack all the w_k into a $d \times K$ matrix W and the b_k into a K-dimensional vector b, we can define the vector scoring function $h : \mathbb{R}^d \to \mathbb{R}^K$:

$$h(x) = W^\top x + b. \tag{3.8}$$

Choosing the square loss, the function h can be found by solving the following optimization problem [Hastie et al., 2009]:

$$\min_{W \in \mathbb{R}^{d \times K},\ b \in \mathbb{R}^K} \quad \frac{1}{N} \sum_{n=1}^{N} \|y_n - W^\top x_n - b\|_2^2. \tag{3.9}$$

Let us define the design matrix X in $\mathbb{R}^{N \times d}$ whose n-th row is $x_n^\top$. Similarily, we define the label matrix Y in $\{0,1\}^{N \times K}$ whose n-th row is $y_n^\top$. Using this notation, the problem can be written in a more compact, matricial form as:

$$\min_{W \in \mathbb{R}^{d \times K},\ b \in \mathbb{R}^K} \quad \frac{1}{N} \|Y - XW - \mathbf{1}_N b^\top\|_F^2, \tag{3.10}$$

where $\|.\|_F$ denotes the matrix frobenius norm.

However, in the case of DIFFRAC, the label matrix Y is not known *a priori*, and is what we want to recover. Let us replace this matrix Y by a latent variable Z over which we will optimize our cost function. To simplify notation, we will denote by $\mathcal{Z}$ the set of binary matrices Z in dimensions $N \times K$ such that $Z\mathbf{1}_K = \mathbf{1}_N$ (sums up to one along rows). Then our discriminative clustering problem can be written as follows:

$$\min_{Z \in \mathcal{Z}} \quad \left[\min_{W \in \mathbb{R}^{d \times K},\ b \in \mathbb{R}^K} \quad \frac{1}{N} \|Z - XW - \mathbf{1}_N b^\top\|_F^2 + \lambda \|W\|_F^2 \right]. \tag{3.11}$$

The inner optimization over parameters W and b can be carried out in closed form and yields the following expressions:

$$W^* = (X^\top \Pi_N X + N\lambda I_d)^{-1} X^\top \Pi_N Z, \tag{3.12}$$

$$b^* = \frac{1}{N} (Y - XW^*)^\top \mathbf{1}_N, \tag{3.13}$$

where I_d is the identity matrix in dimension d and Π_N is the centering matrix in dimension N: $\Pi_N = I_N - \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^\top$. Using these optimal parameters, our problem is equivalent to:

$$\min_{Z \in \mathcal{Z}} \ \mathrm{Tr}\left(ZZ^\top A\right), \tag{3.14}$$

where A is a matrix that depends on the data X and on the regularization parameter λ:

$$A = \frac{1}{N}\Pi_N \left(I_N - X(X^\top \Pi_N X + N\lambda I_d)^{-1} X^\top\right) \Pi_N. \tag{3.15}$$

The problem formulated in Eq. 3.14 is a integer quadratic program. We can easily see that it does indeed correspond to a clustering problem as the cost function does not depend on the permutation of labels. Indeed, if we permute the columns of the matrix Z, the product $ZZ^\top$ remains identical and therefore yields the same objective. It does not matter for a given sample to which class it belongs to, only with which samples does it share this class.

This is even more general, the cost function is invariant to multiplication of Z by a orthogonal matrix in dimension K. Indeed, let O be such a matrix, then $ZO(ZO)^\top = ZZ^\top$. Given this observation, the authors propose to perform a lifting, and define a variable $M = ZZ^\top$ such that Z is in $\mathcal{Z}$. Let us call the set of such equivalence matrices as $\mathcal{M}$. We can have an explicit description for $\mathcal{M}$ as follows:

$$\mathcal{M} = \{M \in \{0,1\}^{N\times N},\ M = M^\top,\ M \succeq 0,\ \mathrm{rank}(M) \leq K,\ \mathrm{diag}(M) = \mathbf{1}_N\}. \tag{3.16}$$

Different optimization algorithms can be used to solve problem over this domain. The one presented in [Bach and Harchaoui, 2007] involves performing a relaxation, by changing some of the constraints; the resulting relaxed problem can be solved as an SDP. Further relaxations allow to write the optimization problem as an eigenvalue problem, which can be solved using standard algorithms. Joulin et al. [2010] propose to solve the same optimization problem following [Journée et al., 2010], a method for optimization on the manifold of low rank semidefinite matrices. However, in our case, working with the equivalence matrix is unpractical as it does not allow to impose

class specific constraints.

3.2.4 Adding constraints

In our work however, we do care about the "meaning" of classes and want to incorporate constraints on Z. We not only want to group faces of a given character in the same cluster, but we also want to make sure that they correspond to "Jack". Class-specific constraints are impossible to impose on the equivalence matrix M described in the previous section. The only kind of additional information that one can impose on M are *must link* and *must not link* constraints. As we will describe in the following chapters, we want to impose constraints that we derive from the text associated with video as well as structural constraints on Z. Therefore, we do not perform the lifting and solve the quadratic problem under constraints on Z directly.

The fact that we impose constraints on Z that are class specific, changes the problem from clustering to weakly-supervised classification. The constraints that we use usually do not present symmetries with respect to classes. For instance, in Chapter 4, we will describe how we model that at least one sample amongst a subset is of a given class. This kind of constraints applies to a given column of Z and therefore breaks the symmetries. The change from a totally unsupervised setup to a weakly-supervised one is very subtle however.

The framework that we developped in this thesis is very flexible and allows a wide range of applications. Using the same kind of cost functions, and solely changing the amount of constraints, one can move from clustering on one hand to fully supervised classification on the other. These constraints come in our case from structural assumption and meta-data. As soon as an assumption can be written as a linear equality or inequality constraint, it can be directly added to the framework. In the following chapter, we will describe how constraints can be derived from script data to guide person and action recognition in movies.

Chapter 4

Weakly supervised labeling of persons and actions in movies

Abstract

In this chapter, we address the problem of recognizing characters and actions in movies using weak supervision provided by scripts. We formulate the recognition as an optimization problem with a cost function composed of two terms, one that is based on discriminative clustering, and a normalized cuts one. We extract character/action pairs from the script and use them as constraints for the aforementioned problem. The corresponding optimization problem is a quadratic program under quadratic constraints. People in video are represented by automatically extracted and tracked faces together with corresponding motion features. We apply the proposed framework to the task of learning names of characters in movies and show significant improvements over previous methods used for this task. Second, we explore joint character/action constraints and show its advantage for weakly supervised action recognition. We demonstrate the performance of our model by localizing and recognizing characters and their actions in feature length movies Casablanca and American Beauty.

4.1 Introduction

As mentioned in the introduction, most of recent advances in computer vision rely on supervised learning. This is true in a variety of different tasks: image categorization, object detection, facial identification or action recognition. However, manual annotations are typically expensive to obtain and fully annotating a whole movie is a

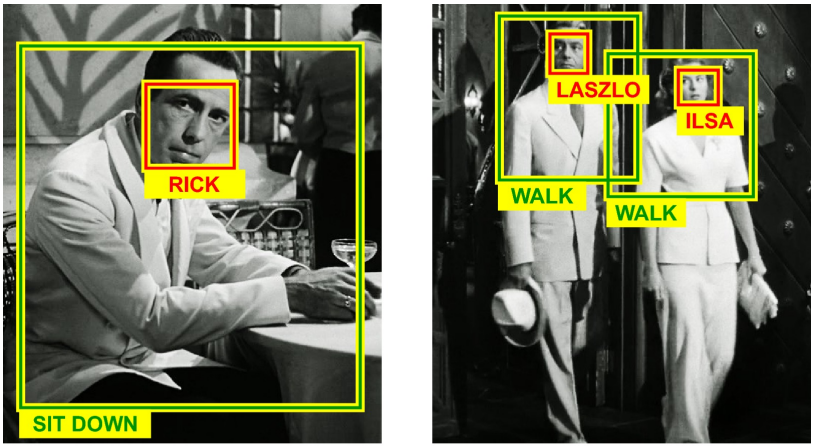


Figure 4-1 – Result of our automatic detection and identification of characters and their actions in the movie Casablanca. The automatically resolved correspondence between video and script is color-coded.

very tedious task. Many recent works propose to resort to crowdsourcing solutions in order to construct datasets. This on the other hand induces many challenges as the annotations obtained this way are often very noisy.

In this chapter we describe an alternative source of weak supervision, and present a model that takes the temporal imprecisions of this supervision into account. We specifically focus on the problem of character identification and action recognition in feature movies. As mentioned before, for video data such as movies, shooting scripts can easily be found on the web. Scripts contain a lot of information, including people's names, their actions, interactions and emotions, object properties, scene layouts and more. This kind of data has been used as supervision for automatic annotation of characters in TV series in the past [Cour et al., 2009, Ramanathan et al., 2014, Sivic et al., 2009, Tapaswi et al., 2012]. Scripts have also been used as supervisory signal for action recognition [Duchenne et al., 2009, Laptev et al., 2008, Marszalek et al., 2009].

Using textual descriptions to learn character and action models remains difficult due to the lack of precise temporal and spatial alignment. Usually, every scene description or dialogue has only a rough position in time. In practice, video scripts

provide no spatial localization of people or objects, and the temporal localization of events is often imprecise. Moreover, the description is in natural language, and needs to be encoded somehow into a convenient representation.

In this chapter, we observe that the description of a scene often involves objects, people and actions. Knowing that "Rick sits down" in a piece of video can help recognizing what a "sitting down" action is if we can localize "Rick", and vice versa (see Figure 4-1). The main point is that recognizing characters can be useful for learning models of infrequent actions (e.g. hand shaking). We follow this intuition and address *joint* weakly supervised recognition of characters and actions by exploiting their co-occurrence in movies. Contrary to prior work, we use character-action co-occurrences derived from scripts to constrain the weakly supervised optimization problem. An example output of our algorithm for a short movie clip and the associated script section is shown in Figure 4-1.

4.1.1 Contributions of this chapter.

The contributions presented in this chapter are threefold: **(i)** We consider a rich use of textual information, that goes beyond using character names. We constrain both person and action recognition using pairs of names and actions co-occurring in the text. **(ii)** We develop a clustering model jointly labeling actions and people, and incorporating text annotations as constraints. The corresponding optimization is formulated as a quadratic program under quadratic constraints. We propose a convex relaxation that we solve as a sequence of quadratic programs. **(iii)** We demonstrate the validity of the model on two feature-length movies and the corresponding movie scripts, and demonstrate improvements over earlier weakly supervised methods.

This chapter is based on an article presented at ICCV 2013 [Bojanowski et al., 2013], but includes some additions: first, we propose to include a normalized-cut cost based on descriptor similarity for face identification. Second, we introduce exclusion constraints which prevent the model from predicting the same character for two simultaneous tracks. We demonstrate an improvement in performance for character recognition for both of the proposed modifications.

4.2 Joint Model of characters and Actions

In this section, we formulate the problem of jointly recognizing characters and actions in movies. We pose this problem as the minimization of a quadratic cost function under constraints [Bach and Harchaoui, 2007, Joulin et al., 2010]. We incorporate text-based knowledge as a suitable set of constraints on the cluster membership matrices.

4.2.1 Notations and problem formulation

Let us suppose that we have two label sets $\mathcal{P}$ and $\mathcal{A}$. We define the sizes of these sets as $|\mathcal{P}| = P$ and $|\mathcal{A}| = A$. In practice, one can think of these label sets as person and action classes. Their construction will be discussed in the following sections.

Let us suppose that we are given N data points. You can think of these data points as person tracks in a movie. For every sample $n \in \{1, \ldots, N\}$ we are provided with two feature vectors ϕ_n and ψ_n in $\mathbb{R}^d$. We use the same dimension for both features to make the notations less cumbersome. We define the matrices Φ and Ψ in $\mathbb{R}^{d \times N}$ as the horizontal concatenation of all the individual vectors. You can think of these features as of face and action descriptors respectively.

Every sample n belongs to a class in $\mathcal{P}$ and a class in $\mathcal{A}$. The sample class membership is *a priori* unknown. For each sample we define a "person" latent variable z_n in $\{0,1\}^P$ such that $z_n^T \mathbf{1}_P = 1$. Similarly, we define an "action" variable t_n in $\{0,1\}^A$ such that $t_n^T \mathbf{1}_A = 1$. These variables indicate to which person and action class the sample is assigned. In the following, we will denote the p–th element of a vector z_n as z_{np}.

We define, Z as a $P \times N$ matrix with columns z_n and T as a $A \times N$ matrix with columns t_n. To simplify the notations, we will denote by $\mathcal{Z}$ and $\mathcal{T}$ the sets composed of all matrices Z and T respectively:

$$\mathcal{Z} = \{Z \in \{0,1\}^{P \times N}, \text{ such that } Z\mathbf{1}_P = \mathbf{1}_N\}$$
$$\mathcal{T} = \{T \in \{0,1\}^{A \times N}, \text{ such that } T\mathbf{1}_A = \mathbf{1}_N\}$$

Given weak supervision that we will describe later, our goal is to recover the latent variables Z and T. We formulate our problem as follows:

$$\begin{aligned} \min_{Z \in \mathcal{Z},\ T \in \mathcal{T}} \quad & D(Z,T) + G(Z,T), \\ \text{s.t.} \quad & C(Z,T) \leq 0, \end{aligned} \tag{4.1}$$

where D and G are cost functions that we will describe in section 4.2.2. The weak supervision we use is represented by the function C in the constraints and will be described in section 4.2.4.

4.2.2 Discriminative loss

The problem formulated in the previous section has a cost function composed of two terms D and G. We use a discriminative clustering cost for the definition of D. Let us denote by $\mathcal{F}$ the set of person classifiers $f : \mathbb{R}^d \to \mathbb{R}^P$ and by $\mathcal{G}$ the set of action classifiers $g : \mathbb{R}^d \to \mathbb{R}^A$. We denote by Ω a convex regularization function over the set of classifiers of interest: $\Omega : \mathcal{F} \times \mathcal{G} \to \mathbb{R}_+$.

Let us consider two multi-class loss functions:

$$\begin{aligned} \ell_P &: \{0,1\}^P \times \mathbb{R}^P \to \mathbb{R}_+, \\ \ell_A &: \{0,1\}^A \times \mathbb{R}^A \to \mathbb{R}_+. \end{aligned}$$

When supervision is available for data samples, and learning the classifiers is done by minimizing the empirical risk:

$$\min_{\substack{f \in \mathcal{F} \\ g \in \mathcal{G}}} \quad \frac{1}{N} \sum_{n=1}^{N} \ell_P(z_n, f(\phi_n)) + \frac{1}{N} \sum_{n=1}^{N} \ell_A(t_n, g(\psi_n)) + \Omega(f,g). \tag{4.2}$$

In our problem, however, the sample class memberships are unknown (z_n and t_n are not given). We will use the same cost function as for the empirical risk minimization,

but Z and T being variables. In the following, we use linear classifiers of the form:

$$f(\phi) = W_P^T \ \phi + b_P,$$
$$g(\psi) = W_A^T \ \psi + b_A,$$

where W_P is a matrix in $\mathbb{R}^{d\times P}$, b_P is a vector in $\mathbb{R}^P$, W_A is matrix in $\mathbb{R}^{d\times A}$ and b_A is a vector in $\mathbb{R}^A$. We use for losses ℓ_P and ℓ_A the squared Frobenius norms. We define the regularizer Ω by:

$$\Omega(f,g) = \lambda_P \|W_P\|_F^2 + \lambda_A \|W_A\|_F^2.$$

Our cost function becomes:

$$\begin{aligned} D(Z,T) = \min_{\substack{W_P,\ b_P \\ W_A,\ b_A}} \quad & \frac{1}{N}\sum_{n=1}^{N} \|z_n - W_P^T\phi_n - b_P\|_F^2 + \lambda_P\|W_P\|_F^2 \\ & + \frac{1}{N}\sum_{n=1}^{N} \|t_n - W_A^T\psi_n - b_A\|_F^2 + \lambda_A\|W_A\|_F^2. \end{aligned}$$

Using the notations from the previous section, this can also be rewritten in a more compact manner as:

$$\begin{aligned} D(Z,T) = \min_{\substack{W_P,\ b_P \\ W_A,\ b_A}} \quad & \frac{1}{N}\|Z - W_P^T\Phi - b_P\mathbf{1}^T\|_F^2 + \lambda_P\|W_P\|_F^2 \\ & + \frac{1}{N}\|T - W_A^T\Psi - b_A\mathbf{1}^T\|_F^2 + \lambda_A\|W_A\|_F^2. \end{aligned} \tag{4.3}$$

We get the optimum of the minimization problem with respect to W_P, b_p, W_A and b_A as the solution of a ridge regression problem. In particular, for fixed Z and T,

setting the gradients with respect to W_P, b_P, W_A and b_A to zero yields:

$$\begin{aligned} W_P &= (\Phi\Pi_N\Phi^T + N\lambda_P I_d)^{-1}\Phi\Pi_N Z^T, \\ b_P &= \frac{1}{N}(Z - W_P^T\Phi)\mathbf{1}, \\ W_A &= (\Psi\Pi_N\Psi^T + N\lambda_A I_d)^{-1}\Psi\Pi_N T^T, \\ b_A &= \frac{1}{N}(T - W_A^T\Psi)\mathbf{1}. \end{aligned}$$

Plugging these expressions back in the cost function yields:

$$D(Z,T) = \mathrm{Tr}\left(Z^T Z A\right) + \mathrm{Tr}\left(T^T T B\right), \tag{4.4}$$

where:

$$\begin{aligned} A &= \frac{1}{N}\Pi_N\left(I_N - \Phi^T\left(\Phi\Pi_N\Phi^T + N\lambda_P I_d\right)^{-1}\Phi\right)\Pi_N, \\ B &= \frac{1}{N}\Pi_N\left(I_N - \Psi^T\left(\Psi\Pi_N\Psi^T + N\lambda_A I_d\right)^{-1}\Psi\right)\Pi_N. \end{aligned}$$

Using kernels. Following Bach and Harchaoui [2007], we notice that A can be expressed only in terms of the Gram matrix $\Phi\Phi^T$. Therefore, we can use any kernel and replace occurrences of this Gram matrix by the kernel matrix K. Let us define a kernel for person identities that we will denote K_P and one for motion actions that we will denote K_A. This gives us the following updated expressions for A and B:

$$\begin{aligned} A &= \lambda_P\Pi_N\left(\Pi_N K_P\Pi_N + N\lambda_P I_N\right)^{-1}\Pi_N, \\ B &= \lambda_A\Pi_N\left(\Pi_N K_A\Pi_N + N\lambda_A I_N\right)^{-1}\Pi_N. \end{aligned} \tag{4.5}$$

4.2.3 Grouping term

In addition to the previously described discriminative term D, we propose to use a normalized-cut term. In the following we restrict ourselves to the person variable Z, but a similar cost can be defined for actions. To this end, we define a graph whose vertices correspond to person tracks and weighted edges encode similarity between

these nodes. The weights of the edges are encoded in an $N \times N$ matrix W such that:

$$W_{nn'} = \exp(-\gamma d(\phi_n, \phi_{n'})), \tag{4.6}$$

where d is a distance between the descriptors of the two tracks. We will discuss in details our choice for d in the experimental section. We then compute the normalized laplacian of this graph as:

$$L = I_N - S^{-\frac{1}{2}} W S^{-\frac{1}{2}}, \tag{4.7}$$

where:

$$S = \text{Diag}(W\mathbf{1}_N), \tag{4.8}$$

is the diagonal matrix whose diagonal corresponds to the row sum of the weight matrix W. Then, the normalized cut term of our cost function is defined as:

$$G(Z, T) = \text{Tr}(Z^\top L Z). \tag{4.9}$$

4.2.4 Constraints on latent variables

In our problem formulation, we propose to use task specific constraints that reduce the search space for Z and T by including information mined from scripts. Using these constraints allows us to add additional knowledge, and has two other important advantages. First, given our cost function, the minimizations with respect to Z and T are totally independent. Adding constraints that depend on both variables will allow us to tie them together. Second, both cost functions defined in this section are symmetric with respect to classes: as for any clustering problem, the objective is invariant to permutations of labels. Nothing really ties a specific cluster to a specific character in the movie. In order to fix this permutation we propose to use constraints on latent variables. We will describe in this section the different constraints that we can generate using the aligned scripts.

Mention Constraints

We want to constrain our problem by coupling person and action labels. We do so by using information mined from movie scripts. After aligning scripts with time-stamped subtitles as in [Everingham et al., 2006], we get a temporal location for every scene description and closed caption. We remind the reader that a scene description is a set of sentences that describe what is played on screen. A closed caption is a piece of speech, as pronounced by the specified character.

We make the assumption that if the text mentions *someone* doing *something*, this action should appear on screen. From the aforementioned data, we can extract person–action couples (p, a) to generate our constraints. Remember that p is an integer in $\{1, \ldots, P\}$ and a is in integer in $\{1, \ldots, A\}$.

We index the scene descriptions by an integer i in $\{1, \ldots, I\}$. Every scene description is a set of sentences associated with a beginning and end frame (f_b and f_e respectively). We search for person tracks that appear between frames f_b and f_e. We denote this subset of tracks by $\mathcal{N}_i$.

We process the sentences in the scene description i to detect person–action pairs. All occurrence of a subject–verb pair are denoted by a set Λ_i. Λ_i contains indexed (p, a) pairs. If the subject is a pronoun we denote the corresponding tuple as $(\varnothing, a)$.

For every scene description i, we can write the following set of equations:

$$\forall (p, a) \in \Lambda_i, \sum_{n \in \mathcal{N}_i} z_{np}\, t_{na} \geq 1, \tag{4.10}$$

$$\forall (\varnothing, a) \in \Lambda_i, \sum_{n \in \mathcal{N}_i} t_{na} \geq 1. \tag{4.11}$$

Constraints using the (p, a) pairs couple the two sub–problems. Please note however that the inequalities in Eq. (4.10) are non necessarily convex in general. The inequalities defined using $(\varnothing, a)$ generate linear constraints on the action latent variables.

Speaker Constraints

We can additionally constraint the problem using the subtitles. Indeed, if a character is speaking at a given time, it is likely he / she will appear in the surrounding video signal. Let us denote by $\mathfrak{C}$ the set of closed captions for a given movie. For every closed caption $\mathfrak{c}$ in $\mathfrak{C}$, we get the begin and end frame (f_b and f_e). The character that speaks in the closed caption $\mathfrak{c}$ will be called p. We select the set of shots that contains the time interval $[f_b, f_e]$ and find the person tracks that appear within. Let us denote $\mathcal{N}_\mathfrak{c}$ this set of tracks for the closed caption $\mathfrak{c}$.

For every closed caption $\mathfrak{c}$ in $\mathfrak{C}$, we can write the following constraint:

$$\sum_{n \in \mathcal{N}_\mathfrak{c}} z_{np} \geq 1.$$

This yields a set of linear constraints on the person latent variable.

Exclusion Constraints

We add another constraint to prevent our model from predicting the same character name for two overlapping tracks. It is reasonable to assume that two overlapping tracks are unlikely to be the same person. The only counter example would be in scenes involving mirrors. We will consider this specific case as an exception that can be handled by the slack variables presented in Sec. 4.2.5.

In order to implement this assumption as a constraint we need to find the set of overlapping tracks. We search for maximal groups of overlapping constraints. To find these maximal groups we first build the graph $G = (V, E)$ of overlaps between tracks. The nodes of this graph correspond to person tracks. We add an edge between two nodes v_1 and v_2 if the corresponding tracks overlap in time.

Finding all the maximal cliques in a graph is a classical NP-complete problem [Karp, 1972]. Instead, we first list the set of connected components in this graph. Because of shot boundaries, most connected components are of reasonable size. If a connected component contains less than 20 nodes, we use the Bron-Kerbosch algorithm [Bron and Kerbosch, 1973] to list all maximal cliques. Otherwise we limit

ourselves to listing all cliques of size 3.

Let us denote by $\mathscr{C}$ the set of cliques found using the aforementioned procedure. For every clique c in $\mathscr{C}$, for every label p in $\{1, \ldots, P\}$, we can write the following constraint:

$$\sum_{n \in c} z_{np} \leq 1 \tag{4.12}$$

This corresponds to a set of linear constraints on Z and can be easily added to our optimization problem. Please note however that the equivalent constraints for T do not hold. Indeed, people can perform the same action simultaneously (quite frequent for some classes: e.g. "Walk").

"Scene Constraints"

Another type of constraints was proposed by Ramanathan et al. [2014]. The authors observe that movies are usually divided into units called "scenes". A scene is a set of shots that take place in the same location and share a narrative thread. The movie scripts we use with our movies are provided with scene indications.

The scene constraints defined by Ramanathan et al. [2014] are the following: If a character name is never mentioned in a scene in the script, he or she never appears on screen in that scene. For a given scene s in the script, one has to find the corresponding beginning and end frames (f_b and f_e) in the video. Let $\mathcal{N}_s \subset \{1, \ldots, N\}$ be the set of person tracks appearing between f_b and f_e.

In the script, within a scene s, we search for characters that speak or are mentioned in a scene description. Let us call this set of characters $\mathcal{P}_s \subset \{1, \ldots, P\}$. Using the aforementioned sets, we can define the following set of constraints:

$$\forall\ s, \quad \forall\ n \in \mathcal{N}_s, \quad \forall\ p \notin \mathcal{P}_s, \quad z_{np} = 0. \tag{4.13}$$

These are simple linear equality constraints that can be added to our optimization problem. We do not use this kind of constraints in our model and only include their description for completeness.

External information

Our method allows the use of additional supervised data. For example, the cast of characters can be found, and images of the respective characters can be downloaded from the web. We can also get manually annotated videos corresponding to the action set $\mathcal{A}$. Supervision can be added to our model by simply appending the corresponding data points to the data matrix. We then constrain the corresponding latent variables to be equal to the provided annotations. This corresponds in the end to adding a set of linear equalities on the matrices Z and T.

Note however that we have to be careful and extract relevant descriptors for these additional examples. The features need to be comparable to those extracted from person tracks in the movie of interest. Indeed, the main application for this kind of constraints is to populate the dummy "background" class. For instance, for person identities, we compute face descriptors of face images randomly sampled from a public face figures dataset. However, the descriptors for face tracks in video and face detections in images differ in number.

4.2.5 Slack Variables

In practice, the constraints we have defined in the previous section can be erroneous. There are many reasons for that, including script mis–alignment, hypotheses not met *etc.* For instance, a person–action pair in a scene description may not have corresponding person tracks in the video. This can happen due to failures of automatic person detection and tracking.

To cope with these issues, we introduce slack variables allowing the constraints to be violated. We define a slack variable vector ξ. Suppose there are in total J different constraints. There is one ξ_j per constraint and therefore ξ is a real vector of length J. We add the slack to every constraint and add a quadratic penalization to our cost

function:

$$\min_{\substack{Z\in\mathcal{Z},T\in\mathcal{T},\\ \xi\in\mathbb{R}^J}} D(Z,T)+G(Z,T)+\kappa\,\xi^T\xi, \qquad \text{s.t. } C(Z,T)\leq 0.$$

4.3 Optimization

The problem we have formulated in the previous section is the minimization of a quadratic cost over a discrete set. This discrete optimization domain is defined by a set of quadratic or linear constraints, that are described in Sec. 4.2.4. In order to make the optimization of our problem tractable, we first relax the optimization domain. Then, since the domain is defined by quadratic constraints, we propose an alternating optimization scheme. We will describe in more details the two procedures in the following section.

4.3.1 Relaxation

Our optimization variables Z and T are defined as binary stochastic matrices. Solving quadratic optimization problems over these sets is known to be NP–hard [Schrijver, 2003]. We therefore propose to use a classical relaxation and replace $\mathcal{Z}$ and $\mathcal{T}$ by their convex hulls. These convex hulls (respectively $\overline{\mathcal{Z}}$ and $\overline{\mathcal{T}}$) correspond to the simplices $\mathcal{S}_N^P$ and $\mathcal{S}_N^A$:

$$\mathcal{S}_N^P=\{Z\in[0,1]^{N\times P},\ Z\mathbf{1}_P=\mathbf{1}_N\}, \tag{4.14}$$

$$\mathcal{S}_N^A=\{T\in[0,1]^{N\times A},\ T\mathbf{1}_A=\mathbf{1}_N\}. \tag{4.15}$$

By replacing the sets $\mathcal{Z}$ and $\mathcal{T}$ by their convex hulls, our problem becomes a continuous optimization problem.

4.3.2 Splitting the Optimization

As mentioned above, our optimization program minimizes a quadratic cost under several types of constraints. The cost function is a sum of two separate convex

quadratic functions of Z and T . The optimization domain $\mathcal{D}$ is defined by the intersection of sets $\overline{\mathcal{Z}}$ and $\overline{\mathcal{T}}$, linear equalities on Z and T separately, linear inequalities on Z and T separately and joint quadratic constraints on Z and T. Even though the domain is now continuous, it is not clear whether it is convex.

Quadratic constraints are non convex in general unless the constraint matrix is shown to be positive semidefinite. In our case, the matrices of the quadratic constraints are neither positive nor negative semidefinite. We observe that our relaxed optimization domain is defined as the intersection of potentially non–convex sets. Therefore, we cannot easily conclude on the convexity of $\mathcal{D}$ based on that. Because of that, we cannot simply use quadratically constraints quadratic program (QCQP) solvers. We propose to use a simple block-coordinate optimization scheme in order to solve our problem.

We alternate between optimizing in Z and T, fixing the value of the other one. Please note that we optimize over ξ on both occasions. At every iteration, the quadratic constraints become linear ones in the variable which is optimized. Therefore, each step boils down to the minimization of a quadratic function over a convex set defined by linear constraints. Each of these steps can be solved using an interior-point method. Many implementations are available online, and we perform our experiments using the MOSEK solver.

We start the algorithm by first freezing the T variable and optimizing over Z and ξ. We initialize T with the uniform assignment matrix $T = \frac{1}{A}\mathbf{1}_N\mathbf{1}_A^\top$.

4.3.3 Rounding

At the end of the optimization of the relaxed problem we have a real valued solution $(\hat{Z}, \hat{T})$. However, we want to obtain a feasible point of the initial, non–relaxed problem. The rounding procedure we propose is based on a geometric criterion: getting the feasible point closest to $\hat{Z}$ and $\hat{T}$ according to some distance. This way of rounding

the solution can be written as the following problem:

$$\min_{Z\in\mathcal{Z},\ T\in\mathcal{T}} \ \|Z-\hat{Z}\|_F^2 + \|T-\hat{T}\|_F^2, \qquad \text{s.t. } C(Z,T)\leq 0. \tag{4.16}$$

By expanding the norms, one observes that:

$$\|Z-\hat{Z}\|_F^2 = \|Z\|_F^2 + \|\hat{Z}\|_F^2 - 2\mathrm{Tr}(Z^\top \hat{Z}). \tag{4.17}$$

The two first terms are constant, because of the definition of $\mathcal{Z}$ and because $\hat{Z}$ is fixed. Therefore the problem described in Eq. (4.16) is equivalent to the following:

$$\min_{Z\in\mathcal{Z},\ T\in\mathcal{T}} \ -\mathrm{Tr}(Z^\top \hat{Z}) - \mathrm{Tr}(T^\top \hat{T}), \qquad \text{s.t. } C(Z,T)\leq 0. \tag{4.18}$$

However, this remains a complex integer programming problem with quadratic constraints. We propose to split the optimization problem into two and solve:

$$\min_{Z\in\mathcal{Z}} \ -\mathrm{Tr}(Z^\top \hat{Z}), \qquad \text{s.t. } C(Z,\hat{T})\leq 0. \tag{4.19}$$

Problem (4.19) is an integer linear program (ILP), which is known to be NP–hard [Schrijver, 2003]. Because of that, for the rounding procedure, we drop all the aforementioned constraints and simply solve:

$$\max_{Z\in\mathcal{Z}} \ \mathrm{Tr}(Z^\top \hat{Z}).$$

This in turn is solved exactly by independently taking the maximum value along each row of $\hat{Z}$. Please note that the constraints that defined the optimization domain in the previous section are dropped. This rounding procedure provided satisfactory performance in our experiments.

4.4 Relation to Diffrac [Bach and Harchaoui, 2007]

Our problem formulation in (4.4) is closely related to the discriminative clustering approach Diffrac [Bach and Harchaoui, 2007, Joulin et al., 2010]. When latent classes are treated equally, the minimization of a convex relaxation of (4.4) results in a trivial solution [Guo and Schuurmans, 2007]. To overcome this issue, one can perform a lifting and optimize (4.4) with respect to the equivalence matrix $M = ZZ^T$ instead (under a suitable set of constraints).

Working with M is problematic in our case since our constraints in (4.10) are defined on the elements of Z rather than on M. Class-dependent constraints in our case, however, break the symmetry in class labels, and enable (4.4) to be solved directly for Z. In practice we found that modifying the value of 1 to a larger constant on the right sides of inequalities (4.10) leads to a more stable solution of (4.4).

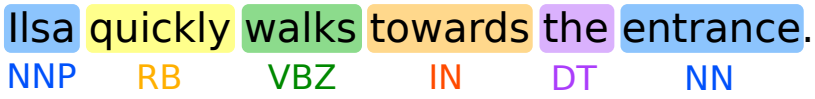
4.5 Features and Dataset

4.5.1 Text processing

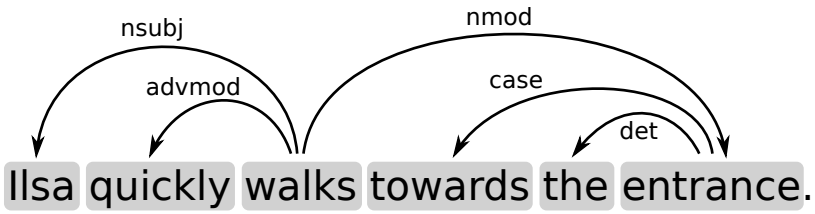
The method described in this chapter relies on a semantic representation of text. Indeed, we propose to exploit textual information to constrain our learning problem. To this end, we need to automatically extract `<NAME,ACTION>` pairs from natural language. There are many different ways one could do that, and we discuss some of these in this section.

The simplest way one can extract this kind of information from text is to find subjects and verbs. This can be done in an unstructured way using part-of-speech (POS) tagging. However, in a sentence there can be multiple nouns, and multiple verbs, yielding unwanted uncertainties. Some heuristic may be needed to pair nouns to verbs which is a problem on its own.

Another straightforward way is to use the root and direct dependencies of a dependency tree. This reliably provides the main subject and verb for most sentences. However, even though the detection should be correct most of the time, verbs do not



(a) Part of speech tags



(b) Dependency tree

Figure 4-2 – Illustration of the output of a Part-of-speech tagger (up) and a dependency tree parser (down). The part-of-speech tags are as follows: NNP - proper noun, RB - adverb, VBZ - verb in present tense, IN - preposition, DT - determiner, NN - common noun. The dependencies are: nsubj - nominal subject, advmod - adverbial modifier, nmod - nominal modifier, case - case marking, det - determiner.

correspond to action classes. Verbs could be grouped using some semantic similarity measure from resources such as VerbNet [Kipper et al., 2008].

In this work, we propose to extract person-action pairs from text using a semantic role labeling parser. Semantic role labeling consists of identifying arguments (agent, instrument, manner, cause) of a stereotypical situation to a predicate (for example a verb). Intuitively, this amounts to answering questions such as: "Who" does "What", "When", "Where" and "Why"? We define events as entries from the FrameNet database. This database defines so called "frames" which correspond to templates of an event. A frame is typically composed of subjects, relations, participants, objects *etc.* A few sample frames from FrameNet are given for illustration in Table 4.1.

Detecting these frames in natural language is a hard problem, but several statistical parsers are available on-line. In our work we use SEMAFOR [Das et al., 2012]. This is a statistical model trained on the annotated data associated with FrameNet. The performance of this model heavily relies on the quality of the manual annotations, which for some frames can be very scarce. We limit our experiments to two simple frames that occured often enough in the movies we considered and that have an associated agent: "ChangePosture" and "SelfMotion". From each detected occur-

Manipulation	The words in this frame describe the manipulation of an **Entity** by an **Agent**.
Losing	This frame describes a situation in which an **Owner** loses his or her **Possession**. The **Owner** may have failed to keep ownership of their **Possession** (i.e. robbery or gambling) or may have unintentionally misplaced their **Possession**.
Self_motion	The **Self_mover**, a living being, moves under its own direction along a **Path**. Alternatively or in addition to **Path**, an **Area**, **Direction**, **Source**, or **Goal** for the movement may be mentioned.
Being_wet	An **Item** is in a state of wetness with the possibility of the wetting **Liquid** being mentioned, along with the **Degree** of wetness.

Table 4.1 – A few sample frames as defined in the FrameNet project [Baker et al., 1998]. Every frame comes with *lexical units* that evoke the frame and a set of *frame elements*. In the examples provided here we put the core frame elements (the most important ones) in bold. For instance, lexical units for the frame Manipulation include the following verbs: caress, grab, hold, kiss, pull *etc...*

rence of the frame in the text we use the "agent" and the "target verb" as the name and action pair.

4.5.2 Video features

The aim here is to design a representation of video that can be related to the `<person,action>` structure extracted from text. We do so by extracting tracks of people from video. Each track is then represented by face appearance features to capture identity and body motion features to represent the action. See Fig. 4-3.

To extract person tracks, we compute face tracks and extrapolate a body bounding box from these. We run the multi–view face detector of [Zhu and Ramanan, 2012] and associate detections across frames using point tracks in a similar manner to what was proposed by Everingham et al. [2006], Sivic et al. [2009]. To represent faces we follow the work of Sivic et al. [2009]: we extract facial features and rectify each face into a canonical frame using a similarity transformation. We re-compute facial feature positions in the rectified image and extract SIFT descriptors at multiple scales from each facial landmark. Finally, each track is represented by the set of descriptors, one

for each landmark, for each face in the track.

To represent actions, we compute bag-of-features on dense trajectories [Wang et al., 2011] extracted from each person track. We take the trajectories that fall into the spatio-temporal volume defined by the upper-body bounding box in each frame. As mentioned before, the upper-body bounding box is defined here by simply extrapolating the face bounding-box using a hand-crafted linear transformation. This way, we are sure that for every face we always have a corresponding upper-body region. Also, it allows us to get rid of the face to body association problem which can be a problem on its own.

Please note that our discriminative cost function allows the use of kernels. For face tracks, we follow Sivic et al. [2009] and use the sum of "min-min kernels" computed separately for each facial feature as well as frontal and profile faces. This results in a total of 38 face track kernels (24 for frontal features and 14 for profile features) that are summed with uniform weights. We do not optimize the weights of the kernel mixture as the average has been shown to work well in practice. For motion descriptors, following common practice, we use the exponentiated chi-square kernel [Wang et al., 2011].

4.5.3 Dataset

We report results for our method on two full-length feature movies, Casablanca and American Beauty. We process both movies and extract person tracks and associated descriptors (as described in the previous section). We discard person tracks that have unreliable facial features, based on the landmark localization score. For Casablanca, this yields 1,273 person tracks containing 124,423 face detections. For American Beauty we use 1,330 person tracks containing 131,741 face detections. Please note that our evaluation is person–track dependent: we suppose that undetected faces do not exist. The ground truth does not cover faces that were not detected or dropped due to poor facial features.

By processing the corresponding movie scripts, we extract 17 names for the main characters in Casablanca and 11 names for the main characters in American Beauty.

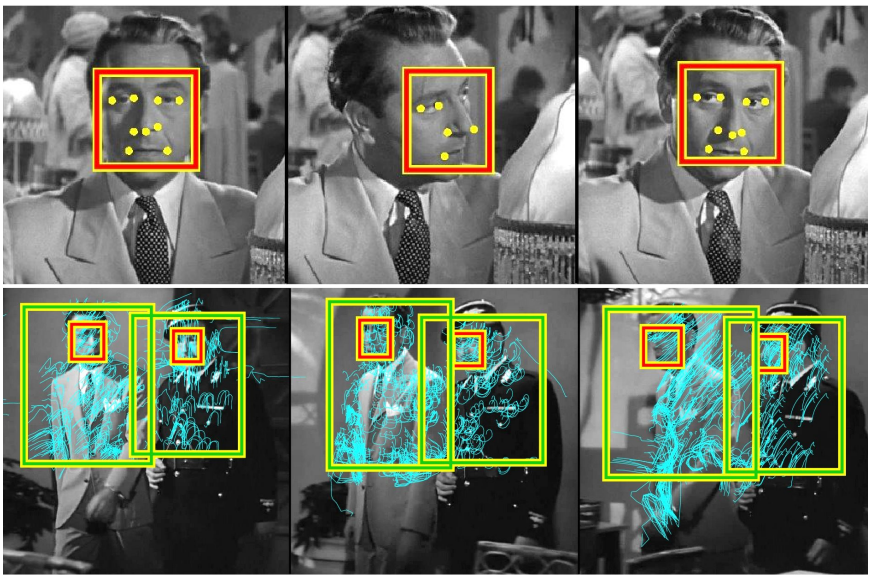


Figure 4-3 – **Representing video.** Top: face track together with extracted facial features. Bottom: Motion features based on dense point trajectories extracted from tracked upper body bounding boxes.

For each movie we select two most frequent action classes (frames), *i.e.*, *walking, sit down* for Casablanca and *walking, open door* for American Beauty. For Casablanca we obtain 42 action/name pairs and 359 occurrences of names with no associated actions. For American Beauty the corresponding numbers are 31 and 330, respectively.

To explicitly model non-named characters in the movie (side characters and extras) as well as unnamed action classes, we introduce an additional "background" class for both faces and actions. However, our model lacks a good definition of what the background class exactly is. In order to add some knowledge about the background, we collect background examples as follows. For faces, we collect 500 additional random faces from the Labeled Faces In The Wild dataset [Huang et al., 2007]. For actions, we randomly sample 500 person tracks from the Hollywood2 dataset [Marszalek et al., 2009] using the corresponding movie scripts to discard actions considered in this work. For all "background" samples, we constrain latent variables to take values corresponding to the "background" class. We have found that including this additional data helps resolving confusion in label assignment for our target classes.

4.6 Experiments

In this section we experimentally demonstrate the benefits of the proposed approach. We first test the sensitivity to parameter choices in a controlled character identification setup. Second, we show that even for learning names alone (without actions), the proposed method outperforms other state-of-the-art weakly supervised learning techniques designed for the same task. Finally, we demonstrate benefits of learning names and actions jointly compared to resolving both tasks independently.

4.6.1 Learning names: controlled set-up

Here we assess the sensitivity of the proposed method to the following four important parameters: the number of bags $|I|$, the number of classes P, the number of samples per bag $|\mathcal{N}_i|$ and the number of annotations per bag $|\Lambda_i|$. We will use real data – 1,273 face tracks and their descriptors from the movie Casablanca – but group the tracks into bags in a controlled manner. Each track is labeled with a ground truth name from the set of 18 main characters (or other). To create each bag, we first sample a track from a uniform distribution over characters and then complete the bag with up to $|\mathcal{N}_i|$ tracks by randomly sampling tracks according to the true distribution of the characters in the movie. Each bag is annotated according to the first sample. Given this data, we solve the sub-problem related to faces, *i.e.*, no joint action labels are used in this experiment.

As discussed in Section 4.2, each face track is assigned to a class by maximizing the rows of Z. Following the work of Everingham et al. [2006], Sivic et al. [2009] we measure performance by plotting a curve of per-sample accuracy vs. proportion of labeled tracks. Ideally, the accuracy would be one for all confidence values, but in practice the accuracy drops for samples with lower confidence. We illustrate results for different bag layouts in Figure 4-4.

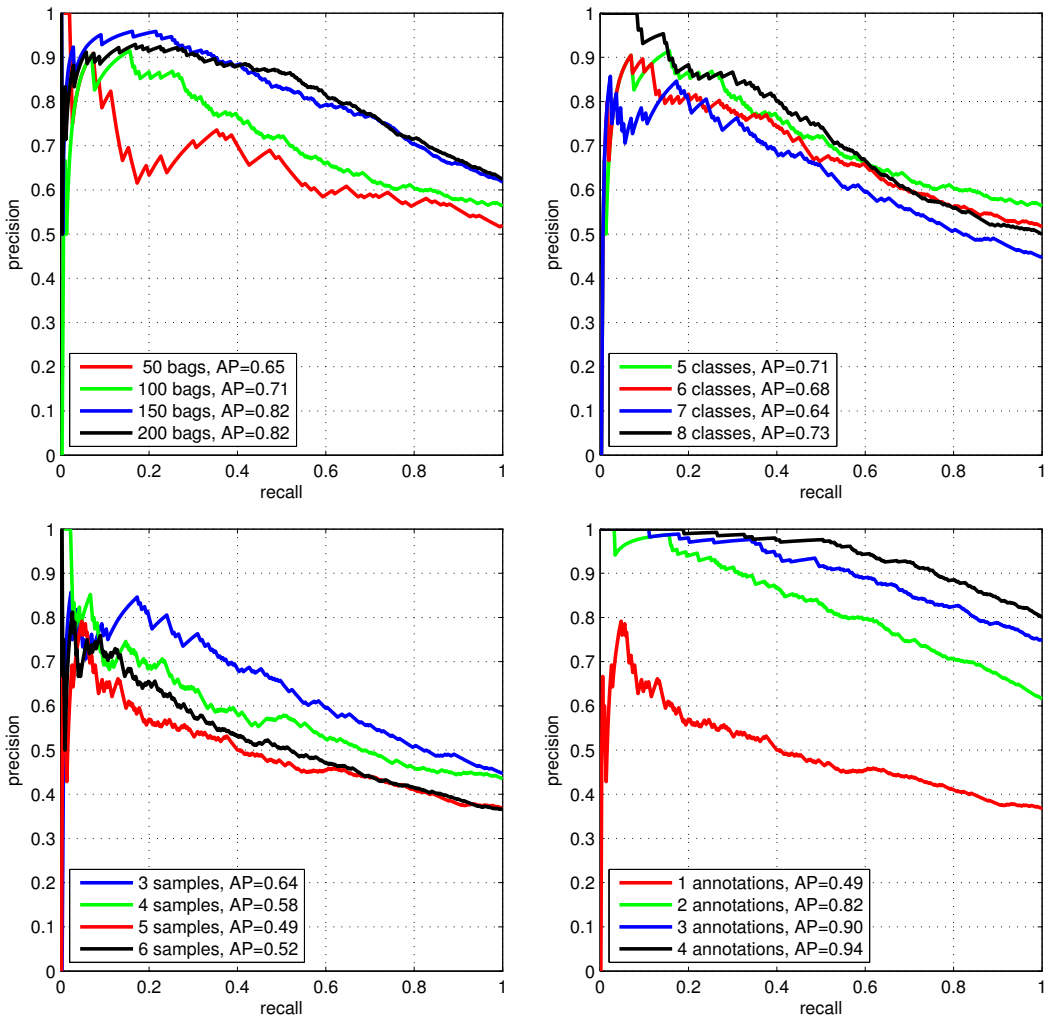
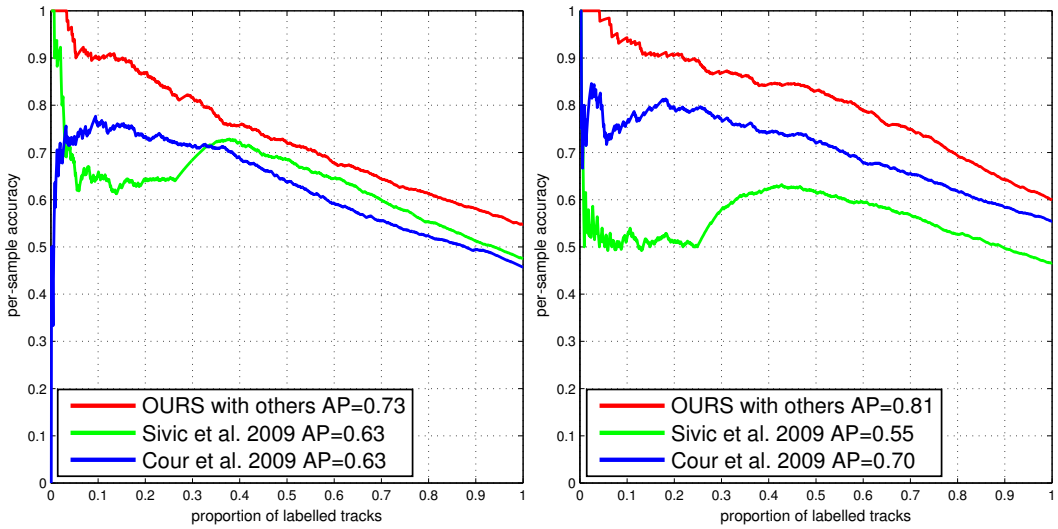


Figure 4-4 – **Performance for different bag layouts in a controlled set-up. (a)** First, we vary the number of bags while fixing 3 samples and 1 annotation per bag, and the number of classes to 5. As expected, performance improves with more bags. **(b)** Keeping 150 bags in total, we increase the number of classes. The effects of this modification are mixed. By adding more classes, the problem is harder but the per bag confusion is smaller. **(c)** Keeping 7 classes, we increase the number of samples per bag showing that more samples per bag increase confusion resulting in a lower performance. **(d)** Keeping 5 samples per bag, we increase the number of annotations per bag, clearly showing the benefits of having more annotations.

4.6.2 Comparison with other weakly supervised methods

Here we compare our method with other weakly supervised face identification approaches. We use the code adapted from Sivic et al. [2009] and an on-line available implementation of the method described by Cour et al. [2009]. We run all methods on 1,273 face tracks from Casablanca and 1330 face tracks from American Beauty using noisy name annotations obtained from movie scripts. To have a fair comparison, no action labels are used. While Cour et al. [2009] and Sivic et al. [2009] have been evaluated on television series, here we address a more challenging setup of full-length movies. First, the training data within a film is limited as it is not possible to harvest face tracks across multiple episodes as in TV series. Second, the cast of characters in a film is often larger than in TV series with many additional extras. Third, films often employ a wider set of cinematographic techniques compared to often simpler structure of a TV show with many close-ups and "shot-reverse shot" dialogues.

Comparative results for the two movies in Figure 4-5 demonstrate the superior performance of our method. The lower performance of Sivic et al. [2009] can be explained by the fact that it relies on visual speaker identification and lip motion estimation. While our adaptation of the code obtained from Sivic et al. [2009] works well on the data from their paper, we have found that the speaker detection achieves only 64.2% and 50.2% accuracy (with about 25% speaker labeled tracks) on Casablanca and American Beauty, respectively. The lower accuracy, compared to the accuracy of more than 80% on the TV series data from Sivic et al. [2009], could possibly be due to the challenging illumination conditions with strong shadows present in the two films. The approach of Cour et al. [2009] assumes that correct labels are included into the set of "ambiguous" labels. This assumption is often violated in movies as side characters and extras are often not mentioned in the script. In contrast, our approach suffers less from this problem since (a) it can handle multiple annotations for bags of multiple tracks and (b) the noise in labels and person detections is explicitly modeled using slack variables.



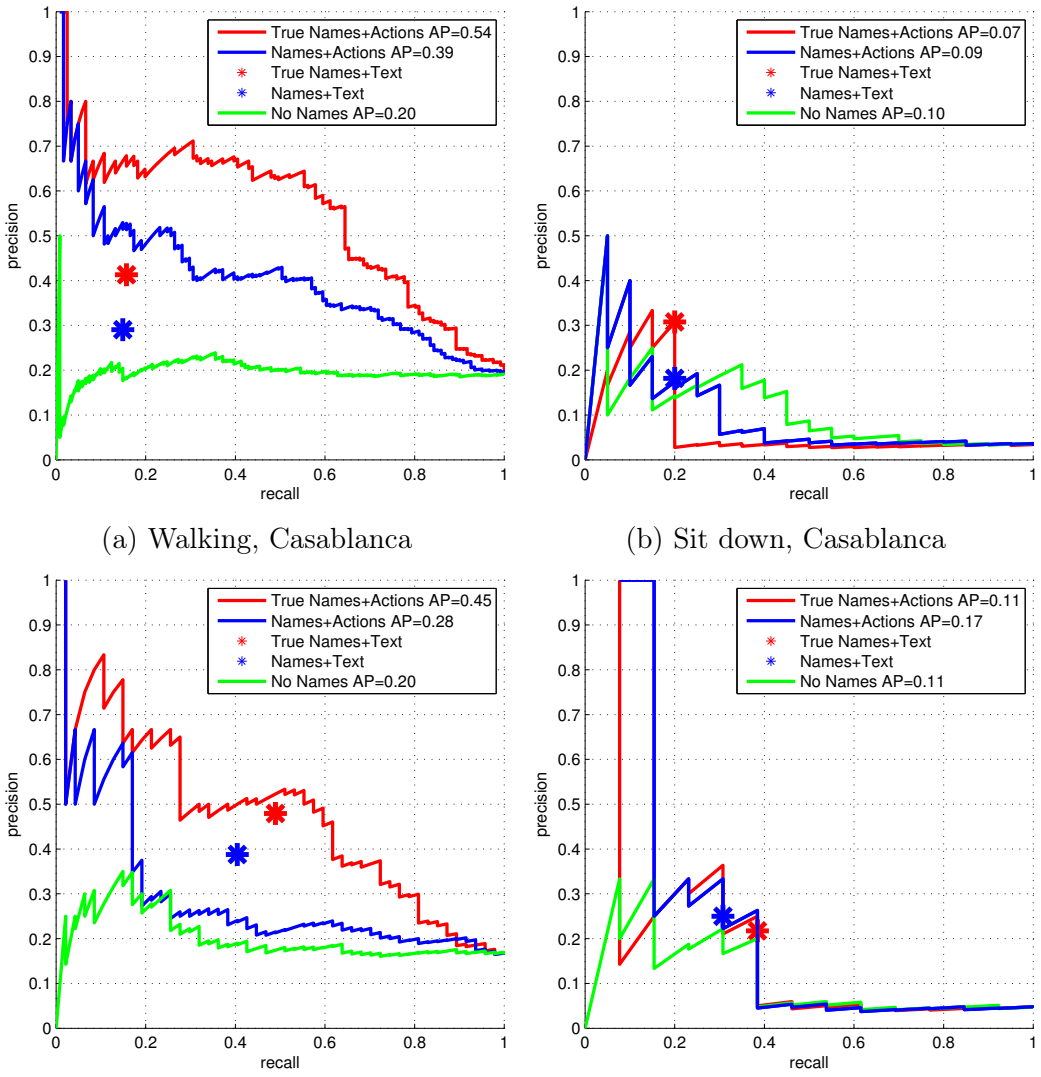
(a) Casablanca (b) American Beauty

Figure 4-5 – Results of automatic person naming in movies. Our method is compared with weakly supervised face identification approaches of Cour et al. [2009] and Sivic et al. [2009].

4.6.3 Learning names and actions

We next evaluate benefits of learning names and actions jointly. This is achieved by first learning the name assignments Z for all tracks. The name assignments are then fixed and used as additional constraints when learning the likely action assignments T for each track. While this procedure can be iterated to improve the assignment of character names with the help of estimated action labels, we found that the optimization converges after the first iteration.

The distribution of action classes in our data is heavily unbalanced with the "background" class corresponding to more than 78% of person tracks. We therefore evaluate the labeling of each target action in each movie using a standard one-vs-all action precision-recall measure. We compare the following methods: **Names+Actions** corresponds to our proposed method of learning person names and actions jointly. **No Names** uses constraints on actions only without considering joint constraints on actions and names. **True Names+Actions** uses the ground truth person names as constraints on actions instead of the automatic name assignment. This provides an upper bound on the action classification performance provided perfect assignment



(a) Walking, Casablanca

(b) Sit down, Casablanca

(c) Walking, American Beauty

(d) Open door, American Beauty

Figure 4-6 – Results of action labeling in movies Casablanca and American Beauty. See Section 4.6.3 for more details.

of person names. Finally, we evaluate two "dummy" baselines which blindly assign action labels based on person names and person-action pairs obtained from scripts. The purpose of these baselines is to verify that visual action classification improves the performance. **Names+Text** learns face assignments for each person track and assigns action labels using person-action pairs. **True Names+Text** assigns action labels based on person-action pairs and ground truth person names. This baseline, hence, does not "look" at image pixels at all. Note that the last two baselines produce a single point on the precision-recall plot as no confidence values are available when transferring action labels from scripts.

Precision-recall plots for the target action classes in two movies are shown in Figure 4-6. We first observe that our full method (blue curves) outperforms the weakly supervised learning of actions only (green curves) in most of the cases. This shows the benefit of learning actions and names jointly. As expected, action classification can be further improved using ground truth for name assignments (red curves).

For the frequent action *walking* for which many person-action constraints are available in scripts, automatic person naming in our method provides a large benefit. However, even with ground truth face assignments the action classification performance is not perfect (True Names+Actions). This is likely due to two reasons. First, the ambiguity in the weak supervision is not reduced to zero as a single character may do several different actions in a single clip (bag). Second, the current action representation could be significantly improved.

Recognizing less frequent actions *sit down* and *open door* appears to be more difficult. While several examples are ranked high, all methods suffer from a small number of available person-action constraints. In addition, a significant portion of these constraints is incorrect. Incorrect constraints often occur due to the failure of face detection as characters often turn away from the camera when sitting down and opening doors. To explicitly quantify the loss due to failures of automatic person tracking, we have manually annotated person tracks in the movie Casablanca. The performance of our full method is significantly improved when run on correct person tracks yielding AP=0.36 and AP=0.63 for the *sit down* and walk actions, respectively.

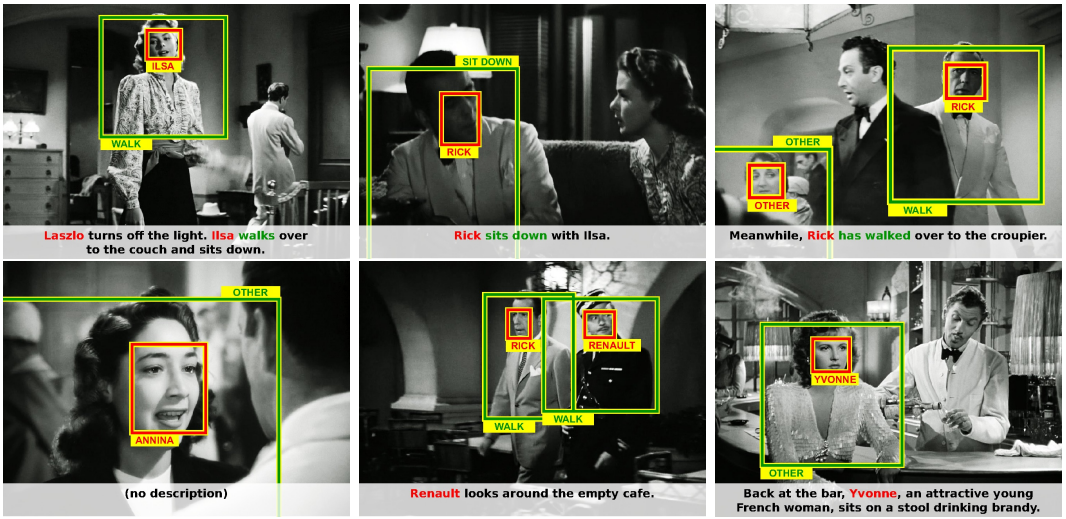


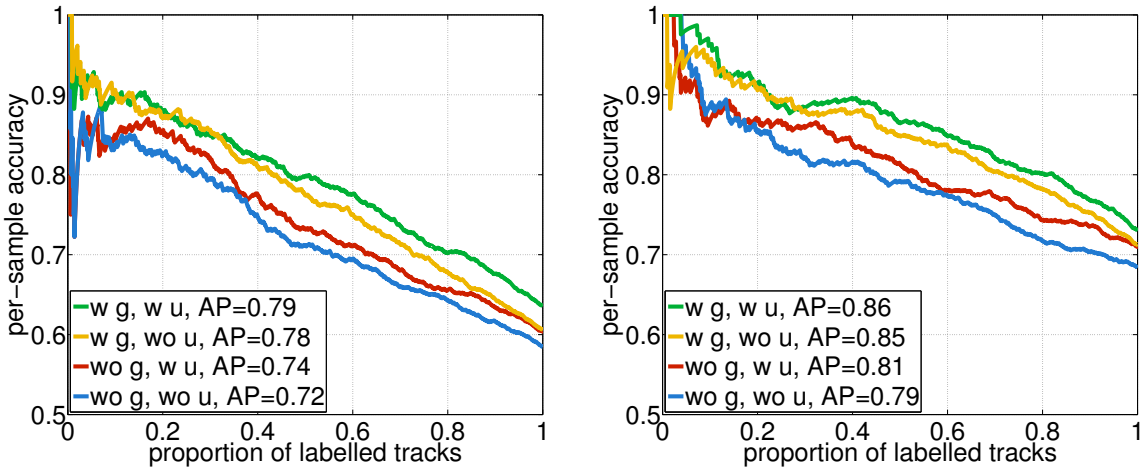
Figure 4-7 – **Examples of automatically assigned names and actions in the movie Casablanca. Top row:** Correct name and action assignments for tracks that have an character/action constraint in the script. **Bottom row:** Correct name and action assignments for tracks that do not have a corresponding constraint in the script, but are still correctly classified. Note that even very infrequent characters are correctly classified (Annina and Yvonne).

This emphasizes the need for better automatic person detection and tracking methods. Qualitative results for automatic labeling names of characters and actions using our method (Names+Actions) are illustrated in Fig. 4-7.

We show sample qualitative results obtained on the movie Casablanca in Fig. 4-7.

4.6.4 Improvements

We propose in this thesis two main improvements to the model which was first proposed in [Bojanowski et al., 2013]: adding a grouping term and uniqueness constraints. We want to experimentaly validate these improvements by showing that they yield better performance for the task of person recognition. To this end we run our algorithm, and limit ourselves, as in Sec. 4.6.2, to the face recognition part. We optimize the model with and without using both the grouping term and the exclusion constraints. We use exactly the same setup and performance measure as in the aforementioned section. Empirical results computed for both Casablanca and American Beauty are presented in Fig. 4-8.



(a) Casablanca (b) American Beauty

Figure 4-8 – Evaluation of the two model improvements introduced in this thesis.

We observe a significant improvement due to the two model modifications that we propose. For both movies, adding a grouping term and using exclusion constraints boosts the performance by 7%. It is interesting to notice that most of the improvement is due to the use of a grouping term. However, both modifications contribute to the higher average precision, as we see by looking at partially stripped-down versions of the model. When using the grouping term, the exclusions constraints seem to contribute to performance at high recall (propotion of labeled tracks).

4.7 Conclusion and future work

In this chapter, we have presented a new weakly supervised model to jointly recognize actions and characters in video. We have shown that the model can be apllied on feature-length movies together with their shooting scripts. We demonstrated a significant improvement over other weakly supervised methods for person recognition in movies with associated descriptions. Our experiments on action recognition show the benefits of recognizing the agent of the action. We believe that this kind of joint representation will help automatic video understanding, both at the level of local action models and global plot understanding.

The main limitation of this work is that the vocabulary of actions is very limited. In

order to move towards larger vocabularies, and higher number of samples, one needs to work on many more movies. As actions are shared across movies, applying the model over multiple movies simultaneously opens-up the possibility of automatically learning classifiers for a large vocabulary of action classes.

Chapter 5

A convex relaxation and efficient algorithm for aligning video and text

Abstract

We present a discriminative model for aligning a video with its textual description. We propose to learn an affine map between the feature representation of the two modalities with a latent alignment. Using a square loss function allows us to invert the problem and obtain a cost that only depends on the alignment variable. The aforementioned map remains implicit and can be recovered at any time given the optimal correspondence. Our model is trained using an efficient algorithm which exploits the temporal structure of the problem. The performance of our approach is evaluated on three different datasets, allowing us to assess the importance of all our design choices.

5.1 Introduction

Much progress on joint models for visual and textual data has been achieved in the last few years. Automatic image annotation has been intensively studied in the last decade and many different models have been proposed. Recent breakthroughs in image captioning are due to advances in deep image and text representations. These on the other hand are tightly tied to modern computational power and large image datasets now available for training. The construction of large manually annotated datasets can prove costly and time consuming.

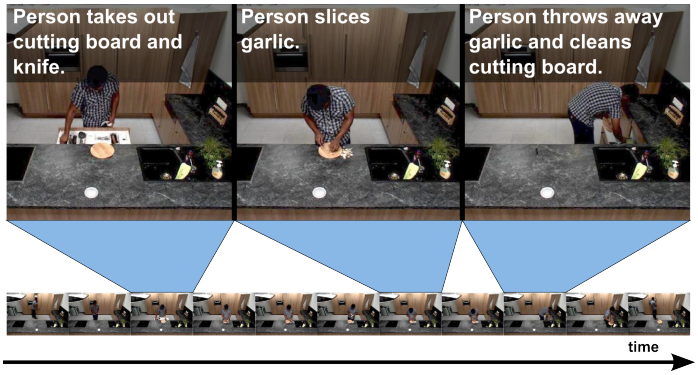


Figure 5-1 – Illustration of the problem addressed in this chapter. Given a video as a stream of frames and a text as a stream of sentences, we want to recover the temporal correspondence between frames and sentences. The problem is weakly supervised as we know that the assignment needs to respect the temporal order constraint.

Recently, several models have been proposed for automatic video description. Data for training joint video and text models is quite abundant and easy to find. Indeed, movies are often available along with their shooting scripts, sport events are associated with commentaries, and online "do it yourself" videos are often paired with a step-by-step recipies. The textual descriptions found in such paired text/video data precisely describe events that are happening on the screen. Manually corrected datasets composed of videos with textual descriptions have been proposed and have been used to train joint models.

However, exploiting such paired textual and video data is hard for several reasons. First of all, the descriptions are not always perfect and present a significant amount of noise. For instance, shooting scripts do not always correspond to what was shown in the final movie due to editing or actor's performance. In "how to" videos, the narrator sometimes anticipates future events instead of describing what he is doing at a given time. Second, there is a strong uncertainty in the temporal and spatial correspondence between the text and the video. As for image captioning, there is no spatial relation between nouns and objects, and additional uncertainty comes from the temporal dimension.

In this work, we cope with this temporal uncertainty and build a joint text and video model capable of recovering the temporal correspondence between sentences

and frames. We propose a model that learns a mapping between video and textual features while finding the optimal correspondence. In our experiments we only use global sentence and frame representations. We do not try to spatially localize parts of the video that correspond to a given word.

This chapter is structured as follows: In Sec. 5.2 we explain the problem setting and describe the proposed model. We also review the variable parametrization, the main cost function and additional priors that we propose. We present the proposed optimization algorithm in Sec. 5.3. That includes the convex relaxation, the Frank-Wolfe algorithm and the rounding procedure that we use. In Sec. 5.4 we briefly describe how we handle the use of additional supervised data. Finally, in Sec. 5.5 we present the empirical evaluation of the proposed model.

Contributions. This chapter makes the following contributions: **(i)** We present a general model for aligning two streams of data, **(ii)** we propose a convex relaxation for our problem and an efficient algorithm to solve the relaxation, **(iii)** the performance of the proposed model is evaluated on three different dataset. We analyze the influence of our design choices on the final performance. This chapter is based on two previously published works: [Bojanowski et al., 2014] and [Bojanowski et al., 2015]. Here we combine and extend the models found in these two publications. We also include additional evaluation with experiments carried out in a controlled setup on a toy dataset.

5.2 Proposed model

5.2.1 Problem statement and approach

Let us assume that we are given a data stream, associated with two modalities, represented by temporally ordered features ϕ_i $(i = 1, \ldots, I)$ and ψ_j $(j = 1, \ldots, J)$ respectively. In both cases, the indices respect the temporal order, that is, ϕ_i occurs before $\phi_{i'}$ if $i < i'$ and, likewise, ψ_j occurs before $\psi_{j'}$ if $j < j'$. We also write more compactly $\Phi = [\phi_1, \ldots, \phi_I]$ and $\Psi = [\psi_1, \ldots, \psi_J]$ where Φ and Ψ are respectively

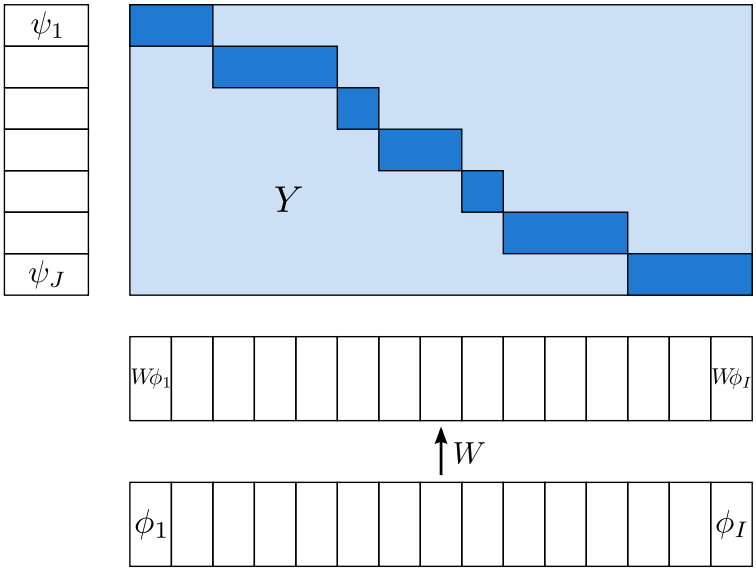


Figure 5-2 – Illustration of the notations and model used in this chapter. The video features Φ are mapped to the same space as text features using the map W. The temporal alignment of video and text features is encoded by the assignment matrix Y. Light blue entries in Y are zeros, dark blue entries are ones. See text for more details.

$D \times I$ and $E \times J$ matrices.

In the context of video to text alignment, Φ is a description of the video signal, made up of I temporal intervals while Ψ is a textual description, composed of J sentences (or, more generally, text fragments). Every ψ_j is the representation of sentence j, for instance its term-frequency representation. If applied to videos with a corresponding sequence of labels, Ψ is the sequence of labels, composed of J elements. Then every ψ_j is simply the indicator vector of the j-th class in the sequence. We focus in this chapter on video to text alignment, but our model is general and can be applied to other types of sequential data (biology, speech, music, *etc.*). In the rest of the chapter, except of course in the experimental section, we consider the abstract problem, considering two generic data stream modalities.

Our goal is to map every element i in $\{1, \ldots, I\}$ to an element j in $\{1, \ldots, J\}$. At the same time, we also want to learn a linear map between the two feature spaces, parametrized by W in $\mathbb{R}^{E \times D}$ [1]. If the element i is assigned to an element j, we want

1. As usual, we actually want an affine map. We will provide more details bout that in Sec. 5.2.2. For the time being, please assume that a constant row is added to Φ.

to find W such that $\psi_j \approx W\phi_i$, and the order of the two streams is preserved. If we encode the assignments in a binary matrix Y, which must of course satisfy a number of constraints to preserve the temporal order, this can be written in matrix form as: $\Psi Y \approx W\Phi$. The notations introduced here are illustrated in Fig. 5-2. The precise definition of the matrix Y will be provided in the following section.

5.2.2 Basic model

Let us begin by defining a binary *assignment matrix* Y in $\{0,1\}^{J\times I}$. The entry Y_{ji} is equal to one if i is assigned to j and zero otherwise. For Y to encode a mapping, it must satisfy the constraint

$$\forall i,\ \sum_{j=1}^{J} Y_{ji} = 1, \tag{5.1}$$

which implies that every element i is assigned to exactly one element j.

Our main assumption is that the temporal ordering of the two modalities is preserved in the assignment. Therefore, if the element i is assigned to j, then $i+1$ can only be assigned to j' such that $j' \geq j$. Since we don't want to skip any element j, the map has to be subjective, and therefore $i+1$ can only be assigned to j or $j+1$. In the following, we will denote by $\mathcal{Y}$ the set of matrices Y that satisfy this property.

Given Y, ΨY is a matrix whose i-th column is ψ_j if i is mapped to j in Y (Fig. 5-3). We then measure the discrepancy between ΨY and $W\Phi$ using the squared L_2 loss. Using an L_2 regularizer for W, our learning problem can be written as:

$$\min_{Y\in\mathcal{Y}} \min_{W\in\mathbb{R}^{E\times D}} \frac{1}{2I}\|\Psi Y - W\Phi\|_F^2 + \frac{\lambda}{2}\|W\|_F^2. \tag{5.2}$$

When Y is fixed, this is a ridge regression problem; In particular, we can rewrite (5.2) as:

$$\min_{Y\in\mathcal{Y}} q(Y), \tag{5.3}$$

where $q : \mathcal{Y} \to \mathbb{R}$ is defined for all Y in $\mathcal{Y}$ by:

$$q(Y) = \min_{W\in\mathbb{R}^{E\times D}} \left[\frac{1}{2I}\|\Psi Y - W\Phi\|_F^2 + \frac{\lambda}{2}\|W\|_F^2\right]. \tag{5.4}$$

The solution of ridge regression problem in Eq. (5.4) can be obtained in closed form as:

$$W^* = \Psi Y \Phi^\top \left(\Phi\Phi^\top + I\lambda \mathrm{Id}_D\right)^{-1}, \tag{5.5}$$

where Id_k is the identity matrix in dimension k. Similar to DIFFRAC [Bach and Harchaoui, 2007], substituting the optimal W^* in (5.4) yields:

$$q(Y) = \frac{1}{2I}\mathrm{Tr}\left(\Psi Y Q Y^\top \Psi^\top\right), \tag{5.6}$$

where Q is a matrix depending on the data and the regularization parameter λ:

$$Q = \mathrm{Id}_I - \Phi^\top \left(\Phi\Phi^\top + I\lambda \mathrm{Id}_D\right)^{-1} \Phi. \tag{5.7}$$

Symbolic labels or continuous representations

The model presented here can be used in several ways. For example, we can use Ψ to encode a sequence of action labels: Suppose that there are L different action classes. Let us define the sequence of labels as an array a of J indices a_j in $\{1, \ldots, L\}$:

$$a = [a_1, \ldots, a_J], \tag{5.8}$$

the column order reflecting as before temporal order. Then the matrix Ψ can be constructed as the concatenation of the binary encodings of these classes. We have that $\psi_j = e_k$ if and only if $a_j = k$ where e_k is the k-th vector of the canonical basis of dimension E. The product ΨY corresponds to the assignment of elements in $\{1, \ldots, I\}$ to classes in $\{1, \ldots, E\}$. This product ΨY is exactly equal to (up to a transposition) the class indicator matrix Z in $\{0, 1\}^{J \times E}$ as defined in [Bojanowski et al., 2014]. We illustrate the product ΨY in when Ψ encode action labels in Fig. 5-3.

The great advantage of the proposed model, as opposed to [Bojanowski et al., 2014], is that the data (Ψ) and the alignment variable (Y) are separate. This allows us to use representations for the data stream Ψ that are not necessarily binary.

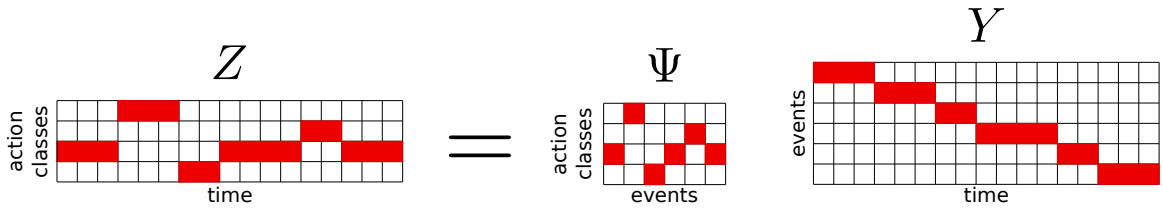


Figure 5-3 – Illustration of the Ψ matrix when aligning a sequence of action labels to a stream of video frames. In that context, we can associate elements i to time steps and elements j to *events*. The dimension E of the feature space of Ψ corresponds to action classes. The product ΨY is equal to a class indicator matrix that assigns each frame to a given action class. The formulation presented in [Bojanowski et al., 2014] makes use of a matrix Z instead of representing separately Ψ and Y.

These could be for instance label embeddings, continuous word representations or bag-of-words sentence representations. Moreover, this allows us to impose event-specific constraints on Y, which was impossible using the matrix $Z = \Psi Y$.

The background class

We have assumed so far that each element i maps onto one element j. However, in many cases, some (or even most) of the elements in $\{1, \ldots, I\}$ do not actually correspond to any element in $\{1, \ldots, J\}$. For instance, if Φ is a sequence of video frames and Ψ encode a list of action labels, only some frames depict an action. The rest should be assigned to the background. This is implemented in [Bojanowski et al., 2014] by adding an explicit background class, and inserting columns in the matrix Ψ. Suppose that there are E classes of interest, the matrix Ψ is thus expanded as follows:

$$\tilde{\Psi} = \begin{bmatrix} \mathbf{0}_E & \psi_1 & \mathbf{0}_E & \psi_2 & \ldots & \psi_J & \mathbf{0}_E \\ 1 & 0 & 1 & 0 & \ldots & 0 & 1 \end{bmatrix}. \tag{5.9}$$

In this presentation, we insert instead zero vectors of dimension E in between the columns of Ψ:

$$\tilde{\Psi} = \begin{bmatrix} \mathbf{0}_E & \psi_1 & \mathbf{0}_E & \psi_2 & \ldots & \psi_J & \mathbf{0}_E \end{bmatrix}, \tag{5.10}$$

and elements i that are not assigned to a proper element j are mapped onto $\mathbf{0}_E$. This has the advantage of not requiring an explicit background model (an additional row

in W).

Affine maps and trivial solutions

As mentioned earlier, we parameterize the mapping between features as a linear one. Since we solve a regression problem, we actually want an affine one instead. This can be done explicitly by adding a bias term b in $\mathbb{R}^E$ and considering the map $\Phi \mapsto W\Phi + b\mathbf{1}_I^\top$. In this work, we choose instead to add a constant row with entries equal to k to the feature Φ:

$$\bar{\Phi} = \begin{bmatrix} \Phi \\ k\mathbf{1}_I^\top \end{bmatrix}, \tag{5.11}$$

and use a linear map defined as $\bar{W} = [W, \frac{1}{k}b]$, which is a classical trick in machine learning [Hastie et al., 2009]. Doing so allows us to have a simpler presentation, simplifies the expressions and the corresponding implementation. Using these augmented variables, we have:

$$\bar{W}\bar{\Phi} = W\Phi + b\mathbf{1}_I^\top. \tag{5.12}$$

The Frobenius norm regularizer is now equal to:

$$\frac{\lambda}{2}\|\bar{W}\|_F^2 = \frac{\lambda}{2}\|W\|_F^2 + \frac{\lambda}{2k^2}\|b\|_2^2. \tag{5.13}$$

Using this augmented linear map is equivalent to using a regularized bias term, which is not desirable in regression problems [Hastie et al., 2009]. However, if we set the value of k large compared to $\sqrt{\lambda}$, the bias become unregularized and we recover the classical formulation.

Discriminative clustering models are known to be plagued by trivial solutions [Bach and Harchaoui, 2007, Joulin et al., 2010, Bojanowski et al., 2013, 2014]. When an explicit bias term is used, the expression of Q contains centering matrices Π. Then it is easy to see that the constant vector $\mathbf{1}_I$ is an eigenvector of Q associated with the eigenvalue 0. However, in our case, when we augment the features with a constant

row, this is less clear. Let us define the augmented quadratic cost matrix $\bar{Q}$ as:

$$\bar{Q} = \mathrm{Id}_I - \bar{\Phi}^\top \left(\bar{\Phi}\bar{\Phi}^\top + I\lambda \mathrm{Id}_{D+1}\right)^{-1} \bar{\Phi}. \tag{5.14}$$

The block matrix inversion formula can be used to show that $\mathbf{1}_I$ is an eigenvector of $\bar{Q}$ associated with eigenvalue 0 as long as k is sufficiently large (See Appendix A). In the rest of the presentation we drop the bar notation, but the reader should keep in mind that we augment the features with a constant row (with values typically around 10^4).

Multiple streams

Suppose now that we are given N data streams (videos in our case), indexed by n in $\{1, \ldots, N\}$. The approach proposed so far is easily generalized to this case by taking Ψ and Φ to be the horizontal concatenation of all the matrices Ψ_n and Φ_n corresponding to these streams:

$$\Psi = [\Psi_1, \ldots, \Psi_N], \tag{5.15}$$

$$\Phi = [\Phi_1, \ldots, \Phi_N]. \tag{5.16}$$

The matrices Y in $\mathcal{Y}$ have to be block-diagonal in this case, the diagonal blocks being the assignment matrices of every stream:

$$Y = \begin{bmatrix} Y_1 & & 0 \\ & \ddots & \\ 0 & & Y_N \end{bmatrix}.$$

Using this parametrization we have:

$$\|\Psi Y - W\Phi\|_F^2 = \sum_{n=1}^{N} \|\Psi_n Y_n - W\Phi_n\|_F^2, \tag{5.17}$$

which is the desired sum of losses over the clips. This is the model actually used in our implementation.

5.2.3 Priors and constraints

As mentioned earlier, our formulation of alignment allows us to incorporate task-specific knowledge in our model by adding constraints on the matrix Y. This allows us to avoid, for example, the degenerate solutions known to plague discriminative clustering [Bach and Harchaoui, 2007, Bojanowski et al., 2014, Guo and Schuurmans, 2007, Joulin et al., 2010]. Indeed, as discussed in Sec. 5.2.2, the constant vector $\mathbf{1}_I$ is in the kernel of Q. Let us consider an example: the (near) degenerate assignment matrix Y depicted in Fig. 5-4 (a) is very close to $\mathbf{e}_1\mathbf{1}_I^\top$. Therefore, the cost function associated with this assignment is close to zero. This motivates the use of priors and constraints that drive the assignment matrix away from this trivial solution.

In this section, we will propose two solutions. The first one consists in modeling the number of elements i that should be assigned to element j. This leads to a quadratic cost in Y that we add to the optimization problem. The second solution corresponds to traditional constraints used when working with dynamic programming. Instead of using hard constraints, however, we propose to dualize them and include them as a linear penalization in our cost function.

Duration priors

Our formulation in terms of an assignment variable Y allows us to reason about the number of elements i that are assigned to element j. In the context of text to video alignment, since each element i is a video time interval, this number is the *duration* of text element j. More formally, the duration $\delta(j)$ of element j is obtained as: $\delta(j) = \mathbf{e}_j^\top Y \mathbf{1}_I$, where $\mathbf{e}_j$ is the j-th vector of the canonical basis of $\mathbb{R}^J$. Given a target duration μ, we can penalize the squared deviation of δ from μ. Assuming for simplicity a single variance parameter for all units, this leads to the following duration

penalty:

$$r(Y) = \frac{1}{2\sigma^2}\|Y\mathbf{1}_I - \mu\|_2^2. \tag{5.18}$$

This can be interpreted as having a Gaussian prior on the duration of each event. Some heuristics for estimating suitable values of σ and the mean vector μ are needed. In our experiments, we pick σ by validation and fix $\mu = \frac{I}{J}\mathbf{1}_J$. Given prior information on the duration of classes, more clever ways to estimate these parameters could be used. For instance, the mean and standard deviation of the duration of actions could be computed from an annotated action classification dataset.

Path priors

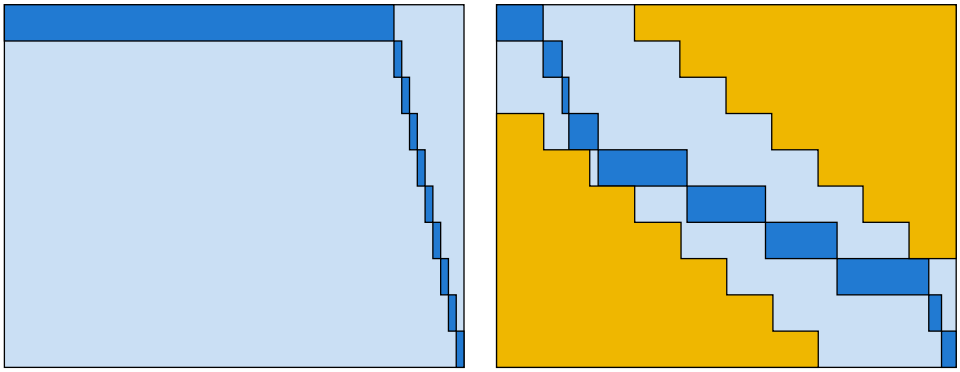
Some elements in $\mathcal{Y}$ correspond to very unlikely assignments. In speech processing and related tasks [Rabiner and Juang, 1993], the warping paths are often constrained, forcing for example the path to fall in the Sakoe-Chiba band or in the Itakura parallelogram [Sakoe and Chiba, 1978]. Such constraints allow us to encode task-specific assumptions and to avoid degenerate solutions associated with the fact that constant vectors belong to the kernel of Q. Band constraints, as illustrated in Fig. 5-4 (b), successfully exclude the kind of degenerate solutions presented in Fig. 5-4 (a).

Let us denote by Y_c the band-diagonal matrix of width β, such that the diagonal entries are 0 and the others are 1; such a matrix is illustrated in Fig. 5-4 (b) in yellow. In order to ensure that the assignment does not deviate too much from the diagonal, we can impose that at most C non zero entries of Y are outside the band. We can formulate this constraint as:

$$\mathrm{Tr}(Y_c^\top Y) \leq C.$$

This constraint could be added to the definition of the set $\mathcal{Y}$. However, doing so would prohibit a key step to our optimization algorithm, as described in Sec. 5.3. We therefore propose to add a penalization term to our cost function, corresponding to the Lagrange multiplier for this constraint. Indeed, for any value of C, there exists an α such that, if we add

$$l(Y) = \alpha\mathrm{Tr}(Y_c^\top Y) \tag{5.19}$$



(a) A (near) degenerate solution. (b) A constrained solution.

Figure 5-4 – **(a)** depicts a typical near degenerate solution where almost all the the elements i are assigned to the first element, close to the constant vector element of the kernel of Q. **(b)** Illustration of the constraint we propose. We want to force the alignment to stay outside of a given region (shown in yellow), which may be a band or a parallelogram. The dark blue entries correspond to the assignment matrix Y while the yellow ones represent the constraint set. See text for more details. (Best seen in color.)

to our cost function, the two solutions are equal, and thus the constraint is satisfied. The function l is linear and adding it to our cost does not change the nature of our objective function. In practice, we select the value of α by doing a grid search on a validation set. In the experimental section, we will evaluate the impact of the width of the band and of the parameter α on the performance of our method.

5.2.4 Full problem formulation

Including the constraints defined in Sec. 5.2.3 into our objective function yields the following optimization problem:

$$\min_{Y \in \mathcal{Y}} \; q(Y) + r(Y) + l(Y), \tag{5.20}$$

where q, r and l are the three functions respectively defined in (5.6), (5.18) and (5.19). This is of course a difficult combinatorial optimization problem since the variables Y are binary and must satisfy complex constraints. In the following section we will describe the continuous and convex relaxation and the optimization algorithm that we use to obtain an approximate solution to this problem.

5.3 Optimization

Let us first describe how we relax the set of assignment matrices $\mathcal{Y}$. This relaxation transforms our problem into the minimization of a convex cost function over a convex domain. We then describe the Frank-Wolfe algorithm which efficiently solves the relaxed problem by exploiting the inherent structure of $\mathcal{Y}$. Finally, we discuss several rounding procedures which, given a solution to the relaxed problem, provide us with a feasible point of the initial problem.

5.3.1 Relaxation

Our problem, as formulated in Sec. 5.2.4, is the minimization of a quadratic function over a very large and discrete domain. It is quite easy to see that the set $\mathcal{Y}$ is composed of $\binom{I-1}{J}$ binary matrices: Indeed, the number of different assignment matrices corresponds to the different ways one can choose to "switch" from j to $j+1$. There are J "switches" to choose amongst the $I-1$ possible choices, hence the aforementioned cardinality.

We use a simple convex relaxation and replace the discrete set $\mathcal{Y}$ by its convex hull $\overline{\mathcal{Y}}$. The set $\overline{\mathcal{Y}}$ is a polytope that could be explicitly described by a polynomial number of linear inequalities. One advantage of the optimization algorithm proposed in the rest of this section is that it does not require having such an explicit description. Even though iterates will be inside the polytope, we don't need to know the parametrization of its facets.

Given an explicit description of $\overline{\mathcal{Y}}$, our problem reduces to a classical quadratic program, that could in principle be solved using any off-the-shelf optimization toolboxes, provided that the quadratic cost matrix and the constraints fit in memory. We want to avoid using the explicit parametrization of the polytope as it involves a large amount of linear constraints and therefore long runtimes. Moreover, by writing our own solver, we manage to avoid storing the full Q matrix in memory. This can be problematic when the datasets get large (I typically of the order of 10^4). In the following section we describe the Frank-Wolfe algorithm which will allow us to minimize

our cost over $\overline{\mathcal{Y}}$ without access to its explicit description.

5.3.2 The Frank-Wolfe Algorithm

We want to minimize a convex function over a polytope $\overline{\mathcal{Y}}$. This set is defined as the convex hull of a large but finite set of integer points exhibiting an alignment structure. When it is possible to optimize a linear function over a constraint set of this kind, but other usual operations are not tractable, a good way to optimize a convex objective function is to use the iterative Frank-Wolfe algorithm (a.k.a. conditional gradient method) [Bertsekas, 1999, Frank and Wolfe, 1956]. We show in Sec. 5.3.3 that we can indeed minimize linear functions over $\overline{\mathcal{Y}}$ efficiently, so this is an appropriate choice in our case.

In order to make notations simpler, let us just address in this section the problem of minimizing q over $\overline{\mathcal{Y}}$:

$$q(Y) = \frac{1}{2I}\mathrm{Tr}(\Psi Y Q Y^\top \Psi^\top), \tag{5.21}$$

ignoring the two additional terms (r and l). The derivations including these two terms are very similar and only make notations more cumbersome.

The idea behind the Frank-Wolfe algorithm is rather simple. At every step k of the algorithm, one computes an affine approximation L_k of the objective function q at the current iterate Y_k. In our case, this affine approximation has the following expression:

$$L_k(Y) = \mathrm{Tr}(Y \nabla_Y q(Y_k)) - q(Y_k), \tag{5.22}$$

where

$$\nabla_Y f(Y) = \frac{1}{I} Q Y^\top \Psi^\top \Psi. \tag{5.23}$$

$L_k(Y)$ is minimized yielding a point Y^* on the boundary of $\overline{\mathcal{Y}}$:

$$Y^* \in \arg\min_{Z \in \overline{\mathcal{Y}}} L_k(Y). \tag{5.24}$$

The constant term in Eq. (5.22) can be safely ignored in this optimization problem

and we therefore focus on minimizing only the linear part. Finally, the iterate is updated as a convex combination of Y^* and the current point Y_k. This is repeated until convergence (see Alg. 1).

The interpolation parameter γ can be chosen either by using the universal step size $\frac{2}{k+1}$, where k is the iteration counter (see [Jaggi, 2013] and references therein) or, in the case of quadratic functions, by solving a univariate quadratic equation. Let us define the Frank-Wolfe descent direction at step k as $\delta Y_k = Y^* - Y_k$. To find the optimal γ, we solve the following problem:

$$\min_{\gamma \in \mathbb{R}} \quad q(Y_k + \gamma \delta Y_k), \tag{5.25}$$

which corresponds to performing an exact line search in the descent direction δY_k. Given our definition for q, this is equivalent to:

$$\begin{aligned} \min_{\gamma \in \mathbb{R}} \quad & \mathrm{Tr}(\Psi Y_k Q Y_k^\top \Psi + 2\gamma \Psi Y_k Q \delta Y_k^\top \Psi^\top \\ & + \gamma^2 \Psi \delta Y_k Q \delta Y_k^\top \Psi^\top). \end{aligned} \tag{5.26}$$

This is a simple second order polynomial in γ and the optimal step size is:

$$\gamma^* = -\frac{\mathrm{Tr}(\Psi \; Y_k \; Q \; \delta Y_k^\top \; \Psi^\top)}{\mathrm{Tr}(\Psi \; \delta Y_k \; Q \; \delta Y_k^\top \; \Psi^\top)}. \tag{5.27}$$

Despite the minus sign, the value of γ^* is actually positive, due to the definition of δY_k.

A good feature of the Frank-Wolfe algorithm is that it provides a gap Δ_k (referred to as the linearization duality gap [Jaggi, 2013]) that can be used as a certificate of sub-optimality and stopping criterion. From now on, we just refer to it as a *gap* and ignore its dual interpretation. At iteration k, Δ_k can be computed using the gradient

$k \leftarrow 0$
$\Delta_0 \leftarrow \infty$
while $\Delta_k \geq \epsilon$ **do**
  Compute the current gradient in Y, $\nabla_Y q(Y_k) = \frac{1}{I} Q Y_k^\top \Psi^\top \Psi$.
  Choose Y^* in $\arg\min_{Y \in \overline{\mathcal{Y}}} \mathrm{Tr}(Y \nabla_Y q(Y_k))$ using dynamic programming.
  Compute the Frank-Wolfe descent direction δY_k.
  Compute the optimal Frank-Wolfe step size γ^*.
  $Y_{k+1} \leftarrow Y_k + \gamma^* \delta Y_k$
  $\Delta_k \leftarrow -\mathrm{Tr}(\delta Y_k \nabla_Y f(Y_k))$.
  $k \leftarrow k + 1$.
end

Algorithm 1: The Frank-Wolfe optimization procedure.

and the descent direction as follows:

$$\begin{aligned}
\Delta_k &= q(Y_k) - L_k(Y^*), \\
&= \frac{1}{I}\mathrm{Tr}(\Psi Y_k Q Y_k^\top \Psi^\top) - \frac{1}{I}\mathrm{Tr}(Y^* Q Y_k^\top \Psi^\top \Psi), \\
&= \frac{1}{I}\mathrm{Tr}((Y_k - Y^*) Q Y_k^\top \Psi^\top \Psi), \\
&= -\mathrm{Tr}(\delta Y_k \ \nabla_Y q(Y_k)).
\end{aligned} \tag{5.28}$$

The complete optimization procedure is described in the special case of our relaxed problem in Algorithm 1. Figure 5-5 illustrates one step of the optimization and the computation of the gap is illustrated in Fig. 5-6.

5.3.3 Minimizing Linear Functions over $\overline{\mathcal{Z}}$ by dynamic programming

The optimization algorithm described in the previous section requires us to minimize linear functions over the relaxed set $\overline{\mathcal{Y}}$. It turns out to be easy to minimize such linear functions over the binary set $\mathcal{Y}$. A classical result in linear programming (see for instance Prop B.21 of [Bertsekas, 1999]) shows that the solution of a linear program over $\mathcal{Y}$ is also a solution over $\overline{\mathcal{Y}}$. Intuitively, the solution over $\overline{\mathcal{Y}}$ is either a vertex of the convex hull, or a whole face (including its vertices). Please note that this face can potentially be lower-dimensional (for instance, in $\mathbb{R}^3$, it could be an edge). In both

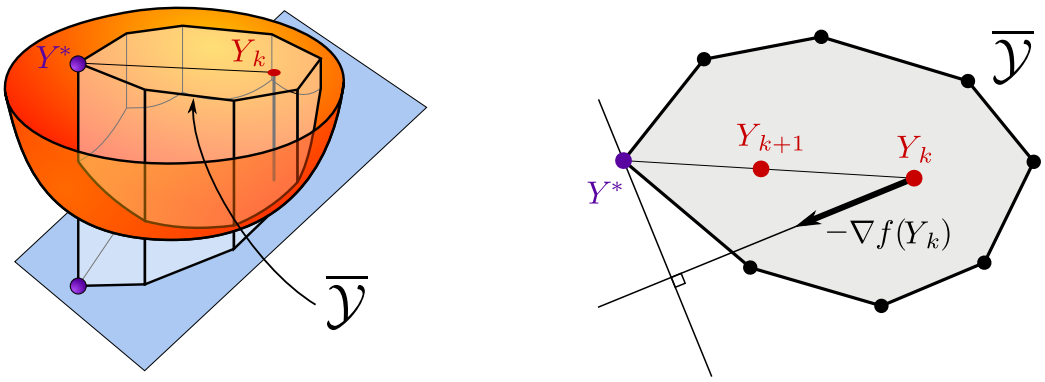


Figure 5-5 – Illustration of a Frank-Wolfe step (see [Jaggi, 2013] for more details). **Left**: the domain $\overline{\mathcal{Y}}$ of interest, objective function (red), and its linearization at current point (blue). **Right**: top view of $\overline{\mathcal{Y}}$.

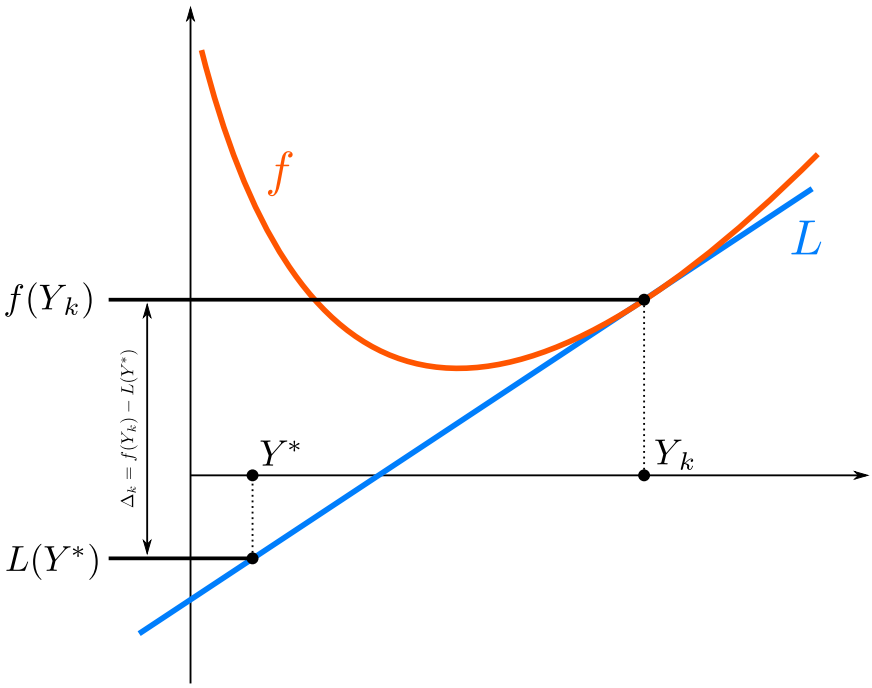


Figure 5-6 – Restriction of the function f in the direction of the Frank-Wolfe descent. We show the plot of the function as well as the linearization L. The gap is computed as the difference between the current objective and the linearization at Y^*. If the optimum is inside the set, the gap is equal to zero because the gradient is simply equal to zero. If the optimum is on a face of the polytope, the linearization has a constant value on the face.

cases, optimizing the same linear cost only over the vertices provides a valid solution over the convex hull. We will therefore focus on the minimization problem over $\mathcal{Y}$ and keep in mind that it also gives a solution over $\overline{\mathcal{Y}}$ as required by the Frank-Wolfe algorithm.

Minimizing a linear function over $\mathcal{Y}$ amounts to solving the problem:

$$\min_{Y \in \mathcal{Y}} \operatorname{Tr}(C^\top Y), \tag{5.29}$$

where C is a cost matrix in $\mathbb{R}^{I \times J}$. This can be re-written as:

$$\min_{Y \in \mathcal{Y}} \sum_{i=1}^{I} \sum_{j=1}^{J} Y_{ij} C_{ij}. \tag{5.30}$$

For each i, there is only one j such that Y_{ij} is equal to one. Let us define a function $m : \{1, \ldots, I\} \to \{1, \ldots, J\}$, such that $m(i) = j$ if $Y_{ij} = 1$. Because of the alignment structure of Y, if $m(i) = j$, then $m(i+1) \in \{j, j+1\}$. Let us call $\mathcal{M}_{IJ}$, the set of such functions such that $m(1) = 1$ and $m(I) = J$. Using this representation, the previous cost can be rewritten as:

$$\min_{m \in \mathcal{M}_{IJ}} \sum_{i=1}^{I} C_{im(i)}, \tag{5.31}$$

which can be solved using dynamic time warping. Indeed, let us define for all $i \in \{1, \ldots, I\}$ and $j \in \{1, \ldots, J\}$:

$$P^*_{ij} = \min_{m \in \mathcal{M}_{ij}} \sum_{t=1}^{i} D_{tm(t)}. \tag{5.32}$$

We can think of P^*_{ij} as the cost of the optimal path from $(1, 1)$ to (i, j) in the graph defined by admissible assignments. We have the following dynamic programming recursion:

$$P^*_{ij} = C_{ij} + \min(P^*_{i-1\ j-1}, P^*_{i-1\ j}). \tag{5.33}$$

The optimal value P^*_{IJ} can be computed in $O(IJ)$ time using dynamic programming, by incrementally computing the P^*_{ij} values, and maintaining at each node (i, j) back

pointers to the the minimal neighbors. The actual solution is found by backtracking the solution from (I, J).

5.3.4 Rounding

Solving the relaxed problem provides a continuous solution Y^* in $\overline{\mathcal{Y}}$ and a corresponding optimal linear map W^*. Our original problem is defined on $\mathcal{Y}$, and we thus need to round Y^*. We propose three rounding procedures, two of them corresponding to Euclidean norm minimization problems and a third one using the map W^*. All three roundings boil down to solving a linear problem over $\mathcal{Y}$, which can be done once again using dynamic programming. Since there is no principled way to pick one of these procedures over the others, we conduct an empirical evaluation in Sec. 5.5 to assess their strengths and weaknesses.

Rounding in $\mathcal{Y}$

The simplest way to round Y^* is to find the closest point Y according to the Euclidean distance in the space $\mathcal{Y}$. This can be written as the following optimization problem:

$$\min_{Y \in \mathcal{Y}} \|Y - Y^*\|_F^2. \tag{5.34}$$

Since we have that:

$$\|Y - Y^*\|_F^2 = \mathrm{Tr}(Y^\top Y) + \mathrm{Tr}(Y^{*\top} Y^*) - 2\mathrm{Tr}(Y^\top Y^*).$$

The first two terms are constant and therefore this problem reduces to a linear program over $\mathcal{Y}$. As mentioned in the previous section, this can be solved efficiently using dynamic programming. Please note that the euclidean distance is not necessarily the most appropriate for this space.

Rounding in $\Psi\mathcal{Y}$

We can also try to find the closest point in the space where the original least-squares minimization is formulated. To this end we use a weighted euclidean distance, and solve the following problem:

$$\min_{Y \in \mathcal{Y}} \|\Psi(Y - Y^*)\|_F^2, \tag{5.35}$$

which weighs the distances using the feature matrix Ψ. As before, expanding the norm gives:

$$\begin{aligned} \|\Psi(Y - Y^*)\|_F^2 = \mathrm{Tr}(\Psi Y Y^\top \Psi^\top) + \mathrm{Tr}(\Psi Y^* {Y^*}^\top \Psi^\top) \\ - 2\mathrm{Tr}(\Psi Y^* Y^\top \Psi^\top). \end{aligned} \tag{5.36}$$

This time, the first term is not constant. We observe that $YY^\top = \mathrm{Diag}(Y\mathbf{1}_I)$, where:

$$\begin{aligned} \mathrm{Diag} : \mathbb{R}^J &\to \mathbb{R}^{J \times J} \\ v &\mapsto \begin{bmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_J \end{bmatrix}. \end{aligned} \tag{5.37}$$

The j-th element of the diagonal counts how many elements i were assigned to j. We have:

$$\mathrm{Tr}(\mathrm{Diag}(Y\mathbf{1})\Psi^\top \Psi) = \mathrm{Tr}(\mathbf{1}_I^\top \mathbf{Y}^\top \mathrm{Diag}^*(\Psi^\top \Psi)), \tag{5.38}$$

where:

$$\begin{aligned} \mathrm{Diag}^* : \mathbb{R}^{J \times J} &\to \mathbb{R}^J \\ A &\mapsto \begin{bmatrix} A_{ii} \\ \vdots \\ A_{JJ} \end{bmatrix}. \end{aligned} \tag{5.39}$$

We can use the invariance of the trace under cyclic permutations and permute the matrices. A simple calculation then shows that the previous problem is equivalent to:

$$\min_{Y \in \mathcal{Y}} \operatorname{Tr}\left(Y^\top \left(\operatorname{Diag}^*(\Psi^\top \Psi)\mathbf{1}_I^\top - 2\Psi^\top \Psi Y^*\right)\right).$$

As before, this can be solved using dynamic programing.

Rounding in W

Our optimization procedure gives us two outputs, namely a relaxed assignment $Y^* \in \overline{\mathcal{Y}}$ and a model W^* in $\mathbb{R}^{E \times D}$. We can use this model to predict an alignment Y in $\mathcal{Y}$ by solving the following quadratic optimization problem:

$$\min_{Y \in \mathcal{Y}} \|\Psi Y - W^* \Phi\|_F^2. \tag{5.40}$$

This rounding is interesting as it makes use of the actual cost function we optimize to choose a point in $\mathcal{Y}$. As before, by expanding the norm we observe that this is equivalent to a linear program. An important feature of this rounding procedure is that it can also be used on previously unseen data.

5.4 Semi-supervised setting

The proposed model is well suited to semi-supervised learning. Incorporating additional supervision just consists in constraining parts of the matrix Y. Let us assume that we are given a triplet (Ψ_s, Φ_s, Y_s) representing supervisory data. The rest of the data is denoted by (Ψ_u, Φ_u, Y_u). Using the additional data amounts to solving (5.20) with matrices (Ψ, Φ, Y) defined as:

$$\Psi = [\Psi_u,\ \kappa\ \Psi_s], \quad \Phi = [\Phi_u,\ \kappa\ \Phi_s], \quad Y = \begin{bmatrix} Y_u & 0 \\ 0 & Y_s \end{bmatrix}, \tag{5.41}$$

where κ is a real weighting parameter.

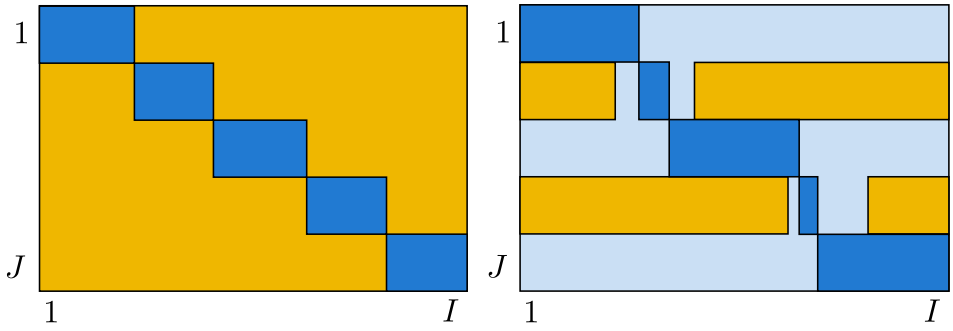
Weighting the features this way corresponds to using the following loss:

$$\|\Psi Y - W\Phi\|_F^2 = \|\Psi_u Y_u - W\Phi_u\|_F^2 + \kappa^2 \|\Psi_s Y_s - W\Phi_s\|_F^2. \quad (5.42)$$

The parameter κ allows us to properly weigh the supervised and unsupervised data. This can have a significant impact on the performance in practice. Since Y_s is given, we can optimize over $\mathcal{Y}$ while constraining the lower right block of Y. As far as implementation is concerned, this means that we fix the lower-right entries in Y to the ground-truth values during optimization.

We have observed that manual annotations of videos are sometimes imprecise, or incoherent with the prediction of our method. In video for instance, our model, based on a discriminative cost, focuses on the most characteristic part of the action. However, human annotators often deem the actions to last longer. In the case of drinking for example, annotations often also include pouring stuff in the glass as part of the action. Therefore, we propose to include these annotations in a soft manner instead.

As mentioned in Sec. 5.2, in order to model a "background class", odd columns in Ψ are filled with zeros. In the context of text to video alignment, this allows some video frames not to be assigned to any text. Instead of imposing that the assignment Y coincides with the ground-truth annotations, we constrain it to lie "within" annotated intervals. For any even (non null) element j, we force the set of video frames that are assigned to j to be a subset of those in the ground truth interval. That way, we allow the assignment to pick the most discriminative parts of the video within the annotated interval. We illustrate these constraints in Fig. 5-7. This way of incorporating supervision empirically yields much better performance. We experimetnally demonstrate this improvement in Sec. 5.5.2.



(a) Y fixed to ground truth. (b) Corresponding constraints.

Figure 5-7 – Two ways of incorporating supervision. **(a)** the assignments are fixed to the ground truth: the dark blue entries exactly correspond to Y_s, and golden entries are forbidden assignments; **(b)** the assignments are constrained. For even rows, assignments must be outside the golden strips. Light blue regions correspond to authorized paths for the assignment.

5.5 Experimental evaluation

We first present some experiments on synthetic data in a controlled setup, and present them in Sec. 5.5.1. Doing so allows us to analyze the limitations of the model, with respect to various problem-dependent constants (such as T, K *etc.*). We then describe in Sec. 5.5.2 the dataset used for our real-world experiments with Hollywood movies. We evaluate how well our model labels videos and temporally localizes actions. Finally, in Sec. 5.5.3, we present text-to-video alignment experiments where Ψ is a continuous text representation. We perform these experiments on a video dataset composed of cooking activities with textual annotations called TACoS [Regneri et al., 2013].

Experimental Setup. The model described in this chapter does not include a training and testing phase *per se*. In order to carry out experiments in a proper fashion, we split datasets into three parts (Fig. 5-8). In all our experiments we select these splits at random, using several random seeds in order to obtain confidence intervals. We denote these three splits as *Sup* (for supervised), *Eval* (for evaluation) and *Val* (for validation). *Sup* is the part of data that has time-stamped annotations that are used in the semi-supervised setup. During the optimization, only a portion

Figure 5-8 – Illustration of our scheme for splitting the dataset for validation and evaluation. During optimization, our algorithm has access to the union of the three sets. When in the semi-supervised setup, ground-truth assignments are provided on a portion of the *Sup* set. Hyper parameters are chose based on the performance on the *Val* set and performance is reported on *Eval*. In our semi-supervised experiments we evaluate the performance of the method for different values of s.

S of the annotations of the *Sup* set is used. *Val* is the set of annotated examples on which we automatically adjust the hyper-parameters for our method $(\lambda, \sigma, \alpha, \kappa)$. In practice we fix the *Val* set to contain 5% of the clips in the dataset. All the numbers we provide are computed on the *Eval* set. We optimize our cost function over the union of all three sets, so samples from the *Eval* set (without the annotations) are visible during optimization. Please note that in any case, our method requires time-stamped annotations on a validation set in order to select hyper parameters.

Performance measure. We want to use a performance measure that quantifies the difference between a ground-truth assignment Y_{gt}, and the predicted one, Y^*. Since we interleave background columns in Ψ, we only evaluate the even rows of Y^*. For every such row j, we have a corresponding (prediction, ground-truth) interval pair that we will respectively denote G_j^* and G_j. We measure the assignment quality for each row by computing the following quantity:

$$\frac{|G_j \cap G_j^*|}{|G_j^*|}. \tag{5.43}$$

This performance measure is well suited to our problem since: **(1)** it is high if the prediction is included in the ground-truth annotation; **(2)** it is low if the prediction is larger than the annotation (it includes it); **(3)** it is lowest if the prediction is out of the annotation; and **(4)** it does not take into account the prediction of the background class. The score is averaged across all elements j of interest. The perfect score of 1 is achieved when all actions are assigned to the correct annotations. However, accurate

temporal segmentation is not required as long as the predicted bounds are within the ground truth interval. This measure is similar to the standard Jaccard measure used for comparing ensembles [Jaccard, 1912]. However, in the Jaccard measure, the denominator is $|G_j \cup G_j^*|$, therefore penalizing predictions that are too short, which is not desirable in our case.

5.5.1 Controlled setup

In order to assess the performance of the proposed model when characteristics of the dataset vary, we perform some experiments on toy data. This allows us to control the length of both streams, the representation for both Φ and Ψ, and the ground-truth Y. Using this data, we will analyze the performance of the three proposed rounding schemes and the appearance of trivial solutions. But let us first describe the procedure in use to generate the synthetic dataset.

Dataset

In this section, we use a binary class encoding for the matrix Ψ and consider K different classes. We pick at random using a uniform distribution the average duration for each of the K classes and the length of the $(K+1)$-th background class. Let us fix the number of clips to N and for each clip n, uniformly select at random the length of the stream Ψ_n. Then, in order to build the stream, we select a sequence of integers in $\{1, \ldots, K\}$ by drawing them with replacement. Using the binary class representation, and interleaving background class in between columns, we obtain the matrix Ψ_n in $\{0,1\}^{K \times J_n}$. For each of the J_n elements, we select a length using a gaussian distributions whose parameters were selected beforehand. That way, we build the ground-truth assignment matrix Y_n.

Once these are computed, we pick a random map W, by taking a matrix in $\mathbb{R}^{K \times d}$ with $d-1$ columns filled with independent Gaussian noise and the last one filled with a constant. For each clip n, we obtain the feature representation Φ_n by solving the

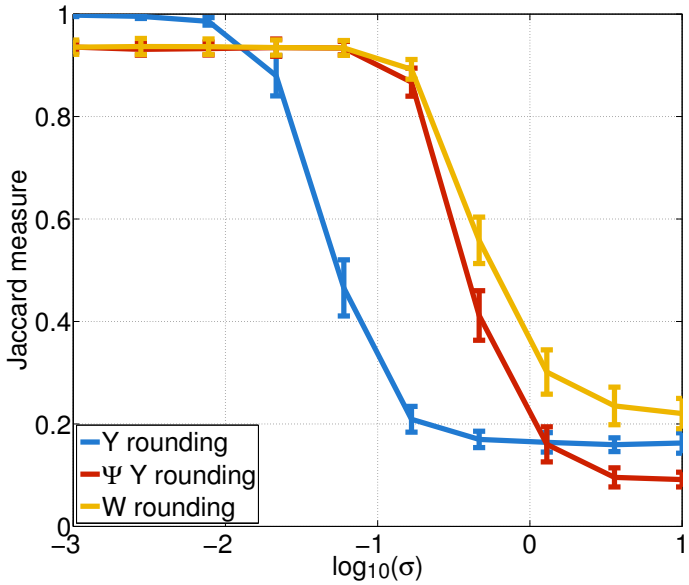


Figure 5-9 – On the toy dataset, we check the performance of the three rounding procedures as the noise in Φ increases.

least squares problem:

$$\min_{\Phi_n} \|\Psi_n Y_n - W\Phi_n\|_F^2, \tag{5.44}$$

and then adding Gaussian noise of variance σ to each of the Φ_n. Using this procedure to generate our toy dataset, we have data that corresponds to our model. This allows us to observe our model's inherent shortcomings. In our experiments, we will observe the evolution of performance as a function of the noise variance.

Roundings and noise

The first thing that we want to analyze on this synthetic data is the performance of the three proposed roundings. We propose to observe their performance as we change the amount of noise that we add to the features Φ. We split the dataset as described at the beginning of this section and pick hyper parameters on the validation set. In this experiment, we don't use any of the priors that we described so we only have to select λ. All experiments are performed without supervision on the *sup* set ($S = 0$). We run them for ten different random seeds and report the average performance along with the standard deviation.

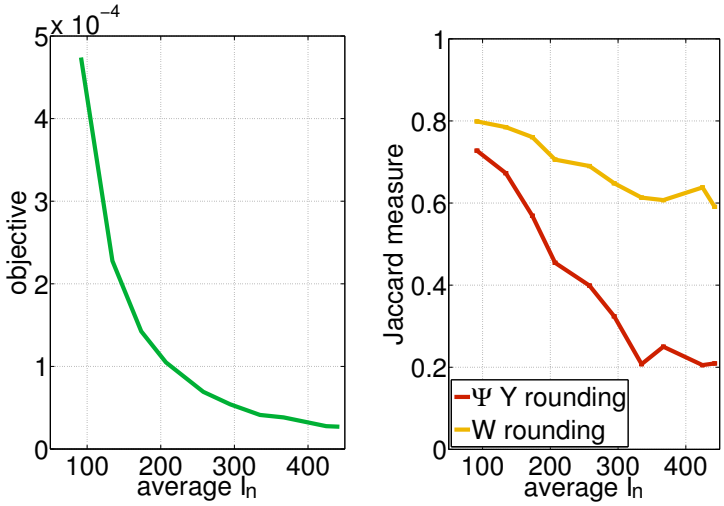
The performance of the three rounding schemes as a function of the noise σ in log scale is presented in Fig. 5-9. The first obvious observation is that the performance of all three roundings degrades as the noise increases. There are three main regimes: absence of noise / a phase transition / very noisy features. When the noise is negligible, the solution obtained using the Y rounding performs close to perfection. The two other roundings have a slightly worse performance. The loss in performance is observed for streams that have the same element appearing multiple times in a row in Ψ. The ΨY and the W roundings cannot make a distinction between the two occurrences and elements of Φ are assigned to one of them arbitrarily.

When the noise increases, the Y rounding degrades very quickly while the two other ones keep a decent performance. As explained in Sec. 5.3.4, the three roundings proposed are arbitrary heuristic procedures. When inserting null columns in Ψ, we realize that the values of odd rows in Y do no change the objective. Performing the rounding using a euclidean distance in the space $\overline{\mathcal{Y}}$ will select a binary Y while taking into account the values of odd rows of the relaxed solution. When rounding in ΨY or W, the influence of these rows is discarded, therefore focusing only on the rows of interest. This illustrates the fact that the Y rounding should not be used, as it happens in a space where the norm has no relation whatsoever with the cost function.

The best performing rounding procedure, for realistic noise ranges is the one in W. This is expected as it is the only one that actually uses a cost which is tightly related to the objective function of our problem. As we will see it in the following section, it is also the rounding procedure which is most robust to the appearance of trivial solutions.

Trivial solutions

In the next experiment, in a controlled setup, we want to see how the trivial solutions appear. To this end, we run our algorithm on several variants of our toy dataset. The main parameters that we change are the lengths I_n of the streams Φ_n. In practice we change I_n by simply changing the average length of the background



(a) objective (b) performance

Figure 5-10 – Evolution of the objective function and of the performance on the evaluation set as the length of stream Φ increases. We clearly see that the trivial solutions described in Sec. 5.2.2 take over when the average I_n increases.

element (odd elements in the stream Ψ). However, for each clip n, we keep the same number of elements J_n (number of columns in Ψ). In order to make the problem nontrivial, we set the value of the noise parameter σ to 0.2.

In order to exhibit the appearance of these trivial solutions, we run the algorithm for various values of I. We show the evolution of the objective function and of the evaluation performance as the Φ stream gets longer. We plot these values as a function of the average stream length $\frac{1}{N}\sum_{n=1}^{N} I_n$ in Fig. 5-10.

We clearly see that as the length of the streams increases, both the performance and the objective drop. The evolution of the objective and how it decreases towards 0 indicates that the reason for this are the trivial solutions described before. Indeed, matrices Y with constant rows yield an objective value of 0. However, such matrices are not in the relaxed optimization domain $\overline{\mathcal{Y}}$. As I_n gets larger, the set of matrices $\overline{\mathcal{Y}}$ grows very quickly and gets closer to the aforementioned set of degenerate solutions. Please note that this happens because these degenerate solutions yield a lower score than the actual ground truth assignment Y_{gt}. This is the case only when the noise in the data is non-trivial and if the features are "perfect" ($\sigma = 0$), we don't observe this drop.

5.5.2 Aligning sequences of actions [Bojanowski et al., 2014]

We evaluate the performance of our model on the data used in [Bojanowski et al., 2014]. It consists of videos annotated with a sequence of actions, where the precise temporal position of the actions is unknown. This has practical applications as providing precise temporal boundaries of actions in video is hard and time consuming. Using the proposed model, the annotator would only be asked to provide the sequence of actions that happens in the video. We first describe the dataset used in this experiment and the feature representation that we use. We then discuss the baselines to which we compare to and eventually present our results.

Dataset

Our input data consists of challenging video clips taken from feature movies that are annotated with sequences of actions. One possible source for such data is movies with their associated scripts [Bojanowski et al., 2013, Duchenne et al., 2009, Laptev et al., 2008, Sivic et al., 2009]. However, the annotations provided are noisy and do not provide ground-truth time-stamps for evaluation. To address this issue, we have constructed a new action dataset, containing clips explicitly annotated by sequences of actions. We have taken the 69 movies from which the clips of the Hollywood2 dataset were extracted [Laptev et al., 2008], and manually added full time-stamped annotations for 16 action classes (12 of these classes are already present in Hollywood2). To build clips that form our input data, we search in the annotations for action chains containing at least two actions. To do so, we pad the temporal action annotations by 250 frames and search for overlapping intervals. A chain of such overlapping annotations forms one video clip with associated action sequence in our dataset. In the end we obtain 937 clips, with the number of actions ranging from 2 to 11. We subdivide each clip into temporal intervals of length 10 frames. Clips contain on average 84 intervals, the shortest containing 11, the longest 289.

Feature representation

We have to define a feature vector ϕ_i for every interval i of a video stream Φ. We split the video into 10 frames long intervals and represent this interval using a bag-of-words vector ϕ_i. To aggregate enough features, we decided to pool features from the 30-frame-long window centered on the interval. We compute video descriptors following [Wang and Schmid, 2013], and generate a vocabulary of size 2000 for HOF features. We only use one channel to improve the running time, being aware that by doing so we sacrifice some performance. In our experiments, we also tried the MBH channels yielding very close performance. We use the Hellinger kernel as it has shown good performance for action recognition in the past. We do so by computing the explicit feature map: square-rooting the l_1 normalized histograms.

The label sequence provides a weak supervisory signal that can be used as our features Ψ. We consider a language composed of sixteen words, where every word corresponds to a label. Then, the representation ψ_j of every element j is the indicator vector of the j-th label in the sequence. Since we do not model background, we simply interleave zero vectors in between meaningful elements.

Baselines

We compare the performance of our full model with some striped-down variants and several simple baselines. Given the very poor performance of the Y rounding, we don't report it in this experiment. However, for the two other roundings, we report the performance of both fixed and constrained semi-supervised models. As described in Sec. 5.4, the supervision available on the *Sup* set can be incorporated in two ways.

We also compare our model to a simple supervised square loss baseline (SL). For a given fraction S of annotated data on the *Sup* set, we train the affine model using only the supervised data:

$$\min_{W} \ \frac{1}{2I_s}\|\Psi_s Y_s - W\Phi_s\|_F^2 + \frac{\lambda}{2}\|W\|_F^2. \tag{5.45}$$

Given the optimal map W^* we can get the corresponding integer assignment Y^* using

the same procedure as for the W rounding. For this baseline, we compare two simple variants: as described in Sec. 5.2.2, we either explicitly model the background as in Eq. (5.9), or skip it as in Eq. (5.10). We refer to these two variants as respectively (SL with-back) and (SL no-back).

As a sanity check, we compare the performance of our algorithm to that of a random assignment that follows the priors. This random baseline obtains a performance measure of 32.8 with a standard error of 0.3.

Evaluating the assignment

We present the performance of our algorithm and the described baselines as a function of the amount of supervised data in Fig. 5-11. We run all methods for 10 different random splits and report the average performance and standard error. This experiment allows us to make several observations. First, the best performing rounding on this dataset is the one using the weighted norm (ΨY). This is not the same as what we observed on the synthetic and on the TACoS datasets.

Second, our semi-supervised model always works better than the supervised square loss baseline (SL). This shows that our weakly-supervised model is capable of making good use of the additional supervised data. The difference in performance between our best model and (SL no-back) is consistently better by at least 2% of performance measure. We can also notice that using only a small portion of supervised data ($S = 0.1$), our model performs as well as (SL no-back) with $S = 0.5$.

Another interesting fact is the big difference in terms of performance between (SL no-back) and (SL with-back). We observe that for any amount of supervised data, not having an explicit model of the background works better. When the background is not modeled, the classes are better balanced, therefore avoiding scaling problems. Indeed, since background is what is happening in between classes of interest, this dummy class is overrepresented.

Eventually, when comparing the two ways to incorporate full supervision, we observe a significative difference. It is always better to use the supervision as constraints rather than explicitly fixing the assignment. This is observed for both the W and the

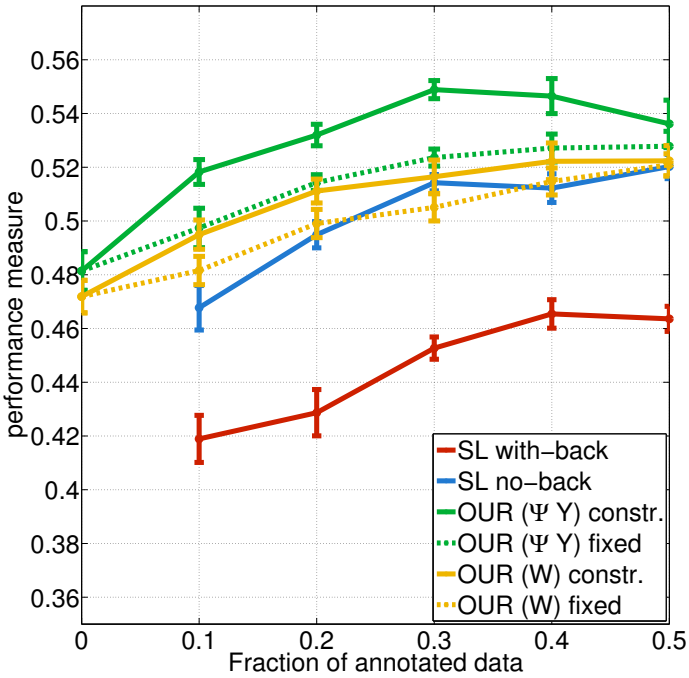


Figure 5-11 – Evaluation of our model and supervised baselines on the data from [Bojanowski et al., 2014]. We observe that on this data, using the ΨY rounding works better than using W.

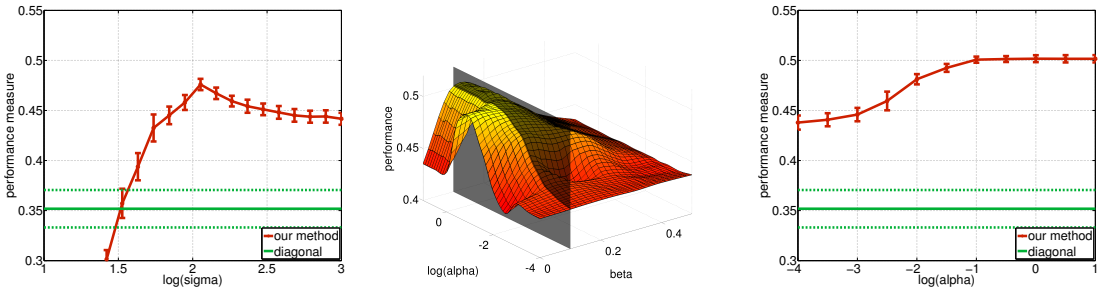
ΨY roundings. However, even when fixing the assignment to the ground-truth we don't observe the drop at the beginning of the curve as in [Bojanowski et al., 2014]. This is probably due to the fact that we don't explicitly model the background class.

5.5.3 Text-to-video alignment

We also evaluate our method on the TACoS dataset [Regneri et al., 2013]. This allows us to evaluate it on a corpus including actual natural language sentences. On this dataset, we use the W rounding as it is the one that empirically gives the best test performance. We do not have yet a compelling explanation as to why this is the best performing one.

Dataset

The TACoS dataset is composed of 127 videos picturing people who perform cooking tasks. Every video is associated with two kinds of annotations. The first



(a) duration prior (b) band prior (c) band prior

Figure 5-12 – Evaluation of the priors we propose in this chapter. **(a)** We plot the performance of our model for various values of σ. When σ is big, the prior has no effect. We see that there is a clear trade off and an optimal choice of σ yields better performance. **(b)** Performance as a function of α and width of the band. The shown surface is interpolated to ease readability. **(c)** Performance for various values of α. This plot corresponds to the slice illustrated in (b) by the black plane.

one is composed of low-level activity labels with precise temporal location. We do not make use of these fine-grained annotations in this work. The second one is a set of natural language descriptions that were obtained by crowd-sourcing. Annotators were asked to describe the content of the video using simple sentences. Each video Φ is associated with k textual descriptions $[\Psi^1, \ldots, \Psi^K]$. Every textual description is composed of multiple sentences with associated temporal extent. We consider as data points the pairs (Ψ^k, Φ) for k in $\{1, \ldots, K\}$.

Feature representation

We build the feature matrix Φ by computing dense trajectories [Wang and Schmid, 2013] on all videos. We compute dictionaries of 500 words for HOG, HOF and MBH channels. These experimentally provide satisfactory performance while staying relatively low dimensional. For a given temporal window, we concatenate bag-of-words representations for the four channels. As in the Hellinger kernel, we use the square root of L_1 normalized histograms as our final features. We use temporal windows of length 150 frames with a stride of 50.

To apply our method to textual data, we need to build a suitable feature representation Ψ for sentences. For every sentence in a textual description, we build a feature ψ_i. In our experiments, we explore multiple ways to represent sentences and empiri-

cally compare their performance (Table 5.1). We discuss two ways to encode sentences into vector representations, one based on bag of words, the other on continuous word embeddings [Mikolov et al., 2013].

To build our bag-of-words representation, we construct a dictionary using all sentences in the TACoS dataset. We run a part-of-speech tagger and a dependency parser [Manning et al., 2014] in order to exploit the grammatical structure. These features are pooled using three different schemes. (1) ROOT: In this setup, we simply encode each sentence by its root verb as provided by the dependency parser. (2) ROOT+DOBJ: In this setup we encode a sentence by its root verb and its direct object dependency. This representation makes sense on the TACoS dataset as sentences are in general pretty simple. For example, the sentence "The man slices the cucumber" is represented by "slice" and "cucumber". (3) VNA: This representation is the closest to the usual bag-of-words text representation. We simply pool all the tokens whose part of speech is Verb, Noun or Adjective. The two first representations are very rudimentary versions of bags of words. They typically contain only one or two non zero elements.

We also explore the use of word embeddings [Mikolov et al., 2013], trained on three different corpora. First, we train them on the TACoS corpus. Even though the amount of data is very small (175,617 words), the vocabulary is also limited and the sentences are simple. Second, we train the vector representations on a dataset of 50,993 kitchen recipes, downloaded from allrecipes.com. This corresponds to a corpus of roughly 5 million tokens. However, the sentences are written in imperative mode, which differs from the sentences found in TACoS. For completeness, we also use the WaCky corpus [Baroni et al., 2009], a large web-crawled dataset of news articles. We train representations of dimension 100 and 200. A sentence is then represented by the concatenation of the vector representations of its root verb and its root's direct object.

Baselines

On this dataset, we considered two baselines. The first one is [Bojanowski et al., 2014] using the ROOT textual features. Verbs are used in place of labels by the method. The second one, that we call Diagonal, corresponds to the performance obtained by the uniform alignment. This corresponds to assigning the same amount of video elements i to each textual element j.

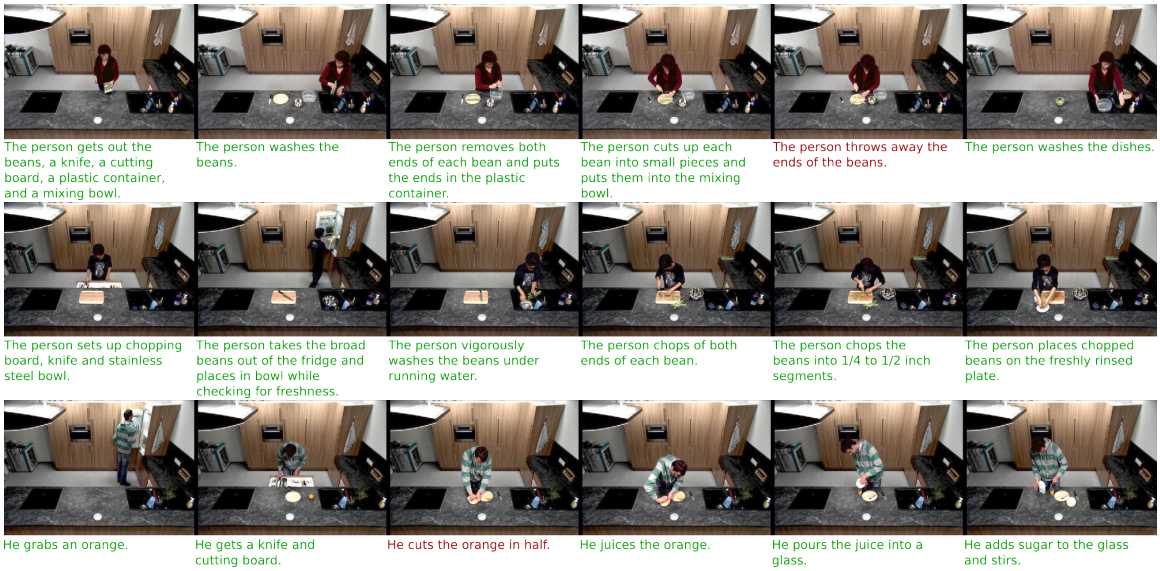


Figure 5-13 – Representative qualitative results for our method applied on TACoS. Correctly assigned frames are in green, incorect ones in red.

Evaluation of the priors

We propose in Sec. 5.2.3 two priors for including prior knowledge and avoiding degenerate solutions to our problem. In this section, we evaluate the performance of the two proposed priors on TACoS. To this end, we run our method with the two

text representation	Dim. 100	Dim. 200
W2V UKWAC	43.8 (1.5)	46.4 (0.7)
W2V TACoS	48.3 (0.4)	48.2 (0.4)
W2V ALLRECIPE	43.3 (0.7)	44.7 (0.5)

Table 5.1 – Comparison of text representations trained on different corpora, in dimension 100 and 200.

different models separately. We perform this experiment using the ROOT+DOBJ text representation. The results of this experiment are illustrated in Fig. 5-12.

We see that both priors are useful. The duration prior, when σ is carefully chosen, allows us to improve performance from 0.441 (infinite σ) to 0.475. There is a clear trade-off in this parameter. Using a bit of duration prior helps us to get a meaningful Y^* by discarding degenerate solutions. However, when the prior is too strong, we obtain a degenerate solution with decreased performance.

The band prior (as depicted in Fig. 5-12, b and c) improves performance even more. We plot in (b) the performance as a joint function of the parameter α and of the width of the band β. We see that the width that provides the best performance is 0.1. We plot in (c) the corresponding performance as a function of α. Using large values of α corresponds to constraining the path to be entirely inside the band, which explain why the performance flattens for large α. When using a small width, the best path is not entirely inside the band and one has to carefully choose the parameter α.

We show in Fig. 5-12 the performance of our method for various values of the parameters on the evaluation set. Please note however that when used in other experiments, the actual values of these parameters are chosen on the validation set. Sample qualitative results are shown in Fig. 5-13

Evaluation of the text representations

In Table. 5.1, we compare the continuous word representations trained on various text corpora. The representation trained on TACoS works best. It is usually advised

text representation	nosup	semisup
Diagonal	35.2 (3.7)	
ROOT	49.9 (0.2)	59.2 (1.0)
ROOT+DOBJ	48.7 (0.9)	65.4 (1.0)
VNA	45.7 (1.4)	59.9 (2.9)
W2V TACoS 100	48.3 (0.4)	60.2 (1.5)

Table 5.2 – Performance when no supervision is used to compute the assignment (nosup) and when half of the dataset is provided with time stamped sentences (semisup).

to retrain the representation on a text corpus that has similar distribution to the corpus of interest. Moreover, higher-dimensional representations (200) do not help probably because of the limited vocabulary size. The representations trained on a very large news corpus (UKWAC) benefits from using higher-dimensional vectors. With such a big corpus, the representations of the cooking lexical field are probably merged together.

In Table. 5.2, we experimentally compare our approach to the baselines, in an unsupervised setting and a semi-supervised one. First, we observe that the diagonal baseline has reasonable performance. Note that this diagonal assignment is different from a random one since a uniform assignment between text and video in our context makes some sense. Second, we compare to the method of [Bojanowski et al., 2014] on ROOT, which is the only set up where this method can be used. This baseline is higher than the diagonal one but pretty far from the performances of our model using ROOT as well.

Using bag-of-words representations, we notice that simple pooling schemes work best. The best performing representation is purely based on verbs. This is probably due to the fact that richer representations can mislead such a weakly supervised method. As additional supervision becomes available, the ROOT+DOBJ pooling works much better that only using ROOT validating the previous claim.

5.6 Conclusion and discussion.

We presented a method able to align a video with its natural language description. We would like to extend our work to even more challenging scenarios including feature movies and more complicated grammatical structures. Also, our use of natural language processing tools is limited, and we plan to incorporate better grammatical reasoning in future work.

Chapter 6

Conclusion

We presented in this thesis two models that allow us to draw links between video and textual signal. The first one uses names of characters and verbs in the script to learn person and action models. Using names in scripts and character identification in video we provide a link between sentences in the script and tracks in the video. We evaluated the character identification model and showed that it improved over state-of-the-art methods at the time of publication while being a very flexible framework. The action recognition performance was encouraging but low redundancy of verbs in movies made it hard to scale the method up. Simple improvements proposed in this thesis showed better performance and prove that the model is very flexible and that it is easy to incorporate such improvements.

A potential path for future work would be to scale this model up to many movies and many action classes. Even though the set of characters is specific to every movie, "action classes" or verbs, are shared. One could imagine training character models movie by movie and having a global action model that would be shared. However, this would probably generate optimization issues, as the number of samples to consider would grow linearly with the number of movies. The cost function that we proposed involved a quadratic program that might become intractable. Stochastic variants of the model could be considered, where the algorithm would not have access to all movies at the same time.

The second model allows to align sentences to a video stream by exploiting tem-

poral ordering constraints. This alignment is recovered based on a joint model of video and text that we learn while finding the alignment. We have shown some experimental results on a dataset composed of cooking videos along with textual directions in the form of steps. While being quite challenging, the videos in this dataset were shot in a controlled environment. The model could also be applied to more complex datasources, for instance "do it yourself" cooking video found on the web.

The aforementioned temporal alignment model can also be used to align videos with action labels. The model automatically localizes the part of the video that corresponds to a given labels, therefore removing some annotation burden from the annotator. As argued in the introduction, providing precise temporal bounds to actions in video is very time consuming. Moreover, all annotators do not agree on the position of these bounds therefore creating annotations that are not all distributed the same way. By using the kind of alignment models that we presented, the most discriminative portion of the video is selected, using the human annotation as weak supervision. An interesting extension of this work would be applying this kind of methods to modern, very large, action recognition datasets. Indeed, most of the videos found in these datasets contain a lot of noise and the video clips are not trimmed in a coherent way.

Both models try to establish a link between textual and video data. One of the golden goals of automatic video understanding is the automatic generation of textual descriptions of the scene pictured in the video. Recently published video captioning models show some promising results, opening up a new field of research. However, in order to properly describe the content of a video, one needs to properly understand the interactions between agents. Current video and image understanding models lack a proper representation of relations between objects and people.

As far as the formulations are concerned, all our models are based on linear classifiers and squared euclidian norm losses. These models could be made richer by considering a class of more complex classification functions and better losses. Standard, heuristic feature could be replace by learned representations based on deep learning. Instead of learning a linear mapping on top of fixed video features, one could train the whole representation by back propagating the gradient.

Bibliography

Francis Bach and Zaïd Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *NIPS*, 2007.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *COLING-ACL*, 1998.

Kobus Barnard, Pinar Duygulu, David A. Forsyth, Nando de Freitas, David M. Blei, and Michael I. Jordan. Matching Words and Pictures. *JMLR*, 2003.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Ressources and Evaluation*, 2009.

Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee Whye Teh, Erik G. Learned-Miller, and David A. Forsyth. Names and Faces in the News. In *CVPR*, 2004.

Tamara L Berg, Alexander C Berg, Jaety Edwards, and David A Forsyth. Who's in the picture. *NIPS*, 2005.

Dimitri Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

Piotr Bojanowski, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Finding Actors and Actions in Movies. In *ICCV*, 2013.

Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014.

Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. Weakly-supervised alignment of video with text. In *ICCV*, 2015.

Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 1973.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 1990.

Timothée Cour, Benjamin Sapp, Chris Jordan, and Benjamin Taskar. Learning from ambiguously labeled images. In *CVPR*, 2009.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *SEM*, 2012.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014.

Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis Bach, and Jean Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.

Pinar Duygulu, Kobus Barnard, Nando de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, 2002.

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

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David A. Forsyth. Every Picture Tells a Story: Generating Sentences from Images. In *ECCV*, 2010.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 1956.

Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

Adrien Gaidon, Zaïd Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *IJCV*, 2014a.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *ECCV*, 2014b.

Edouard Grave. A convex relaxation for weakly supervised relation extraction. In *EMNLP*, 2014.

Edouard Grave and Noémie Elhadad. A convex and feature-rich discriminative approach to dependency grammar induction. In *ACL*, 2015.

Yuhong Guo and Dale Schuurmans. Convex Relaxations of Latent Variable Training. In *NIPS*, 2007.

Abhinav Gupta and Larry S. Davis. Beyond Nouns: Exploiting Prepositions and Comparative Adjectives for Learning Visual Classifiers. In *ECCV*, 2008.

Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

David Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2009.

Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 2013.

Harold Hotelling. Relations between two sets of variates. *Biometrika*, 1936.

Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 1912.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

Armand Joulin, Francis Bach, and Jean Ponce. Discriminative Clustering for Image Co-segmentation. In *CVPR*, 2010.

Armand Joulin, Francis Bach, and Jean Ponce. Multi-class cosegmentation. In *CVPR*, 2012.

Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *ECCV*, 2014.

Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 2010.

Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

Andrej Karpathy, Armand Joulin, and Fei Fei F Li. Deep fragment embeddings for bidirectional image sentence mapping. In *NIPS*, 2014.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. A large-scale classification of english verbs. *Language Resources and Evaluation*, 2008.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *NIPS*, 2015.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C. Berg, and Tamara L. Berg. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, 2011.

Suha Kwak, Bohyung Han, and Joon Hee Han. Scenario-based video event recognition by constraint flow. In *CVPR*, 2011.

Tian Lan, Yang Wang, and Greg Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011.

Ivan Laptev. On space-time interest points. *IJCV*, 2005.

Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006.

Rémi Lebret, Pedro O Pinheiro, and Ronan Collobert. Phrase-based image captioning. *arXiv preprint arXiv:1502.03671*, 2015.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1982.

Jie Luo, Barbara Caputo, and Vittorio Ferrari. Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In *NIPS*, 2009.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. What's cookin'? interpreting cooking videos using text, speech and vision. *NAACL*, 2015.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL (Demo.)*, 2014.

Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR*, 2009.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

Minh Hoai Nguyen, Zhen-Zhong Lan, and Fernando de la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

Juan Carlos Niebles, Chih-Wei Chen, and Fei-Fei Li. Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In *ECCV*, 2010.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, 2011.

Lawrence R. Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. Linking people in videos with "their" names using coreference resolution. In *ECCV*, 2014.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *TACL*, 2013.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. A dataset for movie description. In *CVPR*, 2015.

Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012a.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script Data for Attribute-Based Recognition of Composite Activities. In *ECCV*, 2012b.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. Translating video content to natural language descriptions. In *ICCV*, 2013.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing*, 1978.

Pramod Sankar, CV Jawahar, and Andrew Zisserman. Subtitle-free movie to script alignment. In *BMVC*, 2009.

Scott Satkin and Martial Hebert. Modeling the Temporal Extent of Actions. In *ECCV*, 2010.

Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003.

Nataliya Shapovalova, Arash Vahdat, Kevin Cannons, Tian Lan, and Greg Mori. Similarity constrained latent support vector machine: An application to weakly supervised action classification. In *ECCV*, 2012.

Josef Sivic, Mark Everingham, and Andrew Zisserman. "Who are you?" - Learning person specific classifiers from video. In *CVPR*, 2009.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2014.

Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. "Knock! Knock! Who is it?" probabilistic person identification in tv-series. In *CVPR*, 2012.

Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. Book2movie: Aligning video scenes with book chapters. In *CVPR*, 2015.

Arash Vahdat, Kevin Cannons, Greg Mori, Sangmin Oh, and Ilseo Kim. Compositional models for video event detection: A multiple kernel learning latent variable approach. In *ICCV*, 2013.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *ICCV*, 2015a.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL*, 2015b.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *ICCV*, 2013.

Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

Y. Wang and G. Mori. A discriminative latent model of image region and object tag correspondence. In *NIPS*, 2010.

Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

Linli Xu, James Neufeld, Bryce Larson, and Dale Schuurmans. Maximum Margin Clustering. In *NIPS*, 2004.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015.

Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 2015.

Appendix A

Zero eigenvalues of Q

We want to show that $\mathbf{1}_I$ is in the kernel of $\bar{Q}$ (eigenvector associated with eigenvalue 0). Let us use the augmented definition of $\bar{Q}$ and define B:

$$\bar{Q} = \mathrm{Id}_I - \bar{\Phi}^\top (\underbrace{\bar{\Phi}\bar{\Phi}^\top + I\lambda \mathrm{Id}_{D+1}}_{B})^{-1} \bar{\Phi}.$$

B can be expressed as:

$$B = \begin{bmatrix} \Phi\Phi^\top + I\lambda \mathrm{Id}_D & k\Phi\mathbf{1}_I \\ k\mathbf{1}_I^\top \Phi^\top & k^2 I + \lambda I \end{bmatrix}.$$

Let us define:

$$A = \Phi\Phi^\top + I\lambda \mathrm{Id}_D,$$

then using the block matrix inversion formula, we can get an explicit expression for B^{-1}:

$$B^{-1} = \begin{bmatrix} A^{-1} + k^2 A^{-1}\Phi\mathbf{1}_I u^{-1}\mathbf{1}_I^\top \Phi^\top A^{-1} & -kA^{-1}\Phi\mathbf{1}^I u^{-1} \\ -ku^{-1}\mathbf{1}_I^\top \Phi^\top A^{-1} & u^{-1} \end{bmatrix},$$

where

$$u = (k^2 + \lambda)I - k^2 \mathbf{1}_I^\top \Phi^\top A^{-1} \Phi \mathbf{1}_I.$$

Then we have that

$$B^{-1}\bar{\Phi} = \begin{bmatrix} A^{-1}\Phi + \frac{k^2}{u}A^{-1}\Phi\mathbf{1}_I\mathbf{1}_I^\top\Phi^\top A^{-1}\Phi - \frac{k^2}{u}A^{-1}\Phi\mathbf{1}_I\mathbf{1}_I^\top \\ -\frac{k}{u}\mathbf{1}_I^\top\Phi^\top A^{-1}\Phi + \frac{k}{u}\mathbf{1}^\top \end{bmatrix},$$

and

$$\begin{aligned}\bar{\Phi}^\top B^{-1}\bar{\Phi} = \Phi^\top A^{-1}\Phi + \frac{k^2}{u}[&\Phi^\top A^{-1}\Phi\mathbf{1}\mathbf{1}_I^\top\Phi^\top A^{-1}\Phi \\ &- \Phi^\top A^{-1}\Phi\mathbf{1}_I\mathbf{1}_I^\top - \mathbf{1}_I\mathbf{1}_I^\top\Phi^\top A^{-1}\Phi + \mathbf{1}_I\mathbf{1}_I^\top].\end{aligned}$$

If we write the scalar s as $s = \mathbf{1}_I^\top\Phi^\top A^{-1}\Phi\mathbf{1}_I$, we compute:

$$\begin{aligned}\bar{\Phi}^\top B^{-1}\bar{\Phi}\mathbf{1}_I = \Phi^\top A^{-1}\Phi\mathbf{1}_I + \frac{k^2}{u}[&s\Phi^\top A^{-1}\Phi\mathbf{1}_I \\ &- I\Phi^\top A^{-1}\Phi\mathbf{1}_I - s\mathbf{1}_I + I\mathbf{1}_I],\end{aligned}$$

which can be factorized as follows:

$$\bar{\Phi}^\top B^{-1}\bar{\Phi}\mathbf{1}_I = \Phi^\top A^{-1}\Phi\mathbf{1}_I + \frac{k^2(I-s)}{u}\left[\mathbf{1}_I - \Phi^\top A^{-1}\Phi\mathbf{1}_I\right].$$

We observe that we can factor out k from the experssion of u:

$$u = k^2((1 + \frac{\lambda}{k^2})I - s),$$

and then compute $\bar{Q}\mathbf{1}_I$ as:

$$\bar{Q}\mathbf{1}_I = \mathbf{1}_I - \Phi^\top A^{-1}\Phi\mathbf{1}_I - \frac{I-s}{I-s+\frac{I\lambda}{k^2}}\left[\mathbf{1}_I - \Phi^\top A^{-1}\Phi\mathbf{1}_I\right],$$

which in turn can be simplified as:

$$\bar{Q}\mathbf{1}_I = \frac{I\lambda}{k^2(I-s)+I\lambda}\left[\mathbf{1}_I - \Phi^\top A^{-1}\Phi\mathbf{1}_I\right].$$

Thus the result:

$$\lim_{k\to\infty} \bar{Q}\mathbf{1}_I = \mathbf{0}_I,$$

where $\mathbf{0}_I$ is the zero vector in dimension I.

Résumé

Les vidéos représentent des scènes complexes, comprenant des humains et des objets, illustrant les interactions entre ces derniers et leur enviorment. Les relations entre agents sont susceptibles d'évoluer dans le temps et les agents peuvent effectuer des "actions". La compréhension automatique des vidéos nécessite de correctement localiser les agents à la fois dans l'espace et dans le temps. De plus, il faut décrire les relations entre ces agents et leur evolution temporelle.

La vision par ordinateur repose souvent sur l'apprentissage supervisé, où des échantillons ettiquetés permettent d'apprendre les parametres d'un modèle. Cependant, pour des données aussi riches que la vidéo, l'ettiquetage est coûteux et compliqué. Les étiquettes symboliques ne sont pas suffisament riches pour encoder les interactions entre personnes, objets et scènes. Le langage naturel offre une puissance descriptive qui en fait un modalité pratique pour annoter des données vidéo. Nous proposons de mettre l'accent sur la modélisation conjointe de vidéo et de texte. Nous explorons des modèles joints dans le contexte de films avec leurs scripts de tournage. Le principal défi auquel nous sommes confrontés est que les scripts de films ne fournissent pas de localisation spatiale et temporelle des objets et des actions.

Nous présentons d'abord un modèle permettant d'associer automatiquement des étiquettes de personne et d'action aux detections de personnes dans les films. Le modèle utilise une fonction de coût de clustering discriminatif, et une supervision faible sous la forme de contraintes que nous obtenons à partir de scripts. Cette approche nous permet de localiser spatialement et temporelement les agents et les actions qu'ils effectuent dans la video, tel que décrit dans le script. Cependant, la localisation temporelle et spatiale est principalement due à l' utilisation de détections de personnes. Dans une seconde contribution, nous décrivons un modèle permettant d'aligner des phrases avec les images de la vidéo. La correspondance temporelle est obtenue en utilisant un modèle discriminatif sous contraintes d'ordre temporel. Ce modèle d'alignement est appliqué à deux ensembles de données : un composé de vidéos associées à un flux d'étiquettes ; un autre composé de vidéos et descriptions sous la forme d'étapes (recettes de cuisines par exemple).

Mots Clés

vision par ordinateur, reconnaissance d'actions, texte et vidéo, apprentissage faiblement supervisé

Abstract

Videos often depict complex scenes including people, objects and interactions between these and the enviorment. Relations between agents are likely to evolve in time and agents can perform actions. The automatic understanding of video data is complicated as it requires to properly localize the agents both in space and time. Moreover, one need to automatically describe the relations between agents and how these evolve in time.

Modern approaches to computer vision heavily rely on supervised learning, where annotated samples are provided to the algorihtm to learn parametric models. However, for rich data such as video, the labeling process starts to be costly and complicated. Also, symbolic labels are not sufficient to encode the complex interactions between people, objects and scenes. Natural language offers much richer descriptive power and is thus a practical modality to annotated video data. Therefore, in this thesis we propose to focus on jointly modeling video and text. We explore such joint models in the context of movies with associated movie scripts, which provide accurate descriptions of the pictured events. The main challenge that we face is that movie scripts do not provide precise temporal and spatial localization of objects and actions.

We first present a model for automatically annotating person tracks in movies with person and action labels. The model uses a discriminative clustering cost function, and weak supervision in the form of constraints that we obtain from scripts. This approach allows us to spatially and temporaly localize agents and the actions they perform, as described in the script, in the video. However, the temporal and spatial localization is due to the use of person detection tracks. Then, in a second contribution, we describe a model for aligning sentences with frames of the video. The optimal temporal correspondance is again obtained using a discriminative model under temporal ordering constraints. This alignment model is applied on two datasets: one composed of videos associated with a stream of symbolic labels; a second one composed of videos with textual descriptions in the form of key steps towards a goal (cooking recipes for instance).

Keywords

computer vision, action recognition, video and text, weakly-supervised learning

Numéro national de thèse :
xxxxx