INP Grenoble – ENSIMAG
École Nationale Supérieure dInformatique et de Mathématiques Appliquées de Grenoble
UFR Informatique et Mathématiques Appliquées de Grenoble

Rapport de stage de Master 2 et de projet de fin d'études

Effectué au sein de l'équipe LEAR, I.N.R.I.A., Grenoble

Action Recognition in Videos

Gaidon Adrien
3e année ENSIMAG – Option I.I.I.
M2R Informatique – spécialité I.A.

04 février 2008 – 04 juillet 2008

LEAR,
I.N.R.I.A., Grenoble
655 avenue de l'Europe
38 334 Montbonnot
France

Responsable de stage
Mme. Cordelia Schmid
Tuteur école
M. Augustin Lux
Jury
M. Roger Mohr
M. Yves Demazeau
M. James Crowley
M. Philippe Mulhem

Abstract

We construct a fully automatic large-scale system for visual retrieval of realistic action samples of different human action classes from TV-series and movies. We first propose a text-driven approach using the synchronization of transcripts and subtitles. In practice this yields a substantial part of irrelevant samples. We handle them by ranking the whole retrieved set by visual consistency using some partially incorrect training data. Our main contribution is to provide a new generic and fully automatic iterative training scheme for support vector regression that can handle such erroneous supervision to improve the ranking quality. We validate our approach by conducting experiments on realistic video data and showing that it performs better than existing state-of-the-art unsupervised and supervised methods.

Contents

1

Introduction

1.1 Human action recognition in videos

With the growing success of digital medias, satellite imaging and digital video streaming, the research effort in computer vision has intensified. Among the recent subjects, the goal of recognizing human actions in videos has recently become of high interest for the scientific community as well as for industry. Understanding the visual content of videos is a challenging task of computer vision. It requires efficient tools to process quickly many hours of videos, and very accurate classifiers that can handle the strong variability of filming conditions, background clutter, human appearances and behaviors. Its applications range from robotics to video-surveillance, or management of video collections.

The visual recognition and machine learning communities have made tremendous progress during the past decade, allowing fast and reliable object segmentation, localization and recognition in images. For instance, state-of-the-art methods are now capable of identifying faces in news photos [27], and recognizing classes of objects in complex natural scene images [42].

Due to this evolution, the problem of classifying actions in videos, i.e. deciding whether an action is performed in a video sample or not, has recently started to yield encouraging results [3, 21, 30, 32, 35, 36, 45, 46, 57] and to get more and more interest from the scientific community. Nevertheless, there are many problems that are still far from being solved.

The lack of standard realistic dataset of human actions is an obstacle to actual progress in the field of action recognition. A set of various and numerous realistic actions is necessary to correctly evaluate any action recognition framework and show its actual potential. Existing datasets for action recognition (e.g. the KTH actions [53]) provide usually few samples, only a small number of action classes and controlled and simplified settings. This is in contrast to action recognition in real-world videos where (1) people vary in expression, posture, motion and clothing; (2) the image undergoes perspective effects, camera motions and illumination variations and (3) occlusions and variation in scene surroundings occur.

Note that recognition and localization of actions in movies has recently been addressed in [35] for a limited but realistic dataset, i.e., manual annotation of two action classes. In [36], a more developed realistic dataset of action samples is "semi-automatically" retrieved from some movies, as long as associated subtitles and transcripts (which they manually synchronize) are available. The problem of such datasets, and of the methods to design them, is that the number of samples and classes is very limited.

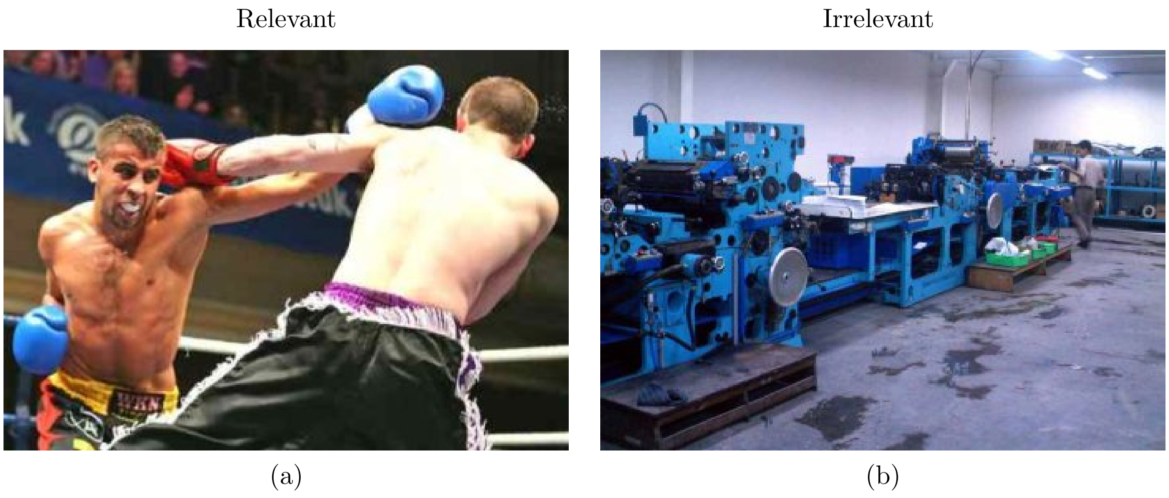

Figure 1.1: Two images resulting from the query “punching” on Google image search. (a) is a visually relevant sample, whereas (b) is visually irrelevant.

Manually annotating videos is time-consuming, because to localize actions in a clip of one hour, a human has to actually spend around one hour to do it. The incompressible cost of human annotation makes it impossible to be conducted on very large databases, like the web for instance. Therefore, an automatic annotation procedure is required. The counterpart to such an automatic process is that it is less precise than a human and thus returns many wrong labels. This limits the interest of using the retrieved videos to learn to recognize actions, because learners have only a limited robustness to noise [36].

Search engines (e.g. Google) usually provide an image search function, which is a typical example of limited automatic annotation tools. Multimedia document retrieval is a complex task which is getting more and more interest from both industry and research sectors. Especially since the huge increase of available multimedia data and the dazzling popular success of social networks and other sharing systems (e.g. *Youtube, Flickr*) on the web.

Though there have been many clear advances in semantic understanding of the visual content of videos and images, most of the currently used retrieval systems are only based on metadata attached to a multimedia resource: user-defined tags, captions of images, surrounding text, etc.. Text-driven-only retrieval procedures of video or images are often containing a lot of false matches [36].

Those irrelevant documents are troublesome, because, as there are usually much more irrelevant documents than relevant ones in the whole collection that is searched, they can overwhelm the useful and desired information. For a great part, those errors cannot be efficiently removed with textual information only, because the main cause is the discrepancy between textual and visual content, that can be completely unrelated. For instance, in figure 1.1, we show two images resulting from the query “punching” on Google image search. The irrelevant sample was retrieved because it is a “punching machine”, which is visually completely different than an actual punching action.

Therefore, researchers tried to use the content of the retrieved images and videos, and not just the surrounding available textual information, to take into account the specificities of *visual* information retrieval. Several works [52, 8, 25, 40] automatically build collections of images for specific object classes. The idea is to search for images on the web based on textual queries and then to re-rank the images based on visual information in order to increase precision. Similarly, approaches for naming characters in images [7, 47] and video [23] determine the correspondence between faces and names. They start with a simple form of text processing, i.e., extract the named entities. In parallel they extract faces and their description from the images and then attempt to put faces and names in correspondence.

1.2 Our contribution

This report addresses two main problems. The first one is the creation of a large and realistic video dataset for human action recognition, using textual information in an automatic manner. The second issue is to automatically improve the quality of the retrieval process. We therefore tackle the issue of ordering a set of videos containing some samples related to a certain action, and some unrelated ones.

1.2.1 Automatic retrieval of human actions in movies

First, we address the lack of realistic and large-scale human actions dataset. In order to automatically produce large amounts of correctly annotated real-world video examples, two options are available: use an accurate annotation technique or filter out wrong annotations. We propose to combine both in an automatic retrieval system, returning correctly annotated samples of human action samples.

We first use textual information in order to do fully automatic text-based retrieval of action samples, whereas the method in [36] requires human annotation of some of the textual data. Like in [23, 36], we use transcripts and subtitles of movies and TV series in our experiments. More precisely, we use the 144 40 minutes episodes of the famous TV series *Buffy the vampire slayer* (see figure 1.2).

Though yielding less errors than other similar reported methods [23, 36], there is still a high level of erroneously labeled data. We therefore use a post-processing step, using visual information to detect inconsistent samples. We use show that this problem can be solved using outlier removal techniques, and conduct experiments using two standard unsupervised outlier removal methods to automatically handle retrieval errors.

1.2.2 Automatic visual ranking of videos

We then propose a more general approach to handle errors in a set of retrieved samples. We learn to rank samples in such a way that errors lie in the lowest ranks. Ranking can be used in retrieval systems to present results in a proper order. They can be used to improve classification, because the ranks provide some supplementary information. Finally, the rankings can be used for outlier removal, by considering that all the samples after a certain cut-off rank can be regarded as outliers. It can therefore be used for a visual cleaning step of partially incorrectly labeled data retrieved from textual information.

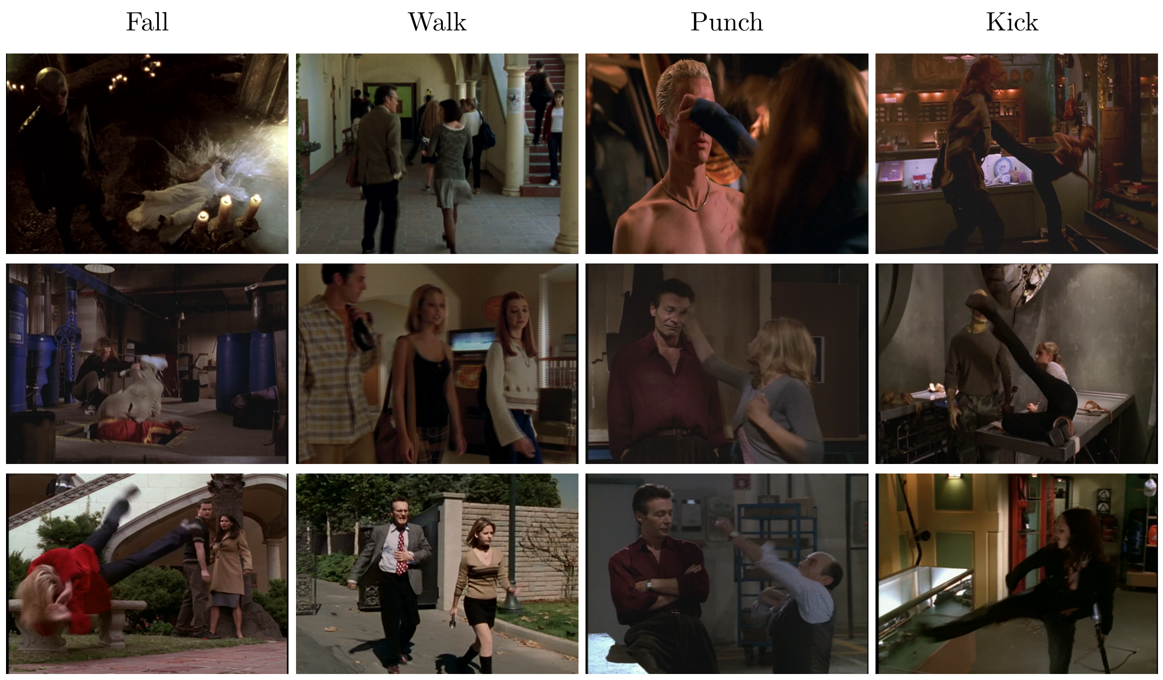

Figure 1.2: Key frames of sample video clips extracted from "Buffy the Vampire Slayer" TV series. Note that the presented video samples corresponding to four queried human actions were automatically retrieved with our system.

We show that outlier removal techniques can be successfully adapted to the ranking problem. Furthermore, we improved this visual ranking process by considering some easily available imperfect supervision. For instance, partially correctly annotated data can be automatically obtained using the retrieved set as positive instances relevant to the query, and random samples from the whole collection as negatives, i.e. examples of what is not-relevant, like the retrieval errors.

We propose to apply regression techniques to the problem of automatically ranking a set of video samples, containing some irrelevant ones, by their consistency. The goal is to have a ranked list of samples where errors are ranked lower than relevant samples. Our main contribution is to provide a new iterative training scheme for the application of support vector regression [55] to ranking by consistency. Our approach allows this learning algorithm to automatically distinguish between good and bad training samples, and therefore, avoids being misled by errors and even use them to improve.

1.3 Structure of the report

In chapter 2 we provide some background material on the computer vision and machine learning tools we will use or refer to in this report. We also describe our evaluation framework.

In chapter 3, we explain the principles of our text-based retrieval techniques for mining actions in movies. We also discuss the problem of the high proportion of wrong results, and show that a visual filtering step is necessary.

In chapter 4, we first place the problem of learning to rank with errors. We provide two solutions. The first one makes use of unsupervised outlier removal techniques to compute a certain degree of “outlierness”, or inconsistency of samples. The second approach addresses the limitations of the unsupervised paradigm, and provides a novel way to automatically deal with imperfect label information, without assuming a certain model on the errors.

Finally, in chapter 5, we provide a conclusion on the current achievements, a discussion on the concept of visual consistency and we describe our future work.

2

Background material

In this chapter, we provide knowledge on some computer vision and machine learning tools we use. In section 2.1 we detail how we represent videos using the *bag-of-features* approach. In section 2.2 we describe the main machine learning techniques that we use, the Support Vector Machine and the densest component search in a graph. Finally, in section 2.3 we give details about the experimental setup we use to validate our research.

2.1 Representation of visual content using local features

Many different approaches have been tried to address the difficult problem of human action recognition in videos, but very few attempted to recognize actions in realistic videos. This section presents our approach for action representation and classification. It builds on existing bag-of-features approaches for video description [21, 45, 53] and uses recent extensions of advances in static image classification to videos [10, 38, 41]. We build one the current best reported results [36] of application of *Bag-of-Features* to realistic action recognition.

2.1.1 Space-time features

Building upon the experience of the object recognition domain, sparse local features, very efficient for static images [58], have been successfully adapted to the spatio-temporal domain and shown good performance for action description [21, 30, 45, 53]. They provide a compact video representation and tolerance to background clutter, occlusions and scale changes. The idea is to first detect interest points in the video and then describe the region around these key points, so that we can efficiently compare them.

Here, we use the extension of the Harris corner detector provided in [34]. As Lazebnik et al. [38] showed that a spatial pyramid, i.e., a coarse description of the spatial layout of the scene, improves recognition, we use a multi-scale approach, in the same way as in [36]. We extract features at different levels of spatio-temporal scales (σ_i^2, τ_j^2) with $\sigma_i = 2^{(1+i)/2}, i = 1, ..., 6$ and $\tau_j = 2^{j/2}, j = 1, 2$. Interest points detected for two frames with human actions are illustrated in figure 2.1.

To characterize motion and appearance of the detected regions we use a coarse histogram of oriented gradients of space-time volumes in the neighborhood of detected points. The size

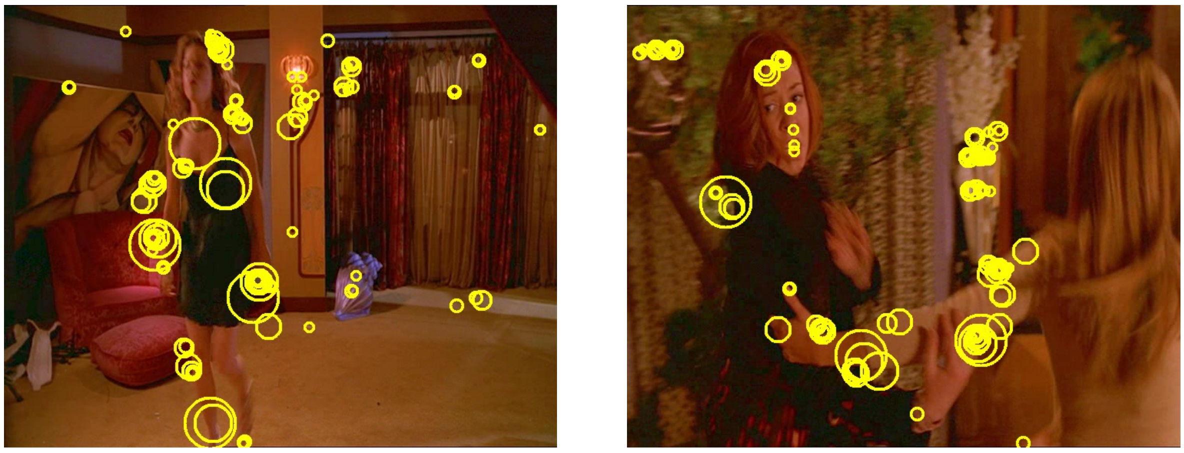
Figure 2.1: Space-time interest points detected for two video frames with actions walk (left) and hug (right).

of each volume $(\Delta_x, \Delta_y, \Delta_t)$ is related to the detection scales by $\Delta_x, \Delta_y = 2k\sigma$, $\Delta_t = 2k\tau$. Each volume is subdivided into a (n_x, n_y, n_t) grid of cuboids. We use parameter values $k = 9$, $n_x, n_y = 3$, $n_t = 2$. The histogram descriptors form our local spatio-temporal features, cf. figure 2.2. It is very close in spirit to the SIFT descriptor (proved to be very efficient for object recognition in images [43]).

2.1.2 Spatio-temporal bag-of-features

In analogy with [58], given a set of features, which amount depends on the number of key points detected, we build a bag-of-features representation. It is based on the use of a vocabulary of visual words to represent the set of features of a video as a histogram of visual word frequency over the vocabulary.

The visual vocabulary is constructed using the *k-means* clustering technique (where k, the number of clusters, is the desired number of visual words) on the set of features coming from the videos. In our experiments we cluster a subset of $100k$ features randomly sampled from the training videos' representations. The number of clusters is set to $k = 1000$, which has shown empirically to give good results and is consistent with the values used for static image classification.

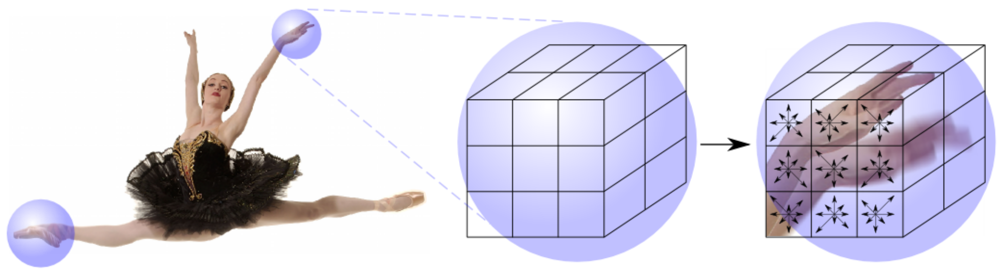
Figure 2.2: Extraction of a spatio-temporal descriptor for a space-time volume.

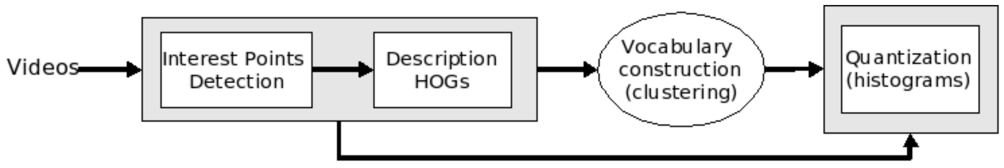
Videos
Interest Points Detection
Description HOGs
Vocabulary construction (clustering)
Quantization (histograms)

Figure 2.3: Work-flow of the spatio-temporal bag-of-features approach to action recognition.

The bag-of-features representation is achieved by simple feature vector quantization. Each feature is assigned to the closest (we use Euclidean distance) vocabulary word (cluster centroid) and the video is sparsely represented by the histogram of its visual word occurrences.

The global work-flow of the spatio-temporal bag-of-features approach to action recognition is depicted in figure 2.3.

Further details about interest points, descriptors and spatio-temporal bag-of-features can be found in [34].

2.2 Machine learning methods

We now describe the machine learning tools suited to extract the power of local representations of visual content in the form of Bag of Features. We first define the concepts of supervised and unsupervised learning. We then provide a complete background on the main tool we will use, the support vector machine (SVM), its different extensions and other tools derived from it, one-class SVM and support vector regression (SVR). Finally, we give some details about the algorithm we will use to find the densest component of an undirected graph.

2.2.1 Supervised and unsupervised paradigms

We consider two different families of machine learning algorithms, differentiated by the type of data they deal with.

When there is no other information available on the samples considered than just their representation, the algorithms that are trying to directly infer some interesting characteristics are said to belong to *unsupervised learning*. For instance, clustering, the task of grouping samples with similar features together in clusters, is an unsupervised learning problem.

On the opposite, supervised algorithms use data samples annotated by their true labels. This ground truth is usually obtained by human annotation in order to have accurate labels, i.e. a strong and reliable supervision. But labels can also be obtained by automatic annotation procedures, which generally lack in precision and therefore produce a weak, or imperfect, supervision. Supervised classifiers try to learn from these training labels and predict those of new features according to this information. Classification is often addressed within this perspective.

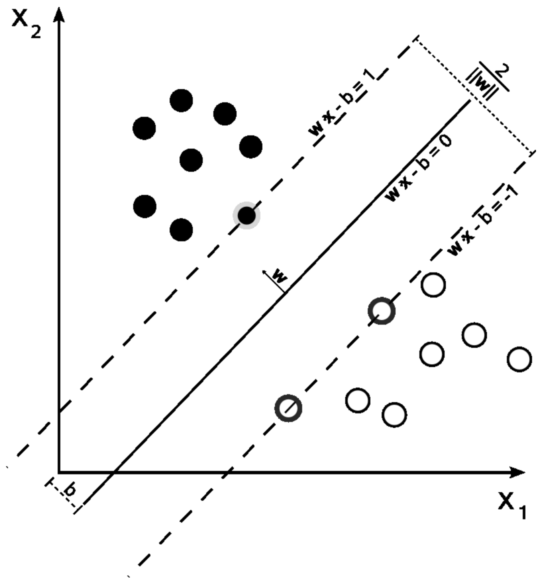

Figure 2.4: Maximum-margin hyperplane and margins for a SVM trained with samples from two classes. Samples on the margin are called the support vectors.

2.2.2 Support Vector Machines

Principle

Support Vector Machines [55, 56] (SVM) are large margin classifiers that have shown high efficiency (among many other fields) in histogram-based image classification [14] and human action recognition using local features [53, 36].

For a binary classification problem, the basic idea of SVMs is to find the optimal separating hyperplane between a set of labeled training samples, formed of positives (samples belonging to class of interest) and negatives (not in the class). The goal is to have all of the positives on one side of the hyperplane, the negatives on the other side, with an as large as possible margin separating them. The larger the margin, the better the generalization is expected to be [56]. The closest points lying on each side of the margin are called *support vectors* and are enough to characterize the optimal hyperplane. See figure 2.4 for a graphical illustration.

Formulation

In the following, we present the optimization problem corresponding to the training of the SVMs we use.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1..n}$ be the training data, where the y_i is indicating the class to which the point $\mathbf{x}_i$ belongs.

We want to find the maximum-margin hyperplane $\mathbf{w} \cdot \mathbf{x} - b = 0$ (where $\mathbf{w}$ is a normal vector and b is the offset of the hyperplane from the origin), which divides the points having $y_i = +1$ (the positives) from those having $y_i = -1$ (the negatives).

We want to find $\mathbf{w}$ and b that maximize the margin, or distance between the parallel hyperplanes that are as far apart as possible, while still separating the data. These hyperplanes can be described by the equations: $\mathbf{w} \cdot \mathbf{x} - b = 1$ and $\mathbf{w} \cdot \mathbf{x} - b = -1$.

Note that if the training data is linearly separable, we can select the two hyperplanes of the margin in a way that there are no points between them and then try to maximize their distance. The distance between these two hyperplanes is $2/||w||$, so we want to minimize $||w||$. As we also have to prevent data points falling into the margin, we add the following constraint: $y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1$, for each $i = 1..n$.

We can put this together to get the quadratic optimization problem:

$$min_{\mathbf{w},b} \ \frac{1}{2}||\mathbf{w}||^2 \text{ subject to: } y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \quad \text{for all } i = 1..n \tag{2.1}$$

This problem is in practice solved using its dual formulation with Lagrange multipliers α_i:

$$\begin{array}{rc} max_\alpha & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j \\ \text{subject to:} & \alpha_i \geq 0 \ , \ i = 1..n \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{array} \tag{2.2}$$

Non-linear C-SVMs

The first extension of SVMs we use are *slack variables* [19], noted ξ_i. They allow the possibility of examples being misclassified (i.e. on the wrong side of the margin), with a penalty factor C.

The second improvement is the use of the *kernel trick* in order to have non-linear SVMs [11]. These non-linear classifiers are obtained by mapping the input features into a higher-dimensional feature space. Thanks to the dual formulation of the optimization problem, the mapping Φ needs not to be known, only the inner product, called a kernel $K(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$, of this new space is required (this is the *kernel trick*).

We obtain the following primal problem:

$$\begin{array}{rc} min_{\mathbf{w},b,\xi} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to:} & y_i(\mathbf{w}^t\Phi(\mathbf{x_i}) - b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \ , \ i = 1..n \end{array} \tag{2.3}$$

and its equivalent dual, where the slack variables disappear and only K is needed:

$$\begin{array}{rc} max_\alpha & \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to:} & 0 \leq \alpha_i \leq C \ , \ i = 1..n \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{array} \tag{2.4}$$

Support vectors are the $\mathbf{x_i}$ for which $\alpha_i \neq 0$. The corresponding decision function is:

$$f(\mathbf{x}) = sign\Big(\sum_{i=1}^n \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) - b\Big) \tag{2.5}$$

The advantage of such SVMs, called C-SVMs, are their high generalization capabilities (thanks to slack variables), their discrimination power (thanks to the choice of an appropriate feature space thanks to the kernel), and the fact that the solution of the optimization problem is compact (only support vectors are needed) and can be quickly computed (the dual can be solved using quadratic programming techniques, we use Sequential Minimal Optimization [48]).

Kernel

We use a χ^2 kernel to measure the distance between bag-of-features. It is based on a generalized Gaussian kernel defined as:

$$K(H_i, H_j) = \exp\big(-\frac{1}{A} D(H_i, H_j)\big) \tag{2.6}$$

where $H_i = \{h_{in}\}$ and $H_j = \{h_{jn}\}$ are the histograms, i.e., bags-of-features for samples i and j. $D(H_i, H_j)$ is the χ^2 distance defined as:

$$D_c(H_i, H_j) = \frac{1}{2}\sum_{n=1}^{V} \frac{(h_{in} - h_{jn})^2}{h_{in} + h_{jn}} \tag{2.7}$$

with V denoting the vocabulary size. The parameter A is the mean value of the distances between all training samples [61].

ν-SVM

In practice, we use a slightly modified version of the C-SVM, called ν-SVM because it replaces the C parameter with a new $\nu \in (0, 1]$ parameter. This parameter is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors [56], and must be carefully chosen.

The formulation is the following:

$$\begin{array}{rc} min_{\mathbf{w},b,\xi} & \frac{1}{2}||\mathbf{w}||^2 - \nu b + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to:} & y_i \mathbf{w}^t \Phi(\mathbf{x_i}) \geq b - \xi_i, \\ & \xi_i \geq 0 \ , \ i = 1..n \end{array} \tag{2.8}$$

for the primal. Its equivalent dual is:

$$\begin{array}{rc} min_{\alpha} & \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to:} & 0 \leq \alpha_i \leq \frac{1}{n} \ , \ i = 1..n \\ & \sum_{i=1}^{n}\alpha_i \geq \nu \ , \ \sum_{i=1}^{n}\alpha_i y_i = 0 \end{array} \tag{2.9}$$

The corresponding decision function is:

$$f(\mathbf{x}) = sign\Big(\sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) - b\Big) \tag{2.10}$$

One-class SVM

We also use an unsupervised version of SVM that was introduced to estimate the distribution's support of high dimensional data [51].

Given training vectors $\mathbf{x_i}$, $i = 1..n$, without any class information, the one-class SVM maps the data into a higher dimensional feature space and then tries to use a hyper-sphere to describe the data in this feature space. The goal is to have a sphere as small as possible while at the same time, including most of the data.

Let $\nu \in (0, 1]$ (which can be interpreted in the way as for the ν-SVM, a mapping Φ and the corresponding kernel K, the primal form is:

$$\begin{array}{rc} min_{\mathbf{w},b,\xi} & \frac{1}{2}||\mathbf{w}||^2 - b + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \\ \text{subject to:} & \mathbf{w}^t\Phi(\mathbf{x_i}) \geq b - \xi_i, \\ & \xi_i \geq 0 \ , \ i = 1..n \end{array} \tag{2.11}$$

and its equivalent dual:

$$\begin{array}{rc} min_{\alpha} & \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to:} & 0 \leq \alpha_i \leq \frac{1}{\nu n} \ , \ i = 1..n \\ & \sum_{i=1}^{n}\alpha_i = 1 \end{array} \tag{2.12}$$

The confidence values are:

$$f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i K(\mathbf{x_i}, \mathbf{x}) - b \tag{2.13}$$

and the decision function (the hyper-sphere's boundary) is $sign\big(f(\mathbf{x})\big)$.

Support Vector Regression

Finally, we use the support vector machine's application to regression [55, 22, 54] (ϵ-SVR). The goal of ϵ-SVR [55] is to find a function $f(\mathbf{x})$ that has at most ϵ deviation from the training targets y_i, and at the same time is as flat as possible. In other words, errors in a tube of width ϵ around f are not counted, but any sample outside of the tube will have a certain cost (cf. figure 2.5 for the illustration of this ϵ-sensitive loss function). The C parameter determines the trade-off between the flatness of f and the amount up to which deviations larger than ϵ are tolerated.

The formulation of the optimization problem is the following:

$$\begin{array}{rc} min_{\mathbf{w},b,\xi,\xi^*} & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{subject to:} & (\mathbf{w}^t\Phi(\mathbf{x_i}) - b) - y_i \leq \epsilon + \xi_i, \\ & y_i - (\mathbf{w}^t\Phi(\mathbf{x_i}) - b) \leq \epsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0 \ , \ i = 1..n \end{array} \tag{2.14}$$

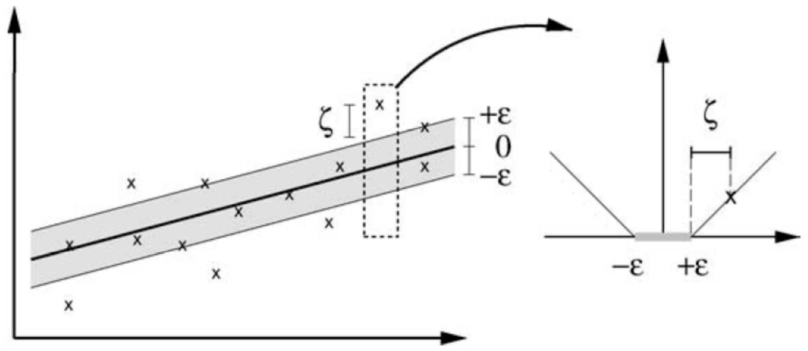

Figure 2.5: Soft margin loss for a linear ϵ-SVR.

By solving its equivalent dual:

$$\begin{array}{ll} min_{\alpha,\alpha^*} & \frac{1}{2}\sum_{i,j}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)K(\mathbf{x_i},\mathbf{x_j}) \\ & +\epsilon\sum_{i=1}^n(\alpha_i+\alpha_i^*)+\sum_{i=1}^n y_i(\alpha_i-\alpha_i^*) \\ \text{subject to:} & 0\le\alpha_i\le C \ , \ i=1..n \\ & \sum_{i=1}^n\alpha_i y_i=0 \end{array} \tag{2.15}$$

The corresponding regression function is:

$$f(\mathbf{x})=\sum_{i=1}^n(\alpha_i^*-\alpha_i)K(\mathbf{x_i},\mathbf{x})-b \tag{2.16}$$

2.2.3 Finding the densest component of a graph

We will also have to deal with graph structures. We will especially look for dense components in an undirected graph.

Let $\mathbf{x_i}$, $i=1..n$ be our training vectors and K a similarity measure between training vectors, i.e. the greater it is, the closer the samples are (it is the opposite for a distance measure between the training vectors).

First, we construct the complete undirected graph $G=(V,E)$, where the nodes $V=\{i=1..n\}$ represent the training vectors and the edges $E=\{(i,j),i,j\in V\}$ are weighted by the similarity between the concerned nodes $w_{i,j}=K(\mathbf{x_i},\mathbf{x_j})$.

The degree of a vertex v is then $deg(v)=\sum_{j\in V}w_{v,j}$. The average degree of a node in G is $avg\_deg(G)=\frac{\sum_{v\in V}deg(v)}{card(V)}$, where $card(V)$ is the cardinal of V.

For $S\in V$, we define the subgraph induced by S, $G(S)=(S,E(S))$, where $E(S)=\{(i,j),i,j\in S\}$. In the case where $w_{i,j}=1$ for all $i=1..n$, density is usually defined by $f(S) = \frac{card(E(S))}{card(S)}$. $w_{i,j} = 1$ implies $\sum_{v \in S} deg(v) = 2card(E(S))$. We can then deduce $avg\_deg(G(S)) = 2\frac{card(E(S))}{card(S)}$ which shows that the density can also be written:

$$f(S) = \frac{1}{2} avg\_deg(G(S)) \tag{2.17}$$

By analogy, we use the definition of eq. 2.17 for the case where weights are similarity measures between nodes.

We then use the greedy 2-approximation algorithm of Charikar [16] to find the densest sub-graph of G, i.e. $argmax_{S \subseteq V} f(S)$. It consists in iteratively deleting the node with minimal degree and updating the graph until all nodes are removed. The densest component is then the sub-graph obtained at the iteration where the density (i.e. the average degree, cf. eq. 2.17) is maximal.

2.3 Evaluation and notations

In order to evaluate the efficiency of ranking algorithms and binary classifiers, we consider two performance measures, inherited from the field of information retrieval. Those measures are well established in pattern recognition papers and used in some computer vision challenges (like the PASCAL VOC challenge).

In a binary classification problem (i.e. knowing if a sample belongs to a certain class - what we call a positive sample - or not, i.e. a negative), classifiers often output a confidence value that is compared to a threshold in order to make a proper decision (cf. eq. 2.5 for the case of SVMs). Judging from the ground truth (i.e. the actual label information), the labeled sample resulting from the output decision can be either:

- a true positive (TP): the sample is classified as being from the class and it actually is, according to the ground truth;
- a false positive (FP): the sample is wrongly classified as positive;
- a true negative (TN): the sample is correctly classified as a non-class sample;
- or a false negative (FN): the sample is actually positive, but is classified as negative.

The total number of positives is $P = TP + FN$, and of negatives $N = TN + FP$.

Precision-recall curves

They are 2-D plots of recall versus precision, when using different thresholds on the confidence values or rankings, where these two measures are defined as follows:

- **recall**: $\frac{TP}{TP+FN}$, the ratio of true positives correctly classified as such;
- **precision**: $\frac{TP}{TP+FP}$, the ratio of true positives among all the samples classified as positive.

The bigger the area under this curve, the better the algorithm performs, the perfect result being 100% precision at 100% recall, meaning perfect classification of all the samples.

Average precision

Average precision is commonly used evaluation criteria for retrieval, ranking and classification systems, since it captures recall and precision at the same time. It can be defined as the average of precisions computed at all recall rates. It is actually a measure of the area under a precision-recall curve. As it is a scalar, it is easier to use for comparison purposes than precision-recall curves.

For stability and coherence reasons, it is often estimated (like in the case of the PASCAL VOC Challenge) by the following formula:

$$AP = \frac{1}{11}\sum_{i=0}^{11} max\big(precision(recall \geq r_i)\big), \text{ where } \mathbf{r} = [0, 0.1, 0.2, ..., 1.0] \tag{2.18}$$

3

Text-based retrieval of video samples

In this chapter, we provide a new human actions retrieval system, that uses textual information present from subtitles and transcripts of movies and TV-series to automatically extract video clips containing actions.

In section 3.1, we justify the use of movies and TV-series as interesting sources of realistic video data and we explain our text-based retrieval approach. In section 3.2, we describe in details all the different steps of our work-flow: extracting subtitles from DVDs using OCR techniques, synchronizing subtitles and transcripts and finally proper text parsing to find actions. Finally, in section 3.3 we demonstrate the practical applicability and efficiency of our method on a large collection of realistic videos: all the episodes of the TV-series *Buffy the vampire slayer*.

3.1 Exploration of actions in movies

3.1.1 Movies and TV-series as a source of human actions

Realistic datasets of videos are crucial for the progress and evaluation of the research on human action recognition. They should contain multiple action classes, happening in many different natural scenes, executed by different actors within different conditions, . . . In general, they should be diverse and complete in order to give insights on the strengths and weaknesses of any recognition algorithm, and not to bias any evaluation or comparison attempt. The development of image datasets with such properties is one of the main causes for the recent outstanding evolution of object recognition in still images. Therefore, there is a need for such video datasets. That is what we try to address.

Movies and TV series are great sources of realistic action samples. They provide a lot of different characters, scenes (indoors, outdoors), camera perspectives and motions, lighting conditions, type of actions, etc. Furthermore, they are often available together with reliable textual information: subtitles and transcripts. This textual information can be used to infer what is actually happening at a certain time of the video. We go further than [36] and show that this text-driven retrieval of action samples can be done in a fully automatic way, thanks to clever text parsing algorithms.

We choose TV-series in order to perform large-scale retrieval and we show results on *Buffy the vampire slayer* TV series (because it displays many different characters, situations and actions) in order to prove the applicability of our concepts.

3.1.2 Overview of our approach

We present an approach for automatically exploring the significant actions in videos. The purpose of such an approach is two-fold: (1) we can quickly obtain large amounts of annotated real-world video data for learning actions and (2) we can automatically explore the video, i.e., determine which actions occur often within a video and sample their most characteristic video sequences.

Our approach differs from existing approaches [23, 35, 36] in that (1) it is completely unsupervised, (2) it can retrieve very large amounts of video samples. It also goes beyond static image, i.e., finds examples for action classes in real-world videos. We first deal with a fully automatic robust subtitle and transcript alignment method. In order to find verbs corresponding to actions, the transcripts are parsed with a linked grammar. Given a sentence, the parser assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The parser also produces a "constituent" representation of a sentence (showing noun phrases, verb phrases, etc.). Given the verbs, we check their semantics in WordNet. WordNet is a semantic lexicon for the English language, containing around 150,000 words and a total of 207,000 word-sense pairs. It distinguishes between nouns, verbs, adjectives and adverbs. Note that not all verbs denote actions and WordNet allows to prune only some of the verbs describing occurrence or state of being. As we focus on visual actions, we assume in the following that all verbs correspond to actions and treat the non-action verbs in the same way as non-visual action classes.

After having retrieved the verb semantics from WordNet, we measure action occurrence frequencies over the dataset to find the significant ones. A verb chart of the results of the parsing on the Buffy transcripts can be found in figure 3.1. The video sequences corresponding to the individual instances of verbs are obtained by aligning the transcripts (in which the verbs appear) with the video based on the subtitles [23].

In the following, we give details about the extraction of the video clips with the corresponding labels.

3.2 Mining actions

Our goal is to automatically obtain a sufficient number of visual action samples (video clips) corresponding to the semantic query provided by the user. It is therefore important to have a large set of videos and the processing has to be completely automated—we don't want, for instance, to adapt the alignment procedure to each video. In the following we first describe how to extract the subtitles. We then present the method for aligning subtitles and transcripts to infer the time information for scripts. Finally, we present our method for extracting verbs.

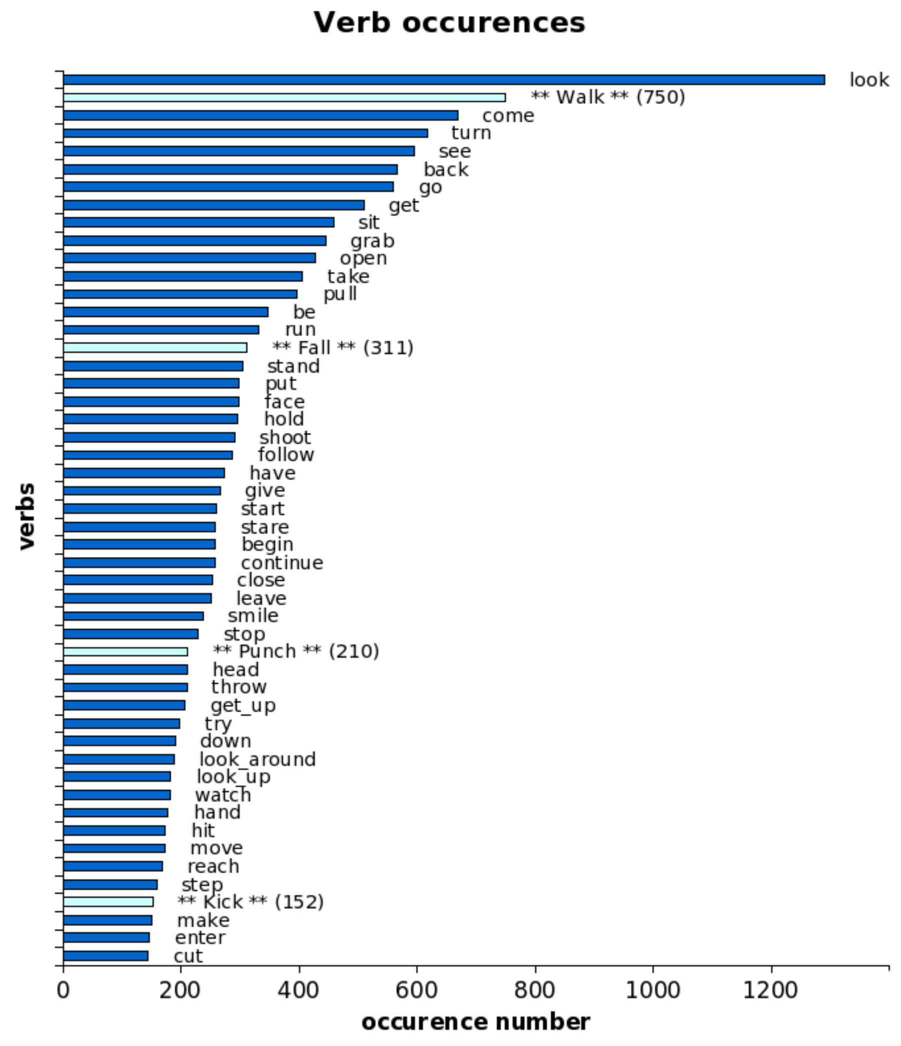

Figure 3.1: The most frequent verbs in the Buffy transcripts.

3.2.1 Subtitle extraction

Most (if not all) videos are released today with subtitles. The subtitle stream is time-synchronized with the video stream and contains the textual representation of dialogs and occasionally environmental sounds. In our framework we use the subtitles as an intermediate link between the video stream (referenced by time) and the transcript (referenced by dialog lines). However, obtaining appropriate subtitles for such a task is not trivial.

Firstly, the subtitles have to be well synchronized with the videos. Even though subtitles are in many cases available online, the subtitle version might be appropriate for a different video release. Any kind of manual matching of the videos with the available subtitles is prohibitive in large-scale setups. As digital versatile discs (DVDs) are released together with their version of the subtitles in many languages, they usually allow to access the video stream together with a synchronized English subtitle stream. Unfortunately, the DVD format encodes subtitles as images. This conflicts with a second requirement for our subtitle stream — the textual representation of the dialogs is required for matching with the scripts.

A solution is to use an optical character recognition (OCR) algorithm of good quality. Note that we first tried off-the-shelf popular OCR software like GOCR which did not deliver satisfying results. Even if optical character recognition is a well studied problem and recognition of printed text with no noise may seem to be a trivial problem, we often encountered subtitle lines with overlapping characters. According to Chellapilla and Simard [17], one of the most difficult elements of an OCR system is character segmentation—this might explain the difficulties we have encountered. The OCR system that gave satisfying results was Tesseract [50] developed by Hewlett-Packard and Google. Note that the system can be tuned to a specific font or dictionary. However, as we head for a fully automatic system, we did not use this option, but resorted to the standard pre-trained setup. This resulted in occasional recognition errors, but we can handle those gracefully in the later stages of our system. As a result, we have managed to automate previous approaches based on human-aided subtitle extraction [23] or manual matching of resources available online [36] that turned out to be not applicable to large-scale setups.

We have successfully extracted video streams and the corresponding OCRed English dialog lines from 39 “Buffy the Vampire Slayer” DVDs. Except media handling, i.e., inserting the DVDs into the computer, the processing for all 144 Buffy episodes was fully automatic.

3.2.2 Aligning subtitles and transcripts

Transcripts describe the content of videos. They cover both character dialogs and plot events describing the content of the video. For popular film series complete sets of transcripts are available for download from fan sites. Unfortunately, even if the available transcript set is complete, transcripts are usually written by different fans and are far from uniform. An automatic conversion to plain-text format is possible with automated tools like textual web-browsers, but the text layout varies and it is often difficult to localize dialog lines without human-based algorithm adaptation.

Unlike previous work [23, 36], we avoid layout interpretation and head straight for text matching. We have tried the popular word-based dynamic time warping algorithm [44], but it turned out to be surprisingly ineffective when dealing with OCR recognition “typos” in the subtitles. We have therefore moved to character-level text stream matching, taking a related dynamic programming approach.

We first find the longest common subsequence (LCS) in two character streams formed from OCRed dialogs and plain-text scripts. This is a well studied dynamic programming problem and the only problem is the $O(mn)$ space complexity of a naive approach. However, as a $O(m+n)$ space complexity algorithm exists [28], we can effectively discover LCSes, even for long videos. Note that the longest subsequence match does not guarantee unique matches, but as the matched character streams have a significant overlap, in practice most character matches are stable. After checking for match uniqueness, we transfer the timestamps from subtitles characters to transcript characters and extend those to sentences. The timestamps for sentences without matches can be inferred from timestamps of neighboring timed sentences, see fig. 3.2.

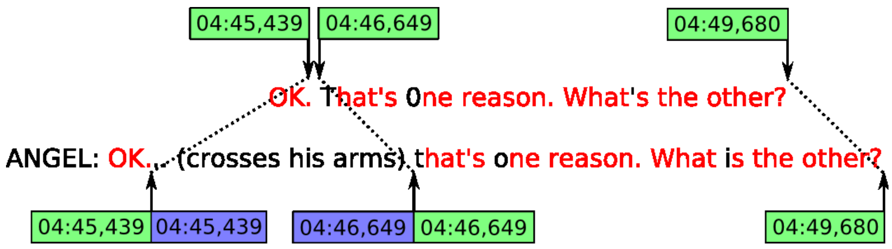
04:45,439
04:46,649
04:49,680
OK. That's One reason. What's the other?
ANGEL: OK... (crosses his arms) that's one reason. What is the other?
04:45,439
04:45,439
04:46,649
04:46,649
04:49,680

Figure 3.2: Aligning subtitles (above) and transcript (below). Characters belonging to the longest common subsequence are in red. Matches allow to transfer timestamps and the timeframe for "(Angel) crosses (arms)" action can be inferred. Note the robustness to OCR errors and dialog line variability.

Note that as a side effect of the applied procedure we obtain a good identification of the dialog sections in the script. In the following we focus on the parts of the script that fall between the dialogs, but the matched sentences, i.e., dialogs could be used in applications like automatic character annotation [23].

We have successfully inferred synchronization timestamps for almost 50000 non-dialog script sentences. The precision of the timestamps depends on the frequency of dialogs in a video, but over 50% non-dialog sentences could be localized within the time range of 10 seconds. Except transcript download, all the processing was again fully automatic. The synchronization of a 40-minutes-long episode took about 30 seconds on a personal computer.

3.2.3 Text parsing

Given sentences describing short video samples, we can automatically retrieve video sequences corresponding to a semantic query. A naive approach would be to perform textual queries. We have found, however, that with the widely available processing tools for English language we can interpret the textual description of video samples and formulate queries already at the semantic level.

```
    +----------------Xp----------------+
    |                +------Os------+      |
    +----Wd--+--Ss--+      +--Ds--+      |
    |           |        |       |       |      |
LEFT-WALL Buffy kicks.v the vampire.n .

(S (NP Buffy)
   (VP kicks
       (NP the vampire))
   .)
```

Figure 3.3: Link grammar parse-graph for a sample sentence. The constituent tree is given below. Note the "Ss" link that indicates subject-verb relationship.

We first parse the sentences using the link grammar [33]. This allows to obtain a parse-graph of the sentence. The graph determines semantic links between words. Among various relationships, a group of subject-verb links can be found. From those links the graph can be explored further to identify valid action verbs (often accompanied by adverbs), actors performing the action and objects to which the action may be applied, see fig. 3.3.

We use WordNet [24] to retrieve the semantics of the discovered verb expressions. We employ its morphological processor to reduce the verbs to their canonical form and retrieve the corresponding synonym sets. Then we explore the available range of actions and retrieve video samples with a semantic query.

3.3 Experiments: the Buffy human actions dataset

From the set of over 15000 video samples with descriptions, we have mined a set of commonly occurring verb expressions and retrieved the video sequences corresponding to them. The process was fully automatic except the choice of the interesting actions, which were picked from the discovered list of the most common verb expressions, for presentation of the experiments.

In short, the TV-series Buffy represents 7 seasons, i.e. 39 DVDs, i.e. 144 episodes, i.e. a little bit more than 100 hours of video which equals to approximatively ten millions of frames. Therefore, our experiments on such a large set of realistic videos shows the viability of our approach for large realistic databases.

In order to have an idea of the accuracy of the retrieval process, we manually annotated the 100 shortest video clips retrieved for seven different human actions frequently occurring in Buffy: walking, falling, punching, kicking, throwing, getting up and kissing. The ground truth statistics can be found in table 3.1.

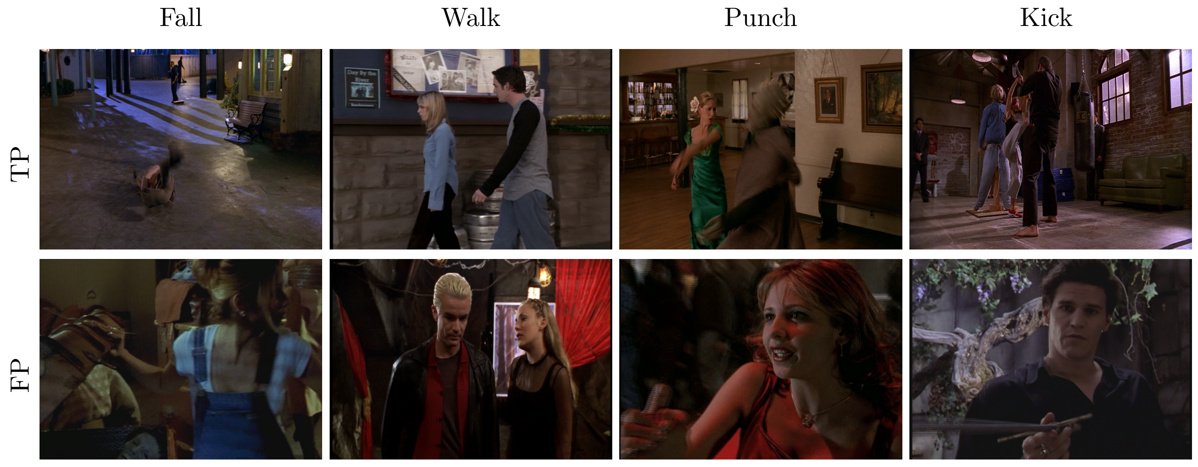

Figure 3.4: Example results for our retrieval procedure on four actions. We show the key frames of some true/false positives/negatives.

	walk	fall	punch	kick	throw	kiss	get up
true positives	80	59	72	73	61	71	74
false positives	12	34	23	21	31	8	19
error rate	13.0%	36.5%	24.2%	22.3%	33.6%	10.1%	20.4%
unclear (unused)	8	7	5	6	8	8	7

Table 3.1: Ground truth statistics for the human action samples retrieved from Buffy.

3.4 Discussion

Some samples were unclear as to whether they were relevant or not, therefore we didn't consider them for the future evaluation steps. We used them for the training, because the ground truth is assumed unknown during the training stage. The mean error rate is 22.87%, the maximum one being for falling (36.5%); the minimum for kiss (10.1%). It shows that our retrieval, though automatic, is performing better than [36] where they observe around 40% of errors.

Nevertheless, the results seem to strongly depend on the action class considered. The standard deviation of the error rates for our seven actions is of 9.74%. Furthermore, the error rate can come close enough to levels where they hurt the performance of classifiers that would use the retrieved samples as training data [36].

Therefore, there is a need for either more robust methods or explicit error management. In the next chapter, we propose to handle errors by ranking samples using visual information.

4

Visual ranking of a partially consistent set of videos

In this chapter, we try to tackle the problem of ranking a set of samples, sharing common visual properties, by consistency. An application of this problem is ranking by relevance a set of videos, obtained by querying for a topic with some text-based retrieval system. We propose an automatic ranking technique that can deal with partially consistent data. In section 4.1 we provide an introduction to the problem we address and to our contribution. In section 4.2 we transfer existing outlier removal approaches to the ranking problem and compare their performance with a standard classifier using automatically generated imperfect supervision. Finally, in section 4.3 we present our main contribution: using imperfect supervision and regression to improve the ranking process.

4.1 Introduction

We shortly describe the general machine learning problem we deal with, then the main motivation for our work and finally we present the proposed solutions.

4.1.1 Ranking by consistency to handle errors

The problem of supervised learning with erroneous training data as been rarely addressed, or only in very specific cases [37, 4]. Most of the time, the machine learning researchers assume that the training data is at most noisy, generally considering an additive noise model: $\mathbf{x} = \mathbf{x}^* + \epsilon$, where ϵ is a random variable (if $\epsilon \sim N(0, \sigma)$, it is called Gaussian white noise). But here we have misclassified training data, i.e. erroneous labels or noise on the target values y_i, whereas the usual meaning of noise is a perturbation on the features $\mathbf{x_i}$, i.e. the input values. Noise-robust supervised algorithms, for instance Support Vector Machines [56] (SVM) used in conjunction with slack variables, yielding high generalization capabilities [12], behave poorly in the presence of too much errors in the annotations of the training data [36]. Such errors generally occur when the labelling of the training data is done automatically on the basis of some machine output.

We propose to rank a set of samples using imperfectly labeled training data. The goal is to have errors pulled towards the lowest ranks. That enables identification of errors and a learner can then take some benefit from this supplementary information. For instance, erroneous training data can be used to improve supervised recognition tasks: false positives are examples of what is not relevant to the current class or topic of interest. We therefore try to keep and take advantage of all the data, knowing that there are errors.

The scope of the following sections is the general problem of ranking a set of documents relatively to their consistency. The goal is to have the most consistent samples at the top, while the samples that are less likely to come from the main underlying distribution are pushed towards the bottom of the ranking. For instance, in the context of information retrieval, the consistency of a set of documents retrieved by querying for a certain topic can be understood as their relevance, as the main part of the retrieved set is supposed to have features representing the topic (that is if the retrieval system's output is sensible off course).

4.1.2 Motivation

The cost of supervision

Classifiers require ground truth annotations, which is often human annotated training data, in order to learn what characterizes the classes and be able to generalize well to unseen samples. This strongly limits the applicability of such classification systems, that cannot reasonably deal with too many classes and too numerous training samples. For each class, such learners require labeling some actual class members and some that don't belong to this class. It is also important that the training data reliably reflects the distribution of the classes. This brings up the problem of how to construct a set of non-class samples, or error representatives.

This heavy manual pre-processing step that supervised techniques require is one of the main motivations for the development of unsupervised methods, requiring no annotations.

Errors in text-based retrieval

Due to the fact that the information provided by textual supervision is not always linked with the actual content of the concerned multimedia document, there are usually many false positives in the results of text-driven retrieval systems (cf. figure 3.4 for examples of errors). It is a global problem crippling the actual image and video search engines (like Google image search for instance), which are all using textual meta-data (tags, surrounding text, captions). For instance, querying for "punching" in Google image search, in the first top 100 samples, we obtain only around 20 actual punch examples (see figure 1.1 for an example of a correctly retrieved sample and of an irrelevant one).

The main problem of erroneous labels in the training data is that they mislead the learner by feeding him contradictory information. If the amount of label noise is too high, then the performance of a learner who learned using this wrong supervision crashes (cf. [36] for some details about SVMs robustness to noise in a bag-of-features framework).

Detecting and removing errors poses the problem of the global characterization of what is an error. In a general framework, errors are hard to formalize objectively. They are defined relatively to the concept of interest or to the rest of the data: an error is something that is not related to the topic, or that is markedly different from the other data samples. Therefore, it seems hard to fit a general model, or find some absolute criterion to explicitly define what is an error, without making any strong assumption on the data (like in [37, 4]).

The intrinsic relativity of errors still allows them to be defined on a per class basis by the simple formulation: an error doesn't belong to the class. Thus, we can imagine building a binary classifier for each class and use it as an error detector. The problem with such an approach is the amount of human effort required to get sufficient annotated training data, i.e. good examples of class members and non-class members for each class.

This can be bypassed by using (unsupervised) clustering algorithms on some visual features of the retrieved documents, at a very low retrieval rate but with high precision. While this preserves the automaticity of the system, it nevertheless means only retaining a tiny proportion of the available data. It is often the most representative subset, which is not necessarily the most discriminative one (which is what matters for recognition), or does not necessarily correspond to the needs of the users.

4.1.3 Proposed automatic ranking algorithms

As we deal with the previously mentioned problem of ranking a set of samples by their consistency in order to handle errors, we now shortly describe the two main solutions we consider.

We propose here to rank the samples, automatically, without any need for human supervision, by their visual consistency. Errors, or irrelevant documents, can be seen as samples whose visual content differs from a substantial subset of the data, the set of relevant documents correctly retrieved. This requires the assumption that there is such a large enough subset of relevant data retrieved, but, knowing the performance of most up-to-date retrieval systems, it is reasonable to assume that it is at least half of the retrieved set.

Furthermore, those rankings can be used for what they are (quantification of representativeness) to improve a learning process, or they can be used to remove errors (in the lower ranks), according to what they define as their tolerance level (the cut-off rank).

We provide a kernel method to learn efficiently with all of the training data, including errors. We propose to iteratively re-train Support Vector Regressors [55] (SVRs) to differentiate good and bad supervision by ranking the samples by consistency: samples consistent with the remainder of the data are pushed towards the top ranks, whereas the outliers are pulled down. The output of our algorithm is a training list ranked by visual consistency and a kernel based discriminative classifier trained on this ranked list. Furthermore, as we require only weak supervision, we can render our procedure fully automatic, like unsupervised clustering algorithms. We show that our approach produces better results than state-of-the-art unsupervised techniques, thanks to the use of supervision.

We first introduce these unsupervised techniques which we derived from state-of-the-art unsupervised outlier removal algorithms. Then, we present our main contribution: an iterative training scheme to learn to rank using regression and imperfect supervision.

Outlier removal for ranking

An interesting set of techniques that can be used to solve the problem of ranking samples by visual consistency is based on the unsupervised outlier removal paradigm, sometimes considered for mislabeled data detection. Unsupervised machine learning methods are particularly adapted to this setup, where we only have a set of documents from which we want to determine, with no prior on the data, common characteristics. Furthermore, outliers are often defined as "observations that deviate markedly from others" or "observations inconsistent with the remainder of the data" [5].

Therefore, we can formalize non-relevant documents as outliers. As a result, any unsupervised outlier detection method that computes a measure of "outlierness", or inconsistency, can be directly used as a scoring function inducing a reliable consistency ranking. We applied and compared two existing methods that we describe in details in section 4.2.

Furthermore, we use a standard ν-SVM with misclassified training data and we show that it performs better than the chosen unsupervised methods, i.e. that imperfect supervision helps a learner, even if he doesn't differentiate good and bad annotations.

Bipartite ranking using regression

The generic and well studied framework of bipartite ranking [26, 2, 1] shares some similarities with the binary classification task. Basically, the goal is, from a given training set containing binary relevance labels, to learn a scoring function that ranks future relevant documents higher than non-relevant ones.

One problem of such an approach is that it often requires strong supervision (human annotated data) in order to create a training set. Most supervised learning algorithms assume that the training data reflects the distribution of samples, in order to generalize to new data.

Many problems ofter a way to automatically generate labeled data, and therefore get rid of any human intervention. But they might create a high amount of actually misleading wrong supervision, e.g. positively labeled samples that have actually nothing to do with the class of interest.

In section 4.3 we propose a novel way to solve the bipartite ranking problem in the case of misclassified training data, i.e. imperfect supervision, using regression tools.

Regression techniques have been successfully applied to the ranking problem (e.g. [13, 20]). They often either require training data with preference constraints, ratings (e.g. movie ratings) or training samples for each rank (ordinal regression). We differ from existing approaches by using regression for ranking using only binary relevance labels. Furthermore, we consider the case of partially incorrect training labels.

Our main contribution is to show how to use Support Vector Regression [55, 22, 54] with imperfect binary supervision (prone to errors) as a way of obtaining a scoring function, inducing a ranking reflecting the consistency of a given dataset of documents that are likely to share some common content. Imperfect supervision can generally be easily obtained. The subset of documents we want to rank can be considered as positives, and a cheap and automatic way to obtain correct negatives is often easy to find, because we are interested in bipartite ranking between consistent (assumed relevant) samples and inconsistent ones (outliers, errors, assumed irrelevant). For instance, by sampling the whole collection from which the subset we want to rank is extracted, or even by sampling directly the feature space.

We propose, provided we have such a weak supervision, a novel iterative re-training scheme for SVRs to properly handle the arbitrary amount of errors during the training stage of a SVR. The goal is to produce a sensible ranking of our documents such that the most consistent ones are above the inconsistent ones.

4.2 Estimation of outlierness

We here describe two state-of-the-art unsupervised methods classically used for outlier removal. We explain how we use them to compute a measure of inconsistency, or "outlierness", of samples in order to rank them using this score. We also use a SVM with imperfect supervision and no specific error management policy to compare the performance of a traditional supervised technique and of state-of-the-art unsupervised methods, in the case of many errors in the data.

4.2.1 Estimation of a distribution's support

A first intuitive idea to remove outliers would be to estimate the underlying probability density of the data in order to check if it is likely that a sample is drawn from this distribution. If not, it can be considered as "atypical" or inconsistent: i.e. an outlier.

One-class SVMs

This problem can be addressed using a one-class SVM [51]. The goal is to define a border around the main consistent part of the data, thus estimating a contour line of the underlying data density. Such lines can be used to separate "typical" objects from "atypical" ones, i.e. outlier candidates. Furthermore, this approach has the advantage of directly computing a boundary, as opposed to first estimating the whole density, which follows one of the main declination of Ockham's razor in machine learning stated by Vapnik: avoid solving too hard intermediate problems.

Ranking using confidence values

The goal here is to rank samples by consistency (or inconsistency), not to remove outliers. Therefore, we need to find a scoring metric that induces such a ranking. As the margin represents a boundary of normality, the distance from margin $\frac{|f(x)|}{||\mathbf{w}||}$, where $\mathbf{w}$ is the margin size and f is the confidence value function defined in eq. 2.13, can be considered as a measure of how much a sample deviates from normality, i.e. quantify its inconsistency. In the experimental section we provide conclusive facts that show that this approach is usable in practice.

4.2.2 Proximity-based approach

A family of more classical unsupervised outlier removal methods are based on a proximity criterion [29]. The main idea is that a sample found in a region of the feature space that is locally sparse might be an outlier. The sparsity is generally evaluated locally by computing the distance to the nearest neighbours or to the k^{th} neighbour [49], using a certain metric. If this measure of spatial density is below a certain threshold, the sample is considered as an outlier.

Densest component

An very close approach is representing the similarity (or distance in a suited feature space) in a graph structure. The set of most similar documents can then be assimilated to the densest component of the graph. A recent example of a successful practical application of this perspective to a realistic machine vision problem can be found in [47]: naming faces in news photographs. Starting from a set of face images, represented by SIFT descriptors and retrieved by querying for a certain person on captions of news photos, the authors represent the visual similarities of the images using a graph structure. They then compute the densest component of the obtained graph in order to find the set of most similar faces, reasonably assumed to be the queried person, and discard non-relevant faces retrieved due to wrong caption-based supervision.

The idea is that every node that doesn't belong to the densest subgraph is in a sparse region and can therefore be considered as an outlier.

Ranking by pruning order

Finding the densest component is not the final goal we aim for. In order to rank the samples, we need either a scoring metric or directly an order. We achieve the latter case by simply considering the pruning order of the greedy 2-approximation algorithm we use to find the densest component (iteratively deleting the node with minimal degree, cf. section 2.2.3). The motivation for this is that the greedy algorithm removes the samples with minimum density first, i.e. samples that have a sparse neighbourhood. We therefore can rank the samples by pruning order: in the lowest ranks we have the first removed samples, lying in the sparsest regions of the graph representation of the gram matrix, and in the highest we have the core nodes of the densest subgraph.

4.2.3 Binary ν-SVM with imperfect supervision

We also used a standard SVM directly using imperfect supervision without any special error treatment, for comparison purposes. We used a ν-SVM, defined in section 2.2.2. Like for the one-class SVM, we used the confidence values defined in eq. 2.10 as a scoring function to induce the ranking.

4.2.4 Experiments

We describe experiments on seven action classes automatically retrieved from the Buffy video dataset (cf. section 3.3 for more details).

The Buffy data

We explore actions in 144 episodes of the TV series “Buffy the Vampire Slayer”. Each episode is 40-45 minutes long. Transcripts for all the episodes are available at the Buffy World website and contain about 800 lines each. We have synchronized the transcripts to the videos and retrieved 100 video samples for seven discovered common human actions: *walking*, *falling*, *punching*, *kicking*, *throwing*, *kissing* and *getting up*. We then manually annotated the samples (cf. table 3.1 for the ground truth statistics) and applied our ranking algorithms on all the classes.

Imperfect supervision

For each action class we used the 100 retrieved samples as positives, thus yielding a rate of false positives equal to the retrieval error rate (see table 3.1). For the negative samples we used one set of 100 video clips automatically retrieved from random non-dialog sentences in the transcripts. We used the same negatives for every action class.

It is important to notice that, though we use supervised techniques, we still have a fully automatic procedure that doesn't require any human annotated training data.

Parameters

We have explored the possible ν settings and found $\nu \in [0.6, 0.9]$ to be a reasonable generalization factor for our Buffy dataset. Therefore, in the following we set $\nu = 0.8$ for the ν-SVM and the one class SVM. Regarding the choice of the kernel K, we used the χ^2 kernel defined in 2.6.

Results

Figure 4.1 presents the results[1] of the three different ranking algorithms on six (because of space limitations) action classes and table 4.1 gives the corresponding average precisions for all the seven classes.

For most action classes, the ν-SVM performs clearly better than the best unsupervised method. (up to +2.8% for “walking”), except for “punching” and “throwing”.

For the unsupervised approaches, the one-class SVM performs better than the densest component approach, on all classes except “throw”. For easy classes, with a low error rate (e.g. walk, kiss, kick), the two approaches perform comparably.

[1] The recall-precision points were obtained using different thresholds on the confidence values computed by the SVM, and different translation parameters t for the edges' weights in the densest component approach.

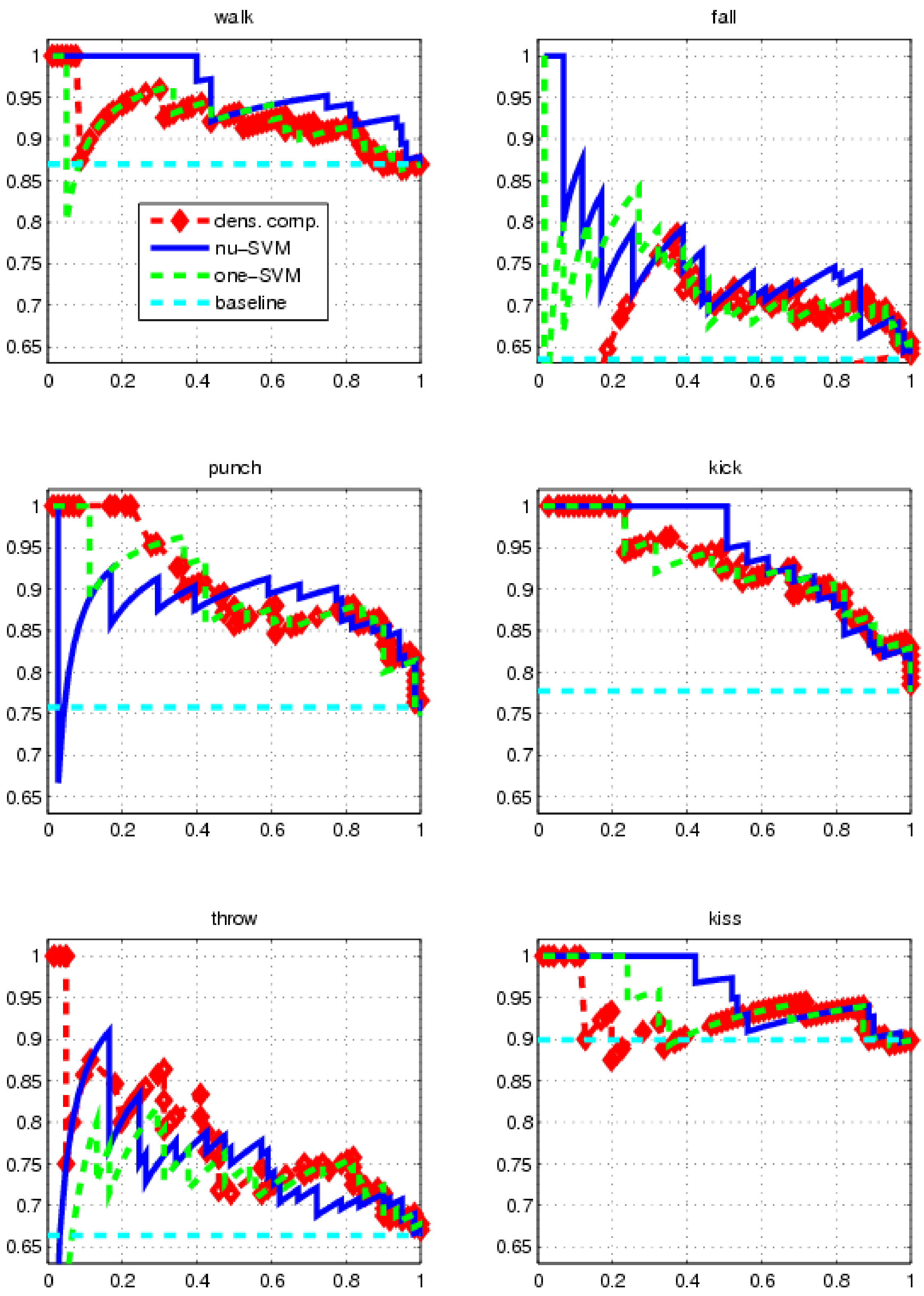

Figure 4.1: (Recall,Precision) plots for ranking six Buffy action classes with the nu-SVM, one class SVM and densest component approach.

	walk	fall	punch	kick	throw	kiss	get up
ν-SVM	**96.4%**	**77.6%**	89.7%	**94.5%**	77.9%	**96.6%**	**88.2%**
One class SVM	93.6%	76.5%	**91.0%**	93.2%	76.6%	95.1%	87.6%
Densest component	93.2%	73.6%	90.7%	93.0%	**80.5%**	94.6%	81.5%
Random	87.0%	63.5%	75.8%	81.7%	76.4%	89.9%	79.6%

Table 4.1: Average Precisions for the binary nu-SVM, the one-class SVM and densest component approach to ranking, on seven human action samples retrieved from Buffy.

Those experiments first show that our outlier removal approach to consistency ranking is efficient in a realistic scenario, because, on all classes, there is a clear improvement of the performance comparing to random ranking

The value of supervision The main interesting result is that, even with a wrong supervision and no appropriate treatment of the errors, the supervised classification approach of ν-SVM works better than both unsupervised outlier removal methods. This can be explained by two factors. First, the robustness of SVM to noise. This is not the full explanation because ν-SVM performs better than the one-class SVM. The main difference between the two methods is the use of supervision. The experiments allow us to claim that on realistic data, a wrong supervision is still better than no supervision at all. That is what we call the value of supervision.

4.3 Bipartite ranking with misclassified training data

In this section, we address the problem of ranking samples when partially incorrect binary relevance labels are available. The problem of ranking samples, with binary relevance labels is called the *bipartite ranking problem* [26, 2, 1]. We propose to use imperfect supervision relative to the visual consistency of a set of partially consistent videos, in order to rank them. We use support vector regression [55, 22] to rank samples and we provide a novel iterative re-training scheme to handle the misclassified data.

4.3.1 Related work

Link with the unsupervised paradigm

The previous experiments have illustrated the intuition that unsupervised methods are intrinsically less efficient than supervised methods, simply because they use less information. For instance, the densest component approach in [47], in a sense, forgets the supervision information provided by the inaccurate caption-based retrieval, and transforms the problem as an unsupervised clustering one. The fact that the ν-SVM outperforms the other outlier removal methods shows that even a bad supervision is better than no supervision at all. This justifies the attempt to propose a support vector approach for taking benefit of this wrong, or imperfect, supervision.

Semi-supervised learning

The machine learning community is nowadays more and more concerned about the problem of learning with partially annotated data. Traditional classifiers use only labeled data (feature - label pairs) to train. Labeled instances however are often difficult, expensive, or time consuming to obtain, as they often require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect. Semi-supervised learning (cf. [63] for a survey) addresses this problem by using large amount of unlabeled data, together with a small set of labeled data, to build better classifiers. Because semi-supervised learning requires less human effort and gives higher accuracy, it has raised the curiosity of many researchers and proved a good approach in theory [9, 6, 15], as well as in practice (cf. [31] for an application of semi-supervised, or "transductive", SVMs to text classification).

These semi-supervised methods, though better than unsupervised ones, are still below the performance of supervised classifiers using properly annotated data. Nevertheless, they require only a low level of human intervention, to annotate the training set at least, and can benefit from high amounts of unlabeled data.

We go even further in this direction and show that we can use a supervised approach, in a fully automatic manner, that yields better performance than unsupervised techniques. Instead of considering unlabelled data, we consider easily available imperfect labels, which contain more information. Even the erroneous information, if handled properly, can become a source of improvement. In a sense, a bad teacher is better than no teacher at all.

Maximum margin clustering

Our approach is closely related to the maximum margin clustering paradigm [60, 62]. The authors in [62] proved the efficiency and strong improvement potential of using an iterative re-training of SVRs starting from binary target values.

The first main difference with our work is that we are in a supervised setup, whereas they start from unlabeled data and initialize the binary target values using a two class clustering algorithm. We assume that we have weakly labeled data containing many mislabeled positives. We designed our algorithm in a way to exploit this prior. We also relax the constraints on the target values, in order to make "smoother" decisions and interpret the decision values as measures of consistency. The concept of what is consistent is shown to be improving at each iteration, resulting in a growing quality of the induced rankings.

4.3.2 Iterative re-training of SVRs to learn with errors

In the following, we explain our main contribution: the iterative re-training scheme for SVRs. We first recall the definition of SVR. We then give some insights on the principle our method and finally explain our algorithm in details.

Iteratively detecting errors in order to improve

When faced with weak or erroneous supervision during training, supervised machine learning techniques face mainly two threats: either over-fitting by blindly trusting the supervision (it learns everything including errors), or insufficiently learning the specificities of the concerned classes by not trusting the label information enough (thus resulting in predictions containing a lot of false positives).

Our main idea is quite intuitive and can be understood as follows. As we are in a setup of imperfect supervision, in the first iteration, the SVR, not differentiating good and wrong supervision, will be misguided by the erroneous target values, thus resulting in regression values inducing a poor ranking. Nevertheless, the generalization capabilities of SVRs allow some of the most inconsistent target values to clearly shift from the original ones, thus reducing the misleading factors of this supervision.

We show in figure 4.2 an illustration of our idea on a one dimensional toy example.

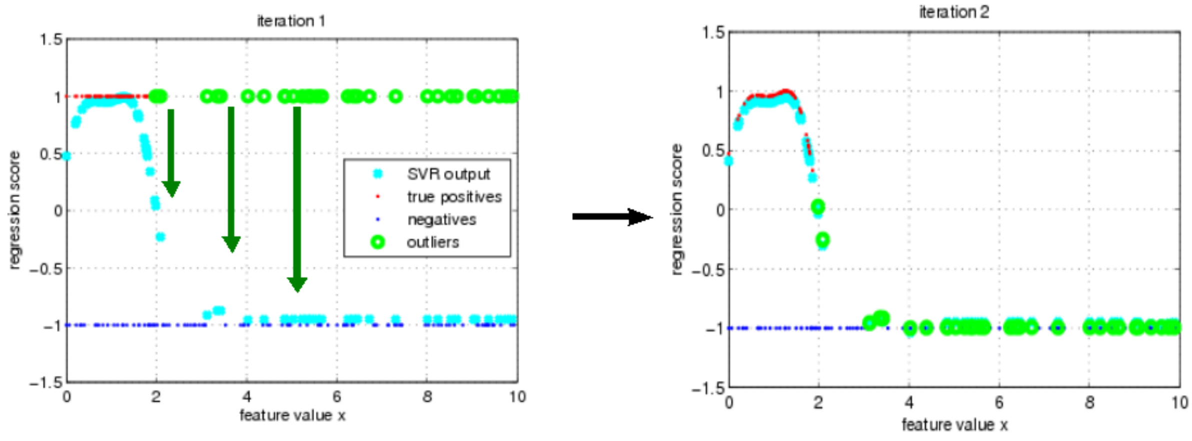

Figure 4.2: Toy example using an SVR with a linear kernel on 1-D data. We show the initial labels of the TP and the outliers (FP), of the negatives and the predicted values after one step ("SVR output"), for the first and second iterations.

The 70 true positives are drawn from a Gaussian distribution of small variance in order to represent the consistency of the relevant samples. The 30 outliers (false positives) are uniformly randomly sampled from the rest of the world (outside $[min(TP), max(TP)]$). The negatives are also uniformly randomly sampled but from the totality of the world $[0, 10]$. We used an ϵ-SVR with a Gaussian (RBF) kernel and $\epsilon = 0.01$, $C = 10$. The first iteration of our algorithm on such artificial data show that only the "class members", i.e. the red samples on the left in figure 4.2, are keeping high decision values, and therefore are ranked higher, whereas outliers, the green samples on the right, have decision values far from their original target value $(+1)$. As we replace the previous labels with the estimated regression values, the inconsistent samples are strongly moved towards the negative label (-1), as we can see in the second iteration. In this toy example, the problem is so easy (it is linearly separable) that the second iteration nearly changes nothing regarding the rankings, that is why the regression function is so close to the new "labels".

Iterative SVR for ranking

Let $P = x_1, \cdots, x_n$ be a set of documents, e.g. a set of videos retrieved by some text-based approach. We assume that the documents are likely to share some common characteristics, e.g. relevance to a certain topic. The goal is to rank this set by consistency in order to differentiate samples that are likely to be inconsistent with others, i.e. not sharing the same properties. We also assume that another set of documents $N = x_1^*, \cdots, x_n^*$ is available and provides some weak estimation of what is not consistent with the rest of the data (we will show that this is a cheap assumption in practice).

We consider all x_i as positives and assign them the target value $+1$. In the same way, we consider N as the negative set with target labels being -1. We then have imperfect supervision in the sense that there are many mislabeled positives and some false negatives (but only a very low proportion).

We then train a SVR on this labeled data, and test it on the positive set P. We obtain regressed values y_i $i = 1..n$, which, after renormalization of all values between 0 and 1, we consider as new target values for the x_i $i = 1..n$. We re-train the SVR using the same negative targets as before, but with the new positive ones.

We iterate this process until convergence, i.e. until the target values of the x_i $i = 1..n$ don't change anymore. Finally we use the last target values as scores and rank P according to them.

4.3.3 Experiments

We now provide some experiments on the seven action classes retrieved from the Buffy dataset 3.3). We show that our iterative re-training scheme improves the rankings obtained by a SVR, and that it converges very fast.

Parameter selection

The values of the penalization parameter C and of the thickness of the tube ϵ are directly influencing the generalization properties of SVRs. Therefore, it's important to have a way of selecting sensible values. Thanks to the popularity of SVMs, this problem of parameter selection has been extensively studied for SVRs, and in practice, we can use the proposed C formula of Cherkassky [18] which yields good results, thanks to a certain robustness to outliers, and doesn't require by-hand tuning. The formula is the following:

$$C = max(|m_y + 3\sigma_y|, |m_y - 3\sigma_y|) \qquad (4.1)$$

where m_y and σ_y are the mean and standard deviation of the target values y of the training data.

The selection of the ϵ parameter is more complicated because it requires an estimation of the standard deviation of the noise ratio [18], which is changing at every step of our re-training algorithm. For now, we didn't find an automatic procedure to compute a suited ϵ at each iteration. We therefore chose for each class a constant ϵ parameter yielding the global best performance that can be achieved using a constant ϵ, in order to give some insights on the potential of our method.

Improving the rankings

We now show the performance of our iterative algorithm on Buffy videos. We used the same imperfect supervision as used by the experiments in section 4.2.4. In figure 4.4 we provide the plots of the evolution of average precision at each iteration (i.e. re-ranking step), for six action classes. The starting point is the result of ranking the samples using the direct output of a SVR, without any re-labeling (target value re-assignment). The best result of three previously tested methods (usually obtained with the ν-SVM) is also present on the plots.

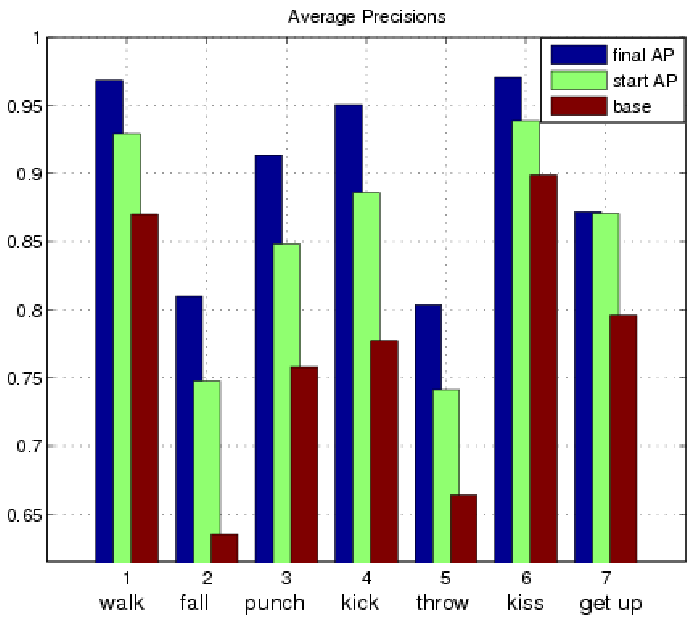

Figure 4.3: Histogram of the performance of our approach ("final"), the SVR without re-training ("start") and a random classifier ("base").

For all classes, without exception, convergence is obtained very quickly and we observe a clear improvement of the performance compared to the starting point, cf. figure 4.3. The table 4.2

	walk	fall	punch	kick	throw	kiss	get up
Iterative SVR	**96.8%**	**81.0%**	**91.4%**	**95.1%**	80.4%	**97.1%**	87.2%
ν-SVM	96.4%	77.6%	89.7%	94.5%	77.9%	96.6%	**88.2%**
One class SVM	93.6%	76.5%	91.0%	93.2%	76.6%	95.1%	87.6%
Densest component	93.2%	73.6%	90.7%	93.0%	**80.5%**	94.6%	81.5%
Random	87.0%	63.5%	75.8%	81.7%	76.4%	89.9%	79.6%

Table 4.2: Average Precisions for the iterative SVR, binary nu-SVM, one-class SVM and densest component approach to ranking, on seven human action samples retrieved from Buffy.

contains all the average precisions of the different tested methods. It shows that for all classes, except "throw" and "get up", the iterative SVR approach performs better, but only slightly. The main fact noticeable from figure 4.4 is that the SVR usually starts a lot lower than the previous best result (often with the ν-SVM), but still manages to go above.

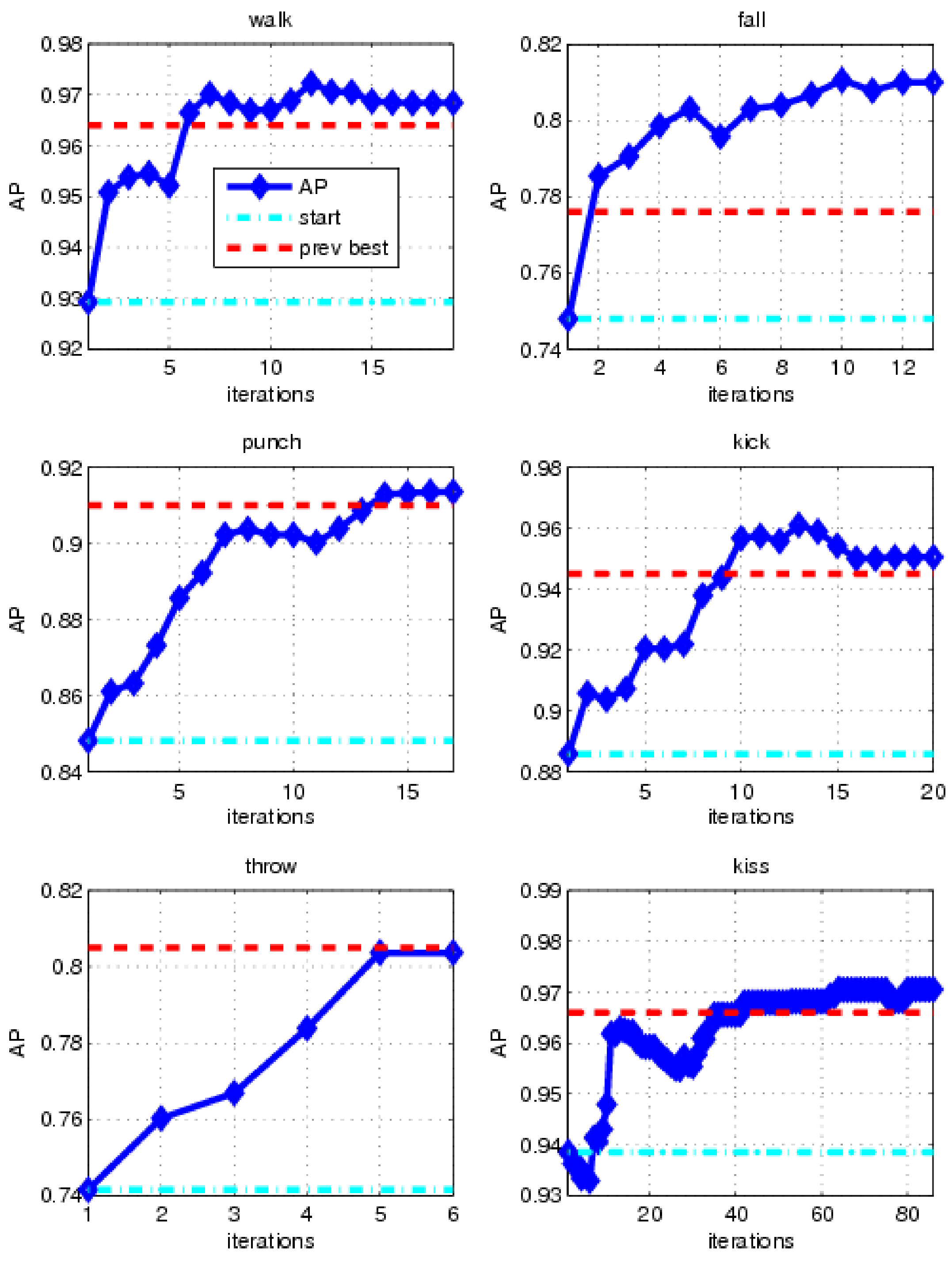

Figure 4.4: Results for re-ranking six Buffy action classes with our approach, “start” is the result of the ranking produced by the SVR without re-training, i.e. our first iteration, “prev best” is the previous best result, cf. table 4.1

5

Conclusion

During this internship, we addressed the lack of realistic and large-scale human actions dataset. We provided a fully automatic textual retrieval procedure that uses subtitles and transcripts to extract video samples of frequent visual actions from movies and TV-series. We then transfered existing unsupervised outlier removal algorithms to the problem of ranking a set of retrieved videos by visual consistency. This was done with the aim of using visual information in order to improve text-driven image or video retrieval.

Furthermore, we improved this visual re-ranking process by considering easily available imperfect supervision: all retrieved samples can be labeled as positives (class members) and random samples extracted from the whole collection can be seen as negatives (like errors). Our main contribution is the development of a new re-training scheme for support vector regression, that allows the learning algorithm to differentiate good and bad supervision, and therefore goes from being misled by wrongly annotated training data to using it at his own advantage. Furthermore, this algorithm is completely automatic and is shown to outperform other unsupervised methods.

Combining our text retrieval and the proposed visual re-ranking process, we obtain a fully automatic multi-modal retrieval procedure that is capable of extracting lots of actions samples for many different action classes, from large-scale video collections, without any human intervention. Note that our approach is not just restricted to videos and can also be applied to image web search for instance. We also provide some results on the general problem of learning with misclassified training data.

Finally, we successfully applied our approach on a large collection of realistic videos, showing encouraging results and practical applicability.

Future work

In the future we intend to investigate the difference in the performance obtained using ν-SVM versus ϵ-SVR. Bridging the observed gap could result in even better performance of our iterative approach, currently penalized by the lower performance in the first iteration. One of the possible directions is modeling the ϵ parameter and varying it from iteration to iteration. We also plan to extend the Buffy dataset to more actions and samples, and to use other TV-series and movies in order to see how our learning algorithms generalize from one to another.

On the longer term, regarding the machine learning theory, we will try to explore into more details the link and eventual extension of our ideas to semi-supervised learning techniques, like “transductive” SVMs [31]. An interesting question to answer would be to know if it is better to forget wrong labels (use unlabeled data like in semi-supervised learning) or to explicitly use them (like we tried during this project). Another set of techniques we want to compare to is ordinal regression algorithms, which are very efficient for ranking [39]. Finally, it would also be worth investigating cost-sensitive extensions to SVMs that can be very useful to handle different losses for misclassification instances and has already been applied to ranking [59].

We will also investigate a bit deeper the notion of visual consistency and its relation with the “visualness” of an action. Some actions are a priori not “visual”, in a sense that recognizing them using only visual information is not likely (e.g. waiting, seeing, thinking, feeling something). Using our approach, we could automatically mine different actions and compute the visual consistency of classes in order to automatically discover visually recognizable actions. We could answer questions like: can a machine recognize someone interested by some advertisement, someone attracted by someone else, a person having a suspicious behavior, can a robot understand the feelings of a human by just looking at him, etc..

Finally, we want to refine our action recognition framework in order to combine action, actor and object, perform accurate localization and use all the elements available in the scene (e.g. background elements) to improve recognition of human actions in realistic videos.

Bibliography

[1] S. Agarwal, T. Graepel, R. Herbrich, and D. Roth. A large deviation bound for the area under the ROC curve. *Advances in Neural Information Processing Systems*, 17:9–16, 2005.

[2] S. Agarwal and D. Roth. Learnability of bipartite ranking functions. *Proceedings of the 18th Annual Conference on Learning Theory*, 2005.

[3] S. Ali, A. Basharat, and M. Shah. Chaotic Invariants for Human Action Recognition. *ICCV*, pages 1–8, 2007.

[4] M.R. Amini and P. Gallinari. Semi-supervised learning with an imperfect supervisor. *Knowledge and Information Systems*, 8(4):385–413, 2005.

[5] V. Barnett and T. Lewis. Outliers in statistical data. In *3rd edn, John Wiley and Sons*, 1994.

[6] K. Bennett and A. Demiriz. Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, 11:368–374, 1998.

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

[8] T.L. Berg and D.A. Forsyth. Animals on the web. In *CVPR*, 2006.

[9] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[10] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, 2007.

[11] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[12] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[13] B. Carterette and D. Petkova. Learning a ranking from pairwise preferences. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 629–630, 2006.

[14] O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, pages 1055–1064, 1999.

[15] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, 15:1, 2003.

[16] M. Charikar. Greedy approximation algorithms for finding dense components in a graph.

[17] K. Chellapilla and P. Simard. Using machine learning to break visual human interaction proofs (hips). In *NIPS*, 2004.

[18] V. Cherkassky and Y. Ma. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, 17(1):113–126, 2004.

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:1–25, 1995.

[20] D. Cossock and T. Zhang. Subset ranking using regression. *Proceedings of the Conference on Learning Theory (COLT 2006)*, pages 605–619, 2006.

[21] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, pages 65–72, 2005.

[22] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. 1997.

[23] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in TV video. In *BMVC*, 2006.

[24] C. Fellbaum, editor. *Wordnet: An Electronic Lexical Database*. Bradford Books, 1998.

[25] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, 2005.

[26] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, 4(6):933–969, 2003.

[27] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Automatic face naming with caption-based supervision. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2008.

[28] D. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 1975.

[29] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[30] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.

[31] T. Joachims. Transductive inference for text classification using support vector machines. *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, 1999.

[32] Y. Ke, R. Sukthankar, and M. Hebert. Event Detection in Crowded Videos. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

[33] J. Lafferty, D. Sleator, and D. Temperley. Grammatical trigrams: A probabilistic model of link grammar. In *AAAI Conference on Probabilistic Approaches to Natural Language*, 1992.

[34] I. Laptev. On space-time interest points. *IJCV*, 64(2/3):107–123, 2005.

[35] I. Laptev and P. Pérez. Retrieving actions in movies. In *ICCV*, 2007.

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

[37] N.D. Lawrence and B. Scholkopf. Estimating a kernel Fisher discriminant in the presence of label noise. *Proc. Int. Conf. Machine Learning*, 2001.

[38] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[39] L. Li and H.T. Lin. Ordinal Regression by Extended Binary Classification. *NIPS*, 2007.

[40] L.J. Li, G. Wang, and L. Fei Fei. Optimol: automatic online picture collection via incremental model learning. In *CVPR*, pages 1–8, 2007.

[41] M. Marszałek, C. Schmid, H. Harzallah, and J. van de Weijer. Learning object representations for visual object class recognition, 2007. The PASCAL VOC'07 Challenge Workshop, in conjunction with ICCV.

[42] Marcin Marszałek, Cordelia Schmid, Hedi Harzallah, and Joost van de Weijer. Learning object representations for visual object class recognition, oct 2007. Visual Recognition Challange workshop, in conjunction with ICCV.

[43] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

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

[45] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In *BMVC*, 2006.

[46] S. Nowozin, G. Bakir, and K. Tsuda. Discriminative Subsequence Mining for Action Classification. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

[47] D. Ozkan and P. Duygulu. A graph based approach for naming faces in news photos. In *CVPR*, 2006.

[48] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods: support vector learning table of contents*, pages 185–208, 1999.

[49] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[50] S. Rice, F. Jenkins, and T. Nartker. The fourth annual test of ocr accuracy, 1995.

[51] B. Schölkopf, R. Williamson, A. Smola, and J. Shawe-Taylor. Sv estimation of a distribution's support. *Advances in Neural Information Processing Systems*, 1999.

[52] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *ICCV*, 2007.

[53] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local SVM approach. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 3, 2004.

[54] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[55] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. 1996.

[56] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2000.

[57] D. Weinland, E. Boyer, R. Ronfard, and G. LJK-INRIA. Action Recognition from Arbitrary Views using 3D Exemplars. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7, 2007.

[58] J. Willamowski, D. Arregui, G. Csurka, C. R. Dance, and L. Fan. Categorizing nine visual classes using local appearance descriptors. In *IWLAVS*, 2004.

[59] J. Xu, Y. Cao, H. Li, and Y. Huang. Cost-sensitive learning of SVM for ranking. *ECML*, 4212:833, 2006.

[60] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. Maximum margin clustering. *Advances in Neural Information Processing Systems*, 17:1537–1544, 2005.

[61] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2007.

[62] K. Zhang, I.W. Tsang, and J.T. Kwok. Maximum margin clustering made practical. *Proceedings of the 24th international conference on Machine learning*, pages 1119–1126, 2007.

[63] X. Zhu. Semi-Supervised Learning Literature Survey. *Computer Science, University of Wisconsin-Madison.*