UNIVERSITÀ
DEGLI STUDI
FIRENZE

Scuola di Ingegneria
Corso di Laurea Magistrale in Ingegneria
Informatica

CCN for Automatic Data Collection and Filtering

Basi di Dati Multimediali

Saverio Meucci

Anno Accademico 2015/2016

Introduction

Objective: the creation of a procedure, based on the work of [1], thanks to which build a large dataset of images of faces that can be used in the training process of a **CNN**.

This procedure takes advantage of the modern **search engines**, and, at the same time, has the objective of decreasing at a minimum the cost in terms of human time necessary to the acquisition and annotations of the images of the dataset.

Introduction

The proposed method is divided in phases, as follows:

- A **dataset creation** phases, by exploiting modern search engines.
- Detect the faces inside the downloaded images thanks to a **face detector**.
- A **duplicate removal** phase, thanks to which identical duplicate images are removed.
- A **classification** phase, to determine if the faces are correctly associated to an identity.
- A **visual validation** phase, done by a user with the help of a web application.

Dataset Creation

The phase for the **dataset creation** consists in the acquisition of images exploiting the modern *image search engines*.

It is necessary to decide a list of **identities** that are sufficiently popular, so that for each identity there will be a high number of images, between 500 and 1000) as results of the search engines.

The acquisition procedure is divided in two parts:

- A collection phase.
- A download phase.

Collection

The **collection** phase takes advantage of three search engines, such as Bing, Yahoo and AOL, to query and obtain images for each identity.

Each HTML page returned contains the links to the images related to the query. Because every search engine has a different layout for the returned HTML pages and a different way of presenting the links in the pages, it is necessary to implement a **parser** for every search engine used.

The links are then saved in the database with information about the related identity, the search engine used and the rank of the result.

Download

The **download** phase takes place once the database of links have been created.

The procedure is straight forward: the images are downloaded and saved in a folder related to the identity, using the links obtained in the previous step.

Face Detector

The successive phases have the objective of **filtering** the obtained dataset, in order to remove, for each identity, the images that are not relevant.

The results of the queries to the image search engines for a specific identity, will also contain images that are not related to that identity, thus the need of a filtering phase.

Since the objective of this paper is the creation of datasets of images containing **faces**, we need to detect and select the faces in the retrieved images.

Face Detector

This task is accomplished implementing a **face detector**, written in C++ using the **Dlib** library [2].

The **Dlib** C++ library implements a face detector by using the classic **histogram of oriented gradients** (**HOG**) feature combined with a linear classifier, an image pyramid and sliding window detection scheme.

The face detector takes an image, or list of images, and returns the coordinates of the **bounding box** surrounding the detected faces, which are then saved in the database.

Face Detector

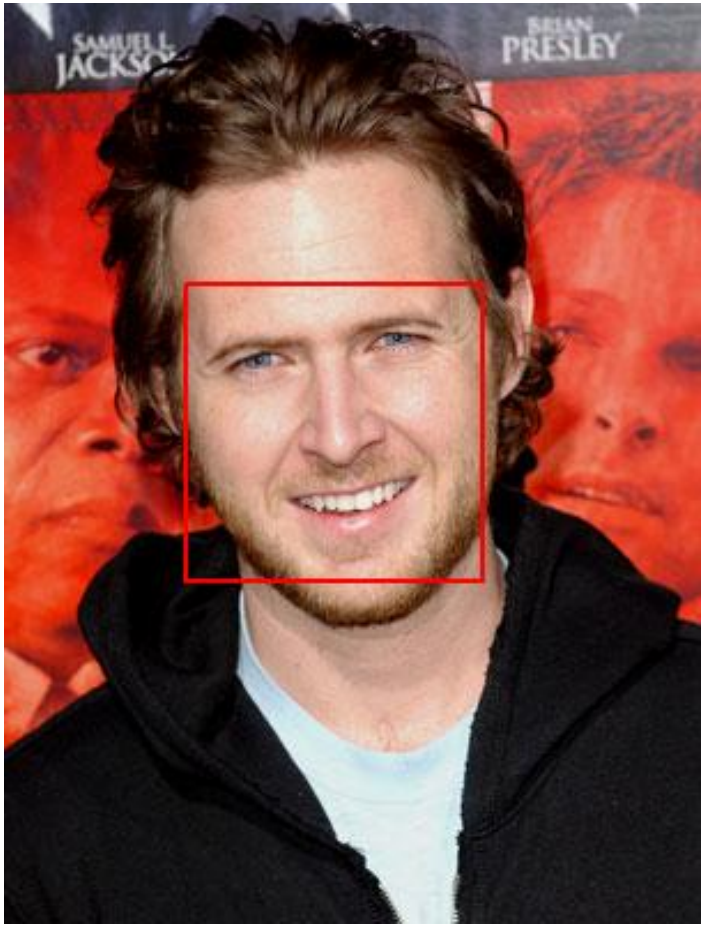


Figura : Example of a face found by the detector with its relative bounding box.

Duplicate Removal

In the results of an image search engine there will be images that are **duplicates**, both because the returned links refer to the same location and because the same image is stored in two different locations.

In the created datasets there will be presents a certain number of duplicated images for each identity.

That is not desirable, since the dataset need to be used to train a **CNN**. A training phase need to generalized as much as possible the objects that is learning; a duplicated image does not add any useful information to this process.

Duplicate Removal

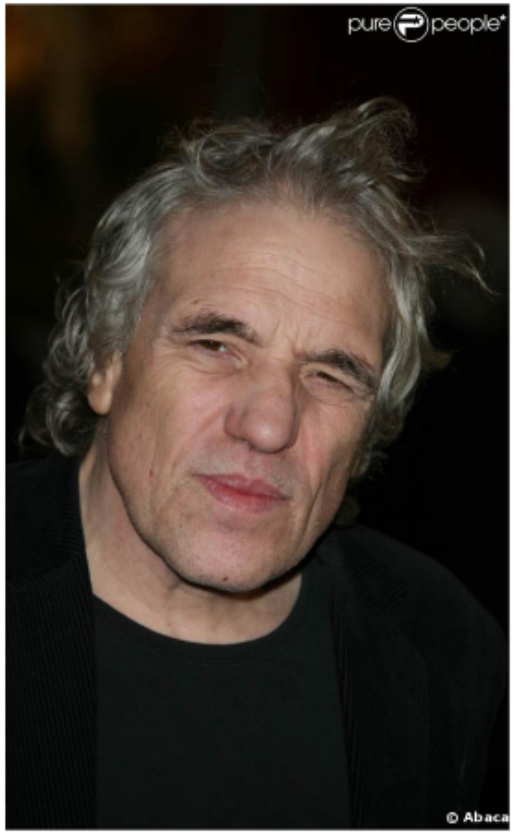
The images are represented as extended bounding boxes related to the detected faces, in order to consider not only the face but also the contest of the image.

For each image, a **feature vector** is computed using the **Vector of Locally Aggregated Descriptor** (**VLAD**) encoding, using the implementation from the **VLFeat** library [5].

The **VLAD** is a feature encoding and pooling method that encodes a set of local feature descriptors extracted from an image using a feature dictionary built using a clustering method.

These feature vectors are then **clustered** within the images for each identity; only a single element per cluster will be retained in order to removed the duplicated images.

Duplicated Removal



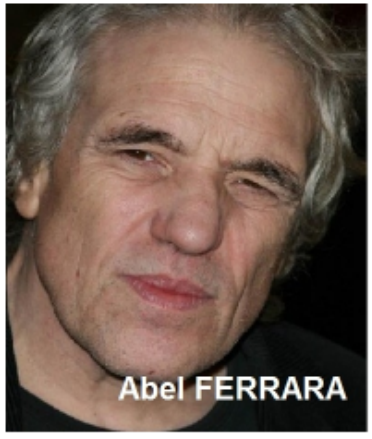


Figura : Example of two images, representing the same photo, but with different resolutions and scales so that the duplicate removal phase is not able see them as duplicates.

Classification

It is necessary a **classification** phase in order to clean the dataset from the images that are not relevant to the associated identity.

We consider as images the bounding boxes extracted by the face detector, in order to focus the classification to the detected faces.

Taking advantage from a **pre-trained CNN** of faces from 1000 identities, we extract from the net the last layer of convolution as feature vector for each images in the dataset. For each identity, a **linear SVM** will be trained.

The feature vectors were extracted by the net using the **MatConvNet** toolkit [4] and the model was trained using the **LIBSVM** library [3].

Classification

In order to reduce the computational complexity, each identity is trained against other K (ex. $K = 5$) identities randomly chosen, taking 50 images from each one.

The images that are used in the training phase, were chosen by taking into account the **ranking** of the search engines, assuming that the higher the ranking the more relevant the images.

It is a simple **binary** classification for each identity and so we need to train each identity separately.

Validation

The faces used in the training of the models might not be completely clean.

For example, if an image contains two faces, each of them will have the same ranking; thus, if the ranking is high, they will both be used in the training phase but only one of them will correctly represent the identity.

Since it is not possible to resolve these problems in the previous steps, we have classification models that have been obtained with a training set containing errors.

Validation

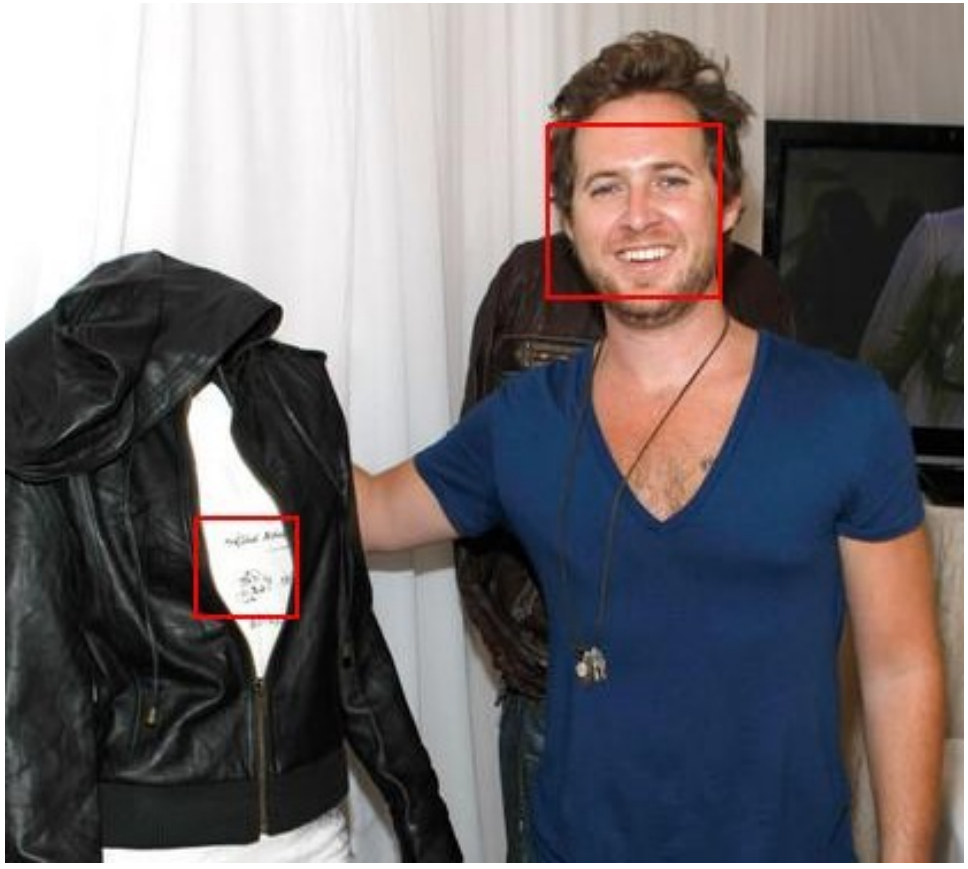


Figura : Example of an image for which the face detector return a face but also an error.

Validation

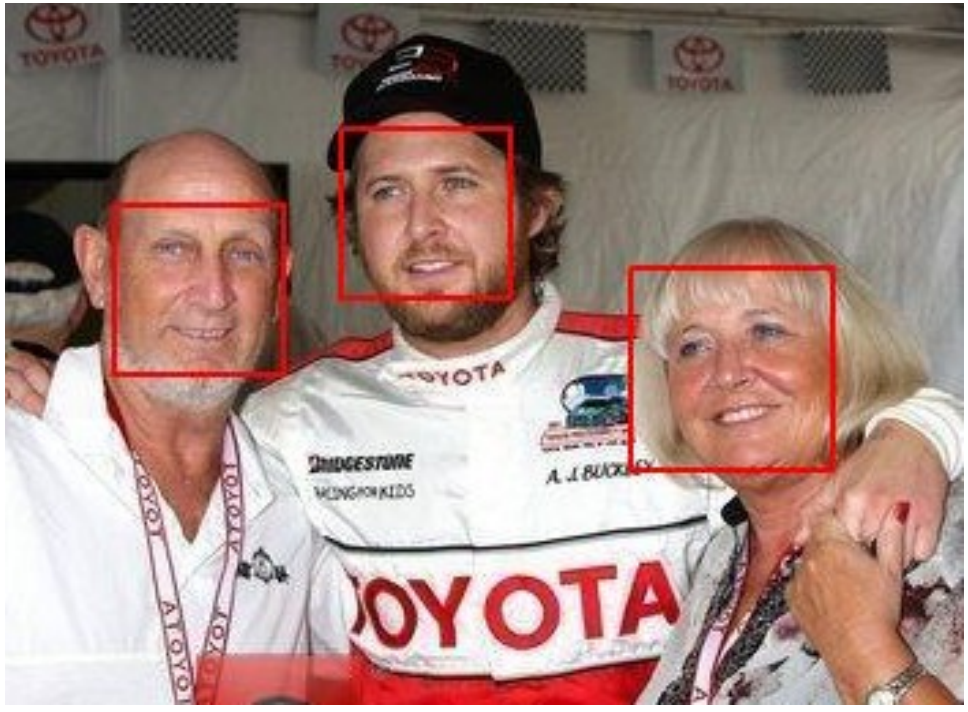


Figura : Example of an image for which the face detector detects more than one face.

Validation

In order to resolve these errors and to evaluated the performance of the procedure, a method for visual validation was implemented.

A **web application** was created for this purpose, thank to which are shown to a **user** the images associated to each identity and the results of the classification.

The images with a **red overlay** represent the ones that the classifier consider not belonging to that identity.

A **user** can interact with the web application, by browsing each identity and the images that are shown in a gallery; the user can also double click an image to **correct** the results of the classification. A user can also decide to completely remove an identity from the dataset.

Validation

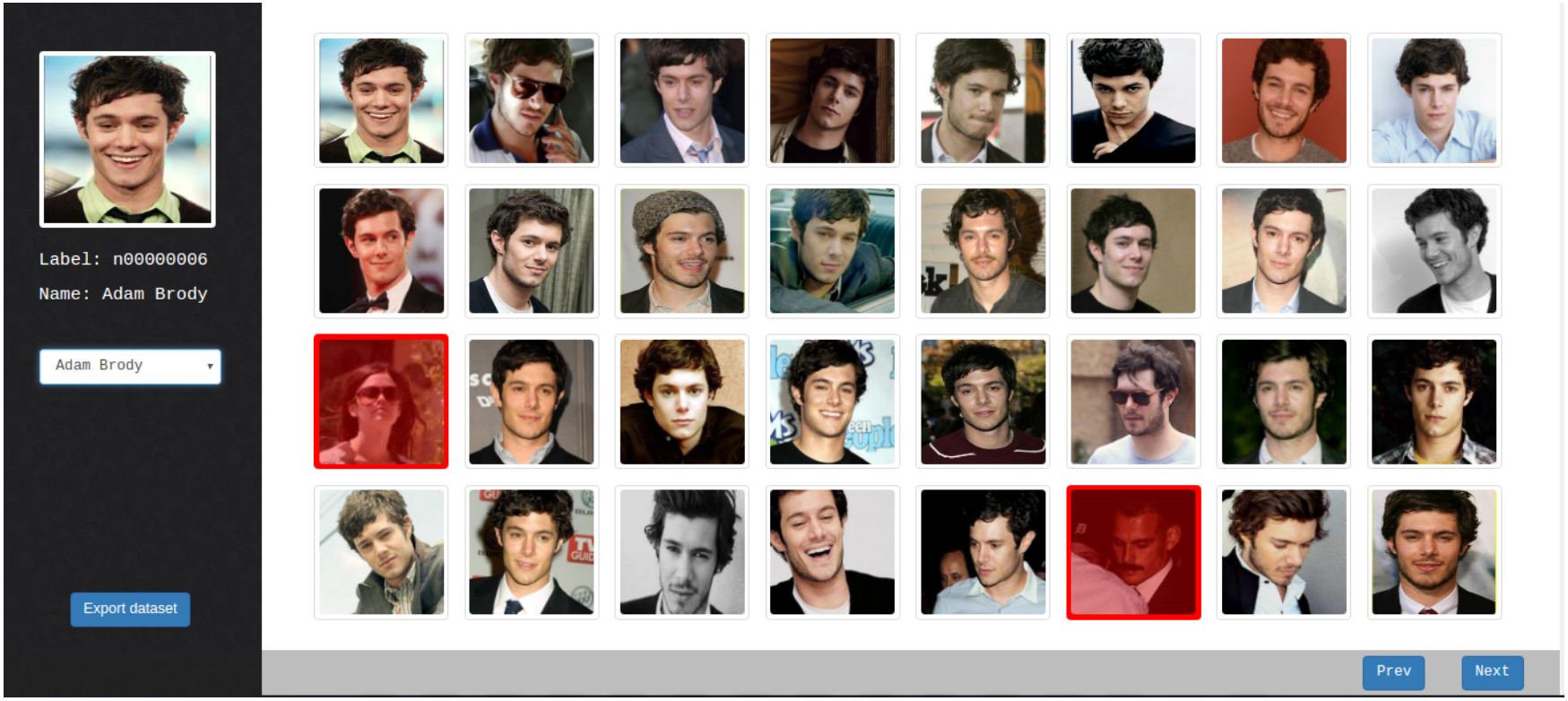


Figura : A screenshot of the implemented web application.

Results

To evaluate the performance of this procedure, **experiments** have been performed with two different datasets:

One dataset (test A) contains 50 identities, such as **celebrities** and **actors**, the other (test B) contains 50 identities, such as **football players**.

The identities for the first dataset are a selection of the ones used by the pre-trained CNN that was used to extract the feature vectors for the images.

Results

Using the **ground-truth** created by the validation phase, it was possible to compare the results of the automatic classification.

Tabella : Accuratezza del clustering

Test	TPR	TNR	FPR	FNR	Accuracy
A	0,993	0,874	0,126	0,007	0,973
B	0,978	0,839	0,161	0,022	0,952

The average number of images per identity for the **prediction** is 409 for test A and 439 for test B; the average number of images per identity after the **validation** is 412 for test A and 448 for test B.

Conclusions

The implemented procedure works as follows.

For each identity, the images are **downloaded** from selected search engines.

A **face detector** is used to detect the faces contained in the images.

A **duplicate removal** method is used to removed images that are duplicate.

Using a **pre-trained CNN**, feature vectors are computed for each image and **linear SVM** is trained for each identity in order to remove the images that do not belong to that identity.

Finally, the results of the classification are validated and corrected by an user thanks to a **web application** specifically developed for this purpose.

Conclusions

However, there are some **problems** that need to be addressed:

- The collection phase is not reliable.
- The duplicate removal phase is only able to remove perfect duplicate.
- Having more than one faces in an image can lead to a training set for the classification that contains error, negatively affecting the computed model.
- An identity can have a name that leads to bad results in the search.
- The whole procedure is quite slow especially for the classification phase and for the visual validation.

References

[1] O. M. Parkhi, A. Vevaldi, A. Zisserman, Deep Face Recognition, British Machine Vision Conference, 2015.

[2] Davis E. King, Dlib-ml: A Machine Learning Toolkit, Journal of Machine Learning Research, 2009, 10, 1755-1758.

[3] Chang, Chih-Chung and Lin, Chih-Jen, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent System and Technology, 2, 3, 2011, 27:1—27:27.

[4] A. Vevaldi and K. Lenc, MatConvNet: Convolutional Neural Networks for MATLAB, Proceedings of the ACM Int. Conf. On Multimedia, 2015.

[5] A. Vevaldi and B. Fulkerson, VLFeat: An Open and Portable Library of Computer Vision Algorithms, 2008.