CS229 Project Report: Automated photo tagging in Facebook

Sebastian Schuon, Harry Robertson, Hao Zou
schuon, harry.robertson, haozou @stanford.edu

Abstract

We examine the problem of automatically identifying and tagging users in photos on a social networking environment known as Facebook. The presented automatic facial tagging system is split into three subsystems: obtaining image data from Facebook, detecting faces in the images and recognizing the faces to match faces to individuals. Firstly, image data is extracted from Facebook by interfacing with the Facebook API. Secondly, the Viola-Jones' algorithm is used for locating and detecting faces in the obtained images. Furthermore an attempt to filter false positives within the face set is made using LLE and Isomap. Finally, facial recognition (using Fisherfaces and SVM) is performed on the resulting face set. This allows us to match faces to people, and therefore tag users on images in Facebook. The proposed system accomplishes a recognition accuracy of close to 40%, hence rendering such systems feasible for real world usage.

1 Introduction

Recently image sharing on the social network platform Facebook has become very popular. Here photo albums are combined with social networking, so individuals can be marked in uploaded pictures. This marking is refered to as 'tagging' on Facebook. Currently this task is done manually, but with a exponentially growing number of images, an automatic solution is desirable. We break this task into three sub-tasks, namely: data collection, face detection and face recognition. The paper's structure follows this processing pipeline. Thus section 2 outlines our method for obtaining images from Facebook. We extract faces from these images in section 3, and finally perform face recognition in section 4. We present our conclusions and ideas for future work in section 5.

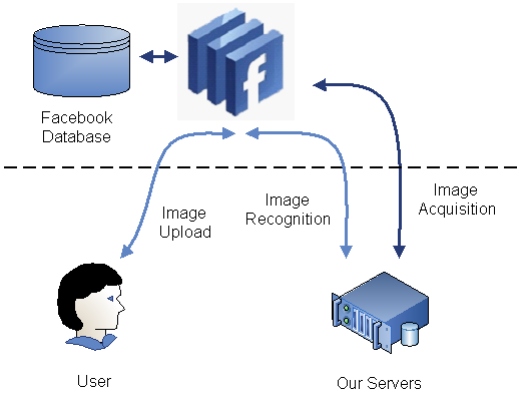

Figure 1: Schematic of the Facebook crawler

2 Collecting data from Facebook

To implement an automatic image tagging system, one first has to acquire the image data. For the platform we have chosen to work on, namely Facebook, this task used to be very sophisticated, if not impossible. But with the recent publication of an API to their platform, this has been greatly simplified. The API allows third party applications to integrate into their platform and most importantly to access their databases (for details see figure 1 and [3]). We have set up an application on this platform, which allows the user to gather image data on himself and his friends. This data is in turn fed into the image processing pipeline to learn the user's faces.

The Facebook platform allows for users to upload pictures and tag people on these images. Here *tagging* means the process of placing a box around the person's faces in the image and identifying them. Figure 2 shows an example of a placed tag. In Facebook's current tagging implementation, a constant-sized rectangle is placed over a persons face, and only the center of the box is stored.

To date, these tags must be manually placed by users who recognize the faces of their friends in an image. We want to improve on this and have the tagging operation performed automatically when a picture is added

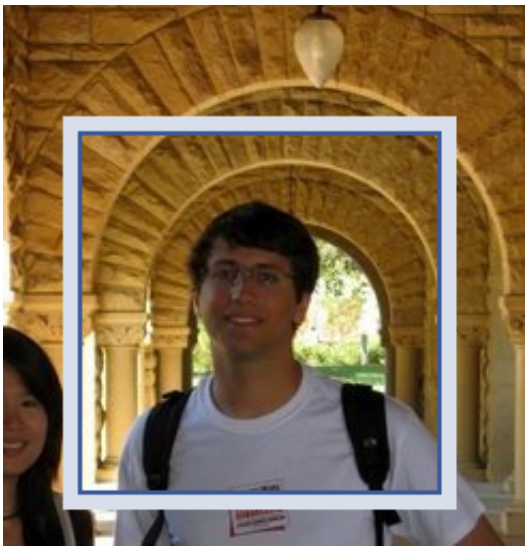

Figure 2: A typical tagged Facebook photo

Figure 3: A typical set of detected faces for a Facebook user

to the platform. However, the tags already placed by users are very useful for the learning process, since they supply the algorithm with numerous training examples of a person's face (if we were limited to profile pictures, we would have only one training image per person).

For our current training set, our crawler extracted data for 100 individuals from the Facebook platform. This includes 92 profile images plus 2509 photos which contain at least one tag. In total 2928 tags have been collected, hence some of the photos contained more than one person.

Examining the data, it is notable that for about 50 percent of the users, we have less than five images, which will make the learning process hard. Also, people strike very different poses and are taken from different angles, which will be an additional challenge for the recognition task (see figure 3 for some examples).

3 Face Detection

After the data (i.e. images, tags and user information) has been crawled and downloaded, it needs to be processed to put it in an appropriate format for the following steps. Depending on the learning algorithm, different post processing steps may be applied later, but the most crucial is the reduction of the images to snapshots of faces.

Our face detection task can be characterized by two properties. Firstly, it needs to be very robust, since individuals may appear with turned or partly occluded faces. Secondly, we need an algorithm which performs the detection process with low computational requirements, since a large number of images needs to be processed and the user will not tolerate waiting a long time for his images to be tagged.

We experimented with two different algorithms for the face detection task, one based on Haar-features by Viola and Jones [9] and one based on fixed size, separable patches named FDlib [4]. The results of both algorithms are given in table 1. We notice first of all that for about 35 percent of the tags no face could be recognized. This is partly due to non-available images (i.e. download error, privacy retriction), which account for about one quarter of the missed faces. The other three quarters are situations where the algorithm was not able to detect a face. Here often the algorithm is to blame, but in a significant amount of cases, people just tagged non-face objects as 'people'.

Both algorithms detect roughly the same amount of faces, with Viola-Jones yielding a higher rate on faces which actually belong to the person indicated[1]. The other two categories for returned images are ones which feature the face of a different person and ones with no face at all. The latter is due to misclassification of an object as a face, whereas the former is due to the presence of multiple people on an image. Both error categories will have consequences of varying severity, depending on whether face detection is used to find faces in a newly uploaded picture or generate training data from existing pictures).

Given the results in table 1, we selected Viola-Jones for face detection, since it not only yields a higher rate of correct faces detected, but also has an optimized implementation in Intel's OpenCV library [5]. The results can be improved further if both algorithms are run in parallel and faces are only chosen if they are found by both algorithms. Furthermore Viola-Jones allows for adjustment of the weights of its weak classifiers, so the algorithm could be adapted to typical poses of Facebook users.

Face detection errors within the training set lead to inferior recognition rate in the following face recognition. These images can be considered as outliers, and we attempt to filter them out.

Two recently introduced algorithms, Locally Linear

[1] For error evaluation, all face data was hand labeled, so a certain error due to human involvement should be assumed.

	Total Tags	Total Faces Detected	Face-person Identical	Face-Person mismatch	No Face Present
Viola-Jones	2928	1987	1391	369	227
FDLib	2928	1953	781	139	1033

Table 1: Face detection results

Embedding (LLE) [6] and isometric feature mapping (Isomap) [8], have acquired a growing interest among researchers because of their excellent performance in dimension reduction for nonlinear data. The key idea here is that all faces of a person lie on a manifold, whereas outliers are detached from the manifold's surface. So far this approach has mainly been applied to facial expressions [6].

LLE computes low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs expected to lie on the manifold. Isomap is also a nonlinear dimensionality reduction algorithm. But unlike LLE, it performs low-dimensional mapping by tracing the shortest paths confined to the manifold of observed inputs. Since both algorithms are unsupervised ones, they do not require a training phase.

For LLE, the number of nearest neighbors (the most important parameter for the algorithm) is chosen to be seven after trial and error for the Facebook images, which is at the lower bound of the recommended range [7]. The original dimension of an image is 60 by 60, a total of 3600 dimensions, which we map to two dimension data using LLE and Isomap. The outlier classification is performed by calculating the center of the low-dimensional data, and labelling images beyond a certain distance as outliers. In table 2 we have listed the reduced error rates achieved. We only obtained a marginal reduction in error rate by employing LLE and Isomap. Therefore further research is needed on how to eliminate outliers, or how the later face recognition algortihms can be made more robust towards outliers.

Original	LLE	Isomap
32.4%	31.7%	28.6%

Table 2: Error rates before and after LLE and Isomap filtering

4 Face recognition

Having extracted the faces of each person, we can use them as training data to be able to identify faces in uploaded images. To evaluate our recognition performance, we have created two sets from the crawled data: a reduced set, with six people and 50 photos per person (referred to as '6 people'), and a 'real-world' set, with over 30 people, and from 10 to 200 photos per person. The first serves primarily to evaluate the algorithms. The second is typical of the set of all photos accessible in the peer group of a user. Both sets were randomly divided into training and test sets with an $80/20$ ratio. We separated each set into two further sets, one 'brute' set containing the raw data generated by the face detection step, and one 'hand filtered' set from which outliers have been removed manually.

There is an abundance of face recognition algorithms available to choose from, and the face recognition problem has been well studied. However, our usage does not correspond to their usual test-cases: face recognition algorithms are generally evaluated by their performance on a few photo databases available to the academic community, which are made up of perfectly lit and framed photos. In contrast, the task we present features numerous poorly lit and chaotically taken snapshots.

Hence several algorithms were tested. The following algorithms were evaluated on the described test sets:

- Eigenfaces
- Fisherfaces
- Support Vector Machines

In the following two subsections we will present the performance of these algorithms.

4.1 Eigenfaces and Fisherfaces

We initially attempted to use the Eigenfaces algorithm on the detected Facebook images. Eigenfaces uses the Principal Component Analysis (PCA) to choose a dimension-reducing linear projection that maximizes the variance of all the projected samples. However, the mapping direction selected by PCA does not take into consideration the data's class structure, or the fact that the maximum variance direction achieved for all data points may not be good for classifying points into different classes. The Eigenfaces algorithm fails completely for the Facebook images, and returns an almost random recognition rate in our tests.

Another algorithm, Fisherfaces, which use Fisher's Linear Discriminant and takes into consideration both the between class covariance (or scatter) matrix and

in-class covariance matrix, tries to find a mapping direction that maximizes the variance between different classes, and is thus more suitable than Eigenfaces for classification problems. The recognition rate for Fisherfaces, shown in table 3, is elevated compared to Eigenfaces but still low. This was to be expected as Eigenfaces is a special case of Fisherfaces and generally has lower performance [1].

'6 person' set		'Real world' set	
Brute	Hand filtered	Brute	Hand filtered
21.6%	28.3%	5.5%	6.8%

Table 3: Fisherfaces recognition results

4.2 Support Vector Machines

The results obtained by Support Vector Machines (SVM) are significantly better than those of Fisherfaces. In the following paragraphs we outline our SVM approach.

4.2.1 Pre-processing

Training an SVM directly on raw images is not very effective in most cases, therefore various pre-processing steps were evaluated to improve the overall performance:

Transforms There are a variety of tranformations that can be applied to images to extract features which are more relevant for face recognition. In particular we investigated the use of Haar wavelet transformation. It is one of the oldest and most basic transformations, obtained by convolving the image matrix with a step function. Although Haar features have demonstrated their power (i.e. with the Viola-Jones algorithm used for face detection), they did not lead to superior results.

Scaling It is desirable to scale all the features to occupy a constant interval (e.g. $[-1, +1]$). This is not really an issue for raw images (where the features are pixel values in $[0, 256]$), but is important if a transformation has been applied.

4.2.2 Parameters

When it comes to actually training a SVM, the key parameters are the kernel used, the cost coefficient C and the termination criterion ϵ. These were selected as follows:

Kernels Making an appropriate choice of kernel is key for the SVM. Three types of kernels were evaluated: linear, polynomial (with degree 3), and Radial Basis Function (RBF)[2]. While the RBF kernel can produce the richest behavior, in practise our most severe problem by far was overfitting the training data, hence the simpler models actually gave better results (refer to section 4.2.4 for details).

Cost coefficient and termination criterion To determine values adapted to our case, we scripted a grid search, running the algorithm for each (C, ϵ) tuple on a logarithmicly scaled grid and returning the values which gave the least leave-one-out cross-validation error. The grid search is extremely costly in execution time, so we ran it once over a random sub-sample of the full training set, and kept the resulting parameter values ($C = 8$, $\gamma = 0.125$). The same procedure has been used to find the optimal RBF coefficient γ.

4.2.3 Implementation

Looking for a fast implementation, we initially based our classifier on the open-source C++ library libsvm [2], which has the advantage of being optimized, stable and relatively well documented. Eventually we built our own system in C++ to address our needs better and to be able to deploy our system on the Facebook platform.

4.2.4 Results

Using the pre-processing and choice of parameters described above, tables 4 and 5 reports the accuracy in face recognition.

	Linear	Polynomial	RBF
Brute images	41.6%	35.0%	18.1%
Hand filtered	42.2%	28.9%	20.0%
Haar wavelet	37.8%	35.5%	20.0%

Table 4: 6 person training set results

	Linear	Polynomial	RBF
Brute images	25.4%	21.9%	11.5%
Hand filtered	37.1%	28.3%	15.2%

Table 5: 'Real world' training set results

The most significant figure is the 37.1% accuracy on the large training set. One has to keep in mind random identification yields accuracy of below 5% for this set,

[2]The Radial Basis Function is defined by: $K(x_i, x_j) = exp(-\gamma \left\| x_i - x_j \right\|^2)$

and given the challenging nature of the images this rate is remarkable.

For the other figures, the most striking observation is that the more complex the kernel, the worse the results. One would expect a more sophisticated kernel to yield better results, but our results indicate the reverse. This is probably because our SVM tends to overfit the training data, with the number of support vectors of the same order as the number of training examples. That can be explained by the nature of our training data: our base training examples are extremely diverse, containing photos of the same person with different poses, lighting conditions, focus and/or apparel. The frontier between classes is therefore very hard to define.

The second observation is the impact of preprocessing the data. Applying the Haar wavelet transformation has little impact; this is most likely due to the fact that the 1D-Haar wavelet transform used fails to expose more significant features (contrast here Viola-Jones, where 2D-Haar wavelets are used). A far greater gain in accuracy is achieved by hand-filtering outliers: it yields a substantial improvement in terms of accuracy on the large training set (its lack of impact on the 6 person training set is due to the fact that those training images were already of above-average quality, and so few were filtered out).

5 Conclusion

Automatic tagging in Facebook is challenging from both the face detection and the face recognition standpoints. The huge variations in pose, facial appearance, lighting conditions and image quality all affect facial detection accuracy. We find Viola-Jones' face detection to be the most successful algorithm on Facebook data, while outlier filtering with LLE and Isomap has little effect. For face recognition, algorithms which are reported to have high recognition rate for standard academic face databases, such as FisherFaces, gave poor results on our Facebook database. Using SVM, we achieved a final recognition rate of close to 40%, a rate which makes a publicly deployed facial tagging application feasible.

For our automatic tagging system to be widely adopted, the most important factor is still the detection and recognition rate. In our future work, we seek to continue improving the facial detection and recognition rate by developing and optimizing image processing algorithms adapted to Facebook data. Additional information from Facebook can also be utilized to boost the recognition rate. For example, if two people (close friends) consistently appear in photos together, then the recognition of one of them in a new photo increases the probability of the presence of the other. The integration of such priors promises to significantly increase the recognition rates by fully leveraging the strengths of the Facebook platform. Another avenue we wish to explore is returning a set of most likely matches, instead of the one most likely. This will clearly increase the probability of the correct person being among our results, and could thus increase the usability of our application.

References

[1] P.N. Belhumeur, J. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *Proc. ECCV*, pages 45–58, 1996.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. 2001.

[3] Facebook Developer Documentation. http://developers.facebook.com/, 2007.

[4] W. Kienzle, G.H. Bakir, M. Franz, and B. Scholkopf. Face Detection Efficient and Rank Deficient. *Advances in Neural Information Processing Systems*, 17:673–680, 2005.

[5] Open Source Computer Vision Library. http://opencvlibrary.sourceforge.net/.

[6] L.K. Saul and S.T. Roweis. Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds. *Journal of Machine Learning Research*, 4(2):119–155, 2004.

[7] S. Schuon. Locally linear embedding and its application. *Thesis, Technische Universitaet Muenchen*, 2007.

[8] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–23, 2000.

[9] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, 1:511–518, 2001.