Unsupervised Face-Name Association via Commute Distance

Jiajun Bu[1], Bin Xu[1], Chenxia Wu[1], Chun Chen[1], Jianke Zhu[1], Deng Cai[2], Xiaofei He[2]

[1]Zhejiang Provincial Key Laboratory of Service Robot
College of Computer Science, Zhejiang University, Hangzhou, China
{bjj,xbzju,chenxiawu,chenc,jkzhu}@zju.edu.cn
[2]State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, Hangzhou, China
{dengcai,xiaofeihe}@cad.zju.edu.cn

ABSTRACT

Recently, the task of unsupervised face-name association has received a considerable interests in multimedia and information retrieval communities. It is quite different with the generic facial image annotation problem because of its unsupervised and ambiguous assignment properties. Specifically, the task of face-name association should obey the following three constraints: (1) a face can only be assigned to a name appearing in its associated caption or to *null*; (2) a name can be assigned to at most one face; and (3) a face can be assigned to at most one name. Many conventional methods have been proposed to tackle this task while suffering from some common problems, *e.g.*, many of them are computational expensive and hard to make the null assignment decision. In this paper, we design a novel framework named face-name association via commute distance (FACD), which judges face-name and face-null assignments under a unified framework via commute distance (CD) algorithm. Then, to further speed up the on-line processing, we propose a novel anchor-based commute distance (ACD) algorithm whose main idea is using the anchor point representation structure to accelerate the eigen-decomposition of the adjacency matrix of a graph. Systematic experiment results on a large scale and real world image-caption database with a total of 194,046 detected faces and 244,725 names show that our proposed approach outperforms many state-of-the-art methods in performance. Our framework is appropriate for a large scale and real-time system.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Performance

MM'12, October 29–November 2, 2012, Nara, Japan.

(a) Britain's Prime Minister **Gordon Brown** and U.S. Democratic presidential candidate **Barack Obama** shake hands before meeting. April 17, 2008.

(b) **Novak Djokovic** ended **Roger Federer**'s winning run at his home-town tournament to lift the *Davidoff* Swiss Indoors title in Basle this afternoon.

Figure 1: Examples of image-caption pairs. Each name in bold is associated with a face in the image while the italic name is not matched.

Keywords

Face-name association, Unsupervised, Commute distance

1. INTRODUCTION

There are a great number of images accompanying by text captions on the Internet. A high percent of these images contain people faces and their captions contain people names. A common observation for these image-caption data is that the name of a face in the image is very likely mentioned by the corresponding caption and vice-versa. This motivates the recent research interest of exploiting the latent associations between faces and names automatically for image-caption pairs. The task is called face-name association [1, 8, 14]. Fig.1 shows two typical examples. The persons in the images are all mentioned by their captions.

Unsupervised face-name association is helpful to generate a huge amount of training annotations at a low cost for many applications, such as face image (instance) retrieval and automatic face annotation. At the same time, it is an important technique for many web systems, such as the social network (*e.g.* Facebook) and the photo sharing service (*e.g.* iPhoto). The unsupervised face-name association problem can be seen as the following challenge: for a given image and its corresponding caption (also called an image-caption item in this paper), determine whether there exist some faces in the image and some names in the caption; if so, point out the latent associations between faces and names,

without any explicit manual intervention. With the help of face detector[1] and named entity detector[2] [14], we can directly filter out the image-caption pair without a face in the image or a name in the caption. That is, the remain key task is the automatic face-name matching for a given small list of faces and names, which are extracted automatically from the image-caption pair.

This seems like a traditional classification (or facial image annotation) problem that each of the names in the caption is a class label and we need to assign the labels to the faces. However, the face-name association problem is essentially different with a standard supervised classification problem due to the following important properties:

- Their are no training data: To train a classifier, one needs to collect the training data. In our scenario, we only have a large collection of image-name items, and each item may contain multiple faces and multiple names. There are no exact associations. This problem is essentially an unsupervised learning problem.
- The associations are ambiguous: Not every name mentioned in the caption appears in the image, and, vice-versa, not every face in the image is mentioned by the caption. That is to say, the match result is not necessary to cover all the names and faces in a pair. This is a very common case in real world data, *e.g.*, in Fig.1, the name *Davidoff* is not related to any person appearing in the image. However, for a general classification algorithm, it's possible to match multiple faces to a single name, and it is unknown how to make the face-null association if we don't make any modification.

Many conventional methods have been proposed to tackle the problem of unsupervised face-name association [1, 8, 14]. These methods suffer from some common problems. For example, many of them are computational expensive, i.e, they are not appropriate for a large scale real-time system. Moreover, some methods are hard to make *null* assignment decisions (a face is not mentioned by the caption). To overcome these drawbacks, in this paper, we design a new framework named FACD (stands for *Face-name Association via Commute Distance*). FACD has two stages: one off-line preprocessing stage and one on-line face naming stage. In the off-line stage, we build an inverted index structure for the large scale name and face database, based on which, in the on-line stage, we efficiently handle the image-caption items one by one. Specifically, we adopt commute distance (CD) on a graph as the metric to match the faces and names, receiving a significant performance improvement over many state-of-the-art methods. Moreover, to further speed up the computation, we propose a novel anchor-based commute distance (ACD) method, which is much faster than CD especially on a large scale database. This nice property makes the proposed ACD more appropriate for a large scale real-time face-name association system. A typical scenario is as follows: one user uploads an image including some people faces as well as some text description including some names to a web system and then, the system automatically match the corresponding names and faces in a short time.

For experiments, we use the large scale database FAN-Large [14], which is the largest and most realistic database supporting the research of face-name association. The whole database contains 125,479 image-caption items with a total of 194,046 detected faces and 244,725 names.

The main contribution of this paper include: (1) we design a new framework FACD, appropriate for a large scale and real-time face-name association system. (2) we propose to solve the face-name association problem via commute distance algorithm, by which both the face-name and face-null assignment problems can be integrated into a unified framework; (3) we propose a novel anchor-based commute distance algorithm to further speed up the computation of commute distance.

The rest of this paper is organized as follows. We briefly discuss some related works in section 2 and some preliminary knowledge in section 3. The framework FACD and the anchor-based commute distance algorithm are described in section 4. We present experimental results and comparisons in section 5. Finally, we provide a conclusion in section 6.

2. RELATED WORK

In this section we briefly review some closely related works to our paper. They belong to the big topic of automatic face annotation, including the model-based and retrieval-based face annotation and the face-name association.

Generally, the model-based face annotation is formulated as an extended face recognition problem, in which many traditional classifiers are trained from a collection of labeled face instances. Typical examples include face annotation for images [2, 27, 28, 30] and for videos [15, 23]. In [30], the authors proposed a transductive kernel fisher discriminant scheme for the task of face annotation. The main idea is to solve the Fisher's discriminant using deformed kernels incorporating the information of both labeled and unlabeled data. Although the annotation accuracy is high in many works, these methods suffer from some problems in common. For example, it is very expensive and time consuming to get a large collection of labeled face instances and the annotation performance decreases when the number of classes (names) increases. Moreover, after gathering some new labeled data, these systems need to re-train the model which is time consuming.

Instead of training classifiers for model-based face annotation approaches, some recent researches focus on exploring retrieval-based face annotation algorithms [18, 19, 21]. These algorithms utilize the information of a large amount of weakly name-face labeled data collected from public search engines (*e.g.*, Google) to assign the most possible name to a user provided face image. This is a promising way to solve the face annotation task via utilizing the content-based image retrieval technique [3, 4, 6, 9, 22, 26] in mining massive weakly labeled facial images on the web. However, retrieval-based face annotation algorithms have to face two key challenges: the first is how to retrieve most similar facial images from a large scale facial image database efficiently and the second is how to exploit the noisy and incomplete web images and their weak label information for the face annotation task [19]. The work [21] addressed the first challenge via employing both local and global features as facial image representation. In [19], the authors proposed a weak label regularized local coordinate coding technique, which learned sparse features via local coordinate coding principle as well as the graph-based weak label regularization.

Recently, many researchers focus on the topic of automatic

[1] `http://torch3vision.idiap.ch`
[2] `http://opennlp.sourceforge.net`

face annotation for captioned images (the task of face-name association). That means, the faces appearing in an image can only be assigned to a name mentioned by the corresponding caption or to *null*. Most of the existing methods [1, 8, 14] can be regarded as a type of constrained clustering approach [14], where each cluster of faces is assigned to one name. The algorithm of [1] is a generative model, which treats image captions as bags of names, disregarding any contextual cues. The model associates a Gaussian density in the appearance feature space to each name as well as to the *null* label. In [8], the authors constructs a similarity graph with nodes representing faces and edges weighting by the similarity between two faces. Then, the algorithm searches a number of dense subgraphs with each one corresponding to a person's name. The objective function is formulated as a maximization of the sum of edge weights within each subgraph, and the iterative solution stops at a local optimum.

3. PRELIMINARIES

We introduce some preliminary knowledge including the the random walk [13] technique and two distance measures highly related to the random walk – the hitting time and the commute time [7, 10, 13, 16] in this section. They are helpful for understanding our proposed algorithm in this paper.

3.1 Random Walk

The **random walk** [13] defined on a graph is a sequence of nodes represented by a finite Markov chain. Generally, a graph can be denoted as $G = (V, E, W)$, where V is a set of nodes ($|V| = n$) in which each node represents a data point, $E \subseteq V \times V$ is a set of edges ($|E| = m$) connecting related vertices, and W is an adjacency matrix recording the pairwise weights between nodes. The element $w_{ij} > 0$ if (i, j) is an edge, otherwise, $w_{ij} = 0$. The probability that the random walk on node i at time t jumps to node j at time $t + 1$ is determined by the edge weight on the graph:

$$\Pr(state(t+1) = j | state(t) = i) = p_{ij} = w_{ij}/d_{ii}, \quad (1)$$

where $d_{ii} = \sum_j w_{ij}$ is the degree of the node i. Thus the transition matrix P is given by $P = D^{-1}W$. Let $\pi(t) = [\pi_1(t)], \pi_2(t), \ldots, \pi_n(t)]^T$ be the state probability distribution at time t, the next distribution is calculated by $\pi(t + 1) = P^T\pi(t)$. It is not hard to see that, the stationary distribution of the random walk is proportional to the vertex degree [29], *i.e.*, $\pi = D\mathbf{1}/V_G$, where $V_G = \sum_{i=1}^{n} d_{ii}$ is the volume of the graph. We get the conclusion from the observation: $P^T\pi = WD^{-1}D\mathbf{1}/V_G = D\mathbf{1}/V_G = \pi$.

3.2 Hitting Time and Commute Time

The **hitting time** $h(i, j)$ is the expected steps that a random walk will expend from node i to node j for the first time [13]. It is defined as:

$$h(i,j) = \begin{cases} 0 & \text{if } i = j \\ 1 + \sum\limits_{k \in \mathcal{N}(i)} p_{ik} h(k,j) & \text{otherwise,} \end{cases} \quad (2)$$

where $\mathcal{N}(i)$ is the set of the nodes each has an edge connected with i and

$$p_{ik} = \begin{cases} w_{ik}/d_i & \text{if } (i,k) \text{ belong to an edge} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

is the transition probability from node i to node k.

The **commute time** $c(i, j)$ is the expected steps that a random walk starting from node i will take to arrive at node j and then go back to i again for the first time [13]. It is defined as:

$$c(i, j) = h(i, j) + h(j, i). \quad (4)$$

It is easy to show that the commute time is a metric [7], so it is also called as **Commute Distance**.

4. ALGORITHM FRAMEWORK

In this section, we describe our face-name association framework FACD in detail. FACD has two stages: one off-line preprocessing stage and one on-line face naming stage. In the off-line stage, we build an inverted index structure for the large scale name and face database, based on which, we efficiently handle the image-caption pairs one by one in the on-line stage.

4.1 Problem Formulation

Given a collection of web image-caption pairs with each image having at least one face and each caption has at least one name (We filter out the pairs without a face or a name), find the associations between faces and names for each pair,without any explicit manual help. Moreover, the result should obey the following constraints [14]:

C1: A face can only be assigned to a name appearing in its associated caption, or to *null* if there is no corresponding name in the caption.;

C2: In an image-caption pair, a name can be assigned to at most one face;

C3: In an image-caption pair, a face can be assigned to at most one name.

These constraints are widely considered and used in many existing face-name association systems [1, 8, 14].

4.2 Off-line Stage

Similar to the term-document inverted index used in Information Retrieval, we construct the name-item inverted index. Here an item is a image-caption pair. Fig.2 is a schematic diagram. We first associate each item to all the names detected in its caption and then, extract the faces in each item to form a name-face index. With this index, we can efficiently retrieve all the related faces given a name (appearing in the same item), which serves the foundation of the efficient processing in the on-line stage.

Unfortunately, the face lists contain many noisy faces – one image may have several faces. To improve the quality of the index, we make effort to re-rank the faces in each list, expecting to put the correct faces on the top positions, and then, try to filter out some mismatched (noisy) faces. Without any manual help, our processing is inspired by some heuristic ideas as follows.

H1: For each list of faces associated with a name (each line in the index), the correct person appears the most times in a high probability.

Since the face images of the same person tend to be similar to each other, in each line in the index, the correct faces (to the name) tend to have higher sum of similarities to all the other faces. Thus, for each line of the index, we compute the

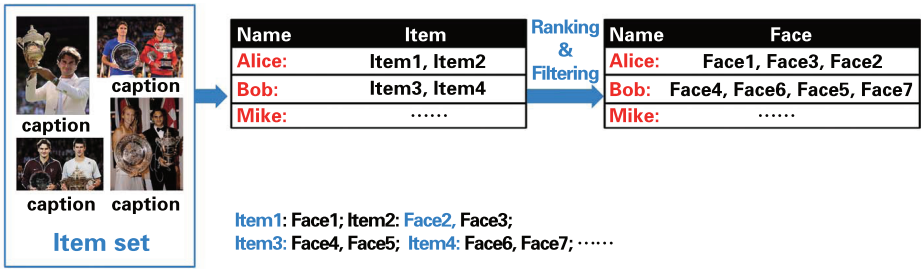


Figure 2: An example of the inverted index established in the off-line stage. An Item represents an image-caption pair. The item set is organized into a name-item inverted index (middle). Then, after ranking and filtering, we have a name-face index (right).

pairwise visual similarities between all the faces. Then we re-rank the face list by the sum of similarities as a metric, which is defined by (the higher the better):

$$S_i = \sum_j s_{ij}, \quad i = 1, \ldots, r \tag{5}$$

where s_{ij} is the similarity of faec i and j in the list and r is the number of faces. The visual similarity is computed by

$$s_{ij} = \exp\left(-d^2(\mathbf{v}_i, \mathbf{v}_j)/\sigma\right), \tag{6}$$

where σ is the heat kernel parameter and $d(\mathbf{v}_i, \mathbf{v}_j)$ is the Euclidean distance of face i and j.

H2: One name can not refer to two or more faces belonging to one image.

We set the rule that buddy faces (appearing in the same image) can not be associated with the same name. So we filter out the the faces (or put them to the tail of the list) which have a buddy face in front of the list. For example, Fig.3 is a toy case of the filtering scheme. The first line of faces are sorted by S_i and the faces connecting with a blue broken line are buddy faces. So we drop the faces f_2,f_4,f_7 and f_8, getting the second line of faces associated with the name.

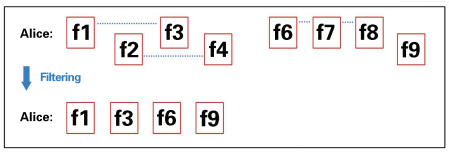


Figure 3: An example of the filtering strategy for a sorted face list. The faces connecting with a blue broken line are buddy faces.

4.3 On-line Stage

The most important property that a face-name association system should have is the low response time for an on-line request. That is to say, the image-caption pairs should be processed one by one (independently) and each processing is very light-weight.

Based on the index architecture established in the off-line stage, one may design a simple strategy called Majority Voting (MV). For a given image-caption pair, the names in the caption are used as queries to get a set of *candidate* faces from the index. Each candidate face is related to a name. Then for each request face, we could search the most visually similar faces from the candidate set, and assign it the name with maximum number of related faces.

Although the majority voting algorithm is very simple to implement, it is not conform with the constraint C2 – two or more faces may be assigned to the same name. At the same time, it is hard to make the *null* assignment (constraint C1) when there are more names than faces.

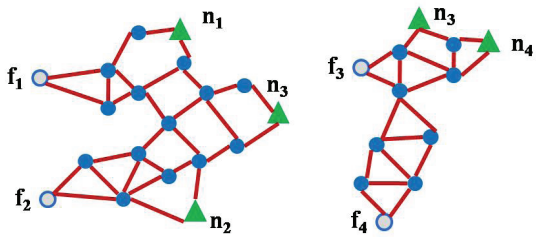
4.3.1 *Association by Commute Distance*

To conform with all the association constraints described in section 4.1, we need a unified metric to judge which face-name pairs are more likely to be correct pairs, and which faces are not mentioned by the names, and which names are not associated with any face in the image. To achieve these goals, we propose to use a graph structure to model the face-name relationships. In the next, we first give two toy examples to show the intuitive insight and then describe our algorithm in detail.

Fig.4 shows two examples: faces and names are denoted by circles and triangles respectively. Nodes $f_1 - f_4$ are the request faces. Intuitively, in Fig.4(a), we want f_1 associate with n_1, f_2 associate with n_2, and the name n_3 is not used. In Fig.4(b), we want f_3 is associated with n_3, and f_4 is assigned to *null* since it is far away to all the names on the graph. Our judgements are made by the distance estimation on the graph. For an arbitrary and complex graph, we need a unified distance measurement. The commute distance (CD) is a reasonable choice. Recall its definition for two nodes on a graph – it is the expected steps that a random walk will take from the first node to the second and go back again for the first time. It is different with the *shortest path* – even if two nodes have a short path, they still probably have a long commute distance if the subgraph between them is very dense [7]. It is more like a average jump distance estimation on a graph. For this point, the commute distance is conform with our intuitive insight above.

Our algorithm's detailed steps are as follows: An unified *undirected* connected graph is built for an item, whose nodes represent the faces (request faces and candidate faces) and names. The candidate faces are formed by top-p faces from the name-face index for each name of the item. For example, if the target item has a names and b request faces, the graph size (number of nodes) for an item is $(a+b+a*p)$. Parameter p will be discussed in the experiment section.

The candidate faces are bridges of associations between request faces and names. Two types of edges need to be de-



(a) Example1 (b) Example2

Figure 4: Two toy graph examples in which circle nodes representing faces and triangle nodes representing names. $\mathbf{f_1 - f_4}$ denote the request faces.

fined: For face-face edges, the weights represent visual similarities of faces measured by Eq.(6); For face-name edges, the weights are 1 or 0 representing whether a candidate face and a name come from the same index list or not. (Note that only the request faces' closest candidates are judged.) So we construct an adjacency matrix W with each element W_{ij} saving the corresponding edge weight.

Instead of running a random walk on the graph and computing the average jump steps for each two nodes, the commute distance can be computed via the Moore-Penrose pseudoinverse of the graph Laplacian matrix . Let $L = D - W$ be the graph Laplacian matrix (D is the degree matrix with $D_{ii} = \sum_j W_{ij}$) and L^+ be the pseudoinverse, then the commute distance $c(i,j)$ is [7, 10]:

$$\begin{aligned} c(i,j) &= V_G(l^+_{ii} + l^+_{jj} - 2l^+_{ij}) \\ &= V_G(\mathbf{e}_i - \mathbf{e}_j)^T L^+ (\mathbf{e}_i - \mathbf{e}_j), \end{aligned} \tag{7}$$

where $V_G = \sum_{i=1}^n d_{ii}$ is the graph volume, l^+_{ij} is the (i,j) element of the matrix L^+, and $\mathbf{e}_i$ is a n dimensional column vector with a $\mathbf{1}$ at location i and all $\mathbf{0}$ elsewhere.

Using the equation above, we can directly compute the commute distances between each request face and each name in an image-caption pair. Then, the remain problem is how to associate them automatically and correctly. A big advantage of the commute distance approach is that the distances are all comparable. That is to say, a shorter distance pair is more likely to be a right association. What's more, if we can not find any name has close commute distance to a face, we just assign *null* to the face. Thus we tackle the *null* assignment problem naturally in the same framework. The main steps of the association after we getting the commute distances is described in **Algorithm 1**. $\{P_i = (f^i_a, n^i_b)\}$ is a list of face-name pairs, sorted by the commute distance CD_i corresponding to P_i (ascending). The threshold ϵ is a predefined parameter and we will discuss how to decide this parameter in the experiment.

Using Algorithm 1, we could get all the face-name associations. This is our *Face-name Association by Commute Distance* (FACD) framework. In the next section, we discuss how to accelerate the computation.

4.4 Anchor-based Commute Distance

If there are q nodes (including the candidate faces, request faces and names) in a graph built in the on-line stage, the eigen-decomposition of the graph Laplacian can be written

Algorithm 1: Face-Name Association After Commute Distance Computation

Input : $\{P_i = (f^i_a, n^i_b)\}$, $\{CD_i\}$, $\{f_k\}$, threshold ϵ.
//$\{P_i\}$ is a list of face-name pairs, ascendingly sorted by the commute distance CD_i. $\{f_k\}$ is the set of request faces. ϵ is a predefined threshold.
Output: face-name association list G.
Initialize i = 1
while $CD_i < \epsilon$ **do**
  if f^i_a *is not named* && n^i_b *is not used* **then**
    add P_i to G;
  end
  i = i + 1;
end
//F is the number of faces in an item
for $k = 1$ *to* F **do**
  if f_k *is not named* **then**
    add (f_k, NULL) to G;
  end
end

as $L = U\Sigma U^T$, in which $\Sigma = diag(\lambda_1, \lambda_2, \ldots, \lambda_q)$ $(0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_q)$, and $U = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_q)$ where $\mathbf{u}_i$ is the eigenvector corresponding to the eigenvalue λ_i. The pseudoinverse of L is (assume that the nonzero eigenvalues of L start from λ_2):

$$L^+ = \sum_{i=2}^{q} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T. \tag{8}$$

Thus, we need to compute the whole eigen-decomposition of L to form the matrix L^+, which has a complexity of $O(q^3)$. This is the main computational cost for each on-line processing via commute distance algorithm. If we could find an efficient way to build L^+, we can accelerate the computation.

The normalized graph Laplacian [17] is (I is an identity matrix) :

$$\begin{aligned} \tilde{L} &= D^{-1/2} L D^{-1/2} \\ &= I - D^{-1/2} W D^{-1/2} \\ &= I - \tilde{W} \end{aligned} \tag{9}$$

which can be used to replace the original graph Laplacian L in Eq.(7). Thus we get a **Normalized Commute Distance**. Let $\tilde{\lambda}_i$ be a eigenvalue of the matrix $\tilde{L}$ $(0 = \tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \ldots \leq \tilde{\lambda}_q)$ and vector $\mathbf{v}_i$ be the associated eigenvector of $\tilde{\lambda}_i$, then we have the following derivations:

$$\begin{aligned} & \tilde{L}V = V\Lambda \\ \Rightarrow \quad & (I - \tilde{W})V = V\Lambda \\ \Rightarrow \quad & \tilde{W}V = V(I - \Lambda), \end{aligned} \tag{10}$$

where $\Lambda = diag(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_q)$ and $V = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q)$. The equations mean that, the normalized matrix $\tilde{W}$ and $\tilde{L}$ **share** the same eigenvectors and complementary eigenvalues, *i.e.*, the largest eigenvalue of $\tilde{W}$ is the smallest one of $\tilde{L}$. This inspire us to utilize $\tilde{W}$'s eigen-decomposition to construct $\tilde{L}^+$, rather than $\tilde{L}$'s eigen-decomposition.

We propose a novel anchor-based commute distance (ACD) algorithm whose main idea is constructing $\tilde{W}$ with the anchor-based representation $Z \in \mathbb{R}^{d \times q}$ $(d \ll q)$:

$$\tilde{W} = D^{-1/2} Z^T Z D^{-1/2} = H^T H, \tag{11}$$

based on which the eigen-decomposition can be solved in a cost of $O(d^3) + O(qd^2)$, much smaller than the cost of $O(q^3)$. A number of recent works have studied the anchor-based technique (the anchor points are some representative data points), especially for some semi-supervised and graph-based algorithms [5, 11, 12, 20, 22, 24, 25], *e.g.*, in [24], the authors proposed that each data point on manifold can be approximated by a linear combination of its nearby anchor points. Liu et al. [12] designed the adjacency matrix in a probabilistic measure and used it for scalable semi-supervised learning. These works inspire us much.

4.4.1 Graph Construction and Anchor Selection

Now we introduce how to build an anchor graph to model the data. Suppose we have a node set $\mathcal{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_{q_f}\}$ representing the faces (including request faces and candidate faces), a set $\mathcal{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_{q_n}\}$ representing the names, and a set $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_d\}$ denoting the anchor points selected from the face set. Easy to see that $q_f + q_n = q$. We aim to find a weight matrix $Z \in \mathbb{R}^{d\times q}$ that measures the relationships between each node in the graph and each anchor point. Without loss of generality, let

$$Z = [\ Z_f \quad Z_n\ ] \tag{12}$$

where $Z_f \in \mathbb{R}^{d\times q_f}$ is anchor-face weight matrix and $Z_n \in \mathbb{R}^{d\times q_n}$ is anchor-name weight matrix. For anchor-face weights, we use the same measurement in Eq.(6). That is

$$Z_f(\mathbf{a}_i, \mathbf{f}_k) = \exp\left(-d^2(\mathbf{a}_i, \mathbf{f}_k)/\sigma\right), \tag{13}$$

and for anchor-name weights, we simply assign a $\mathbf{1}$ for each edge connecting an anchor and its correlated name node.

How can we get the anchors? Clustering is a considerable choice. For example, we can use k-means algorithm and then select the points nearest to the centers as anchors. However, k-means is expensive. Random selection is another choice which is extremely efficient, however the quality of anchors might not be good. Taking the advantage our off-line index structure, we propose to use another efficient way: selecting top-d faces from candidate faces for the names appearing in the caption as anchors. The selection is made one-by-one, *i.e.*, we firstly select the top face for one name from its index list and then from the next name's list. The selection won't stop until d non-repeated anchors are collected. If the number of candidate faces is less than d, all of them are selected as anchors.

4.4.2 Adjacency Matrix Construction

The weight matrix $Z \in R^{d\times n}$ can be seen as a d-dimensional anchor-based representation of the data set [5, 12, 22], d is the number of anchor points. That is to say, data points (or nodes) can be represented in the new feature space, no matter what the original features are. With the inner product as the metric to measure the adjacent weights between nodes, we construct the adjacency matrix by

$$\begin{aligned} W &= Z^T Z \\ &= \begin{bmatrix} Z_f^T \\ Z_n^T \end{bmatrix} [Z_f \quad Z_n] \\ &= \begin{bmatrix} Z_f^T Z_f & Z_f^T Z_n \\ Z_n^T Z_f & Z_n^T Z_n \end{bmatrix} \end{aligned} \tag{14}$$

which means that if two data points are correlative (W_{ij} is large), they tend to have similar anchor representations. By sharing the same or similar anchors, data points have similar semantic concepts in a high probability. Thus, this formula is helpful to explore the semantic relationships of the data points on a graph. Moreover, it naturally preserves the nonnegativeness of W, which guarantees the positive semi-definite property of the graph Laplacian.

Note that the bottom right part of W, $Z_n^T Z_n$, should be a diagonal matrix in our consideration, which makes the random walk can't jump from one name node to another name node in one step. That is, one anchor point should not connect to more than one name in the anchor graph.

4.4.3 Efficient Computation

The key point of ACD is efficiently constructing the matrix $\tilde{L}^+$ to speed up commute distance computation. We now offer the derivation with the help of anchor-based representation. For improving the readability, some matrices below are marked with their size. Without loss of generality, we set the number of anchor points be much smaller than the graph size, *i.e.*, $d \ll q$. Let

$$H^T = D^{-1/2} Z^T, \tag{15}$$

and H's SVD decomposition be

$$H_{d\times q} = U_{d\times d} S_{d\times d} V_{q\times d}^T, \tag{16}$$

where S is the singular value matrix with $S = diag(\sigma_1, \ldots, \sigma_d)$, $U_{d\times d} = (u_1, u_2, \ldots, u_d)$ and $V_{q\times d}^T = (v_1, v_2, \ldots, v_d)^T$, then

$$\begin{aligned} \tilde{W}_{q\times q} &= D^{-1/2} Z^T Z D^{-1/2} = H_{d\times q}^T H_{d\times q} \\ &= V_{q\times q} \Sigma_{q\times q} V_{q\times q}^T. \end{aligned} \tag{17}$$

Since $\text{rank}(\tilde{W}) \le \min(d, q) = d$, the matrix $V_{q\times q}$ and $\Sigma_{q\times q}$ can be represented as

$$V_{q\times q} = [V_{q\times d}\ V_{q\times c}] \textbf{ and } \Sigma_{q\times q} = \begin{bmatrix} \Sigma_{d\times d} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{c\times c} \end{bmatrix}, \tag{18}$$

where $c = q - d$ and $\Sigma_{d\times d} = S_{d\times d}^2$. We know that $V_{q\times q}$ is orthogonal and has unit length for each column, so $V_{q\times q} V_{q\times q}^T = I_{q\times q}$ and $V_{q\times q}^{-1} = V_{q\times q}^T$. Then, we get the following important derivations:

$$\begin{aligned} \tilde{L}^+ &= (I - \tilde{W})^+ = V_{q\times q}(I_{q\times q} - \Sigma_{q\times q})^+ V_{q\times q}^T \\ &= [V_{q\times d}\ V_{q\times c}] \begin{bmatrix} (I_{d\times d} - \Sigma_{d\times d})^+ & \mathbf{0} \\ \mathbf{0} & I_{c\times c} \end{bmatrix} \begin{bmatrix} V_{q\times d}^T \\ V_{q\times c}^T \end{bmatrix} \\ &= V_{q\times d}(I_{d\times d} - \Sigma_{d\times d})^+ V_{q\times d}^T + V_{q\times c} V_{q\times c}^T \\ &= V_{q\times d}(I_{d\times d} - \Sigma_{d\times d})^+ V_{q\times d}^T + I_{q\times q} - V_{q\times d} V_{q\times d}^T \\ &= V_{q\times d}[(I_{d\times d} - \Sigma_{d\times d})^+ - I_{d\times d}] V_{q\times d}^T + I_{q\times q} \\ &= V_{q\times d} B_{d\times d} V_{q\times d}^T + I_{q\times q} \\ &= \tilde{V}_{q\times d} \tilde{V}_{q\times d}^T + I_{q\times q}. \end{aligned} \tag{19}$$

Note that $B_{d\times d} = [(I_{d\times d} - \Sigma_{d\times d})^+ - I_{d\times d}]$ is a diagonal matrix and $\tilde{V}_{q\times d} = V_{q\times d} B_{d\times d}^{1/2}$. The derivation implies that

the critical problem now is how to compute the matrix $V_{q\times d}$ and $\Sigma_{d\times d}$ efficiently. From Eq.(16), it is not hard to get the following equations:

$$H_{d\times q}H_{d\times q}^T = U_{d\times d}\Sigma_{d\times d}U_{d\times d}^T, \tag{20}$$

and

$$V_{q\times d} = H_{d\times q}^T U_{d\times d}S_{d\times d}^{-1}. \tag{21}$$

Thus we first compute the eigen-decomposition of HH^T, which is in the size of $d \times d$ ($d \ll q$), to get the matrix $U_{d\times d}$ and $\Sigma_{d\times d}$, and then calculate the matrix matrix $V_{q\times d}$ via Eq.(21) (note that $S_{d\times d} = \Sigma_{d\times d}^{1/2}$). Finally, $\tilde{L}^+$ can be constructed by Eq.(19).

What's more, to compute the commute distances between request faces and names, we don't need to really construct the entire matrix $\tilde{L}^+$ by the last line of Eq.(19), with a complexity of $O(dq^2)$. From Eq.(7) and Eq.(19), we have

$$\begin{aligned} c(i,j) &= V_G(\mathbf{e}_i - \mathbf{e}_j)^T\tilde{L}^+(\mathbf{e}_i - \mathbf{e}_j) \\ &= V_G(\sum_{l=1}^{d}(\tilde{v}_{li} - \tilde{v}_{lj})^2 + 2). \end{aligned} \tag{22}$$

where $\tilde{\mathbf{v}}_l$ is the l-th column of $\tilde{V}_{q\times d}$. Thus each commute distance $c(i,j)$ can be calculated in a short time $O(d)$, if we have got the matrix $\tilde{V}_{q\times d}$. As the constant parts of Eq.(22) don't affect the commute distance when it is used as a distance metric, we just simplify the formulation as

$$c(i,j) \doteq \sum_{l=1}^{d}(\tilde{v}_{li} - \tilde{v}_{lj})^2. \tag{23}$$

This is an interesting result. It says, our anchor-based commute distance essentially finds a data representation of the data set, denoted by $\tilde{V}_{q\times d}$, and the distance is just the Euclidean distance on the d-dimensional feature space.

When all the commute distances $c(i,j)$ have been calculated, we use the same association algorithm described in **Algorithm 1** to match face-name or face-null for each image-caption item. This is the end of our proposed algorithm anchor-based commute distance.

4.5 Overview of FACD

Here we make a brief summary of our framework face-name association via commute distance (FACD). We do the following steps:

Off-line Stage:

1a. Establish the name-item inverted index to get the name-face index structure. (Section 4.2)

1b. Re-rank the faces in each name-face list and filter out the buddy faces. (Section 4.2)

On-line Stage:

2a. Build a connected graph for CD algorithm (Section 4.3.1), or an anchor graph for ACD algorithm (Section 4.4.1-4.4.2).

2b. Calculate the distances between request faces and names on the graph by CD (Section 4.3.1), or ACD (Section 4.4.3).

2c. Make the face-name or face-null assignments for each face via Algorithm 1.

4.6 Complexity Analysis

We offer the complexity analysis of our proposed algorithm CD and ACD used in the on-line stage. Specifically, both CD and ACD have two main steps.

CD: (a) building a connected graph (computing W); (b) computing the matrix L^+ via eigen-decomposition of L.

(a) To build the connected graph, we compute visual similarities of all the faces via Eq.(6) as the weights of face-face edges, and assign 0 or 1 for face-name edges. So the construction has a total computational complexity $O(q^2)$, where q is the graph size.

(b) Solving the eigen-decomposition of L requires the complexity of $O(q^3)$.

So the total computational complexity is $O(q^3)$ for CD.

ACD: (a) building an anchor graph (computing Z); (b) solving the eigen-decomposition of $\tilde{W}$ to compute the commute distance.

(a) To build the anchor graph, we compute the similarities of each node to all the d anchors. So the construction has a total computational complexity $O(qd)$, where d is the number of anchors and q is the graph size. Thus, the number of anchors determines the efficiency of the anchor graph construction. In our experiments, $d \ll q$, making the construction very fast.

(b) Solving the eigen-decomposition of $\tilde{W}$ requires to compute the SVD of $H_{d\times q}H_{d\times q}^T$ and form the matrix $V_{q\times d}$, with the complexity of $O(qd^2) + O(d^3)$ and $O(qd^2)$ respectively.

Considering that $d \ll q$, the total complexity of ACD is $O(d^3)+O(qd^2)$, which is much smaller than $O(q^3)$, the complexity of CD. In our experiments, d is usually very small (*e.g.*, $d = 25$), so the algorithm is extremely fast for each on-line processing. In other words, ACD is more appropriate for a real-time system.

5. EXPERIMENTS

In this section, we presents the detailed experimental results and comparisons to evaluate the effectiveness and efficiency of our proposed approach on a large scale and real world image-caption database.

5.1 Database Statistics

A large scale image-caption database *Face And Names Large scale database* (FAN-Large) was published[3]. To our best knowledge, it is the largest and most realistic database supporting the research of face-name association. The data set contains 125,479 image-caption pairs (Items) with a total of 194,046 detected faces and 244,725 names. A nice property of this database is that, all the detected faces are manually assigned a ground truth name or null, which can be used to result evaluation. Detailed information is listed in Table 1.

Besides the entire database, we also conduct our experiments on its four subsets: *Easy*, *Hard*, *Life* and *Buddies*. We follow the same data partitions and experiments setup used in [14]: the face features are extracted using the tool

[3]`http://www.vision.ee.ethz.ch/~calvin/fan-large`

Table 1: Basic statistics of the FAN-Large database.

	FAN-Large
# of items	125,479
# of detected faces	194,046
# of detected names	244,725
# of unique names	34,645
# of names appear at least 20 times	1,437

Table 2: Statistics of the whole database and four interesting subsets. The *Hard* subset has the maximum number of average faces and names.

Subset	images	faces/image	names/caption
Easy	39,987	1.55	1.95
Hard	25,607	**1.82**	**4.19**
Life	17,459	1.38	1.78
Buddies	13,651	1.68	3.12
All	125,479	1.55	1.95

Torch3vision[4] and the dimensionality is reduced to 100 using the principal component analysis technique. The visual similarity of any two faces is computed via Eq.(6). Table 2 reports the number of images, average number of detected faces per image and the average number of detected names per caption for each of the subset, as well as the whole database. The four subsets help us to understand more comprehensive about the performance of each algorithm on different data sets. Specifically, in the *Easy* subset each caption has at most 2 names, while in the *Hard* subset each caption has 3 or more names. The *Life* subset has captions written by professional editors and the *Buddies* subset contains items with people frequently appearing together (names that appear at least 50 times together in the whole database).

5.2 Evaluation Metric

We evaluate the face-name association performance via two different methods.

- The first evaluation metric is *Accuracy* [14], which measures the percentage of correct assignments over all detected faces including null assignments (a face is not mentioned by a name);
- The second evaluation metric is *Precision* [14], which is the percentage of correct assignments over all faces assigned to a name (not null).

The null assignment is a special point of the face-name association task distinguishing from the face annotation task. Thus, accuracy is a compositive way to measure both the face-name and face-null assignments, while precision only measures the face-name correctness.

5.3 Comparisons and Experimental Results

We investigate many state-of-the-art unsupervised face-name association algorithms including Constrained GMM (CGMM, [1,8]), Graph-based Clustering (GC, [8]) and Constrained K-means (CKM, [14]). Among these methods, CKM does not conform to the constraint C2 and C3. Moreover, we also conduct the algorithm majority voting described in 4.3 as a simple baseline (BASELINE). Our proposed approach FACD has two versions – association by commute distance and anchor-based commute distance. We use CD and ACD to represent them respectively.

[4] `http://torch3vision.idiap.ch`

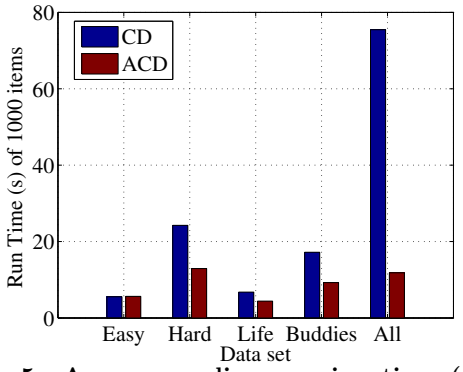


Figure 5: Average on-line running time (seconds) of 1000 items for CD and ACD on each of the four subsets and the whole database.

We evaluate the face-name association for each item of the whole database and its four subsets, and report the average accuracy and precision results in **Table 3** and **Table 4**. Note that, our experimental setup for unsupervised face-name association is exactly the same with that in [14]. From the tables, we find that our algorithms CD and ACD are significantly better than the other comparison methods in some cases. *e.g.*, CD and ACD perform best on the *hard* subset for both accuracy and precision, suggesting that they are particularly well suited for cases in which the captions are noisy (more names than faces). While for the *Buddies* subset, CD and ACD have close performance to the other methods, indicating that they are affected by the high co-occurrence problem. In the buddies item graphs, the buddy people would appear multiple times, making the commute distance less distinguishable for the right associations. In the all, CD and ACD are the best two in the whole database (*all*), which demonstrates the effectiveness of our framework.

Besides the accuracy and precision results, we also record the on-line running time of our method CD and ACD on each of the data set. Note that CD and ACD are the best two methods in performance, so we ignore the running time of the other methods. The time cost is dependent on both the number of items and data complexity (number of faces or names in each item). In Fig.5, the blue/red bar stands for the average running time of 1,000 items for CD/ACD on each data set. We find that, ACD is faster than CD in most cases. Specially, when the data scale is large (*All*), CD requires much more time than ACD. This is because CD's cost is depend on q while ACD is depend on d. The results prove the efficiency of our anchor-based method.

5.4 Parameter Selection

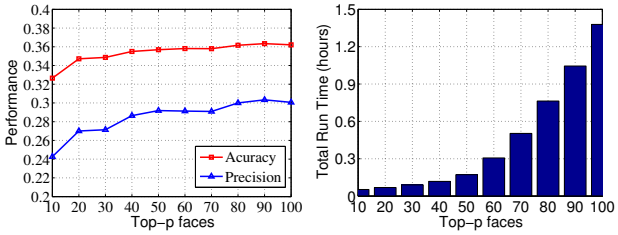
Parameter selection plays a key role to many algorithms. For our methods, there are three main parameters: p and d and ϵ. Parameter p is the number of top faces selected from each name to form the candidate face set in on-line stage. In Fig.6, we record the accuracy results as well as the total running time of CD on *Hard* dataset. When p increases, the total running time grows very fast. So we fix

Table 3: Accuracy (%) comparisons on the four subsets and the entire database. The best result in each row is indicated by the bold font. If the second best is very close to the first, it is also shown in bold.

	Comparison methods			Methods proposed in this paper		
Method:	CGMM	GC	CKM	BASELINE	CD	ACD
Easy	55.0	56.6	54.3	45.8	**58.2**	**58.2**
Hard	31.4	28.9	22.5	22.7	**35.7**	**36.0**
Life	50.9	**51.6**	51.5	41.5	48.8	49.4
Buddies	32.9	**34.9**	33.3	29.1	33.5	**34.7**
All	48.1	47.4	42.0	37.0	49.0	**50.9**

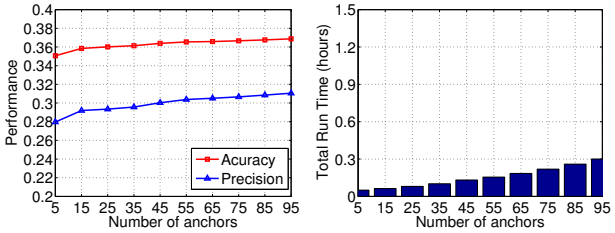
Table 4: Precision (%) comparisons on the four subsets and the entire database. The best result in each row is indicated by the bold font. If the second best is very close to the first, it is also shown in bold.

	Comparison methods			Methods proposed in this paper		
Method:	CGMM	GC	CKM	BASELINE	CD	ACD
Easy	59.2	58.2	56.0	50.1	**62.9**	**63.0**
Hard	25.6	26.2	22.3	21.0	**29.2**	**29.3**
Life	52.6	53.4	53.1	45.9	**58.9**	**59.2**
Buddies	32.8	34.0	33.3	29.4	35.2	**36.0**
All	46.2	44.4	41.4	35.6	**52.4**	47.6



(a) Performance (b) Running Time

Figure 6: Performance and running time vs. parameter p selection for algorithm CD. All the results are evaluated on the *Hard* subset (25,607 itmes).

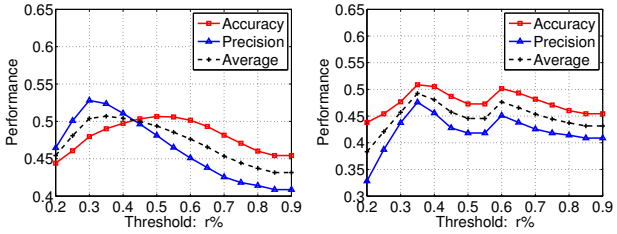


(a) Performance (b) Running Time

Figure 7: Performance and running time vs. number of anchors for algorithm ACD. All the results are evaluated on the *Hard* subset (25,607 itmes).

p to 50 (relative high accuracy and low time cost) for both CD and ACD. Parameter d is the number of anchors used in an anchor graph. In Fig.6, we record the accuracy results as well as the total running time of ACD on *Hard* dataset. When d increases, the accuracy and running time grows very slowly. So we fix d to 25 for ACD.

Threshold ϵ is selected beforehand. In our experiments, we set it to the value at $r\%$ position of all the face-name commute distances (ascending). How to select the parameter r is a problem. We carefully tune it and choose a balance point between good accuracy and precision. Fig.8 is an example: We draw the performance of CD and ACD on the whole set when r varies. The average of accuracy and precision is denoted by a black broken line. Thus, we set $r\% = 0.35$ in this case. Finally, we use 0.4 for *Hard* set, 0.35 for *All* set, and 0.6 for the other three sets. In addition, we set the parameter σ in Eq.(6) to 5.



(a) CD (b) ACD

Figure 8: Performance vs. threshold r % on the entire database (*All*). The black broken line is the average of Accuracy and Precision.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a face-name association via commute distance (FACD) framework as a comprehensive framework capable of dealing with the unsupervised face-name association task. FACD has one off-line index stage and one on-line match stage. In the off-line stage, a name-face index structure is established for efficient face retrieval. In the on-line stage, each request image-caption item is processed independently. Specifically, we build a unified graph for each item and measure the face-name relationship via commute distance on the graph. In this way, both the face-name and face-null assignments can be solved in the same framework. Moreover, we propose a novel anchor-based commute distance algorithm to accelerate the distance computation. Experiment results have demostrate the effectiveness and efficiency of our proposed approach.

In our framework, we adopt a simple strategy to determine the threshold ϵ - we use a global threshold. Can we adopt some other method to select it? Is it possible to find a local threshold for each image-caption item? In the future work, we will study these questions. The "right to left" position information is valuable for face naming. However, the data set is collected from the web and the caption style is very free. Only a few images are described in order. What's more, most of the captions don't have the explicit "left to right" or "right to left" words, so we don't know the right order. We will try to use this information in the future work.

7. ACKNOWLEDGEMENT

This work was supported in part by National Natural Science Foundation of China under Grant 91120302 and 61173186, National Basic Research Program of China (973 Program) under Grant 2012CB316404, Fundamental Research Funds for the Central Universities, Program for New Century Excellent Talents in University (NCET-09-0685), Zhejiang Provincial Natural Science Foundation under Grant No Y1101043 and Foundation of Zhejiang Provincial Educational Department under Grant No Y201018240 and Fundamental Research Funds for the Central Universities.

8. REFERENCES

[1] T. Berg, A. Berg, J. Edwards, and D. Forsyth. Who's in the picture? volume 17, pages 137–144, 2005.

[2] T. Berg, A. Berg, J. Edwards, M. Maire, R. White, Y. Teh, E. Learned-Miller, and D. Forsyth. Names and faces in the news. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–848, 2004.

[3] D. Cai, X. He, and J. Han. Spectral regression: A unified subspace learning framework for content-based image retrieval. In *Proceedings of the 15th ACM International Conference on Multimedia*, pages 403–412, 2007.

[4] D. Cai, X. He, W.-Y. Ma, J.-R. Wen, and H. Zhang. Organizing WWW images based on the analysis of page layout and web link structure. In *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo*, pages 113–116, 2004.

[5] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[6] R. Datta, D. Joshi, J. Li, and J. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.

[7] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[8] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Automatic face naming with caption-based supervision. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[9] X. He, D. Cai, and J. Han. Learning a maximum margin subspace for image retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):189–201, 2008.

[10] N. Khoa and S. Chawla. Robust outlier detection using commute time and eigenspace embedding. *Advances in Knowledge Discovery and Data Mining*, pages 422–434, 2010.

[11] Y. Lin, R. Jin, D. Cai, and X. He. Random projection with filtering for nearly duplicate search. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence*, 2012.

[12] W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 679–686, 2010.

[13] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

[14] M. Özcan, L. Jie23, V. Ferrari, and B. Caputo. A large-scale database of images and captions for automatic face naming. In *Proceedings of the British Machine Vision Conference*, 2011.

[15] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in news videos. *IEEE Transactions on Multimedia*, 6(1):22–35, 1999.

[16] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 563–568, 2008.

[17] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[18] D. Wang, S. Hoi, and Y. He. Mining weakly labeled web facial images for search-based face annotation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, pages 535–544, 2011.

[19] D. Wang, S. Hoi, Y. He, and J. Zhu. Retrieval-based face annotation by weak label regularized local coordinate coding. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 353–362, 2011.

[20] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu. Semi-supervised nonlinear hashing using bootstrap sequential projection learning. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[21] Z. Wu, Q. Ke, J. Sun, and H. Shum. Scalable face image retrieval with identity-based quantization and multi-reference re-ranking. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3469–3476, 2010.

[22] B. Xu, J. Bu, C. Chen, D. Cai, X. He, W. Liu, and J. Luo. Efficient manifold ranking for image retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information*, pages 525–534, 2011.

[23] J. Yang and A. Hauptmann. Naming every individual in news video monologues. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 580–587, 2004.

[24] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. volume 22, pages 2223–2231, 2009.

[25] K. Zhang, J. Kwok, and B. Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240, 2009.

[26] L. Zhang, C. Chen, W. Chen, J. Bu, D. Cai, and X. He. Convex experimental design using manifold structure for image retrieval. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 45–54, 2009.

[27] L. Zhang, L. Chen, M. Li, and H. Zhang. Automated annotation of human faces in family albums. In *Proceedings of the 8th ACM international conference on Multimedia*, pages 355–358, 2003.

[28] L. Zhang, Y. Hu, M. Li, W. Ma, and H. Zhang. Efficient propagation for face annotation in family albums. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 716–723, 2004.

[29] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. volume 16, pages 169–176, 2004.

[30] J. Zhu, S. Hoi, and M. Lyu. Face annotation using transductive kernel fisher discriminant. *IEEE Transactions on Multimedia*, 10(1):86–96, 2008.