A Unified Framework for Context Assisted Face Clustering

Liyan Zhang Dmitri V. Kalashnikov Sharad Mehrotra

Department of Computer Science
University of California, Irvine

ABSTRACT

Automatic face clustering, which aims to group faces referring to the same people together, is a key component for face tagging and image management. Standard face clustering approaches that are based on analyzing facial features can already achieve high-precision results. However, they often suffer from low recall due to the large variation of faces in pose, expression, illumination, occlusion, etc. To improve the clustering recall without reducing the high precision, we leverage the heterogeneous context information to iteratively merge the clusters referring to same entities. We first investigate the appropriate methods to utilize the context information at the cluster level, including using of "common scene", people co-occurrence, human attributes, and clothing. We then propose a unified framework that employs bootstrapping to automatically learn adaptive rules to integrate this heterogeneous contextual information, along with facial features, together. Experimental results on two personal photo collections and one real-world surveillance dataset demonstrate the effectiveness of the proposed approach in improving recall while maintaining very high precision of face clustering.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Clustering*

Keywords

Face Clustering, Context Information, Bootstrapping

1. INTRODUCTION

With the explosion of massive media data, the problem of image organization, management and retrieval has become an important issue [11] [21]. Naturally, the focus in many image collections is people. To better understand and manage the human-centered photos, face tagging that aims to help users associate people names with faces becomes an essential task. The fundamental problem towards face tagging and management is face clustering, which aims to group faces that refer to the same people together.

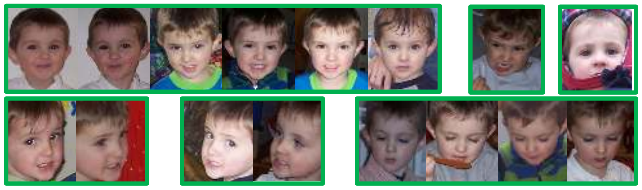


Figure 1: Example of Face Clusters by Picasa

Clustering faces based on facial appearance features is the most conventional approach. It has been extensively studied and significant progress has been achieved in the last two decades [2] [6] [7]. These standard techniques have already been employed in several commercial systems such as Google Picasa, Apple iPhoto, and Microsoft EasyAlbum. These systems usually produce face clusters that have high precision (faces in each cluster refer to the same person), but low recall (faces of a single person fall into different clusters). In addition, a large number of small/singleton face clusters are often returned, which bring heavy burden on the users to label all the faces in the album. Fig. 1 illustrates the example of face clustering result, where faces of a single person fall into six different (pure) clusters, instead of one. One reason for low recall is due to the large variation of faces in pose, expression, illumination, occlusion, etc. That makes it challenging to group faces correctly by using the standard techniques that focus primarily on facial features and largely ignore the context. Another reason is that when systems like Picasa ask for manual feedback from the user, users most often prefer to merge pure (high-precision) clusters rather than manually clean contaminated (low-recall) ones. Consequently, such systems are often tuned to strongly prefer the precision over recall. The goal of our work is to leverage heterogeneous context information to improve the recall of cluster results without reducing the high precision.

Prior research efforts have extensively explored using contextual features to improve the quality of face clustering [16] [17] [19] [20]. In general, in contrast to our work, such techniques often aim at exploring just one (or a few) contextual feature types, with the merging decision often made at the image level only. We, however, develop a unified framework that integrates heterogeneous context information together to improve the performance of face clustering. The framework learns the roles and importance of different feature types from data. It can take into account time decay

This work was supported in part by NSF grants CNS-1118114, CNS-1059436, CNS-1063596. It is part of NSF supported project *Sherlock @ UCI* (`http://sherlock.ics.uci.edu`): a UC Irvine project on Data Quality and Entity Resolution [1].

ICMR'13, April 16–20, 2013, Dallas, Texas, USA.
Copyright 2013 ACM 978-1-4503-2033-7/13/04 ...$10.00.

of features and makes the merging decision at both image and cluster levels. Examples of types of contextual cues that have been used in the past include geo-location and image capture time [21], people co-occurrence [14] [16] [17] , social norm and conventional positioning observed [10], human attributes [13], text or other linked information [4] [18], clothing [9] [20], etc. For instance, [13] proposes to employ human attributes as an additional features. However, the authors do not explore the different roles that each attribute type plays in identifying different people. Social context, such as people co-occurrence, has been investigated in [14] [16] [17]. But these approaches do not deal with cluster-level co-occurrence information. Clothing information has been used extensively in face clustering [9] [20]. However, these techniques do not employ the important time decay factor in leveraging clothing information.

The overall unified framework is illustrated in Figure 2. We start with the initial set of clusters generated by the standard approach for the given photo collection. The initial clusters have high precision but low recall. We iteratively merge the clusters that are likely to refer to the same entities to get higher recall. We use contextual and facial features in two regards: for computing similarities (how similar are two clusters) and for defining constraints (which clusters cannot refer to the same person). The framework then uses bootstrapping to learn the importance of different heterogeneous feature types directly from data. To achieve higher quality, this learning is done adaptively per cluster in a photo collection, because the importance of different features can change from person to person and in different photo collections. For example, clothing is a good distinguishing feature in a photo album where people's clothes are distinct, but a weak feature in a photo collection where people are wearing uniform. We employ the ideas of bootstrapping to partially label any given dataset in automated fashion without any human input. These labels then allow us to learn the importance of various features directly from the given photo collection. Clusters are then merged iteratively, based on the importance of the learned features and computed similarity, to produce a higher quality clustering.

The rest of this paper is organized as follows. We start by formally defining the problem in Section 2. In Section 3, we describe how to leverage the context information at the cluster level, including common scene, people co-occurrence, human attributes, and clothing. In Section 4, we propose the unified framework which automatically learns rules to integrate heterogeneous context information together to iteratively merge clusters. The proposed approach is empirically evaluated in Section 5. Finally, we conclude in Section 6 by highlighting key points of our work.

2. PROBLEM DEFINITION

Suppose that a human-centered photo album P_h contains K images $\{I_1, I_2, \ldots, I_K\}$, see Figure 2. Assume that n faces are detected in P_h, with each face denoted as f_i for $i = 1, 2, \ldots, n$, or $f_i^{I_k}$ (that is, f_i is extracted from image I_k). Suppose that by applying the standard algorithm which is based on facial features, we obtain N clusters $\{C_1, C_2, \ldots, C_N\}$, where each cluster is assumed to be pure, but multiple clusters could refer to the same entity. Our goal is to leverage heterogeneous context information to merge clusters such that we still get very high precision clusters but also improve the recall.

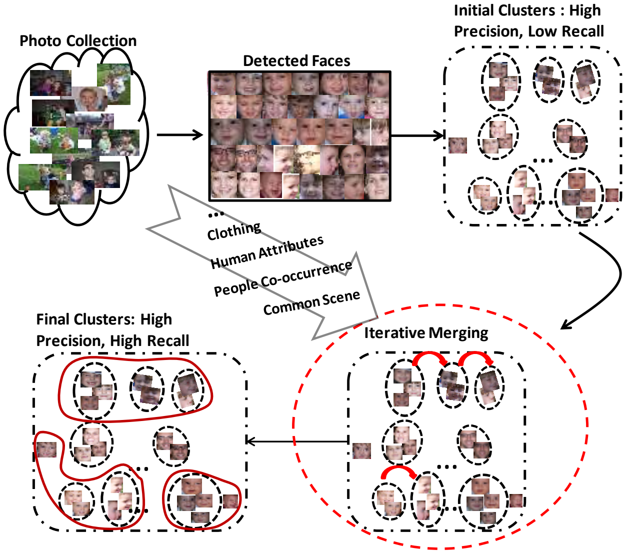


Figure 2: The General Framework

There have been many studies that analyze behaviors of different metrics for measuring quality of clustering. A recent prominent study by Artiles et al. suggests that B-cubed precision, recall and F-measure is one of the best combination of metrics to use according to many criteria [3]. Let $C(f_i)$ be the cluster that f_i is put into by a clustering algorithm. Let $L(f_i)$ be to the real category/label (person) f_i refers to in the ground truth. Given two faces f_i and f_j, the correctness $Correct(f_i, f_j)$ is defined as:

$$Correct(f_i, f_j) = \begin{cases} 1 & \text{if } L(f_i) = L(f_j) \wedge C(f_i) = C(f_j) \\ 0 & \text{otherwise} \end{cases}$$

B-cubed precision of an item f_i is computed as the proportion of correctly related items in its cluster (including itself): $Pre(f_i) = \frac{\sum_{f_j : C(f_i) = C(f_j)} Correct(f_i, f_j)}{\|\{f_j | C(f_i) = C(f_j)\}\|}$. The overall B-cubed precision is the averaged precision of all items: $Pre = \frac{1}{n} \sum_{i=1}^{n} Pre(f_i)$. Similarly, B-cubed recall of f_i is the proportion of correctly related items in its category: $Rec(f_i) = \frac{\sum_{f_j : L(f_i) = L(f_j)} Correct(f_i, f_j)}{\|\{f_j | L(f_i) = L(f_j)\}\|}$. The overall recall is then: $Rec = \frac{1}{n} \displaystyle\sum_{i=1}^{n} Rec(f_i)$. The F-measure is then defined as the harmonic mean of the precision and recall.

3. CONTEXT FEATURE EXTRACTION

Most prior research effort focus on leveraging context features directly at the face level [9] [13] [14]. That is, the similarity is computed between two faces and not two clusters. In this section, we will describe how to utilize context features at the cluster level. Context features are not only able to provide additional contextual *similarity* information to link clusters that co-refer (refer to the same entity), but also generate *constraints* that identify clusters that cannot co-refer (cannot refer to the same entity).

3.1 Context Similarities

3.1.1 Common Scene

It is common for a photographer to take multiple photos of the same "scene" in a relatively short period of time. This phenomenon happens for example when the photographer wants to ensure that at least some of the pictures taken will be of acceptable quality, or when people pose for photos

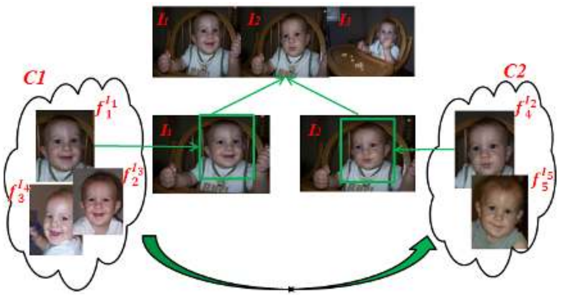


Figure 3: Example of Common Scene

and change their poses somewhat in the sequence of common scene photos. Common scene photos are often taken within small intervals of time from each other and they contain almost the same background and almost the same group of people in each photo. Surprisingly, we are not aware of much existing work that would use common scene detection to improve face-clustering performance. However common scene detection can provide additional evidence to link clusters describing the same entity, since images in a common scene often contain the same people.

To divide images into common scene clusters, some EXIF information (such as image captured time, geo-location, camera model, etc.), and image visual features (color, texture, shape) and image file name can be leveraged. Suppose that in a photo album P_h containing K images $\{I_1, I_2, ..., I_K\}$, the algorithm finds M common scene clusters. Let $CS(I_k)$ denotes the common scene of image I_k. Based on the assumption that two images forming the common scene might describe the same entities, two entities even with dissimilar facial appearances might be linked by the common scene.

For example, as shown in Figure 3, C_1 and C_2 are two initial face clusters based on face appearance. Face $f_1^{I_1}$, extracted from image I_1, belongs to cluster C_1, and face $f_4^{I_2}$ extracted from image I_2 is put into C_2. Since images I_1 and I_2 share the common scene $CS(I_1) = CS(I_2)$, it is possible they describe the same entities. Thus faces $f_1^{I_1}$ and $f_4^{I_2}$ have some possibility to be the same, and the two face clusters C_1 and C_2 are linked to each other via the common scene.

Thus the context similarity $S^{cs}(C_m, C_n)$ of two face clusters C_m and C_n based on common scene is defined as the number of distinct common scenes between the pairs of images from each cluster:

$$\mu_{mn}^{cs} = \{CS(I_k)|CS(I_k) = CS(I_l))\wedge(f_i^{I_k} \in C_m)\wedge(f_j^{I_l} \in C_n)\} \quad (1)$$

$$S^{cs}(C_m, C_n) = \| \mu_{mn}^{cs} \| \quad (2)$$

Thus μ_{mn}^{cs} is the set of common scenes across two face clusters C_m and C_n. $S^{cs}(C_m, C_n)$ is the cardinality of set μ_{mn}^{cs}. The larger value $S^{cs}(C_m, C_n)$ is, the higher the likelihood that C_m and C_n refer to the same entity.

3.1.2 People Co-occurrence

The surrounding faces can provide vital evidence in recognizing the identity of a given face in an image. Suppose that "Rose" and "John" are good friends and often take photos together, then the identity of one person will probably imply the other. In [17], Wu et al. investigated people co-occurrence feature and proposed a social context similarity measurement by counting the common co-occurred single clusters between two clusters. However, this measurement could be greatly improved because single cluster linkage alone is not strong evidence. In this section, we propose a new social context similarity measurement, which use the common cluster-group as evidence to link clusters. Experiments reveal that the linkage of cluster-groups is more reliable than the linkage of single cluster.

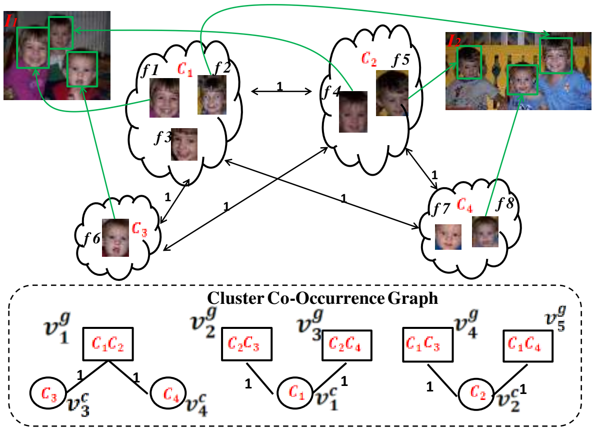


Figure 4: Example of People Co-occurrence

Cluster co-occurrence. First, let us define the co-occurrence relationship between two clusters. We will say that clusters C_m and C_n *co-occur* in/via image I_k, if I_k contains at least two faces such that one is from C_m and the other one is from C_n. In general, the co-occurrence measure $Co(C_m, C_n)$ returns the number of distinct images in which C_m and C_n co-occur:

$$Co(C_m, C_n) = \| \{I_k | \exists f_i^{I_k}, f_j^{I_k} \text{ s.t. } (f_i^{I_k} \in C_m)\wedge(f_j^{I_k} \in C_n)\} \|$$

The co-occurrence relationship between three and more face clusters has a similar definition. Consider the faces in Figure 4 as an example. There, C_1, C_2, C_3, C_4 are four initial face clusters. Since there exists an image I_1 that contain three faces f_1, f_4 and f_6 such that $f_1 \in C_1$, $f_4 \in C_2$, $f_6 \in C_3$, thus $Co(C_1, C_2, C_3) = 1$. Similarly, for the clusters C_1, C_2, C_4 it holds $Co(C_1, C_2, C_4) = 1$. Based on common sense, we know that a person cannot co-occur with himself in an image unless the image is doctored or contains a reflection, e.g., in a mirror. Consequently, clusters connected via a non-zero co-occurrence relationship should refer to different entities. This property will be used later on by the framework to generate context *constraints*.

Co-occurrence graph. The co-occurrence of two face clusters reveals the social relationship between them and between the people they correspond to. We now will describe how to construct cluster co-occurrence graph. Observe that if two face clusters have similar co-occurrence relationships, then the two face clusters might refer to the same entity. This is since people tend to appear with the same group of people in photos, e.g., the same friends. In the example in Figure 4, both C_3 and C_4 co-occur with C_1 and C_2. Such co-occurrence can serve as extra evidence that C_3 and C_4 possibly refer to the same entity. Notice, to demonstrate this graphically, we can represent C_3 and C_4 as nodes in a graph both of which are linked together via a different node that corresponds to C_1 and C_2 as a single cluster-group.

To analyze the various co-occurrences among clusters, we construct the cluster co-occurrence graph $G = (V, E)$. G is a labeled undirectional graph. The set of nodes V in the graph consists of two types of nodes: $V = V^c \cap V^g$. Node $v_i^c \in V^c$ corresponds to each single face cluster C_i. Node $v_j^g \in V^g$ corresponds to each face cluster-group found in an image. The group nodes are constructed as follows. For each image

I_k that contains at least two faces, let Φ^{I_k} denote the set of all the clusters that contain faces present in I_k. We construct $\parallel \Phi^{I_k} \parallel$ cluster-groups, where each group is a set of clusters $\Phi^{I_k} \setminus \{C_j\}$ for each $C_j \in \Phi^{I_k}$. For example, if image I_1 has faces for three clusters $\Phi^{I_1} = \{C_1, C_2, C_3\}$, then the groups are going to be $\{C_1, C_2\}$, $\{C_1, C_3\}$, and $\{C_2, C_3\}$. A node v^g_j is created once per each distinct group. Edge $e_{ij} \in E$ is created between nodes v^c_i and v^g_j only when v^c_i occurs in the context of group v^g_j at least once, that is when exists at least one image I_k such that $v^c_i \cap v^g_j = \Phi^{I_k}$. Edge e_{ij} is labeled with the number of such images, i.e., edge weight $w_{ij} = \parallel \{I_k | v^c_i \cap v^g_j = \Phi^{I_k}\} \parallel$.

Consider Figure 4 as an example. For images I_1 and I_2 we have $\Phi^{I_1} = \{C_1, C_2, C_3\}$, $\Phi^{I_2} = \{C_1, C_2, C_4\}$. Thus we construct four V^c nodes for C_1,C_2,C_3,C_4, and five V^g nodes for $\{C_1, C_2\}$, $\{C_1, C_3\}$, $\{C_2, C_3\}$, $\{C_1, C_4\}$, $\{C_2, C_4\}$. Edges are created accordingly.

From the cluster co-occurrence graph, we observe that if two V^c nodes v^c_m and v^c_n connects to the same V^g node v^g_k, then v^c_m and v^c_n possibly refer to the same entity. For instance, in Figure 4, both C_3 and C_4 connects with $\{C_1, C_2\}$, so C_3 and C_4 are possibly the same. The context similarity from cluster co-occurrence $S^{co}(C_m, C_n)$ for C_m and C_n can be then defined as the flow between these two clusters,

$$S^{co}(C_m, C_n) = \sum_{V^g{}_k \leftrightarrow V^c{}_m, V^g{}_k \leftrightarrow V^c{}_n} \min(w_{mk}, w_{kn}) \qquad (3)$$

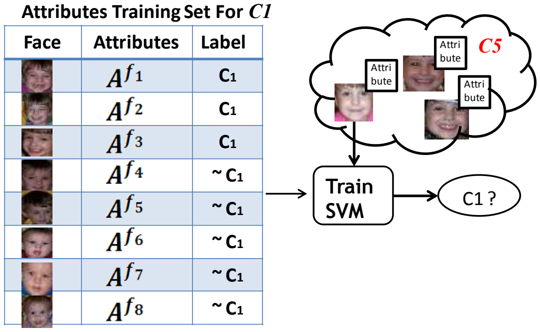
In general, the co-occurrence similarity between two clusters can be measured as the sum of weights of paths which link them through V^g nodes. The larger the number/weight of paths that link C_m and C_n, the higher the likelihood that C_m and C_n refer to the same entity.

3.1.3 Human Attributes

Human attributes, such as gender, age, ethnicity, facial traits, etc., are important evidence to identify a person. By considering attributes, many uncertainties and errors for face clustering can be avoided, such as confusing "men" with "women", "adults" with "children", etc. To get attribute values for a given face, we use the attribute system [13]. It returns values for the 73 types of attributes, such as "black hair", "big nose", or "wearing eyeglasses". Thus, with each face f_i we associate a 73-D attribute vector denoted as A^{f_i}.

In [13], Kumar et al. suggests that attributes can be used to help face verification by choosing some measurement (e.g., cosine similarity) to compute attribute similarities. However, the importance of each type of attribute usually differs when identifying different entities. For example, in a photo album containing just one baby, age is an important factor for identifying this baby; while if several babies exist in an album, then age is not a strongly discriminative feature. Thus, it is essential to determine the importance of attributes for identifying a given entity in the photo collections.

To achieve this, we learn the importance of attributes from the face cluster itself, by leveraging *bootstrapping*. Here, bootstrapping refers to the process of being able to automatically label part of the data, without any human input, and then use these labels to train a classifier. The learned classifier is then used to label the remaining data. One of the main challenges in applying bootstrapping is to be able to provide these partial labels. The general idea of our solution is that faces that belong to one face cluster are very likely to refer to the same entity due to the purity of the initial clusters, hence they can form the positive samples. In turn, faces from two clusters that co-occur in the same image most likely refer to the different people (since a person cannot co-occur with himself in a photo), which can be used to construct the negative samples.



Face	Attributes	Label
	A^{f_1}	C1
	A^{f_2}	C1
	A^{f_3}	C1
	A^{f_4}	~C1
	A^{f_5}	~C1
	A^{f_6}	~C1
	A^{f_7}	~C1
	A^{f_8}	~C1

Figure 5: Example of Human Attributes

Based on the above discussion, the training dataset can be constructed for each cluster. Figure 5 illustrates the attribute training dataset for identifying C_1 from the example in Figure 4. Three faces f_1, f_2, f_3 fall into C_1, so the attributes of these three faces A^{f_1},A^{f_2},A^{f_3} are labeled as C_1. Since the other three clusters C_2, C_3, C_4 have the co-occurrence relationship with C_1, they are considered to describe different entities. Thus the attributes of faces from the other three clusters can be treated as the negative samples. In this way, the attribute training dataset can be constructed automatically for each cluster.

After the attribute training dataset is constructed, a classifier, such as SVM, can be learned for each cluster C_m. Given a 73-D attribute feature A^{f_i} for any face f_i, the task of the classifier is to output whether this face f_i belongs to C_m. In addition to outputting a binary yes/no decision, modern classifiers can also output the probability that f_i belongs to C_m, denoted as $P^A(f_i \in C_m)$. Thus, by applying classifier learned for C_m to each face in an unknown face cluster C_n, we can compute the average probability that C_n belongs to C_m, denoted as $S^A(C_n \rightsquigarrow C_m)$:

$$S^A(C_n \rightsquigarrow C_m) = \frac{1}{\parallel C_n \parallel} \sum_{f_i \in C_n} P^A(f_i \in C_m) \qquad (4)$$

Attribute similarity between C_m and C_n is defined as,

$$S^{attr}(C_m, C_n) = \frac{S^A(C_n \rightsquigarrow C_m) + S^A(C_m \rightsquigarrow C_n)}{2} \qquad (5)$$

That is, the attribute based similarity $S^{attr}(C_m, C_n)$ between two clusters is the average of the average probability of one cluster to belong to the other.

3.1.4 Clothing Information

Clothing information could be a strong feature for determining the identity of a person. However, clothing is a time-sensitive feature since people can change their clothes. Clothing has been considered in the previous work for face clustering, e.g. in [20], but not as a time-sensitive feature described next.

In this section, we introduce time decay factor to control the effect of clothing in identifying people. We propose that the similarity between f_i and f_j should be a function of time:

Notice, in general, there could be different models for assigning weights to paths in addition to the flow model considered in the paper. For example, paths that go through larger group nodes could be assigned higher weight since larger groups of people tend to be better context than smaller ones.

$$S^c(f_i, f_j) = sim(ch_{f_i}, ch_{f_j}) \times e^{-\triangle t/2s^2} \qquad (6)$$

In the above formula, $sim(ch_{f_i}, ch_{f_j})$ refers to the clothing similarity computed only on visual features. Notation $\triangle t$ refers to the capture time difference between 2 faces. By construction, the above time-decay function incorporates the relationship between $\triangle t$ and the effectiveness of clothing features. The smaller $\triangle t$ is, the more effective clothing feature is. With the time difference value $\triangle t$ growing, the effectiveness of clothing feature is decreasing. When the time difference $\triangle t$ is much larger than the time slot threshold s, the clothing feature becomes ineffective.

To compute the clothing similarity, the first step is to detect the location of clothing for the given face, which can be implemented by leveraging the techniques from [9] or simply using a bounding box below detected faces. After that, some low level image features (color, texture) can be extracted to represent the clothing information, and then similarities can be computed.

To obtain the cluster similarity from clothing information, we can compute the clothing similarity between each pair of faces and then choose the maximum value:

$$S^{cloth}(C_m, C_n) = \max_{f_i \in C_m, f_j \in C_n} S^c(f_i, f_j) \qquad (7)$$

Thus the clothing similarity between C_m and C_n is computed by selecting the maximum clothing similarity between each pair of faces respectively falling in the 2 face clusters.

3.2 Context Constraints

In the previous section we have explained how context features can be used as extra positive evidence for computing similarity between clusters. Context features, such as people co-occurrence and human attributes, can also provide *constraints* or negative evidence, which can be used to identify clusters that should refer to different entities.

From cluster co-occurrence relationship, we can derive that two face clusters with $Co(C_m, C_n) > 0$ should refer to definitely different entities, because a person cannot co-occur with himself (in normal cases). Thus we can define that if $Co(C_m, C_n) > 0$, the context dissimilarity from co-occurrence feature is 1, denoted as $D^{co}(C_m, C_n) = 1$.

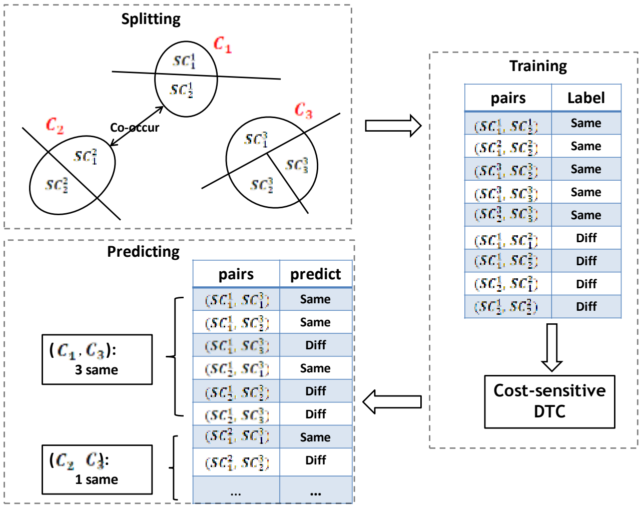
From human attributes, we can derive that two clusters with vastly different attributes values, such as age, gender, ethnicity information should refer to different entities. Thus we can define that if two clusters C_m and C_n have distinct age, gender, ethnicity attribute values, then context dissimilarity from human attributes feature is 1, referred as $D^{attr}(C_m, C_n) = 1$. Then we can define the context dissimilarity measurement between two clusters as follows:

$$D(C_m, C_n) = \begin{cases} 1 & \text{if } D^{co}(C_m, C_n) = 1 \text{ or } D^{attr}(C_m, C_n) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Thus $D(C_m, C_n) = 1$ means C_m and C_n are most likely different, $D(C_m, C_n) = 0$ means that the dissimilarity measure between C_m and C_n cannot tell if they are different or not. The context constraints will be leveraged to implement the bootstrapping ideas explained in the following section.

4. THE UNIFIED FRAMEWORK

In the previous section we have discussed how to leverage the context information from two aspects: computing context similarities (S^{cs}, S^{co}, S^{attr}, S^{cloth}) and context constraints (D^{co}, D^{attr}). In this section, we will develop an approach for integrating these heterogeneous context features together to facilitate face clustering.



pairs	Label
(SC^1_1, SC^1_2)	Same
(SC^2_1, SC^2_2)	Same
(SC^3_1, SC^3_2)	Same
(SC^3_1, SC^3_3)	Same
(SC^3_2, SC^3_3)	Same
(SC^1_1, SC^2_1)	Diff
(SC^1_1, SC^2_2)	Diff
(SC^1_2, SC^2_1)	Diff
(SC^1_2, SC^2_2)	Diff

pairs	predict
(SC^1_1, SC^3_1)	Same
(SC^1_1, SC^3_2)	Same
(SC^1_1, SC^3_3)	Diff
(SC^1_2, SC^3_1)	Same
(SC^1_2, SC^3_2)	Diff
(SC^1_2, SC^3_3)	Diff
(SC^2_1, SC^3_1)	Same
(SC^2_1, SC^3_2)	Diff
...	...

Figure 6: Example of Bootstrapping Process

One possible solution for aggregating these context features is to compute the overall similarity as weighted linear sum of the context similarities. The overall similarity can then be used to merge clusters that do not violate the context constraints. However, this basic solution has several limitations: it is too coarse-grained and it could be difficult to set the weights that would work best for all possible photo collections. Alternatively, the other option is to automatically learn some rules to combine these context features together to make a merging decision. If the rules are satisfied, the two face clusters can be merged. For example, a rule could be if $S^{cs}(C_m, C_n) > 3$ and $S^{co}(C_m, C_n) > 4$, then merge C_m and C_n. The experiments reveal that if the rules are defined appropriately, significantly better merging results can be achieved compared to the basic solution.

Nevertheless, it is hard to define and fix rules that would work well for all possible photo albums. Instead, rules that are automatically tuned to each photo collection would naturally perform better. This is since the importance of each type of context feature usually varies due to the diversity of image datasets. For example, clothing might be important evidence in a photo album where people's clothing is distinct, but it will lose the effect in a photo collection where people wearing uniform. Thus, inspired by [5] [12] [15], we propose a unified framework that can automatically learn and adapt the rules to get high quality of face clustering.

4.1 Construction of Training Dataset

To automatically learn the rules, training dataset is often required. However, since we are trying to automatically learn and tune the rules per each photo collection, it is unlikely that training data will be available, as it will not accompany each given collection. Nevertheless, such rules could be learned by leveraging bootstrapping and semi-supervised learning techniques. To apply those techniques, we need to automatically partially label the dataset. The constructed training dataset should contain positive samples (same face cluster pairs) and negative samples (different face cluster pairs). The key challenge is to be able to automatically, without any human input, label the positive and negative samples for part of the data.

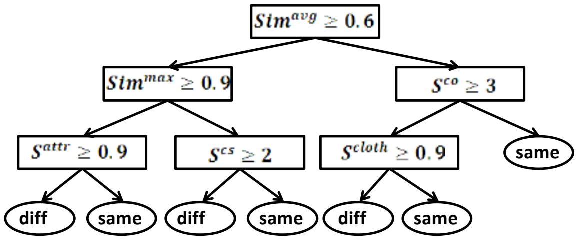


Figure 7: Example of Decision Tree Classifier

In the above section, we discuss that the context information can provide constraints to distinguish clusters referring to different entities. For example, two face clusters with co-occurrence relationship, or distinct attribute values (age, gender, ethnicity), are most likely different. Based on this observation, the negative samples can be constructed.

Then the next issue becomes how to obtain the positive pairs. Due to the purity of initial face clusters, faces that are part of one face cluster refer to the same entity. If we split an initial face cluster into smaller clusters, then these split smaller clusters should refer to the same entity. Thus the split smaller clusters will form the positive sample pairs.

4.1.1 Strategy for Splitting Clusters

Many splitting strategies can be adopted for splitting existing pure clusters into subclusters. For example, one equipart strategy is to split each initial face cluster into two (or other fixed number of) roughly equally-sized subclusters. An alternative equi-size strategy is to predefine the subcluster size (e.g., $sz = 10$ faces) and then split each cluster into subclusters of that size. The equi-size strategy has demonstrated a consistent advantage over other tested options since some of the context features similarities depend on cluster sizes. For example, the context similarity between two large clusters is usually stronger than the similarity between two small clusters. Thus, by considering split clusters of roughly the same size, the effect of cluster size is reduced.

Consider N initial pure face clusters $C_1, C_2, \dots, C_N$, and the predefined subcluster size is sz. Then each cluster C_m with $\| C_m \| > sz$, can be randomly divided into $\left\lceil \frac{\|C_m\|}{sz} \right\rceil$ subclusters, denoted as $\{SC_1^m, SC_2^m, \dots\}$. Figure 6 illustrates an example of splitting clusters.

4.1.2 Automatic Labeling

After splitting clusters into subclusters, the next task is to automatically label the positive and negative training samples. Due to the purity of the initial face clusters, if two subclusters come from the same initial cluster, they form the positive sample, labeled as the "same" pair. If two subclusters come from two different clusters that have co-occurrence relation or distinct attribute values, then the two subclusters form the negative sample, labeled as "diff" pair. Thus, given two subclusters SC_i^m and SC_j^n, the label $La(SC_i^m, SC_j^n)$ can be generated as follows:

$$La(SC_i^m, SC_j^n) = \begin{cases} same & \text{if } m = n, \\ diff & \text{if } D(C_m, C_n) = 1, \\ unknown & \text{otherwise.} \end{cases}$$

Figure 6 illustrates how to construct the training dataset. As shown in Figure 6, subcluster pairs coming from the same initial cluster are labeled as "same" pairs, e.g., (SC_1^1, SC_2^1), (SC_1^2, SC_2^2), etc. Since C_1 and C_2 have the co-occurrence relationship, each subcluster pair respectively deriving from C_1 and C_2 will compose the "diff" pairs, e.g., (SC_1^1, SC_1^2), (SC_1^1, SC_2^2), etc.

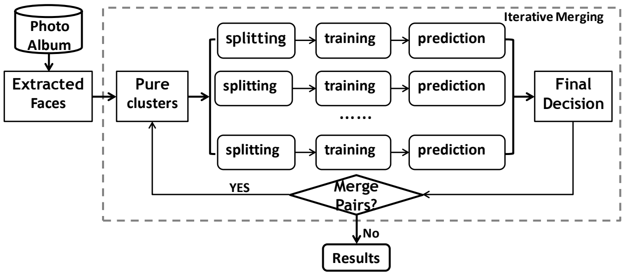


Figure 8: Iterative Merging Framework

4.1.3 Feature Construction

After splitting clusters into subclusters, the algorithm will try to determine which subclusters refer to the same entity. To do that, it first needs to associate a feature vector with each subcluster pair. After that, it will use a classifier to predict whether or not the pair co-refers.

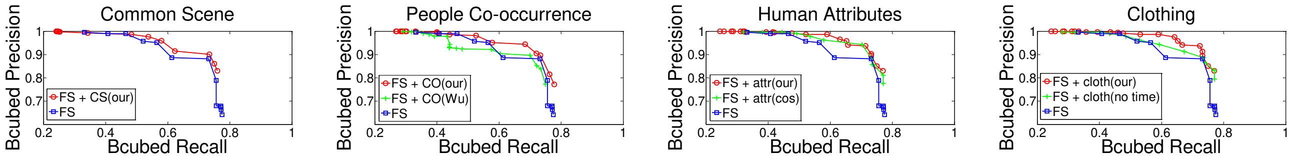
Specifically, for each pair of subclusters SC_i^m and SC_j^n the algorithm associates four features that correspond to the cluster level context similarities S^{cs}, S^{co}, S^{attr}, S^{cloth}, as described in Section 3. In addition, the face appearance similarities between two subclusters are also important, which are measured in three ways: (1) the maximum similarity between face pairs, denoted as Sim^{max}; (2) the minimum similarity between face pairs Sim^{min}; (3) the average similarity of face pairs, referred as Sim^{avg}. Therefore, the algorithm associates 4 types of context features and 3 types of face-based features with each subcluster pairs. Other types of features can also be integrated to this unified framework.

4.2 Classifier Training and Predicting

After the automatic construction of the partially labeled training dataset, the next goal is to learn the merging rules from this training data. Then the learned rules can be applied to predict "same/different" labels for the pairs of subclusters that have been labeled "unknown" before. In this scenario, we choose to use *cost-sensitive* variant of the Decision Tree Classifier (DTC) as the classifier to learn the rules, though other classifiers might also be applied. The reason for using cost-sensitive and not regular DTC is that a single incorrect merge decision can very negatively affect the precision of clusters. That would defeat the purpose of our goal of improving the recall while maintaining the same high precision of the initial clustering. The cost-sensitive version of DTC allows to set the cost of false-positive errors to be much higher than that of false-negative errors. Therefore, we train a very conservative classifier which will try to avoid the false-positive errors thus ensuring high precision of the resulting clusters. To avoid over-fitting problem, we prune the over-fitted branches from the DTC. Figure 7 illustrated an example of the learned DTC.

As shown in Figure 6, the learned DTC can be applied to relabel previously "unknown" pairs by assigning "same" or "diff" labels. For example, in Figure 6, pair (SC_1^1, SC_1^3) is predicted to be "same", and pair (SC_1^1, SC_3^3) to be "diff".

To make the overall merging decision for the face clusters, we need to combine the decisions of the corresponding subclusters. For example, in Figure 6, analyzing the predictions for face cluster pair (C_1, C_3), we discover that 3 subcluster pairs are labeled "same", and 3 pairs are labeled "diff". Similarly, for cluster pair (C_2, C_3), 1 subcluster pair is labeled "same", and 5 subcluster pairs are labeled "diff". Hence the issue is how to make the final merging decision.



(a) (b) (c) (d)

Figure 9: Effectiveness of Extracted Context Features

4.3 Final Merging Decision

Due to the randomness of the splitting strategy, the prediction results might differ with differently split clusters. To reduce the uncertainty introduced by the random splitting strategy, we propose to repeat the "splitting-training-predicting" process multiple times. If two face clusters are predicted to be "same" every time, then the face cluster pair should have higher probability to refer to the same entity.

4.3.1 *Multiple Splitting-Training-Predicting*

Based on the above discussion, the algorithm repeats the "splitting-training-predicting" process multiple times. Each time, the algorithm splits the initial face clusters into subclusters randomly, constructs the training dataset, trains the classifiers, predicts the "unknown" pairs, and then map the subcluster pairs predictions into the merge decisions. Let $T^{same}(C_m, C_n)$ be the number of times that face cluster pair C_m and C_n are predicted to be "same". Similarly, let $T^{diff}(C_m, C_n)$ be the number of times they are predicted to be different. Naturally, the larger T^{same} is, the higher the probability is that this cluster pair refer to the same entity.

4.3.2 *Final Decision*

After perform the "splitting-training-predicting" process t times (e.g., $t = 5$), we can compute T^{same} and T^{diff} values for each pair of clusters, based on which the final merging decision can be made. For example, merge a pair when its $\frac{T^{same}+1}{T^{diff}+1}$ ratio exceed a certain threshold. To avoid early propagation of the incorrect merges, a higher threshold can be selected in the first several iterations, which can be decreased gradually in the subsequent iterations.

4.4 Iterative Merging Strategy

Figure 8 demonstrates the overall iterative merging framework. As shown in Figure 8, after the faces are extracted from the photo album, facial visual features are used to group the faces into initial clusters, which are very pure (high precision, low recall). Then our goal is to merge the pure cluster pairs in order to improve the recall without reducing the high precision. Leveraging multiple context information, and applying bootstrapping ideas, we perform the "splitting-training-predicting" process several times, and then make the combined merging decision. Based on the final decision, some face cluster pairs will be merged and updated, and then the next iteration will be repeated until no merging pairs are obtained. Then the final clustering results are achieved.

5. EXPERIMENTS AND RESULTS

In this section, we evaluate our algorithm on three human-centered data collections: Gallagher, Wedding, and Surveillance. The characteristics of these datasets are listed in Table 1. Gallagher [9] is a public family album containing photos of three children, other family members and their friends. The wedding dataset has been downloaded from Web Picasa. It captures people in a wedding ceremony, including the bride, the groom, their relatives and friends. The surveillance dataset contains images that capture the daily life of faculty and students in the 2nd floor of a computer science building. To evaluate the performance of the proposed approach, we use B-cubed precision and recall defined in Eqs. (1) and (2) as the evaluation metrics.

Dataset	#Images	#Faces	#People	Image Pixels
Gallagher	591	1064	37	2576 × 1716
Wedding	643	1433	31	400 × 267
Surveillance	1030	70	45	704 × 480

Table 1: Experimental Dataset

5.1 Experimental Results

First, we run some experiments to demonstrate the importance of using different context feature types. Then we compare our clustering results with those obtained by Picasa and affinity propagation [8] algorithms, to illustrate the overall effectiveness of our unified framework.

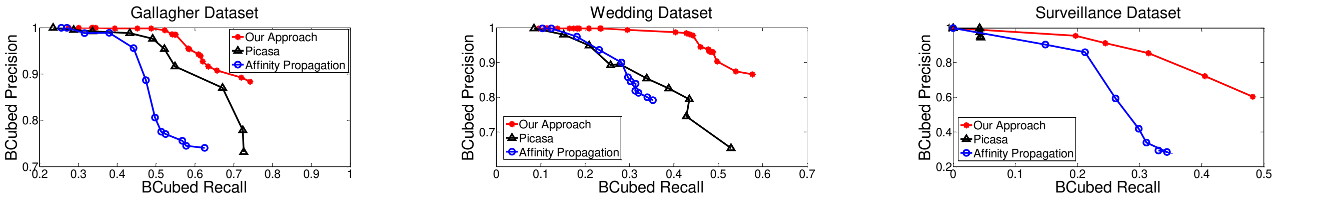
5.1.1 *Context Feature Comparison*

As shown in Figure9, a series of experiments are performed on the Gallagher dataset to test the effectiveness of the proposed 4 types of context similarities. Each plot in Figure 9 corresponds to one type of context similarity. Each plot compares the baseline algorithm that uses only face similarity (denoted as FS) with our framework which is allowed to use just one given context feature type instead of all 4 types.

Figure 9(a) illustrates that the clustering performance can be improved by combining common scene (CS) feature with facial similarities (FS). The improvement is not very significant because only 50 cluster pairs are linked by common sense feature in Gallagher dataset. Figure 9(b) shows the comparison between our approach (CO(our)) and Wu's approach (CO(Wu)) [17] in leveraging people co-occurrence feature. The performance of our approach is much better than Wu's approach because we use the cluster groups as evidence to link two clusters, which is more reliable than the linkage of single cluster. Figure 9(c) demonstrates that our approach (attr(our)) outperforms the cosine similarity measurements (attr(cos)) in using human attributes feature. The advantage of our approach is because we automatically learn the relative importance of various attribute types in identifying different people. Figure 9(d) shows the advantage of our approach (cloth(our)) compared with the approach without considering time factor (cloth(no time)) in utilizing clothing information. This demonstrates the advantage of adding time decay factor to clothing information.

5.1.2 *Clustering Results Comparison*

To evaluate the performance of the proposed unified framework, we compare our clustering results with affinity propagation (AP) [8] and Picasa's face clustering toolkit, as shown in Figure 10. Four types of context information and facial visual similarities are integrated into our framework. B-cubed precision and recall are computed as the evaluation metric-



(a) (b) (c)

Figure 10: Comparison of Clustering Performance with Affinity Propagation and Picasa on Three Datasets.

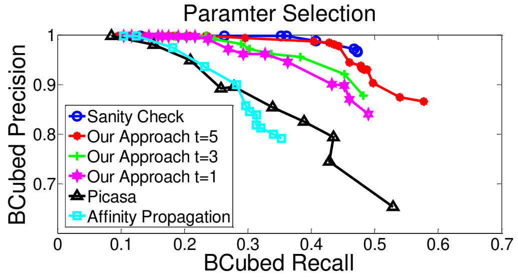


Figure 11: Comparison of Different Parameters

s. In our clustering framework, several parameters need to be selected, the split cluster size (sz), and number of times to perform "splitting-training-predicting" process t. In this experiment we set $sz = 10$ and $t = 5$.

When performing affinity propagation, we combine the 4 types of context features with equal weight, and then aggregate context features and facial feature with equal weight to construct the overall similarity. By adjusting the preference parameter p in AP, we are able to control the precision vs. recall tradeoff for AP. Picasa allows users to specify the cluster threshold (from 50 to 95) to control the precision vs. recall tradeoff. With the increasing of threshold, the recall reduces and the precision increases.

As demonstrated in Figure 10, our unified framework outperforms Affinity Propagation (AP) and Picasa in all the three datasets. The gained advantage is due to leveraging the bootstrapping idea to automatically learn and tune the merging rules per each dataset, and due to using the conservative merging strategy that guarantees the high precision. In addition, our framework is more reliable for data with lower quality images because the context features are less sensitive to image resolution. This is not the case for Picasa, as its performance drops dramatically with the decreasing of image quality. The experiments illustrate that our unified framework reaches high quality and at the same time is more reliable than the other two techniques.

5.1.3 *Effectiveness and Efficiency*

Figure 11 shows the comparison of clustering results when choosing different values for parameter t (the number of times to perform "splitting-training-predicting" process). The larger t can provide more reliable clustering results because it can reduce the uncertainties introduced by random splitting. However, The larger t will reduce the efficiency of the algorithm. The experiments illustrate that when $t = 5$, our performance is approaching the "sanity check" results (merging rules learned from ground truth). And when $t = 1$, our results are still better than affinity propagation and Picasa. Thus our approach is able to achieve a good result without sacrificing efficiency.

6. CONCLUSION

In this paper we have proposed a unified framework for integrating heterogeneous context information (including common scene, people co-occurrence, human attributes and clothing) to improve the quality/recall of face clustering. The context information has been used for both: computing context similarities to link clusters that co-refer as well as for generating context constraints to differentiate clusters that do not co-refer. The proposed unified framework leverages bootstrapping to automatically learn the adaptive rules to integrate heterogeneous context information together to iteratively merge clusters, in order to improve the recall of clustering results. Our experiments on the real-world datasets demonstrated the effectiveness of the extracted context features and of the overall unified framework.

7. REFERENCES

[1] Project sherlock @ uci. http://sherlock.ics.uci.edu.

[2] T. Ahonen, A. Hadid, and et al. Face description with local binary patterns: Application to face recognition. In *IEEE Trans. Pattern Anal*, 2006.

[3] E. Amigo and et al. A comparison of extrinsic clustering evaluation metrics based on formal constraints. In *Technical Report*, 2008.

[4] T. L. Berg, A. C. Berg, and et al. Names and faces in the news. In *IEEE ICPR*, 2004.

[5] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Adaptive graphical approach to entity resolution. In *JCDL*, 2007.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. In *AVBPA*, 1997.

[8] B. J. Frey and D. Dueck. Clustering by passing messages between data points. In *Science*, 2007.

[9] A. Gallagher and T. Chen. Clothing cosegmentation for recognizing people. In *IEEE CVPR*, 2008.

[10] A. Gallagher and T. Chen. Understanding images of groups of people. In *IEEE CVPR*, 2009.

[11] J.Tang, S. Yan, R. Hong, G. Qi, and T. Chua. Inferring semantic concepts from community-contributed images and noisy tags. In *ACM Multimedia*, 2009.

[12] D. V. Kalashnikov, Z. Chen, S. Mehrotra, and R. Nuray-Turan. Web people search via connection analysis. In *TKDE*, 2011.

[13] N. Kumar and et al. Describable visual attributes for face verification and image search. In *IEEE TPAMI*, 2011.

[14] Y. J. Lee and K. Grauman. Face discovery with social context. In *BMVC*, 2011.

[15] R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Exploiting web querying for web people search. In *ACM TODS*, 2012.

[16] K. Shimizu, N. Nitta, and et al. Classification based group photo retrieval with bag of people features. In *ICMR*, 2012.

[17] P. Wu and F. Tang. Improving face clustering using social context. In *ACM Multimedia*, 2010.

[18] J. Yagnik and A. Islam. Learning people annotation from the web via consistency learning. In *MIR*, 2007.

[19] L. Zhang, R. Vaisenberg, S. Mehrotra, and D. V. Kalashnikov. Video entity resolution: Applying er techniques for smart video surveillance. In *PerCom Workshops*, 2011.

[20] W. Zhang and et al. Beyond face: Improving person clustering in consumer photos by exploring contextual information. In *ICME*, 2010.

[21] M. Zhao, Y. Teo, and et al. Automatic person annotation of family photo album. In *CIVR*, 2006.