Noname manuscript No.
(will be inserted by the editor)

Context Assisted Face Clustering Framework with Human-in-the-Loop

Liyan Zhang · Dmitri V. Kalashnikov · Sharad Mehrotra

Received: date / Accepted: date

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. Finally, we discuss a novel methodology for integrating human-in-the-loop feedback mechanisms that leverage human interaction to achieve the high-quality clustering results. 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.

Keywords Face Clustering · Context Information · Bootstrapping · Human-in-the-loop

1 Introduction

With the explosion of massive media data, the problem of image organization, management and retrieval has become an important challenge [13] [19] [20] [28].

Department of Computer Science, University of California, Irvine
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].

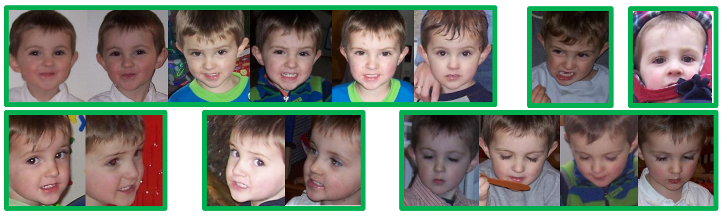


Fig. 1 Example of Face Clusters by Picasa

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.

Clustering faces by utilizing 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] [4] [5] [8] [9]. 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 [18] [21] [24] [25] [27]. 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 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 im-

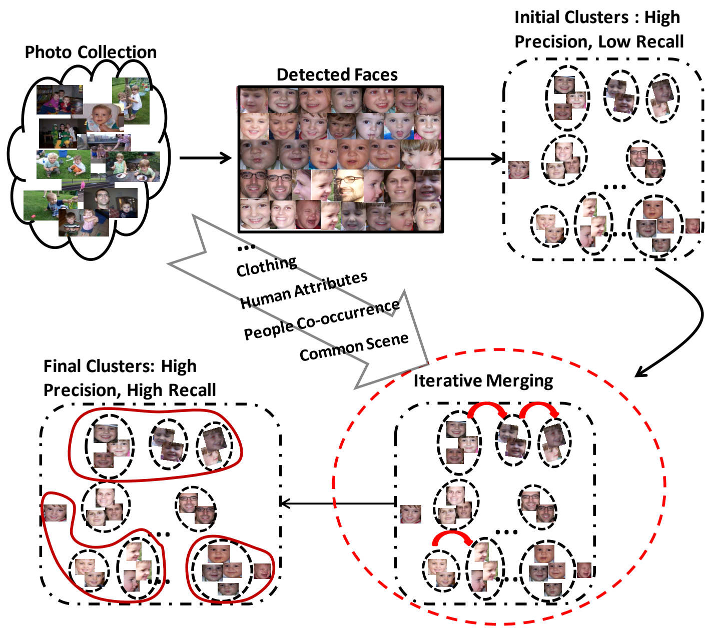


Fig. 2 The General Framework

age capture time [28], people co-occurrence [16] [18] [21] , social norm and conventional positioning observed [12], human attributes [15], text or other linked information [6] [22], clothing [11] [27], etc. For instance, [15] 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 [16] [18] [21]. But these approaches do not deal with cluster-level co-occurrence information. Clothing information has been used extensively in face clustering [11] [27]. However, these techniques do not employ the important time decay factor in leveraging clothing information.

This paper is an extension of our previous work [23], above on which we integrate human-in-the-loop to the unified context assisted framework. The 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. Finally, we discuss the proper method to integrate human-in-the-loop in order to leverage human interaction to achieve the very high-quality clustering results.

The rest of this paper is organized as follows. We start by introducing related work in Section 2, and then formally define the problem in Section 3. In Section 4, we describe how to leverage the context information at the cluster level, including common scene, people co-occurrence, human attributes, and clothing. In Section 5, we propose the unified framework which automatically learns rules to integrate heterogeneous context information together to iteratively merge clusters. In Section 6, we discuss the proper method to integrate human-in-the-loop process. The proposed approach is empirically evaluated in Section 7. Finally, we conclude in Section 8 by highlighting key points of our work.

2 Related Work

In the prior literature, plenty of works have explored the issues of face recognition/clustering. The most conventional approaches are merely based on facial features, which are very sensitive to the variations of image captured conditions. To deal with this problem, researchers shift their research interests to context assisted methods. In the following, we will summarize the prior related works about these two types of approaches.

2.1 Facial Appearance based Approach

As illustrated in Figure 3, facial appearance based methods are the most conventional strategies to handle face clustering/recognition issues. Given a photo collection, after the process of face detection and alignment, different types of low-level facial features can be extracted to represent the detected faces, including Local Binary Pattern (LBP) [2], Histogram of Oriented Gradients (HOG) [8], and Local Phase Quantization (LPQ), etc. Following that, the extracted high-dimensional features will be projected into low-dimensional spaces leveraging dimensionality reduction techniques such as Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Then the low-dimensional discriminant features can be used to compute face similarities, and further construct face groups by performing clustering algorithms such as K-Means, Affinity Propagation [10], etc.

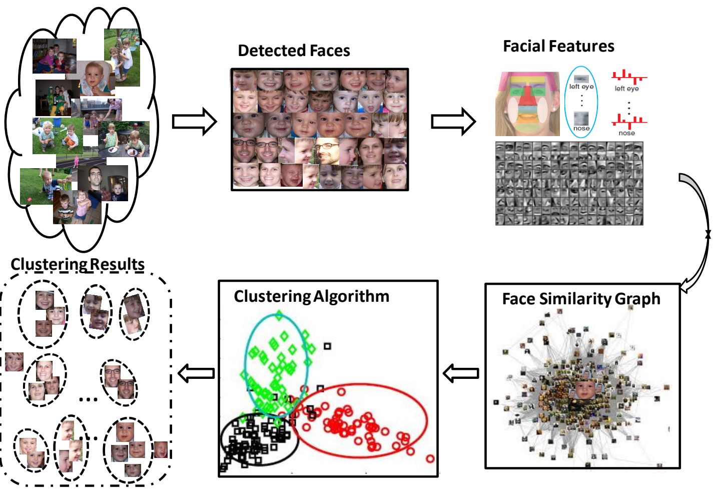


Fig. 3 Conventional Framework for Face Clustering

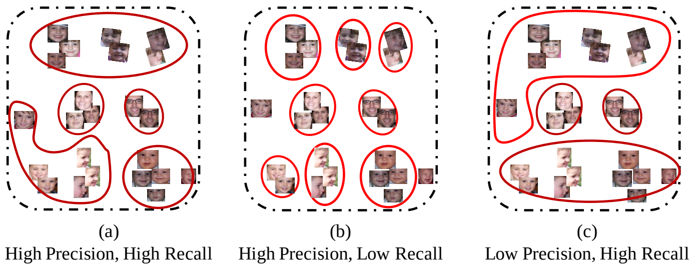

(a)
High Precision, High Recall
(b)
High Precision, Low Recall
(c)
Low Precision, High Recall

Fig. 4 Clustering Results by selecting different thresholds. (a)The ideal clustering results. (b) Tight threshold leads to high precision but low recall results. (c) Loose threshold leads to low precision and high recall results.

These traditional approaches are able to achieve good performance under controlled conditions, but they tend to suffer when dealing with uncontrolled situations where faces are captured with a large of variation in pose, illumination, motion blur, occlusion, expression, scale, etc. These nuisance factors may cause the visual differences between faces captured in distinct conditions referring to the same entity to be greater than those between two different entities under similar conditions. Therefore, the techniques that rely on low level facial features only are not sufficient to handle the issues and achieve the very high-quality clustering performance. Just as demonstrated in Figure 4, the tight threshold will lead to results with high precision but low recall, while the loose threshold will lead to high recall but low precision results. In consequence, researchers shift their attentions to context assisted methods.

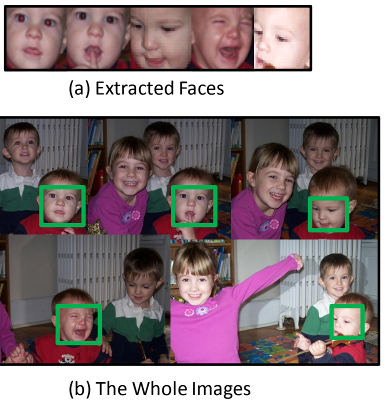


Fig. 5 Example of Face Clusters by Picasa

2.2 Context Assisted Approaches

The utilization of context information could bring significant improvement on top of the techniques that rely on facial features only. As illustrated in Figure 5, it is even challenging for a human to determine whether the five faces (in Figure 5(a)) describe the same entity. However, considering the entire collection of whole images which contain various context information, it becomes obvious that the five faces refer to the same baby.

In the prior literature, many types of context information have been investigated as additional clues to facial features for face recognition/clustering, such as Geo-location and image captured time [28], people co-occurrence [18] [21] [16], social norm and conventional positioning observed [12], human attributes [15], text or other linked information [22] [6], clothing [11] [27], etc. In [15], Kumar et al. proposed that human attributes can be employed as an additional information to improve face verification performance, however, they did not consider the different roles that each attribute type plays in identifying different people. Social context such as people co-occurrence, has been investigated in [18] [21] [16] , but none of these works propose an appropriate way to leverage people co-occurrence information in cluster level. Clothing information has attracted much attention in face clustering [11] [27]. However, these works did not introduce time decay factor in leveraging clothing information.

2.3 Contributions over the Conference Version of the Article

In this article, we study an approach that utilizes heterogeneous context information to address the problem of face clustering for personal photo collections. We first explore the appropriate methods to leverage context features including common scene, people co-occurrence, human attributes, clothing on the cluster level, and then propose a unified framework that employs bootstrapping idea to integrate these context features together. Finally, we discuss a novel methodology for integrating human-in-the-loop feedback mechanisms that leverage human interaction to achieve the high-quality clustering results. Compared with the other related works which only investigate one (or a few) context feature types on the face/image level, our contribution lies in that we propose a unified framework integrating heterogeneous context features and make merging decisions on the cluster level. This article is an extended version of our initial study [23] and the new contributions are as follows.

1. We integrate a new component of human-in-the-loop feedback mechanism to the previous framework in [23]. In the human-in-the-loop process, we choose the appropriate user interface where all the clusters will be ranked when users specify a target person. We propose a novel ranking strategy with the consideration of relevance and impact factors.
2. We have conducted more detailed experimental study, including new experiments on the human-in-the-loop part.

3 Problem Definition

Suppose that a human-centered photo album P_h contains K images denoted as $\{I_1, I_2, \ldots, I_K\}$, 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.

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: $Rec = \frac{1}{n}\sum_{i=1}^{n} Rec(f_i)$. The F-measure is then defined as the harmonic mean of the precision and recall.

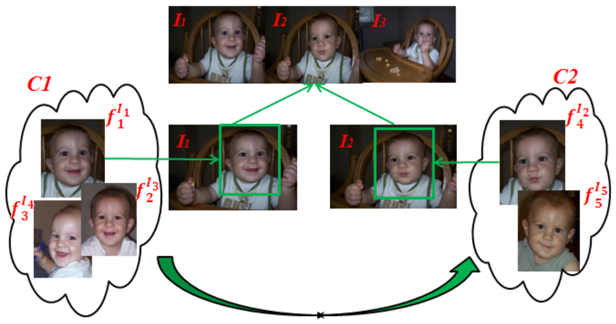


Fig. 6 Example of Common Scene

4 Context Feature Extraction

Most prior research effort focus on leveraging context features directly at the face level [11] [15] [16]. 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).

4.1 Context Similarities

4.1.1 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 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 6, 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:

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

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

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.

4.1.2 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 [21], 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.

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:

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

The co-occurrence relationship between three and more face clusters has a similar definition. Consider the faces in Figure 7 as an example. There, C_1, C_2, C_3, C_4 are four initial face clusters. Since there exists an image I_1

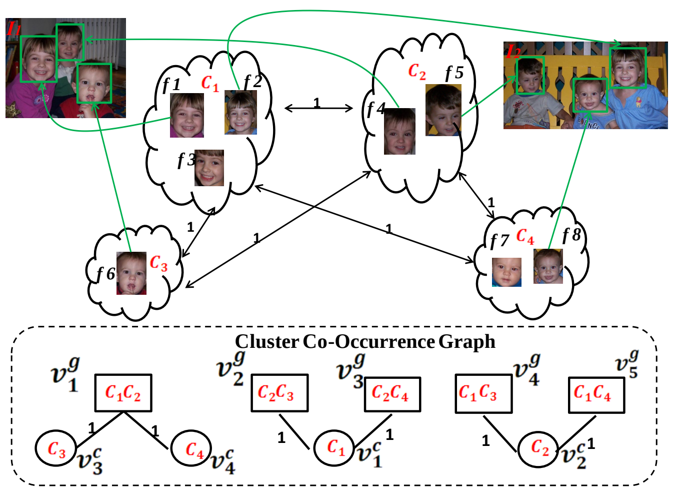


Fig. 7 Example of People Co-occurrence

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 7, 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 $v^c_i \in V^c$ corresponds to each single face cluster C_i. Node $v^g_j \in V^g$ 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 $\| \Phi^{I_k} \|$ 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 $w_{ij} = \| \{I_k | v^c_i \cap v^g_j = \Phi^{I_k}\} \|$.

Consider Figure 7 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 7, 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,

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

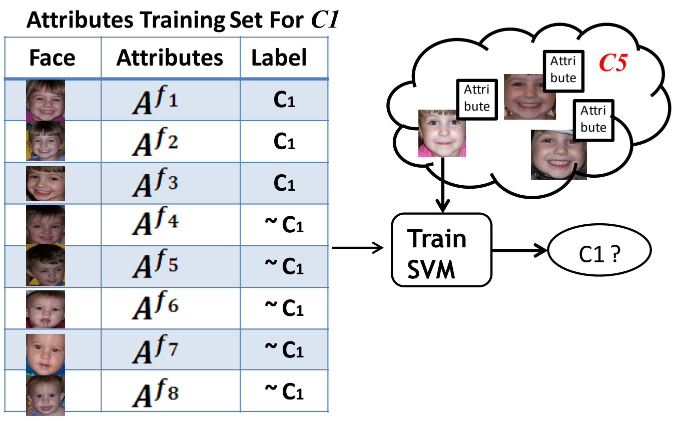
In general, the co-occurrence similarity between two clusters can be measured as the sum of weights of paths which link them through V^g nodes. The larger the number/weight of paths that link C_m and C_n, the higher the likelihood that C_m and C_n refer to the same entity.[1]

4.1.3 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 [15]. 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 [15], 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.

[1] 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.



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

Fig. 8 Example of Human Attributes

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.

Based on the above discussion, the training dataset can be constructed for each cluster. Figure 8 illustrates the attribute training dataset for identifying C_1 from the example in Figure 7. 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 $S^A(C_n \rightsquigarrow C_m)$:

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

Attribute similarity between C_m and C_n is defined as,

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

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.

4.1.4 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 [27], 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:

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

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 [11] 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:

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

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.

4.2 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:

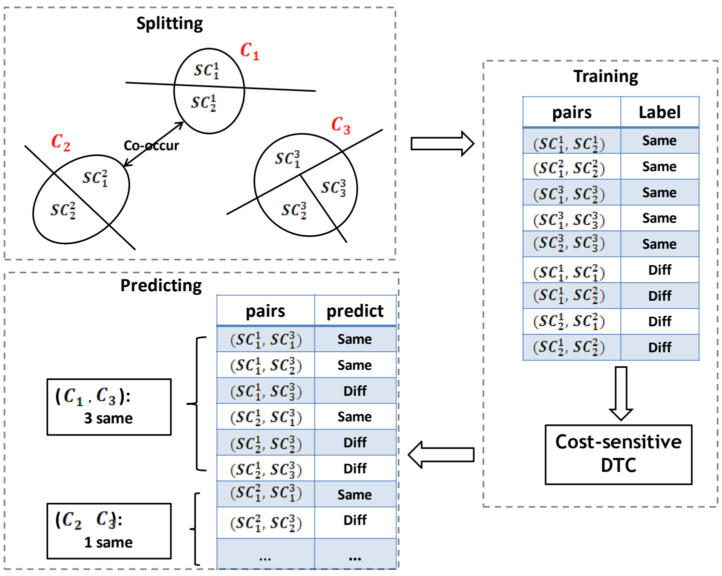
$$D(C_m, C_n) = \begin{cases} 1 \text{ if } D^{co}(C_m, C_n) = 1 \text{ or } D^{attr}(C_m, C_n) = 1 \\ 0 \text{ otherwise} \end{cases}$$

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.

5 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.

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.



Training

pairs	Label
(SC_1^1, SC_2^1)	Same
(SC_1^2, SC_2^2)	Same
(SC_1^3, SC_2^3)	Same
(SC_1^3, SC_3^3)	Same
(SC_2^3, SC_3^3)	Same
(SC_1^1, SC_1^2)	Diff
(SC_1^1, SC_2^2)	Diff
(SC_2^1, SC_1^2)	Diff
(SC_2^1, SC_2^2)	Diff

pairs	predict
(SC_1^1, SC_1^3)	Same
(SC_1^1, SC_2^3)	Same
(SC_1^1, SC_3^3)	Diff
(SC_2^1, SC_1^3)	Same
(SC_2^1, SC_2^3)	Diff
(SC_2^1, SC_3^3)	Diff
(SC_1^2, SC_1^3)	Same
(SC_1^2, SC_2^3)	Diff
...	...

Fig. 9 Example of Bootstrapping Process

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 [7] [14] [17] [26], we propose a unified framework that can automatically learn and adapt the rules to get high quality of face clustering.

5.1 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.

In the above section, we discuss that the context information can provide constraints to distinguish clusters referring to different entities. For example,

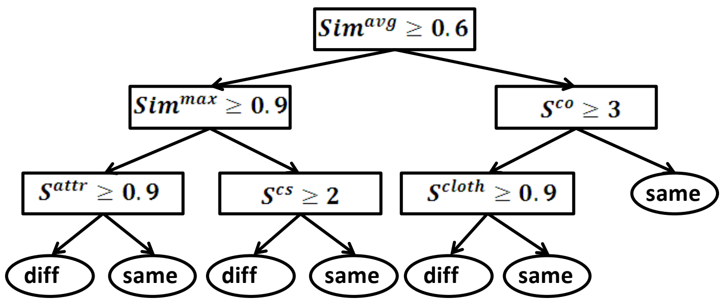


Fig. 10 Example of Decision Tree Classifier

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.

5.1.1 Strategy for Splitting Clusters

Many splitting strategies can be adopted for splitting existing pure clusters into subclusters. For example, one equi-part 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 $\parallel C_m \parallel > sz$, can be randomly divided into $\left\lceil \frac{\|C_m\|}{sz} \right\rceil$ subclusters, denoted as $\{SC_1^m, SC_2^m, \dots\}$. Figure 9 illustrates an example of splitting clusters.

5.1.2 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 9 illustrates how to construct the training dataset. As shown in Figure 9, 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.

5.1.3 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.

5.2 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

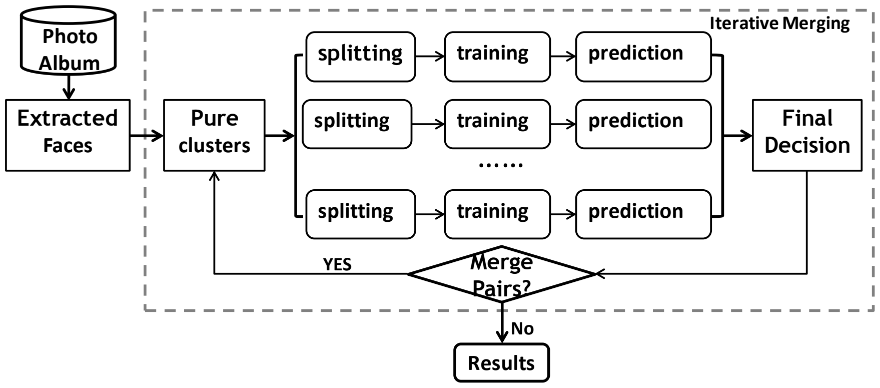


Fig. 11 Iterative Merging Framework

the resulting clusters. To avoid over-fitting problem, we prune the over-fitted branches from the DTC. Figure 10 illustrated an example of the learned DTC.

As shown in Figure 9, the learned DTC can be applied to relabel previously "unknown" pairs by assigning "same" or "diff" labels. For example, in Figure 9, 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 9, 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.

5.3 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.

5.3.1 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.

5.3.2 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.

5.4 Iterative Merging Strategy

Figure 11 demonstrates the overall iterative merging framework. As shown in Figure 11, 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. Before the actual merging operation, we need to guarantee that the merging pairs are not conflict based on constraints. And then the next iteration will be repeated until no merging pairs are obtained. Then the final automatic clustering results are achieved.

6 Improving Results by Leveraging Human-in-the-Loop Techniques

Thus far we have considered the construction of face clusters by only utilizing fully automated techniques that do not require any human involvement. While the proposed automated context-assisted framework reaches very high quality results, naturally it can make mistakes as well. This is especially the case after certain point in the execution: if the algorithm is forced to continue performing merges, then the resulting precision will drop, as the algorithm will start to base its decisions on weaker evidence.

This problem can be mitigated with the help of user feedback, that is, by using human-in-the-loop techniques. In general, human-in-the-loop mechanisms have been actively employed for similar purposes by many commercial applications, such as Google Picasa. The high-level goal of such techniques is to be able to get very high quality face clustering results while minimizing user participation. That is, the task is to provide an appropriate user interface as well as to design the right question-asking strategy, as naive solutions

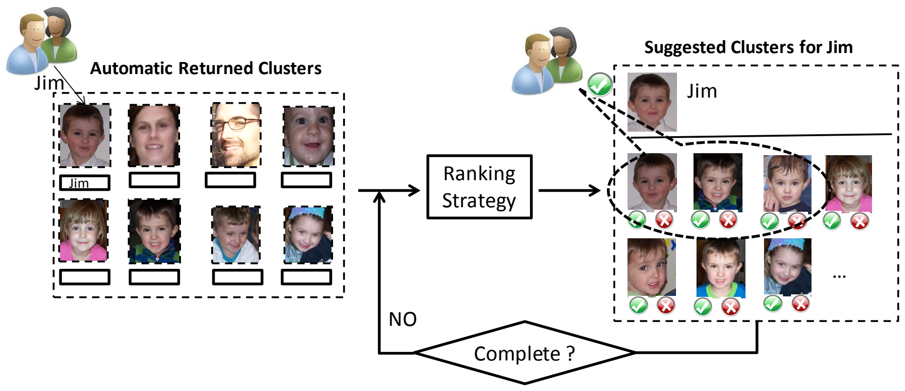


Fig. 12 An Example of Human Computer Interaction Interface

can easily overburden the user with too many unnecessary questions. We next describe the user interface and the algorithm for choosing questions to ask that are utilized in our approach. In Section 7.3 we will demonstrate that the proposed techniques outperform various baselines and reach higher quality results.

6.1 User Feedback Interface

Before asking the user for feedback, the algorithm first applies the fully-automated techniques described in the previous sections to merge clusters. It stops the merging process at the tipping point where the recall is significantly improved, precision is still very high, but the precision is about to start to drop. At that stage, the algorithm stops the fully automated process and starts leveraging the user feedback by asking the user to label the yet-unlabeled clusters manually.

Figure 12 illustrates the interface for the user feedback. It is based on the observation that often users prefer to disambiguate one person at a time instead of disambiguating all people at once. For instance, the user might want to first find and merge all clusters of say herself only, and then of her friends and relatives, and so on.

To accomplish this task, the interface initially shows all clusters to the user. To minimize the clutter, the interface represent each cluster by a single image selected to stand for this entire cluster. The user then chooses and labels one cluster as the “starting” or “target” cluster, signaling the algorithm that she now wants to focus on disambiguating this specific person. In Figure 12 (left part), this step is exemplified by the user choosing one cluster and labeling it as “Jim”. At that stage, the algorithm tries to help the user to disambiguate the chosen person by ranking the yet-unlabeled clusters and then presenting K of them in the ranked order to the user, see Figure 12 (right part). We will explain different ranking strategies later in this section.

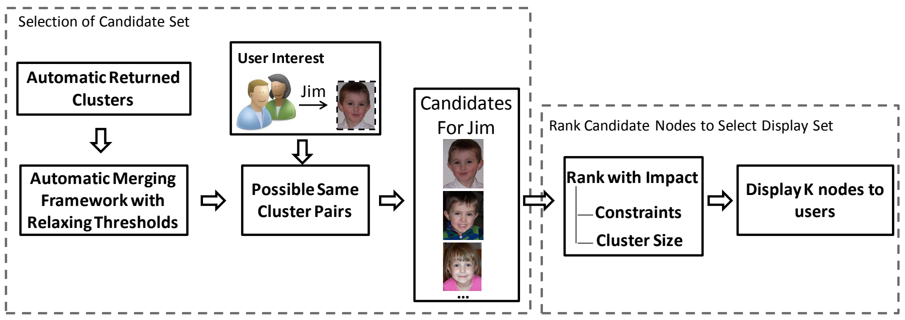

Selection of Candidate Set
Automatic Returned Clusters
User Interest
Jim
Automatic Merging Framework with Relaxing Thresholds
Possible Same Cluster Pairs
Candidates For Jim
...
Rank Candidate Nodes to Select Display Set
Rank with Impact
Constraints
Cluster Size
Display K nodes to users

Fig. 13 2-Step Framework for Choosing K User Questions.

The user then has the opportunity to provide feedback to the algorithm by clicking *yes* or *no* on all or some of the K clusters, indicating whether these clusters represent the *same* or *different* person as the starting cluster. After that, the algorithm factors in the newly provided user labeling, reranks the remaining yet-unlabeled clusters, and the process continues until the disambiguation is complete or when the user wishes to move on to the next person to disambiguate. We will next describe the ranking algorithm used in our solution. The algorithm leverages the fully automated approach itself to generate candidates for labeling and is capable of incorporating newly added user constraints to effectively filter out unnecessary questions.

6.2 Ranking Algorithm

The task of the ranking algorithm is to choose a set of K yet-unlabeled clusters to show for the user for the yes/no feedback. Intuitively, to generate this set the ranking algorithm will need to balance two criteria. First, the chosen clusters should represent the same person to the one being currently disambiguated. Second, the chosen clusters should have the most impact on reducing the overall uncertainty in the system and on increasing the quality (e.g. recall) of disambiguating the chosen person.

Figure 13 illustrates the proposed ranking algorithm. It consists of two steps: (1) the selection of candidate set and (2) the ranking of candidate nodes selected on Step 1. Based on the results of these two steps, K clusters are chosen to display to the user on the next iteration. This 2-step process is repeated iteratively after each time the user provides the feedback. In the first step, the algorithm chooses a pool of (more than K) potential candidate clusters. The choice is made based on the likelihood for these candidate clusters to be the same as the person being disambiguated. In the second step, the algorithm examines the chosen pool of clusters to pick K of them, such that asking feedback for them would result in the most overall impact. We next describe these two steps in detail.

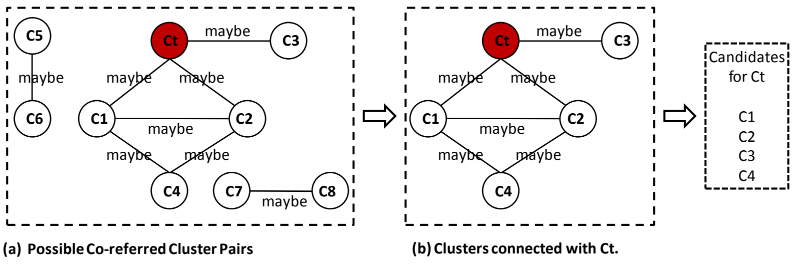


Fig. 14 An Example of Candidate Set Construction

6.2.1 Step 1: Choosing the Candidate Set of Clusters

Recall that the overall goal of the ranking algorithm is to help the user disambiguate the "target" person quickly with minimal effort. In turn, the task of the first step is to find clusters that are most similar to the target person being disambiguated. For the latter goal, the algorithm leverages the automatic merging process (that have been described in the previous sections) to generate the set of candidate clusters that are possibly the same as the target cluster. Although this pool might contain erroneous clusters, the overall automated framework is known to produce high-quality result and thus it generates the candidate set of high-quality as well.

To generate the candidate set, the algorithm relaxes the threshold and observes which additional clusters would merge with the target cluster (possibly through transitive closure) – these clusters are added to the candidate set. In this process the algorithm also factors in all the constraints available in the system. They include the constraints in the form of "no" answers provided for the clusters by the user in the previous feedback iterations, indicating that certain clusters are not the same.

Figure 14 illustrates an example of the process of candidate set construction for the target cluster C_t. The algorithm first uses the automated framework to construct a cluster relationship graph. In this graph, each node corresponds to a cluster C_i, and each edge (C_i, C_j) corresponds to the fact that clusters C_i and C_j will merge if the threshold is relaxed. The candidate pool of clusters are then all the clusters that have either direct or indirect connection to the target cluster C_t, as demonstrated in Figure 14(b). Therefore, the candidate set $\mathbb{S}_{cand}$ for C_t is $\mathbb{S}_{cand} = \{C_1, C_2, C_3, C_4\}$.

Then, the goal of Step 2 of the algorithm is to select the set Ω of K nodes from the candidate set $\mathbb{S}_{cand}$ to show to the user for feedback.

6.2.2 Step 2: Ranking Candidate Nodes Based on Impact

Following the above procedure, we have obtained the set of candidate clusters $\mathbb{S}_{cand}$ which potentially co-refer with the target cluster C_t. The next task is to rank them and in order to choose a subset $\Omega \subseteq \mathbb{S}_{cand}$ of K clusters to display to users for feedback. Intuitively, the most similar to C_t clusters should be

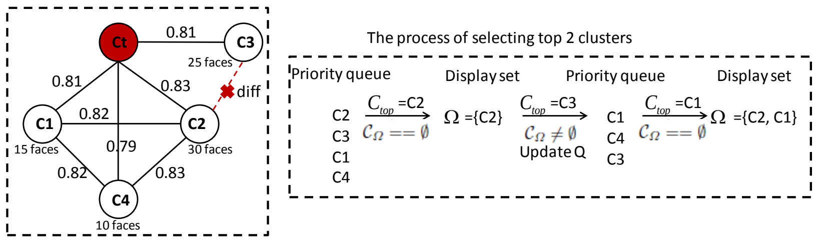


Fig. 15 An Example of Selecting Top K Clusters from Candidate Set

ranked higher in order for the user to quickly merge clusters that co-refer. This factor has already been taken into consideration in Step 1. Since the goal is to minimize the burden on the user, Step 2 of the algorithm takes into account another factor: the impact of resolving clusters, that is, the amount of the reduction of the uncertainty and the ability to maximize recall.

Augmenting the Graph with Similarity Values and Constraints. To better illustrate the concepts related to the impact metric, we will use an augmented version of the above-defined graph to also encode the similarities and constraints computed by the system. Like in the above-defined graph, in the augmented graph each node also corresponds to a cluster C_i and a presence of an edge (C_i, C_j) corresponds to the fact that (C_i, C_j) would merge by the fully-automated part of the framework if the thresholds are relaxed accordingly. The label of the edge (C_i, C_j) corresponds to the similarity S_{ij} computed by the automated part of the algorithm and reflects the probability that C_i and C_j co-refer. For instance, the similarity between C_t and C_2 in Figure 15 is 0.83. Similarly, there is a special label that encode the constraint that two clusters are different, e.g. as for clusters C_2 an C_3 in Figure 15. Notice, such a constraint could arise as part of the automated process (see Section 4.2) or it could reflect the “no” answer provided by the user in one of the previous feedback iterations.

Definition of Impact. We will refer to the process of determining whether the given node (cluster) C_i is same/different from the target cluster C_t as *resolving* the node (cluster) C_i. A node (cluster) C_i can be directly resolved by asking a question to the user on whether C_i and C_t co-refer. Later on we will also see that, in some cases, a node (cluster) C_i could be indirectly resolved without asking the user a question about C_i but rather by factoring in constraints associated with C_i.

Generally, the impact of a node refers to the uncertainty reduction that results from resolving the node. In our case, we can define the impact of one node as the number of uncertain faces that will be resolved (that is, determined whether or not they co-refer with C_t) by providing positive or negative feedback for the node. For example, for two clusters $C_i, C_j \in \mathbb{S}_{cand}$, if C_i and C_j are comparable in their similarity to C_t, but $|C_i| = 100$ (size) whereas

$|C_j| = 1$, then the algorithm might prefer to put C_i into Ω over C_j, as disambiguating C_i has the potential to resolve many more faces thus improving the recall quicker.

The size of a node, while important, is not the only factor to consider for computing the impact of resolving the node. The framework should also take into account which faces will be resolved as the result of enforcing the existing constraints. For example, Figure 15 encodes the fact that clusters C_2 and C_3 do not co-refer based on the existing constraints, denoted as $D(C_2, C_3) = 1$. Thus, if we resolve C_2 to be, say, "yes" (co-refer with C_t), then C_3 should automatically be "no" (does not co-refer with C_t) due to the constraint and vice versa. Therefore, resolving C_2 or C_3 might lead to more faces to be resolved than just their respective sizes.

Overview of the Naive Exponential Solution. Having defined the impact as the overall number of faces that get resolved as the outcome of the user feedback, we now can outline a naive exponential solution that would choose the optimal subset Ω of K clusters from $\mathbb{S}_{cand}$ to maximize impact. Notice, while we do not know whether the user will answer yes/no for a given cluster $C_i \in \Omega$, we can compute the impact in the expected sense, by assuming the user will answer *yes* with probability S_{it} which can be generated from the automatic procedure. The algorithm will simply need to enumerate over all different combinations of K clusters from $\mathbb{S}_{cand}$. For each such combination Ω, the algorithm in turn can enumerate all yes/no responses that the user can generates, which should be valid with respect to the constraints. For each possible response to the K questions, the algorithm can compute the impact by computing how many faces will get resolved. The expectation then can be computed by factoring in the probability of yes/no answer for each given cluster in Ω. After that, the algorithm can choose one (optimal) combination Ω of K clusters from $\mathbb{S}_{cand}$ that leads to the maximal expected impact.

Fast Heuristic Solution. While the above naive algorithm will choose the optimal solution, it is clearly exponential and thus impractical. We now will consider a heuristic solution that is not only fast, but also reaches very high quality results.

First, observe that if we were to include only one cluster in Ω, we can compute the impact factor $\mathcal{I}(C_i)$ of each cluster $C_i \in \mathbb{S}_{cand}$ as:

$$\mathcal{I}(C_i) = |C_i| + S_{it} \sum_{C_j : D(C_i, C_j) = 1} |C_j| \tag{8}$$

That is, after resolving C_i we will know whether or not its $|C_i|$ faces co-refer with C_t, where $|C_i|$ is the number of faces in C_i. In addition, if the user gives the *yes* answer to C_i, then for each cluster C_j such that $D(C_i, C_j) = 1$ we will know that its $|C_j|$ faces do not co-refer to C_t. Eq (8) reflects that, given that the probability that the user will give the *yes* answer can be estimated as S_{it}.

Eq (8) allows us to rank different nodes based on their impact. For example, if we apply it to rank the clusters in Figure 15, the sorted order based on the impact will be $C_2 > C_3 > C_1 > C_4$.

Updating Impact Based on the Already Chosen Clusters Assume that we have already chosen some clusters to be included in Ω and now want to add to it one more cluster C_i from the remaining clusters $\mathbb{S}_{cand} \setminus \Omega$. Notice that we no longer can use Eq. (8) to compute the impact of C_i due to presence of constraints. For instance, if $\Omega = \{C_2\}$ then it is incorrect to compute the impact of adding C_3 to Ω using Eq. (8) – as for instance with probability 0.83 the user will say *yes* to C_2 and thus automatically resolving C_3 to *no* due to $D(C_2, C_3) = 1$ constraint. Using the process identified in the above naive solution will can compute the expected impact, however that computation will be exponential. Instead, we will use heuristic solution that will estimate the impact $\mathcal{I}(C_i|\Omega)$ of C_i given that certain clusters have already been chosen to be in Ω.

For instance, consider computing the impact $\mathcal{I}(C_3|\Omega)$ of C_3 from Figure 15 given that C_2 has already been added to Ω, that is, $\Omega = \{C_2\}$. It can be estimated as $\mathcal{I}(C_3|\Omega = \{C_2\}) = (1 - S_{2t})|C_3|$, because resolving C_2 with probability of S_{2t} will cause a resolution of C_3 and resolving C_3 will only cause a resolution of additional $|C_3|$ faces of the anwser for C_2 was *no*.

In general, we can estimate $\mathcal{I}(C_i|\Omega)$ as follows. Let $\mathcal{C}_\Omega$ be the set of clusters from the display set Ω that conflict with C_i, that is, $\mathcal{C}_\Omega = \{C_j : C_j \in \Omega \wedge D(C_i, C_j) = 1\}$. Similarly, let $\mathcal{C}_S$ be the set of the remaining clusters from $\mathbb{S}_{cand} \setminus \Omega$ that conflict with C_i, that is, $\mathcal{C}_S = \{C_j : C_j \in \mathbb{S}_{cand} \setminus \Omega \wedge D(C_i, C_j) = 1\}$. Both of these sets could be empty. Resolving a cluster C_i will create an additional $|C_i|$ impact only if the user answers to all clusters in $\mathcal{C}_\Omega$ as *no*. The probability that all these answers are *no* answers can be computed as $P_\Omega = \prod_{S_j \in \mathcal{C}_\Omega} (1 - S_j)$. In the special case where $\mathcal{C}_\Omega = \emptyset$, we set P_Ω as $P_\Omega = 1$. If the user answers to all clusters in $\mathcal{C}_\Omega$ as *no* and then she identifies C_i as a *yes* (which should happen with probability S_{it}), then additional conflicting clusters from $\mathcal{C}_S$ will be automatically labeled as *no*, thus contributing additional $\sum_{S_j \in \mathcal{C}_S} |S_j|$ faces to the impact. Overall, we get:

$$\mathcal{I}(C_i|\Omega) = P_\Omega \left(|C_i| + S_{it} \sum_{S_j \in \mathcal{C}_S} \right) \tag{9}$$

We can see that Eq. (9) is a generalization of Eq. (8). Specifically, Eq. (8) is equivalent to Eq. (9) wherein $\Omega = \emptyset$, that is, no clusters have been yet added to Ω.

Greedy Algorithm Using Eq. (9) we can design a greedy algorithm whose pseudocode is shown in Algorithm 1. The algorithm starts with the empty display set $\Omega = \emptyset$ (Step 1); eventually it will greedily add K clusters to Ω one by one. Initially all clusters in the candidate set $\mathbb{S}_{cand}$ are ranked by using Eq. (9) (Step 4) and then inserted into the priority queue Q (Step 5). To choose the

Algorithm 1: The Greedy Algorithm of Selecting Top K Clusters

```
input  : Target cluster $C_t$; Candidate set $\mathbb{S}_{cand}$; Parameter $K$
output: Display set $\mathit{\Omega}$, where $|\mathit{\Omega}| = K$
$\mathit{\Omega} \leftarrow \emptyset$                                   // Initialize the display set
$Q \leftarrow \emptyset$                                   // Initialize the priority queue
foreach node $C_i \in \mathbb{S}_{cand}$ do
    Compute $\mathcal{I}(C_i)$                         // Use Eq. (9) to compute impact
    $Q.insert(C_i, \mathcal{I}(C_i))$          // Insert $C_i$ into $Q$ ranked by impact $\mathcal{I}(C_i)$
while $|\mathit{\Omega}| < K$ and $|Q| > 0$ do
    $C_{top} \leftarrow Q.pop()$                    // Get the first top element of $Q$
    $\mathcal{C}_{\mathit{\Omega}} \leftarrow \{C_j : C_j \in \mathit{\Omega} \wedge D(C_{top}, C_j) = 1\}$
    if $\mathcal{C}_{\mathit{\Omega}} = \emptyset$ then                    // $C_{top}$ doesn't conflict with $\mathit{\Omega}$
        $\mathit{\Omega} \leftarrow \mathit{\Omega} \cup \{C_{top}\}$                          // Add $C_{top}$ to $\mathit{\Omega}$
        $\mathbb{S}_{cand} \leftarrow \mathbb{S}_{cand} \setminus \{C_{top}\}$
    else
        Compute $\mathcal{I}(C_{top})$
        $Q.insert(C_{top}, \mathcal{I}(C_{top}))$                 // Reinsert $C_{top}$ to $Q$
return $\mathit{\Omega}$
```

next element to add to $\mathit{\Omega}$, the algorithm pops the top element/cluster C_{top} from Q (Step 7). The algorithm then checks whether the top ranked node C_{top} needs to be updated (Step 9). If not, then it adds C_{top} to $\mathit{\Omega}$ and either proceeds back to Step 7 to retrieve the next node from the priority queue, or stops if $|\mathit{\Omega}| = K$. If it needs to recompute the rank of C_i, it does so (Step 13), reinserts C_{top} back into the priority queue (Step 14), and proceeds back to Step 7 to retrieve the next node with the best rank.

For instance, for our running example in Figure 15 the algorithm will work as follows. The input of the algorithm is the candidate set $\mathbb{S}_{cand} = \{C_1, C_2, C_3, C_4\}$, parameter $K = 2$, and target node C_t. The algorithm starts with display set $\mathit{\Omega} = \emptyset$, and eventually it will add two clusters to $\mathit{\Omega}$. Initially, all the candidate nodes are ranked based on impact $\mathcal{I}(C_i)$ and inserted into priority queue $Q = C_2 > C_3 > C_1 > C_4$. Then the algorithm will popup the top element from Q and set $C_{top} = C_2$. Since the current display set $\mathit{\Omega} = \emptyset$, then we compute $\mathcal{C}_{\mathit{\Omega}} = \emptyset$. Therefore, we directly add C_2 into display set $\mathit{\Omega} = \{C_2\}$. On the next iteration the algorithm will repeat Step 7 and will set $C_{top} = C_3$. Then it will determine that $\mathcal{C}_{\mathit{\Omega}} = \{C_2\}$ due to $D(C_2, C_3) = 1$. Since $\mathcal{C}_{\mathit{\Omega}} \neq \emptyset$, we need to re-compute the impact $\mathcal{I}(C_3)$ of C_3, and reinsert C_3 into the priority queue, which at this point will become $Q = C_1 > C_4 > C_3$. On the next iteration the algorithm will the first top element from Q, which is going to be $C_{top} = C_1$. It will next determine that $\mathcal{C}_{\mathit{\Omega}} = \emptyset$ for C_1, so it will add C_1 to $\mathit{\Omega}$. At this point, $|\mathit{\Omega}| = 2$, so the algorithm will stop and return the display set $\mathit{\Omega} = \{C_2, C_1\}$.

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

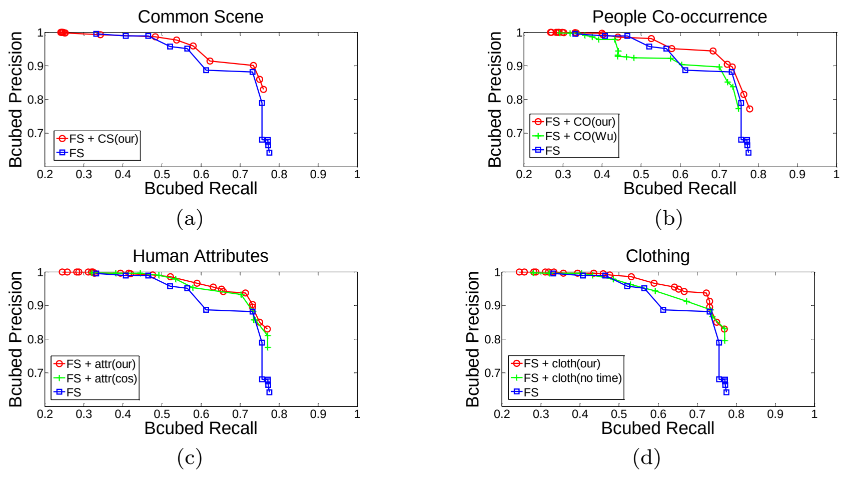


Fig. 16 Effectiveness of Extracted Context Features

7 Experimental Evaluation

In this section, we evaluate our algorithm on three human-centered data collections: Gallagher, Wedding, and Surveillance.[2] The characteristics of these datasets are listed in Table 1. Gallagher [11] 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. 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 [10] algorithms, to illustrate the overall effectiveness of our unified framework.

[2] We note that while larger dataset exists, (e.g., LFW, PubFig), these datasets (LFW and PubFig) are not suitable for our work because they only provide single face rather than the whole image, whereas we focus on disambiguating faces in a photo collection.

7.1 Context Feature Comparison

As shown in Figure16, 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 16 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 16(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 16(b) shows the comparison between our approach (CO(our)) and Wu's approach (CO(Wu)) [21] 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 16(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 16(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.

7.2 Automatic Clustering Results Comparison

To evaluate the performance of the proposed unified framework, we compare our clustering results with affinity propagation (AP) [10] and Picasa's face clustering toolkit, as shown in Figure 17. Four types of context information and facial visual similarities are integrated into our framework. B-cubed precision and recall are computed as the evaluation metrics. 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 17, 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

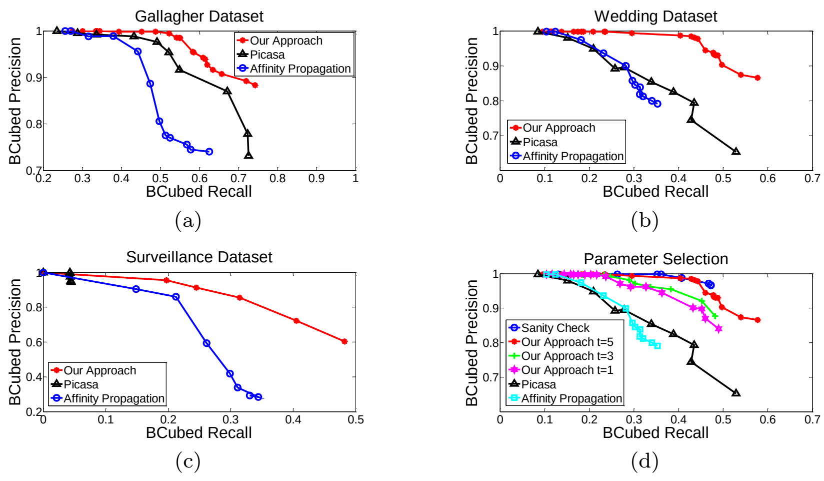


Fig. 17 Comparison of Clustering Performance with Affinity Propagation and Picasa on Three Datasets (see Figure (a)(b)(c)). Figure(d) Comparison of Different Parameters.

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.

7.2.1 Effectiveness and Efficiency

Figure 17(d) 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.

7.3 Human-in-the-loop

In this part, we perform a series of experiments to evaluate the effectiveness of the proposed approaches for leveraging human-in-the-loop methodology to improve data quality. As explained in Section 6, the user interface allows the user to choose the target cluster C_t and then displays the unlabeled clusters in

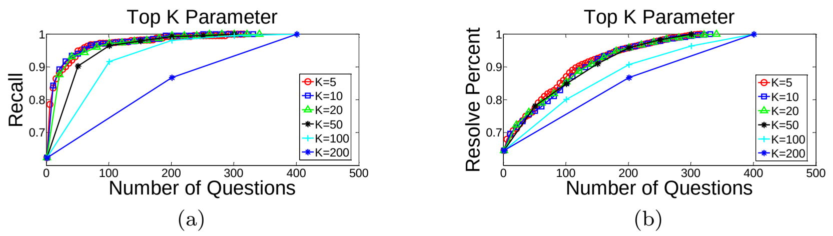


Fig. 18 Comparison of Results with Different Parameter Setting

a ranking order with respect to the chosen C_t. On each iteration, the interface displays K clusters to the user. On each iteration, after receiving user feedback, the algorithm updates the cluster status by merging the clusters answered "yes" with the target cluster C_t, and by labeling those answered "no" as "diff" ones. Then it re-ranks the unlabeled clusters and choose the next K clusters to display. To simulate the user behavior, we assume the user will select 10 largest groups (10 people with the largest number of photos) as target clusters, and then use the average results to evaluate the performance.

7.3.1 Evaluation Metrics

Generally, a good ranking strategy should be able to facilitate the user to quickly label all the related data with minimal effort. Therefore, we propose to evaluate our approach from two aspects: (1) how quickly the user can label all relevant to C_t faces; and (2) how fast the user can resolve all the uncertain (unlabeled) data in general. For this goal, we employ two evaluation metrics:

– **Recall.** Given a target cluster C_t, recall is defined as $Recall(C_t) = \frac{size(C_t)}{\|\{f_i | L(f_i)=L(C_t)\}\|}$, where $L(f_i)$ refers to the labeled ground truth for face f_i. Recall measures the proportion of correctly related faces compared with all the faces which refer to the specified entity.
– **Face resolve ratio.** Face resolve ratio is defined as $Resolve(C_t) = \frac{size(C_t)+\sum_{D(C_t,C_l)=1} size(C_l)}{\|\{f_i\}\|}$, where $D(C_t, C_l) = 1$ denotes that we confirm C_t and C_l can not co-refer. It measures the proportion of resolved faces compared with all the faces in the dataset.

On each iteration, we compute recall and face resolve ratio, and then plot the improvement of them respectively with the number of iterations increasing. A good ranking strategy should be able to achieve the fast improvement for both recall and resolve ratio with the increase of iteration number.

7.3.2 Parameter Selection

An interesting question to study is how to choose the size K of the display set Ω: that is, how many clusters to show to the user at once on each iter-

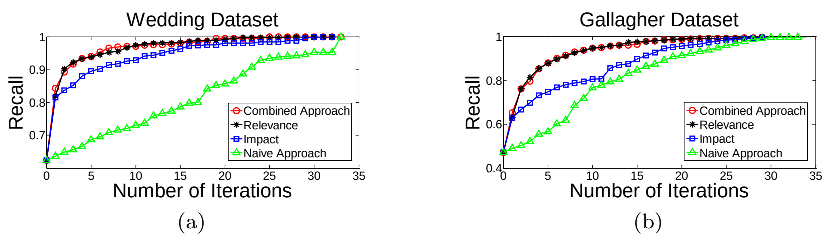


Fig. 19 Recall Improvement with Iterations Increasing

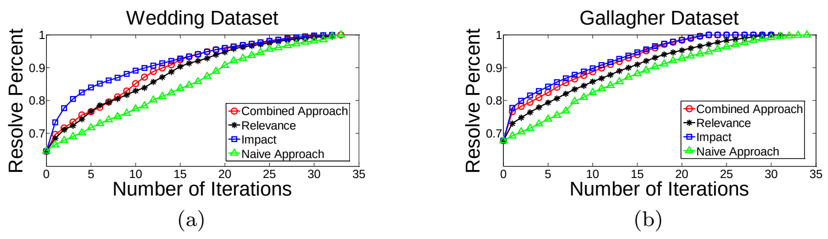


Fig. 20 Face Resolved Percentage Improvement with Iterations Increasing

ation. To choose the appropriate value for parameter K, we select different values (from 5 to 200) and plot the tendency of recall and resolved rate. As illustrated in Figure 18, we discover that the larger values of K lead to the slower improvement of data quality, and more questions required to resolve all the uncertain data. This is because if we choose a larger value for K, the cluster status cannot be updated promptly, and some unnecessary questions will be asked. However, a small K value will result in too many iterations and computation load. Therefore, to trade off between the two factors, we set $K = 10$ in our framework.

7.3.3 Results Comparison and Analysis

In Section 6, we propose that the uncertain data should be sorted by considering two ranking metrics (relevance and impact), and we also present a combined approach that takes into consideration both of the two factors. To evaluate the effectiveness of the proposed approaches, we run experiments with several different ranking strategies as follows.

– *Naive approach*: sort the uncertain clusters in a random order.
– *Relevance based approach*: sort all the uncertain clusters based on their relevance (similarities) to the target cluster, without considering the impact of clusters. This method tends to rank the most relevant clusters higher.
– *Impact based approach*: sort all the uncertain clusters based on impact, without considering the relevance. This method does not include the candidate set selection process.

– *Combined approach*: sort the uncertain clusters considering both relevance and impact. As described in Section 6, this approach consists of two components, selection of candidate set based on relevance and ranking candidates based on impact.

Figure 19 and Figure 20 illustrate the comparison between the above strategies on the two photo collections. Figure 19 demonstrates the tendency of recall and Figure 20 demonstrates that of face resolve ratio, with the increase of iteration number. From the recall plots we can see that the proposed strategies allow to very quickly, in just a few feedback iterations, disambiguating most of the faces for the target cluster as compared to the naive random strategy. Analyzing the overall results, we discover that the relevance-based approach can achieve better performance on recall but lower performance on the resolve ratio. This is because this method aims to put the most relevant clusters first, which facilitate the improvement of recall; however, it does not consider the impact of clusters, which leads to the lower improvement of face resolve ratio. In contrast, the impact-based approach can achieve better performance on the face resolve ratio but lower performance on the recall. That is because this method tends to put the clusters with larger impact first, without the consideration of relevance, which might lead to some irrelevant clusters with larger size ranked higher. Compared with the two methods, the combined approach is able to achieve better performance in both recall and face resolve ratio, because it considers both relevance and impact by consisting of two steps, the selection of candidate set based on relevance and the ranking of candidate nodes based on impact. In addition, we discover that all the three strategies (relevance, impact and combined approaches) can achieve much better performance than the naive approach on both recall and face resolve ratio. Therefore, the comparison in Figure 19 and Figure 20 demonstrates the effectiveness of the proposed ranking metrics (relevance and impact), and also illustrates the superiority of the combined approach in the capability of improving both recall and face resolve ratio. The results imply that the combined approach not only helps the users to effectively disambiguate clusters of interest, but also facilitates quicker reduction in uncertainty in the entire collection.

8 Conclusion and Future Work

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. Finally, we discuss several methods to integrate human-in-the-loop in order to achieve very high-quality clustering results. Our experiments on the real-world datasets demonstrated the effectiveness of the extracted context features and of the overall unified framework.

It should be noted that no system is perfect and several conditions might decrease the accuracy of our system. One particular case is when only the face is provided instead of the entire image, and thus several types of contextual information simply cannot be leveraged by our approach. The unavailability of context features will influence the effectiveness of our system. Another challenging problem is to deal with the unreliable and noisy context features. We need to make sure that the automatically extracted context information is relatively reliable; otherwise, too much noise will impact the clustering performance although our system adopts very conservative strategy to make the merge decisions. In addition, some assumptions are made in our system, such as "faces co-occurred in the same image can not refer to the same person". However, in the real world, exceptions might exist, for example, images processed by some software (e.g., Photoshop). Under these conditions, the performance of our system will be impacted and techniques should be developed to detect doctored images. In general, to handle the aforementioned issues, one of the possible solutions is to leverage the help from users. In this paper, several simple strategies to integrate human-in-the-loop techniques have been discussed. In the future, we plan to explore the human-in-the-loop strategies from a formal probabilistic perspective and in the context of answering SQL-style user queries.

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. L. An, B. Bhanu, and S. Yang. Boosting face recognition in real-world surveillance videos. In *IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 270–275, 2012.
5. L. An, M. Kafai, and B. Bhanu. Dynamic bayesian network for unconstrained face recognition in surveillance camera networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 3(2):155–164, 2013.
6. T. L. Berg, A. C. Berg, and et al. Names and faces in the news. In *IEEE ICPR*, 2004.
7. Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Adaptive graphical approach to entity resolution. In *JCDL*, 2007.
8. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
9. K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. In *AVBPA*, 1997.
10. B. J. Frey and D. Dueck. Clustering by passing messages between data points. In *Science*, 2007.
11. A. Gallagher and T. Chen. Clothing cosegmentation for recognizing people. In *IEEE CVPR*, 2008.
12. A. Gallagher and T. Chen. Understanding images of groups of people. In *IEEE CVPR*, 2009.

13. 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.
14. D. V. Kalashnikov, Z. Chen, S. Mehrotra, and R. Nuray-Turan. Web people search via connection analysis. In *TKDE*, 2011.
15. N. Kumar and et al. Describable visual attributes for face verification and image search. In *IEEE TPAMI*, 2011.
16. Y. J. Lee and K. Grauman. Face discovery with social context. In *BMVC*, 2011.
17. R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Exploiting web querying for web people search. In *ACM TODS*, 2012.
18. K. Shimizu, N. Nitta, and et al. Classification based group photo retrieval with bag of people features. In *ICMR*, 2012.
19. J. Tang, R. Hong, S. Yan, T.-S. Chua, G.-J. Qi, and R. Jain. Image annotation by *k*nn-sparse graph-based label propagation over noisily tagged web images. *ACM Transactions on Intelligent Systems and Technology*, 2(2):14–23, 2011.
20. J. Tang, Z.-J. Zha, D. Tao, and T.-S. Chua. Semantic-gap-oriented active learning for multilabel image annotation. *IEEE Transactions on Image Processing*, 21(4):2354–2360, 2012.
21. P. Wu and F. Tang. Improving face clustering using social context. In *ACM Multimedia*, 2010.
22. J. Yagnik and A. Islam. Learning people annotation from the web via consistency learning. In *MIR*, 2007.
23. L. Zhang, D. V. Kalashnikov, and S. Mehrotra. A unified framework for context assisted face clustering. In *ACM International Conference on Multimedia Retrieval (ACM ICMR 2013)*, Dallas, Texas, USA, April 16–19 2013.
24. L. Zhang, D. V. Kalashnikov, S. Mehrotra, and R. Vaisenberg. Context-based person identification framework for smart video surveillance. *Machine Vision and Applications*, pages 1–15, 2013.
25. L. Zhang, R. Vaisenberg, S. Mehrotra, and D. V. Kalashnikov. Video entity resolution: Applying er techniques for smart video surveillance. In *PerCom Workshops*, 2011.
26. L. Zhang, K. Zhang, and C. Li. A topical pagerank based algorithm for recommender systems. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 713–714, 2008.
27. W. Zhang and et al. Beyond face: Improving person clustering in consumer photos by exploring contextual information. In *ICME*, 2010.
28. M. Zhao, Y. Teo, and et al. Automatic person annotation of family photo album. In *CIVR*, 2006.