Harnessing Social Networks for Social Awareness via Mobile Face Recognition

By

Mark Bloess

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies University of Ottawa in partial fulfillment of the requirements for the degree of Masters in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

With more and more images being uploaded to social networks each day, the resources for identifying a large portion of the world are available. However the tools to harness and utilize this information are not sufficient. This thesis presents a system, called PhacePhinder, which can build a face database from a social network and have it accessible from mobile devices. Through combining existing technologies, this is made possible. It also makes use of a fusion probabilistic latent semantic analysis to determine strong connections between users and content. Using this information we can determine the most meaningful social connection to a recognized person, allowing us to inform the user of how they know the person being recognized. We conduct a series of offline and user tests to verify our results and compare them to existing algorithms. We show, that through combining a user's friendship information as well as picture occurrence information, we can make stronger recommendations than based on friendship alone. We demonstrate a working prototype that can identify a face from a picture taken from a mobile phone, using a database derived from images gathered directly from a social network, and return a meaningful social connection to the recognized face.

Acknowledgements

I would like to thank the University of Ottawa and the Faculty of Graduate and Postdoctoral Studies, and my supervisor Prof. Dr.-Ing. Abdulmotaleb El Saddik, for providing me with the opportunity and scholarships to carry out this research. A special thanks goes to Dr. Heung-Nam Kim for his continued support and assistance on various aspects of my research throughout the completion of this thesis and associated work.

I would also like to thank my family and friends who provided me with constructive feedback and motivation, particularly my girlfriend Allie.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API: Application Programming Interface

FLD: Fisher's Linear Discriminate

GUI: Graphical User Interface

HMD: Head Mounted Display

HMM: Hidden Markov Model

JSON: JavaScript Object Notation

KLT: Karhunun Loeve Transform

LSA: Latent Semantic Analysis

PCA: Principle Component Analysis

PLSA: Probabilistic Latent Semantic Analysis

UI: User Interface

Chapter 1

Introduction

Video surveillance of the world just continues to grow. Whether it be traffic light cameras, security cameras, or personal cameras, people are captured by cameras almost anywhere they go. Government agencies have access to user images from government issued id, such as driver's licenses or passports. Using these they can run face recognition on any of the video captured. This is a great tool for law enforcement, but what if this tool was available to anyone. Of course the public does not have access to government files, but people give away this information publically every day. Social networks allow users to share everything about their lives, and most interestingly give plenty of picture identification. "Tagging", the act of annotating an image with a person's name and location within the image, is enough to match a name and a face. There is no reason a person could not use this information to identify someone. Adding this to the fact that most people carry a camera with them at all times in the form of a mobile phone, the possibility for personal computer assisted recognition begins to be realized.

By harnessing social networks, we are gaining access to a publicly available source for a face recognition database. Although this can be used for personal use, it also poses as an untapped source for identification for law enforcement as well. In particular this can be useful for young offenders who may not appear in a government database. But since there is no restriction on who can access this database, it would be possible for anyone with a Smartphone to identify people. Such a tool could put an end to personal anonymity.

To take things further, social networks have much more information about a person than just their identity. Many users share almost every aspect of their life. Just finding out a person's name is interesting, but what if you could begin to actually know a

person just by seeing their face. If the goal is to be able to know a person well enough to approach them, then the most important piece of information is to see if some form of social tie between you and the other person already exists. The most useful information for determining this is a user's friendship information. Using this you can determine if you have some sort of social connection, whether they are your friend, a friend of a friend, and so on. Similar to friendship information, looking at who a user appears with in images can help determine who a person knows. Determining if a connection exists is only a small step to knowing a person. A weak connection, relations based on users that neither party is very familiar with, may not hold much meaning to either user. The real problem comes in trying to find the best connection to a person.

Determining strong friendships falls into the same category as friend recommendation, which is in turn a form of content recommendation. Content recommendation is an area of increasing interest. As the amount of electronic media grows, it becomes more and more difficult to find what you are looking for. Especially when you consider that most people don't know what they are looking for. Recommender systems try to make good recommendations based on existing knowledge of the user and their interests. Friend recommendation often uses existing friendships as the basis for its decision. This is done for good reason, as it gives a solid social connection to another user, and can indicate how closely tied they are to the same social circles. But if we can also use picture occurrence information we can help to strengthen our recommendations. As a result, a fusion model that makes use of both friendship information and picture occurrence information can give a greater indication of the strength of a relationship between two users.

1.1 Motivation

Social networking is a continually growing medium for online photo sharing. Facebook[1] alone claims 250 million new photos are uploaded each day [1]. With the addition of tagging this makes these images rich with information. Combining this information with face recognition technologies, it is possible to build face recognition databases. With the majority of cell phones today coming equipped with cameras, the ability to capture your surroundings at any given time has never been more wide spread. Combining these two technologies would allow someone to use their phone to capture the people around them and then use face recognition to identify them.

Combining social networks and face recognition is already an area of interest. But we want to see if we can collect these images for face recognition, and use them in conjunction with other information to develop a face recognition system which gives more meaningful recognitions than just a name and a face. By making use of a fusion probabilistic latent semantic analysis (PLSA) [2] we are able to determine which connections between users are the most relevant, in an attempt to give back the most meaningful social connections. It would combine the relationships that exist through friendships and picture co-occurrence in order to give a stronger social awareness than either one alone. This will allow for face recognition that not only identifies a person, but gives us their social relevance in relation to us as well.

1.2 Problem Statement

Remembering people is an issue for some people, and not being able to do so can lead to awkward moments when a person remembers you but you do not remember them. Technology helps to increase our social circles, through social networks and other web 2.0 communities. This just makes the number of people we are expected to remember even larger. Since technology helps connect people, it only makes sense that it should help us remember them. Combining face recognition technologies with the information

[1] www.facebook.com

already existing on social networks can help people identify each other. By further harnessing a social network, we can also inform users how they may know a recognized person, in the form of a social connection. By combining friendships with picture co-occurrences on social networks we can determine the best connection between a user and a recognized person. When applied on a mobile environment this allows for on-the-fly identification of any user of a social network.

1.3 Thesis Contribution

In this thesis work, we have the following contributions

1. Design and development of an image gathering algorithm to construct face recognition databases directly from a social network.
2. Design and development of a fusion probabilistic latent semantic analysis algorithm combining friendship information and picture occurrence information from a social network with the goal of determining relevant content in terms of friends and images.
3. Design and Implementation of a server application to build a face recognition database from a social network as well as a construct a social graph based on a fusion probabilistic latent semantic analysis, and a mobile client application which can perform face recognition on said database.

1.4 Publications Resulting From This Research

The following papers have been accepted to be published and are directly related to the thesis topic.

1. **Mark Bloess**, Heung-Nam Kim, Abdulmotaleb El Saddik, “PhacePhinder: Harnessing Social Networks to Build Social Face Databases for Mobile Devices", ACM MM Proceedings, Oct. 29, 2012, Nara, Japan (accepted).

2. **Mark Bloess,** Heung-Nam Kim, Abdulmotaleb El Saddik, “Knowing Who You Are And Who You Know: Harnessing Social Networks to Identify People via Mobile Devices ", MMM ‘13 Proceedings, Jan. 7, 2013, Huangshan, China (accepted).

1.5 Thesis Organization

The remainer of this thesis is organized as follows:

Chapter 2 will cover any background information required to follow the rest of the thesis. It will also go into detail on several related works pertaining to the research covered in this thesis.

Chapter 3 will detail the methodology of our research. This will include an explanation of the system we are proposing, and go into detail of the algorithms we are implementing. We will also detail the requirements our system is intended to fulfill.

Chapter 4 will detail the implementation of the PhacePhinder system. It will cover the system design, database design, and user detail of the user interface. This includes details of our working prototype.

Chapter 5 will detail various tests intended to validate our research. This will involve leave-one-out tests for face recognition accuracy, offline cross validation testing, comparing our fusion PLSA model to existing algorithms, as well as user testing pertaining to content recommendation.

Finally in **chapter 6** we will conclude our research, as well as propose future directions this research can take.

Chapter 2

Background Information and Related Work

This thesis covers several different areas of research. It includes face recognition, social networking, and content recommendation. The following section will give background information on these topics in order to help understand the thesis to a fuller degree. Additionally we will discuss several related works that share similar properties to our research, and highlight the differences between them.

2.1 Face Recognition

Face Recognition has been a popular form of pattern recognition for a while now [45][46][47]. It has made its way into many commercial products and applications. The software for face recognition itself has become its own commercial product. It is used by law enforcement to identify criminals [48], used by cruise lines to sell pictures to passengers that they appear in, and sometimes used for security access [49]. As the technology improves, its incorporation with other products will only continue to increase.

2.1.1 Hidden Markov Model Face Recognition

Hidden Markov models are a set of Markov models which have unobserved states [9]. The output from a state can be observed, but the state itself cannot, in other words it is hidden. HMM is used for many different recognition tasks, such as speech [19], character recognition [20], and bioinformatics [21].

Nefian describes Hidden Markov Model (HMM) face recognition as follows [9]. HMM face recognition works by extracting features and putting them into a one dimensional continuous HMM. It starts by separating hair, forehead, eyes, nose, and mouth. These are all easily separated from top to bottom. Even under slight rotation this is still true.

Karhuen-Loeve Transform (KLT) coefficients are used for the observation vectors. The feature blocks (hair, forehead, etc.) are extracted and used to form vectors. The eigenvectors of the covariance matrix, which correspond to the largest eigenvalues, are used to form the KLT basis. The mean vector, of the vectors used to compute the covariance matrix, is subtracted from each of the feature vectors. The result is projected onto the eigenvectors which in turn results in the observation vectors. Training consists of performing this method on several images. However for the first image, the training data is segmented into 6 states, and for each image after, a Viterbi segmentation is performed instead. The Viterbi segmentation must converge before the HMM is initialized and then a Baum-Welch algorithm is used until convergence to result in the final parameters of HMM. The ability to account for some change in pose makes HMM an appealing choice for dealing with social network images.

2.1.2 Principle component analysis and Eigenfaces

Principle component analysis (PCA) as used for Eigenfaces was created by Sirovich et al. [25]. They showed that an individual face can be recorded as a set of numbers and images known as Eigenfaces. It was then used by Turk et al. [26] for face detection and recognition. It works by reducing the dimensionality of the face image in order to reduce storage space and increase speed. This kind of technique is called a Karhuen-Loeve method. It makes use of a dimensionality reducing linear projection which will project from image space, which would be the original images, into feature space, which would be the Eigenfaces. Although this method is computationally conservative, it is very susceptible to variation in lighting and pose. This is because the variation in

lighting may be projected into feature space, which will increase the "scatter within a class" (difference within a specific identify), while ideally it is the "between class scatter" (difference between separate identities) you wish to maximize [28]. This is not ideal when dealing with images from social networks as they are not likely to be taken under ideal circumstances, and the conditions can vary drastically from image to image.

2.1.3 Fisherfaces

Fisherfaces can be considered an improved version of Eigenfaces. Eigenfaces' linear combinations come from PCA, while Fisherfaces obtains its linear combinations from Fisher's linear discriminate (FLD) [28]. FLD is considered a "class specific method" which helps to shape the scatter instead of just trying to maximize it. Fisherfaces still uses PCA to reduce the dimensionality of the feature space, but then uses FLD to reduce it further, while giving it shape. This allows for the increase in total scatter offered by Eigenfaces, but offering an improved (reduced) level of within class scatter, which results in easier classification.

2.2 Probabilistic Latent Semantic Analysis

Content on the internet is growing at an extremely rapid rate. As a result, content recommendation systems have been an area of growing interest [29][30][31]. They have a large variety of applications, whether it be recommending movies a person may want to watch [40], items they may want to purchase [41][42], or even people they may want to get to know [43][44]. Looking through all the information manually can be an overwhelming task, so recommender systems are becoming more of a necessity than a commodity. There are many different algorithms which have the capacity for recommendation.

The algorithm most pertaining to this thesis is Probabilistic Latent Semantic Analysis (PLSA). It has been applied in fields such as filtering [50] and

recommendation [51]. Latent Semantic Analysis (LSA) was created in 1988 by Dumais et al. [16] with the intention of dealing with the issue of matching vocabulary to documents they occur in. PLSA was made by Hofmann to improve upon LSA [17]. It is based on a statistical approach as opposed to one based on linear algebra. Considering the original application of matching words to the documents they are found in, it can be easily related to other topics. In the realm of friend recommendation, the difference would be that a word can appear in a document multiple times, but a person can only be friends with another person once. So it would relate in a case where each word would only appear in any single document once, but can appear in many different documents.

PLSA works by associating an unobserved class variable with each observation, which we refer to as hidden topics in this thesis. Using the example of words belonging to documents, we can look at the joint probability model [17] of:

$$P(w \mid d) = \sum_{z \in Z} P(w \mid z) P(z \mid d) \qquad (1)$$

where w is a word, and d is a document. $P(w|d)$ is the probability of a word belonging to a document. z is the hidden topic belonging to the set of hidden topics Z. The number of z variables in the set Z is less than the number of words or documents, and as a result acts as a bottleneck when it comes to predicting words. The model is then fit using an expectation maximization algorithm.

2.3 Social Networks

Social networks are a main part of everyday life for many people now. Their popularity has continued to grow, even when large ones seem to fade away such as Friendster[2], which has now relaunched as a social gaming platform, or Myspace[3]. Facebook is by far the largest of the online social networks, with 955 million monthly active users [1]. Online social networks allow users to stay connected with each other through

[2] www.friendster.com
[3] www.myspace.com

communication or shared media. The most shared form of visual media would be images. Users post images online and annotate them with other users' names so that the images are brought to their attention and linked to them for others to see.

With Facebook going public in 2012, the need to prove the financial worth of a social network is growing too. Advertising is the biggest way to make money, however the information on social networks is also very valuable. Currently Facebook does not sell users' information, however policies may change if the company value continues to drop. The past trend of social networks rising and falling may repeat itself.

Due to the amount of information existing on social networks, they have become a target for various types of research. These range from automated detection of social groups within a social network [32][33], to mining for key users for targeted marketing [34], to a desire for improved ways to store large amounts of data [38]. Their large scale and source of real life information makes social networks an appealing target for research. The research conducted using social networks, in particular Facebook, extends to studies on social behavior [39][37] and even the emotional impacts of sharing all this personal information[36]. Most of this research involves exploiting the information which people give away freely in some way or another, and our research is no different.

2.4 Related Works

Combining social networks with face recognition has been an area of high interest. Specifically improving the accuracy through use of context gathered from social networks. Often times the interest is in auto-annotation, which is the act of automatically tagging an image. The work conducted by Stone et al. [3] very closely resembles the database construction portion of our research. They collect face images from the social network Facebook, and use these to construct a face recognition database. Using context such as which users are often tagged by a specific user, or which users often appear in photos together, you can make assumptions on who a specific user may be, even before performing face recognition. However since this is for

automatic annotation, they can also make assumptions such as the user most likely residing in the friends list of the uploader. In our case we cannot make this assumption. Since we could be recognizing anyone, it is not safe to assume the recognized person would be closely related to the user taking the picture.

Similarly, Mavridis et al. [4] also demonstrates how using context gained from a social network can improve recognition accuracy. Their research is intended for use in the FaceBot system, which is a robot which can access social networks and perform real time face recognition. If you consider this robot to be a mobile platform, similar to a smartphone, then this relates even more closely to our research. However, again they use context such as friendships and co-occurrences to improve accuracy. Their methodology could be useful for recognizing multiple faces in the same image. They determine that two users appearing in an image together have approximately 80% likelihood that they are friends. So if the confidence in the recognition of one face is high, we can use this as seed information to improve the recognition of the second face. However, if there is only one face to be recognized, we should not assume it is going to be a friend of the user taking the picture. The intended use for our research is to identify any person, which could mean any level of social separation between users.

Another example of context helping improve recognition is done by Davis et al. [5]. In their research, they developed a mobile application which is the source for their entire dataset. So although the end result is a similar issue, the fact that they control all of the data makes it a different problem. Controlling the data ensures that the dataset will contain all the information they want. So although it does demonstrate that context can improve on pure computer vision face recognition, we must determine how to use the context data from a social network when there is no guarantee it even exists.

Akbari et al. [24] suggest using gender estimation techniques in order to improve recognition rates. They state that if they can accurately estimate gender, then they can improve the recognition rates of PCA face recognition by as much as 7%. This closely relates to one portion of our research. Social networking sites often restrict access to

most private information pertaining to users. An exception to this is the gender of a user, which is often public. As a result, we too use gender to increase the accuracy of recognition. The difference is that they use gender estimation techniques. In the practical use of our application, a user is trying to recognize one user at a time, so we can let the user input the gender. So this means we will always know the gender prior to recognition, so we can always benefit from this increase in accuracy.

The second part of our research deals with determining whether connections exist, and which of the connections is the most relevant to the user. Finding connections between users is also a topic of interest for research, especially in the field of friend recommendation. Kim et al. [6] suggest using image co-occurrence as a factor when trying to find strong connections between two users. They constructed a face co-occurrence network to store the picture occurrence information which is used to help determine new friendships. They show that face co-occurrence, based on person tags in images, can be a strong basis for determining new friends on a social network. Although picture occurrence is shown to be useful, we do not want to completely ignore existing friendships as a source for determining new friendships. As a result we combine the picture co-occurrence information with the friendship information to determine a stronger relationship than either alone. Another difference here is that although we can use this information to determine new friends, we also want to use it to determine the strength of existing friendships, not only between the main user, but between other users which could make up a social path to a recognized person. So in our case the end user is known, in the form of a recognized face, and instead we are trying to find the best connection to this user, if one exists at all.

Table 2.1 summarized these related works, describing how they are similar, and how they differ.

Table 2.1 Related Work Comparison Summary

Paper	Similarity	Difference
Stone et al. (2010) [3]	We collect images from a social network in order to build a face recognition database very similarly to them.	We take it further and use the image occurrence and friendship relations to find meaningful social connections to the recognized person.
Davis (2005) [5]	We build a face recognition database, with various user information, similarly to them. They show that the contextual information can assist in improving face recognition accuracy.	They collect all their information from a mobile app. It does not interact with a social network in the way ours does, so they control all the information being collected. This ensures that the information is complete and correct.
Mavridis et al. (2010) [4]	They have a face recognition robot which collects information from Facebook for face recognition which acts similarly to our mobile application. The robot can take pictures and recognize a user based on information from a social network.	Their research is centered on user relations to the robot as a person. It relies on assumptions such as the people in the photos being closely related to the person taking the picture(in this case the robot, as it has its own Facebook profile). In our research we cannot make this assumption, and instead find social connections post recognition.
Akbari et al. (2010) [24]	They show that if the gender of the face being recognized is know, they can improve face recognition accuracy up to 7%.	They use gender estimation techniques to attempt to determine the gender of the face being recognized. We allow a user to state the gender of the person to be recognized, eliminating the need for estimation techniques.
Kim et al. (2012) [6]	They use picture occurrence information for a basis for friend recommendation. They build a graph where co-occurrences make up an edge between users.	We use picture co-occurrence information in conjunction with friendship information to strengthen the recommendations. Our social graph is comprised of both relations instead of just one. We also deal with finding the best existing relations as opposed to finding the best new connections.

Chapter 3

Proposed Approach

In this chapter we draw the requirements of our proposed system and discuss the methodology for performing the system functions.

3.1 Requirements of the Proposed Social Network Recognition System

The system should be mobile: The usability of the system increases greatly once it is introduced to a mobile environment. By allowing the user to identify people on the fly, the entire concept becomes much more desirable. The alternative would be taking a picture and identifying the faces in the picture at a later time. Although this would have some real life application, it would lean more towards an automatic annotation system rather than an identification system.

The system should identify a user by more than just their name: Matching a name to a face can be difficult. If the system only returned the names of a recognized user, it could still leave the user unsure of the identity of the face being recognized. In order to avoid this, a profile image should also be returned so that the user can see the face which belongs to each possible name. Additionally the user should be able to see their connection to the recognized user. This will help them understand how they are socially related to the person they are attempting to recognize.

There should be a complete separation of the client interaction and the information gathering: The client should not be aware of any of the background functionality of the system. Their only interaction with the collection process should be logging in and granting permission to their information. Any problem with the gathering component should have no impact on the client.

To see a detailed set of use cases for system functionality see Appendix A1.

3.2 Image Gathering

In order to build a face recognition database, we need to collect faces. In order to obtain these faces we will retrieve images from a social network. When collecting these face images, we only consider images on a social network in which there occur manual tags as well as detected faces. Any image that has tags but no face is detected, or that faces are detected but has no tags, is not considered at all. Since all the tags are done manually there are issues that arise. For example, if a tag is placed on the back of someone's head, there is now a tag that cannot be matched to the correct user. To make things worse, if there is a very clear face without a tag, it is possible that this face could be incorrectly matched to another tag. It is very difficult to decide what distance is too far to be considered anymore. As well, if a person were to incorrectly tag someone, there is no way to avoid adding the incorrect face to a specific user. We could attempt to perform face recognition to a face to ensure it matches the user we are adding the image to. However this could only be done once the database has grown to a reasonable size where it can adequately recognize users.

In order to solve the issue of mismatching tags and faces, we do three things. First, faces found on social networks are referred to as “in the wild” [3] or unconstrained. This means they can vary immensely such as in pose, lighting, and rotation. So any face which is detected must undergo a second face detection process in order to reduce the chance of a non-face being detected, and reduce the number of obscured faces, such as in Fig. 3.1.

Figure 3.1 Two faces which passed one face detection but failed at a second

Second, we will then only consider a tag to belong to a face if it occurs within the *x*, *y* plane of the detected face, as is done in [8]. This is based on the assumption that most people will click on a face when adding a tag. Fig 3.2 illustrates an image taken from a social network which has both a social network tag and a detected face. The white box is the manual tag from the social network, while the red box is where the face detection found a face. In this case the center of the tag is well within the area of the detected face and is an example of a face which would be added to the database. Third, if more than one tag occurs within a detected face we calculate the distance from the center of the face to each of the tags, and assume the closest distance is the correct tag. Eq. (2) is used to calculate distance, where *tagX* and *tagY* are the *x* and *y* coordinates of the center of the manual tag respectively, and *faceX* and *faceY* are the *x* and *y* coordinates of the center of the detected face respectively.

$$\texttt{distance} = \sqrt{(tagX - faceX)^2 + (tagY - faceY)^2} \tag{2}$$

This criteria excludes many images which do have both tags and faces. However, it ensures that only high quality face images are added to the database, which could in turn reduce the accuracy more than having a smaller number of images.

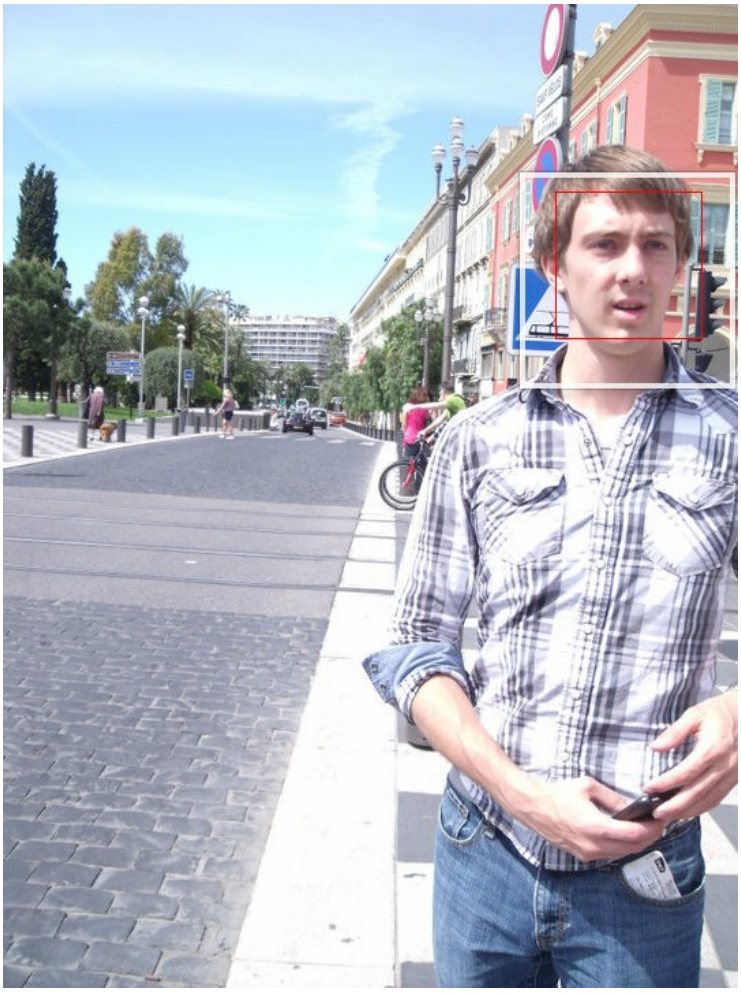


Figure 3.2 Social network image with a detected face and manual tag

3.3 Information Gathering

In addition to building the face recognition database, we also collect information about the users. We attempt to gather information such as name, gender, birth date, hometown, current location, etc. The hope is that this contextual information can be used to improve accuracy as is done in [5][4][7]. Even something as simple as gender can effectively reduce the search space by half. Akbari et al. [24] demonstrated that determining gender prior to recognition can increase accuracy as much as 7%.

Even more importantly, we gather friendship information. This records which users are friends with each other. A friendship can be thought of as a binary relation. $f(u_a, u_b) \in \{0,1\}$ where $f(u_a, u_b)$ is a friendship between two different users u_a and u_b.

Where a 0 value would indicate that no friendship exists, and 1 would indicate a friendship does exist. This could be represented as an *n* by *n* matrix such as Figure 3.3 where *n* is the total number of users collected from the social network. All friendships are bi-directional. As a result the matrix is symmetric. Additionally $f(u_i,u_i)$, for any user u_i, is always 0 as this would represent a friendship with themselves.

	U_1	U_2	U_3	…	U_{n-1}	U_n
U_1	0	1	0	.	1	0
U_2	1	0	1	.	0	0
U_3	0	1	0	.	0	1
…	.	.	.	.	.	.
U_{n-1}	1	0	0	.	0	1
U_n	0	0	1	.	1	0

Figure 3.3 Sample of the user by user matrix used to indicate friendships

Similarly we record which images users appear in when we collect the tagged faces. This information can be represented very similar to the friendship information. $p(u_a, i_b) \in \{0,1\}$ where $p(u_a, i_b)$ is the picture occurrence information between a user u_a and an image i_b. A value of 1 indicates that the user appears in that image, and a value of 0 indicates that the user does not appear in that image. This can be represented as an *n* by *m* matrix, where *n* is the number of users and *m* is the number of images, such as in Figure 3.4.

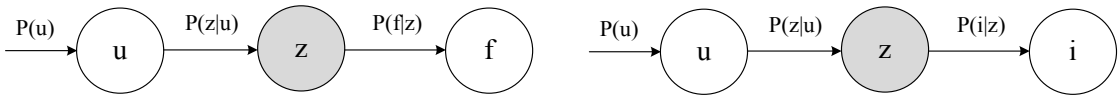
	I_1	I_2	I_3	…	I_{m-1}	I_m
U_1	0	1	0	.	1	0
U_2	1	0	1	.	0	0
U_3	0	0	0	.	0	1
…	.	.	.	.	.	.
U_{n-1}	1	0	0	.	0	0
U_n	0	0	1	.	0	0

Figure 3.4 Sample of the user by image matrix used to indicate proto occurrences

As one could imagine, the majority of this matrix would be 0s, since a typical image would only contain a few people. As a result, when storing both of these matrices it is only necessary to save the entries with a value of 1 to help keep storage space to a minimum.

3.4 Content Recommendation

The information we gather can help us gauge the connection between users and photos. In order to do so, we use the PLSA model [**Error! Reference source not found.**] to determine a quantitative representation of a user's connection to other users and photos. The PLSA model attempts to explain a set of co-occurrence pairs in terms of a set of k latent variables (or factors), $Z = \{z_1, z_2, \ldots, z_k\}$. As shown in Fig. 3.5, in our study a latent variable $z \in Z$ is associated not only with every user–friend pair (u, f)—meaning that user u is connected to friend f—but also with every user–photo pair (u, i)—meaning that user u appears in photo i.



(a) user-user friendship (b) user-picture occurrence

Figure 3.5. Graphical representation of two distinct PLSA models

By assuming that user *u* and friend *f* or user *u* and photo *i* are rendered conditionally independent given a latent variable, one can define the following models for computing predictive conditional distributions of friends and photos given a user, respectively:

$$P(f \mid u) = \sum_z P(f \mid z)P(z \mid u) \tag{3}$$

$$P(i \mid u) = \sum_z P(i \mid z)P(z \mid u) \tag{4}$$

Since both models share the common factors $P(z|u)$, we unify the two models into a probabilistic fusion model in a manner similar to Baum et al. [18]. Following the maximum likelihood approach to statistical inference, we determine $P(f|z)$, $P(i|z)$, and $P(z|u)$ by maximizing the log-likelihood function:

$$L = \sum_u \left[\alpha \sum_f \log P(u)P(f \mid u) + (1-\alpha) \sum_i \log P(u)P(i \mid u) \right] \tag{5}$$

where parameter α, $0 \le \alpha \le 1$, is a relative weight that grants more significance to either photo occurrence or friendship. The typical procedure for finding maximum likelihood parameters is to use the well-known Expectation Maximization (EM) algorithm. During the E-step, the posterior probabilities of the latent variables associated with each observation can be computed by:

$$P(z \mid f, u) = \frac{P(f \mid z)P(z \mid u)}{\sum_{z'} P(f \mid z')P(z' \mid u)} \tag{6}$$

$$P(z \mid i,u) = \frac{P(i \mid z)P(z \mid u)}{\sum_{z'} P(i \mid z')P(z' \mid u)} \tag{7}$$

Then, the conditional distributions are recomputed in the M-step as follows:

$$P(f \mid z) = \frac{\sum_{u} P(z \mid f,u)}{\sum_{f',u} P(z \mid f',u)} \tag{8}$$

$$P(i \mid z) = \frac{\sum_{u} P(z \mid i,u)}{\sum_{i',u} P(z \mid i',u)} \tag{9}$$

along with the following fusion:

$$P(z \mid u) = \alpha \frac{\sum_{f} P(z \mid f,u)}{|F(u)|} + (1-\alpha) \frac{\sum_{i} P(z \mid i,u)}{|I(u)|} \tag{10}$$

where *F(u)* is the set of friends user *u* has and *I(u)* is the set of photos user *u* occurs in. Once the model parameters are learned, we use probability distributions over potential friends for a given user *u*, i.e. $P(f|u)$, to recommend new friends as well as to estimate a relationship weight between user *u* and his/her current friends. It is worth noting that conditional distributions of photos given a particular user, i.e. $P(i|u)$, can also be used for recommending/ranking photos personally tailored to him/her.

The friendship input for this algorithm would be an *n* by *n* matrix, where *n* is the number of users. This matrix would be binary, so either there exists a friendship or not. The friendship input would be a matrix such as Figure 3.3. Similarly, the picture occurrence input is also a binary matrix. However it is an *n* by *m* where *n* is again the number of users and *m* is the number of photos. Either a user appears in a photo or they do not. Figure 3.4 is an example of what the photo occurrence input matrix would look like.

Chapter 4

Design and Implementation

The implementation is separated into a client-server architecture. The client portion is intentionally minimalistic. This is due to it running on a mobile device. As a result, the server side performs the majority of the functionality, including the population of the database, and the actual face recognition. We call this system PhacePhinder (pronounced Face Finder). This section will describe in detail the design and implementation of the entire PhacePhinder system.

4.1 System Layout

4.1.1 High Level System Architecture

The system is a client-server application. Fig. 4.1 illustrates the high level architecture for the PhacePhinder system. Each component performs as follows.

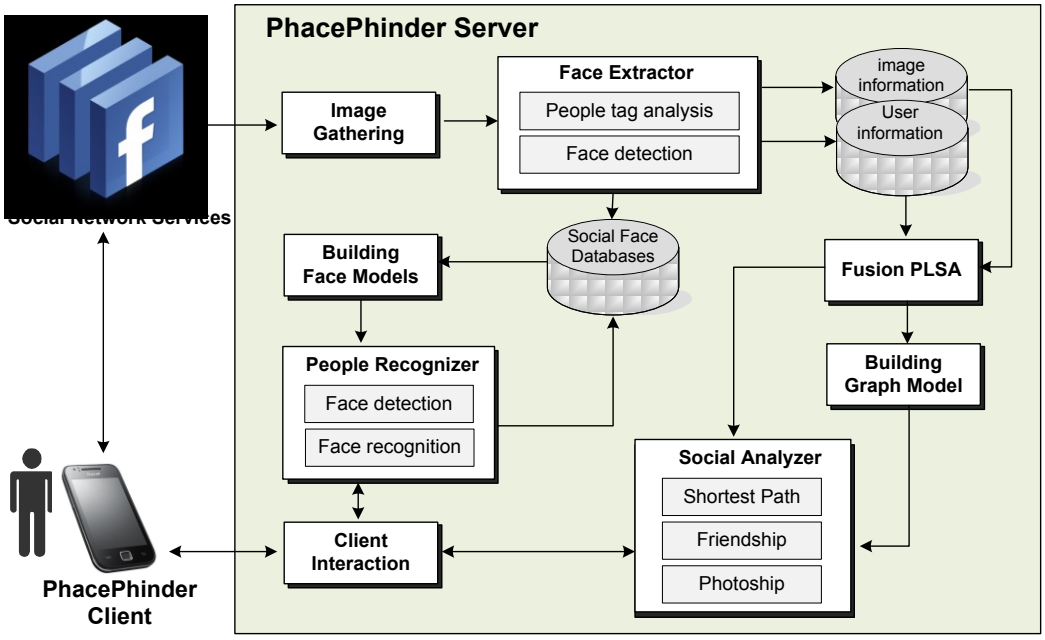


Figure 4.1 PhacePhinder system layout

PhacePhinder Client: This portion resides on a mobile device. It communicates with the social network only to allow the user to grant permission to access the user's information to the PhacePhinder application and verify the login information. All other communication is with the PhacePhinder server. It allows the user to take and send images. The client will receive back the recognition results. The client can also request to view a social path between the user and a recognized user.

Face Gathering Component: This component is responsible for going through photos and information on the social network. It connects to a primary user's account, where a primary user is a user that has granted permissions to our application, on the social network, in order to gain access to private information. It passes images to the face extractor component to be analyzed.

Face Extractor Component: This component receives images from the image gathering component. If the image has tags it will attempt to detect faces and match them to the tags using the methodology described in section 3.2. Once this is done it will store the user and image information to the information database. This face image is stored in the face database.

Building Face Models Component: This component is responsible for using the extracted faces to train the face recognition database for each user.

Face Detection and Recognition Component: This component is responsible for all face detection and recognition on images coming from the client. When an image is received, face detection occurs and face recognition is performed using the recognition information from the face models component.

Client Interaction Component: This component is a web server which deals with the http requests from a user and issues the appropriate responses. It passes images to the face recognition component to be recognized, as well as stores the user information received from the client.

Fusion PLSA Component: This component is responsible for running the fusion PLSA on the information in the database and recording the results. This information is used for building the social graph and content recommendation.

Building Graph Model Component: This component builds the weighted social graph which is used to find the connections between two users. It uses the values generated by the fusion PLSA component as weights for the graph.

Social Analyzer Component: This component is responsible for finding the best social connection between two users. It will receive two user ids from the client interaction component. Using the social graph it will find the shortest path between these two users and return this to the client interaction component.

Social Face Database: This database stores all face images and face recognition information for users in the form of Hidden Markov Models.

The User Information Database: This database stores all other information about a user, including personal information, user and photo relations, and the social graph.

4.1.2 Server Class Diagram

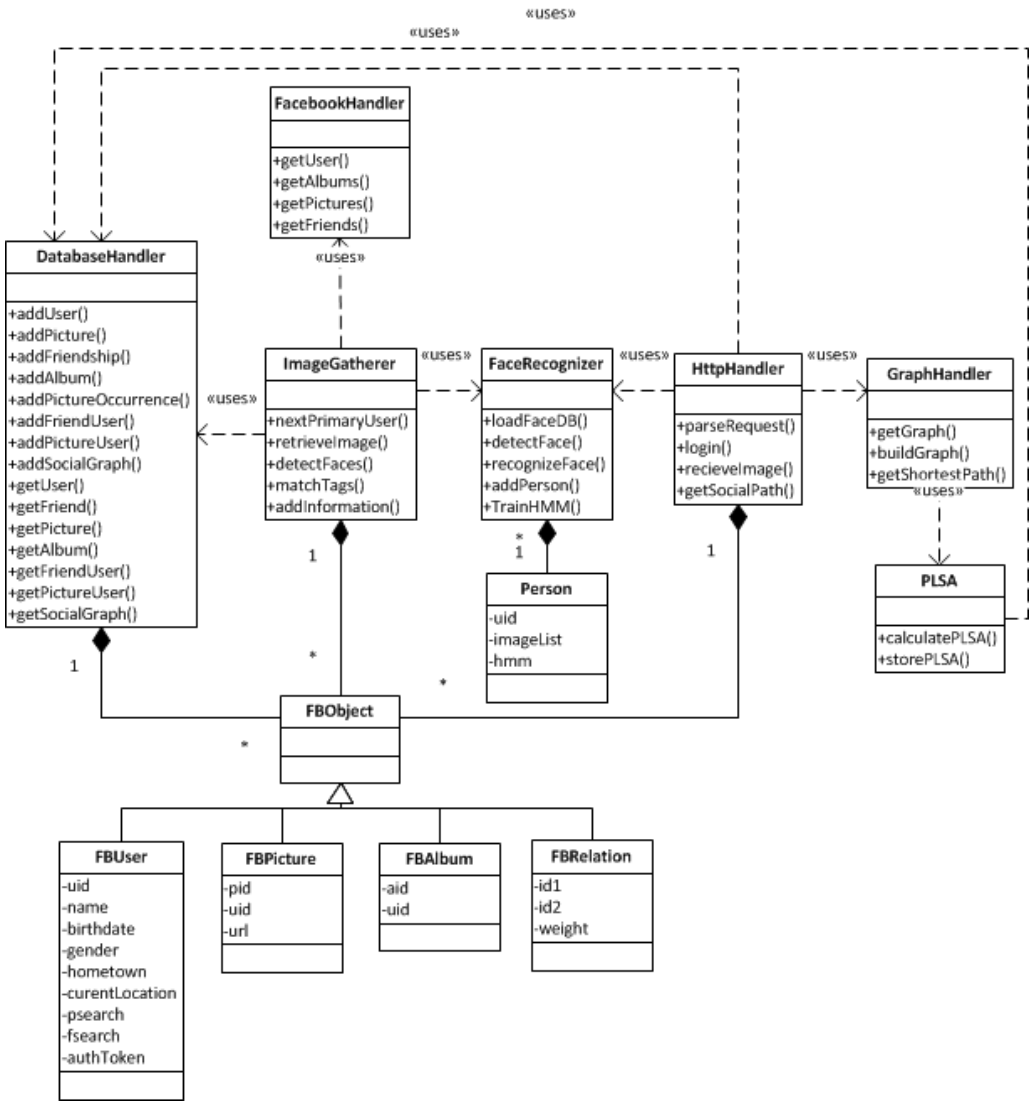


Figure 4.2 UML diagram of the PhacePhinder server

Figure 4.2 illustrates a UML class diagram of the PhacePhinder server. The structure closely resembles that of the system level architecture. The FacebookHandler class is what processes all requests and responses to and from Facebook. The HttpHandler class handles all requests and responses to and from the client application. The DatabaseHandler class deals with all reading and writing to the MySQL database which

holds all the user information. The FaceRecognizer component performs all face detection and face recognition, and deals with the face database directly. The Facebook objects allow for the handling of each of the Facebook classes, involving users, pictures, albums, and relations. A relation can be a picture occurrence or a friendship. The weight attribute is used when handling PLSA values. The ImageGatherer class manages gathering images from Facebook. The Person object contains the face recognition information for a user, and is only used by the FaceRecognizer class.

4.2 Database Design

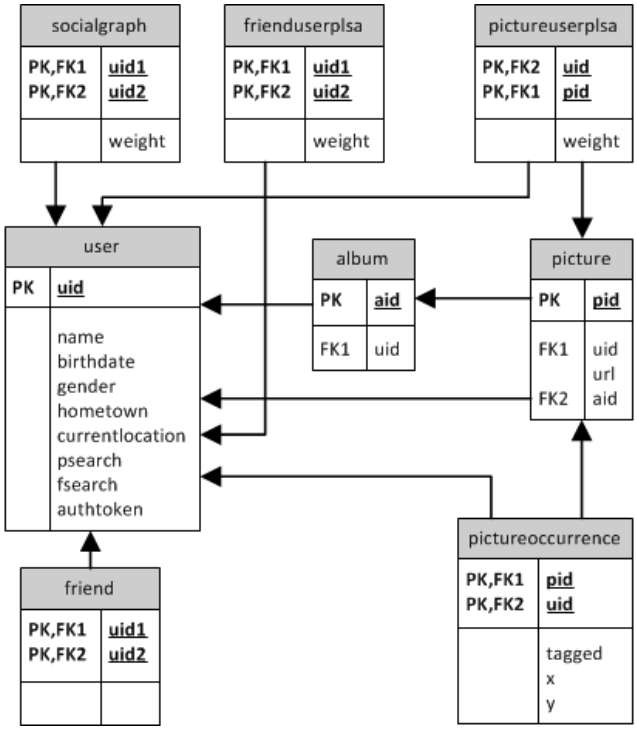


Figure 4.3 Database design for the PhacePhinder system

user: This table records all the information for each user. It contains their id, name, birth date, gender, hometown, current location, an indicator if the user's photos have

been searched, an indicator whether their friends list has been searched, and their authentication token. The user id and authentication token are all that is required to access the user's information. The token is generated by Facebook upon the user granting the application permission to access their account information. The indicators, psearch and fsearch, are meant to make sure we do not waste time going through a user's images or friends twice.

picture: This table contains the picture id, the uploader id, and the picture URL. These URLs are public. Although they are hosted by facebook, you do not need to be logged in to view them. As a result, they could be used to display the image to anyone. These are also the URLs which the application uses to download the images.

pictureoccurance: This table contains a picture id and user id. These are both foreign keys and together create the primary key. As well as these it contains the x, and y coordinates of the tag, as given from the social network. It also contains the tagged indicator. This is an indicator which details if this was a manual tag. The intention for this indicator is if the application eventually performs autotagging. It could be important to distinguish which tags are confirmed by a user and which are not. However, at this point in time no automated tagging is done so this value is always the same. The entries in this table indicate that a user appeared in a specific picture. More importantly, each entry in this table correlates to a face image saved to the face recognition database. This is important information for the fusion PLSA algorithm and for constructing the social graph.

friend: This table contains two user ids. Both are foreign keys and together form the primary key. It is simply a record of which users are friends with each other. This information is important for the fusion PLSA algorithm and for constructing the social graph.

frienduserplsa: This table also contains two user ids. Both are foreign keys and together form the primary key. Additionally it contains a weight. This table stores the value generated by the fusion PLSA between each pair of users. This table will be a source for the weighted social graph, as well as the basis for friend recommendation.

pictureuserplsa: This table contains a picture id and a user id. Both are foreign keys and together form the primary key. It also contains a weight. This table stores the value generated by the fusion PLSA between each picture and each user. This table is the basis for picture recommendation.

socialgraph: This table also contains two user ids as. Both are foreign keys and together form the primary key. Again it also contains a weight. The difference is that this table only contains a pair of users for which there exists an actual connection. This means the two users appear in a picture together or they are friends on the social network. The weights in this table are the inverse of those in the frienduserplsa table. The reason is that they will later be used to find the shortest path between users, so a smaller number should mean a closer connection.

4.3 Client Side Implemented Functionality

The client side application has been developed for the Android[4] operating system with the intention of being run on mobile phones equipped with this operating system. The social network we chose to incorporate with was Facebook. We made use of Facebook's provided Android library to handle interaction with the social network. This required registration of an application on Facebook. In order for this application to gain access to user information on Facebook, a user must first grant it permission. Upon

[4] http://www.android.com/

opening the application for the first time, the user is requested to sign into Facebook. After signing in, permission is requested to access their basic information, which is standard for all Facebook applications. In addition to this, several extended permissions are requested. Permission to access the user's photos is requested, so that they can be collected for the database, as well as additional information such as their birth date, their hometown, and their current location. We ask for permission to access all the same information on their friends' accounts as well. Information such as their name and gender is public by default. We also requested offline access to their account allowing the application to gather their information even while they are not logged in, however this functionality has since been deprecated by Facebook.

Once the user has granted these permissions, the client side will send the user's id and OAuth token to the server. OAuth is an open standard for authorization [10] which is used by Facebook. Originally this token would never expire with offline access, however now if the user logs out of the application, it will expire and need to be renewed. Additionally, the user can revoke the application's permission to their information via their Facebook profile.

Now the application will prompt the user to take a picture. Once they do, they are prompted to state the gender of the user. This image and the gender are then sent to the server via an http request. The client now waits to receive the names of the top three closest matches of the face in the image, and displays them to the user. If the server does not detect a face it will return a message informing the client. The android OS does have face detection functionality, so detection could occur client-side if bandwidth usage is a concern. However, currently we just perform the detection server-side.

Assuming the server detected a face, the client receives back the top three recognition results. The user can then select one of these results to view their social connection to the recognized user. The names and faces of the users along the social path are shown.

4 .3.1 Client Interface

4.3.1.1 Login

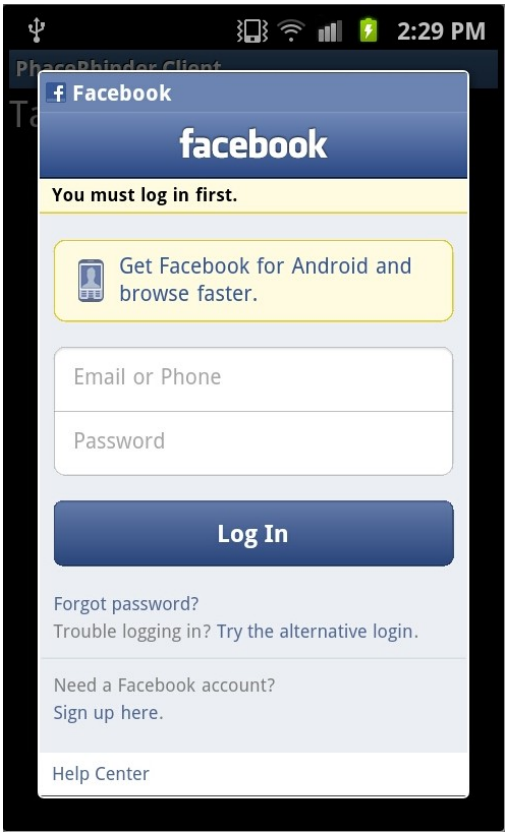
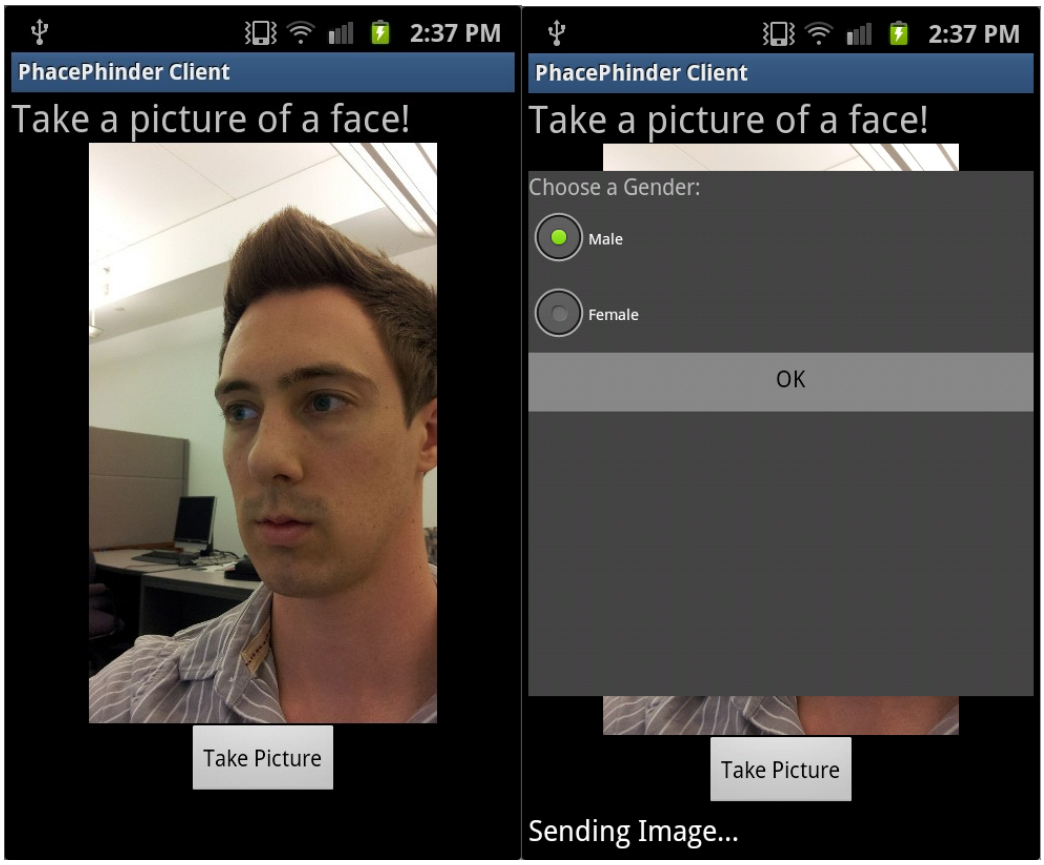


Figure 4.4 “Login Page” of the PhacePhinder client application

The system has to be aware of who the user is. It also needs to connect to Facebook in order to make sure permission to their information has been granted. Both these tasks can be done using the Facebook supported android API. Figure 4.4 shows the login page of the client application. This is the standard Facebook login for android devices.

4.3.1.2 Take Picture Page



A B

Figure 4.5 “Take Picture Page” and “State Gender Popup” of the PhacePhinder client application

The user must be able to take a picture of a person`s face with their phone. As a result, as soon as the user logs in the camera begins to stream to the interface. Once the user presses the “Take Picture” button, the application will capture the feed at that time, as can be seen in Figure 4.5 A. Immediately after taking the picture, the user will be asked to enter the gender of the person they just took a picture of, as shown in Figure 4.5 B.

4.3.1.3 View Results Page

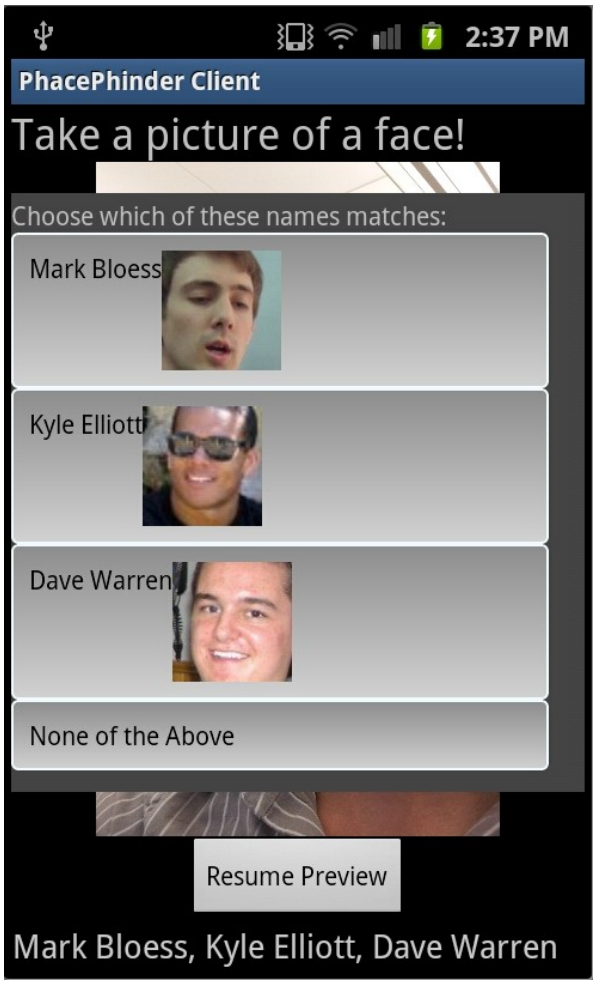


Figure 4.6 “View Results Page” of the PhacePhinder client application

Since we want the name and face of the top three recognition results we display a profile image beside each name, as can be seen in Figure 4.6. These profile images were generated automatically from the server. They are the detected face from an image from Facebook which had a tag to that user. Although this is a popup overtop of the Take Picture Page, it performs functionally as its own interface. The user can use the mobile

phone's "back" button to return to the Take Picture Page, or press the "None of the Above" button.

4.3.1.3 View Social Path Page

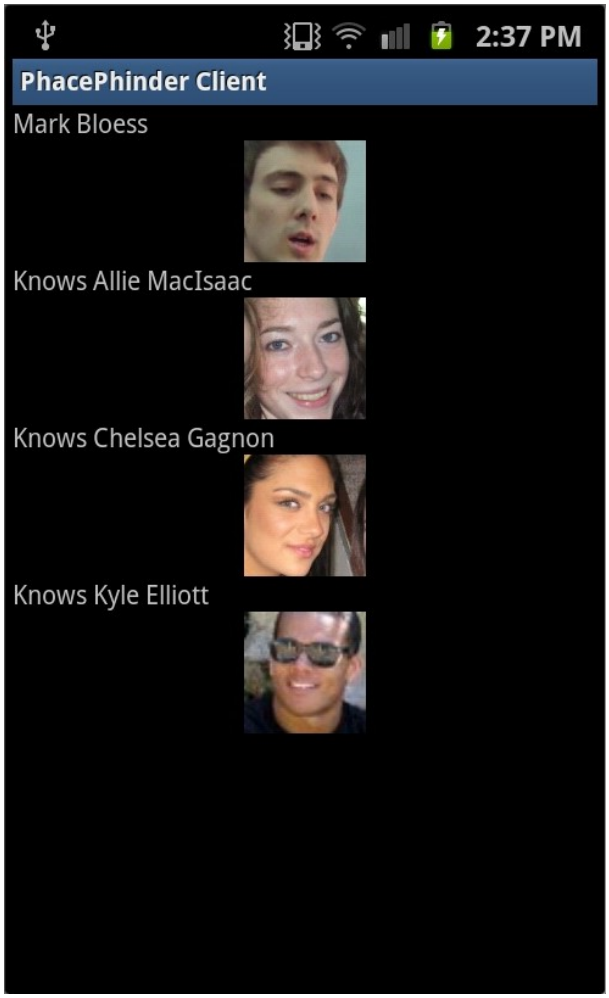


Figure 4.7 "View Social Connection Page" of the PhacePhinder client application

The user can select any of the suggested results to see their social path to the recognized user. The social path is indicated by the names and faces of the people that connect the user to the recognized person. This path is the shortest path on the social graph, in which the weights were determined from our fusion PLSA. Fig 4.7 shows the path from

the user to one of the recognition suggestions. By using the mobile phone's "back" button they can return to the View Results Page.

4.4 Server Side Implemented Functionality

The server side is responsible for the majority of the functionality associated with the application. Once a client connects to the server, it will retrieve the user's id and OAuth token. This information will be stored to the database. If the user already exists in the database it will update their information in order to keep the database up to date. Once it has this information it will proceed to gather all the images associated with that user and their friends. First it will retrieve the user ids of all the user's friends. It will then collect the album ids for these users. With this information we can collect the information for each picture in the album. This includes the tag information. The system now has enough information to start going through the images to detect faces.

First we check if an album has any tags associated with it. If there are no tags we will skip the album, since even if a face is detected in an image in that album, we will have no name or id to go with it. This is often the case with profile picture albums, where the images originally occurred in a separate album on the social network, so they are not tagged a second time. If an album does have tags associated with it, we will proceed to look at each picture. Similarly to how we handled albums, if an image does not have any tags we will not bother to detect faces in it. If an image does have tags, we will attempt to detect any faces in the image.

We use OpenCV, an open source computer vision library [11], to perform face detection. If no faces are detected in the image, we skip to the next image. If any faces are detected, we then put them through a second face detection. This is intended to reduce the occurrence of non-faces or noisy images. If any faces pass the second detection we now have to determine which, if any, of these tags belong to a detected face. Facebook gives an x, y coordinate in terms of a percentage of pixels in the image, where as OpenCV gives back the number of pixels. So a conversion from percentage of

pixels to number of pixels allows the two sets of coordinates to be comparable. Looking at the coordinates of the centre of the tag, we check to see if it occurs within the x, y plane of the detected face. This is based on the assumption that users typically click the centre of a face when they manually add a tag [8]. In the event of two tags occurring within a single detected face we must determine which is most likely to be correct. By taking an absolute distance of each tag from a specific detected face, using Equation 2 in section 3.2, we make the assumption that the smallest result indicates a matching tag and face pair.

Each time a new user is found, they are stored in the database. Much of a user's information is not public, and unless we have permission to view that information it is not accessible to our application. As a result, we often only get the user's id, name, and gender. In some cases gender is not even available, but only if the user never stated it in their profile. However, when we are looking at a user that is a friend of the primary user, then we have access to more information. We will store their id, name, gender, as well as their birth date, hometown, and current location if possible. Since we will always have the most information if the user is a friend of the original user, we will update the data in the database if the user entry already exists. Although this additional information is not of any use as of now, it may prove to be useful if the application were to evolve further.

A Facebook provided API is incorporated with our application to assist in issuing requests. All interaction with Facebook is done through their Graph API. Facebook responds in the form of JavaScript Object Notation (JSON). We parse the JSON responses and extract the data we need. Unfortunately, a Facebook application cannot retrieve the friend list of a user who has not given permission to the application to do so. In other words, we cannot view a user's friend list unless they are a primary user. However, we can see the friends that a user has in common with a primary user. So, in an attempt to construct a friend list which would normally be private, we check which friends a user has in common with each primary user, in order to determine as many

friends as possible. However, as a result, this gives us a friendship database that is extremely biased towards common friends. This issue will come into play later when we perform testing on our fusion PLSA model. However, as the number of primary users grows this issue will diminish.

This friend information, along with the picture occurrence information, is used to construct a weighted graph. Each edge represents a friendship or a picture occurrence. The edge will exist as long as one of these relations exits. In order to give weights to this graph, the system performs our fusion PLSA algorithm for all users in the database. Currently this functionality is done offline, so the entire graph is repopulated each time. This graph is stored to the database to be used when the client needs to view a social connection.

The server also responds to the client wanting to recognize a face. If the client side is sending an image it will first tell the server how big the file is and the gender of the face. The server will then receive the image from the client. The image is then converted from bytes to a bitmap and the server then tries to detect a face. In the event that no face is detected, the server will inform the client that no face was detected. If a face is detected, face recognition is attempted. We then get the top three most likely users and send their names and a link to their profile pictures back to the client.

Once the client has the top 3 recognition results, he can choose one to view the social connection from the current user to the recognized face. When this happens, the client will send the id of the selected recognized user. The server will then perform Dijkstra's algorithm on the weighted social graph, where the current user is the source and the recognized user is the sink. The server will then return back all users along the path, and a link to their profile images.

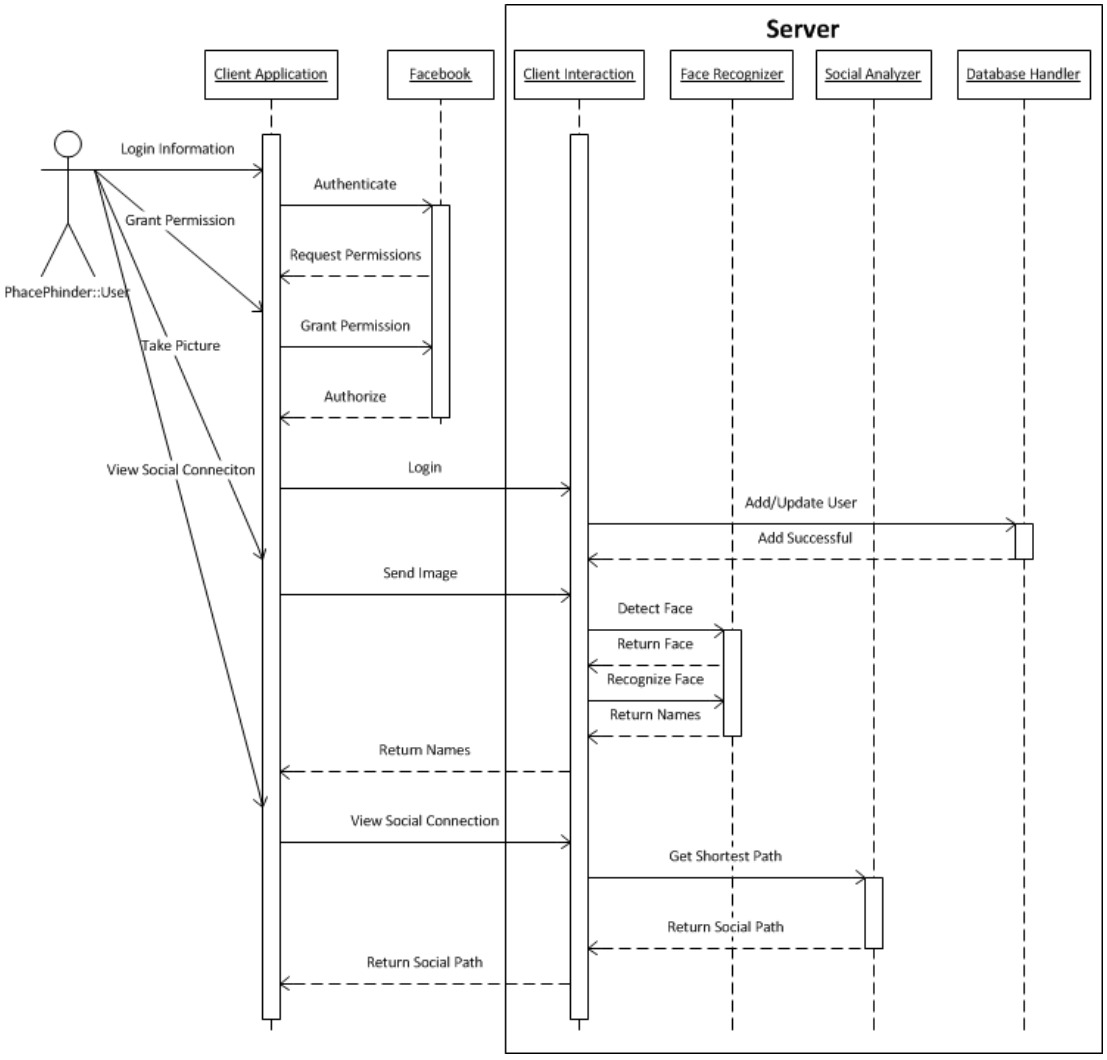


Figure 4.8 Interaction Diagram of a User Taking a Picture and Viewing a Social Path

The main functionalities of the server can be described by two interaction diagrams. Figure 4.8 illustrates, via interaction diagram, the sequence of events which the system undertakes when the user logs in, takes a picture, and views a social path. The client will originally have to authenticate the user login via Facebook. If the login is correct, it will then prompt the user to grant permission to the application. Upon granting permission, Facebook will then give the user id and OAuth Token to the client. The

client will then pass this information on to the server, which will handle the request and store the information to the database.

The user can now take a picture. The client will then send the image to the server. The server will handle the request and attempt to detect any faces in the image. Once a face is detected, it will attempt to recognize it. The recognizer will pass the names back to the client interaction component which will return the names via an http response. The client will display the recognition results to the user. The user can then click on a result to view a social path. This request is sent to the server, which will handle the request and pass it to the social analyzer. The social analyzer will determine the shortest path between the two requested users and return the result. The server will issue this result as an http response and the client will display it.

Figure 4.9 illustrates, via interaction diagram, the process in which the server gathers an image from the social network. The server first finds a user with unsearched photos. It then makes a request to Facebook for the user's albums. Upon receiving the albums it proceeds to determine if the album has any tags. If the album has tags it will then request the image information. Once it has the images, it will check that the image has a tag. If it has a tag it will then attempt to detect a face via the face recognizer. It will then match the detected faces to the tags. It will then make a request to Facebook for any additional information about the user. It will then add or update the user to the database via the database handler. It will then update the user's HMM and save the face image to the face database via the face recognizer. It will then save the image to the database and then save the image occurrence relation to the database via the database handler. This process will repeat for each face-tag pair in the image, then for each image in the album, then for each album for that user.

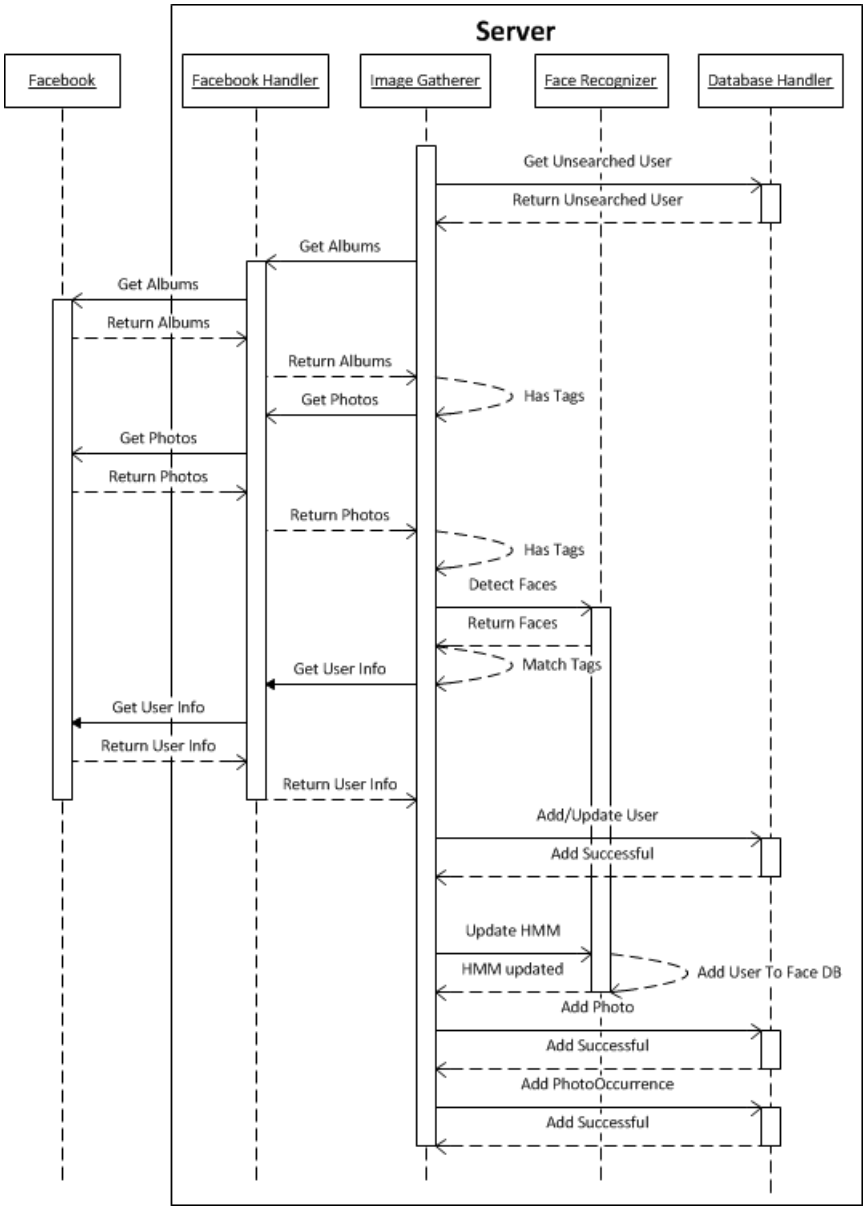


Figure 4.9 Interaction Diagram of the PhacePhinder Server Gathering an Image

Chapter 5

Experiments and Results

5.1 Image Gathering

Tables 5.1, 5.2 and 5.3 show the details of the database after 1, 2 and 3 primary users have granted permission.

Table 5.1 Statistics of collected data from Facebook from first primary user

Primary user's friend list size	185
New Individual user entries	1370
New Albums Searched	1792
New Pictures Gathered	3280
New Face Images Gathered	4178

Table 5.2 Statistics of collected data from Facebook from second primary user

Primary user's friends list size	478
New Individual user entries	3404
New Albums Searched	4235
New Pictures Gathered	7809
New Face Images Gathered	10425

Table 5.3 Statistics of collected data from Facebook from third primary user

Primary user's friends list size	95
New Individual user entries	164
New Albums Searched	375
New Pictures Gathered	549
New Face Images Gathered	414

The second primary user had a much larger friend list than the first primary user. Additionally, they had 48 friends in common, so a total of 430 new users' albums were

searched. However, the third primary user had a very small friend list and had 43 friends in common with the first two primary users. As a result they contributed very little growth to the database compared to the other primary users.

Table 5.4. Statistics of collected data from Facebook after all primary users

Number of Recorded Friendships	68748
Individual user entries	4938
Albums Searched	6402
Pictures Gathered	11638
Face Images Gathered	15017

This illustrates that with only a few primary users granting permission we gain access to a large number of users and faces, so growth of the database will occur very quickly. The constraints put on image gathering, detailed in section 3.2, are the reason why the number of photos and faces gathered is low. In reality, the number of images with tags and faces was more than double. However many were excluded because they did not pass the matching criteria.

In order to get access to information and images on Facebook, we require a user to grant permission to said data. This is a huge limitation, since the database will only grow if the user base grows. As well, a new user who has ties close to a previous user will not grant as large a contribution due to the fact that they are likely to have a large number of friends which are already in the database. An example of this can be seen from the third primary user. As a result of this, growth of the database will slow over time.

Figure 5.1 illustrates the distribution of face images and friendships per user. The number of face images per user drops much faster than the number of friends per user. This is important to note for our fusion PLSA model. We need to determine how much weight to give to the picture occurrence portion and the friendship portion. This chart

shows that there is much more friendship information than picture occurrence information, and this will be taken into consideration later in the thesis.

It is also important to note that a large number of user only have one friend or one image. As we stated in section 4.2, we can only know the friends of a non-primary user who are mutual friends of a primary user. So this is why we have a small number of friends for a large portion of users. In fact, there are a small number of users with no friends at all. This means they were tagged in an image we collected, but have no relation to any of the primary users. Additionally there are a few users who have no images. This is even more rare as it could only happen to a user who either is a primary user (highly unlikely) or a friend of a primary user who was never tagged in an image.

Another weakness of this method is that it relies on manual tags done by a Facebook user. There is no guarantee that the tag is actually correct. Our matching criteria described in section 3.2 helps to prevent non-faces being tagged, but it cannot prevent a misplaced tag. In other words if a face is tagged incorrectly there is no way of knowing. This will result in a face belonging to one user being added under a separate user, which can impact accuracy since we are now training based on the wrong face. One way to prevent this would be to perform face recognition on the tagged face to check if it matches the tag. This could only be done if a sufficient number of faces have been collected for that user previously. So the issue would still exist for the first faces being collected. As a result that solution could lead to more problems, since, if the first faces used for training are incorrect, we will end up rejecting more legitimate ones. As a result we just assume enough legitimate tags will be done to counteract any false tags, which are not very common since a Facebook user would be notified of a tag, and would likely remove it if it is incorrect.

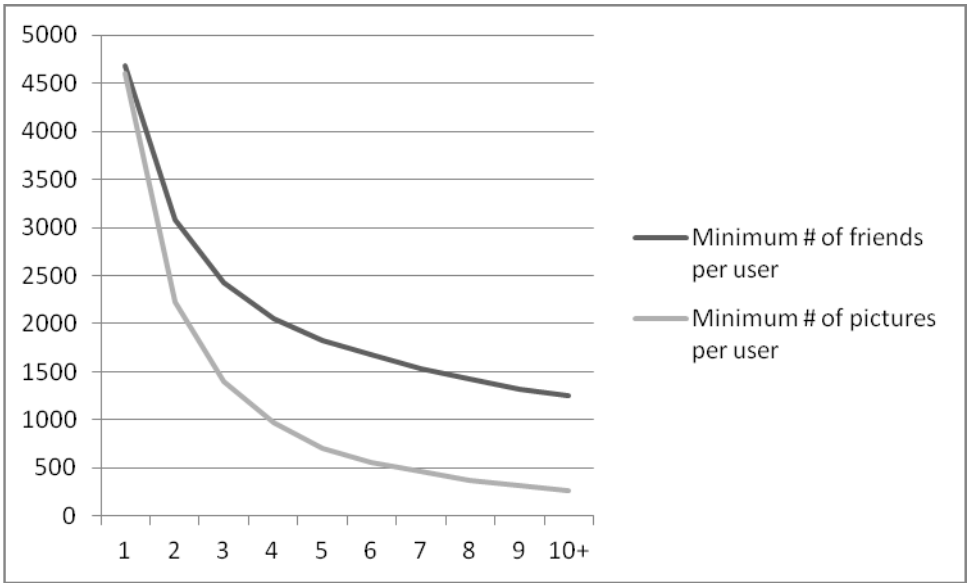


Figure 5.1 Friends and images per user comparison

5.2 Face Recognitions

As for the accuracy of the system, we ran two tests using a subset of collected images. The Hidden Markov Model [9] was used for face recognition. A subset of 50 individual user faces was removed at random from the total database. The only constraint being the user had to have at least 4 face images. The Hidden Markov Models were then retrained to never have included those images. We then attempted to recognize the removed faces among 1370 users, without stating the gender of the face. The results are as shown in Table 5.5.

Table 5.5 Subset test results without gender specified

# of hits in the first place	23
# of hits in the second place	4
# of hits in the third place	3
# of total hits in the top-3 list	30

This result shows that the hit-ratio for getting the correct name at the top of the list is 46%. However, we would also like to consider if the correct name was the second or third match. So if we consider the correct name being in the top three, our hit-ratio increases to 60%. The majority of correct identifications are the top match, at 76.67% of all correct identifications being the top suggestion.

Table 5.6 Subset test results with gender specified

# of hits in the first place	24
# of hits in the second place	4
# of hits in the third place	4
# of total hits in the top-3 list	32

We performed the same test again, however this time we assumed the gender of the face was known. This test is intended to show whether or not gender information improves the hit-ratio. Gender is the only contextual information we can guarantee we have for the majority of users, since it is public information on Facebook. The results can be seen in Table 5.6. With gender as a known variable, the hit-ratio for a top match is now 48% and the hit ratio for anywhere in the top 3 is 64%. From this we can make the assumption that if the system is not recognizing a face, it is not likely due to confusion in gender. However, there is a slight improvement; therefore it justifies its addition to the application. This supports the claim made by Akbari et al. [24].

5.3 Cross Validation Testing

5.3.1 Alpha Tuning

Alpha (α) is the value which gives more weight to either the friendship or picture occurrence side of the fusion PLSA algorithm. Finding the right balance is essential. In order to find the appropriate alpha value we performed a series of cross validation tests. Looking at all immediate friends of the first two primary users who have at least 20 friends, we come up with 490 test users. We then randomly removed 10 users from each

of their friend list. For these tests we used 100 hidden topics. We then retrieved a list of top 10 friend recommendations from each, and checked how many of the recommendations for the test user are in fact their friend. Since this is a top 10 list, we will measure the results in terms of Mean Reciprocal Rank (MRR) so that the order of the list is also taken into consideration. When looking at the whole set the results were as in Figure 5.2.

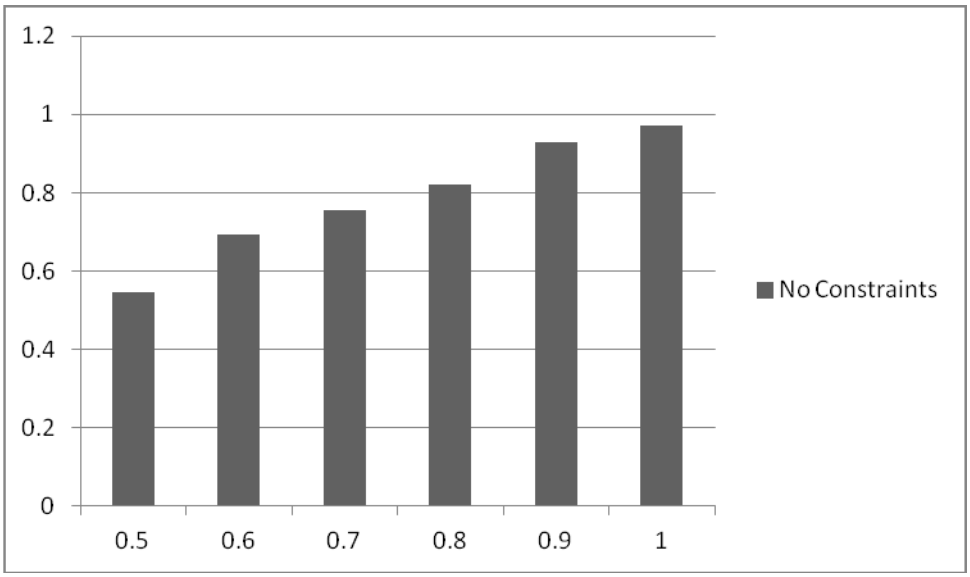


Figure 5.2 Average MRR for various alpha values on entire test set

Clearly we can see that the performance improves as the alpha value increases. In other words, the less we take image occurrence into consideration, the better the algorithm performs. Upon further inspection of the test set, we noticed that many users have very few image occurrences. This means that if image occurrence information is insufficient, it can have an adverse effect on the results of the fusion PLSA. This is why the PLSA which only considers friendship information (ie. alpha = 1) outperforms any of the fusion models. In order to get a better idea of the performance of the fusion

model we added further constraints to the test set. In particular, how many images the user occurs in. The results are show in Figure 5.3.

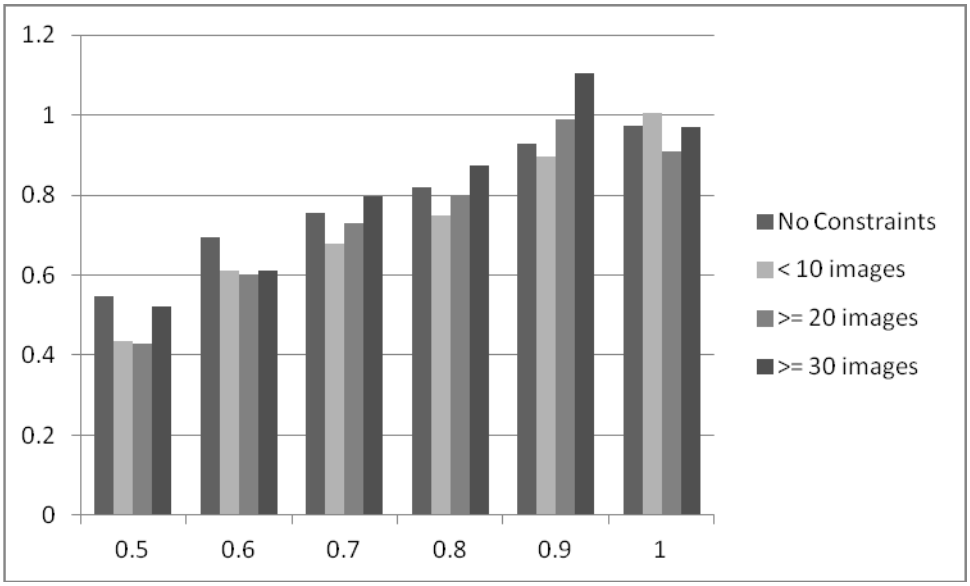


Figure 5.3 Average MRR for various alphas with added constraints

We left the values with no constraints to assist with the comparison. The results when only considering users with less than 10 image occurrences (i.e < 10 images) are similar, in relation to each alpha value, as to those with no constraints. Again the PLSA with alpha = 1 outperformed all other alpha values. However, once we consider users with 20 or more images (i.e. >= 20 images) the fusion model with alpha = 0.9 outperforms the PLSA with alpha = 1. These results continue to be true, with an even larger difference, once only users with 30 or more images (i.e. >= 30 images) are considered. This shows that once there is sufficient image occurrence information, the fusion model for PSLA does assist in friend recommendation. However, it is important to note that it will decrease the performance if the image occurrence information is too low, so in that case it is better to simply use a friendship occurrence PLSA (i.e. alpha = 1).

5.3.2 Hidden Topic Tuning

The second variable which we can control in the fusion PLSA is the number of hidden topics (i.e. *z* in the fusion PLSA algorithm in section 3.4). We ran the test on four different numbers of hidden topics. Using the same test set as for the alpha tuning, we performed the PLSA with alpha = 0.9. We show the results for users with 20 or more images (i.e. >=20 images) and 30 or more images (i.e. >= 30 images), as these were the best results in the alpha tuning tests. The results are as in Figure 5.4.

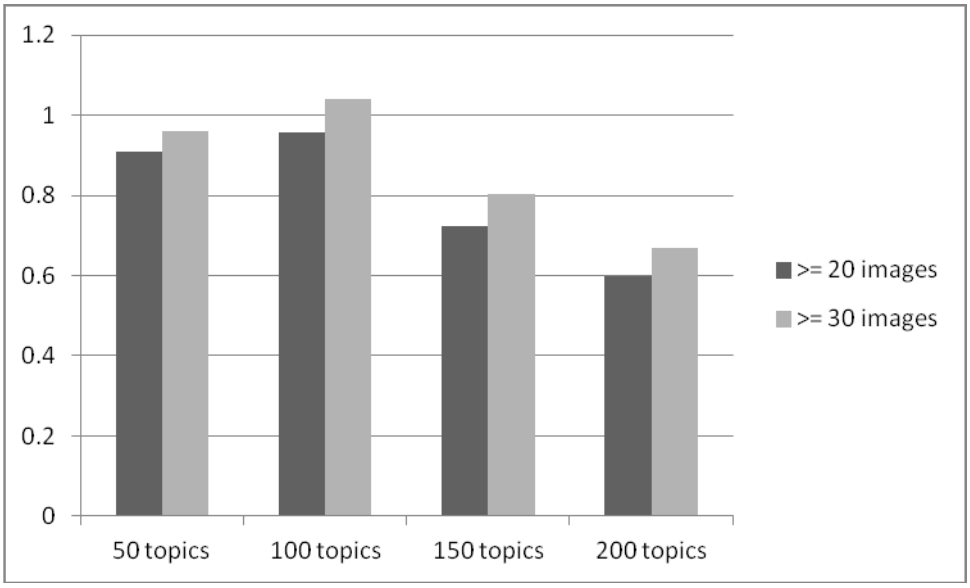


Figure 5.4 Average MRR for a various number of hidden topics.

There is an incline between 50 and 100 topics, however there is a sharp decline between 100 and 150 topics. The decline continues between 150 topics and 200 topics. Based on these results 100 hidden topics provides the best performance. As a result, this is the number of hidden topics we will use for all further testing.

5.3.3 Multiple Algorithm Comparison

5.3.3.1 Benchmark Algorithms

In order to validate the performance of our fusion PLSA model, we must compare it to other algorithms in the field of content recommendation. This section will describe these algorithms and how they are a relevant comparison to ours.

KATZ Algorithm

Katz [27] was originally intended for the use of determining the social significance of a person. The sample scenario considers a group of people who are asked to vote on which people they like within the group. Instead of just looking at who got the most votes, you also want to consider who is voting. Someone who received a lot of votes will have more of an impact with their own vote. In our case we are dealing with friendships instead of votes, but the principle is the same. The value of the relation between any two users is defined by the sum of the number of times a path exists between the two users at a certain level of connections (or certain degree of separation), multiplied by a set predictor value. The Katz algorithm can be defined as [35]:

$$score(x, y) = \sum_{\ell=1}^{\infty} \beta^{\ell} \bullet | \, paths_{x,y}^{(\ell)} \, | \qquad (11)$$

Where *x* and *y* represent two users, β is the predictor, ℓ is the level of depth of the connection, and $paths_{x,y}^{(\ell)}$ is the number of paths between user *x* and *y* at a depth of ℓ. Liben-Nowell et al. [35] shows the algorithm performed the best when β=0.005. This is the predictor value used when we run this algorithm to compare to our fusion PLSA model.

PAGERANK Algorithm

PageRank [23] was created by the founders of Google Inc. and was used in their search engine for ranking the significance of web pages on the World Wide Web. We can consider the internet as a graph, in which each web page is a vector and a link from one page to another represents a directional edge. Simplified Page rank [23] can be defined by the following equation:

$$R(u) = c\sum_{v \in Bu} \frac{R(v)}{N(v)} \tag{12}$$

Where R is the set of PageRanks, where u would be the page we are getting the rank of, so $R(u)$ and $R(v)$ is the PageRank for page u and page v respectively. B_u is the set of pages which link to u. $N(v)$ is the number of links coming from page v . c is a normalization factor.

Traditional PageRank could be used to rank each user's significance on the entire social network, but this is not useful to us. However, Brin et al. also propose a Personalized PageRank. The idea behind this is that a user could have PageRank values tailored to them, based on something such as the bookmarks in their web browser. This is done by adding a bias to the random resets. The adapted algorithm can be defined by the following equation [23]:

$$R'(u) = c\sum_{v \in Bu} \frac{R'(v)}{N(v)} + cE(u) \tag{13}$$

Where $R'(u)$ is the PageRank vector tailored specifically to a user where $||R'||_1 = 1$. In other words, each user would have their own tailored R' vector. $E(u)$ is to be a vector over the web pages which corresponds to a source of rank. This would correspond to a decay factor. Brin et al. use $||E||_1 = 0.15$ for their experiments, and Liben-Nowell et al. [35] also show that this value yields good results. As a result, we use 0.15 as the decay

factor when we compare PageRank to our fusion PLSA model. In the case of our scenario, each page would be considered another user in the social network, where outbound links and inbound links are always equal as they represent a bidirectional friendship.

5.3.3.2 Comparison Results

For these comparisons we used the same test set as for the alpha and hidden topic tuning. We ran our fusion PLSA with the alpha value of 0.9 and 1, and 100 hidden topics, as those were the best performing values in the tuning process. We also ran the PageRank and Katz algorithms for comparison.

Table 5.7 Comparison of Mean Reciprocal Rank of top-10 results

Algorithm	No constraints	< 10	≥ 20	≥ 30
PageRank	0.7727	0.7371	0.8259	0.8563
Katz	0.8604	0.8345	0.8890	0.9266
Fusion PLSA ($\alpha = 0.9$)	0.9280	0.8960	0.9895	1.1035
PLSA ($\alpha = 1$)	0.9724	1.0064	0.9079	0.9704

Table 5.7 shows the MRR results. When looking at all 490 test users (i.e., no constraints), we saw that the fusion PLSA algorithm outperformed both the PageRank algorithm and Katz algorithm. However, the PLSA based solely on friendship information (i.e. $\alpha = 1$) outperformed the fusion PLSA. This result is affected by the fact that many test users have very few images, as we saw earlier while tuning the algorithm. Accordingly, the average MRR results are lower when images are taken into consideration. As we did in the tuning phase, we again added the image constraints to the test set. When the test set was limited to users appearing in less than 10 images (i.e. < 10), the results were similar to the total dataset, with PLSA excelling. However, when we considered test users having at least 20 images (i.e. ≥ 20), the fusion PLSA outperformed the others. When the limit of at least 30 images was added (i.e. ≥ 30),

each algorithm's performance improved, but the fusion PLSA had the most drastic improvement. There was an increase of 0.114 for the fusion PLSA, while the next largest increase was 0.0625 for the non-fusion PLSA. This shows that with sufficient image information, the fusion PLSA model excels well above the benchmark algorithms.

5.4 User Study

In order to validate our system's recommendation functionality, we performed several tests involving users. These tests involve having users rate friendship and picture recommendations, as well as stating if the social path determined via a weighted graph derived from our fusion PLSA model is superior to simply finding the shortest path on an equal edge graph.

5.4.1 Social Path Recommendation User Test

First we wished to test the social path recommendation. In order to do so, we got 10 volunteers that existed in the database. We then determined the path to 50 random non-friends of each user. We determined the path based on a graph with weights based on our fusion PLSA algorithm, and a graph where all weights are equal. The results are shown in table 5.8.

Table 5.8 User Feedback on Social Path Comparison

Subject	Different Paths	Fusion PLSA	Equal Edge
U 1	7	5	0
U 2	7	3	2
U 3	28	5	7
U 4	9	6	0
U 5	33	31	1
U 6	32	24	5
U 7	32	6	0
U 8	20	15	4
U 9	29	10	4
U 10	19	18	1

The different paths column indicates how many of the paths from the original 50 actually differed between the two algorithms. The PLSA column indicates how many times the test subject chose the PLSA path as the better result. The Equal Edge column indicates the number of times the test subject chose the result based on a graph where all edges are equal. From these results we can see the fusion PLSA more often determined a more meaningful path than a graph with equal edges. However, many times there was no difference in the paths between the two algorithms. This is a result of either there being only one path that exists between two users, or that the best possible path just happened to be the first shortest path found. Only one path existing between users would become less frequent as the database grows and the interconnectivity of users increases. In some cases we can see that the path with equal edges was chosen more often than the fusion PLSA path. However, many times it was the same two paths that were being compared. Although the final connection differed, the intermediary connections were the same. So the user was actually making the same observation several times. Several of the paths, although different, users claimed there to be no social difference. By this they mean that both paths are equally significant. This is because although the system may have determined a stronger relationship, the user does not consider one friend greater than the other. Aside from this, for most cases if a better path existed, the fusion PLSA graph would show it, with the ratio of correct choices to incorrect choices being almost 5:1.

5.4.2 Friend Recommendation User Test

Next we wished the test content recommendation. Again we chose 10 volunteers from the database and gave them each a top 10 friend recommendation list. They were asked to give the recommendations a score on a five point Likert scale, where the ranks are:
1 – very poor, 2 – poor, 3 – neutral, 4 – good, 5 – very good

A user typically would give a rank of 1 if they had no idea who the person was, and in a few cases when they particularly disliked a person despite knowing them very well.

We also gave them the top 10 list based on mutual friends, PageRank, and Katz. The results are in tables 5.9 and 5.10.

Table 5.9 Average User Feedback on Friend Recommendation

Subject	Average Recommendations From PLSA	Average Recommendations From Mutual Friends	Average Recommendations from PageRank	Average Recommendations from Katz
U 1	4.3	3.8	3.9	3.4
U 2	2.7	3.5	3.2	3.8
U 3	2.1	3.2	1.9	3
U 4	3.1	2.2	2.3	2.8
U 5	1.8	1.7	1.7	1.6
U 6	2.8	1.8	3.0	2.9
U 7	2.1	2.3	2.2	2.1
U 8	3.2	3.2	3	3.4
U 9	2.2	3.5	3.1	3.5
U 10	2.3	2.3	1.6	2.3

Table 5.10 Normalized Discounted Cumulative Gain Feedback on Friend Recommendation

Subject	nDCG Recommendations From PLSA	nDCG Recommendations From Mutual Friends	nDCG Recommendations from PageRank	nDCG Recommendations from Katz
U 1	0.983	0.930	0.947	0.879
U 2	0.840	0.958	1	0.957
U 3	0.726	0.825	0.848	0.824
U 4	0.857	0.833	0.950	0.895
U 5	0.889	0.956	0.906	0.919
U 6	0.960	0.787	0.862	0.708
U 7	0.896	0.884	0.927	0.894
U 8	0.923	0.973	0.977	0.967
U 9	0.928	0.880	0.889	0.908
U 10	0.964	0.886	0.901	0.907

Table 5.9 shows the average rating the users gave for each algorithm. Table 5.10 shows the normalized Discounted Cumulative Gain (nDCG), in order to help illustrate

the quality of the ranking for the algorithms. For reference, an nDCG value in the 0.7 to 0.8 range is considered poor, while anything over 0.9 would be considered good. However, a series of poor recommendations can still have a good nDCG score, as long as the order in which the ratings are in is descending from best to worst.

Although this is very limited study, it does help to support our offline study, indicating that our fusion PLSA model can perform alongside the benchmark algorithms. Another important note is that users 1, 2, and 3 are primary users, while the others are not. We had expected them to have very good results; however this was only the case for user 1. This assumption was based on the fact that a primary user would have as much information collected about them as possible, with the exception of full friend lists of their friends, as this would require all of the primary user's friends to also be primary users. Regardless, with more information we can make a better recommendation. For example, user 8 was chosen for this study because they had a lot of friends in common with primary user 2. The hypothesis was that this user would get good recommendations since we have a lot of information on them. This did turn out to be true, even though the recommendations for user 2 were not very good.

5.4.3 Image Recommendation User Test

Next we wished to test the image recommendation potential for our fusion PLSA model. In order to do so we gave 10 volunteers a list of the top 10 recommended photos. They were asked to rate the images, based on a five score Likert scale, on how much interest they had in an image. The rating was as follows:

1 – absolutely no interest, 2 – very little interest, 3 – some interest, 4 – interesting,
5 – very interesting

In this case we do not have a comparison to other algorithms, instead we just show the user results for each image. Table 5.11 shows their results.

Table 5.11 User Feedback on Image Recommendation

	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
Image Ratings	2	1	1	5	5	5	5	1	1	5
	2	1	4	5	4	4	5	5	5	5
	5	3	4	5	4	3	2	1	5	5
	2	3	5	5	4	4	5	1	2	5
	2	5	1	3	4	4	5	5	2	5
	3	5	2	5	4	4	5	5	1	4
	3	1	5	4	4	5	3	5	1	5
	3	5	5	3	5	3	5	5	4	5
	1	3	5	4	3	1	1	5	4	5
	2	5	4	4	3	5	5	5	4	5
Average	2.5	3.2	3.6	4.3	4	3.8	4.1	3.8	2.9	4.9
nDCG	0.845	0.732	0.801	0.993	0.984	0.961	0.966	0.785	0.802	0.997

We can see the test subjects gave many high ratings. Almost all users had an average rating score of 3 or over, with the exception of user 1 and 9. We calculated nDCG in order to give a representation of the correctness of the ranking. Each user with a 1 rating at the top of their list scored a relatively low nDCG, and rightfully so. The top image should be the most relevant, and scoring ideally a 5, however this was only true for half the users.

Users tended to consider a photo of interest if it had them in it. This would mean that an algorithm which just chose images with that user in it could have outperformed our model. However, many users also gave images a high rating if their friends were in it. These results are more interesting, since a social network would typically have brought an image to your attention when you were tagged in it. These results only show that our model can find images of people that the user wants to look at. It would not bring any images to their attentions that do not contain people, so for example an image of a sunset would go unseen. Facebook does not offer enough information to warrant a proper form of content recommendation for images, such as context tags. So, similar results could likely be achieved through selecting images of users based on a user's friendships. The stronger the friendship, the more the person would be interested in the

image. Regardless, in terms of finding images of interesting people, the algorithm typically performed well.

5.4.4 User Test Conclusion

It is hard to draw many conclusions from these sets of user tests. Some users liked their recommendations, while others did not. The actual recommendations are not always to blame. For friend recommendations, some users would rate people low even if the recommendation being given was someone they knew reasonably well. They would state that even though they know the person, for example they may have gone to high school together, does not mean they would want to be friends with them. For other users, going to high school with someone was their reason for rating them high. This makes it difficult to make a statement on which method worked best based on these results. Instead we can say that the fusion PLSA model did determine some good recommendations, and so did each of the other algorithms. There were many recommendations common among the algorithms, but there were also other highly rated results that were not common among the algorithms. So we cannot state that our fusion model should be chosen over another, but we can say at least that it can be used if the good recommendations from another algorithm are exhausted and a new list is needed.
A much more extensive user study would be needed to make any kind of definitive statement based on these results.

Chapter 6

Conclusion and Future Work

In this thesis we have proposed a method of collecting information and images from Facebook. We use this information to build a face recognition database. We then proposed and develop a system in which it can be used by a mobile device to detect people in every day scenarios. We also proposed using the contextual information from Facebook and using a fusion PLSA algorithm to build a weighted graph which can be used to find relations between a user and a recognized face, as well as be used for friend and image recommendation. A series of offline and user testing helps to validate these results.

As more primary users grant permission to the PhacePhinder application, the database will grow rapidly. A larger number of users in the database will likely reduce the accuracy of the face recognition; however we will gain additional information on the users in the database. As a result we can use more context than just gender to reduce our search space. For example, “current location” is a piece of information which users can state on Facebook. If we get the current location of most users in the database, then we can use the GPS location of the mobile phone to indicate where the picture is taken to reduce the possible results to users in that geographical area.

As for finding connections, using Dijktsra’s algorithm could get costly once the database grows extremely large. Instead, a heuristic algorithm should be implemented to improve finding a link between two users. As well, if face recognition accuracy becomes too low, a collaboration of face recognition methods could be used, such as is done by Choi [13]. Additionally, since the fusion PLSA model is currently an offline

algorithm, it would need to be adapted to an online algorithm in order for the system to be deployable in a real world scenario.

Offline testing showed that our fusion PLSA model outperformed other benchmark algorithms. However, our user testing was inconclusive. Extensive user testing would be required to make a conclusive statement on the practicality of our fusion model over other existing models. In particular, testing on more primary users would give an even stronger indication of the algorithm's performance, not only because a primary user has nearly complete data about themselves, but because more primary users would mean and even larger, more connected database.

As the project evolves, it can eventually be adapted to a head mounted display (HMD) equipped with a camera, such as is proposed by Kurze [14]. With the announcement of Project Glass from Google [15], in which they showcase their soon to be released augmented reality wearable glasses, every day face recognition could be common place in the not too distant future with the help of applications such as PhacePhinder.

Bibliography

1. Facebook Statistics [online] Available: http://newsroom.fb.com/content/default.aspx?NewsAreaId=21 (Accessed on 27/3/2012)

2. D. Cohn, T. Hofmann "The Missing Link - A Probabilistic Model of Document Content and Hypertext Connectivity", In Advances in Neural Information Processing Systems 13, pp 430 - 433(2001).

3. Z. Stone, T. Zickler, T. Darrell "Autotagging Facebook: Social network context improves photo annotation." Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp 1-8 (2008)

4. N. Mavridis, W. Kazmi, P. Toulis "Friends with faces how social networks can enhance face recognition and vice versa." Computational Social Network Analysis, pp 453-482 (2010)

5. M. Davis, M. Smith, J. Canny, N. Good, S. King, R. Janakiraman "Towards Context-Aware Face Recognition", Proceedings of the 13th annual ACM international conference on Multimedia, pp 483-486 (2005)

6. H.-N. Kim, A. El Saddik J.-G. Jung, "Leveraging personal photos to inferring friendships in social network services." Expert Systems with Applications vol. 39, no. 8, pp 6955-6966 (2012)

7. Z. Stone, T. Zickler, T. Darrell "Toward large-scale face recognition using social network context." Proceedings of the IEEE vol. 98, no.8, pp1408–1415 (2010)

8. B. C. Becker, E. G. Ortiz "Evaluation of face recognition techniques for application to Facebook." Proceedings of the 8th IEEE International Conference on Automatic Face and Gesture Recognition, (2008), Septemper 17-19; Amsterdam Netherlands

9. A.V. Nefian "Face detection and recognition using hidden Markov models." Proceedings of the 1998 International Conference on Image Processing, vol. 1 pp141 - 145 (1998)

10. E. Hammer-Lahav, D. Recordon, D. Hardt "The OAuth 2.0 Authorization Protocol", Network Working Group Internet-Draft, January (2011) [online] Available: http://tools.ietf.org/pdf/draft-ietf-oauth-v2-12.pdf (Accessed on 12/4/2012)

11. OpenCV: Open Source Computer Vision Library, [online] Available: http://opencv.willowgarage.com/wiki/ (Accessed on 27/3/2012)

12. M. Barbehenn “A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices.” IEEE Transactions on Computers vol. 47,no. 2, pp 263 (1998)

13. J. Y. Choi “Collaborative face recognition for improved face annotation in personal photo collections shared on online social networks”, IEEE Transactions on Multimedia vol. 13, no. 1, pp 14-28 (2011)

14. M. Kurze, A. Roselius “Smart glasses linking real live and social network's contacts by face recognition.” Proceedings of the 2nd Augmented Human International Conference, article 13 (2011),

15. Project Glass [online] Available: https://plus.google.com/u/0/111626127367496192147/posts (Accessed on 11/4/2012)

16. S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, R. Harshman “Using latent semantic analysis to improve access to textual information.” Proceedings of the SIGCHI conference on Human factors in computing systems, 281-285 (1988)

17. T. Hofmann “Probabilistic latent semantic analysis.” Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, 282-296 (1999)

18. L. E. Baum, T. Petrie "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". The Annals of Mathematical Statistics 37 (6): 1554–1563 (1966)

19. J. Baker "The DRAGON system--An overview". IEEE Transactions on Acoustics, Speech, and Signal Processing 23: 24–29 (1975)

20. S. Feng, R. Manmatha “A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books”, Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries, 109-118 (2006)

21. M. Bishop, E. Thompson "Maximum Likelihood Alignment of DNA Sequences". Journal of Molecular Biology 190 (2): 159–165 (1986)

22. G. Jeh, J. Widom “Scaling personalized Web search”, In: Proceedings of the 12th International Conference on World Wide Web, pp. 271–279 (2003)

23. S. Brin, L Page, R. Motwami, T. Winograd “The PageRank citation ranking: bringing order to the Web”, Stanford University, Technical Report (1999)

24. R. Akbari “Performance enhancement of PCA-based face recognition system via gender classification method”, Machine Vision and Image Processing (MVIP), 2010 6th Iranian, pp 1 – 6 (2010)

25. L. Sirovitch, M. Kirby “Low-Dimensional Procedure for the Characterization of Human Faces,” J. Optical Soc. of Am. A, vol. 2, pp. 519-524, (1987)

26. M. Turk, A. Pentland “Eigenfaces for Recognition,” J. Cognitive Neuroscience, vol. 3, no. 1, (1991)

27. L. Katz “A new status index derived from sociometric analysis.” Psychometrika, 18(1), pp. 39–43. (1953)

28. P.N. Belhumeur “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection” Pattern Analysis and Machine Intelligence, vol. 19 (7) pp 711-720 (1997)

29. T. D. Pessemier, T. Deryckere, L. Martens “Context aware recommendations for user-generated content on a social network site” Proceedings of the seventh european conference on European interactive television conference, pp 133-136 (2009)

30. O. Phelan, K. McCarthy, M. Bennett, B. Smyth “Terms of a feather: content-based news recommendation and discovery using twitter” Proceedings of the 33rd European conference on Advances in information retrieval, pp 448-459 (2011)

31. M. B. Dias, D. Locher, M. Li, W. El-Deredy, P.J.G. Lisboa “The value of personalised recommender systems to e-business: a case study” Proceedings of the 2008 ACM conference on Recommender systems, pp 291-294 (2008)

32. N. Du, B. Wu, X. Pei, B. Wang, L. Xu "Community detection in large-scale social networks” Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. pp 16-25 (2007)

33. P. Wu, W. Ding, Z. Mao, D. Tretter “Close & Closer: Discover social relationship from photo collections” Multimedia and Expo, 2009. ICME 2009. IEEE International Conference, pp 1652-1655 (2009)

34. Y. Zhang "Identifying Key Users for Targeted Marketing by Mining Online Social Network" Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference, pp 644 - 649 (2010)

35. D. Liben-Nowell, J. Kleinberg "The Link-Prediction Problem for Social Networks" Proceedings of the twelfth international conference on Information and knowledge management, pp 556-559 (2003)

36. A.D.I. Kramer "The Spread of Emotion via Facebook" Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems, pp 767-770 (2012)

37. M. E. Morris, S. Consolvo, S. Munson, K. Patrick, J. Tsai, A.D.I. Kramer "Facebook for health: opportunities and challenges for driving behavior change" Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems, pp 443-446 (2011)

38. A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, H. Liu "Data warehousing and analytics infrastructure at facebook" Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp 1012-1020 (2010)

39. Shannon M. Oltmann "*Katz* out of the bag: the broader privacy ramifications of using facebook" Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem, vol 47 article 76 (2010)

40. C. Biancalana, F. Gasparetti, A. Micarelli, A. Miola, G. Sansonetti "Context-aware movie recommendation based on signal processing and machine learning" Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation, pp 5-10 (2011)

41. J. Wang, B. Sarwar, N. Sundaresan "Utilizing related products for post-purchase recommendation in e-commerce" Proceedings of the fifth ACM conference on Recommender systems pp 329-332 (2011)

42. T. Iwata, K. Saito, T. Yamada "Recommendation method for extending subscription periods" Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 574-579 (2006)

43. Z. Wu, S. Jiang, Q. Huang "Friend recommendation according to appearances on photos" Proceedings of the 17th ACM international conference on Multimedia, pp 987-988 (2009)

44. Y. Zheng, L. Zhang, Z. Ma, X. Xie, W.-Y. Ma “Recommending friends and locations based on individual location history” ACM Transactions on the Web vol 5 (1), article 5 (2011)

45. S. Krishna, G. Little, J. Black, S. Panchanathan “A wearable face recognition system for individuals with visual impairments” Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility pp 106-113 (2005)

46. B. J. Balas, P. Sinha "Region-based representations for face recognition” Transactions on Applied Perception pp 354-375 (2006)

47. S. Berretti, A.D. Bimbo, P. Pala "Distinguishing Facial Features for Ethnicity-Based 3D Face Recognition” ACM Transactions on Intelligent Systems and Technology vol 3 (3) article 45 (2012)

48. X. Wang “Face Photo-Sketch Synthesis and Recognition” IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 31 (11), pp 1955-1967 (2009)

49. Y. Ijiri, M. Sakuragi, S. Lao "Security Management for Mobile Devices by Face Recognition” Proceedings of the 7th International Conference on Mobile Data Management, pp 49 (2006)

50. T. Hofmann “Collaborative filtering via gaussian probabilistic latent semantic analysis” Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp 259-266 (2003)

51. H. Wu, Y. Wang, X. Cheng “Incremental probabilistic latent semantic analysis for automatic question recommendation” Proceedings of the 2008 ACM conference on Recommender systems, pp 99-106 (2008)

Appendix

A1 Use Cases

Client Side Use Cases

Login	
Prerequisite	Application is open
Steps:	1 User enters Facebook login information
	2 Application prompts user to grant permission
	3 Camera preview is displayed
Alternate: user has granted permission already	2a continues directly to step 3
Alternate: Information is incorrect	2b User is informed that login information is incorrect
	3b Return to step 1

Recognize Face	
Prerequisite	Completed Login use case
Steps:	1 User presses the “Take Picture” button
	2 The user is prompted to enter a gender
	3 The User selects a gender
	4 The image and gender are sent to the server
	5 The top three recognition results are displayed on the screen
Alternate: No face detected	5a The system displays a message that no face was detected

View Social Path	
Prerequisite	Completed Recognize Face use case
Steps:	1 User selects one of the returned users
	2 The system displays the series of social connections between the user and the recognized face
Alternate: No path exists	2a The system displays a message that no social connection exists between the user and the recognized face.

Server Side Use Cases

User Connect	
Prerequisite	Client Side completed Login use case
Steps:	1 Create new session for the user
	2 Record user information and OAuth token for the user from Facebook

Receive Sent Image	
Prerequisite	Client Side sends an image
Steps:	1 Receive the image from the client
	2 Receive the gender from the client
	3 Attempt to detect a face from the image
	4 Perform Hidden Markov Model face recognition
	5 Send a response of the top 3 returned results that match the appropriate gender
Alternate path: No face was detected	4a Send a response that no face was detected

Retrieve Social Connection	
Prerequisite	Client Side clicks on a returned recognition result
Steps:	1 Receive the user id of the recognized user from the client
	2 Perform Dijkstra's algorithm on the social graph from the current user to the chosen user
	3 Return the user names and profile images of the users which consist of the path in the graph
Alternate path: No path found	3a Return a message that no path exists between the two users

Collect Images	
Prerequisite	Completed User Connect use case
Steps:	1 Use OAuth token to access a user's Facebook profile
	2 Retrieve their friend list
	3 Record friend information
	4 Perform face detection on friends' photo albums
	5 Match detected faces with the user tags
	6 Store the face image and user information

Perform PLSA	
Prerequisite	Completed Collect Images use case
Steps:	1 Prepare friendship and photo occurrence information
	2 Perform fusion PLSA on the data
	3 Store results

Build Social Graph	
Prerequisite	Completed Perform PSLA use case
Steps:	1 Check if a friendship or photo co-occurrence exists between each pair of users
	2 Store the inverse of the PLSA result for each existing edge
Alternate Path: No connection exists	2a Move on to next user pair