MateFinder - The Next Generation of Dating Recommendation System

Chuqiao Ren[*] Jinyang Yu[†] Lyujia Zhang[*]
Computer Science[*], Electrical Engineering[†]
Columbia University
[*cr2826*, jy2803, lz2467]@*columbia.edu*

***Abstract*— Dating service is a growing demand for American market. Survey has shown over 40 millions people have used dating services and this number is growing exponentially with the advent of social media and Internet service. Currently there are lots of off-the-shelf dating service products, such as Tinder, OKCupid, etc. While these products provide great services with large user population, they have not yet taken advantages of the modern techniques in machine learning and artificial intelligent technology. Users can often get frustrated when their searches render random or inattractive results. In this paper, we propose a novel system, *MateFinder*, which can utilize multiple non-trivial methods in machine learning and deep learning, to find the best fit among the large user population, based on user's questionnaires as well as facial image. The questionnaires will allocate the proper user cluster and the system will narrow down several best fit users according to the facial similarity. This system shall greatly enhance user's user experience by effectively recommending a matching user profile to the designated user. Developing this product is still an ongoing mission. We are going to extend the work to mobile platform and improve its robustness by collecting non-frontal facial images.**

Keywords- ***dating, recommender system, web application, k-means clustering, unsupervised learning, computer vision***

I. INTRODUCTION

Dating service is a growing demand for American market in recent years. Survey[1] has shown there are over 50 million single adults and over 40 million people have used at least one dating service. There have been lots of online dating service app developed recently and some of them have a large user population, such as Tinder and OKCupid. Studies have shown many successful dating applications are mostly contributed by unique marketing strategies and intuitive user interface. For example, Tinder achieved a great success by its eye-catching strategy and OKCupid gains profits because of the proper advertisements. However, users can often get frustrating search results because most of the dating services have not utilized state-of-the-art machine learning techniques. With the modern development in artificial intelligent, it is viable to develop a dating service which can provide customized recommendation. In this work, we proposed a system, *MateFinder*, which can find the best fit among a large user

population based on several user's input. Basic questionnaires, such as expected weight, zodiac, and etc, will be asked at the beginning to help the system allocate the proper user cluster, then the system will require user's profile photo as a secondary input. Research[2][3] has shown the facial outlook of a couple surprisingly converges thus we believe a face similarity matching can serve as a secondary filter to narrow down the search result once more.

In this work, we proposed *MateFinder*, which serves three main contributions:

1. Incorporated modern machine learning techniques into online dating services.
2. Uniquely proposed a facial similarity matching mechanism for dating recommendation.
3. Implemented an end-to-end, fully executable web application which achieves described functionality with user-friendly interface and real-time matching computation.

In the following sections, we will discuss multiple aspects of this work. In Section II, several machine learning techniques and similar work on online dating services will be introduced. A system architecture and core algorithm description will be provided in Section III and V, respectively. Software package description is provided in the following section. Results and discussions are provided in Section VI and VII. The last section will discuss the conclusion and future work.

II. BACKGROUND/RELATED WORK

In this section, we will first discuss some necessary terminologies we used in our work. Then we will talk about some similar projects.

The system can be divided into two phases: questionnaires input and profile photo upload. During questionnaires phase, user will answer seven questions about his/her requirements on the dating target. These answers will used to allocate the proper user cluster. The first technique we would like to introduce is **clustering**, as part of the **unsupervised learning**. Contradict to **supervised learning**, unsupervised learning does not require input data to have a label. It aims to find a hidden structure or hierarchy of the data being provided. Thus, there exists no error rate evaluation towards such algorithm. Clustering, which we used in this work, is one example of unsupervised learning. The goal of clustering is to reveal the hidden groups(aka clusters) of the full dataset, as shown in Figure 1 below.

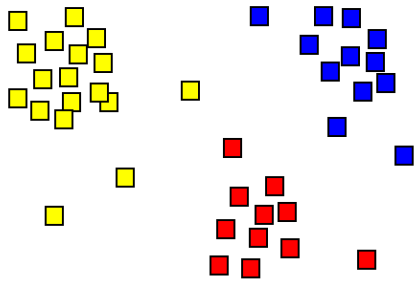

Figure 1: cluster data into three groups[4].

There are several clustering techniques, such as k-means, mixture models, as well as hierarchical models. We used **k-means clustering** in this work and we will discuss this algorithm in details in Section IV.

As described in the previous paragraph, the second phase of this system is to upload user profile photo and find the facial similarity matching. This step is

essentially achieved by using deep learning, especially neural network(NN). We will briefly discuss what neural network is here. Neural network is an architecture of unsupervised learning, shown in Figure 2. The advantage of neural network is that a two-layer NN can approximate almost all of the mathematical equations and by increasing the number of layers and the number of cells on each layer, the approximation result gets better and better.

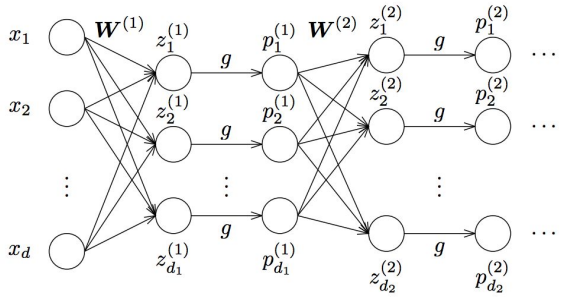

Figure 2: A sample structure for Neural Network with 3 layers and 3 cells in each[5].

The training phase of a NN is called **back propagation**, which is to take partial derivatives of each cell from the previous layer. After the back propagation, a NN will learn a weight W for each layer and cell. For example, cell $z_{i,j}$ will have a weight factor $w_{i,j}$. For each new input, the data will iterate through the entire NN and output several values based on its activation function.

With a very brief introduction on NN, we can then introduce the concept of Convolutional Neural Network(CNN). Why? State-of-the-art research has proven that CNNs are extremely useful in image and video processing for either classification or relocation task. Thus, CNN are almost the same as NN, except the fact that it explicitly specifies the inputs as images. Apart from that, it also adds two different layers besides the regular fully connected layer from NN, which are called convolutional layer and pooling layer.

Convolutional layer is essentially the process of convolution in signal processing. It first defines a window on input data then applies convolution onto the window to generate a new feature value, as shown in Figure 3 below.

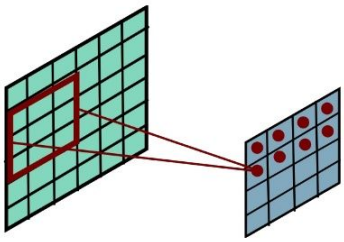

Figure 3: A sample convolutional layer[6]

With the help of pre-trained CNN models, one can take an image as an input and predicts what types of object this image belongs to. This is useful for facial recognition and similarity detection. We will discuss this algorithm in detail in Section IV.

There have been lots of research on incorporating machine learning techniques into online dating services. K.Tu et al. proposed a online dating recommendation system to achieve a similar task to this work. However, the approaches taken were fair different. In [7], they used the messages between users as training data input and perform Latent Dirichlet Allocation to extract latent user preferences from it. Their analysis tackles the recommendation problem by finding what types of preferences tend to match together, based on the observation of latent user preferences. While, in our work, we directly have user profile data and perform k-means cluster on the data.

In other works, people have tried various methods to incorporate machine

learning techniques into online dating services. However, we did not find previous on utilizing facial information to recommend a matching user and we believe our work uniquely contributes to this matter.

III. SYSTEM OVERVIEW

A. Dataset Used

To prepare for the data set, we need two categories of the data, which are text file containing user's basic information and the face image associated with each profile. On the beginning of the data collection, we decided to crawl data from OKCupid website using its public API. However, the API was then proven to be banned from the company later in 2014. Without accessing data through crawling, we did research on former published paper and found a data source for user profile file on the according GitHub[8]. The zip file *profiles.csv* contains nearly 50,000 user profiles information including each user's age, gender, and more. We picked 8 fields out of 32 fields, for better clustering, since some fields such as personal introduction essays would be unnecessary. We also left out features which are dependent on others, for example, income is correlated with education level to some extend. After transforming those fields into numerical values, for example, change gender field from female/male into 0/1, we finally have the data ready to run the clustering algorithm on.

For the face image data, since the OKCupid API was broken, we did not manage to collect image for each profile data we already have. Because *MateFinder* system have two independent steps of recommendation, we decide to collect photos data separately and manually link them to profiles data in each cluster randomly. The face data sources we utilized eventually were Chicago Face Dataset[9] and image folder containing all students' photo who have taken course EECSE6893 at Columbia University[10]. Chicago Face Dataset allows free download, while we used Python ImageScraper tool to collect student's images on course website under Professor Lin's consent. Combining two data sources, we now have around 2,000 face images. After uploading them to personal GitHub.io page for generating a utilizable url, and tagging each url into different gender categories, we can then attach the url to profiles data randomly without messing up with the gender, a complete data set is thus ready.

B. System Design

As briefly described in above sections, the system is separated into two phases: questionnaires and photo. A complete system diagram is shown below in Figure 4. A user needs to first answer seven questions related to age, height, zodiac, etc. The answers will help the system to allocate a correct cluster of users. Then the user will need to upload a photo into the system. The system then will use this photo to find the top K user with the most similar profile photo, within the same cluster. Then the user will be able to pick a profile and look for more details of the interested person.

During these two processes, different algorithms are used, which we will explain in detail in the next section. However, some of the design decisions as well as technique details are worth to mention in this section. First of all, we decided to store all user profile information in Amazon Web Service RDS MySQL

database.

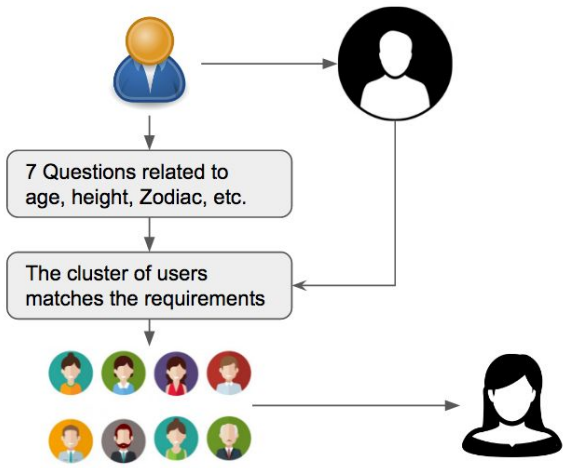

Figure 4: System Diagram of MateFinder

However, it is difficult to store image as part of the database, thus we decided to upload the images onto Github io page and store the image URL instead of the real image into the database, which can be pretty standard for image classification task, for example ImageNet or Microsoft COCO all use the same strategy.

We also had another debate of the programming language which we should use. Initially we decided to use Java Mahout to implement our algorithm because Java generally provides a smooth backend for the web application. However, since a lot of other tasks are implemented in Python, we later decided to use Python to implement the entire project.

Lastly, we initially decided to use decision tree, which is a supervised learning technique, to determine what group of user should be a fit for the designated user. In order to optimize the algorithm, we implemented K-D tree structure for the decision tree algorithm. However, the program still needs a pretty long runtime to perform the calculation and this obeys our design criteria which is to perform the matching calculation in real time. Thus, we later decided to use K-mean clustering as the first algorithm. By using K-mean clustering, we will need to precompute the clusters before the user uses the web application and cache it periodically to receive maximum efficiency. The downside of using K-mean clustering is that it needs to update everytime before a new user registers and uses our service. However, this algorithm seems to have better overall performance and we decided to pick K-means as the first algorithm, which we will explain in the next section.

IV. ALGORITHM

As we discussed before, there are mainly two algorithms we used in this work. The questionnaire phase utilizes **K-mean clustering**, which is a type of clustering algorithm. Lloyd's algorithm(aka K-means) alternatively updates the cluster representative c_i and cluster assignment variable Φ_i until some stopping criteria has met. A sample of k-mean clustering result in 2D is shown below, where k = 3 in this case:

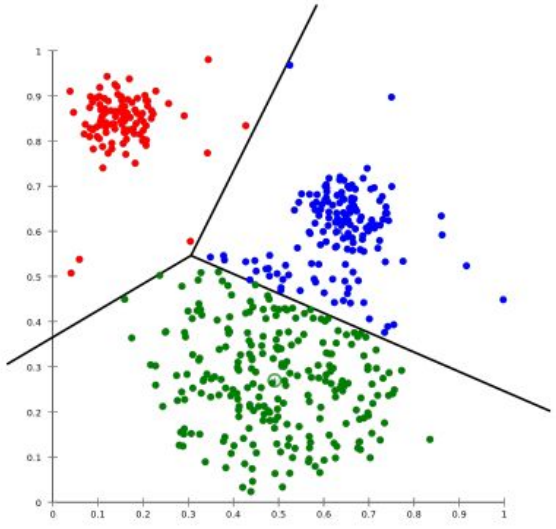

Figure 5: k-mean clustering (k = 3)

As an unsupervised learning algorithm, k-mean clustering does not require the data to have any label, thus works perfectly in our case. Before the user uses our system, it already precompute the clusters. The answers from the user questionnaire will serve as a new data point and the system essentially will fit this new point into the designated cluster. One thing to note is too large/small cluster size can easily cause the data to be overfit/underfit. One needs to finetune the size to achieve optimal results. In order to select the best cluster size, we followed the elbow rule to find the optimal solution.

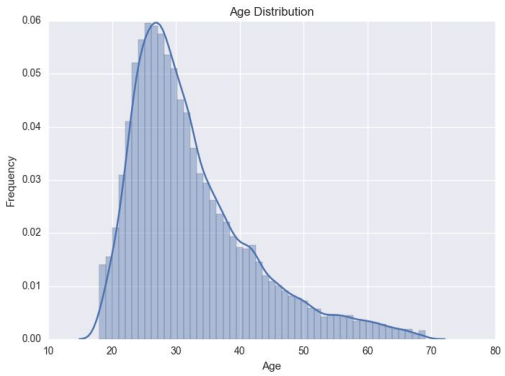

Figure 6: Age distribution of our data

Elbow rule looks at the percentage of the number of clusters: One should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data[11]. We visualized the data in terms of age and height distribution shown below in Figure 6 and 7.

After visualizing the data from all seven features, we further plotted out the elbow rule, shown in Figure 8.

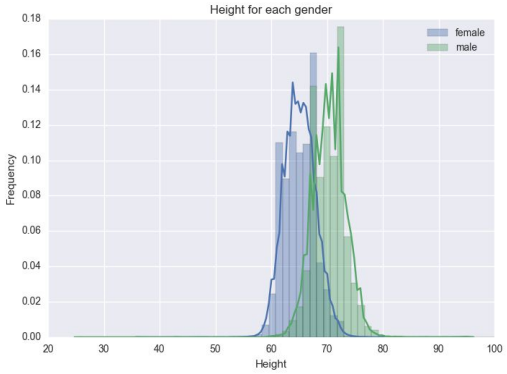

Figure 7: Height distribution for each gender

As we can see from Figure 8, when the cluster size is around 15, the error does not decrease significantly. This is how we come up with k = 15 and we believe this is the optimal solution for our project.

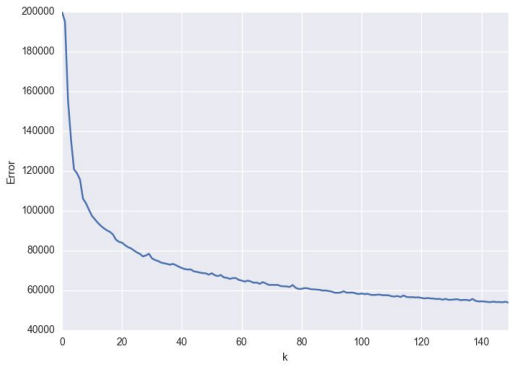

Figure 8: Elbow rule plot

The second phase, photo upload, uses convolutional neural network algorithm. We have already discussed how CNN work in the previous section, so here we will focus on how CNN help us in terms out finding facial similarity. In a typical image classification task, the last layer of the CNN is a fully connected layer, which will match the features from multiple convolutional layers onto different labels. The output layer has a probability distribution and the system will pick the label corresponding to highest

probability. However, this step is unnecessary in order to find the similarity between two faces. One can simply take the output from the last layer before the output layer, which is normally a feature matrix and compute the distance between the two faces. The distance function is somewhat arbitrary and in our case, we decided to use L2 norm because this is one of the most common strategy. In our work, we took the advantage of using Microsoft Face API to help us implement the project. Face API wraps the CNN into functions and uses their pre-trained model to extract features from input faces. We first need to store all images in a face list, and then compare each image in the list with the coming user image. We will describe the Face API in detail in the next section.

V. SOFTWARE PACKAGE

Figure 9 shows the process for our web application. Both processed categorical data as well as the detailed profile data with image url are stored in AWS RDS MySQL database waiting for query. Once the user provides the questionnaire answer and upload a photo, the photo will be stored to Cloudinary and a unique image URL will be returned. This will be used in Microsoft Face API to detect new face. All other information will be stored in the memory, and Python Spark will query the processed categorical data from database and build the kmeans model. The model will then be interpreted by scikit-learn and it will then find the closest cluster for the given user profile information. The resulted cluster information will then be passed to Microsoft Face API. In the Microsoft API, the user's face will be compared to all faces that belong to this cluster, and we will display the top matches (normally less than 10 recommendation results). We used Python Django to connect database with frontend, and we used HTML5, CSS and JavaScript to design the webpage in order to give the best user-experience. We have uploaded our website to AWS Elastic Beanstalk for public usage. You can also run locally by clone our python code through github link: https://github.com/Sapphirine/Mate_Finder_Dating_Website

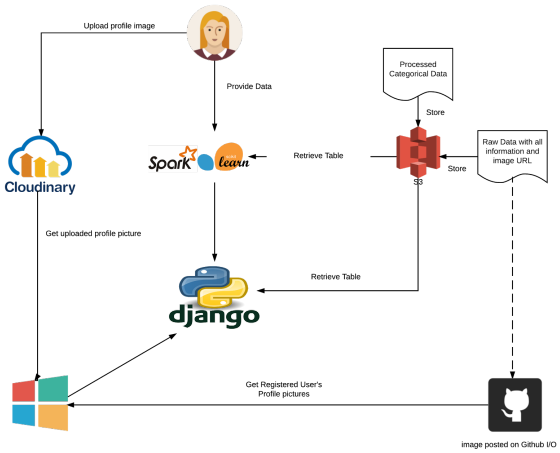

Figure 9: Web application process

Below is a sample work-through of our web application.

VI. DISCUSSION

In this section, we will discuss some alternative approaches we came up with but did not use in the end. We will also discuss many disadvantages of our projects and how we will improve on them.

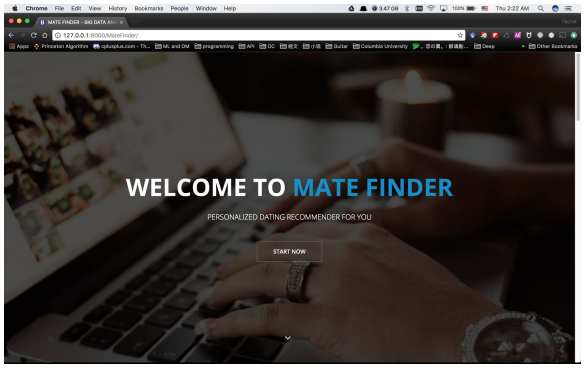

Figure 10: Welcome Page for our website

As we discussed in earlier sections, originally we would like to use decision tree algorithm for splitting the user cluster. We thought this might be a better way to do the task because decision tree, as a supervised learning, can provide us with a much accurate user group. Essentially, the system should run the decision tree each time it receives an answer from the user and runs

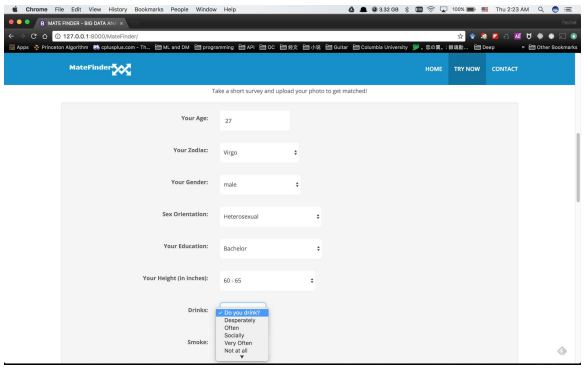

Figure 11: Survey page of our website. You can upload a photo here too.

seven times in total. This indeed gives us satisfying results, however, also has several downsides. First of all, running decision tree seven times is very computationally costly. Every run requires approximately 20-30 second to compile. This is unrealistic since we are aiming at a real-time matching platform and the long runtime gives up a lot of resources and potential users. We then optimize decision tree algorithm with a K-D tree implementation. K-D tree allows the algorithm to search only half of the dataset at each split. This decreases the program's runtime from O(N) to O(logN) and in fact this indeed proves to be successful. However, there is a second disadvantage of using decision tree. Decision tree asks the system to strictly follow the filter, that means, it will filter out some results that are 100% matching in 6 questions but fails at one question. This result is not ideal so we decided to move to another algorithm. We finally came up with using k-means clustering algorithm and used elbow rule to find the optimal cluster size. This algorithm also gives us a very satisfying result in the end.

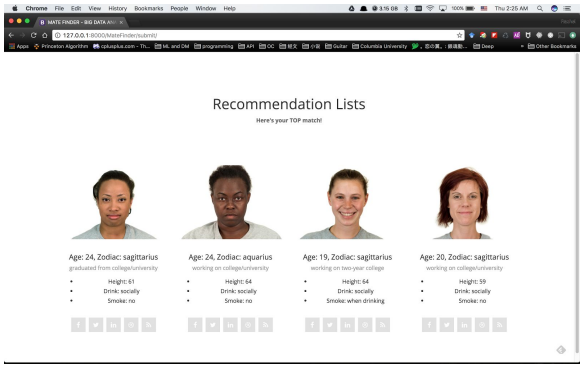

Figure 12: A list of Recommended users with their detailed information after user entered his/her information and uploaded a profile photo.

There is a significant downside of using k-means algorithm, however. Each time a user creates a new profile in the web application, the algorithm will need to be re-runned. This can be problematic as it is not realistic to be done in real-time. So when a user is using the system while another user is creating a profile. This valuable information could be missed. We

are still actively looking for a better algorithm or improvement on k-means clusters to solve this issue.

As for the facial similarity check, there is also a downside. Right now, due to the limitation of training samples, all of the photos are clear, frontal profile photos. Thus the face similarity check can be extremely accurate. We did receive very promising results with our experiments. However, this is not realistic in reality. Users will definitely upload non-frontal photos or even photos which do not contain any faces. This can create a lot of noise within our system. The method we came up with to handle it is to add an image classification layer on top of the face similarity checking layer. Before the photo gets into the pool of candidate users, the system needs to verify whether the photo contains a human's face. This task can be done relatively easy with state of the art techniques in computer vision. We did a brief literature search and decided to use a pre-trained model to complete this task. The specific model is still to-be-determined as we would need to check how the system response to the data which we have.

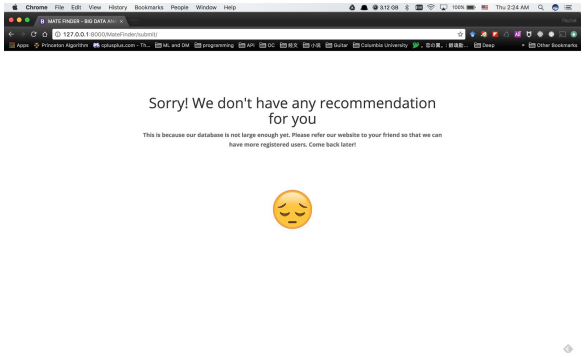

Figure 13: Error page whenever we cannot match someone to user given his/her basic information and photo. This is because the number of our registered users are not big enough.

Moreover, the current system is only applied on web application platform. We are actively looking into extending our product onto mobile platforms, both Android and iOS platform.

VII. CONCLUSION

In this report, we discussed our proposed system, *MateFinder*, the next generation of dating recommendation system. Online dating services have great market potential yet many popular dating applications did not incorporate with state of the art machine learning technologies. Some research have been done in this field but we uniquely proposed our solution with finding facial appearance similarity between users. We used K-mean clustering and CNN to achieve our proposed system and implemented the design with PySpark and Microsoft Face API. We believe our system, *MateFinder*, will yield significant impact among online dating users. We also discussed several disadvantages of our projects and the ways we planned to solve them. In the future, we look forward to improve our system's robustness and effectiveness, as well as extend our product onto mobile platforms.

VIII. ACKNOWLEDGEMENT

We would like to thank Prof. Lin for the fantastic lecture. We would also like to thank all the teaching assistants that helped us with our final project, especially on using PySpark.

IX. REFERENCES

[1]: www.match.com/magazine/article/4671/

[2]: Convergence in the physical appearance of spouses, R. Zajonc et al.

[3]:Goleman, Daniel. "Long-married couples do look alike, study finds." *New York Times* (1987)

[4]: Cluster analysis from Wikipedia, access time: Dec 20, 2016
[5]: COMS 4771 slides from Daniel Hsu
[6]: Why Convolutional Neural Networks, S. Siriwardhana, 2016
[7]: Online Dating Recommendations: Matching Markets and Learning Preferences
[8]: https://github.com/rudeboybert/JSE_OkCupid
[9] http://faculty.chicagobooth.edu/bernd.wittenbrink/cfd/index.html
[10] http://www.ee.columbia.edu/~cylin/course/bigdata/images/
[11]: Elbow Method from Wikipedia, access time: Dec 20, 2016