Project Report Phase 3:

Jiaxuan Pang
Kuan-Ting Shen
Bandon Tseng
Ching-Wen Tu
Ting Xia
Ming Yi

Abstract

In this phase of the project, we worked on, implemented, and experimented different clustering, indexing algorithms to achieve classification and relevance feedback approaches. Those higher level algorithms are build on implementation of two previous phases. Including feature extraction methods including CM, HOG from phase 1, and feature dimension reduction and matrix factorization methods including Principal component analysis (PCA), singular value decomposition (SVD).

Keywords— Image Retrieval, Clustering, Indexing, Classification, Relevance Feedback SVM, PPR, Decision Tree, K-Means, LSH

1 Introduction

Image retrieval is a term for browsing, searching and retrieving images from a larges database of digital images. By adding metadata to images such as title or descriptions, the traditional image retrieval can be performed over these annotate data [6]. However, manually annotate data is considered time consuming; as a result; finding a way to automatically clustering, indexing and classify them into correct labels becomes important. In the previous phases, we implemented and experimented on feature extraction algorithms, which are Color Moment(CM), and Histograms of oriented gradients(HOG). We compared the performance, and complexity of different algorithms. In addition, to further discover deeper latent semantic under different feature representations and to reduce the computation complexity, we also implemented and experimented feature extraction methods including Principal component analysis (PCA), and singular value decomposition (SVD). In this phase, we first proposed our methods to classify image lables by using latent semantics or extracted feature, we then implemented and applied clustering, indexing algorithms including K-Means, PPR, SVM, Decision Tree to solve the classification tasks. We finally implemented and experimented the Locality Sensitive Hashing(LSH) algorithm and applied it on the relevance feedback approach.

1.1 Terminology

In this phase, each image are converted to the YUV color model, and feed to one of feature extraction algorithms from phase 1. There are certain technique needed to further reduce the feature variables and let us focus on most important ones. There are two majority ways of reducing the dimensional, **feature elimination** and **feature extraction/feature dimension reduction**. Unsurprisingly, **feature elimination** will be easiest way to reduce the feature dimension by deleting features. However, the disadvantages is significant, the variables being dropped will definitely contribute no information. **feature extraction/feature dimension reduction** addresses the problem that feature elimination exposed. Intuitively speaking, the feature extraction/feature dimension reduction algorithms create new independent feature variables that being mapped by the original feature variables. Principal component analysis(PCA) is one of representative feature dimension reduction algorithm [2]. Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's usually used to make data easy to explore and visualize. Singular value decomposition(SVD) is another feature dimension reduction technique. Its key concept is matrix decomposition. It describes a given matrix using its constituent elements. The general formalize of SVD is $M = U\Sigma V^T$, which M input matrix, Σ is singular values and is diagonal. We will describe the problem specification and the goal in next chapter. In addition, since the project and the report is collaborate with multiple people, the usage of notation may vary, the notation only work in the scope of each section.

1.2 Goal description (problem specification)

1.2.1 Task 1

Task1 is to use latent semantics to label unlabeled images. Given a set of palmar images and a set of dorsal images, we need to: First, find proper latent semantics of these images. Second, only use those latent semantics to label unlabeled images.

1.2.2 Task 2

Task2 has two parts. The first part is to cluster palmar images and dorsal images separately. The number of clusters is specified by the user. The second part is to label images by using the clustering results.

1.2.3 Task 3

In this task, we are supposed to implement the personalized PageRank formulation based on RandomWalks with re-start which input should be an image-image similarity matrix. Based on this algorithm and input, by given three images, we are supposed to provide **K** most similar images to the three images. To solve the problem, we first need to construct image-image similarity matrix as input to the PPR algorithm, we then need to figure out the formula of the PPR algorithm and how to solve it. The detailed solution will be described in the section 2.3.

1.2.4 Task 4

In this task, we need to implement Support Vector Machine(SVM), Decision Tree, and PPR algorithm based classifiers. By given a folder of labeled (dorsal-hand vs palmar-hand) image as training data, the classifiers should be able to label another folder of unlabeled images to be either dorsal-hand or palmar-hand. Each of these classifiers have their unique issue to solve, for example, we need to figure out how does SVM tolerate the error is the training data is non-linear separable, and when to stop partitioning the data space to prevent over-fitting for Decision Tree algorithm. Both of the solutions will be discussion in detail in section 2.4. In addition, since PPR is a page ranking algorithm, how to turn ranking system to classifier system will also be talked in the same section.

1.2.5 Task 5

Task 5 has two parts. The first part is to implement a Locality Sensitive Hashing (LSH) tool (for Euclidean distance) which can input the number of layers, the number of hashes per layer and a set of vectors and then creates an in-memory index structure containing the given set of vectors. The second part is to implement a similar image search algorithm using this index structure and a visual model function of your choice, for a given query image visualizes and rank the most similar images, and output the numbers of unique and overall number of images considered.

1.2.6 Task 6

In this task, we need to implement either of the four feedback systems: an SVM (Support Vector Machine) based relevance feedback system, a decision tree based relevance feedback system, a PPR-based (Personalized Page Rank) relevance feedback system and a probabilistic relevance feedback system, which enable the user to label some of the results returned by task 5b as relevant and irrelevant and return a new set of ranked results, which hope to improve the accuracy of the result from task 5b.

1.3 Assumptions

This phase is based on the assumption that all of the function in previous phase including phase 1 and phase 2 are full functional. In addition, the input of any dimension reduction methods should be $n * m$ matrix. For tasks requiring meta-data, we assume all of the information in meta-data is correctly provided. For task 1 to task 4, we will use phase 3 sample data as our dataset. The dataset should contain both labeled and unlabeled image and their corresponding metadata. For task 5 and 6, we need to use 11K data and its metadata is provided by the website.

2 Description of the proposed solution and implementation

2.1 Task 1

In task1, the first moment(FM) of the YUV color model and HOG are selected as feature descriptors. SVD is used to create latent semantics(LSs). The algorithm is as follow:

1. Compute two descriptors of all images, including the unlabeled images.
2. Split all labeled images' descriptors into four groups(FM labeled as palmar, FM labeled as dorsal, HOG labeled as palmar, and HOG labeled as dorsal)
3. Create four SVD latent semantics(FM palmar LSs, FM dorsal LSs, HOG palmar LSs, HOG dorsal LSs)
4. For a given image, first, project it into those four LSs. Then compute the L2-norms(Euclidian Distance) of the vectors.
5. Suppose the lengths in four LSs are:

$$CM_{Palmar}, CM_{Dorsal}, HOG_{Palmar}, HOG_{Dorsal}$$

If

$$\frac{(CM_{Palmar} - CM_{Dorsal})}{max(CM_{Palmar}, CM_{Dorsal})} * 60 + \frac{(HOG_{Palmar} - HOG_{Dorsal})}{max(HOG_{Palmar}, HOG_{Dorsal})} > 0$$

,the image is labeled as palmar. Otherwise, it is labeled as dorsal.

Details and choices behind the algorithm:

- Choose first moment and HOG as descriptors: For human eyes, the critical differences between dorsal images and palmar images are in colors and textures. So we choose first moment to try to describe the color of a hand and HOG to capture the wrinkles and patterns. In the beginning, we used all three color moments. Later we found that the most useful one is the first moment. So we reduce the descriptor to gain efficiency. We had also tried LBP descriptor in the first place, but it performed not as good as HOG in this task.
- Use SVD latent semantics and measure length by L2-norm: We can use the first several LSs of SVD to preserve most of the data after sorting them according to their weights. It is likely that for a dorsal image, most of its weights concentrate more on the first several dorsal LSs than on the palmar LSs. Such conjecture was concluded from the observations in the previous phase. We have tried L1

to L4 norms. The L2-norm performs best. Other than SVD, we also tried PCA LSs and cosine similarity(CS). Among four possible combinations (SVD-CS, PCA-CS, SVD-P norm, PCA-P norm), the SVD-L2 norm is the best.

- Combine the lengths: How to combine the lengths is a tricky problem. Simply summing the two lengths up is meaningless as they represent two different characteristics. Here, we compute the relative differences and use factors to adjust their weights. 60 is a magic number coming out of experiments. This combination also provides two benefits: First, it can sometimes label the images more accurate than only using FM or HOG individually. Second, the accuracy is insensitive to the number of LSs k. Thus, it is more robust than using FM or HOG exclusively.

2.2 Task 2

In task2, only the first moment(FM) of the YUV model is selected as feature descriptors. We use k-means clustering with random initialization to create clusters in PCA latent spaces that have the top ten latent semantics(LSs). The algorithm is as follow:

1. Compute the descriptors of all images including the unlabeled images
2. Split all labeled images' descriptors into two groups(FM labeled as palmar, FM labeled as dorsal)
3. Create PCA latent spaces (FM palmar LSs, FM dorsal LSs) based on those two groups and project all labeled images into their corresponding latent spaces(palmar to palmar, dorsal to dorsal).
4. Do k-means clustering in the latent spaces according to the user-specified c. Then choose the centroid of each cluster as its representative(Cpalmarj in FM palmar LSs and Cdorsalj in FM dorsal LSs).
5. For a given unlabeled image, project it into both latent spaces and get two vectors, Opalmari(in FM palmar LSs) and Odorsali(in FM dorsal LSs). Then compute the length

   $$L_{palmar}i = \sum_{j=0}^{c-1} sizeof(C_{palmar}j) * ||O_{palmar}i - C_{palmar}j||$$

   and the length

   $$L_{dorsal}i = \sum_{j=0}^{c-1} sizeof(C_{dorsal}j) * ||O_{dorsal}i - C_{dorsal}j||$$

   If

   $$L_{palmar}i < L_{dorsal}i$$

   , the image is labeled as palmar. Otherwise, it is dorsal.

Details and choices behind the algorithm:

- Use the first moment of the YUV color model: Only using FM seems very lame, but we have tried everything that we could think of and found that all other kinds of descriptors performed poorly.

- Implementation of the k-means algorithm: The initial centroids are selected randomly. The default stop condition is either 500 iterations are done, or the relative differences of centroids between two iterations are less than 0.1%. A random initialization sometime can lead to a lousy clustering. Also, given a large c, some clusters could be empty during executions. Repeating the execution several times and choose the proper one can solve both problems. Therefore, the algorithm chooses the best one, in terms of RMSE, among fifteen executions as the final clustering results.
- Doing clustering in latent spaces: Without dimensional reduction, there will be two problems: First, the dimensionality of each image could be so high that clustering will be prohibitively slow. Second, in a high dimensional space, there could be many noises, which can affect the clustering negatively. PCA is useful here in the sense that it preserves the distances between images where they vary a lot to maximize the separation while eliminates dimensions where the images vary a little to maximize compactness. However, it is also possible that the eliminated dimensionalities are not noises but crucial for making the labeling decision correctly. Our experiments show that the accuracy depends on both test data(unlabeled images) and the number of LSs. We will try to explain that after showing the experimental results.
- Using length to decide labels: The origin of a PCA space is the center of all data. We can treat the cluster's descriptors as borders that define the region of what is acceptable. For an image Oi, If the value $\sum_{j=0}^{c-1} sizeof(C_j) * ||O_i - C_j||$is large, we can think that it deviates from the center so much that crosses the borders. Thus, we compare the two lengths from different latent spaces and choose the smaller one as the final label.

2.3 Task 3

As described in the section 1.2.3, the first sub-task is to construct image-image similarity matrix as the input of PPR algorithm. Given implementation of phase 1, we first use Color Moment as feature descriptor. Please note here that for this task, we did not use any dimension reduction methods. We construct the similarity matrix T, we first flatten each image's feature matrix and do the dot product with other images' flatten feature matrix. Given k indicating outgoing edges to k most similar/related images, we obtained the normalized similarity graph T_G by

1. For each image in T, get k most similar image indexes in the similarity matrix and their similarity values
2. Construct similarity graph T_G, which has the same dimension as T, and fill the similarity values based the index obtained from previous step.
3. Normalize the similarity graph T_G, to make sure that for each image the summation of the probability of outgoing edges is 1

Recalled the equation of personalized PageRank formulation based on RandomWalks with re-start

$$\vec{\Pi} = \alpha T_G \vec{\Pi} + (1 - \alpha)t$$

We use N to represent the total number of node in the graph. $\vec{\Pi}$ is a $1xN$ matrix represents the final steady state for the page rank. When $\vec{\Pi}$ reaches the steady state, it can represent how likely each node will be staying. T_G is the similarity matrix with dimension of NxN, representing for each node with the probability it transfers to its connected nodes . t is the teleportation matrix with $1xN$ dimension , representing for each node with the probability it transfers to nodes that it is not connected. We can fill the teleportation matrix by the user given input. For example, the task asks user to provide 3 specified image ids. As the result, the index of 3 specified image ids in t will be replaced by probability of 1/3. We finally can solve the steady state $\vec{\Pi}$ by

$$\vec{\Pi} = (I - \alpha T_G)^{-1} \cdot (1 - \alpha)t$$

After solving $\vec{\Pi}$, we are able to find K index that has largest probability in the steady state, and map the index with real image ids for output. In this case, we chose $\alpha = 0.8$. We will analyse the result in following section.

2.4 Task 4

2.4.1 Implementation of SVM classifier

We implemented SVM according to the SVM and the gradient descent algorithms described in Chapter 12 of [7]. The weights will be randomly assigned within $[-1.0, 1.0]$ when we initialized a SVM classifier. The dimension of the weight is the length of feature plus 1 - which is for B. The format of training data of SVM is a document-term matrix and a vector of labels which contains True/False that means two categories of corresponding document in the input matrix. We also add one column which all contains 1 to append to last of the input document-term matrix that is also used for B. During training, we multiply weight with the transpose of document-term matrix and find out the difference document. The applying the Gradient Descent algorithm to update the weight. The output of the predict function would also be True/False to indicate the categories. In task 4, the True means dorsal and False means palmar.

2.4.2 Implementation of Decision Tree classifier

To train a Decision Tree model, we need the feature of the training data which may be the image original features, features after extraction or features after the reduction and the label of each image with the binary labeling. With the provided training data, we will fit them into our model. For Decision Tree, we need to decide which feature should be the representative for the node, so we use the Gini Index to calculate the score of each feature. The Gini Index is calculate by

$$Gini = 1 - \sum (p_j)^2$$

where p_j is the probability of class j. The feature which has the smallest Gini score can be the node of the tree. Due to the training data and the number of the total features are not huge enough, we do not let the tree to grow too depth which may cause over-fitting. After the tree has been built, we put the testing data into tree. Each of the image will start from the root of tree and then go to the subtree which features fit the constrains of the root of subtree. Until the image go to the leaf of the tree, then we can decide the label of the unlabel image which can be dorsal or palmar.

2.4.3 Implementation of PPR classifier

To train a PPR model, we just obtain similarity graph T_G from training data. During the verification step, we first expend one low and one column on T_G. And for each unlabeled data, we add normalized probability edges from the unlabeled data to all of other images in the training set. The edge number is determined by the hyper-parameter k. In addition, we construct teleportation matrix t and add one row with probability 1 to represent the query unlabeled data point. We then solve $\vec{\Pi}$ for steady state, we find K most dominate images other than the query unlabeled image. To mark the query image to be either dorsal-hand or palmar-hand, we will ask these dominate images to vote. The voting strategy is weighted, meaning for each image in the dominate images set, it gains $-p_i$ if this is labeled as palmar-hand, otherwise, it gains p_i. We finally mark the query image as dorsal-hand if the summation of all these weights is positive, palmar-hand if the summation of all these weights is negative.

2.5 Task 5

In this task, we implement a Locality Sensitive Hashing (LSH) class based on random projection method. The primilary idea of this method has two steps: building hashing structure and mapping query object to the bucket. We firstly initiate L hash tables with hashes that are K bits in length. The hash function we apply is defined as:

$$h_{a,b}(v) = \left[\frac{a \cdot v + b}{W}\right]$$

where v is the input vector. The vector a is a K-dimensional random vector that the value is sampled from normal distribution. We also make each row vector in vector a to unit length in order to speed up for later processing. The offset b is a random value in the range of [0, W). If the value W is finely tuned, the similar object should hash to the same bucket. After applying the hash function to the input vector, each value in the K length features is assigned to 1 if the value is larger than zero. Otherwise, the value is assigned to zero. By concatenating all the 0 and 1 into a single string (or hash value), we are able to put the object into the corresponding bucket.

So far we have constructed the hash tables and ready to hash the query object. To perform the similarity search, the query object is indexed to a bucket for each hash table, and then we create a candidate set by integrating all the candidates in each bucket that the query object is hashed to. Once the candidate set is built, we rank all the objects in the set by simply calculating the distance between the candidate object and the query object. At last, we return t most similar objects. However, sometimes the amount of the object in the candidate set does not satisfy the requirement. Therefore, we design a mechanism to solve this problem. For each hash value that the query object is hashed to, we change one or more bits of the hash value in order to get more candidates from similar buckets. So changing one bit at each time in the K length hash value, we can get other candidates in K similar buckets to fulfill the requirement. Though this is not best way to implement (for example, we can decrease the K value and reconstruct all the hash tables until the candidate set meets the requirement, but it is time-consuming), we are satisfied with the current outcome.

2.6 Task 6

In this task, we choose to re-rank the results returned from task5 after labeling by users. Ideally, a feedback system should be able to support iterative feedback. In our case, such a system will re-order the same results again and again, which is useless. Therefore, all classifiers will respond once. The implementation of SVM, Decision

Tree, and PPR has been discussed in Section 2.4.1, Section 2.4.2 and Section 2.4.3. For the PRF system, we will discuss it in Section 2.6.1.

2.6.1 Probabilistic Relevance Feedback(PRF) System

In PRF, we use the first moment(FM) of the Y channel in YUV model and downsampled HOG feature vectors as descriptors. We assume each element of a feature vector is independent. The algorithm is as follow:

1. Pre-process all numerical descriptors to boolean ones.
2. Compute four kinds of probabilities of each feature.$P(f_i = true|rel)$, $P(f_i = false|rel)$, $P(f_i = true|\overline{rel})$, $P(f_i = false|\overline{rel})$.
3. Compute each object's probability given it is relevant and irrelevant.$P(O_i|rel)$, $P(O_i|\overline{rel})$
4. Use the ratio $\frac{P(O_i|rel)}{P(O_i|\overline{rel})}$ to re-rank the results.

Details and choices behind the algorithm:

- Converting feature vectors into boolean vectors: If a feature is dominant in an object, its boolean representation is true. Otherwise, it is false. Whether a feature is dominant is determined by thresholds. The thresholds cannot be too small or too big because either case will make the boolean vector less discriminative. The thresholds for FM and HOG are half of their maximum values.
- Choose and reduce feature vectors: We could choose the same descriptors in terms of types and sizes as in task 1. However, if there are too many features, the probabilities for each feature computed in step2 will be too small. Besides, the probabilities computed in step3 can easily become zero. So we choose the Y channel of YUV model which is the most noticeable for humans. Some elements of a HOG feature vector describe the same sub-areas of an image. That means they are not independent of each other. In order to make the probability computation valid, we downsampled the vectors to decorrelate them.
- Compute all kinds of probabilities: The method is based on the textbook [3]. The formulas are:

$$P(f_i = true|rel) = P(f_i = true|R) = \frac{|\{O_i||(O_i \in R) \wedge (\vec{O_i}[j] = true)\}|}{|R|}$$

$$P(f_i = false|rel) = P(f_i = false|R) = \frac{|\{O_i||(O_i \in R) \wedge (\vec{O_i}[j] = false)\}|}{|R|}$$

$$P(f_i = true|\overline{rel}) = P(f_i = true|R) = \frac{|\{O_i||(O_i \in R) \wedge (\vec{O_i}[j] = true)\}|}{|I|}$$

$$P(f_i = false|rel) = P(f_i = false|R) = \frac{|\{O_i||(O_i \in R) \wedge (\vec{O_i}[j] = false)\}|}{|I|}$$

$$P(O_i|rel) = (\prod_{\vec{O}_i[j]=true} P(f_i = true|rel))(\prod_{\vec{O}_i[j]=false} P(f_i = false|rel))$$

$$P(O_i|\overline{rel}) = (\prod_{\vec{O}_i[j]=true} P(f_i = true|\overline{rel}))(\prod_{\vec{O}_i[j]=false} P(f_i = false|\overline{rel}))$$

- Dealing with zero denominators: In step 4, we could encounter zero denominators. In such cases, we need to compare numerators. So we change all zero denominators to the minimum of $P(O_i|\overline{rel})$

3 Interface specifications

There are several parts of our program includes **module**, **model**, **classifier** and each task. Each functions in module and models folder can work independently.

3.1 Models

In the **models** folder, we implemented all four models, Color Moments, Local Binary Patterns, histogram of oriented gradients, and SIFT. We also introduced **modelFactory** in this phase to let us get model more easily.

- **modelFactory.py** - In this module, user can get all supported models' name by method **getSupportModel()** and get a model instance by **creatModel()** with a *str* type argument which is the name of the model, **createModel("cm")** for example.
- **model.py** In the the modules, we firstly craft an interface which is an abstract class for other implementations -**cm.py**, **lbp.py**, **hog.py**, and **sift.py**. In those models, we implemented some common interfaces for our tasks.
  - **extractFeatures()** - Extract features from an image. This function will split image to smaller windows if necessary.
  - **serializeFeatures()** - As we mentioned in the database module, not all data structure can be serialized by pickle. Therefore, the feature must know how to convert their data structure to a data format which can be used for python pickle.
  - **deserializeFeature()** - The load function of python pickle returns a data structure which is output of the **serializeFeatures** function. But the data may not be used directly, so this function will deserialize to the format which can be used as a feature to other functions.
  - **flattenFeature()** - Since the feature extract by **extractFeatures()** is a matrix, to be used as document-term vector, we need to flat the feature matrix as an one-dimensional vector. Each module would implement their method to flat the feature matrix according to the value of their matrix or some restriction. For example, the color moment need to ignore the third moment, skewness, when the dimension reduction method is non-negative matrix factorization(NMF). This function returns an one-dimensional numpy array.

3.2 module

We implemented several modules in **modules** folder for dimension reduction, distance functions, metadata reader, database and image reader.

- **DimRed.py** - We implemented SVD, PCA, LDA, and NMF in this file. There are also some functions for convenience such as $registerMethod()$, $getSupportedMethods()$ and $createReduction()$.
  - **DimRed.registerMethod()** - This function is used to register a new method to our program automatically.

- **DimRed.getSupportedMethods()** - This function return a **list** of supported methods. Each supported method would be store as lower case **str**:
  $$["svd","pca","lda","nmf"]$$
- **DimRed.createReduction()** - We can get a dimension reduction method instance by this method. the argument would be a str which indicates what method we wanted, **createReduction("svd")** for example.
- **printLatentSemantics()** - This function is used to print the image according to the latent semantics.
- **transform()** - We can use this function to transform a vector or matrix to the new space.
- **getLatentSemantics()** - This function returns the latent semantic. We use this latent semantic function to estimate the similarity of each subject in $task6$ and $task7$.

- **database.py** - The database module includes an abstract class, Database. When initialize a database, we have to assign which table we would like to access. The following is the interfaces of a database.

  - **addData()** - Add data to the table with a given key. The existed data can be overwritten if we set overwrite parameter.
  - **getData()** - Get data from the table for a given key. Return None if the data is not available in the database.
  - **keys()** - Get all keys in the table. This could be utilized for traversal the table.

  The **FilesystemDatabase** is an implementation of the abstract class Database. **FilesystemDatabase** stores whole DB in **FileDB** folder. We can also specify the table name when we initialize the database and the table would be a folder inside the **FileDB**. **FilesystemDatabase** basically convert the data into python **pickle** format, which is a built-in python library, and save it as a binary file that filename is key ID we given such as **Hand_0001102.pkl**. To get data from a table of database, the **FilesystemDatabase** will try to find the file in the folder and load the pickle file. The pickle, however, is unable to convert some data formats, such as **KeyPoints** of OpenCV, directly. The **FilesystemDatabase** is not responsible for such situation. We, therefore, have to make sure the input data is usable for pickle.

- **distanceFunction.py** - Regarding the distance estimation, distance module includes classes for calculating distance. We implemented a P-Norm distance as **Norm** and entropy function. We have to pass the order of P-norm when we initialize it. Using it by calling the class instance with two vectors to get the distance. We use this module in **task 2**, **task 4**, **task 5**, **task 6** and **task 7** to calculate the distance among reduced features.

- **handMetadataParser.py** - Because we need information in the matadata for **task 3**, **task 4**, **task 5**, **task 6**, **task 7**. This module is used for metadata parsing. We only support **.csv** format metadata in this project.

  - **getFilelistByLabel()** - This function can be used to get file list of specific label such as left-hand or male. It returns a list of image IDs in the list. This function can be utilized for **task 3**, **task 4**, **task 5**.

  - **getFilelistByID()** - We compare the similarity of subjects for **task 6** and **task 7**. We, therefore, implemented this function for getting the list of image IDs by subject ID. It returns two dictionaries - the key and value of first one are subject ID and list of image IDs, respectively, the second one includes image ID as key and subject ID as value.
  - **getFilelistWithOneHot()** - Since we need one-hot vector for the metadata space in **task 8**, this function generates one-hot vector according to the metadata.
  - **kmeans.py** - It is used in task 2. It is the implementation of clustering algorithm. $def\_\_init\_\_(self, n\_clusters = 8, max\_iter = 500, min\_rimpr = 1e-4, rpt = 15)$
  - **PRF.py** - It is used in task 6.

3.3 Classifier

In this module, we implemented three classifiers, SVM, Decision Tree, and PPR algorithm based classifiers. All of them are inherited from **classifier.py**.

- **classifier.py** - It defines all abstract interfaces and ready to be inherited by all other three classifiers. It contains two generic methods **fit()**, and **predict()**. **Fit** methods requires two parameters which are training data set and ground truth labels. It will train the classifier model depending on the inherited algorithm. **Predict** will take input of testing dataset, and return the label of each data.
- **svm.py, decisiontree.py, ppr.py** - All of them inherit from **classifier.py**. Their fit and predict methods are overwrite based on their own algorithms.

4 System requirements/installation and execution instructions

In this phase of project, we use python as our language. To do dimension reduction, we utilize several libraries such as $Numpy$ and $scikit-learn$.

4.1 Environment Setup

Anaconda is a distribution of *Python* and *R* which aims to simplify package management mainly for scientific computation. Based on the operating system on the computer, the corresponding version can be easily downloaded from the website. After the installation, we can create our own clean environment.

To create a new environment:

```
conda create -n <environment name> python=3.7
```

We then activate the environment:

```
conda activate <environment name>
```

As the new environment is initialized, we begin to install our necessary packages. The packages installed are listed below

- **OpenCV** - An open-source library that support plenty of computer vision algorithms, such as extracting SIFT or HOG features from images
- **Numpy** - A package providing high-performance multidimensional array object
- **Scikit-learn** - An open-source library providing efficient tools for data-mining and data analysis
- **Scikit-image** - An image processing toolbox for SciPy, such as extracting LBP features from the image
- **Matplotlib** - A data visualization library built on NumPy arrays and designed to work with SciPy

To make installation process simple, all the packages are written in *requirements.txt* and can be installed with following command:

```
conda install --yes --file requirements.txt
```

4.2 Execution instructions

Note: Please run phase 1 task 2 first to create descriptor database before running any of the following tasks except task1 and task2. The pre-computation for task 1 and task 2 is p3pre.py.

- Phase 1 task 2 Give input file path with argument -i, use color moments with argument -m, and store to table "test".
  - **-i** - Give input file path.
  - **-m** - use color moments.
  - **-t** - Table test.

  Example:
  ```
  python p1task2.py -i ~/hw/cse515_data/Dataset -m cm -t test
  ```
- Phase 3 pre
  - **-p PATH** - The path of labeled images.
  - **-unp PATH** - The path of unlabeled images.
  - **-t TABLE** - The name of table created to store image features.

  Example:
  ```
  python p3pre.py -p ~/hw/cse515_data/phase3_sample_data/Labelled/Set1 -unp
  ~/hw/cse515_data/phase3_sample_data/Unlabelled/Set1 -t task12
  ```
- Phase 3 task 1
  - **-p PATH** - The path of labeled images.
  - **-unp PATH** - The path of unlabeled images.
  - **-k K** - The top k of latent semantics.
  - **-t TABLE** - The name of table used to store image features.
  - **-meta PATH** -The path of metadata for labeled images.
  - **-test(optional) Test mode** - The program will print the accuracy, the meta has to be the complete one.

Example:
```
python p3task1.py -p ~/hw/cse515_data/phase3_sample_data/Labelled/Set1
-unp ~/hw/cse515_data/phase3_sample_data/Unlabelled/Set1 -k 30 -t task12
-meta ~/hw/cse515_data/phase3_sample_data/labelled_set1.csv
```

- Phase 3 task 2
  - **-p PATH** - The path of labeled images.
  - **-unp PATH** - The path of unlabeled images.
  - **-k K** - The top k of latent semantics.
  - **-c CLUSTER** - The number of clusters.
  - **-t TABLE** - The name of table used to store image features.
  - **-meta PATH** - The path of metadata for labeled images.
  - **-test(optional) Test mode** - The program will print the accuracy, the meta has to be the complete one.
  - **-k(optional)** - of latent semantics: The default value is 10. It is usually set in test mode to show the effect of k.

  Example:
  ```
  python p3task2.py -p ~/hw/cse515_data/phase3_sample_data/Labelled/Set1/
  -unp ~/hw/cse515_data/phase3_sample_data/Unlabelled/Set1 -k 50 -c 5 -t
  task12 -meta ~/hw/cse515_data/phase3_sample_data/labelled_set1.csv
  ```

- Phase 3 task 3
  - **-k K** - The number of outgoing edges.
  - **-lk K** - Most K dominant images.
  - **-t TABLE** - The name of table used to store image features.
  - **-i PATH** - The path of input folder.
  - **-lst ID1,ID2,ID3** - The ID of 3 query images.

  Example:
  ```
  python p3task3.py -k 10 -lk 5 -t set2 -i ~/hw/cse515_data/phase3_sample_data/Labelled/Set2/
  -lst Hand_0008333.jpg,Hand_0006183.jpg,Hand_0000074.jpg
  ```

- Phase 3 task 4 Some arguments for task4:
  - **-c CLASSIFIER** - The classifier will be used.
    - **svm** - Support Vector Machine
    - **dtree** - Decision Tree
    - **ppr** - Personalized Page Rank
  - **-m MODEL(optional)** - Feature extraction model. We support the following feature extraction model.
    - **cavg** - color first moment
    - **cm** - color moment
    - **hog** - histograms of oriented gradients
  - **-t TABLE(optional)** - The name that has been used when creating descriptor database.

  - **-ut TABLE(optional)** - the unlabeled table.
  - **-k K(optional)** - the number of latent semantics.
  - **-d METHOD(optional)** - The method will be used to reduce dimensions.
    * **svd** - Singular value decomposition
    * **pca** -Principal component analysis
  - **-l PATH (optional)** - labeled image folder path
  - **-u PATH** - unlabeled image folder path
  - **-meta PATH(.csv) (optional)** - the path of metadata for labeled images.
  - **-tmeta PATH(.csv) (optional)** - for unlabeled / test images folder. This is optional. If we can provide test label metadata, this task will show the accuracy.
  - **-limg PATH (optional)** - Labled raw image path if you do not want to use feature extraction.
  - **-uimg PATH (optional)** - Unlabeled raw image path if you do not want to use feature extraction.
  - –**svm_pretrained(optional)**: SVM will save its weight to svm/ folder.
  - –**svm_pretrained PATH(optional)**: SVM will load the weight and will NOT adjust its weight later.

  Example:
  ```
  python p3task4.py -c svm -m hog -t set1 -d pca -k 20 -ut tSet1 -meta
  labelled_set1.csv -tmeta unlabelled_set1.csv
  ```

- Phase 3 task 5
  - **-l LAYERS** - The number of layers.
  - **-k HASHES** - The number of hashes per layer.
  - **-i ID** - The ID of query image.
  - **-t T** - The number of most similar images.
  - **-tb TABLE** - The name of table used to store image features.
  - **-d METHOD** - The method will be used to reduce dimensions.
  - **-dir PATH** - The path of images.

  Example:
  ```
  python p3task5.py -l 5 -k 10 -t 20 -i Hand_0000674 -tb 11k -d hog -dir
  /hw/cse515_data/Hands/
  ```

- Phase 3 task 6 Please run p3task5.py first before p3task6.py.
  - **-c CLASSIFIER** - The table name which would be created in database.
  - **-m MODEL** - The target subject ID.
  - **-t TABLE** - The metadata path
  - **-d METHOD** - The image path
  - **-k K** - The top k latent semantics.

  Example:
  ```
  python p3task6.py -c svm -m cavg -d pca -k 20 -t 11k
  ```

5 Query output

5.1 Task1

In each query, we also show the accuracy of each . The final results is the overall accuracy.

5.1.1 Query1

Overall accuracy is 0.46
CM accuracy is 0.69; HOG accuracy is 0.39

Dorsal	Hand_0000972, Hand_0000989, Hand_0006637, Hand_0006638, Hand_0006639, Hand_0006640, Hand_0007734, Hand_0007735, Hand_0007736, Hand_0007877, Hand_0007878, Hand_0007879, Hand_0007880, Hand_0007881, Hand_0007882, Hand_0007883, Hand_0007884, Hand_0007885, Hand_0007886, Hand_0007887, Hand_0007888, Hand_0007916
Palmar	Hand_0000674, Hand_0000675, Hand_0000676, Hand_0000677, Hand_0000678, Hand_0000679, Hand_0000680, Hand_0000681, Hand_0000941, Hand_0000942, Hand_0000950, Hand_0000951, Hand_0000952, Hand_0000953, Hand_0000954, Hand_0000955, Hand_0000956, Hand_0000957, Hand_0000958, Hand_0000959, Hand_0000970, Hand_0000971, Hand_0000973, Hand_0000993, Hand_0000994, Hand_0000995, Hand_0000996, Hand_0000997, Hand_0000998, Hand_0000999, Hand_0006641, Hand_0006642, Hand_0006643, Hand_0006644, Hand_0006645, Hand_0006646, Hand_0006647, Hand_0006648, Hand_0006649, Hand_0006650, Hand_0006651, Hand_0007779, Hand_0007780, Hand_0007781, Hand_0007782, Hand_0007783, Hand_0007784, Hand_0007785, Hand_0007786, Hand_0007787, Hand_0007788, Hand_0007789, Hand_0007790, Hand_0007791, Hand_0007792, Hand_0007793, Hand_0007794, Hand_0007795, Hand_0007796, Hand_0007797, Hand_0007798, Hand_0007799, Hand_0007900, Hand_0007901, Hand_0007902, Hand_0007903, Hand_0007904, Hand_0007905, Hand_0007906, Hand_0007907, Hand_0007908, Hand_0007909, Hand_0007910, Hand_0007911, Hand_0007912, Hand_0007913, Hand_0007914, Hand_0007915

5.1.2 Query2

Overall accuracy is 0.54
CM accuracy is 0.53; HOG accuracy is 0.48

Dorsal	Hand_0000070, Hand_0000075, Hand_0000648, Hand_0000650, Hand_0000776, Hand_0000902, Hand_0000903, Hand_0000915, Hand_0001930, Hand_0001931, Hand_0003137, Hand_0003138, Hand_0004245, Hand_0004572, Hand_0004632, Hand_0005562, Hand_0005578, Hand_0005984, Hand_0005985, Hand_0005986, Hand_0006401, Hand_0007166, Hand_0007168, Hand_0008014, Hand_0008016, Hand_0008017, Hand_0008886, Hand_0009376, Hand_0009377, Hand_0009378, Hand_0009653, Hand_0009654, Hand_0009657, Hand_0010833
Palmar	Hand_0000094, Hand_0000095, Hand_0000101, Hand_0000102, Hand_0000109, Hand_0000111, Hand_0000112, Hand_0000144, Hand_0000145, Hand_0000203, Hand_0000204, Hand_0000205, Hand_0000386, Hand_0000387, Hand_0000403, Hand_0000404, Hand_0000422, Hand_0000423, Hand_0000658, Hand_0000659, Hand_0000774, Hand_0001320, Hand_0001321, Hand_0001828, Hand_0001829, Hand_0001943, Hand_0001944, Hand_0002182, Hand_0002183, Hand_0002184, Hand_0002212, Hand_0002213, Hand_0002214, Hand_0002719, Hand_0002720, Hand_0003388, Hand_0003389, Hand_0003555, Hand_0003556, Hand_0003558, Hand_0004026, Hand_0004631, Hand_0004680, Hand_0004681, Hand_0004945, Hand_0004946, Hand_0005162, Hand_0005163, Hand_0005579, Hand_0006004, Hand_0006005, Hand_0006575, Hand_0006863, Hand_0006864, Hand_0008622, Hand_0008628, Hand_0008885, Hand_0009686, Hand_0009687, Hand_0009791, Hand_0009810, Hand_0010471, Hand_0010474, Hand_0010834, Hand_0011455, Hand_0011726

5.1.3 Query3

Overall accuracy is 0.6
CM accuracy is 0.57; HOG accuracy is 0.57

Dorsal	Hand_0000674, Hand_0000941, Hand_0000942, Hand_0000950, Hand_0000951, Hand_0000952, Hand_0000953, Hand_0000954, Hand_0000955, Hand_0000956, Hand_0000957, Hand_0000958, Hand_0000959, Hand_0000989, Hand_0000993, Hand_0000994, Hand_0000995, Hand_0000996, Hand_0000997, Hand_0000998, Hand_0000999, Hand_0006637, Hand_0006638, Hand_0006639, Hand_0006640, Hand_0006641, Hand_0006642, Hand_0006643, Hand_0006644, Hand_0006645, Hand_0006646, Hand_0006647, Hand_0006648, Hand_0006649, Hand_0006650, Hand_0006651, Hand_0007734, Hand_0007735, Hand_0007736, Hand_0007779, Hand_0007780, Hand_0007781, Hand_0007787, Hand_0007788, Hand_0007789, Hand_0007790, Hand_0007791, Hand_0007792, Hand_0007793, Hand_0007794, Hand_0007795, Hand_0007900, Hand_0007901, Hand_0007902, Hand_0007903, Hand_0007904, Hand_0007905, Hand_0007906, Hand_0007907, Hand_0007908, Hand_0007909, Hand_0007910, Hand_0007911, Hand_0007912, Hand_0007913, Hand_0007914, Hand_0007915, Hand_0007916
Palmar	Hand_0000675, Hand_0000676, Hand_0000677, Hand_0000678, Hand_0000679, Hand_0000680, Hand_0000681, Hand_0000970, Hand_0000971, Hand_0000972, Hand_0000973, Hand_0007782, Hand_0007783, Hand_0007784, Hand_0007785, Hand_0007786, Hand_0007796, Hand_0007797, Hand_0007798, Hand_0007799, Hand_0007877, Hand_0007878, Hand_0007879, Hand_0007880, Hand_0007881, Hand_0007882, Hand_0007883, Hand_0007884, Hand_0007885, Hand_0007886, Hand_0007887, Hand_0007888

5.1.4 Query4

Overall accuracy is 0.71
CM accuracy is 0.67; HOG accuracy is 0.68

Dorsal	Hand_0000070, Hand_0000075, Hand_0000109, Hand_0000111, Hand_0000112, Hand_0000203, Hand_0000205, Hand_0000387, Hand_0000403, Hand_0000404, Hand_0000648, Hand_0000650, Hand_0000658, Hand_0000659, Hand_0000774, Hand_0000776, Hand_0000902, Hand_0000903, Hand_0000915, Hand_0001930, Hand_0001931, Hand_0002182, Hand_0002183, Hand_0002184, Hand_0002212, Hand_0002213, Hand_0002214, Hand_0003137, Hand_0003138, Hand_0003555, Hand_0003556, Hand_0003558, Hand_0004026, Hand_0004631, Hand_0004632, Hand_0004680, Hand_0004681, Hand_0005162, Hand_0005163, Hand_0005562, Hand_0005984, Hand_0005985, Hand_0005986, Hand_0006004, Hand_0006005, Hand_0006401, Hand_0006575, Hand_0006863, Hand_0006864, Hand_0007166, Hand_0007168, Hand_0008014, Hand_0008016, Hand_0008017, Hand_0008622, Hand_0008628, Hand_0008885, Hand_0008886, Hand_0009376, Hand_0009377, Hand_0009378, Hand_0009653, Hand_0009654, Hand_0009657, Hand_0009686, Hand_0009687, Hand_0010474
Palmar	Hand_0000094, Hand_0000095, Hand_0000101, Hand_0000102, Hand_0000144, Hand_0000145, Hand_0000204, Hand_0000386, Hand_0000422, Hand_0000423, Hand_0001320, Hand_0001321, Hand_0001828, Hand_0001829, Hand_0001943, Hand_0001944, Hand_0002719, Hand_0002720, Hand_0003388, Hand_0003389, Hand_0004245, Hand_0004572, Hand_0004945, Hand_0004946, Hand_0005578, Hand_0005579, Hand_0009791, Hand_0009810, Hand_0010471, Hand_0010833, Hand_0010834, Hand_0011455, Hand_0011726

5.1.5 Explanations

First, the overall trend is that the accuracy is higher and higher. Looking into the datasets, we find if the labeled images are homogeneous, the accuracy will be low. Such phenomena often happens in machine learning algorithms when the training data is biased. On the other hand, if the unlabeled images are homogeneous, the system could repeat error which can also leads to low accuracy. Second, the combination of two different descriptors does improve the accuracy a little compared to using them separately. Besides, we also do some experiments with different k and find that a smaller k like 10 can improve some queries' accuracy further.

5.2 Task2

The k-means clustering algorithm is randomized, so the accuracy are shown as a range from more than ten executions. The textual results come from one-time execution. The number of latent semantics used in the following queries is ten.

5.2.1 Query1

Overall accuracy is 0.69-0.79
The dorsal clusters are:

Hand_0002367, Hand_0002368, Hand_0002422, Hand_0005855, Hand_0005856, Hand_0005857
Hand_0003651, Hand_0003652, Hand_0003672, Hand_0004172, Hand_0004173, Hand_0004174, Hand_0005661, Hand_0005662, Hand_0005663, Hand_0008346, Hand_0008347
Hand_0011283, Hand_0011284, Hand_0011285
Hand_0004563, Hand_0004564, Hand_0004565, Hand_0007737, Hand_0007738, Hand_0007739, Hand_0008126, Hand_0008128, Hand_0008129, Hand_0008393, Hand_0008394, Hand_0008395, Hand_0008586, Hand_0008587, Hand_0008588
Hand_0000072, Hand_0000073, Hand_0000074, Hand_0006182, Hand_0006183, Hand_0006184, Hand_0006843, Hand_0006844, Hand_0006845, Hand_0008176, Hand_0008178, Hand_0008179, Hand_0008333, Hand_0008334, Hand_0008335

The palmar clusters are:

Hand_0000087, Hand_0000088, Hand_0000089, Hand_0000850, Hand_0000851, Hand_0000852, Hand_0007994, Hand_0007995, Hand_0007996, Hand_0011739, Hand_0011740
Hand_0000491, Hand_0000492, Hand_0000493, Hand_0002195, Hand_0002196, Hand_0002197
Hand_0002991, Hand_0002992, Hand_0002993, Hand_0003174, Hand_0004163, Hand_0004164, Hand_0004165, Hand_0004883, Hand_0004884, Hand_0004885, Hand_0007401, Hand_0007402, Hand_0007403, Hand_0007432, Hand_0007433, Hand_0007434, Hand_0009052, Hand_0009053, Hand_0009054, Hand_0010210
Hand_0010188, Hand_0010189, Hand_0010190
Hand_0002433, Hand_0002434, Hand_0002968, Hand_0002969, Hand_0003456, Hand_0003457, Hand_0003458, Hand_0008956, Hand_0008959, Hand_0008960

The result of the unlabeled image is shown as below.

Dorsal	Hand_0000674, Hand_0000676, Hand_0000677, Hand_0000678, Hand_0000679, Hand_0000680, Hand_0000681, Hand_0000942, Hand_0000951, Hand_0000952, Hand_0000953, Hand_0000954, Hand_0000955, Hand_0000956, Hand_0000958, Hand_0000971, Hand_0000972, Hand_0000989, Hand_0000993, Hand_0000994, Hand_0000995, Hand_0000996, Hand_0000997, Hand_0000998, Hand_0000999, Hand_0006637, Hand_0006638, Hand_0006639, Hand_0006640, Hand_0006641, Hand_0006642, Hand_0006643, Hand_0006644, Hand_0006645, Hand_0006646, Hand_0006647, Hand_0006648, Hand_0006649, Hand_0006650, Hand_0006651, Hand_0007735, Hand_0007736, Hand_0007877, Hand_0007900, Hand_0007901, Hand_0007902, Hand_0007903, Hand_0007904, Hand_0007905, Hand_0007906, Hand_0007907, Hand_0007908, Hand_0007909, Hand_0007910, Hand_0007911, Hand_0007912, Hand_0007913, Hand_0007914, Hand_0007915, Hand_0007916
Palmar	Hand_0000675, Hand_0000941, Hand_0000950, Hand_0000957, Hand_0000959, Hand_0000970, Hand_0000973, Hand_0007734, Hand_0007779, Hand_0007780, Hand_0007781, Hand_0007782, Hand_0007783, Hand_0007784, Hand_0007785, Hand_0007786, Hand_0007787, Hand_0007788, Hand_0007789, Hand_0007790, Hand_0007791, Hand_0007792, Hand_0007793, Hand_0007794, Hand_0007795, Hand_0007796, Hand_0007797, Hand_0007798, Hand_0007799, Hand_0007878, Hand_0007879, Hand_0007880, Hand_0007881, Hand_0007882, Hand_0007883, Hand_0007884, Hand_0007885, Hand_0007886, Hand_0007887, Hand_0007888

5.2.2 Query2

Overall accuracy is 0.7-0.79
The dorsal clusters are:

Hand_0005661, Hand_0005662, Hand_0005663, Hand_0008346, Hand_0008347
Hand_0002422, Hand_0005855, Hand_0005856, Hand_0005857
Hand_0011283, Hand_0011284, Hand_0011285
Hand_0004563, Hand_0004564, Hand_0004565, Hand_0007737, Hand_0007738, Hand_0007739, Hand_0008129, Hand_0008586, Hand_0008587, Hand_0008588
Hand_0008128, Hand_0008393, Hand_0008394, Hand_0008395
Hand_0006182, Hand_0006183, Hand_0006184
Hand_0003651, Hand_0003652, Hand_0003672, Hand_0004172, Hand_0004173, Hand_0004174
Hand_0002367, Hand_0002368
Hand_0000072, Hand_0000073, Hand_0000074, Hand_0008126, Hand_0008176, Hand_0008178, Hand_0008179, Hand_0008333, Hand_0008334, Hand_0008335
Hand_0006843, Hand_0006844, Hand_0006845

The palmar clusters are:

Hand_0000087, Hand_0000088, Hand_0000089, Hand_0000850, Hand_0000851, Hand_0000852
Hand_0011739, Hand_0011740
Hand_0004883, Hand_0004884, Hand_0004885, Hand_0009052, Hand_0009053, Hand_0009054
Hand_0007401, Hand_0007402, Hand_0007403
Hand_0010188, Hand_0010189, Hand_0010190
Hand_0007994, Hand_0007995, Hand_0007996
Hand_0008956, Hand_0008959, Hand_0008960
Hand_0002991, Hand_0002992, Hand_0002993, Hand_0003174, Hand_0004163, Hand_0004164, Hand_0004165, Hand_0007432, Hand_0007433, Hand_0007434, Hand_0010210
Hand_0000491, Hand_0000492, Hand_0000493, Hand_0002195, Hand_0002196, Hand_0002197
Hand_0002433, Hand_0002434, Hand_0002968, Hand_0002969, Hand_0003456, Hand_0003457, Hand_0003458

The result of unlabeled image are shown as below.

Dorsal	Hand_0000674, Hand_0000675, Hand_0000676, Hand_0000677, Hand_0000678, Hand_0000679, Hand_0000680, Hand_0000681, Hand_0000941, Hand_0000942, Hand_0000950, Hand_0000951, Hand_0000952, Hand_0000953, Hand_0000954, Hand_0000955, Hand_0000956, Hand_0000957, Hand_0000958, Hand_0000971, Hand_0000972, Hand_0000989, Hand_0000993, Hand_0000994, Hand_0000995, Hand_0000996, Hand_0000997, Hand_0000998, Hand_0000999, Hand_0006637, Hand_0006638, Hand_0006639, Hand_0006640, Hand_0006641, Hand_0006642, Hand_0006643, Hand_0006644, Hand_0006645, Hand_0006646, Hand_0006647, Hand_0006648, Hand_0006649, Hand_0006650, Hand_0006651, Hand_0007734, Hand_0007735, Hand_0007736, Hand_0007877, Hand_0007878, Hand_0007900, Hand_0007901, Hand_0007902, Hand_0007903, Hand_0007904, Hand_0007905, Hand_0007906, Hand_0007907, Hand_0007908, Hand_0007909, Hand_0007910, Hand_0007911, Hand_0007912, Hand_0007913, Hand_0007914, Hand_0007915, Hand_0007916
Palmar	Hand_0000959, Hand_0000970, Hand_0000973, Hand_0007779, Hand_0007780, Hand_0007781, Hand_0007782, Hand_0007783, Hand_0007784, Hand_0007785, Hand_0007786, Hand_0007787, Hand_0007788, Hand_0007789, Hand_0007790, Hand_0007791, Hand_0007792, Hand_0007793, Hand_0007794, Hand_0007795, Hand_0007796, Hand_0007797, Hand_0007798, Hand_0007799, Hand_0007879, Hand_0007880, Hand_0007881, Hand_0007882, Hand_0007883, Hand_0007884, Hand_0007885, Hand_0007886, Hand_0007887, Hand_0007888

5.2.3 Query3

Overall accuracy is 0.69-0.74
The dorsal clusters are:

Hand_0002367, Hand_0002368, Hand_0002422
Hand_0011283, Hand_0011284, Hand_0011285
Hand_0004563, Hand_0004564, Hand_0004565, Hand_0007737, Hand_0007738, Hand_0007739, Hand_0008126, Hand_0008128, Hand_0008129, Hand_0008393, Hand_0008394, Hand_0008395, Hand_0008586, Hand_0008587, Hand_0008588
Hand_0000072, Hand_0000073, Hand_0000074, Hand_0006182, Hand_0006183, Hand_0006184, Hand_0006843, Hand_0006844, Hand_0006845, Hand_0008176, Hand_0008178, Hand_0008179, Hand_0008333, Hand_0008334, Hand_0008335
Hand_0003651, Hand_0003652, Hand_0003672, Hand_0004172, Hand_0004173, Hand_0004174, Hand_0005661, Hand_0005662, Hand_0005663, Hand_0005855, Hand_0005856, Hand_0005857, Hand_0008346, Hand_0008347

The palmar clusters are:

Hand_0010188, Hand_0010189, Hand_0010190
Hand_0007994, Hand_0007995, Hand_0007996
Hand_0000491, Hand_0000492, Hand_0000493, Hand_0002968, Hand_0002969, Hand_0003456, Hand_0003457, Hand_0003458
Hand_0002195, Hand_0002196, Hand_0002197, Hand_0002433, Hand_0002434, Hand_0008956, Hand_0008959, Hand_0008960
Hand_0000087, Hand_0000088, Hand_0000089, Hand_0000850, Hand_0000851, Hand_0000852, Hand_0002991, Hand_0002992, Hand_0002993, Hand_0003174, Hand_0004163, Hand_0004164, Hand_0004165, Hand_0004883, Hand_0004884, Hand_0004885, Hand_0007401, Hand_0007402, Hand_0007403, Hand_0007432, Hand_0007433, Hand_0007434, Hand_0009052, Hand_0009053, Hand_0009054, Hand_0010210, Hand_0011739, Hand_0011740

The result of unlabeled image are shown as below.

Dorsal	Hand_0000075, Hand_0000094, Hand_0000095, Hand_0000109, Hand_0000111, Hand_0000112, Hand_0000204, Hand_0000205, Hand_0000403, Hand_0000404, Hand_0000648, Hand_0000650, Hand_0000658, Hand_0000659, Hand_0000774, Hand_0000776, Hand_0000902, Hand_0000903, Hand_0000915, Hand_0001828, Hand_0001829, Hand_0001930, Hand_0001931, Hand_0003137, Hand_0003138, Hand_0004631, Hand_0004632, Hand_0004680, Hand_0004681, Hand_0006575, Hand_0008014, Hand_0008016, Hand_0008017, Hand_0009376, Hand_0009377, Hand_0009378, Hand_0009653, Hand_0009654, Hand_0009657, Hand_0009686, Hand_0009687, Hand_0009810, Hand_0010471, Hand_0010474, Hand_0010833, Hand_0011726
Palmar	Hand_0000070, Hand_0000101, Hand_0000102, Hand_0000144, Hand_0000145, Hand_0000203, Hand_0000386, Hand_0000387, Hand_0000422, Hand_0000423, Hand_0001320, Hand_0001321, Hand_0001943, Hand_0001944, Hand_0002182, Hand_0002183, Hand_0002184, Hand_0002212, Hand_0002213, Hand_0002214, Hand_0002719, Hand_0002720, Hand_0003388, Hand_0003389, Hand_0003555, Hand_0003556, Hand_0003558, Hand_0004026, Hand_0004245, Hand_0004572, Hand_0004945, Hand_0004946, Hand_0005162, Hand_0005163, Hand_0005562, Hand_0005578, Hand_0005579, Hand_0005984, Hand_0005985, Hand_0005986, Hand_0006004, Hand_0006005, Hand_0006401, Hand_0006863, Hand_0006864, Hand_0007166, Hand_0007168, Hand_0008622, Hand_0008628, Hand_0008885, Hand_0008886, Hand_0009791, Hand_0010834, Hand_0011455

5.2.4 Query4

Overall accuracy is 0.6-0.71
The dorsal clusters are:

Hand_0006182, Hand_0006183, Hand_0006184
Hand_0003651, Hand_0003652, Hand_0003672, Hand_0005661, Hand_0005662, Hand_0005663, Hand_0008346, Hand_0008347
Hand_0004563, Hand_0004564, Hand_0004565, Hand_0008126, Hand_0008129
Hand_0004172, Hand_0004173, Hand_0004174, Hand_0005855, Hand_0005856, Hand_0005857
Hand_0000072, Hand_0000073, Hand_0000074
Hand_0011283, Hand_0011284, Hand_0011285
Hand_0002367, Hand_0002368, Hand_0002422
Hand_0008128, Hand_0008393, Hand_0008394, Hand_0008395
Hand_0007737, Hand_0007738, Hand_0007739, Hand_0008176, Hand_0008178, Hand_0008179, Hand_0008333, Hand_0008334, Hand_0008335, Hand_0008586, Hand_0008587, Hand_0008588
Hand_0006843, Hand_0006844, Hand_0006845

The palmar clusters are:

Hand_0000491, Hand_0000492, Hand_0000493
Hand_0002433, Hand_0002434, Hand_0002968, Hand_0002969, Hand_0003456, Hand_0003457, Hand_0003458
Hand_0007994, Hand_0007995, Hand_0007996
Hand_0008956, Hand_0008959, Hand_0008960
Hand_0010188, Hand_0010189, Hand_0010190
Hand_0004883, Hand_0004884, Hand_0004885, Hand_0009052, Hand_0009053, Hand_0009054
Hand_0004163, Hand_0004164, Hand_0004165, Hand_0007432, Hand_0007433, Hand_0007434, Hand_0010210
Hand_0002195, Hand_0002196, Hand_0002197
Hand_0002991, Hand_0002992, Hand_0002993, Hand_0003174, Hand_0007401, Hand_0007402, Hand_0007403, Hand_0011739, Hand_0011740
Hand_0000087, Hand_0000088, Hand_0000089, Hand_0000850, Hand_0000851, Hand_0000852

The result of the unlabeled image are shown as below.

Dorsal	Hand_0000075, Hand_0000094, Hand_0000095, Hand_0000109, Hand_0000111, Hand_0000112, Hand_0000203, Hand_0000204, Hand_0000205, Hand_0000403, Hand_0000404, Hand_0000648, Hand_0000650, Hand_0000658, Hand_0000659, Hand_0000774, Hand_0000776, Hand_0000902, Hand_0000903, Hand_0000915, Hand_0001828, Hand_0001829, Hand_0001930, Hand_0001931, Hand_0001943, Hand_0001944, Hand_0003137, Hand_0003138, Hand_0004026, Hand_0004245, Hand_0004572, Hand_0004631, Hand_0004632, Hand_0004680, Hand_0004681, Hand_0006575, Hand_0008014, Hand_0008016, Hand_0008017, Hand_0009376, Hand_0009377, Hand_0009378, Hand_0009653, Hand_0009654, Hand_0009657, Hand_0009686, Hand_0009687, Hand_0009791, Hand_0009810, Hand_0010471, Hand_0010474, Hand_0010833, Hand_0011726
Palmar	Hand_0000070, Hand_0000101, Hand_0000102, Hand_0000144, Hand_0000145, Hand_0000386, Hand_0000387, Hand_0000422, Hand_0000423, Hand_0001320, Hand_0001321, Hand_0002182, Hand_0002183, Hand_0002184, Hand_0002212, Hand_0002213, Hand_0002214, Hand_0002719, Hand_0002720, Hand_0003388, Hand_0003389, Hand_0003555, Hand_0003556, Hand_0003558, Hand_0004945, Hand_0004946, Hand_0005162, Hand_0005163, Hand_0005562, Hand_0005578, Hand_0005579, Hand_0005984, Hand_0005985, Hand_0005986, Hand_0006004, Hand_0006005, Hand_0006401, Hand_0006863, Hand_0006864, Hand_0007166, Hand_0007168, Hand_0008622, Hand_0008628, Hand_0008885, Hand_0008886, Hand_0010834, Hand_0011455

5.2.5 Explanations

Although the data sets used in each query are quiet different from each other, it is hard to tell the differences of accuracy from query to query. Actually, it is the number of latent semantics that affects the accuracy. If the unlabeled images is set1(in query1 and query2) which looks homogeneous, a smaller k is better than a large k. On the other hand, if the unlabeled images is set2(in query3 and query4) which looks heterogeneous, a large k is better than a small k. The reason could be that extra latent semantics are noises for the homogeneous images but are useful information for the heterogeneous images. In addition, if k is two small like one, the accuracy of both cases are less than fifty percents. If k is too large like forty, there is also no benefit gained or it can even deteriorate the accuracy a little bit. Finally, The effect of changing the number c from five to ten is negligible when only considering average accuracy. However, a large c can make accuracy vary abruptly from execution to execution.

5.3 Task 3

We execute the two queries and find top 10 dominate images. As instructed for each image there are 5 outgoing probabilistic out going edges. First query contains three images "Hand_0008333.jpg","Hand_0006183.jpg" and "Hand_0000074.jpg". Second query contains images "Hand_0003457.jpg", "Hand_0000074.jpg", and "Hand_0005661.jpg". Following is the result for both queries. We use **Color Moment** as our feature description model. Following image table shows the output for both two queries.

(a) Hand_0008333.jpg

(b) Hand_0006183.jpg

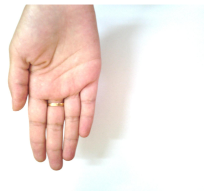
(c) Hand_0000074.jpg

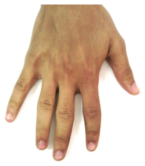
(d) Hand_0008588.jpg

(e) Hand_0008587.jpg

(f) Hand_0008334.jpg

(g) Hand_0008335.jpg


(h) Hand_0000852.jpg


(i) Hand_0008178.jpg

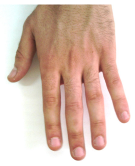
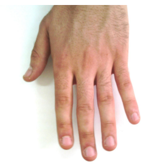
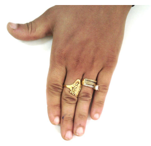
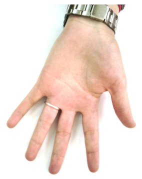
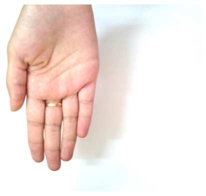

(j) Hand_0000072.jpg

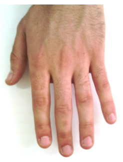
Figure 1: Task 3: Query 1. Image IDs: Hand_0008333.jpg, Hand_0006183.jpg, Hand_0000074.jpg. k: 5, K : 10. Folder : Labelled/Set 2

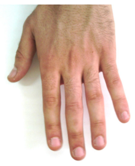

(a) Hand_0000074.jpg

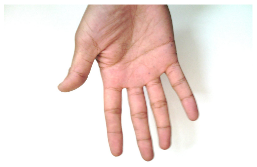

(b) Hand_0005661.jpg

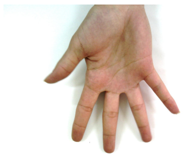

(c) Hand_0003457.jpg

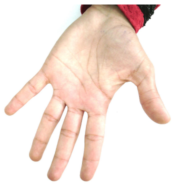

(d) Hand_0000852.jpg


(e) Hand_0000073.jpg


(f) Hand_0000072.jpg


(g) Hand_0000087.jpg


(h) Hand_0010210.jpg


(i) Hand_0002433.jpg

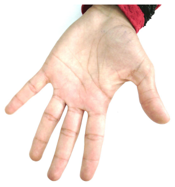

(j) Hand_0002434.jpg

Figure 2: Task 3: Query 2. Image IDs: Hand_0003457.jpg,Hand_0000074.jpg, Hand_0005661.jpg. k: 5, K : 10. Folder : Labelled/Set 2

5.4 Task 4

5.4.1 SVM classifier:

Since the weight of SVM is randomly assigned, the accuracy of two times training are usually huge different. Color Moment with first moment only(CAVG) and PCA are suitable for SVM. The best accuracy of training set 1 with testing set1 and training set 2 with testing set 2 are **85%** and **84%** - CAVG and PCA with K=100. However, the accuracy is possibly be low as **33%** and sometimes higher than **70%**.

The output of the sample query of dataset 2 is shown as below.

Dorsal	Hand_0000070, Hand_0000075, Hand_0000094, Hand_0000095, Hand_0000109, Hand_0000111, Hand_0000112, Hand_0000145, Hand_0000204, Hand_0000205, Hand_0000387, Hand_0000403, Hand_0000404, Hand_0000648, Hand_0000650, Hand_0000658, Hand_0000659, Hand_0000774, Hand_0000776, Hand_0000902, Hand_0000903, Hand_0000915, Hand_0002182, Hand_0002183, Hand_0002184, Hand_0003137, Hand_0003138, Hand_0003555, Hand_0003556, Hand_0003558, Hand_0004026, Hand_0004631, Hand_0004680, Hand_0004681, Hand_0006004, Hand_0006005, Hand_0006575, Hand_0006863, Hand_0006864, Hand_0007166, Hand_0007168, Hand_0008014, Hand_0008016, Hand_0008017, Hand_0009376, Hand_0009377, Hand_0009378, Hand_0009653, Hand_0009654, Hand_0009657, Hand_0009686, Hand_0009687, Hand_0009810, Hand_0010471, Hand_0010474, Hand_0011726
Palmar	Hand_0000101, Hand_0000102, Hand_0000144, Hand_0000203, Hand_0000386, Hand_0000422, Hand_0000423, Hand_0001320, Hand_0001321, Hand_0001828, Hand_0001829, Hand_0001930, Hand_0001931, Hand_0001943, Hand_0001944, Hand_0002212, Hand_0002213, Hand_0002214, Hand_0002719, Hand_0002720, Hand_0003388, Hand_0003389, Hand_0004245, Hand_0004572, Hand_0004632, Hand_0004945, Hand_0004946, Hand_0005162, Hand_0005163, Hand_0005562, Hand_0005578, Hand_0005579, Hand_0005984, Hand_0005985, Hand_0005986, Hand_0006401, Hand_0008622, Hand_0008628, Hand_0008885, Hand_0008886, Hand_0009791, Hand_0010833, Hand_0010834, Hand_0011455

5.4.2 Decision Tree classifier:

We use Decision Tree as our classifier. For testing set 2, we can achieve best accuracy of **82%**. We just use the Color Moment as the features of each labeled and unlabeled image and we do not use any dimension reduction methods. The lower accuracy with the dimension reductions may due to the information loss. In addition, the best result of the testing set 1 with Color Moment is **69%**. We also try other feature extraction methods but their performance are not better than Color Moment.

The output of the sample query of dataset 2 is shown as below.

Dorsal	Hand_0007168, Hand_0002212, Hand_0000404, Hand_0002213, Hand_0009378, Hand_0000403, Hand_0000774, Hand_0003137, Hand_0000776, Hand_0011726, Hand_0009654, Hand_0000658, Hand_0000659, Hand_0009657, Hand_0010474, Hand_0000075, Hand_0010471, Hand_0008014, Hand_0006005, Hand_0006004, Hand_0009653, Hand_0000111, Hand_0009686, Hand_0008017, Hand_0008016, Hand_0009687, Hand_0000112, Hand_0000648, Hand_0000094, Hand_0000902, Hand_0000095, Hand_0000650, Hand_0000109, Hand_0004681, Hand_0004680, Hand_0003138, Hand_0002184, Hand_0005562, Hand_0000387, Hand_0000386, Hand_0007166, Hand_0000145, Hand_0009377, Hand_0002183, Hand_0002182, Hand_0000144
Palmar	Hand_0009810, Hand_0000203, Hand_0011455, Hand_0008886, Hand_0001320, Hand_0001321, Hand_0008885, Hand_0010833, Hand_0002214, Hand_0000205, Hand_0000204, Hand_0005578, Hand_0010834, Hand_0009791, Hand_0005579, Hand_0000102, Hand_0008628, Hand_0003556, Hand_0004946, Hand_0000101, Hand_0001828, Hand_0003555, Hand_0001829, Hand_0004945, Hand_0006575, Hand_0006401, Hand_0000070, Hand_0004572, Hand_0002720, Hand_0000903, Hand_0004026, Hand_0001943, Hand_0008622, Hand_0000915, Hand_0003389, Hand_0001944, Hand_0003558, Hand_0004632, Hand_0003388, Hand_0002719, Hand_0004631, Hand_0005986, Hand_0005985, Hand_0006863, Hand_0004245, Hand_0005984, Hand_0000422, Hand_0005162, Hand_0005163, Hand_0000423, Hand_0001930, Hand_0006864, Hand_0001931, Hand_0009376

5.4.3 PPR classifier:

We experiments on PPR algorithm based classifier. For testing set 2 we can achieve best accuracy of **70%** with **k=6**, and **K=15**. In addition, the best accuracy for testing set 1 is **75%** with **k=68**, and **K=81**. We experimented all the combination of both hyper-parameters and all four feature description models, we found Color Moment describer model can achieve best accuracy among all other three methods. As mentioned previously, we didn't use any dimension reduction methods for PPR algorithm based classifier.

The output of the sample query of dataset 2 is shown as below.

Dorsal	Hand_0007168, Hand_0009810, Hand_0000203, Hand_0000404, Hand_0002213, Hand_0001320, Hand_0008885, Hand_0002214, Hand_0000403, Hand_0000205, Hand_0000204, Hand_0000776, Hand_0009654, Hand_0000658, Hand_0003556, Hand_0000659, Hand_0009657, Hand_0003555, Hand_0006575, Hand_0008014, Hand_0006005, Hand_0000070, Hand_0006004, Hand_0009653, Hand_0000111, Hand_0009686, Hand_0008017, Hand_0008016, Hand_0009687, Hand_0000112, Hand_0000648,Hand_0000902, Hand_0000903, Hand_0004026, Hand_0000650, Hand_0000915, Hand_0000109, Hand_0003389, Hand_0003558, Hand_0003388, Hand_0004631, Hand_0004681, Hand_0004680, Hand_0003138, Hand_0006863, Hand_0005562, Hand_0005984, Hand_0005162, Hand_0005163, Hand_0007166, Hand_0001930, Hand_0001931
Palmar	Hand_0002212, Hand_0011455, Hand_0008886, Hand_0009378, Hand_0001321, Hand_0010833, Hand_0000774, Hand_0003137, Hand_0005578, Hand_0010834, Hand_0009791, Hand_0005579, Hand_0011726, Hand_0000102, Hand_0008628, Hand_0004946, Hand_0000101, Hand_0010474, Hand_0001828, Hand_0000075, Hand_0001829, Hand_0004945, Hand_0010471, Hand_0006401, Hand_0004572, Hand_0002720, Hand_0000094, Hand_0000095, Hand_0001943, Hand_0008622, Hand_0001944, Hand_0004632, Hand_0002719, Hand_0005986, Hand_0005985, Hand_0004245, Hand_0002184, Hand_0000422, Hand_0000387, Hand_0000386, Hand_0000423, Hand_0000145, Hand_0009377, Hand_0002183, Hand_0002182, Hand_0006864, Hand_0009376, Hand_0000144

5.5 Task 5

The result of each query in task 5 seems the same because the input parameters are quite similar. Thus, we retrieve quite similar image. If we change the value of the parameter, we can obtain different result. For the result of the image, we can discover that the input image is dorsal and the output image are all dorsal. We use the extracted features from HOG and these features do not reflect the color of the data. Thus, we can discover that the color of dorsal hand may be vary but the shape of the hands are similar.

5.5.1 Query***

The result of Query*** is shown in Figure 3.

5.5.2 Query 2

The result of Query 2 is shown in Figure 4.

5.5.3 Query 3

The result of Query 3 is shown in Figure 5.

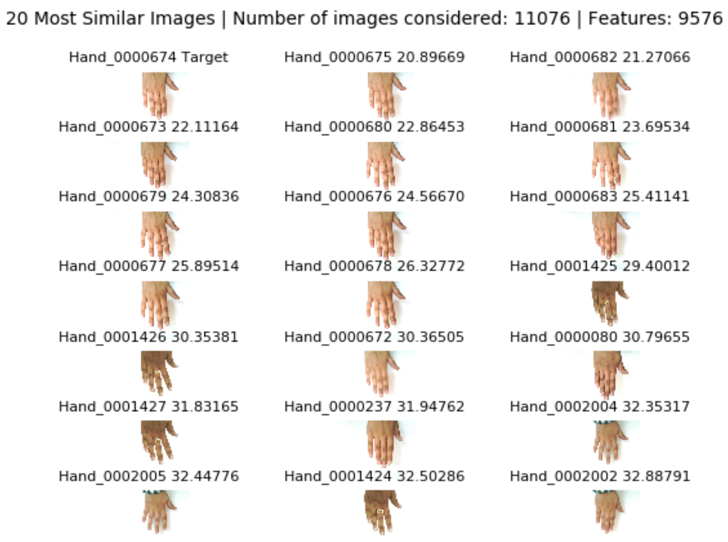


Figure 3: Query***

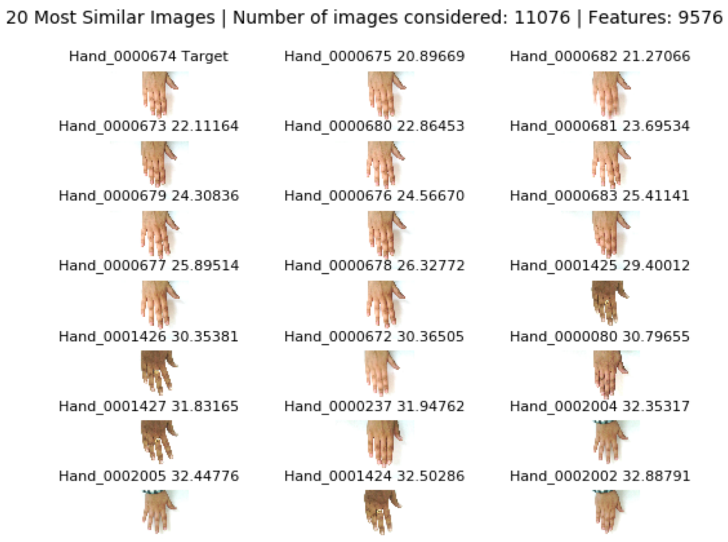


Figure 4: Query 2

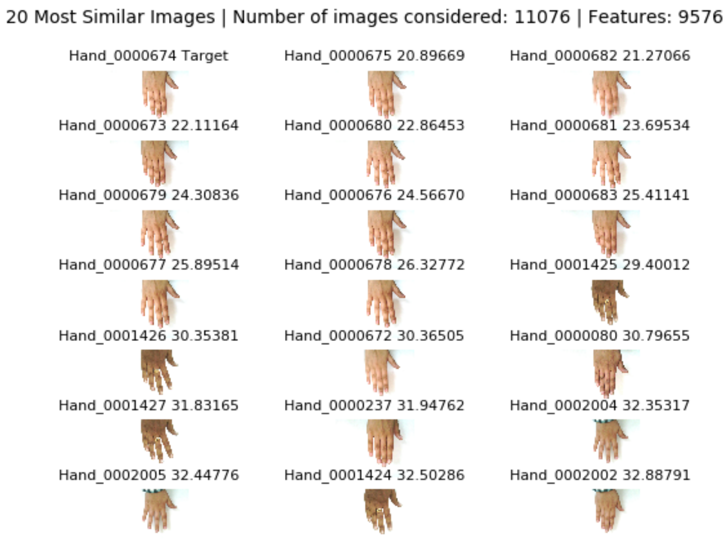


Figure 5: Query 3

5.6 Task 6

In this task, we have user input for relevant and irrelevant image with their image ID and they can also not give the label for the image.

User input:

Relevant	Hand_0000675, Hand_0000682, Hand_0000673, Hand_0000680, Hand_0000681, Hand_0000679, Hand_0000676
Irrelevant	Hand_0000683, Hand_0000677, Hand_0000678, Hand_0001425, Hand_0001426, Hand_0000672, Hand_0000080
Unknown	Hand_0001427, Hand_0000237, Hand_0002004, Hand_0002005, Hand_0001424, Hand_0002002

The result of the re-ranking is based on user input and we can discover that some of the model may modify the order of the original input. However, some of the model may not modify the order of the input. Each of the model has their own algorithm and it may be influenced by the labeling that the user give to the images. If the user label two pretty similar with different label, then the classifier may not partition the data into two space well. Thus, the result may be influenced by the label given by user, the number of relevant/irrelevant images.

5.6.1 Query 1 SVM Based

The result of re-ranking the image is as below showing from most relevant to less relevant.

Re-Rank Result
Hand_0000675
Hand_0000682
Hand_0000673
Hand_0000680
Hand_0000681
Hand_0000679
Hand_0000676
Hand_0000683
Hand_0000677
Hand_0000678
Hand_0001425
Hand_0001426
Hand_0000672
Hand_0000080
Hand_0001427
Hand_0000237
Hand_0002004
Hand_0002005
Hand_0001424
Hand_0002002

5.6.2 Query 2 Decision Tree Based

The result of re-ranking the image is as below showing from most relevant to less relevant.

Re-Rank Result
Hand_0000675
Hand_0000682
Hand_0000673
Hand_0000680
Hand_0000681
Hand_0000679
Hand_0000676
Hand_0000678
Hand_0000683
Hand_0000677
Hand_0001425
Hand_0001426
Hand_0000672
Hand_0000080
Hand_0001427
Hand_0000237
Hand_0002004
Hand_0002005
Hand_0001424
Hand_0002002

5.6.3 Query 3 PPR Based

The result of re-ranking the image is as below showing from most relevant to less relevant.

Re-Rank Result
Hand_0000675
Hand_0000682
Hand_0000673
Hand_0000680
Hand_0000681
Hand_0000679
Hand_0000676
Hand_0000683
Hand_0000677
Hand_0000678
Hand_0001425
Hand_0001426
Hand_0000672
Hand_0001427
Hand_0000237
Hand_0002004
Hand_0002005
Hand_0001424
Hand_0002002
Hand_0000080

5.6.4 Query 4 Probability Relevance Based

The result of re-ranking the image is as below showing from most relevant to less relevant.

Re-Rank Result
Hand_0000675
Hand_0000681
Hand_0000676
Hand_0000680
Hand_0000673
Hand_0000677
Hand_0000679
Hand_0000682
Hand_0000678
Hand_0001424
Hand_0000237
Hand_0002004
Hand_0002005
Hand_0002002
Hand_0001425
Hand_0001427
Hand_0000080
Hand_0000683
Hand_0001426
Hand_0000672

6 Conclusion

In phase 2, we can only use dimension reductions and latent semantic to determinate the class of query images - and the performance was poor. However, in this phase, we not only extend the method of latent semantics we tried in phase 2, but we also learned several types of method, such as learning-based SVM, probability-based LSH, which have better performance when compare to the phase 2. Even though these methods do well in a lot of fields, we still have to find the best parameters for each methods. Some methods are more deterministic, such as decision tree, and some are randomly, SVM for example. Implementation and experiment helped us to better understand how the model work and the restriction.

7 Bibliography

References

[1] Mahmoud Afifi. 11k hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, 78(15):20835–20854, 2019.

[2] Matt Brems. A one-stop shop for principal component analysis, Jun 2019.

[3] K. Seluk Candan and Maria Luisa Sapino. *Data Management for Multimedia Retrieval*. Cambridge University Press, New York, NY, USA, 2010.

[4] Lettier. Your guide to latent dirichlet allocation, May 2019.

[5] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[6] B. E. Prasad, A. Gupta, H. D. Toong, and S. E. Madnick. A microcomputer-based image database management system. *IEEE Transactions on Industrial Electronics*, IE-34(1):83–88, Feb 1987.

[7] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

Appendices

Roles of the group members

Jiaxuan Pang	Task3 Implementation. PPR classifier Implementation. Task 6 PPR module implementation. Report Writing.
Ting Xia	Propose solutions to task 6. Report Writing.
Ching-Wen Tu	Decision Tree Implementation. Report Writing.
Bandon Tseng	Task 4 and Task 6 implementation. SVM implementation. Report Writing.
Kuan-Ting Shen	Task 5 Implementation. LSH Implementation. Report Writing.
Ming Yi	Task1, Task2 and PRF in Task6 Implementation. Report Writing.