CSE 515 Group Project Phase 2

Preston Mott
Brandon Bayles
Ian Bolton
Keenan Rahman
Kunhao Zhang

Abstract

This phase of the project involved the team continuing to utilize the Olivetti Faces dataset of images to extract Color Moments (CM), Extended Binary Local Patterns (ELBP), and Histograms of Oriented Gradients (HOG) feature sets.[1] These are represented as vector models, and then further utilized to perform several tasks related to similarity matrices, dimensional reduction via Principal Component Analysis (PCA), Singular-Valued Decomposition (SVD), Latent Dirichlet Allocation (LDA), and K-Means, graph representation, and graphical node analysis via the ASCOS++ and Personalized PageRank (PPR) measures. With these goals in mind, the various tasks were allocated to the team and combined to produce our software.

Keywords

Feature Extraction, Dimensionality Reduction, Vector Representation, Latent Semantics, Graph Representation, Graph Node Analysis, Principal Component Analysis, PCA, Singular Value Decomposition, SVD, Latent Dirichlet Allocation, LDA, K-Means, Personalized PageRank, ASCOS++

Introduction

When working with high dimensional data that needs to be easily accessible and analyzable, it is important to search for ways to represent the data in fewer dimensions to avoid the dimensionality curse while minimizing the loss of information. This is an important topic for multimedia databases to handle, as properly dimensionally reduced data can be more efficiently stored and processed by recommendation and search algorithms whereas an improperly dimensionally reduced representation of the data may introduce errors and information loss. Several models that make more optimal dimensional reductions exist, including PCA, SVD, LDA, and k-means.

The goal of this phase of the project is to practice working with these dimensional reduction techniques, with an additional two tasks that also explore graph representation and graph node topology analysis.[1] Task 1 generates a collection of latent semantic subject-weight pairs based on the results of an input dimensional reduction model. Task 2 generates a collection of type-weight pairs in a similar manner. Task 3 creates a type-type similarity matrix and then performs an input dimensional reduction method on it to generate latent semantics in the form of type-weight pairs. Task 4 does the same but with a subject-subject similarity matrix and subject-weight pairs. Task 5 takes a query image and latent semantics file and then visualizes the most similar images from the database based on the latent semantics given by the file. Task 6 also takes a query image and latent semantics file but instead attempts to determine the most likely type of the image by utilizing the given latent semantics file. Task 7 once again takes a query image and latent semantics file but attempts to determine the subject ID of the query image based on a given latent semantics file. Task 8 takes a subject-subject similarity matrix and uses it to generate a similarity graph which is then analyzed for the most significant subjects based on the ASCOS++ feature.[1] Task 9 takes a subject-subject similarity matrix and several subject IDs and uses the matrix to generate a similarity graph which is then analyzed to determine the most significant objects relative to the input subject IDs based on the PPR measure.[1]

Assumptions for the overall project include that input images should be 64 by 64 pixels, greyscale, and stored as a PNG file. Furthermore, it is assumed that the runtime environment meets the specifications mentioned in the System Requirements section.

Description of the Proposed Solution/Implementation

Helper/Interface Files

This section describes code that was written by the group that was not necessarily part of a single task but was used throughout this phase of the project.

Dimensionality Reduction Models

PCA

The PCA file takes a k value and a NumPy array dataset of size n x p where n represents the number of data vectors, p represents the number of elements per data vector, and k represents the number of latent semantics to reduce the dimensions to. The program starts by normalizing the data by taking the mean of every column and then subtracting these means from the original data. The covariance matrix of the normalized data is then taken and used to extract the eigenvalues and eigenvectors. The covariance matrix is calculated using the following formula:

$$cov = \frac{(D^{-1} \cdot D)}{n-1}$$

Where *cov* is the covariance matrix, D represents the normalized data matrix and n represents the number of data vectors. The eigenvalues and eigenvectors are determined from this result by utilizing the scipy.linalg.eigh method. The eigenvalues are sorted in descending order and then the k highest associated eigenvectors are selected. The final dimensionally reduced data is found by projecting the selected eigenvectors onto the normalized data with a dot product calculation. The method then returns the dimensionally reduced data, the final selected eigenvectors, the sorted eigenvalues, and the sorted eigenvectors in a list. This code is referenced whenever a task needs to calculate a PCA decomposition.

SVD

The SVD implementation takes in an n x p matrix which represents the dataset; n being the number of data objects, and p being the number of features per object, and k, an integer corresponding to the number of latent semantics that the input matrix will be reduced to. The function generates two square matrices from the following calculations

$$DD^t = D \cdot D^t$$

$$D^tD = D^t \cdot D$$

Where D is the input n x p matrix and D^t is its transpose. From these, the eigenvalues and eigenvectors are extracted. These eigenvectors are presented in two matrices U and V^t, with dimensions n x n and p x p, respectively. U is the left factor matrix, where the n objects are described by the n latent features. Similarly, V^t, or the right factor matrix, describes the p latent features using the p old objects. The eigenvalues are placed in a n x p diagonal matrix S, called the core matrix. Because of the nature of SVD, the input matrix D may be reconstructed as follows:

$$D = U \cdot S \cdot V^t$$

The eigenvalues in S show the order and contribution of each eigenvector present in U and V^t. Thus, S is then sorted into descending order, as are the corresponding eigenvectors in U and V^t. Through this, we may eliminate those eigenvectors that contribute the least to our data matrix. With this, only the k latent semantics remain. As a final calculation, a projection of the data is computed with the following formula:

$$D' = U' \cdot S' \cdot V'^t$$

Where each of the right-side matrices are the dimensionally reduced equivalents to U, S, and V^t, respectively. Furthermore, this projection D' is approximately equivalent to the input matrix D, despite the eliminated features. With this, the program returns the reduced eigenvector matrices U' and V'^t (which now have dimensions n x k and k x p), the projection D', and the reduced sorted k x k matrix S'. This code is utilized for Singular-Value Decomposition throughout our project.

Database

For the database we use the latest version of MongoDb. Database implementation is done under the database.py file. We have a new database named “mwdb_database_phase_2”. This database has multiple tables that are used to contain data for different usage.

1. The 1st table “image_data” contains all the image matrices. It has 2 columns “_id” that has the image file name as id, and “image_matrix” which have the image matrix stored.
2. The 2nd table “matrix_similarity” is used by Task-3 and Task-4 to store subject-subject and type-type similarity matrices. These matrices are then used by Task-8 and Task-9.

3. The 3rd table “reduction_algorithm_output” is used by Task-1 and Task-2 to save the output of the reduction algorithm to be used by other tasks.

Graph

The graph implementation is being done under the graph.py file. This function is used to take a similarity matrix and a n-value as an input. It iterates through each of the nodes of a given graph, takes all the nodes going into or out of it and sorts them out based on the weights provided in the matrix. After sorting the values out in descending order, it takes into consideration the n-nearest nodes or n-nodes with the highest value. After iterating through all the nodes, it returns a graph, G (V, E), where V corresponds to the subjects in the database and E contains node pairs v_i, v_j such that, for each subject v_i , v_j is one of the n most similar subjects in the database. The graph returned by this file is used by Task-8 and Task-9 as input.

Feature Descriptors

Color Moments

Color moments are measures that can be used to differentiate images based on their features of color.[2] In order to calculate the color moments, three ways were used (mean, standard deviation, and skewness). The image matrix was divided into an 8x8 box to perform calculations. All these calculations are combined to calculate the color moments of a given image.

Mean: The first color moment can be interpreted as being the average color in the given image.[2] This is calculated by formula.[32]

$$E_i = \sum_{j=1}^{N} \frac{1}{N} p_{ij}$$

Here N is the total number of points in the given box and p_{ij} is the value of the ith row and jth column of the box.

Standard Deviation: The second color moment is obtained by taking the square root of the variance of the color distribution. This is calculated by formula.[2]

$$\sigma_i = \sqrt{\left(\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - E_i)^2\right)}$$

Here E_i is the mean value of the image.

Skewness: The third color moment measures how asymmetric the color distribution is and thus it gives information about the shape of the color distribution. This is calculated by formula.[2]

$$s_i = \sqrt[3]{\left(\frac{1}{N}\sum_{j=1}^{N}(p_{ij} - E_i)^3\right)}$$

Extended Local Binary Pattern

The extended LBP is a joint distribution of grayscale and rotational invariant LBP with the rotational invariant Variance measure.[4] In order to calculate the ELBP we set the local neighborhood to 8.00 and the radius is set to 1.00 on a given image vector. The formula for ELBP is [3]:

$$\text{LBP} = \sum_{i=0}^{P-1} s(n_i - G_c\Big)2^i$$

$$s(x) = \begin{cases} 1, & if\ x > 0 \\ 0, & otherwise \end{cases}$$

where P is the number of neighborhood pixels, n_i represents the ith neighboring pixel, and c represents the center pixel. The histogram features of size 2P are extracted from the obtained LBP code. [3]

Histogram of Oriented Gradients

The histogram of oriented gradients is a feature descriptor used in image processing. The technique counts occurrences of gradient orientation in localized portions of an image. In order to calculate the HOG, we set the number of orientation bins to 9, cell size to 8 and block size to 2. The block normalization is set to the L2-norm clipping threshold set to 0.2.

$$\text{L2-norm: } f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing.

For each of the models a separate function is created that takes image id and image matrix as input and provides the result for the corresponding model selected.

Task Description

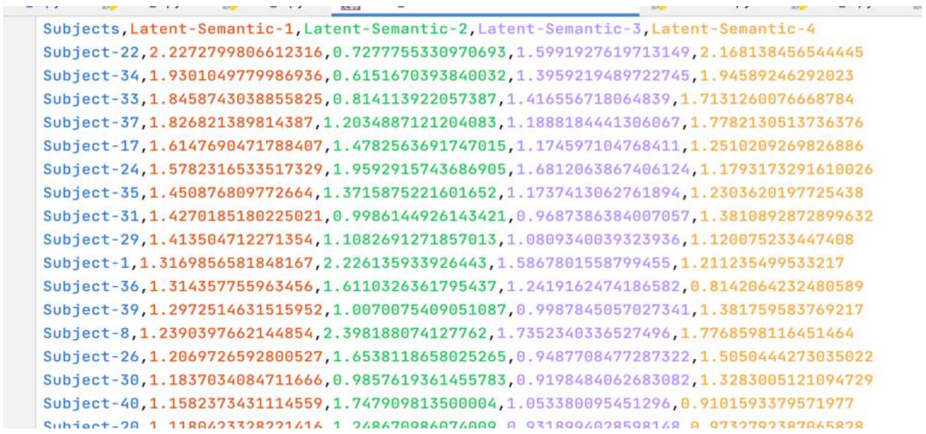
Task 1

In Task-1 we utilized all the feature models (color moments, histogram of oriented gradients, and extended local binary patterns) and dimensionality reduction algorithms (PCA, SVD, LDA, K-Means). This task made use of database.py to extract all the images of a given type X. This dictionary of images is then passed to a function get_object_features_matrix to convert the given dictionary into an object-feature matrix. This is done by first applying a feature extraction algorithm on a while iterating through all the images and then applying linear transformations to the given image. As a result, an image-feature matrix is generated with images as rows and feature values as columns.

After getting the object-feature matrix, this matrix is passed to dimensionality reduction algorithms; the description of each dimensionality reduction algorithm is given in the above headings. These algorithms give us k-latent semantics for the given objects. Now we group all the images of individual subjects together and calculate the average of each latent semantic of the given group of images. This results in a matrix of 40 Subjects as rows and k-Latent Semantics as columns.

These newly created subject-weight pairs are saved into the database and into the output folder named “output/task_1/”. We call this matrix a subject-weight pair.

Expected Output of Task



```
Subjects,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4
Subject-22,2.2272799806612316,0.7277755330970693,1.5991927619713149,2.168138456544445
Subject-34,1.9301049779986936,0.6151670393840032,1.3959219489722745,1.94589246292023
Subject-33,1.8458743038855825,0.814113922057387,1.416556718064839,1.7131260076668784
Subject-37,1.826821389814387,1.2034887121204083,1.1888184441306067,1.7782130513736376
Subject-17,1.6147690471788407,1.4782563691747015,1.174597104768411,1.2510209269826886
Subject-24,1.5782316533517329,1.9592915743686905,1.6812063867406124,1.1793173291610026
Subject-35,1.450876809772664,1.3715875221601652,1.1737413062761894,1.2303620197725438
Subject-31,1.4270185180225021,0.9986144926143421,0.9687386384007057,1.3810892872899632
Subject-29,1.413504712271354,1.1082691271857013,1.0809340039323936,1.120075233447408
Subject-1,1.3169856581848167,2.226135933926443,1.5867801558799455,1.211235499533217
Subject-36,1.314357755963456,1.6110326361795437,1.2419162474186582,0.8142064232480589
Subject-39,1.2972514631515952,1.0070075409051087,0.9987845057027341,1.381759583769217
Subject-8,1.2390397662144854,2.398188074127762,1.7352340336527496,1.7768598116451464
Subject-26,1.2069726592800527,1.6538118658025265,0.9487708477287322,1.5050444273035022
Subject-30,1.1837034084711666,0.9857619361455783,0.9198484062683082,1.3283005121094729
Subject-40,1.1582373431114559,1.747909813500004,1.053380095451296,0.9101593379571977
Subject-20,1.1180423328221416,1.2486709860740009,0.9318994028598148,0.9732792387065828
```

Sample Input

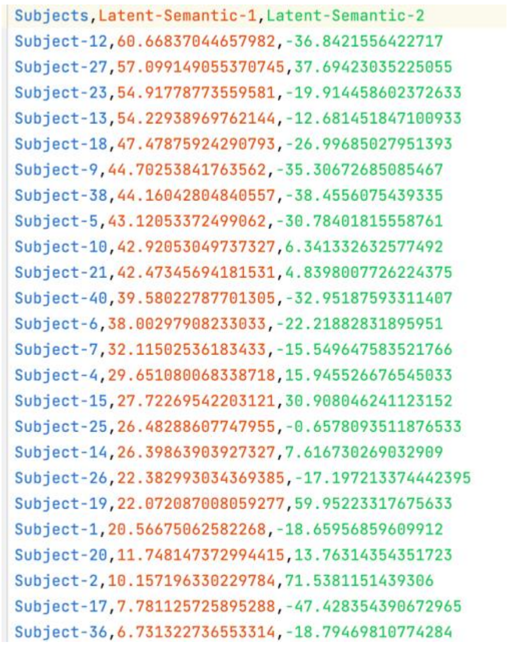
```
Please Enter Model Name: >? local_binary_pattern
```

```
Please Enter Value of X: >? stipple
```

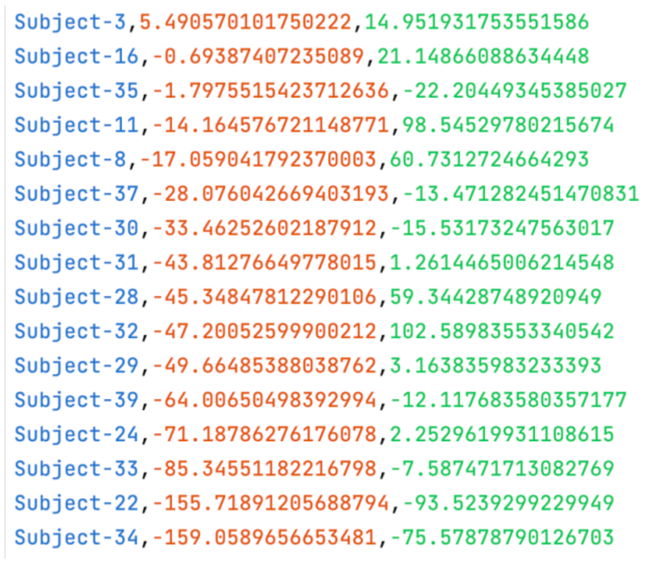
```
Please Enter K-Value: >? 2
```

```
Please Enter Model Name: >? PCA
```

Output



```
Subjects,Latent-Semantic-1,Latent-Semantic-2
Subject-12,60.66837044657982,-36.8421556422717
Subject-27,57.099149055370745,37.69423035225055
Subject-23,54.91778773559581,-19.914458602372633
Subject-13,54.22938969762144,-12.681451847100933
Subject-18,47.47875924290793,-26.99685027951393
Subject-9,44.70253841763562,-35.30672685085467
Subject-38,44.16042804840557,-38.4556075439335
Subject-5,43.12053372499062,-30.78401815558761
Subject-10,42.92053049737327,6.341332632577492
Subject-21,42.47345694181531,4.8398007726224375
Subject-40,39.58022787701305,-32.95187593311407
Subject-6,38.00297908233033,-22.21882831895951
Subject-7,32.11502536183433,-15.549647583521766
Subject-4,29.651080068338718,15.945526676545033
Subject-15,27.72269542203121,30.908046241123152
Subject-25,26.48288607747955,-0.6578093511876533
Subject-14,26.39863903927327,7.616730269032909
Subject-26,22.382993034369385,-17.197213374442395
Subject-19,22.072087008059277,59.95223317675633
Subject-1,20.56675062582268,-18.65956859609912
Subject-20,11.748147372994415,13.76314354351723
Subject-2,10.157196330229784,71.5381151439306
Subject-17,7.781125725895288,-47.428354390672965
Subject-36,6.731322736553314,-18.79469810774284
```



```
Subject-3,5.490570101750222,14.951931753551586
Subject-16,-0.69387407235089,21.14866088634448
Subject-35,-1.7975515423712636,-22.20449345385027
Subject-11,-14.164576721148771,98.54529780215674
Subject-8,-17.059041792370003,60.7312724664293
Subject-37,-28.076042669403193,-13.471282451470831
Subject-30,-33.46252602187912,-15.53173247563017
Subject-31,-43.81276649778015,1.2614465006214548
Subject-28,-45.34847812290106,59.34428748920949
Subject-32,-47.20052599900212,102.58983553340542
Subject-29,-49.66485388038762,3.163835983233393
Subject-39,-64.00650498392994,-12.117683580357177
Subject-24,-71.18786276176078,2.2529619931108615
Subject-33,-85.34551182216798,-7.587471713082769
Subject-22,-155.71891205688794,-93.5239299229949
Subject-34,-159.0589656653481,-75.57878790126703
```

Explanation

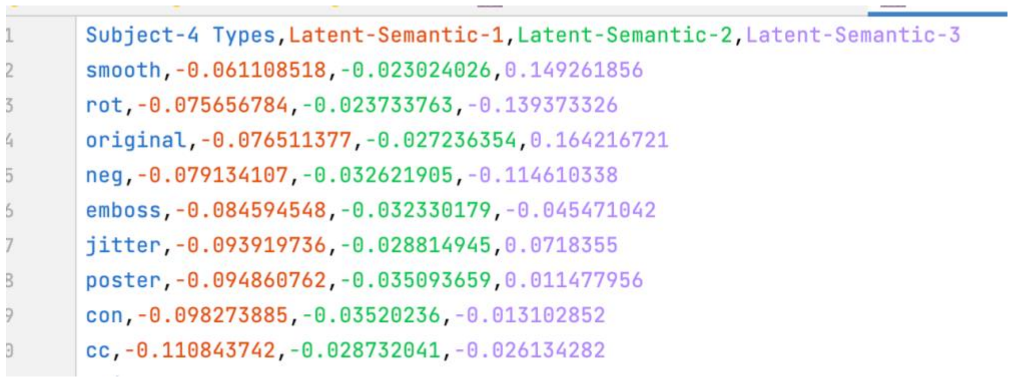
When we ran the task with local_binary_pattern on type “stipple” and dimensionality reduction PCA. It took all images of type “stipple” and applied the ELBP feature extractor on them. After this data is passed to the reduction algorithm to find k-latent semantics which gets sorted in descending order and get saved under the csv file.

Task 2

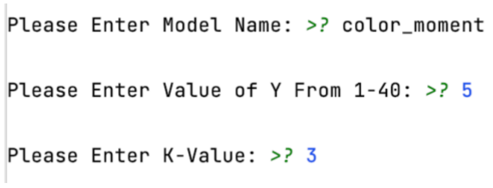
In Task-2 we utilized all the feature models (color moments, histogram of oriented gradients, and extended local binary patterns) and dimensionality reduction algorithms (PCA, SVD, LDA, K-Means). This task made use of database.py to extract all the images of a given subject Y. This dictionary of images is then passed to a function get_object_features_matrix to convert the given dictionary into an object-feature matrix. This is done by first applying a feature extraction algorithm on a while iterating through all the images and then applying linear transformations to the given image. As a result, an image-feature matrix is generated with images as rows and feature values as columns.

After getting the object-feature matrix, this matrix is passed to dimensionality reduction algorithms; the description of each dimensionality reduction algorithm is given in the above headings. These algorithms give us k-latent semantics for the given objects. Now we group all the images of individual types together and calculate the average of each latent semantic of the given group of images. This results in a matrix of 13 Types as rows and k-Latent Semantics as columns. We call this matrix a type-weight pair.
These newly created type-weight pairs are saved into the database and into the output folder named “output/task_2/”.

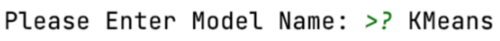
Expected Output of Task


```
Subject-4 Types,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3
smooth,-0.061108518,-0.023024026,0.149261856
rot,-0.075656784,-0.023733763,-0.139373326
original,-0.076511377,-0.027236354,0.164216721
neg,-0.079134107,-0.032621905,-0.114610338
emboss,-0.084594548,-0.032330179,-0.045471042
jitter,-0.093919736,-0.028814945,0.0718355
poster,-0.094860762,-0.035093659,0.011477956
con,-0.098273885,-0.03520236,-0.013102852
cc,-0.110843742,-0.028732041,-0.026134282
```

Sample Input

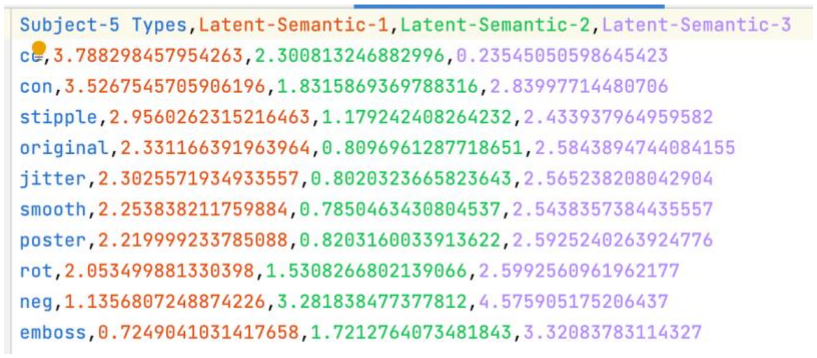


```
Please Enter Model Name: >? color_moment

Please Enter Value of Y From 1-40: >? 5

Please Enter K-Value: >? 3
```



```
Please Enter Model Name: >? KMeans
```

Output



```
Subject-5 Types,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3
cc,3.788298457954263,2.300813246882996,0.23545050598645423
con,3.5267545705906196,1.8315869369788316,2.83997714480706
stipple,2.9560262315216463,1.179242408264232,2.433937964959582
original,2.331166391963964,0.8096961287718651,2.5843894744084155
jitter,2.3025571934933557,0.8020323665823643,2.565238208042904
smooth,2.253838211759884,0.7850463430804537,2.5438357384435557
poster,2.219999233785088,0.8203160033913622,2.5925240263924776
rot,2.053499881330398,1.5308266802139066,2.5992560961962177
neg,1.1356807248874226,3.281838477377812,4.575905175206437
emboss,0.7249041031417658,1.7212764073481843,3.32083783114327
```

Explanation

When we ran the task with color_moment on the subject “5” and dimensionality reduction K-Means. It took all images of subject “5” and applied the color moment feature extractor on them. After this data is passed to the reduction algorithm to find k-latent semantics which get sorted in descending order and get saved under the csv file.

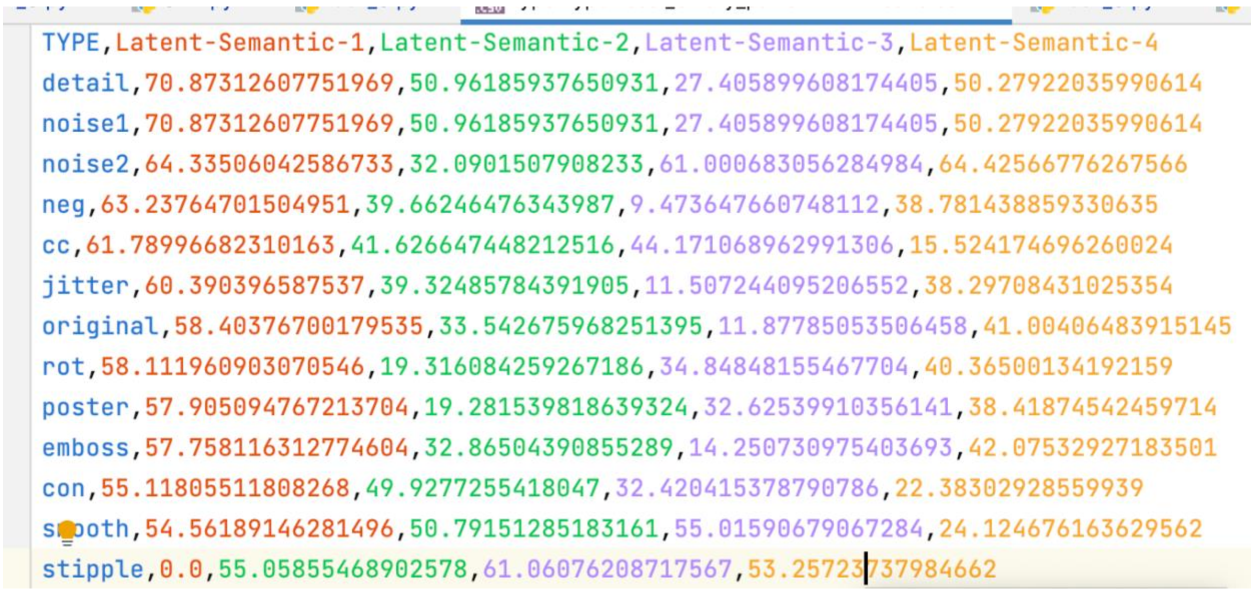
Task 3

In this task, the aim is to create a method of similarity matrix, this method will create a type-type similarity matrix. Since we have 13 types in total, we will have a 13x13 size matrix each row and column representing the relationship among the types, {"cc", "con", "detail", "emboss", "jitter", "neg", "noise1","Noise2 ", "original"," Poster ", "rot", "smooth", "stipple"}.

We created a function get_type_similarities(type1, type2), which takes 2 different types as arguments; get all the images of both the types, compare all images of type1 to all images of type2 using the Euclidean Distance function. After calculating all the distance values, it calculates the mean value of them and returns that value as a similarity measure between the two types. This is done for all the combinations of type pairs. Once this is done, we get the type-type similarity matrix. This similarity matrix gets saved in the database under the table "matrix_similarity".

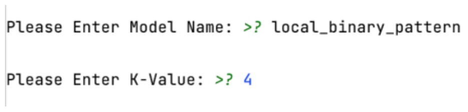
Once done with similarity matrix creation, we apply feature extraction on this similarity matrix and then dimensionality reduction algorithm. This provides us with k-Latent Semantics for each type. We sort this matrix w.r.t highest latent semantic value and save the output in a csv file under the output folder named "output/task_3/".

Expected Output of Task

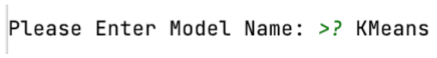


```
TYPE,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4
detail,70.87312607751969,50.96185937650931,27.405899608174405,50.27922035990614
noise1,70.87312607751969,50.96185937650931,27.405899608174405,50.27922035990614
noise2,64.33506042586733,32.0901507908233,61.000683056284984,64.42566776267566
neg,63.23764701504951,39.66246476343987,9.473647660748112,38.781438859330635
cc,61.78996682310163,41.626647448212516,44.171068962991306,15.524174696260024
jitter,60.390396587537,39.32485784391905,11.507244095206552,38.29708431025354
original,58.40376700179535,33.542675968251395,11.87785053506458,41.00406483915145
rot,58.111960903070546,19.316084259267186,34.84848155467704,40.36500134192159
poster,57.905094767213704,19.281539818639324,32.62539910356141,38.41874542459714
emboss,57.758116312774604,32.86504390855289,14.250730975403693,42.07532927183501
con,55.11805511808268,49.9277255418047,32.420415378790786,22.38302928559939
smooth,54.56189146281496,50.79151285183161,55.01590679067284,24.124676163629562
stipple,0.0,55.05855468902578,61.06076208717567,53.25723737984662
```

Sample Input

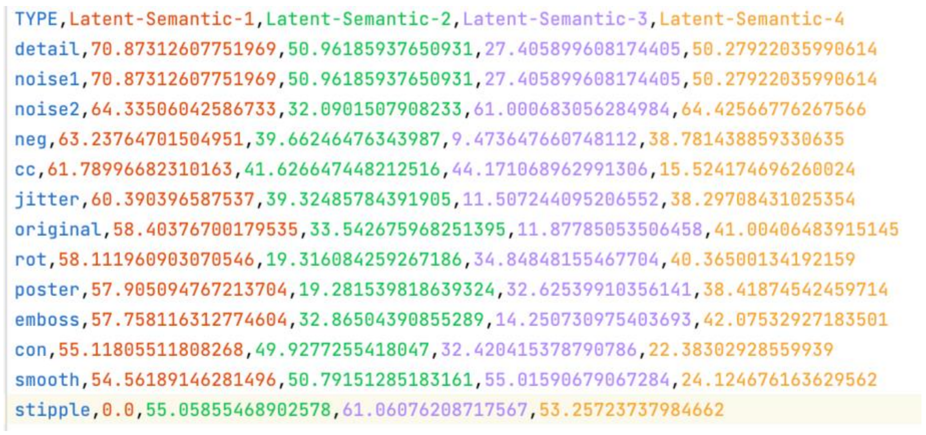


```
Please Enter Model Name: >? local_binary_pattern

Please Enter K-Value: >? 4
```



```
Please Enter Model Name: >? KMeans
```

Output



```
TYPE,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4
detail,70.87312607751969,50.96185937650931,27.405899608174405,50.27922035990614
noise1,70.87312607751969,50.96185937650931,27.405899608174405,50.27922035990614
noise2,64.33506042586733,32.0901507908233,61.000683056284984,64.42566776267566
neg,63.23764701504951,39.66246476343987,9.473647660748112,38.781438859330635
cc,61.78996682310163,41.626647448212516,44.171068962991306,15.524174696260024
jitter,60.390396587537,39.32485784391905,11.507244095206552,38.29708431025354
original,58.40376700179535,33.542675968251395,11.87785053506458,41.00406483915145
rot,58.111960903070546,19.316084259267186,34.84848155467704,40.36500134192159
poster,57.905094767213704,19.281539818639324,32.62539910356141,38.41874542459714
emboss,57.758116312774604,32.86504390855289,14.250730975403693,42.07532927183501
con,55.11805511808268,49.9277255418047,32.420415378790786,22.3830292855939
smooth,54.56189146281496,50.79151285183161,55.01590679067284,24.124676163629562
stipple,0.0,55.05855468902578,61.06076208717567,53.25723737984662
```

Explanation

When we ran the task with color_moment on the k-value "4" and dimensionality reduction K-Means. This provides us with Detail as the highest latent semantic output and most important. This is since the color moments in Detail type image are much higher than compared to other image dataset, making this Type to have the highest value.

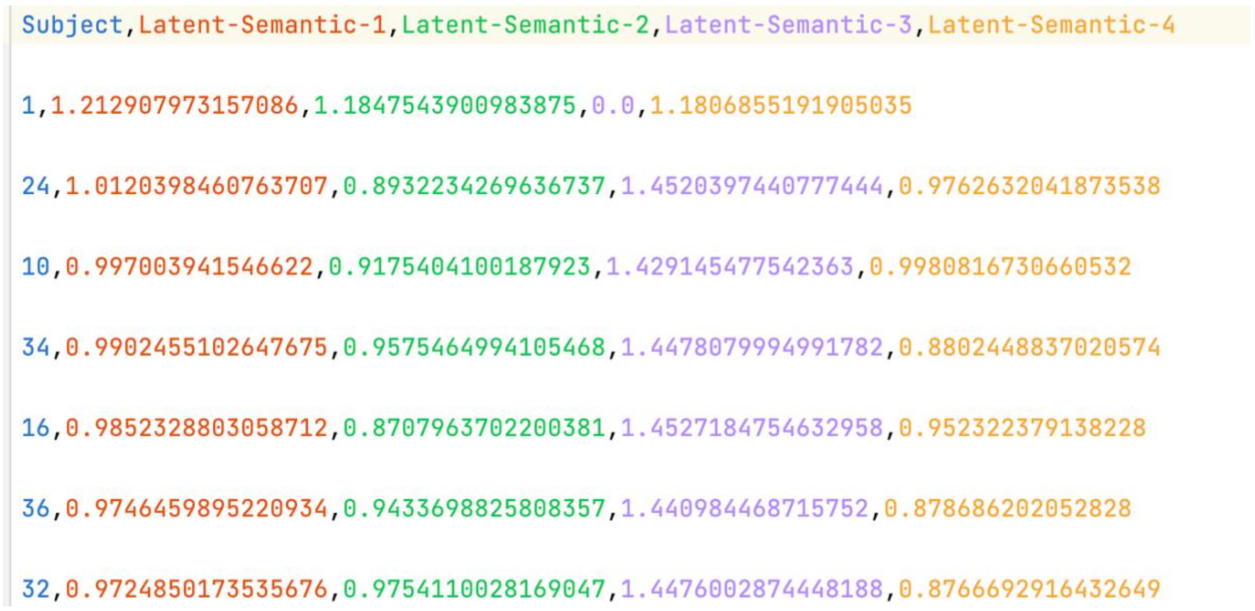
Task 4

In this task, the aim is to create a method of similarity matrix, this method will create a subject-subject similarity matrix. Since we have 40 different subjects in total, we will have a 40x40 size matrix with each row and column representing the relationship among the subjects.

We created a function get_subject_similarities(subject1, subject2), which takes 2 different subjects as arguments; get all the images of both the subjects, compare all images of subjects1 to all images of subjects2 using the Euclidean Distance function. After calculating all the distance values, it calculates the mean value of them and returns that value as a similarity measure between the two subjects. This is done for all the combinations of subject pairs. Once this is done, we get the subject-subject similarity matrix. This similarity matrix gets saved in the database under the table "matrix_similarity".

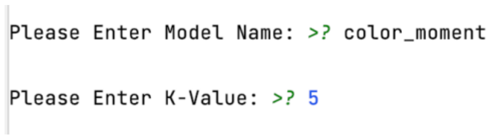
Once done with similarity matrix creation, we apply feature extraction on this similarity matrix and then dimensionality reduction algorithm. This provides us with k-Latent Semantics for each subject. We sort this matrix w.r.t highest latent semantic value and save the output in a csv file under the output folder named "output/task_4/".

Expected Output of Task

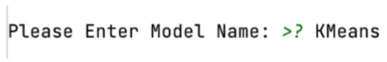


```
Subject,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4

1,1.212907973157086,1.1847543900983875,0.0,1.1806855191905035

24,1.0120398460763707,0.8932234269636737,1.4520397440777444,0.9762632041873538

10,0.997003941546622,0.9175404100187923,1.429145477542363,0.9980816730660532

34,0.9902455102647675,0.9575464994105468,1.4478079994991782,0.8802448837020574

16,0.9852328803058712,0.8707963702200381,1.4527184754632958,0.952322379138228

36,0.9746459895220934,0.9433698825808357,1.440984468715752,0.878686202052828

32,0.9724850173535676,0.9754110028169047,1.4476002874448188,0.8766692916432649
```

Sample Input

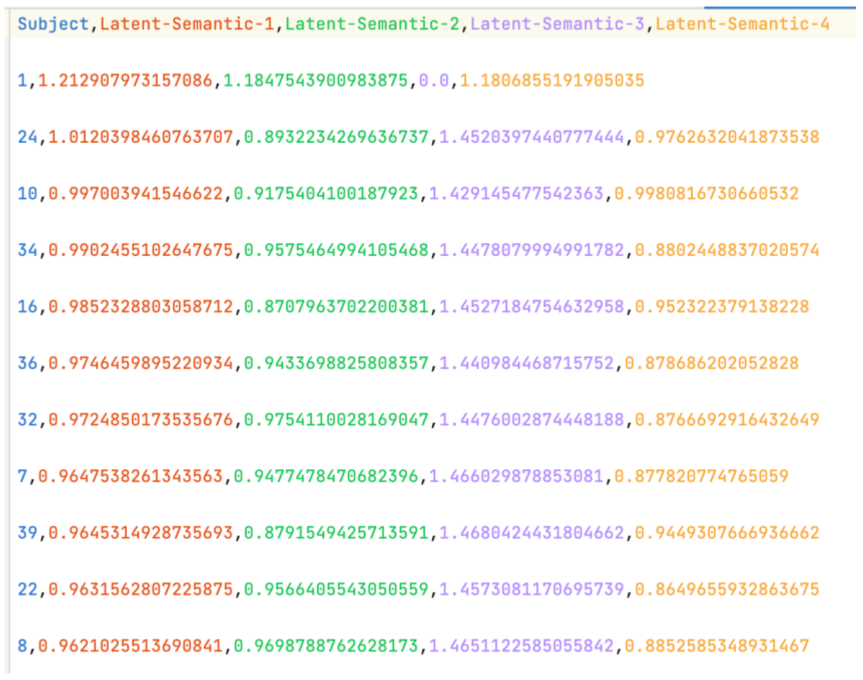


```
Please Enter Model Name: >? color_moment

Please Enter K-Value: >? 5
```



```
Please Enter Model Name: >? KMeans
```

Output



```
Subject,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4

1,1.212907973157086,1.1847543900983875,0.0,1.1806855191905035

24,1.0120398460763707,0.8932234269636737,1.4520397440777444,0.9762632041873538

10,0.997003941546622,0.9175404100187923,1.429145477542363,0.9980816730660532

34,0.9902455102647675,0.9575464994105468,1.4478079994991782,0.8802448837020574

16,0.9852328803058712,0.8707963702200381,1.4527184754632958,0.952322379138228

36,0.9746459895220934,0.9433698825808357,1.440984468715752,0.878686202052828

32,0.9724850173535676,0.9754110028169047,1.4476002874448188,0.8766692916432649

7,0.9647538261343563,0.9477478470682396,1.466029878853081,0.877820774765059

39,0.9645314928735693,0.8791549425713591,1.4680424431804662,0.9449307666936662

22,0.9631562807225875,0.9566405543050559,1.4573081170695739,0.8649655932863675

8,0.9621025513690841,0.9698788762628173,1.4651122585055842,0.8852585348931467
```

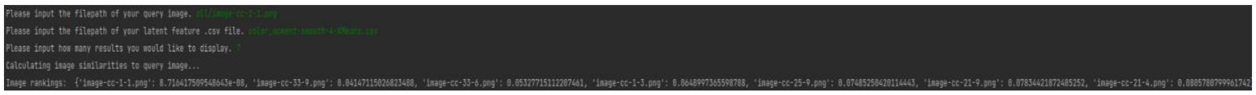
Explanation

When we ran the task with color_moment on the k-value “5” and dimensionality reduction K-Means. This provides us with Subject-1 as the highest latent semantic output, with Subject-24 and Subject-10 being 2nd and 3rd most important. This is since the color moments in Subject-1 are much higher than compared to other image dataset, making this Subject to have the highest value.

Task 5

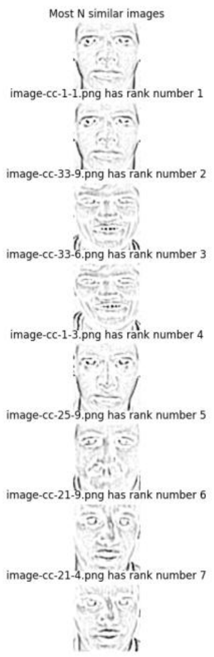
In this task we must compare the user specified query image with the latent semantic input file. The latent semantic file that we are using as input is generated from a previous task, so we know that all the images specified in the file belong to our database. Once each of the object's source image information is stored in a file, we can append the input query image to the end of the file. We then read the input latent semantic file’s name, which contains the parameters required to perform the same latent semantic calculation that generated the input latent semantic file. We then perform this latent semantic calculation on the new source object file that contains our query image. Once this is done, we now have the latent semantic space of the input image but updated with the new query image. We can then perform Euclidean distance to calculate the nearest N other images and visualize them. We use Euclidean distance so that we can compare the quality of results between task 5, 6, and 7 to determine if comparing by object, label, or subject ID is more accurate rather than focusing on the distance function.

Sample Input



```
Please input the filepath of your query image. dl1/image-cc-1-1.png
Please input the filepath of your latent feature .csv file. color_moment-smooth-4-KMeans.csv
Please input how many results you would like to display. 7
Calculating image similarities to query image...
Image rankings:  {'image-cc-1-1.png': 8.716417509548643e-08, 'image-cc-33-9.png': 0.04147115026823488, 'image-cc-33-6.png': 0.05327715112207461, 'image-cc-1-3.png': 0.0648997365598788, 'image-cc-25-9.png': 0.07485250420114443, 'image-cc-21-9.png': 0.07834421872485252, 'image-cc-21-4.png': 0.08057807999617421
```

Output


Most N similar images
image-cc-1-1.png has rank number 1
image-cc-33-9.png has rank number 2
image-cc-33-6.png has rank number 3
image-cc-1-3.png has rank number 4
image-cc-25-9.png has rank number 5
image-cc-21-9.png has rank number 6
image-cc-21-4.png has rank number 7

Explanation

The top n objects that have the nearest Euclidean Distance to the query image in the latent space have their filename and distance output to the terminal. These nodes are the n most similar nodes to the query image according to the latent semantic space. These images are then visualized with the query image at the top and the rest of the images in ascending order according to rank, where rank 1 is the most like the query image.

Task 6

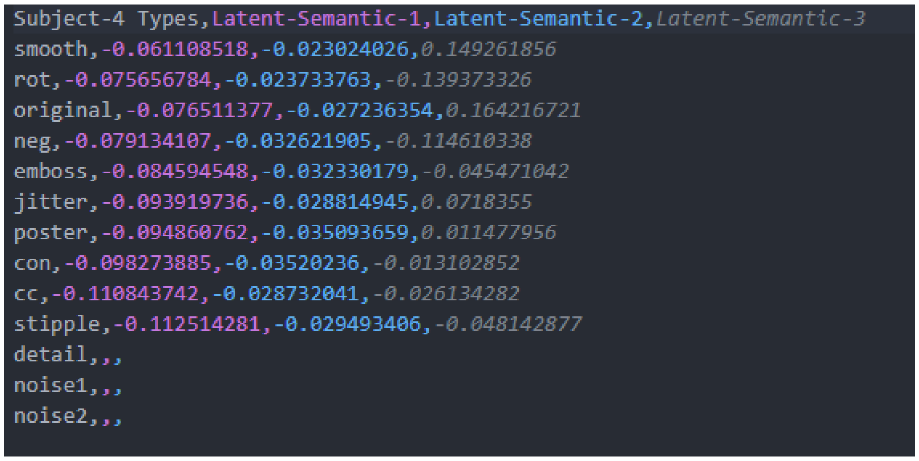
Task 6 involved taking a query image and a latent semantics file and using them to associate a type of label to the given image according to the selected latent semantics. The program utilizes the latent semantics filename to determine the appropriate feature model, k, and dimensionality reduction model to utilize. The task then gets the latent semantics in the same object space as the given query image by recalculating the latent features while including the query image in the dataset. The similarity between generated latent semantics of the query image and every other image is taken as Euclidean distance and then passed into a similarity matrix.

This similarity matrix is then passed into a function that averages these similarities by image type and then weighs those averages by the type-weight pairs provided in the original input latent semantics file to gain the weighted likeliness value for each image type. The latent semantic consideration weight is applied in such a way that the first latent semantic is given double the importance of the latent semantic following it. The exact method of determining this weighted score was multiplying the type of weight from the given latent semantics file with the current latent semantic weight (which starts at 2 and is divided in half with each concurrent latent semantic) and then with the average type of similarity for the current type. Each of these weighed latent semantic values are summed for each type to determine the final score to determine the most likely type with. The type associated with the highest value is returned as the most likely type.

Sample Input

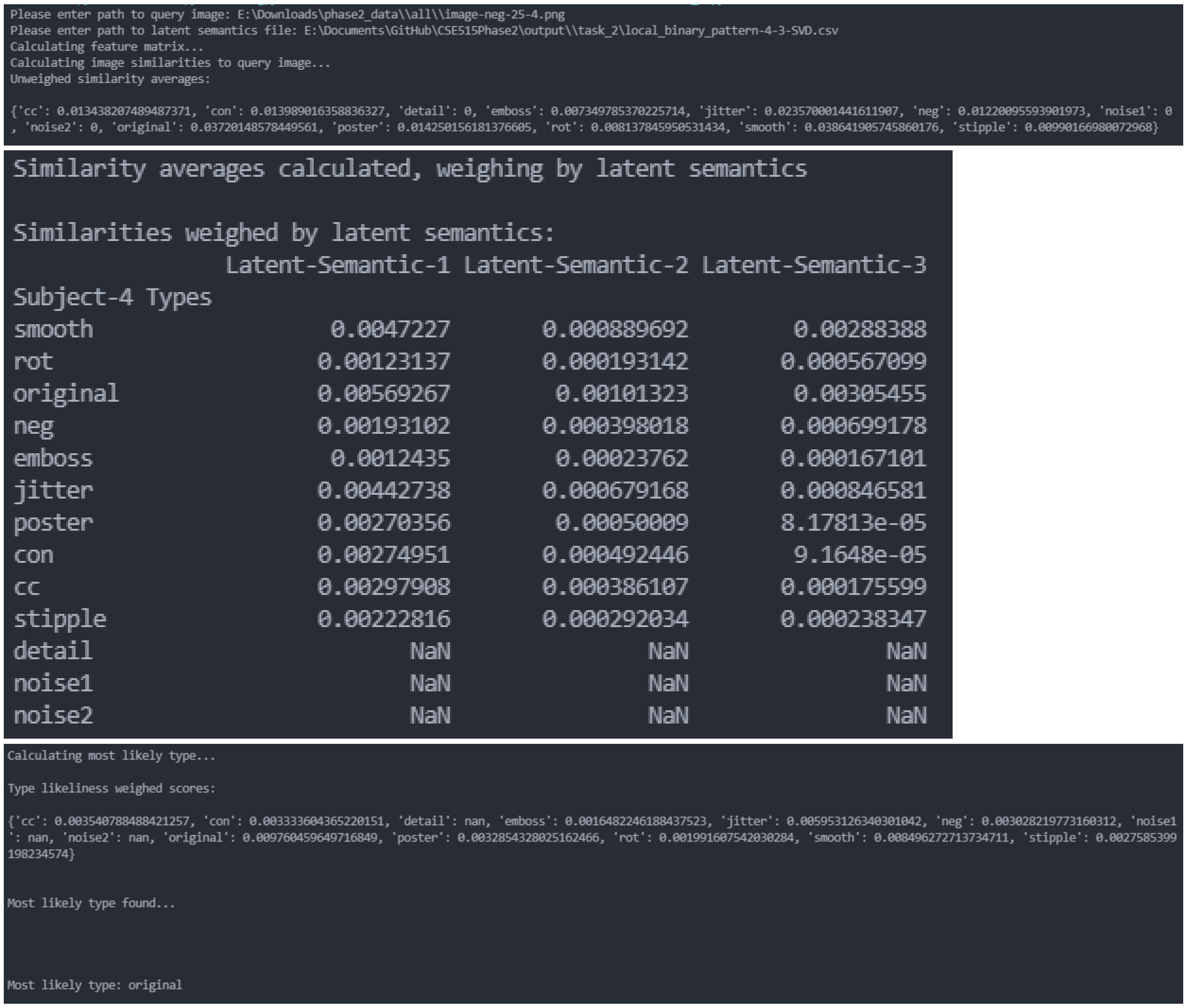
image-neg-25-4.png

local_binary_pattern-4-3-SVD.csv



```
Subject-4 Types,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3
smooth,-0.061108518,-0.023024026,0.149261856
rot,-0.075656784,-0.023733763,-0.139373326
original,-0.076511377,-0.027236354,0.164216721
neg,-0.079134107,-0.032621905,-0.114610338
emboss,-0.084594548,-0.032330179,-0.045471042
jitter,-0.093919736,-0.028814945,0.0718355
poster,-0.094860762,-0.035093659,0.011477956
con,-0.098273885,-0.03520236,-0.013102852
cc,-0.110843742,-0.028732041,-0.026134282
stipple,-0.112514281,-0.029493406,-0.048142877
detail,,,
noise1,,,
noise2,,,
```

Output



```
Please enter path to query image: E:\Downloads\phase2_data\\all\\image-neg-25-4.png
Please enter path to latent semantics file: E:\Documents\GitHub\CSE515Phase2\output\\task_2\local_binary_pattern-4-3-SVD.csv
Calculating feature matrix...
Calculating image similarities to query image...
Unweighed similarity averages:

{'cc': 0.013438207489487371, 'con': 0.013989016358836327, 'detail': 0, 'emboss': 0.007349785370225714, 'jitter': 0.023570001441611907, 'neg': 0.01220095593901973, 'noise1': 0
, 'noise2': 0, 'original': 0.03720148578449561, 'poster': 0.014250156181376605, 'rot': 0.008137845950531434, 'smooth': 0.038641905745860176, 'stipple': 0.00990166980072968}
```

```
Similarity averages calculated, weighing by latent semantics

Similarities weighed by latent semantics:
                Latent-Semantic-1 Latent-Semantic-2 Latent-Semantic-3
Subject-4 Types
smooth                  0.0047227       0.000889692        0.00288388
rot                    0.00123137       0.000193142       0.000567099
original               0.00569267        0.00101323        0.00305455
neg                    0.00193102       0.000398018       0.000699178
emboss                  0.0012435        0.00023762       0.000167101
jitter                 0.00442738       0.000679168       0.000846581
poster                 0.00270356        0.00050009       8.17813e-05
con                    0.00274951       0.000492446        9.1648e-05
cc                     0.00297908       0.000386107       0.000175599
stipple                0.00222816       0.000292034       0.000238347
detail                        NaN               NaN               NaN
noise1                        NaN               NaN               NaN
noise2                        NaN               NaN               NaN
```

```
Calculating most likely type...

Type likeliness weighed scores:

{'cc': 0.003540788488421257, 'con': 0.003333604365220151, 'detail': nan, 'emboss': 0.0016482246188437523, 'jitter': 0.005953126340301042, 'neg': 0.003028219773160312, 'noise1
': nan, 'noise2': nan, 'original': 0.009760459649716849, 'poster': 0.0032854328025162466, 'rot': 0.001991607542030284, 'smooth': 0.008496272713734711, 'stipple': 0.0027585399
198234574}


Most likely type found...



Most likely type: original
```

"Most likely type: original"

Explanation

The usage of the ELBP feature led to similarity scores being largely dependent on the texture of the images instead of the color, so filters that only affected colors and not texture, such as 'neg' which is that of the input image, will still gain high similarity scores with images from different filters that also do not affect texture, like the selected most likely type, 'original'. The latent semantic input weights also yielded a higher overall skew towards the original type as well, influencing the output.

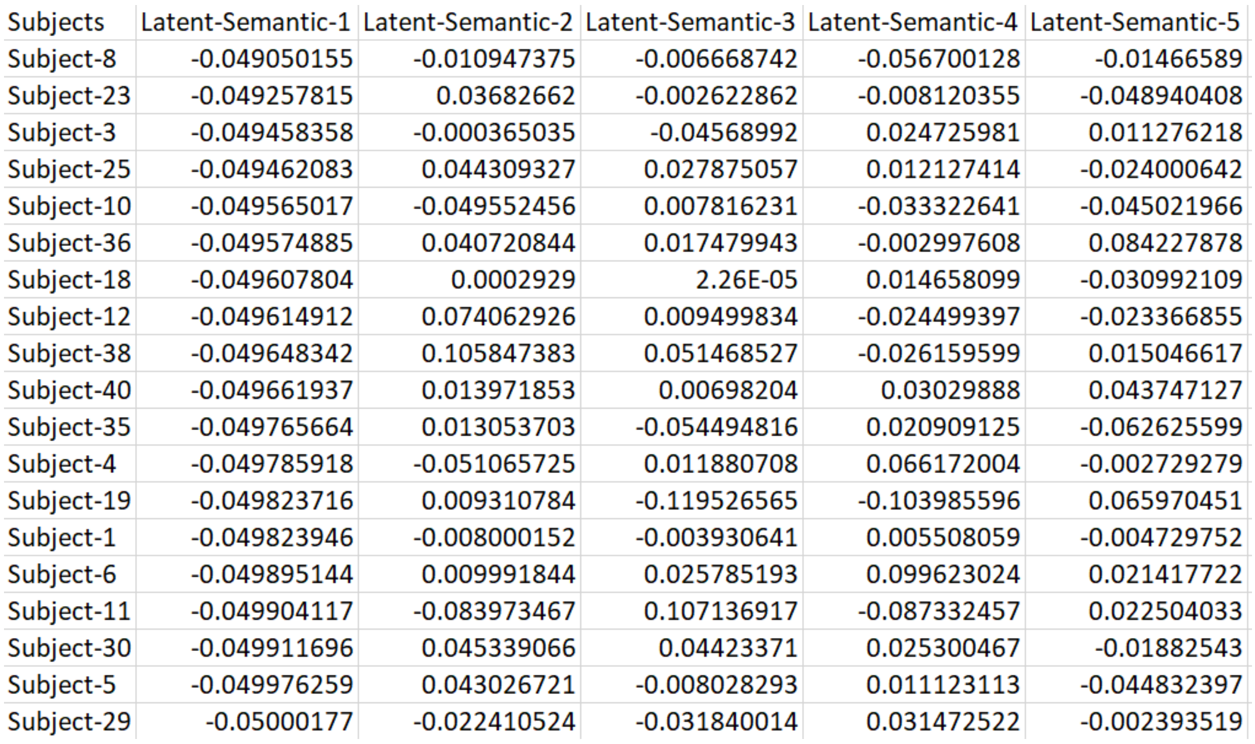
Task 7

This task, much like Tasks 5 and 6, takes as input a query image and a latent semantics file and attempts to associate a Subject ID to the given image. It is assumed that the latent semantics file provided is generated via Task 1, or at least conforms to the same formatting specifications. We first parse the latent semantics file for the desired information, stored in the filename. Following this, the database is queried for the image set and the query image is normalized and appended to this dataset. From here, the feature model and dimension reduction algorithms are applied (retrieved from the latent semantics filename). This provides us with a latent semantic space which contains both our dataset and the query image. Euclidean distance is then applied, generating a similarity matrix between the query image and all other images present.

Finally, we calculate the average similarity for each Subject ID as they appear in the similarity matrix. Following this, a weighted average between all latent semantics is determined for each Subject ID in the given latent semantics file. The weight is determined using a weighted variable and the average for the current Subject ID in question. The weighted calculations are then summed for each Subject ID, and the maximum of these summations is then chosen as the most likely Subject ID for the query image. Thus, the corresponding Subject ID for this maximum is returned as output.

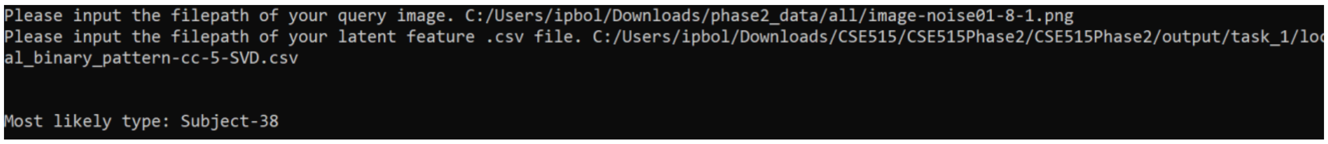
Sample Input

image-noise01-8-1.png



Subjects	Latent-Semantic-1	Latent-Semantic-2	Latent-Semantic-3	Latent-Semantic-4	Latent-Semantic-5
Subject-8	-0.049050155	-0.010947375	-0.006668742	-0.056700128	-0.01466589
Subject-23	-0.049257815	0.03682662	-0.002622862	-0.008120355	-0.048940408
Subject-3	-0.049458358	-0.000365035	-0.04568992	0.024725981	0.011276218
Subject-25	-0.049462083	0.044309327	0.027875057	0.012127414	-0.024000642
Subject-10	-0.049565017	-0.049552456	0.007816231	-0.033322641	-0.045021966
Subject-36	-0.049574885	0.040720844	0.017479943	-0.002997608	0.084227878
Subject-18	-0.049607804	0.0002929	2.26E-05	0.014658099	-0.030992109
Subject-12	-0.049614912	0.074062926	0.009499834	-0.024499397	-0.023366855
Subject-38	-0.049648342	0.105847383	0.051468527	-0.026159599	0.015046617
Subject-40	-0.049661937	0.013971853	0.00698204	0.03029888	0.043747127
Subject-35	-0.049765664	0.013053703	-0.054494816	0.020909125	-0.062625599
Subject-4	-0.049785918	-0.051065725	0.011880708	0.066172004	-0.002729279
Subject-19	-0.049823716	0.009310784	-0.119526565	-0.103985596	0.065970451
Subject-1	-0.049823946	-0.008000152	-0.003930641	0.005508059	-0.004729752
Subject-6	-0.049895144	0.009991844	0.025785193	0.099623024	0.021417722
Subject-11	-0.049904117	-0.083973467	0.107136917	-0.087332457	0.022504033
Subject-30	-0.049911696	0.045339066	0.04423371	0.025300467	-0.01882543
Subject-5	-0.049976259	0.043026721	-0.008028293	0.011123113	-0.044832397
Subject-29	-0.05000177	-0.022410524	-0.031840014	0.031472522	-0.002393519

Output



```
Please input the filepath of your query image. C:/Users/ipbol/Downloads/phase2_data/all/image-noise01-8-1.png
Please input the filepath of your latent feature .csv file. C:/Users/ipbol/Downloads/CSE515/CSE515Phase2/CSE515Phase2/output/task_1/loc
al_binary_pattern-cc-5-SVD.csv


Most likely type: Subject-38
```

Explanation

Due to the nature of how the Extended Local Binary Patterns function, texture was taken as a principal factor which the similarity function depended upon. Furthermore, since the "noise01" image subset consists of particularly grainy images, those with a similarly grainy texture on them are evaluated and determined to be highly similar as a result. This obfuscates other components that may hold more sway in the result. Furthermore, the weights calculated for each subject when determining similarity consider all images of a given subject. Thus, these subjects produced a higher similarity when the filter applied to the image affected texture as opposed to instances where texture was not affected.

Task 8

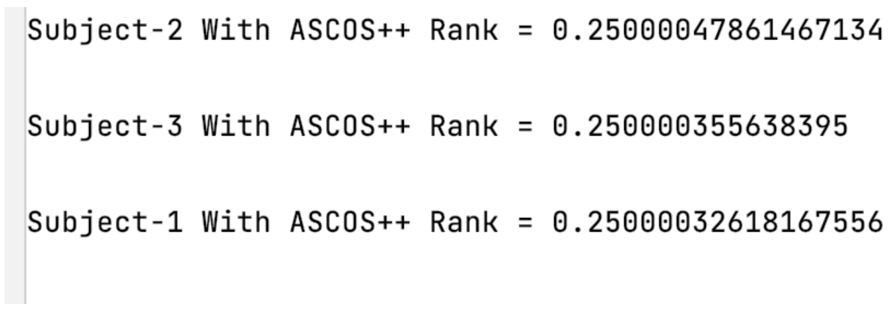
In this task we take a similarity graph and integer n as the input and output object's similarity score to other objects using ASCOS++, using these scores we then output the most n similar objects in the database. The similarity graph will be constructed from a list of images and put into the correct format for task 8 to execute. We can use this graph to get the similarity between all the objects according to ASCOS++.[5] The main equation that is used for this algorithm is:[5]

$$s_{ij} := \begin{cases} c \cdot \sum_{\forall k \in N(i)} \frac{w_{ik}}{w_{i*}} (1 - \exp(-w_{ik})) s_{kj}, & \text{if } i \neq j \\ 1 & \text{if } i = j, \end{cases}$$

When we are iterating over the S matrix, we update each cell using this equation, once the matrix converges, we stop and that will be the similarity matrix for the objects. We can then use this S matrix to find the most similar n images and output them for the user.

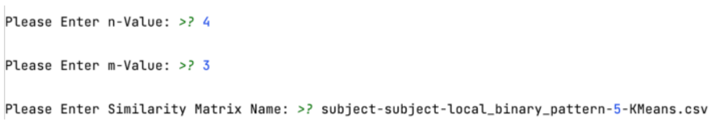
For calculating both the matrix convergence, we calculate the sum of the difference of weights of both the matrices. Once we get the sum, we calculate its average to get the value of the code difference among the two matrices. This is computed for all the nodes of both matrices. We used a converge_parameter that is valued nearly to zero so we can have a convergence that is very much near to each other. This converge_parameter is used to check if the average of the sum of difference of node weights is less than or equal to this value. If yes, then we add one to our counter. After getting the counter value we divide it by the total number of nodes and calculate the percentage change between the previous iteration matrix to current iteration matrix and the. If the percentage is above 99%, we consider both the matrices as converged.

Once matrices are converged, we need to calculate the m-most significant subjects. For this we use a function get_most_significant_subjects that takes the graph represented as a matrix as input and marks all the node ranks using the update_pagerank function. Once all the node ranks are calculated we sort the result and show m-most significant subjects.



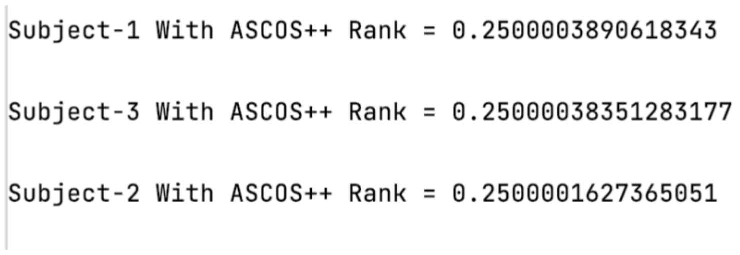
```
Subject-2 With ASCOS++ Rank = 0.25000047861467134

Subject-3 With ASCOS++ Rank = 0.250000355638395

Subject-1 With ASCOS++ Rank = 0.25000032618167556
```

Sample Input



```
Please Enter n-Value: >? 4

Please Enter m-Value: >? 3

Please Enter Similarity Matrix Name: >? subject-subject-local_binary_pattern-5-KMeans.csv
```

Output



```
Subject-1 With ASCOS++ Rank = 0.2500003890618343

Subject-3 With ASCOS++ Rank = 0.25000038351283177

Subject-2 With ASCOS++ Rank = 0.2500001627365051
```

Explanation

When we ran the ASCOS++ on the given latent semantic file we got Subject-1, Subject-3, and Subject-2 as the m-most significant node. The reason for this is when we randomly initialized the graph and loop on it until it converges, we got these 3 nodes as having the highest node values. Being considering the 4 nearest nodes for each of the node in each similarity graph, these subjects got the highest weights.

Task 9

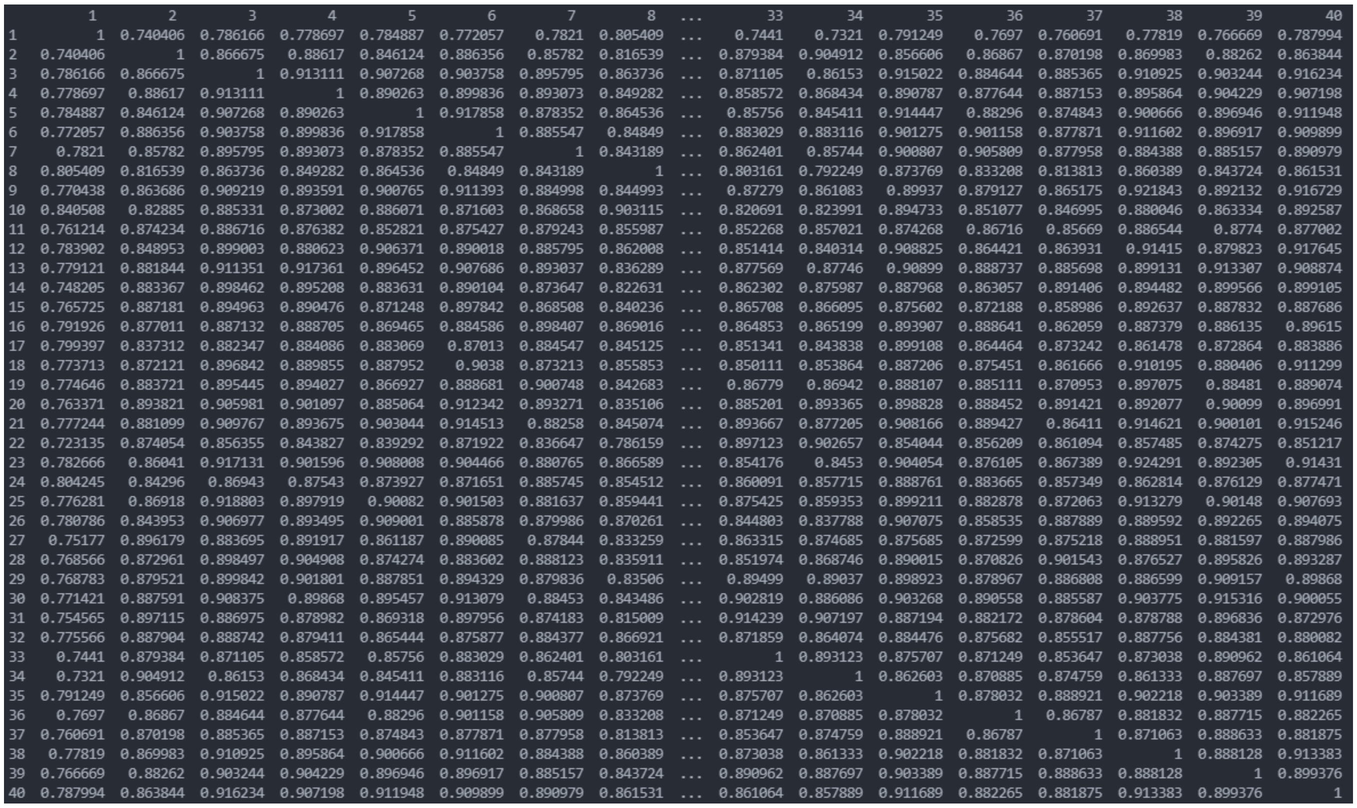
This task involved taking a subject-subject similarity matrix, an integer n, an integer m, and three subject IDs and then using the similarity matrix to create a similarity graph with each subject node having an edge to the n most similar subject nodes. Next, the Personalized PageRank (PPR) measure was applied to the graph to determine the m-most significant subjects in relation to the seed set of the three provided subject IDs. The PPR measure was calculated using the following equation:[4]

$$p = (I - (\beta T_G)) \quad ^{-1}(1 - \beta)c$$

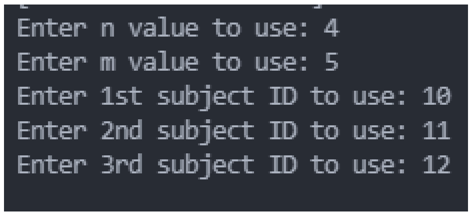
Where p represents the vector of Personalized PageRank scores for each subject, I represents the identity matrix, β represents the 'damping factor' or the probability that the random surfer will go to a neighboring node instead of jumping randomly[4], T_G represents the transition matrix, and c represents the vector of probabilities that the random surfer will jump to each node when the random jump is performed. [4] This c vector represents the part of the PPR algorithm where the 'personalized' aspect comes in. The indexes for the three input subject nodes are given a significantly higher probability of being jumped to here in comparison to all the other nodes. After this p vector is calculated, the scores are sorted in descending order and the subjects associated with the top m scores are returned as output.

Sample Input

subject-subject-histogram_of_oriented_gradients-3-LDA.csv

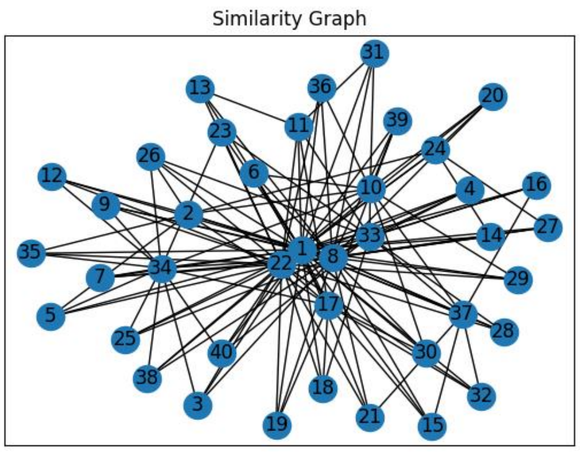


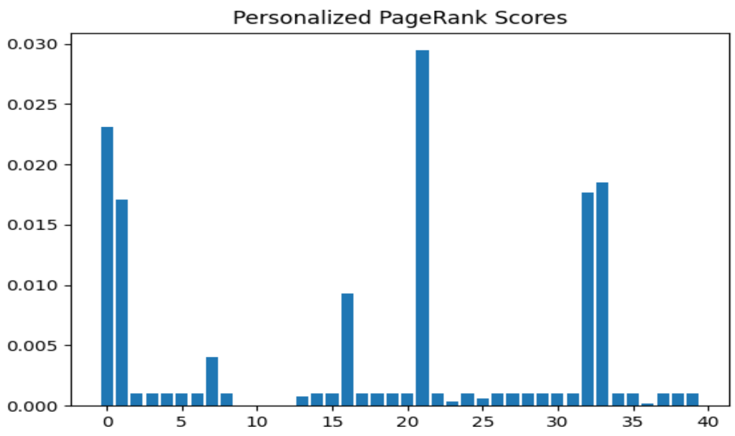
```
          1        2        3        4        5        6        7        8  ...       33       34       35       36       37       38       39       40
1         1 0.740406 0.786166 0.778697 0.784887 0.772057   0.7821 0.805409  ...   0.7441   0.7321 0.791249   0.7697 0.760691  0.77819 0.766669 0.787994
2  0.740406        1 0.866675  0.88617 0.846124 0.886356  0.85782 0.816539  ... 0.879384 0.904912 0.856606  0.86867 0.870198 0.869983  0.88262 0.863844
3  0.786166 0.866675        1 0.913111 0.907268 0.903758 0.895795 0.863736  ... 0.871105  0.86153 0.915022 0.884644 0.885365 0.910925 0.903244 0.916234
4  0.778697  0.88617 0.913111        1 0.890263 0.899836 0.893073 0.849282  ... 0.858572 0.868434 0.890787 0.877644 0.887153 0.895864 0.904229 0.907198
5  0.784887 0.846124 0.907268 0.890263        1 0.917858 0.878352 0.864536  ...  0.85756 0.845411 0.914447  0.88296 0.874843 0.900666 0.896946 0.911948
6  0.772057 0.886356 0.903758 0.899836 0.917858        1 0.885547  0.84849  ... 0.883029 0.883116 0.901275 0.901158 0.877871 0.911602 0.896917 0.909899
7    0.7821  0.85782 0.895795 0.893073 0.878352 0.885547        1 0.843189  ... 0.862401  0.85744 0.900807 0.905809 0.877958 0.884388 0.885157 0.890979
8  0.805409 0.816539 0.863736 0.849282 0.864536  0.84849 0.843189        1  ... 0.803161 0.792249 0.873769 0.833208 0.813813 0.860389 0.843724 0.861531
9  0.770438 0.863686 0.909219 0.893591 0.900765 0.911393 0.884998 0.844993  ...  0.87279 0.861083  0.89937 0.879127 0.865175 0.921843 0.892132 0.916729
10 0.840508  0.82885 0.885331 0.873002 0.886071 0.871603 0.868658 0.903115  ... 0.820691 0.823991 0.894733 0.851077 0.846995 0.880046 0.863334 0.892587
11 0.761214 0.874234 0.886716 0.876382 0.852821 0.875427 0.879243 0.855987  ... 0.852268 0.857021 0.874268  0.86716  0.85669 0.886544   0.8774 0.877002
12 0.783902 0.848953 0.899003 0.880623 0.906371 0.890018 0.885795 0.862008  ... 0.851414 0.840314 0.908825 0.864421 0.863931  0.91415 0.879823 0.917645
13 0.779121 0.881844 0.911351 0.917361 0.896452 0.907686 0.893037 0.836289  ... 0.877569  0.87746  0.90899 0.888737 0.885698 0.899131 0.913307 0.908874
14 0.748205 0.883367 0.898462 0.895208 0.883631 0.890104 0.873647 0.822631  ... 0.862302 0.875987 0.887968 0.863057 0.891406 0.894482 0.899566 0.899105
15 0.765725 0.887181 0.894963 0.890476 0.871248 0.897842 0.868508 0.840236  ... 0.865708 0.866095 0.875602 0.872188 0.858986 0.892637 0.887832 0.887686
16 0.791926 0.877011 0.887132 0.888705 0.869465 0.884586 0.898407 0.869016  ... 0.864853 0.865199 0.893907 0.888641 0.862059 0.887379 0.886135  0.89615
17 0.799397 0.837312 0.882347 0.884086 0.883069  0.87013 0.884547 0.845125  ... 0.851341 0.843838 0.899108 0.864464 0.873242 0.861478 0.872864 0.883886
18 0.773713 0.872121 0.896842 0.889855 0.887952   0.9038 0.873213 0.855853  ... 0.850111 0.853864 0.887206 0.875451 0.861666 0.910195 0.880406 0.911299
19 0.774646 0.883721 0.895445 0.894027 0.866927 0.888681 0.900748 0.842683  ...  0.86779  0.86942 0.888107 0.885111 0.870953 0.897075  0.88481 0.889074
20 0.763371 0.893821 0.905981 0.901097 0.885064 0.912342 0.893271 0.835106  ... 0.885201 0.893365 0.898828 0.888452 0.891421 0.892077  0.90099 0.896991
21 0.777244 0.881099 0.909767 0.893675 0.903044 0.914513  0.88258 0.845074  ... 0.893667 0.877205 0.908166 0.889427  0.86411 0.914621 0.900101 0.915246
22 0.723135 0.874054 0.856355 0.843827 0.839292 0.871922 0.836647 0.786159  ... 0.897123 0.902657 0.854044 0.856209 0.861094 0.857485 0.874275 0.851217
23 0.782666  0.86041 0.917131 0.901596 0.908008 0.904466 0.880765 0.866589  ... 0.854176   0.8453 0.904054 0.876105 0.867389 0.924291 0.892305  0.91431
24 0.804245  0.84296  0.86943  0.87543 0.873927 0.871651 0.885745 0.854512  ... 0.860091 0.857715 0.888761 0.883665 0.857349 0.862814 0.876129 0.877471
25 0.776281  0.86918 0.918803 0.897919  0.90082 0.901503 0.881637 0.859441  ... 0.875425 0.859353 0.899211 0.882878 0.872063 0.913279  0.90148 0.907693
26 0.780786 0.843953 0.906977 0.893495 0.909001 0.885878 0.879986 0.870261  ... 0.844803 0.837788 0.907075 0.858535 0.887889 0.889592 0.892265 0.894075
27  0.75177 0.896179 0.883695 0.891917 0.861187 0.890085  0.87844 0.833259  ... 0.863315 0.874685 0.875685 0.872599 0.875218 0.888951 0.881597 0.887986
28 0.768566 0.872961 0.898497 0.904908 0.874274 0.883602 0.888123 0.835911  ... 0.851974 0.868746 0.890015 0.870826 0.901543 0.876527 0.895826 0.893287
29 0.768783 0.879521 0.899842 0.901801 0.887851 0.894329 0.879836  0.83506  ...  0.89499  0.89037 0.898923 0.878967 0.886808 0.886599 0.909157  0.89868
30 0.771421 0.887591 0.908375  0.89868 0.895457 0.913079  0.88453 0.843486  ... 0.902819 0.886086 0.903268 0.890558 0.885587 0.903775 0.915316 0.900055
31 0.754565 0.897115 0.886975 0.878982 0.869318 0.897956 0.874183 0.815009  ... 0.914239 0.907197 0.887194 0.882172 0.878604 0.878788 0.896836 0.872976
32 0.775566 0.887904 0.888742 0.879411 0.865444 0.875877 0.884377 0.866921  ... 0.871859 0.864074 0.884476 0.875682 0.855517 0.887756 0.884381 0.880082
33   0.7441 0.879384 0.871105 0.858572  0.85756 0.883029 0.862401 0.803161  ...        1 0.893123 0.875707 0.871249 0.853647 0.873038 0.890962 0.861064
34   0.7321 0.904912  0.86153 0.868434 0.845411 0.883116  0.85744 0.792249  ... 0.893123        1 0.862603 0.870885 0.874759 0.861333 0.887697 0.857889
35 0.791249 0.856606 0.915022 0.890787 0.914447 0.901275 0.900807 0.873769  ... 0.875707 0.862603        1 0.878032 0.888921 0.902218 0.903389 0.911689
36   0.7697  0.86867 0.884644 0.877644  0.88296 0.901158 0.905809 0.833208  ... 0.871249 0.870885 0.878032        1  0.86787 0.881832 0.887715 0.882265
37 0.760691 0.870198 0.885365 0.887153 0.874843 0.877871 0.877958 0.813813  ... 0.853647 0.874759 0.888921  0.86787        1 0.871063 0.888633 0.881875
38  0.77819 0.869983 0.910925 0.895864 0.900666 0.911602 0.884388 0.860389  ... 0.873038 0.861333 0.902218 0.881832 0.871063        1 0.888128 0.913383
39 0.766669  0.88262 0.903244 0.904229 0.896946 0.896917 0.885157 0.843724  ... 0.890962 0.887697 0.903389 0.887715 0.888633 0.888128        1 0.899376
40 0.787994 0.863844 0.916234 0.907198 0.911948 0.909899 0.890979 0.861531  ... 0.861064 0.857889 0.911689 0.882265 0.881875 0.913383 0.899376        1
```

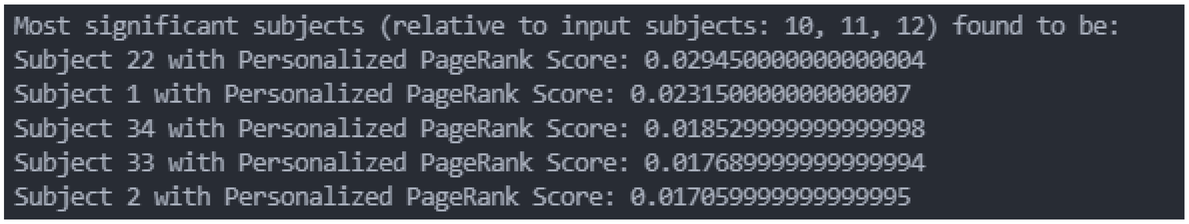


```
Enter n value to use: 4
Enter m value to use: 5
Enter 1st subject ID to use: 10
Enter 2nd subject ID to use: 11
Enter 3rd subject ID to use: 12
```

Output







```
Most significant subjects (relative to input subjects: 10, 11, 12) found to be:
Subject 22 with Personalized PageRank Score: 0.029450000000000004
Subject 1 with Personalized PageRank Score: 0.023150000000000007
Subject 34 with Personalized PageRank Score: 0.018529999999999998
Subject 33 with Personalized PageRank Score: 0.017689999999999994
Subject 2 with Personalized PageRank Score: 0.017059999999999995
```

Explanation

The nodes selected as the most significant tend to be present in the edge lists of some or all the seed nodes, meaning that they are given high importance due to that proximity. Additionally, to that, these nodes also tend to be topologically important in the larger context of the entire similarity graph as well, as can be seen from their centrality in the graph visualization above.

Interface Specifications

Task 1

This task takes in as input a feature extractor name, a name of Type, a k-value in integer, and dimensionality reduction algorithm name. The k-value dictates how many latent-semantics are required for all subjects of a given type. The output of the file gets saved in the “output/task_1” directory.

Task 2

This task takes in as input a feature extractor name, a Subject number in integer, a k-value in integer, and dimensionality reduction algorithm name. The k-value dictates how many latent-semantics are required for all types of a given subject. The output of the file gets saved in the “output/task_2” directory. The value of the subject should be under the range of 1 to 40.

Task 3

This task takes in as input a feature extractor name, a k-value in integer, and dimensionality reduction algorithm name. The k-value dictates how many latent-semantics are required for all types. The output of the file gets saved in the “output/task_3” directory. The type-type similarity matrix created in this task gets saved in the database under table “matrix_similarity”.

Task 4

This task takes in as input a feature extractor name, a k-value in integer, and dimensionality reduction algorithm name. The k-value dictates how many latent-semantics are required for all subjects. The output of the file gets saved in the “output/task_4” directory. The subject-subject similarity matrix created in this task gets saved in the database under table “matrix_similarity”.

Task 5

This task takes in as input a query image file path, a feature model, a latent semantic file, and an integer n. The query image should be of the format .png and the latent semantic file follows a specific naming convention. The latent semantic file should be of the format .csv and specify the feature model, reduction method, and integer k to be used by the reduction method. The integer n will dictate how many output images are returned. In the terminal the nearest n images to the query image will have their filename and ranking displayed. The nearest n images will also be visualized on screen.

Task 6

For this task, we take an input query image file path and a latent semantics file path. For the purposes of this task, it is assumed that the latent semantics file provided is either generated by Task 2 or uses the same formatting conventions as Task 2 with a series of type-weight pairs in a .csv file with a filename of the format '$feature_model-$subject_id-$k-$dimensional_reduction_model.csv' wherein the $ variables are replaced by the appropriate strings. The query image should be in .png format. These inputs are used to regenerate the latent semantics with the new input image in the object space, take the distance of the query image's newly generated latent semantics from those of the rest of the images in the dataset, average those distances by image type, and then sum those averages with decreasing weight applied to each latent semantic column. Those summed averages are then weighed by the input type-weight latent semantics file to bring the output back into the latent semantics space of the input file. These weighted sums are then used to find the highest value to find the most likely image type.

Task 7

This task takes as input a query image file path and latent semantics file path. The query image is expected to be a PNG file with dimensions 64x64 and in grayscale. Furthermore, the latent semantics file is a CSV file expected to be generated in Task 1, or in conformance with the specifications. The file name must contain the feature model, image type identification, image subject identification, and dimensionality reduction technique in that order, separated by hyphens. The output is a String message printed to the terminal containing the Subject ID associated with the query image.

Task 8

This task takes in a similarity graph and an integer n. The similarity graph contains each object in the dataset and has m child nodes which are the m most similar objects to it. This is formatted as an adjacency matrix where the edge values are the similarity score between the objects. We then run ASCOS++ which will output an S matrix. We can then use this matrix to find the smallest n number of values inside of the matrix. These are the n most similar objects that are found inside of the dataset.

Task 9

Task 9 involved taking a subject-subject similarity matrix, a value n, a value m, and three subject IDs and then using them to create a similarity graph from the similarity matrix with each node representing a subject and having an edge to the top 4 most similar subjects. Additionally, the task needed to calculate the Personalized PageRank scores of this graph using the 3 subject IDs as the seed set and then use these scores to return the m most significant subjects.

System Requirements / Installation and Execution Instructions

Prerequisites

- Python 3.6
- MongoDB Server 5.0
- The following Python libraries:
    - os
    - pandas
    - pathlib
    - glob
    - numpy
    - csv
    - PIL
    - matplotlib
    - sklearn
    - math
    - pickle
    - json
    - io
    - pymongo
    - bson
    - math
    - statistics
    - scipy
    - random
    - copy
    - networkx

Install Necessary Python Libraries

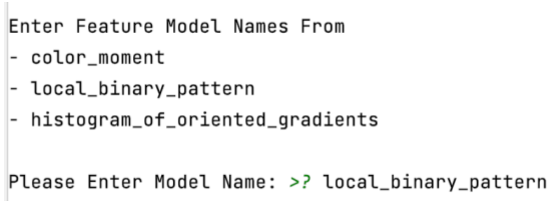
The libraries listed above, if not already installed, can be installed by opening a terminal and entering the command ‘pip install $library_name’ where $library_name represents the name of the library to install.

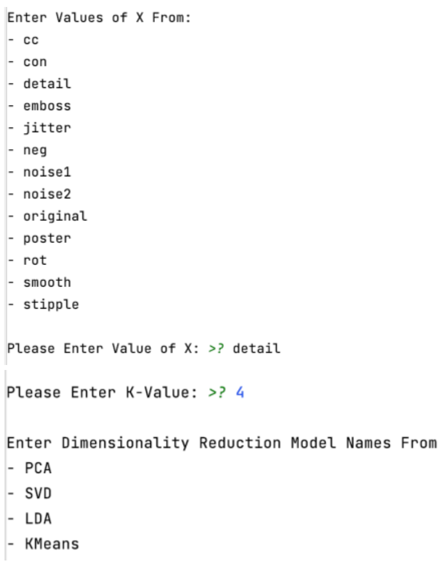
Start Database

Before running any of the tasks, the user should have launched a MongoDB server instance on their localhost network. This can be done by running the ‘mongod.exe’ file located in the \bin folder of the directory they installed MongoDB into. If this is the user’s first time installing and running a MongoDB server, they should also be sure to create the directory MongoDB will save its data into, so there should be a path ‘C:\data\db’. If this path does not exist, use the ‘md’ command to create it. Before running the tasks, the downloaded images for this phase should be moved into the database as well. To do this, open the ‘database.py’ file of our submission and add the line ‘insert_image_dataset($path)’ where $path represents the string of the path where the images have been downloaded to.

Task 1

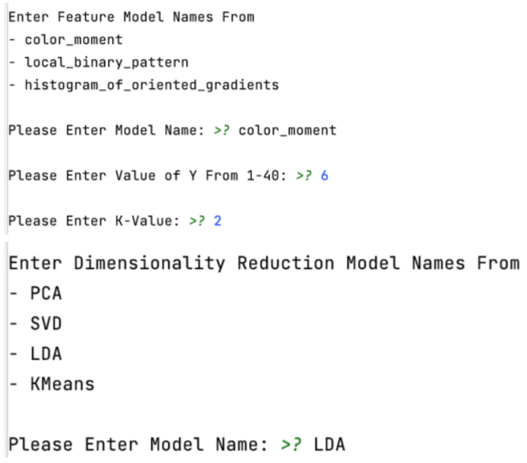
In order to perform Task 1, the python script task_1.py must be executed, for this run this command in command line “python task_1.py”. Upon entering this command, the user will be prompted to input the name of a feature descriptor from (color moments, histogram of oriented gradients, and extended local binary patterns). Then it asks to input Type values from the given set of image types (cc, con, detail, emboss, jitter, neg, noise1, noise2, original, poster, rot, smooth, stipple). Then it takes a k-value as input to report k-Latent Semantics. At last, it asks for a dimensionality reduction algorithm as input from (PCA, SVD, LDA, KMeans)


```
Enter Feature Model Names From
- color_moment
- local_binary_pattern
- histogram_of_oriented_gradients

Please Enter Model Name: >? local_binary_pattern
```


```
Enter Values of X From:
- cc
- con
- detail
- emboss
- jitter
- neg
- noise1
- noise2
- original
- poster
- rot
- smooth
- stipple

Please Enter Value of X: >? detail

Please Enter K-Value: >? 4

Enter Dimensionality Reduction Model Names From
- PCA
- SVD
- LDA
- KMeans
```

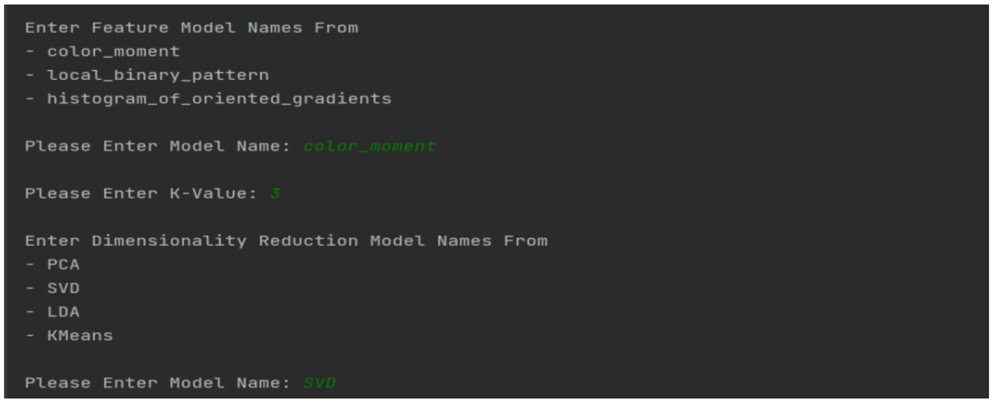
Task 2

In order to perform Task 2, the python script task_2.py must be executed, for this run this command in command line "python task_2.py". Upon entering this command, the user will be prompted to input the name of a feature descriptor from (color moments, histogram of oriented gradients, and extended local binary patterns). Then it asks to input Subject values from 1- 40 as we have 40 subjects in total. Then it takes a k-value as input to report k-Latent Semantics. At last, it asks for a dimensionality reduction algorithm as input from (PCA, SVD, LDA, K-Means)


```
Enter Feature Model Names From
- color_moment
- local_binary_pattern
- histogram_of_oriented_gradients

Please Enter Model Name: >? color_moment

Please Enter Value of Y From 1-40: >? 6

Please Enter K-Value: >? 2

Enter Dimensionality Reduction Model Names From
- PCA
- SVD
- LDA
- KMeans

Please Enter Model Name: >? LDA
```

Task 3

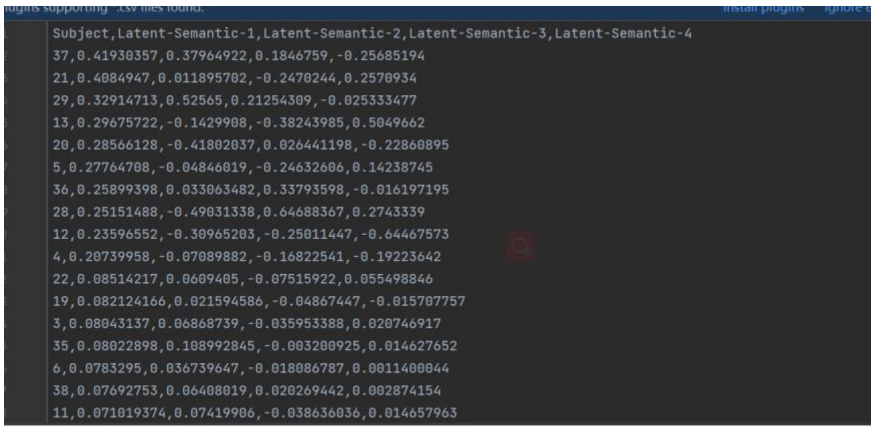
To execute task 3, you must first run the task_3.py python script. Once this is done, the user will be prompted to enter feature model names from one of three options: color moment, local binary pattern, and the histogram of oriented gradients. After entering the model's name, the user will be prompted to enter K-value. The user only needs to input any positive integer value for K-value. After this, the user will be prompted to enter dimensionality reduction model names from one of the four parts------ PCA, SVD, LDA and K-Means. The user needs to enter the model's name from one of the four above.



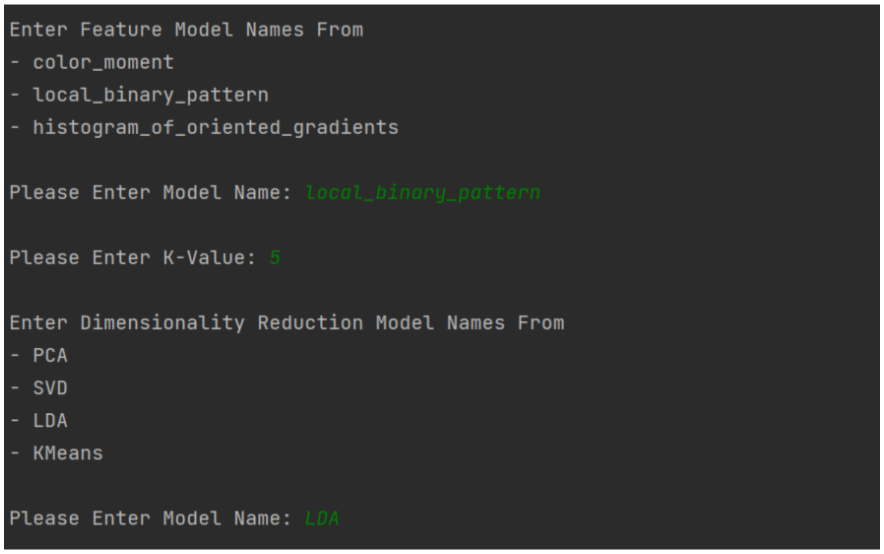
```
Enter Feature Model Names From
- color_moment
- local_binary_pattern
- histogram_of_oriented_gradients

Please Enter Model Name: color_moment

Please Enter K-Value: 3

Enter Dimensionality Reduction Model Names From
- PCA
- SVD
- LDA
- KMeans

Please Enter Model Name: SVD
```

Task 4

To execute task 4, you must first run the task_4.py python script. Once this is done, the user will be prompted to enter feature model names from one of three options: color moment, local binary pattern, and the histogram of oriented gradients. After entering the model's name, the user will be prompted to enter K-value. The user only needs to input any positive integer value for K-value. After this, the user will be prompted to enter dimensionality reduction model names from one of the four parts------ PCA, SVD, LDA and K-Means. The user needs to enter the model's name from one of the four above.



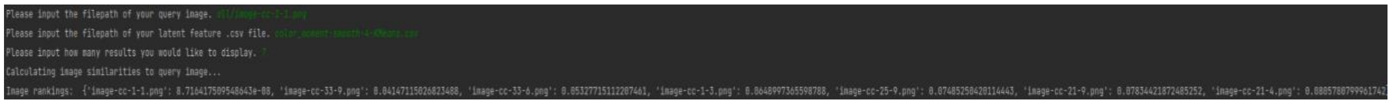
```
Subject,Latent-Semantic-1,Latent-Semantic-2,Latent-Semantic-3,Latent-Semantic-4
37,0.41930357,0.37964922,0.1846759,-0.25685194
21,0.4084947,0.011895702,-0.2470244,0.2570934
29,0.32914713,0.52565,0.21254309,-0.025333477
13,0.29675722,-0.1429908,-0.38243985,0.5049662
20,0.28566128,-0.41802037,0.026441198,-0.22860895
5,0.27764708,-0.04846019,-0.24632606,0.14238745
36,0.25899398,0.033063482,0.33793598,-0.016197195
28,0.25151488,-0.49031338,0.64688367,0.2743339
12,0.23596552,-0.30965203,-0.25011447,-0.64467573
4,0.20739958,-0.07089882,-0.16822541,-0.19223642
22,0.08514217,0.0609405,-0.07515922,0.055498846
19,0.082124166,0.021594586,-0.04867447,-0.015707757
3,0.08043137,0.06868739,-0.035953388,0.020746917
35,0.08022898,0.108992845,-0.003200925,0.014627652
6,0.0783295,0.036739647,-0.018086787,0.0011400044
38,0.07692753,0.06408019,0.020269442,0.002874154
11,0.071019374,0.07419906,-0.038636036,0.014657963
```



```
Enter Feature Model Names From
- color_moment
- local_binary_pattern
- histogram_of_oriented_gradients

Please Enter Model Name: local_binary_pattern

Please Enter K-Value: 5

Enter Dimensionality Reduction Model Names From
- PCA
- SVD
- LDA
- KMeans

Please Enter Model Name: LDA
```

Task 5

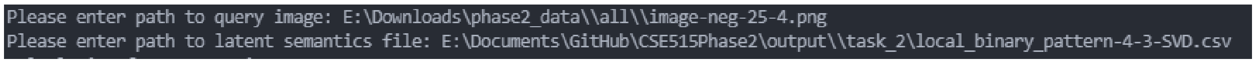
To execute task 5, you must first run the task_5.py python script. Once this is done the user will be prompted to input three values: the file path of the query image, the file path of the latent feature file, and the integer n. The file path of the query image and latent feature file should be input in Unix format. For inputting the integer n, the user only needs to input any positive integer value. Once the user has entered all three specified parameters, task 5 will execute and the user will see the image names with their associated distance rankings displayed to the terminal. The user will also see the visualization of the nearest n source images to the query image.



```
Please input the filepath of your query image. all/image-cc-1-1.png
Please input the filepath of your latent feature .csv file. color_moment-smooth-4-KMeans.csv
Please input how many results you would like to display. 7
Calculating image similarities to query image...
Image rankings:  {'image-cc-1-1.png': 8.716417509548643e-08, 'image-cc-33-9.png': 0.04147115026823488, 'image-cc-33-6.png': 0.05327715112207461, 'image-cc-1-3.png': 0.06489973655987 88, 'image-cc-25-9.png': 0.07485258420114443, 'image-cc-21-9.png': 0.07834421872485252, 'image-cc-21-4.png': 0.08857807999617 42
```

Task 6

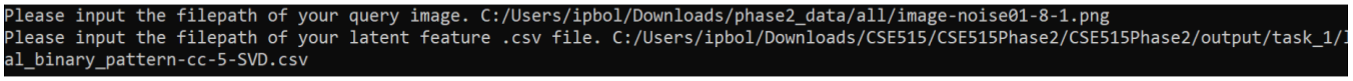
To run task 6, enter the command ‘python task_6.py’. The user will then be asked for a query image path and a latent semantic file path. The expected output will be a string printed to the terminal window that states the most likely type of the query image.



```
Please enter path to query image: E:\Downloads\phase2_data\\all\\image-neg-25-4.png
Please enter path to latent semantics file: E:\Documents\GitHub\CSE515Phase2\output\\task_2\local_binary_pattern-4-3-SVD.csv
```

Task 7

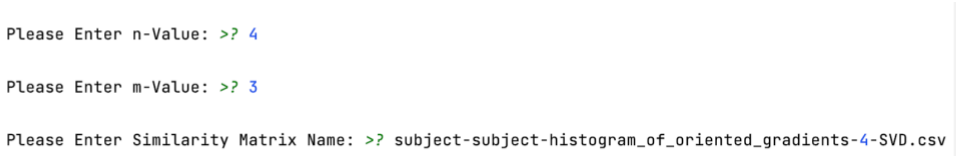
In order to perform Task 7, the python script task_7.py must be executed. Upon entering this command, the user will be prompted for the query image file path and latent semantic file path. The paths are expected to be in a Unix format. The expected output will be a String message printed in the system terminal, containing the most likely Subject ID according to the calculations.



```
Please input the filepath of your query image. C:/Users/ipbol/Downloads/phase2_data/all/image-noise01-8-1.png
Please input the filepath of your latent feature .csv file. C:/Users/ipbol/Downloads/CSE515/CSE515Phase2/CSE515Phase2/output/task_1/l
al_binary_pattern-cc-5-SVD.csv
```

Task 8

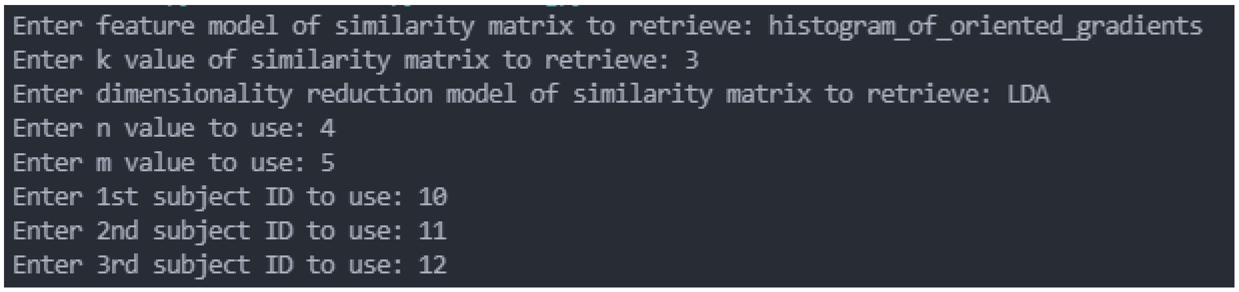
In order to perform Task 8, the python script task_8.py must be executed, for this run this command in command line “python task_8.py”. Upon entering this command, the user will be prompted to input the n-value. This value determines how many nearest nodes one requires while generating the graph. The next input it asks for is the m-value that corresponds to how many significant subjects it requires in the output. The third one requires the name of the similarity matrix file. The similarity matrix file name must exist in the database for it to get loaded.



```
Please Enter n-Value: >? 4

Please Enter m-Value: >? 3

Please Enter Similarity Matrix Name: >? subject-subject-histogram_of_oriented_gradients-4-SVD.csv
```

Task 9

To run task 9, enter the command ‘python task_9.py’. The user should then be prompted for a feature model of the desired subject-subject similarity matrix (this should be represented as one of the following strings: ‘color_moment’, ‘local_binary_pattern’, or ‘histogram_of_oriented_gradients’), k value of the desired subject-subject similarity matrix, dimensionality reduction model of the desired subject-subject similarity matrix (this should be represented as one of the following strings: ‘PCA’, ‘SVD’, ‘LDA’, or ‘K-Means’), n value, m value, and 3 subject IDs. After all inputs are entered, the program will begin execution and print the expected output to the terminal. The expected output includes a string representation of the generated similarity graph and a string representation of the list of m most significant subjects in relation to the given seed subject IDs along with their PPR scores.



```
Enter feature model of similarity matrix to retrieve: histogram_of_oriented_gradients
Enter k value of similarity matrix to retrieve: 3
Enter dimensionality reduction model of similarity matrix to retrieve: LDA
Enter n value to use: 4
Enter m value to use: 5
Enter 1st subject ID to use: 10
Enter 2nd subject ID to use: 11
Enter 3rd subject ID to use: 12
```

Related Work

This phase has implemented dimensionality reduction and feature extraction models from the phase-1 using the vector space model. There are various feature descriptor algorithms in use today out which few are:

- DAISY is a feature descriptor like SIFT formulated in a way that
- allows for fast dense extraction.
- Peak Local Max, find peaks in an image as coordinate list or Boolean mask.
- Binary Robust Independent Elementary Features is an efficient feature point descriptor. It is highly discriminative even when using relatively few bits and is computed using simple intensity difference tests.

In order to calculate the distance among the image vector we used Euclidean Distance Formula. There are other various type of distance/similarity matrix that are in use today out of which few are:

- Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.
- Manhattan distance is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates.
- Minkowski distance is a generalization of the Euclidean and Manhattan distances.
- Jaccard similarity is a statistic used for gauging the similarity and diversity of sample sets.

For this phase we use PCA, SVD, LDA, and K-Means as our dimensionality reduction algorithms. There is various other reduction algorithm that are being used for different cases:

- Factor Analysis is another algorithm that not just reduce the dimensionality of the data. Factor Analysis is a useful approach to find latent variables which are not directly measured in a single variable but rather inferred from other variables in the dataset.
- Kernel PCA is a non-linear dimensionality reduction technique that uses kernels. It can also be considered as the non-linear form of normal PCA.
- MDA is another non-linear dimensionality reduction technique that tries to preserve the distances between instances while reducing the dimensionality of non-linear data.
- Isometric mapping is non-linear dimensionality reduction through Isometric mapping. It is an extension of MDS or Kernel PCA.

We also use ASCOS++ and Personalized Page Rank algorithm for node ranking given a graph of nodes. There is various other node ranking algorithm that are being used for different cases:

- PageRank is a link analysis algorithm that assigns a numerical weight to each object in the information network, with the purpose of “measuring” its relative importance within the object set.
- Hyperlink-Induced Topic Search (HITS) ranks objects based on two scores: authority and hub. Authority estimates the value of the content of the object, whereas hub measures the value of its links to other objects.
- Object Rank aims at ranking the objects according to a keyword-based query in a database.

Conclusion

Dimensionality reduction techniques provide a computationally effective means by which we may query, manipulate, and compute data. As evidenced by the various tasks of this project, these methods become extremely useful in many applications, especially in the context of image feature analysis, selection, and extraction. Tasks 1, 2, 3, and 4 highlight this, as these dimensionality reduction methods are utilized most notably in these sections. Tasks 5, 6, and 7 exemplify the applications of image latent semantic feature analysis. Finally, Tasks 8 and 9 showcase how we may extend the topics of similarity calculation and database significance to a wider scope to include graph representation and implementations. Through these implementations, analyses such as similarity calculations or distance measures can be executed with a considerable effect on execution time. These tasks have demonstrated the usefulness of such techniques, as well as the advantages of one method over another. In addition to the methods most prominent in this project, the implementation of various data structures and their qualities are discussed. Where a vector or matrix representation may not suffice, a graph or other such representation may. The final two Tasks of this project highlight this fact. Through these node/edge relationships, pairwise connections may be drawn between instances. The techniques we employ, data structures we utilize, and other such decisions we make have a profound effect on our data, its representation in the database, as well as our understanding of the results.

Bibliography

1. Candan, K. S. (2021). CSE 515 Multimedia and Web Databases Phase #2 Project Description (pp. 1-2). Published on Piazza
2. Noah Keen. (2005). Color Moments, https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/KEEN/
3. L. Mdakane and F. van den Bergh (2001). Extended Local Binary Pattern Features for Improving Settlement Type Classification of QuickBird Images, CSIR Research Space
4. Huang, S., Li, X., Candan, K. S., Sapino, M. L. (2016). Reducing seed noise in personalized PageRank. Social Network Analysis and Mining, 6(1), 1-25.
5. Hung-Hsuan Chen and C. Lee Giles. 2015. ASCOS++: An Asymmetric Similarity Measure for Weighted Networks to Address the Problem of SimRank. ACM Trans. Knowl. Discov. Data 10, 2, Article 15 (October 2015)

Appendix

During this phase of the project everyone was responsible for implementing certain task and algorithms to keep work divided evenly. All five members coordinate to help validate data outputs and discuss a variety of techniques for calculating intermediary values. Code was maintained on GitHub, and we use online word document to collaborate on report writing task

Preston Mott – Task-5, Task-8 (partial), Graph.py, Report
Brandon Bayles – PCA.py, Task-6, Task-9, Report
Ian Bolton – SVD.py, Task-7, Report
Keenan Rahman – Task-1, Task-2, Task-3 (partial), Task-4, (partial), Task-8 (partial), Report
Kunhao Zhang - Task-3 (partial), Task-4, (partial), Report