

Multimedia and Web Databases

Group 6

Team

1. Aryan Patel (apate294@asu.edu)
2. Tejas Ajay Parse (tparse@asu.edu)
3. Om Patel (opatel14@asu.edu)
4. Tanishque Zaware (tzaware@asu.edu)

Abstract

This phase of the project involves working with the models of phase 1 and using dimensionality reduction techniques to extract latent semantics from the features. The latent semantics are extracted from the features of phase 1 models as well as from the label – label similarity matrices. Four dimensionality reduction techniques are used for extraction of latent semantics, they are: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Latent Dirichlet Allocation (LDA) and Kmeans. The main task was to find either similar videos or similar labels, given an input video/ videoID or a Label along with the user choice for type of latent semantics to use.

Keywords

Multimedia Retrieval, PCA, SVD, K-Means, Latent Semantics.

Introduction

This phase of the project builds upon the feature models created in the previous phase. The main task is to extract latent semantics using PCA, SVD, LDA or Kmeans and retrieve similar videos or video labels. PCA is a dimensionality reduction technique that transforms high dimensional data into low dimensional form while preserving the variance in the data. SVD is a matrix factorization technique used for dimensionality reduction and it preserves the data or data distances. LDA is a supervised dimensionality reduction technique majorly used in document analysis. We will analyze how well this will work in video context. Kmeans is a clustering algorithm that assigns all data to K clusters based on similarity. We have used kmeans in terms of dimensionality reduction by transforming each data object into its distance from the assigned clusters.

Proposed Solution

Task 0a involves assigning and storing unique video ID to both target and non-target videos in a csv mapping file, and further mapping them into six visual feature spaces (R3D18-Layer3-512, R3D18-Layer4-512, R3D18-AvgPool-512, BOF-HOG-480, BOF-HOF-480, COL-HIST) from the previous Phase. The feature vectors corresponding to each feature space have been stored in csv format for ease to fetch. Apart from this for even numbered target videos the category labels are stored as well.

Storage Layout - The video IDs start from 0 and sequentially increment to assign uniquely for subsequent videos. We have videoName column to store file name, label and category columns to store the classification and source of the video. The subsequent columns store the feature vectors corresponding to the videos.

Task 0b involves taking input from the user for video file name or video id (even or odd, target or non target), a feature space from the six listed above and the number of similar videos to fetch. Now we fetch the feature space csv and find the cosine similarity between the query video and other videos using their feature vectors. The similarity scores are stored in a

dictionary with video file names as key and scores as value. Finally we sort the values and take m most similar ones for display and further visualize them accordingly.

Task 1 takes as input a video name or video ID, a feature space from phase 1 models, and 'l' to identify top 'l' similar labels of the videos. Using the csv file of video ID – Name mapping, the task first finds the video ID (if video name was given). It then loads the features of the query video. Now, it finds the cosine similarity between the query video features and features of each even ID target video. We only take the even ID target videos since they are the only ones where their corresponding labels are stored which will be required to predict the similar labels. The similarity value and corresponding label is stored in a list. The similarities are then grouped by the labels. The grouping is done as follows : for each similarity,label pair in the list, the similarity value is added for that label and the count of the label is increased by 1. (For eg : if similarity , label = 0.9 , kick then $\text{sum}[\text{label}] += 0.9$ and $\text{count}[\text{label}] += 1$). We do this for each value in the list so finally we are left with the total similarity sum of each label and label count. Now we normalize the similarity by $\text{sum} / \text{count}$ to get the similarity of each label from query video. Normalization ensures that we don't get biased higher similarity just because the no. of videos under that label were higher(for instance , if label kick has 10 videos each with similarity 0.8 [total:8] while label wave has 50 videos with similarity 0.2[total:10] , addition will give wave to have more similarity then kick). Finally, the list is sorted in descending order and top 'l' labels are printed along with their similarity scores.

We begin Task 2 by taking one of the feature models from **Phase 1**. These feature models include avgpool, layer3, layer4, hog, hof, and col-hist, which are extracted from the output of **task0a**. Next, the user specifies a value for s, which represents the number of features to reduce to during the dimensionality reduction process.

The user is then prompted to select a dimensionality reduction technique from the following options:

-
- **LDA** (Latent Dirichlet Allocation)
 - **PCA** (Principal Component Analysis)
 - **SVD** (Singular Value Decomposition)
 - **K-Means** clustering

Once the user selects both the feature model and the dimensionality reduction method, the appropriate algorithm is applied to the chosen features.

For PCA, We begin by centering the features by subtracting the mean. Then, the covariance matrix of the centered features is computed. Eigenvalues and eigenvectors are derived from the covariance matrix using the numpy library function **eigh**. The eigenvalues are sorted in descending order, and the corresponding eigenvectors are arranged in the same order. Using the top s eigenvectors, the core matrix is formed. We then multiply the centered features with factor matrix to get the reduced feature matrix. These steps result in a reduced feature space that retains the most significant variance of the original features.

For SVD, The features are centered by subtracting the mean. We then compute the covariance matrix for feature to feature mapping matrix and its eigenvalues and eigenvectors using the numpy library function **eigh**. We get the right factor matrix and core matrix. We then compute the left factor matrix by projecting the centered features onto the right singular vectors and normalizing by the singular values. To get the reduced features, we multiply the Left factor matrix and core matrix. This approach gives a compressed feature representation that retains the essential structure of the data.

LDA (Latent Dirichlet Allocation):

The LDA algorithm is applied to the features to reduce them to s latent topics. The LDA transformation provides a representation of the features in terms of topic distributions. We have used `LatentDirichletAllocation` from `sklearn` library to get the reduced features. This transformation is extended to the full dataset for use in future tasks. Additionally, for Avgpool, layer3, layer4 features we have performed a min-max scaling to bring them to range

[0, max_value] where max_value is the maximum possible feature value in features pre scaling. This is done because LDA does not take negative values as input. We do this to maintain the relationship between the values.

K-Means:

K-Means clustering is used to group the features into s clusters. Each feature vector is represented by its proximity to the cluster centroids, resulting in a reduced dimensionality of the feature space. We have used KMeans function from sklearn to get the reduced features.

Output and File Saving

For each dimensionality reduction technique, the relevant matrices and reduced features are saved to files for future tasks:

PCA:

pca_core.npy: The core matrix from the PCA decomposition of even target video features.

pca_factor.npy: The factor matrix from the PCA decomposition.

pca_reduced.csv: This file stores the videoid, label, category, and reduced features of even target videos.

pca_reduced_all.csv: This file stores the reduced features of all videos using the pca_core and pca_factor for future tasks.

SVD:

svd_core.npy: The core matrix from the SVD decomposition of even target video features.

svd_u.npy: The left factor matrix from the SVD decomposition.

svd_v.npy: The right factor matrix from the SVD decomposition.

svd_reduced.csv: This file stores the videoid, label, category, and reduced features of even target videos.

svd_reduced_all.csv: This file stores the reduced features of all videos using the svd_core and svd_v for future tasks.

K-Means:

k_means_weights.pkl: The pickle file containing the K-Means model used to reduce the even target videos.

k_means_reduced.csv: This file stores the videoid, label, category, and reduced features of even target videos.

k_means_reduced_all.csv: This file stores the reduced features of all videos using the K-Means model for future tasks.

LDA:

lda_reduced.csv: This file stores the videoid, label, category, and reduced features of even target videos.

lda_reduced_all.csv: This file stores the reduced features of all videos using the LDA model for future tasks.

Additional Output:

In addition to the above files, for the LDA method, we also output videoid-weight pairs for each latent semantic, ordered by decreasing weight. Given the large size of the output, this information is saved in task_2_output.txt instead of printing to the console.

This proposed solution ensures that the dimensionality reduction process is flexible, allowing the user to select the feature model, the number of reduced features (s), and the dimensionality reduction technique. All results are saved for future use in corresponding files.

Task 3 takes video Id or video name as input and if video name is given than it will find video Id corresponding to the video name using the csv file for video Id to video name mapping. Then it will take inputs for using either feature models from task 0 or latent semantics from

task 2 and will also take m as input for m most similar videos. If feature model from task 0 is selected then it will take the feature model name as input and if latent semantics is selected it will take model, method and s as input where s is top s latent semantics from task 2. Firstly it will filter out data for target videos since we want to find m most similar target videos. Secondly, it will load the features from either feature model or from latent semantics for both the video given as input and all target videos. Now it will compare each target video features with the input video feature and append the target video id, target video name and similarity score to a list where similarity score is calculated based on cosine similarity. Finally, it will sort the list based on the similarity score from highest to lowest and return first m videos. The videos are visualized in a loop where next or previous video can be played using 'd' and 'a' keys and the window can be closed with 's' key. Id and score of the video being played is also written in white on the window.

Task 4 takes any label (target or non target), model, method, s and m as input where model, method and s describes the latent semantics generated in task 2 and m indicates the top m similar target videos for a given label. Firstly, it will find all the videos under a label, either target or non target, by walking through the dataset and finding the folder with the label and storing the video names of all the videos belonging to the given label. Thereafter using the csv file for video id to video name mapping it will give video ids for all videos under the given label. Now to calculate relevance score of a particular target video with the input label, we will calculate average cosine similarity of the target video with all videos under the input label indicating the relevant score of that particular target video. Now we will calculate relevance score of each target video and append it to a list with its video id, video name and relevance score. Finally, we will sort the list based on the relevance score and select first m entries from the list indicating top m most relevant videos for a given input label.

In task 5, we take as input a feature space, a dimensionality reduction technique and number of dimensions to reduce to (s) as input. Using the feature space, we first compute a label – label similarity matrix for target labels. To do so, we take all the videos belonging to each column and take the mean of their features to get a representative feature for each label.

Now, we have 1 feature for each label. We then compute the cosine similarity between every pair of labels to get similarities between each label- label pair. This is then arranged in a matrix form to get a label – label similarity matrix for target labels. The matrix is stored at “Latent Semantics / Task5”. Now, using ‘s’ and dimensionality reduction technique we reduce this label-label similarity matrix to s latent features and store the factor and core matrices where applicable at location “Latent Semantics/Task5”. The implementation of these techniques is same as mentioned in task 2. To print the label – weight pairs, we read the latent semantics column wise and sort the values of the column in descending order. Then we print the sorted values along with their labels. This is done for each latent semantic to get label – weight pairs for each latent semantic.

Task 6 involves taking input from the user for the label (target or non-target), feature model from task 0 or latent semantics from task 2 or task5 and a positive integer I that will be used to display I most similar target_video labels along with their scores.

Using task0 features to identify similar labels is similar to implementation of task 1. However, since input is a label, we first find a representative feature for the label. We take all video features of the label and get their mean to get the representative feature. This task is easy for target labels since we have them stored and we can use them to directly look up the videos from the file. However, in the case of non target labels, we first iterate through the non target videos directory in order to find the label. From here, we save the video names of all the videos belonging to the label. We then use the video mapping csv stored in task0 to lookup the features of these videos . Once we have the video features, we find the representative feature of the label. Now we perform the same steps performed in task 1. In this case our query video feature becomes the label representative feature and we find similar ‘I’ target video labels for this query video feature.

In case the user selects task2 latent semantics, we ask the dimensionality reduction technique and ‘s’ from the user. We use these to dynamically create the latent semantics for the given user specifications. We do this because precomputing every possible combination

of latent semantics takes a lot of time as well as space. This is the case for all tasks that use latent semantics of task 2 or 5. Since input is a label, we find a representative feature for the label using steps mentioned above. This representative feature is then projected into the latent space using the stored data of task 2. If technique is kmeans or lda, we store the model's pickle file. We load this pickle file and use the `model.transform()` method to transform the representative feature to the latent space of kmeans or lda. If the selected technique is pca, we have stored the factor matrix of pca as well. We use this factor matrix to transform the feature using dot product. The same method is used when technique is SVD, we use dot product between the V matrix stored and the representative feature to obtain the transformed feature. Now that we have both the transformed feature as well as the latent semantics, we use the same method described in task 1 to find the most similar 'l' labels.

For latent semantics of task 5, if the input label is target label we directly use the latent semantics (where the features as well as label is stored) to look up the feature vector of the input label and then use cosine distance to find similarity between our feature vector and feature vectors of all other labels. Then, the top 'l' labels with similarities are displayed. Task6 will generate the latent semantics of task 2 or 5 if not present while running the program.

Also, in both Tasks 5 and 6, for LDA implementation, we have used a min max scaler to transform the data before passing it to lda. This is because lda does not accept negative values.

Interface Specifications

Interface description for each task is provided in the readme file attached with the submission. Please refer to the file.

System requirements and Execution Instructions

The code was run with the following version of python and its libraries :

Python : 3.10

Numpy: 1.23.2

Torch: 2.3.1

Torchvision: 0.20.0

Opencv-python: 4.10.0

Scipy: 1.13.1

Scikit-learn: 1.5.0

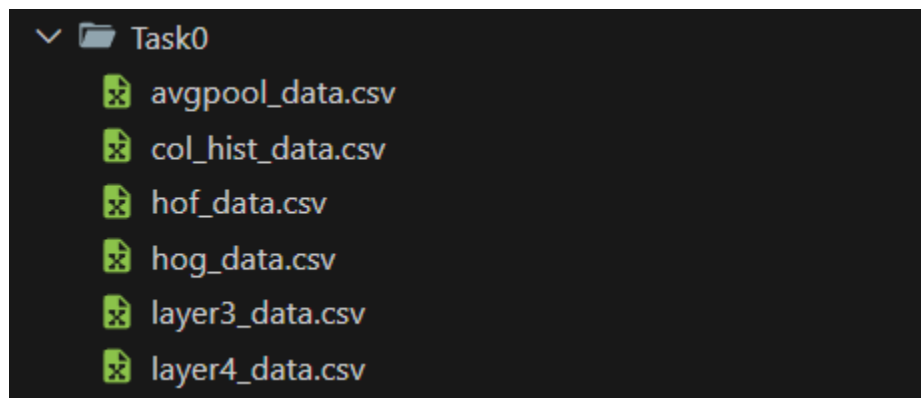
Pandas: 2.2.2

(versions of libraries used)

Detailed execution instructions for each task are provided in the Readme file present in Code Zip File.

Outputs and Discussions

-task0a



The following files should be saved after running Task0a

-task0b

```
(study) tanishquezaware@Tanishques-MacBook-Air Documents % /Users/tanishquezaware/Documents/study/bin/python "/Users/tanishquezaware/Documents/MS/Sem 1/CSE 515 M
ultimedia/Project/Phase 2/Submitted Codes/Task0b.py"
Enter the name of the model from ['avgpool', 'col_hist', 'layer3', 'layer4', 'hof', 'hog'] : layer4
Do you want to query by Video Name (n) or Video ID (i)? Enter 'n' or 'i': n
Enter the video name: crazy_105_sit-ups_situp_f_cm_np2_le_med_1.avi
Enter the number of similar videos to retrieve: 5

Top 5 similar videos for query video or query video_id: crazy_105_sit-ups_situp_f_cm_np2_le_med_1.avi
Video: crazy_105_sit-ups_situp_f_cm_np2_le_med_1.avi, Distance: 0.0
```

```
Video: Sit_ups_and_crunch_situp_f_nm_np1_le_goo_2.avi, Distance: 0.15212860686390828
Video: Ab_Workout_(6_pack_abs)_ab_exercises_for_ripped_abs_situp_f_nm_np1_le_goo_1.avi, Distance: 0.16028015221673275
Video: techniquefrontshot_shoot_bow_u_cm_np1_fr_med_0.avi, Distance: 0.16069606714099804
Video: Mark_Pfoltz_Sets_World_Record_For_sit_ups_situp_f_nm_np1_ri_bad_1.avi, Distance: 0.16183081812611555
```

The user gave an input of situp label video along with feature space and the number of similar videos needed to fetch. Out of the five similar videos, four have situp label and one is shoot_bow which might have been due to similar hand movement. The model would have learned this behaviour of hand action so found it similar.

```
(study) tanishquezaware@Tanishques-MacBook-Air Documents % /Users/tanishquezaware/Documents/study/bin/python "/Users/tanishquezaware/Documents/MS/Sem 1/CSE 515 M
ultimedia/Project/Phase 2/Submitted Codes/Task0b.py"
Enter the name of the model from ['avgpool', 'col_hist', 'layer3', 'layer4', 'hof', 'hog'] : layer3
Do you want to query by Video Name (n) or Video ID (i)? Enter 'n' or 'i': n
Enter the video name: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_8.avi
Enter the number of similar videos to retrieve: 4

Top 4 similar videos for query video or query video_id: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_8.avi
Video: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_8.avi, Distance: -4.440892098500626e-16
2024-10-23 20:56:39.363 Python[40049:1842922] +[IMKClient subclass]: chose IMKClient_Legacy
2024-10-23 20:56:39.363 Python[40049:1842922] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
Video: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_9.avi, Distance: 0.016184445689874538
Video: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_1.avi, Distance: 0.01753018218316904
Video: ArcheryFastShooting_shoot_bow_u_nm_np1_fr_med_20.avi, Distance: 0.025162886970422238
```

The input video for shoot_bow gave all the labels from shoot_bow along with scores.

```
(study) tanishquezaware@Tanishques-MacBook-Air Documents % /Users/tanishquezaware/Documents/study/bin/python "/Users/tanishquezaware/Documents/MS/Sem 1/CSE 515 M
ultimedia/Project/Phase 2/Submitted Codes/Task0b.py"
Enter the name of the model from ['avgpool', 'col_hist', 'layer3', 'layer4', 'hof', 'hog'] : avgpool
Do you want to query by Video Name (n) or Video ID (i)? Enter 'n' or 'i': n
Enter the video name: Turnk_r_Pippi_Michel_somersault_f_cm_np2_ri_med_4.avi
Enter the number of similar videos to retrieve: 6

Top 6 similar videos for query video or query video_id: Turnk_r_Pippi_Michel_somersault_f_cm_np2_ri_med_4.avi
Video: Turnk_r_Pippi_Michel_somersault_f_cm_np2_ri_med_4.avi, Distance: -4.440892098500626e-16
2024-10-23 21:00:32.205 Python[40150:1845695] +[IMKClient subclass]: chose IMKClient_Legacy
2024-10-23 21:00:32.205 Python[40150:1845695] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
Video: Turnk_r_Pippi_Michel_somersault_f_cm_np2_ri_med_3.avi, Distance: 0.04950807017039738
Video: Turnk_r_Pippi_Michel_handstand_f_cm_np2_le_med_5.avi, Distance: 0.07222098606413896
Video: Bodenturnen_somersault_f_cm_np1_ri_med_1.avi, Distance: 0.10957650645652983
Video: Turnk_r_Pippi_Michel_somersault_f_cm_np1_fr_med_0.avi, Distance: 0.113393739988313196
Video: Zwei_hoffnungslose_Pflegeel_lle_beim_Turnen(Part1)_handstand_f_cm_np1_ri_bad_0.avi, Distance: 0.11552329265670569
```

Same for somersault the model gave 4 similar label output and 2 from handstand due to similar hand and leg movement.

-task 1

```
PS C:\Users\aryan\Desktop\Phase 2> python task1.py
Enter the video ID or Video name : 4830

Select a Feature Model:
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 3
You selected: R3D18-AvgPool-512
Enter l : 4
label : golf | Score : 0.8430252052332786
label : shoot_bow | Score : 0.8390369612449271
label : shoot_gun | Score : 0.8164725463512439
label : brush_hair | Score : 0.8135881562077893
PS C:\Users\aryan\Desktop\Phase 2> 
```

The input video ID was a video of label shoot_bow. In the top 5 labels, we get both labels related to shoot. The top 4 labels have 1 thing in common, which is that videos in each label have a person holding something in his hand (a bow, gun , golf stick or a comb). The model may have learned this behaviour to get the labels.

```

PS C:\Users\aryan\Desktop\Phase 2> python task1.py
Enter the video ID or Video name : Best_Of_Skype_Laughter_Chain_laugh_h_nm_np2_fr_med_19.avi

Select a Feature Model:
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 2
You selected: R3D18-Layer4-512
Enter l : 5
label : laugh | Score : 0.8191422824286129
label : smoke | Score : 0.8026774783751005
label : smile | Score : 0.800064898654731
label : chew | Score : 0.7939126169582589
label : clap | Score : 0.784024135228393
PS C:\Users\aryan\Desktop\Phase 2>

```

This time, the input was a video name. The video belongs to laugh label and feature space was layer 4. The topmost label is laugh which is correct. The next 3 labels have videos that involve movement of the mouth or face so it is natural for them to be included as well. For clap label, it was seen that a lot of videos have people clapping while smiling. This may have been detected.

```

PS C:\Users\aryan\Desktop\Phase 2> python task1.py
Enter the video ID or Video name : 2095

Select a Feature Model:
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 3
You selected: R3D18-AvgPool-512
Enter l : 5
label : smoke | Score : 0.7664544760762858
label : laugh | Score : 0.7605681852766939
label : clap | Score : 0.7573522012316956
label : hug | Score : 0.7542518618836667
label : sit | Score : 0.7531216191005023
PS C:\Users\aryan\Desktop\Phase 2>

```

The input is a non target video of label wave. The videos in wave has some hand movement as well as face movement (may be smiling). This may explain how smoke (hand movement) and laugh(smile) are the top 2 labels.

-task2

The following files should be created if we take 'hog' as feature input, 's' as 7 run for all dimensionality reduction techniques. We take input from user. We ask for Feature Model, top 's' features that needs to be reduced to, the dimensionality reduction technique.

```
Tejas@DESKTOP-MU2V8NC MINGW64 /e/Coding/MultimediaWebDatabases/Phase 2
$ C:/Users/Tejas/anaconda3/envs/MMDenv/python.exe "e:/Coding/MultimediaWebDatabases/Phase 2/Submitted Codes/Task2.py"

Task 2

Pick a Feature Model from the menu
1: AvgPool
2: Layer3
3: Layer4
4: HOG
5: HOF
6: COL-HIST

Pick a feature model: 4
Now Pick the top-s features to be reduced to:
Input: 7

Pick a Dimensionality Reduction Technique from the menu:
1: LDA
2: PCA
3: SVD
4: K-Means

Pick a technique: 2
```

```
Tejas@DESKTOP-MU2V8NC MINGW64 /e/Coding/MultimediaWebDatabases/Phase 2
$ C:/Users/Tejas/anaconda3/envs/MMDenv/python.exe "e:/Coding/MultimediaWebDatabases/Phase 2/Submitted Codes/Task2.py"

Task 2

Pick a Feature Model from the menu
1: AvgPool
2: Layer3
3: Layer4
4: HOG
5: HOF
6: COL-HIST

Pick a feature model: 4
Now Pick the top-s features to be reduced to:
Input: 7

Pick a Dimensionality Reduction Technique from the menu:
1: LDA
2: PCA
3: SVD
4: K-Means

Pick a technique: 4
```

```
Tejas@DESKTOP-MU2V8NC MINGW64 /e/Coding/MultimediaWebDatabases/Phase 2
$ C:/Users/Tejas/anaconda3/envs/MwDenv/python.exe "e:/Coding/MultimediaWebDatabases/Phase 2/Submitted Codes/Task2.py"

Task 2

Pick a Feature Model from the menu
1: AvgPool
2: Layer3
3: Layer4
4: HOG
5: HOF
6: COL-HIST

Pick a feature model: 4
Now Pick the top-s features to be reduced to:
Input: 7

Pick a Dimensionality Reduction Technique from the menu:
1: LDA
2: PCA
3: SVD
4: K-Means

Pick a technique: 1

Tejas@DESKTOP-MU2V8NC MINGW64 /e/Coding/MultimediaWebDatabases/Phase 2
$ C:/Users/Tejas/anaconda3/envs/MwDenv/python.exe "e:/Coding/MultimediaWebDatabases/Phase 2/Submission/Code/Task2.py"

Task 2

Pick a Feature Model from the menu
1: AvgPool
2: Layer3
3: Layer4
4: HOG
5: HOF
6: COL-HIST

Pick a feature model: 4
Now Pick the top-s features to be reduced to:
Input: 7

Pick a Dimensionality Reduction Technique from the menu:
1: LDA
2: PCA
3: SVD
4: K-Means

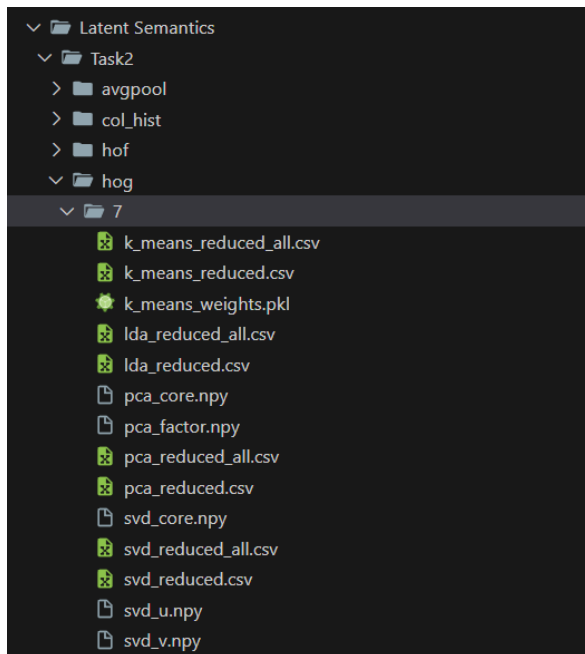
Pick a technique: 3
```

Additionally, Task 2 also generate a task_2_output.txt, that store the ordered videoId-weight pairs for each latent semantic. The file is uploaded in Output Folder of submission

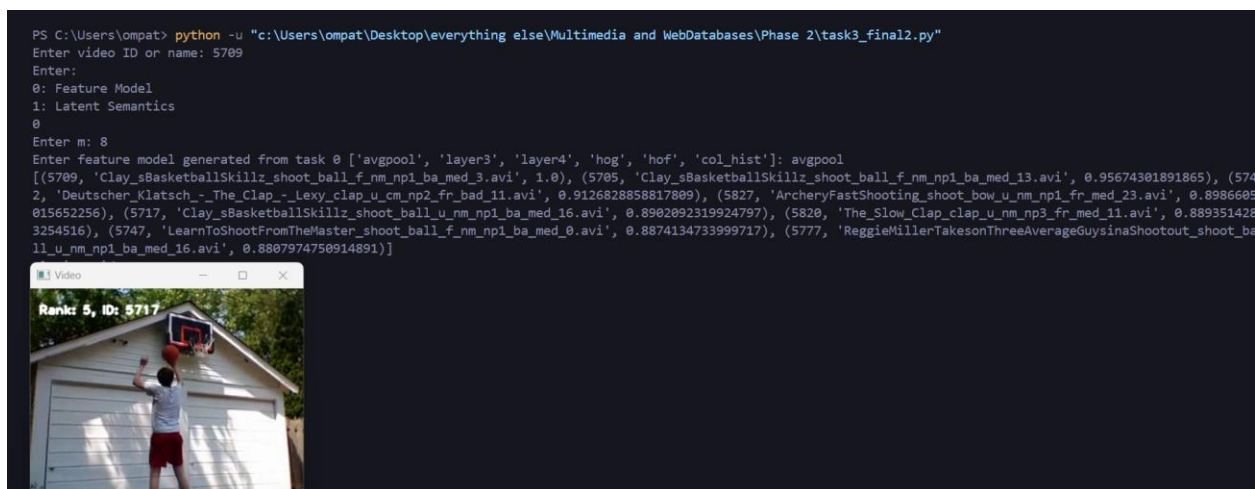
```

task_2_output.txt
1
2 Latent Semantic 1 (ordered by weights):
3 Video ID: 6564, Weight: 23.1872
4 Video ID: 5230, Weight: 22.9634
5 Video ID: 6030, Weight: 22.7903
6 Video ID: 6626, Weight: 22.3959
7 Video ID: 6554, Weight: 22.2056
8 Video ID: 4708, Weight: 22.0508
9 Video ID: 6026, Weight: 21.9678

```



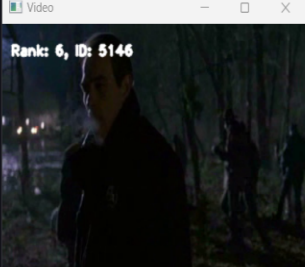
-task 3



The inputs for this are: video ID = 5709, task 0 feature models, m = 8, model = average pool. In this specific combination the video itself is shown as first since the input video itself is a

target video hence it should have maximum cosine similarity with itself. Also one shot bow video is identified as similar to shoot ball, this is due the model learning patterns of equipment usage with hands and it also includes clap video.

```
PS C:\Users\ompat> python -u "c:\Users\ompat\Desktop\everything else\Multimedia and WebDatabases\Phase 2\task3_final2.py"
Enter video ID or name: 1000
Enter:
0: Feature Model
1: Latent Semantics
1
Enter m: 8
{1: 'avgpool', 2: 'layer3', 3: 'layer4', 4: 'hog', 5: 'hof', 6: 'col_hist'}
Enter model from task 2 outputs: 6
{1: 'lda', 2: 'pca', 3: 'svd', 4: 'k_means'}
Enter method from task 2 outputs: 2
Enter s from task 2 outputs: 152
[(5497, 'Glory_smile_h_nm_np1_le_goo_6.avi', 0.626440919299714), (5730, 'Faith_Rewarded_clap_u_cm_np1_le_med_0.avi', 0.5841311940084519), (5837, 'Awesome_Amazing_Creat_Soccer_Free_Kicks_kick_ball_f_cm_np1_ba_bad_0.avi', 0.5483727696980427), (5683, 'BATMAN_BEGINS_hit_u_cm_np1_fr_bad_6.avi', 0.5279396382690573), (6449, 'Glory_sit_u_cm_np1_le_med_27.avi', 0.5066019445350882), (5146, 'The_Fugitive_2_smoke_u_cm_np1_fr_bad_22.avi', 0.5038477243481843), (4484, 'CharlieAndTheChocolateFactory_jump_f_nm_np1_fr_bad_5.avi', 0.503796913725996), (5339, 'Crash_stand_f_cm_np1_rh_bad_12.avi', 0.4929735225518989)]
```




In this case the inputs given are: video id = 1000, m=8, task 2 latent semantics: color histogram, pca and s = 152. The input video is not on the list because the video itself is a non target video and here we are identifying m most similar target videos. Moreover we can notice that the identified videos have dark environmental setting which is also the case in given input video and hence the color histogram feature leads to finding similar color composition videos.

```

PS C:\Users\ompat> python -u "c:\Users\ompat\Desktop\everything else\Multimedia and WebDatabases\Phase 2\task3_final2.py"
Enter video ID or name: Free_Hugs_-_Paris_www_calins-gratuits_com_hug_u_cm_np2_le_med_12.avi
Enter:
0: Feature Model
1: Latent Semantics
1
Enter m: 6
{1: 'avgpool', 2: 'layer3', 3: 'layer4', 4: 'hog', 5: 'hof', 6: 'col_hist'}
Enter model from task 2 outputs: 3
{1: 'lda', 2: 'pca', 3: 'svd', 4: 'k_means'}
Enter method from task 2 outputs: 4
Enter s from task 2 outputs: 202
[(6757, 'Free_Hugs_-_Paris_www_calins-gratuits_com_hug_u_cm_np2_le_med_12.avi', 1.0), (6748, 'Price_giving_ceremony_hug_f_cm_np2_le_med_0.avi', 0.9988681349990836), (6708, 'Free_Hugs_-_Paris_www_calins-gratuits_com_hug_f_cm_np2_ba_med_4.avi', 0.9985597231332367), (3997, 'kick_Best_CHUCK_NORRIS_kick_to_the_nuts!_kick_f_cm_np1_ba_med_3.avi', 0.9984079982863074), (3959, 'kick_Best_CHUCK_NORRIS_kick_to_the_nuts!_kick_f_cm_np1_ba_goo_0.avi', 0.998357755215233), (6744, 'Calins_gratuits_a_Paris_-_Free_Hugs_France_-_version_longue_hug_u_cm_np2_ba_med_13.avi', 0.9982756253644278)]

```



In this output instance we gave video name as input and selected: latent semantics: layer 4, kmeans and s=202, m =6. The given video is a target video and hence is first on the list and using the given combination we are able to identify 3 of the 5 remaining videos which have the action of hugging. In case of the kick video, it is of a scene where a person is kicking another in close proximity, however it is identified as similar to hug video because in that particular scene when the person kicks another person, they come closer in proximity which is similar to pattern of hugging.

-task4

```

PS C:\Users\ompat> python -u "c:\Users\ompat\Desktop\everything else\Multimedia and WebDatabases\Phase 2\task4_final.py"
Enter the label: cartwheel
{1: 'avgpool', 2: 'layer3', 3: 'layer4', 4: 'hog', 5: 'hof', 6: 'col_hist'}
Enter model from task 2 outputs: 1
{1: 'lda', 2: 'pca', 3: 'svd', 4: 'k_means'}
Enter method from task 2 outputs: 4
Enter s from task 2 outputs: 103
Enter m: 7
VideoID: 4631, Filename: (Rad)Schlag_die_Bank!_cartwheel_f_cm_np1_le_med_0.avi, Relevance Score: 1.0
VideoID: 4723, Filename: Cartwheel_Contest_cartwheel_f_cm_np1_fr_med_4.avi, Relevance Score: 0.9991075151207035
VideoID: 6619, Filename: Increase_stride_length_by_running_stairs_working_step_over_action_speed_development_climb_stairs_f_cm_np1_le_med_2.avi, Relevance Score: 0.9989996429700821
VideoID: 4663, Filename: Simon_Bodenturnen_cartwheel_f_cm_np1_ba_med_2.avi, Relevance Score: 0.9989849854922211
VideoID: 4631, Filename: (Rad)Schlag_die_Bank!_cartwheel_f_cm_np1_le_med_0.avi, Relevance Score: 0.9989654037205866
VideoID: 4713, Filename: Acrobacias_de_un_fenomeno_cartwheel_f_cm_np1_ba_bad_8.avi, Relevance Score: 0.9989654037205865
VideoID: 4664, Filename: anna_turnen_cartwheel_f_cm_np1_le_bad_0.avi, Relevance Score: 0.9989293797226573
PS C:\Users\ompat>

```

In this instance, the input label is given as cartwheel, latent semantics are: model = average pool, method = k means, s = 103 and m = 7, here the label given is under target video section and since we have to identify most relevant target videos, the 7 videos identified are

cartwheel videos. Also one climbing stairs video is identified under cartwheel label because the action of running up the stairs is similar to start of cartwheel and hence finds similarity with each cartwheel video so its relevant score increases.

```
PS C:\Users\ompat> python -u "c:\Users\ompat\Desktop\everything else\Multimedia and WebDatabases\Phase 2\task4_final.py"
Enter the label: eat
{1: 'avgpool', 2: 'layer3', 3: 'layer4', 4: 'hog', 5: 'hof', 6: 'col_hist'}
Enter model from task 2 outputs: 1
{1: 'lda', 2: 'pca', 3: 'svd', 4: 'k_means'}
Enter method from task 2 outputs: 4
Enter s from task 2 outputs: 103
Enter m: 6
VideoID: 5298, Filename: Brushing_Her_Hair__[NEW_AUDIO_]UPDATED!!!!_brush_hair_h_cm_np1_le_goo_1.avi, Relevance Score: 0.999168082293652
VideoID: 5252, Filename: sarah_brushing_her_hair_brush_hair_h_cm_np1_ri_goo_0.avi, Relevance Score: 0.999022641525695
VideoID: 5287, Filename: brushing_hair_2_brush_hair_h_nm_np1_ba_med_2.avi, Relevance Score: 0.9989992564698426
VideoID: 5806, Filename: Martin_klatscht_in_die_hnde_und_FURZT_clap_u_cm_np1_fr_med_0.avi, Relevance Score: 0.9989292180756238
VideoID: 5794, Filename: Faith_Rewarded_clap_u_cm_np1_fr_med_43.avi, Relevance Score: 0.9989004074424762
VideoID: 5285, Filename: brushing_jrs_hair_brush_hair_u_cm_np2_le_goo_0.avi, Relevance Score: 0.9988390000024764
PS C:\Users\ompat>
```

In this case label given as input is eat and the latent semantics are: model = average pool, method = kmeans, s = 103, m = 6. The label given is a non target label and videos we have to identify are all target videos hence none of the videos under the given labels will be in the list. Moreover, most of the relevant videos identified are of brushing hair which have a similar hand motion to eating and clap which include movement of hands.

```
PS C:\Users\ompat> python -u "c:\Users\ompat\Desktop\everything else\Multimedia and WebDatabases\Phase 2\task4_final.py"
Enter the label: climb
{1: 'avgpool', 2: 'layer3', 3: 'layer4', 4: 'hog', 5: 'hof', 6: 'col_hist'}
Enter model from task 2 outputs: 5
{1: 'lda', 2: 'pca', 3: 'svd', 4: 'k_means'}
Enter method from task 2 outputs: 4
Enter s from task 2 outputs: 100
Enter m: 5
VideoID: 5988, Filename: (HQ)_Rock_Climbing_-_Free_Solo_Speed_Climb_-_Dan_Osman_climb_f_cm_np1_ba_med_0.avi, Relevance Score: 1.0
VideoID: 3985, Filename: THE_PROTECTOR_kick_f_nm_np1_ba_bad_17.avi, Relevance Score: 0.9998806115140921
VideoID: 4473, Filename: RATRACE_jump_f_cm_np1_fr_bad_38.avi, Relevance Score: 0.9996962091213998
VideoID: 4429, Filename: RATRACE_jump_f_cm_np1_fr_bad_39.avi, Relevance Score: 0.9995918261895557
VideoID: 5858, Filename: The_Monk_Best_Bits_2_kick_ball_f_cm_np2_ri_med_2.avi, Relevance Score: 0.9995284327225115
PS C:\Users\ompat>
```

In this case label given in climb and latent semantics are: model = hof, method = kmeans, s = 100, m = 5. The videos identified here as relevant are videos which capture full body motion and hence are identified to be relevant under climb label because of the similarity of full body motion and rotation in the video.

-task5

For this task, the saved label-label similarity matrices as well as the factor and core matrix for the below given input can be found in the Outputs folder of our submission. The full label-weight pairs that are printed can be found in the appendix section

(input1 : avgpool,Kmeans,6 components)

```
PS C:\Users\aryan\Desktop\Phase 2> python task5.py
Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 3
You selected: R3D18-AvgPool-512
Label-label similarity matrix saved to : Latent Semantics/Task5\label_label_similarity_matrix.csv
Enter the dimensionality reduction technique you want to use from the following:
1: PCA
2: SVD
3: LDA
4: KMeans
4
Enter the number of components (s): 6
Latent semantics saved to Latent Semantics/Task5\latent_semantics_KMeans_6_components_avgpool_data.csv

Label-Weight Pairs (sorted by weight for each latent semantic):

Latent Semantic 1:
cartwheel - 0.2288
somersault - 0.1892
catch - 0.1883
golf - 0.1742
handstand - 0.1712
jump - 0.1593
shoot_ball - 0.1552
```

(input2 : Col-Hist,PCA,3 components)

```
PS C:\Users\aryan\Desktop\Phase 2> python task5.py
Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 4
You selected: col - hist
Label-label similarity matrix saved to : Latent Semantics/Task5\label_label_similarity_matrix.csv
Enter the dimensionality reduction technique you want to use from the following:
1: PCA
2: SVD
3: LDA
4: KMeans
1
Enter the number of components (s): 3
Latent semantics saved to Latent Semantics/Task5\latent_semantics_PCA_3_components_col_hist_data.csv

Label-Weight Pairs (sorted by weight for each latent semantic):

Latent Semantic 1:
golf - 1.1628
catch - 0.8994
```

(input3 : HOG,LDA,2 components)

```
PS C:\Users\aryan\Desktop\Phase 2> python task5.py
Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 6
You selected: BOF-HOG-480
Label-label similarity matrix saved to : Latent Semantics/Task5\label_label_similarity_matrix.csv
Enter the dimensionality reduction technique you want to use from the following:
1: PCA
2: SVD
3: LDA
4: KMeans
3
Enter the number of components (s): 2
here1
Latent semantics saved to Latent Semantics/Task5\latent_semantics_LDA_2_components_hog_data.csv

Label-Weight Pairs (sorted by weight for each latent semantic):

Latent Semantic 1:
cartwheel - 0.0271
brush_hair - 0.0260
```

The output for all these 3 inputs is as described in task 5 implementation.

-Task 6

```
PS C:\Users\aryan\Desktop\Phase 2> python task6.py
Enter a label : laugh
Enter l : 3

Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 3
You selected: R3D18-AvgPool-512

What would you like to use ?
1 : Feature Space (from task 0)
2 : Latent Semantics
1
printing top 3 similar labels :
label : laugh | Score : 0.8997822094495477
label : smile | Score : 0.8908031635188038
label : smoke | Score : 0.8902570082346444
```

Here, we gave 'laugh' label as input , avgpool feature space to find top 3 most similar labels. Laugh is the most similar, and the next 2 are smile (which is naturally similar to laugh) and smoke, which may be present because of similar mouth and lips movement to smile or laugh.

```

PS C:\Users\aryan\Desktop\Phase 2> python task6.py
Enter a label : shoot_bow
Enter l : 3

Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 3
You selected: R3D18-AvgPool-512

What would you like to use ?
1 : Feature Space (from task 0)
2 : Latent Semantics
2

Which Latent Semantics would you like to use ?
1 : From Task 2
2 : From Task 5
1
Enter s : 300
Enter Dimensionality reduction method
1 : LDA
2 : PCA
3 : SVD
4 : Kmeans
Select any one : 4
printing top 3 similar labels :
label : shoot_bow | Score : 0.9925785145434266
label : sit | Score : 0.9924331630049015
label : shoot_gun | Score : 0.9918107223167799

```

Our input label was shoot_bow and we got as output the top label as shoot_bow as well. We also got shoot_gun as a similar label, which has the same shooting action as shoot_bow. I could not find the reason for the label sit to be included here. Also, the dynamically created latent semantics for this input can be found in Outputs folder.

```

PS C:\Users\aryan\Desktop\Phase 2> python task6.py
Enter a label : cartwheel
Enter l : 3

Enter feature model to use :
1. R3D18-Layer3-512
2. R3D18-Layer4-512
3. R3D18-AvgPool-512
4. col - hist
5. BOF-HOF-480
6. BOF-HOG-480
7. Exit
Enter a number (1-6): 2
You selected: R3D18-Layer4-512

What would you like to use ?
1 : Feature Space (from task 0)
2 : Latent Semantics
2

Which Latent Semantics would you like to use ?
1 : From Task 2
2 : From Task 5
2

Enter the dimensionality reduction technique you want to use from the following:
1: PCA
2: SVD
3: LDA
4: KMeans
1

Enter the number of components (s): 10
Loaded latent semantics from Latent Semantics/Task5\latent_semantics_PCA_10_components_layer4_data.csv

Top 3 most similar target video labels to 'cartwheel':
cartwheel - Similarity score: 1.0000
somersault - Similarity score: 0.8990
catch - Similarity score: 0.8119

```

Based on the above input we got top 3 similar labels as cartwheel (the input label itself), somersault, and catch. Both cartwheel and somersault have similar actions so they are expected here. Their presence means the action is captured by the model. Catch is also present but its similarity is a lot lower than the other 2 . The latent semantics can be found in Outputs folder.

Related Work

1. **[1]** This paper discusses on how to handle dimensionality curse and its use in large and high dimensional datasets. It involves use of Balanced Box Decomposition tree for fast and better efficiency in finding the nearest neighbours.
2. **[2]** This paper talks about an efficient method of reducing the dimension of the data while retaining the structure. It mainly works on the distance between the points as it goes on reducing the dimensions by projecting on the line between pivots.

Conclusion

This phase of the project is focused on working on features that were extracted in phase 1 and introduces new methods like dimensionality reduction techniques (PCA, SVD, LDA, KMeans). This phase involves different tasks varying from assigning video IDs, storing the features and latent semantics, finding label-label similarity matrix to label weight pairs and displaying similar videos with scores. We found that in majority of the tasks, avgpool layer works the best.

Bibliography

1. Indyk, Piotr, and Rajeev Motwani. "Approximate Nearest Neighbors." *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC '98*, 1998
2. Faloutsos, C., Lin, K.-I., Christos FaloutsosAT&T Bell Laboratories, M. H., & King-Ip LinDept. of Computer Science, Univ. of M. (1995, May 22). *Fastmap: Proceedings of the 1995 ACM SIGMOD international conference on management of data*.

Appendix

Specific roles of the group members

Aryan Patel - Task 1, Task 5 and (task2 and task5 part of task6)

Tejas Parse - Task 2 and helped with Task 0. Also responsible to create the Readme file.

Om Patel - Task 3 and Task 4

Tanishque Zaware - Task 0 and Task0 part of task 6

- **Modification of Phase1 Task3**

In the updated version of task 3 of phase 1, the color histogram is represented by set of 12 bins under the CIE Lab color space. Firstly, we extract first, middle and last frame for each video under the target category from phase 1. Now to identify 12 colors to represent the 12 bins of histogram, we will find 12 most dominant colors on all the frames. To do this, first we convert each frame from rgb color space to CIE Lab color space which is represented by (l,a,b) where l indicates the lightness, a is chromatic index for red and green and b is chromatic index for blue and yellow. Thereafter we convert each frame into 2d Array of pixels which are then stacked to get single large array where each pixel is in (l,a,b) format. Using kmeans clustering we cluster each pixel into 12 different colors to get the 12 globally dominant colors. To create a color histogram feature for a video, we take first, middle and last frame of the video and divide it into 16 cells with equal height and width. Now we convert each cell into CIE Lab color space and we assign each pixel of the cell to one of the 12 bins using Euclidean distance. This way we get a count of total pixels in 12 bins for each cell. Finally we concatenate the pixel count for all cells and then with remaining two frames to get color histogram feature of the whole video.

- Since we've received new target labels for this phase, the HoF and HoG cluster representatives are recalculated. They are submitted in Outputs Folder.

- **Outputs**

- task 5**

- (first input)

- Latent Semantic 1:

- cartwheel - 0.2288

- somersault - 0.1892

- catch - 0.1883

- golf - 0.1742

- handstand - 0.1712

- jump - 0.1593

- shoot_ball - 0.1552

- hit - 0.1491

- kick_ball - 0.1479

- climb - 0.1285

- chew - 0.1278

- climb_stairs - 0.1273

- brush_hair - 0.1256

- laugh - 0.1236

- smile - 0.1230

- shoot_gun - 0.1214

- situp - 0.1210

- stand - 0.1196

- smoke - 0.1175

shoot_bow - 0.1169

sit - 0.1086

kick - 0.1013

clap - 0.0938

hug - 0.0000

Latent Semantic 2:

chew - 0.1981

laugh - 0.1978

smile - 0.1921

smoke - 0.1803

cartwheel - 0.1746

clap - 0.1525

catch - 0.1387

golf - 0.1318

brush_hair - 0.1290

shoot_gun - 0.1272

stand - 0.1259

hug - 0.1243

situp - 0.1198

sit - 0.1163

somersault - 0.1137

shoot_bow - 0.1097

handstand - 0.1062

jump - 0.1043

kick_ball - 0.0954

hit - 0.0876

shoot_ball - 0.0870

kick - 0.0854

climb_stairs - 0.0302

climb - 0.0302

Latent Semantic 3:

chew - 0.2540

smile - 0.2531

laugh - 0.2492

smoke - 0.2469

stand - 0.2131

sit - 0.2101

brush_hair - 0.2067

clap - 0.2060

shoot_gun - 0.2008

hug - 0.1856

shoot_bow - 0.1736

situp - 0.1635

jump - 0.1587

kick - 0.1367

climb - 0.1318

climb_stairs - 0.1281

hit - 0.1210

kick_ball - 0.1091

handstand - 0.0965

shoot_ball - 0.0893
cartwheel - 0.0761
golf - 0.0697
somersault - 0.0557
catch - 0.0484

Latent Semantic 4:

cartwheel - 0.2926
somersault - 0.2566
catch - 0.2509
handstand - 0.2296
golf - 0.2248
shoot_ball - 0.2200
kick_ball - 0.2177
jump - 0.2137
hit - 0.2040
climb_stairs - 0.1990
climb - 0.1863
kick - 0.1761
stand - 0.1397
shoot_gun - 0.1396
shoot_bow - 0.1392
sit - 0.1343
situp - 0.1341
brush_hair - 0.1251
hug - 0.1201

clap - 0.0842
smoke - 0.0378
laugh - 0.0255
chew - 0.0227
smile - 0.0153

Latent Semantic 5:

cartwheel - 0.2492
catch - 0.1880
somersault - 0.1838
golf - 0.1720
handstand - 0.1429
shoot_ball - 0.1365
chew - 0.1340
laugh - 0.1304
kick_ball - 0.1279
climb_stairs - 0.1261
smile - 0.1224
climb - 0.1120
hit - 0.1110
jump - 0.1062
hug - 0.1035
kick - 0.1024
smoke - 0.0957
situp - 0.0760
clap - 0.0639

stand - 0.0473

sit - 0.0459

shoot_bow - 0.0443

shoot_gun - 0.0359

brush_hair - 0.0345

Latent Semantic 6:

chew - 0.2149

laugh - 0.2099

smile - 0.2073

smoke - 0.1913

cartwheel - 0.1726

clap - 0.1464

hug - 0.1395

brush_hair - 0.1223

sit - 0.1218

situp - 0.1213

stand - 0.1194

golf - 0.1173

shoot_gun - 0.1139

catch - 0.1077

shoot_bow - 0.1038

somersault - 0.0919

climb - 0.0895

climb_stairs - 0.0792

kick - 0.0696

jump - 0.0543

shoot_ball - 0.0537

handstand - 0.0528

hit - 0.0386

kick_ball - 0.0378

(second input)

Latent Semantic 1:

golf - 1.1628

catch - 0.8994

cartwheel - 0.5997

somersault - 0.5343

handstand - 0.4513

climb - 0.3502

shoot_ball - 0.2958

shoot_bow - 0.2935

situp - 0.2563

brush_hair - 0.0917

kick_ball - -0.0498

chew - -0.1361

climb_stairs - -0.1690

hug - -0.2742

hit - -0.2973

laugh - -0.3068

jump - -0.3413

smile - -0.3788

shoot_gun - -0.4031

clap - -0.4280

smoke - -0.4821

kick - -0.5235

sit - -0.5675

stand - -0.5775

Latent Semantic 2:

climb_stairs - 0.2991

climb - 0.2779

shoot_ball - 0.2736

situp - 0.2556

handstand - 0.2463

chew - 0.2276

laugh - 0.2216

shoot_bow - 0.2147

somersault - 0.1932

hit - 0.1888

brush_hair - 0.1615

hug - 0.1437

smile - 0.1078

cartwheel - 0.0403

shoot_gun - -0.0175

smoke - -0.0263

clap - -0.0378

kick_ball - -0.0568

jump - -0.1978

kick - -0.4052

catch - -0.4104

sit - -0.4130

stand - -0.5969

golf - -0.6901

Latent Semantic 3:

kick_ball - 0.2767

jump - 0.1831

cartwheel - 0.1363

catch - 0.0872

clap - 0.0863

brush_hair - 0.0841

laugh - 0.0745

shoot_gun - 0.0728

hit - 0.0482

smile - 0.0416

handstand - 0.0239

smoke - 0.0182

climb_stairs - -0.0072

chew - -0.0285

golf - -0.0287

shoot_bow - -0.0476

kick - -0.0592

climb - -0.0943

somersault - -0.0973

sit - -0.1124

hug - -0.1295

shoot_ball - -0.1541

situp - -0.1774

stand - -0.1966

(Third Input)

Latent Semantic 1:

cartwheel - 0.0271

brush_hair - 0.0260

somersault - 0.0260

situp - 0.0259

climb - 0.0256

kick_ball - 0.0255

clap - 0.0255

jump - 0.0255

climb_stairs - 0.0255

chew - 0.0254

smile - 0.0254

catch - 0.0251

shoot_ball - 0.0251

golf - 0.0249

hit - 0.0249

shoot_gun - 0.0249

smoke - 0.0248

laugh - 0.0248

shoot_bow - 0.0246

kick - 0.0246

hug - 0.0245

stand - 0.0244

sit - 0.0243

handstand - 0.0242

Latent Semantic 2:

handstand - 0.9758

sit - 0.9757

stand - 0.9756

hug - 0.9755

kick - 0.9754

shoot_bow - 0.9754

laugh - 0.9752

smoke - 0.9752

shoot_gun - 0.9751

hit - 0.9751

golf - 0.9751

shoot_ball - 0.9749

catch - 0.9749

smile - 0.9746

chew - 0.9746

climb_stairs - 0.9745

jump - 0.9745

clap - 0.9745

kick_ball - 0.9745

climb - 0.9744

situp - 0.9741

somersault - 0.9740

brush_hair - 0.9740

cartwheel - 0.9729