CSE 515 Phase 3 Report

Group Members:
Hong Ai: hongai1@asu.edu
Xinyi Liu: xliu335@asu.edu
Dunchuan Wu: dunchuan@asu.edu
Wei Xin: wxin4@asu.edu
Wenzhe Zheng: wzheng41@asu.edu
Zian Zhang: zzhan329@asu.edu

Abstract

Data has been kept increasing at an extremely high rate for over a decade and image data plays a very important role in it. Huge number of images are in different databases and we need to index and sort them in order to reduce the calculation time when we need to recommend images to users for instance. Therefore, our goal of this project is to index, classify and sort all the given images of gestures. We finally utilize the results from the above and use different relevance feedback techniques, for instance, Personalized Pangrank (PPR), to give users the most relevant gestures.

Keywords

Image of gestures, Indexing, Sorting, Similarity, Euclidean Distance, Labelling, Classification, Decision Tree, Personalized Pagerank, Relevance Feedback, LSH, KNN.

Introduction

In this project, we deal with a relatively huge amount of gesture images that are formed in multi-dimensional vector space. As the number of images (and dimensions) increases, it becomes harder and harder to search for similar images. Therefore, we use indexing techniques to filter out some irrelevant images and help speed up searching. In particular, we utilize Locality Sensitive Hashing that cluster gestures in approximate groups. Furthermore, our group also uses classification processes including *k*-nearest neighbors(KNN), Personalized Pagerank(PPR) classifier, decision-tree that help classify different gestures to some meaningful groups. Moreover, this phase also allows users to give some feedback to the result of query found in LSH, and two techniques, which are probabilistic relevant feedback and PPR relevant feedback, will be used in order to improve nearest neighbor matches returned from the query.Based on the given dataset for presenting our phase, we have 31 gestures for 3 classes of gestures so it is a total of 93 gestures per component. The dataset has added 10 more gestures based on each gesture and these added gestures are the ones that have noise, in other words, they are the "Slightly Modified" gestures of each original gesture.

Problem Solution

Task 1

In the first task, we utilize the gesture-gesture similarity matrix based on the previous phase. The gesture-gesture similarity matrix is a 93 * 93 dimensional matrix which has already been sorted based on the and the user enters the value k, which prunes all other columns and only saves the first k columns. Here we notice that we have k outgoing edges so our gesture-gesture similarity graph is a directed graph. Now we normalize the matrix by row. For example, if gesture 1 has top 5 similar gestures which are 2, 3, 4, 5, and 6, and we know that here k is 5, so the values should be 1 / k which is 0.2. Therefore we set this specific row as [0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, … , 0]. We do all the 93 gestures and transpose the matrix to do the dot product later. Next, we utilize n gestures from the above 93 gestures where these gestures are picked by the user. After picking the n gestures, we will use these n gestures as our seed nodes in task 1 and form a 1 * 93 dimensional matrix and set the specified gestures as (1 / n) and all other gestures as 0.

After forming the original gesture-gesture matrix and the start vector, we can start the random walk. The usual random walk wouldn't go just one step but a couple of steps. Hence, the probability that you would end up on a node from another node 1 a few steps is the combination of these probabilities. The vq is the query vector which tells us the probability of the current nodes. As we start at the user-defined n nodes, the probability of being at that node is 100%. The dot multiplication is taking this probability and performs a vector multiplication with the probabilities of the next possible node. Hence, we get a vector which tells us the probabilities where we could end up based on the probabilities where we have been before. For the first step, this is simply the first column of the normalized gesture-gesture matrix. Next, a walker has a probability of c (beta) to go on without restarting and therefore (1 - c) to teleport to its starting point again and walk from there. This helps explore more diverse paths. Instead of just multiplying the probabilities where we could be right now with the normalized gesture-gesture matrix, we include the probability that we end up on the starting node.

After all the above implementations, we get a 1 * 93 matrix and sort from biggest probability to the smallest probability. We pick the top m gestures as dominant gestures and output them as a vector. There is a folder named “output_graphs” which plots the dominant gestures’ sensors based on time series.

Task 2

KNN:

K-nearest neighbour algorithms will be used in this classification. The main goal of KNN is that for each unlabeled object, we need to find k nearest labeled objects based on standard Eculidean distance which is shown below:

$$dist(A,B) = \sqrt{\sum_{i=1}^{d} (A_i - B_i)^2} \text{ where}$$

A & B: vector representation for two objects in our database

d: dimension in our vector space

General Approach:

1. For each unlabeled object, we find out a Eculidean distance vector which contains all Eculidean distances from this object to other objects in our vector space.
2. Sort the Eculidean distance vector in ascending order, the shortest distance implies the most similar gesture in our database, the largest distance implies the least similar gesture in our database.
3. Based on the sorted Eculidean distance vector, we take top-k labeled gestures, and label this gesture based on the class that has the highest frequency in top-k labeled gestures. (eg. class 1: 1; class 2: 1; class 3: k-2; the label for this gesture is class 3 as long as k larger than 3, otherwise, will the label for this gesture will be randomly selected from the classes that has same and highest frequency)

PPR:

Our project proposes a way for personalized pagerank(PPR) classifiers to classify all unlabeled gestures in our database. The main idea of PPR is described above. We will utilize unlabeled gestures as the only seed node for performing random walk in PPR algorithms. After convergence of PPR, we will get a list of probabilities which indicate the likelihood of relationship between seed node and corresponding nodes and a list of corresponding gestures. Then we will select top-m labeled gestures, and label the gesture based on the class that has the highest frequency in top-m labeled gestures.

General Approach:

1. For each unlabeled object, we treat it as the only one seed node in our graph that is built based on task1. The seed node will be used in our PPR algorithm and we can get a rank/probablity associate for each other nodes in our database.
2. Sort the probability found in step 1 in descending order.
3. We select top-m dominant labeled gestures found in step 2, and label this gesture based on the class that has the highest frequency in top-k labeled gestures.

Decision Tree:
Raw data extracted from vector models was splitted to training dataset and testing dataset based on labelled and unlabelled gestures. Features in our datasets are words. Features with labelled gestures are used in training, and features with unlabelled gestures are used in testing. We reformatted the training dataset matrix by adding the corresponding labels on the last column. We used CART (Classification and Regression Trees) as a decision tree learning algorithm, which uses gini impurities to find best splits of the decision tree. The gini index calculation formula is shown below:

$$Gini = 1 - \sum_j p_j^2$$

For each feature, we take the value and measure Gini impurity and information gain. Gini impurity represents the uncertainty of a node, and information gain represents how a split will reduce the uncertainty. We always select the highest information gain for finding best splits. Since we won't create a node if the information gain is 0, in order to create as many pure nodes as possible and ignore no information gained nodes, we set up that information gain must be larger than or equal to 0.0000001. If the condition is not satisfied, the algorithm will return the best fitted label and not continue to recursively build the decision tree. We ran the training dataset first to get a decision tree, and used the training decision tree to fit the testing dataset. Predicted labels for testing the dataset will be printed in the console. The accuracy is acceptable after testing.

Task 3

Locality Sensitive Hashing is creating buckets and putting similar gesture images into the same bucket to reduce the dimension of the gesture matrix. The main idea of this task is to reduce the false positives and misses. We have the user define the number of layers and number of hashes per layer and the increasing of layers will significantly reduce the false positives but increase the misses for a little bit because we the reason we say "layers" is because we put each hashed table over the previous one to do conjunction. In order to decrease the misses, we need to do the other way around which is union. This way will increase the false positive so we need to find a balance between conjunction and union.
We take 2 directions to cluster all these gestures. For each direction, we project all points onto the line and for each region, we have several buckets that hold the projected points. If each two

points are in the same buckets and the distance between two points are within a certain threshold, we consider them to be matched. Besides, we want to check if they are in the same hash family. If the matched percentage is less than 60% or the missed percentage is greater than 40%, we consider they are not in the same hash family. The hash family used in our project is shown below:

$$dist(A,B) > \frac{4 * max(distcance\ for\ dataset)}{5}, then\ pr(H(A) = H(B)) <= 0.3$$

$$dist(A,B) < \frac{max(distcance\ for\ dataset)}{2}, then\ pr(H(A) = H(B)) >= 0.7$$

After getting the hash values of each gesture image, we will have a hash table for each layer by concatenating each hash function in the same layer. Then, for each layer, we extracted all gestures that hash into the same hash value as the same as the given gesture. Also, we count the duplication of gestures that hash into the same hash value in different layers. The higher frequency of duplication of gestures implies more similar gestures. Finally, we will count how many unique gestures that have the same hash value in each layer, and output this number.
General Approach:

1. Randomly generate a direction in our original vector space, and check whether this direction satisfies our hash function family or not.
2. Once the direction generated by step 1 is satisfied in our hash function family, we will project all data onto this vector by using dot product.
3. After projection, there will be a range in the new vector from xmin to xmax. Based on the value of this range, we will partition the range into two, one from xmin to (xmin + xmin)/2 and the other range from (xmin + xmin)/2 to xmax.
4. Hash corresponding data that project onto the first range to 0 and hash corresponding data that projected on to the second range to 1
5. Repeat the same procedure for each hash function in the same layer.
6. Concatenate all hash values in the same layer and create a corresponding hash table for each gesture under the same layer.
7. Repeat the same procedure for each layer until the number of layers matches the input from the user.
8. After generating a different hash table for each layer, we need to extract the gestures that have the same hash value as the given gesture in each layer.
9. Sort the result based on the number of same hash values that are the same as the given gesture.
10. Take top-t from step 9 as our result of task 3.

Task 4
Given the data matrix returned by task3, we have the result in terms of gestures that are nearest neighbour matches. Therefore, we consider to improve the result by applying the relevance feedback method to improve the information retrieval result. According to the topic, we will let

users choose which mode (- the relative importances of sensors or -the relative importances of the X, Y, Z, and W components). Next, we then use the corresponding data structure to calculate a new similarity matrix according to the mode selected by the user, and (using the euclidean distance algorithm) to find similar gesture files. Next step is to let a user input a value for any existing gestures after the user reads the original data matrix that contains the gestures. As for the available range of parameters to contribute to make the functions work, to be specific, the input parameter takes a value of 1and 0, indicating that the user assigns a label of “relevant” and “irrelevant(non-relevant)”. If the user chooses 1, then we will not change the similarity value. If the user chooses 0, then we will change the corresponding similarity value to 0. After this step, the data matrix can generate a new list that re-orders the old returned gesture result according to the label assigned by users. It is intuitive that we show them to users in a rank from highest relevant (as rank them in the first item of the result list) as these high-ranked gestures are the most concerned and useful.

Task 5

In this task, we will use PPR based relevant feedback to improve our initial query and return a new ranked and revising query as our result. It takes output from task 3 and a PPR graph constructed from task 1 as inputs. Also, the PPR based relevant feedback allows user labels some result from task 3 as relevant or irrelevant gesture. Then we will use relevant gestures as seed nodes to find a probability associated with relevant gestures to other gestures. Also, we will use irrelevant gestures as seed nodes to find a probability associated with irrelevant gestures to other gestures. After computation, we will get two sets of probability, one set of probability is associated with relevant gestures, and the other set of probability is associated with irrelevant gestures. For each gesture, we will compare the probability associated with the relevant gestures and the probability associated with the irrelevant gestures. If the probability associated with the relevant gestures is higher than the probability associated with the irrelevant gestures, we can conclude that this gesture is relevant. Finally, we will revise the query as our result by sorting all relevant gestures with higher probability in descending order. Also, the systems will allow users to enter “positive” which implies the query is a good result or “negative” which implies the query should be improved again. If the user enters “negative”, the system will re-execute the whole task and task the previous query as input in new execution.

General Approach:

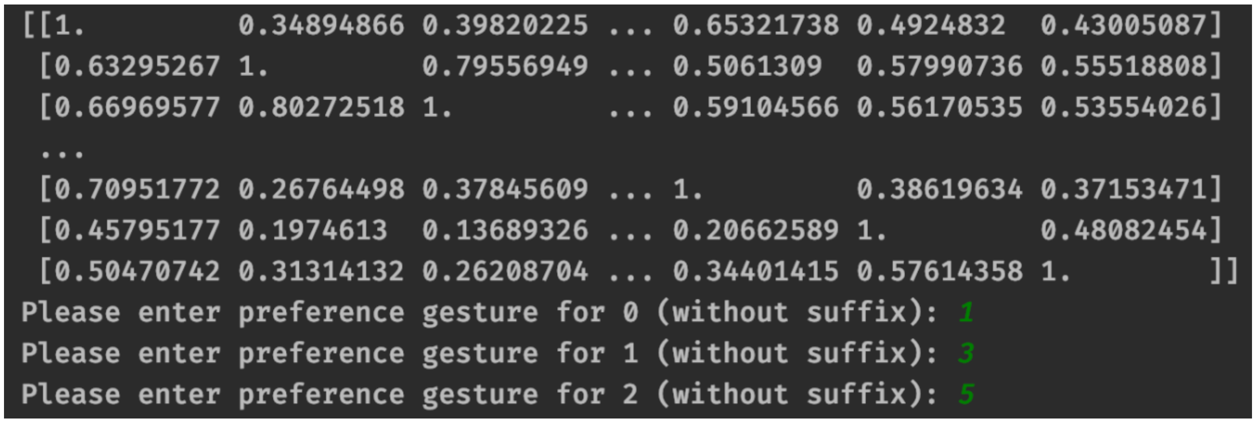
1. Allow user labels relevant or irrelevant to some of the gestures found in our search result.
2. For each relevant gesture, we will treat it as a seed node and assign value 1 to its corresponding random walk vector, and assign value 0 to all irrelevant gestures, and assign 0.5 to other unlabeled gestures. After assignment of the “re-start” vector, we need to normalize in order the sum of all values in the “re-start” vector to be 1.
3. Perform PPR with the initialization mentioned above for relevant gestures.
4. Perform PPR with the initialization mentioned above for irrelevant gestures.

5. For each gesture, compare the probability associated with the relevant gestures and the probability associated with the irrelevant gestures.
6. Sort the new query from highest to lowest.

Task 6
After performing task 4 or task 5, the user is able to provide feedback on the query result printed from task 4 and task 5. If the result is satisfied, then task 6 terminates. If the result is not satisfied, the user will be able to choose to revisit task 4 or task 5 again to enter new query parameters and get new results. Task 6 will be terminated until the user gives positive feedback. The result will always be printed in decreasing order.

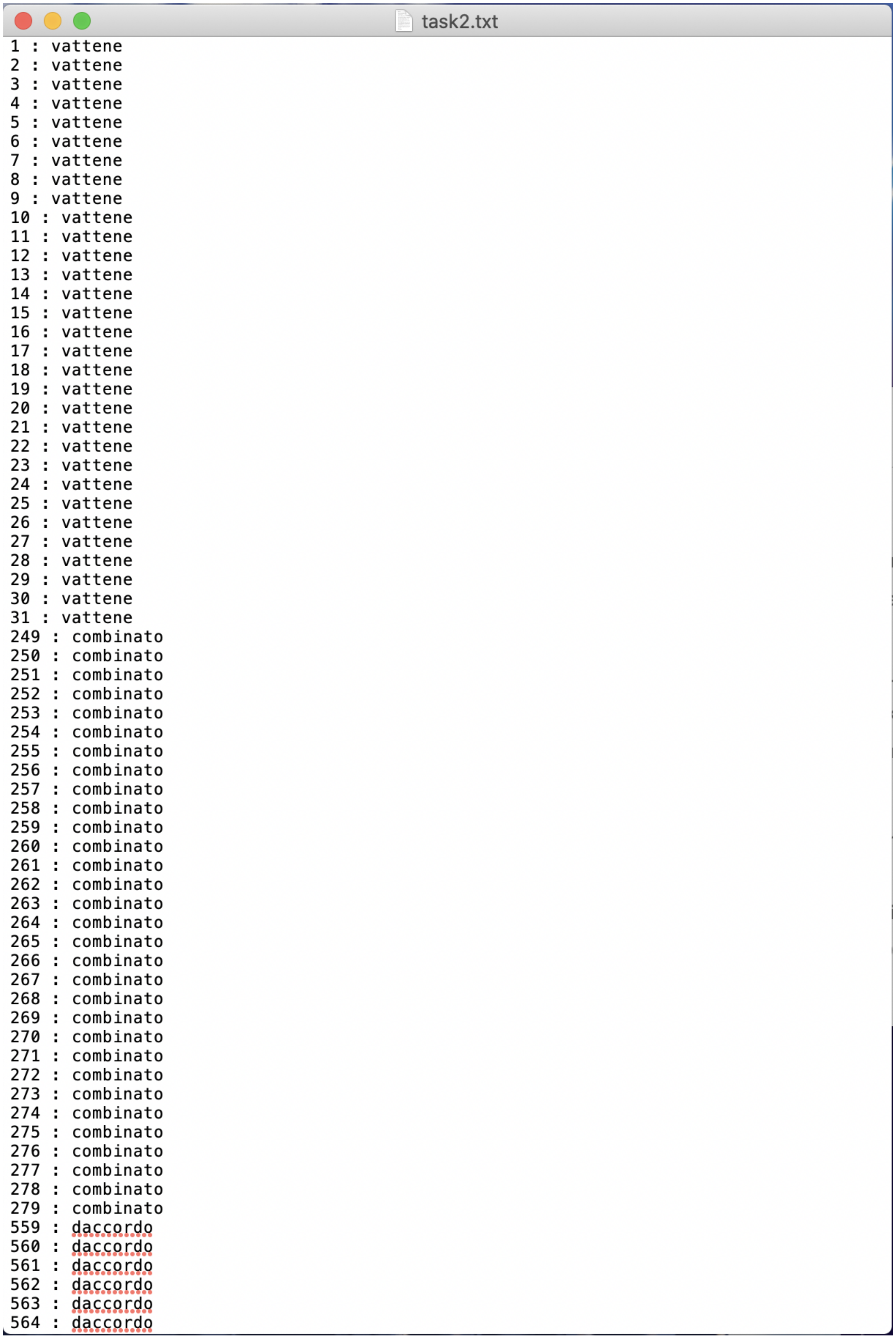
Interface Specification

Task1 Output:



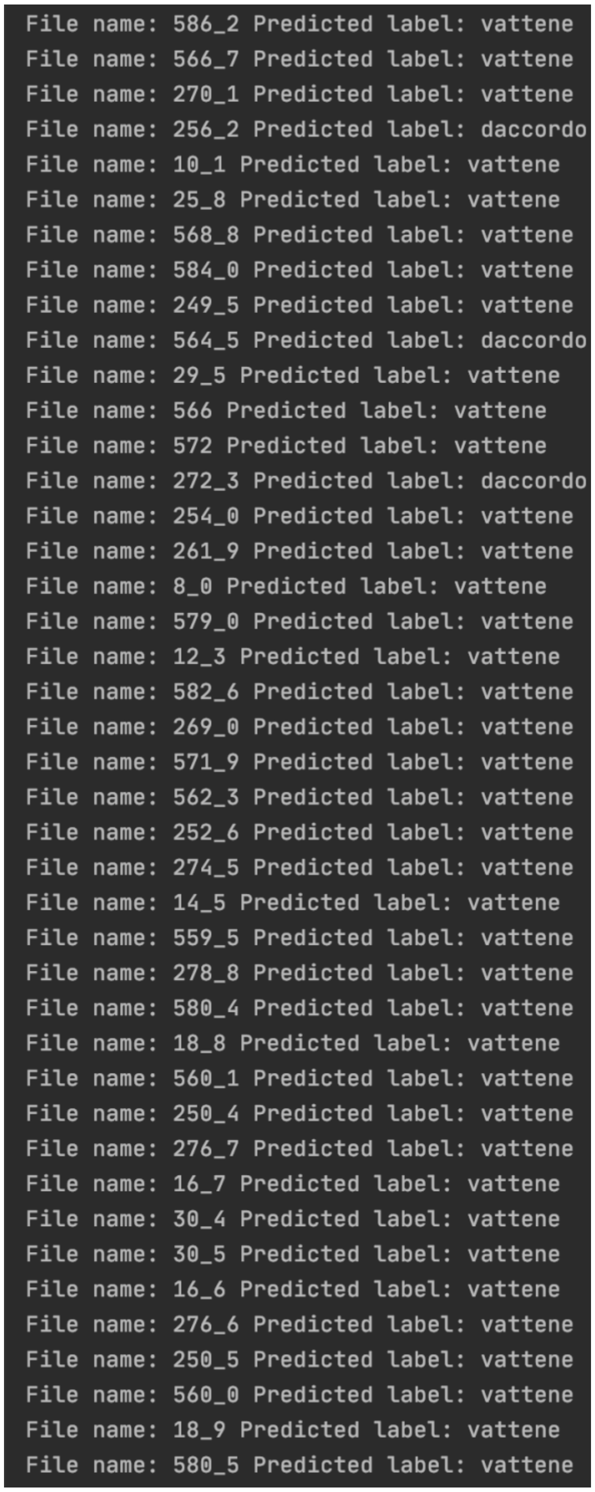
```
[[1.         0.34894866 0.39820225 ... 0.65321738 0.4924832  0.43005087]
 [0.63295267 1.         0.79556949 ... 0.5061309  0.57990736 0.55518808]
 [0.66969577 0.80272518 1.         ... 0.59104566 0.56170535 0.53554026]
 ...
 [0.70951772 0.26764498 0.37845609 ... 1.         0.38619634 0.37153471]
 [0.45795177 0.1974613  0.13689326 ... 0.20662589 1.         0.48082454]
 [0.50470742 0.31314132 0.26208704 ... 0.34401415 0.57614358 1.        ]]
Please enter preference gesture for 0 (without suffix): 1
Please enter preference gesture for 1 (without suffix): 3
Please enter preference gesture for 2 (without suffix): 5
```

Task2 Output:
KNN:


task2.txt
```
1 : vattene
2 : vattene
3 : vattene
4 : vattene
5 : vattene
6 : vattene
7 : vattene
8 : vattene
9 : vattene
10 : vattene
11 : vattene
12 : vattene
13 : vattene
14 : vattene
15 : vattene
16 : vattene
17 : vattene
18 : vattene
19 : vattene
20 : vattene
21 : vattene
22 : vattene
23 : vattene
24 : vattene
25 : vattene
26 : vattene
27 : vattene
28 : vattene
29 : vattene
30 : vattene
31 : vattene
249 : combinato
250 : combinato
251 : combinato
252 : combinato
253 : combinato
254 : combinato
255 : combinato
256 : combinato
257 : combinato
258 : combinato
259 : combinato
260 : combinato
261 : combinato
262 : combinato
263 : combinato
264 : combinato
265 : combinato
266 : combinato
267 : combinato
268 : combinato
269 : combinato
270 : combinato
271 : combinato
272 : combinato
273 : combinato
274 : combinato
275 : combinato
276 : combinato
277 : combinato
278 : combinato
279 : combinato
559 : daccordo
560 : daccordo
561 : daccordo
562 : daccordo
563 : daccordo
564 : daccordo
```

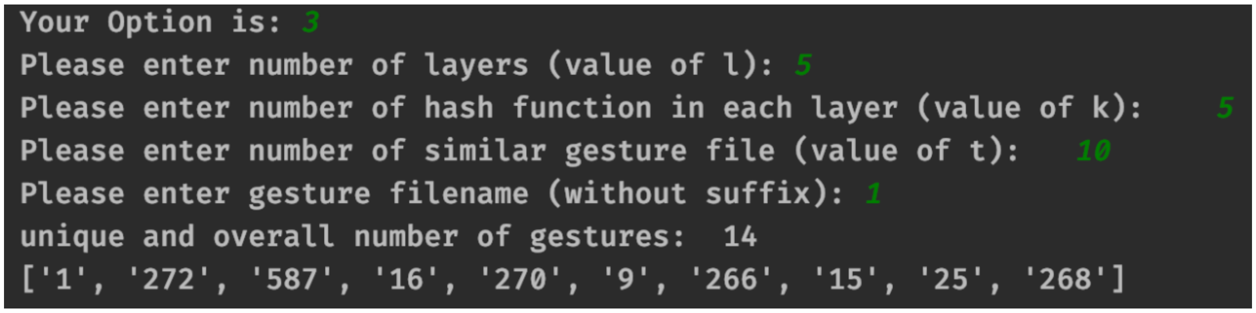
PPR:

Decision Tree:



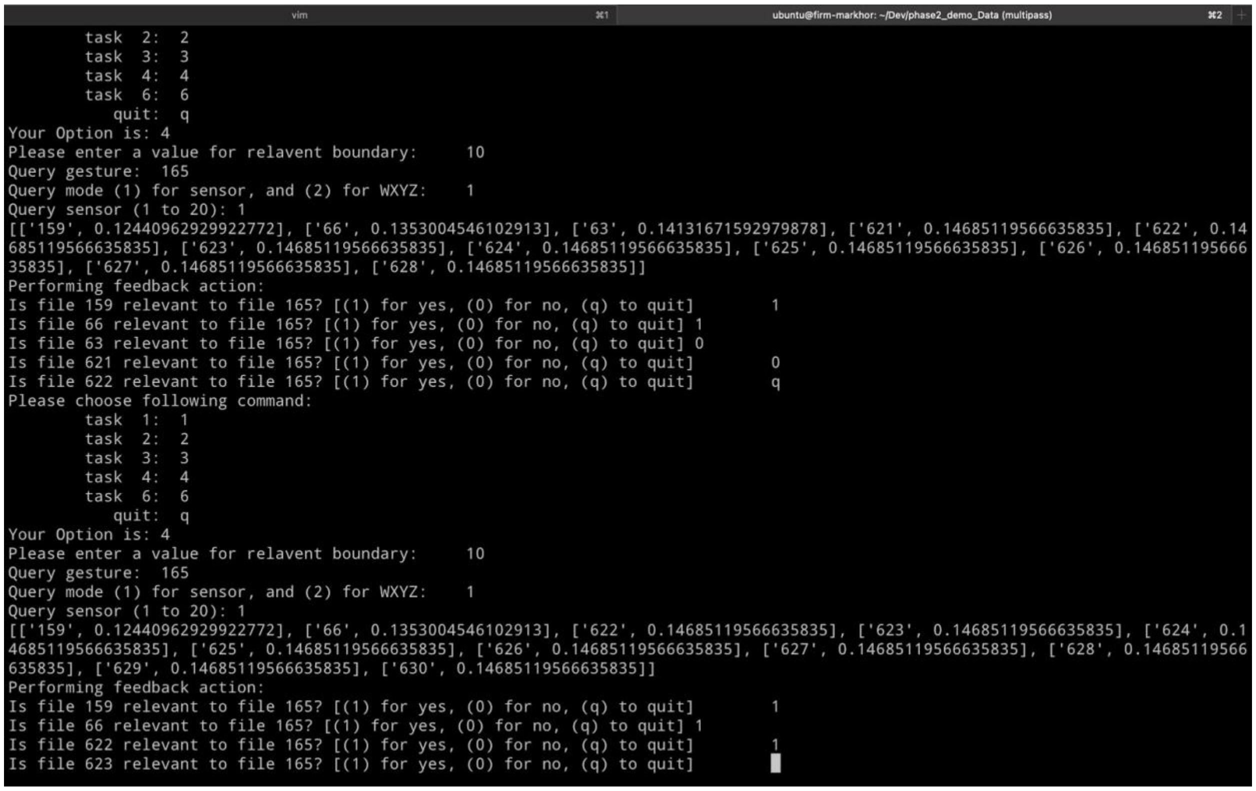
```
File name: 586_2 Predicted label: vattene
File name: 566_7 Predicted label: vattene
File name: 270_1 Predicted label: vattene
File name: 256_2 Predicted label: daccordo
File name: 10_1 Predicted label: vattene
File name: 25_8 Predicted label: vattene
File name: 568_8 Predicted label: vattene
File name: 584_0 Predicted label: vattene
File name: 249_5 Predicted label: vattene
File name: 564_5 Predicted label: daccordo
File name: 29_5 Predicted label: vattene
File name: 566 Predicted label: vattene
File name: 572 Predicted label: vattene
File name: 272_3 Predicted label: daccordo
File name: 254_0 Predicted label: vattene
File name: 261_9 Predicted label: vattene
File name: 8_0 Predicted label: vattene
File name: 579_0 Predicted label: vattene
File name: 12_3 Predicted label: vattene
File name: 582_6 Predicted label: vattene
File name: 269_0 Predicted label: vattene
File name: 571_9 Predicted label: vattene
File name: 562_3 Predicted label: vattene
File name: 252_6 Predicted label: vattene
File name: 274_5 Predicted label: vattene
File name: 14_5 Predicted label: vattene
File name: 559_5 Predicted label: vattene
File name: 278_8 Predicted label: vattene
File name: 580_4 Predicted label: vattene
File name: 18_8 Predicted label: vattene
File name: 560_1 Predicted label: vattene
File name: 250_4 Predicted label: vattene
File name: 276_7 Predicted label: vattene
File name: 16_7 Predicted label: vattene
File name: 30_4 Predicted label: vattene
File name: 30_5 Predicted label: vattene
File name: 16_6 Predicted label: vattene
File name: 276_6 Predicted label: vattene
File name: 250_5 Predicted label: vattene
File name: 560_0 Predicted label: vattene
File name: 18_9 Predicted label: vattene
File name: 580_5 Predicted label: vattene
```

Task3 Output:



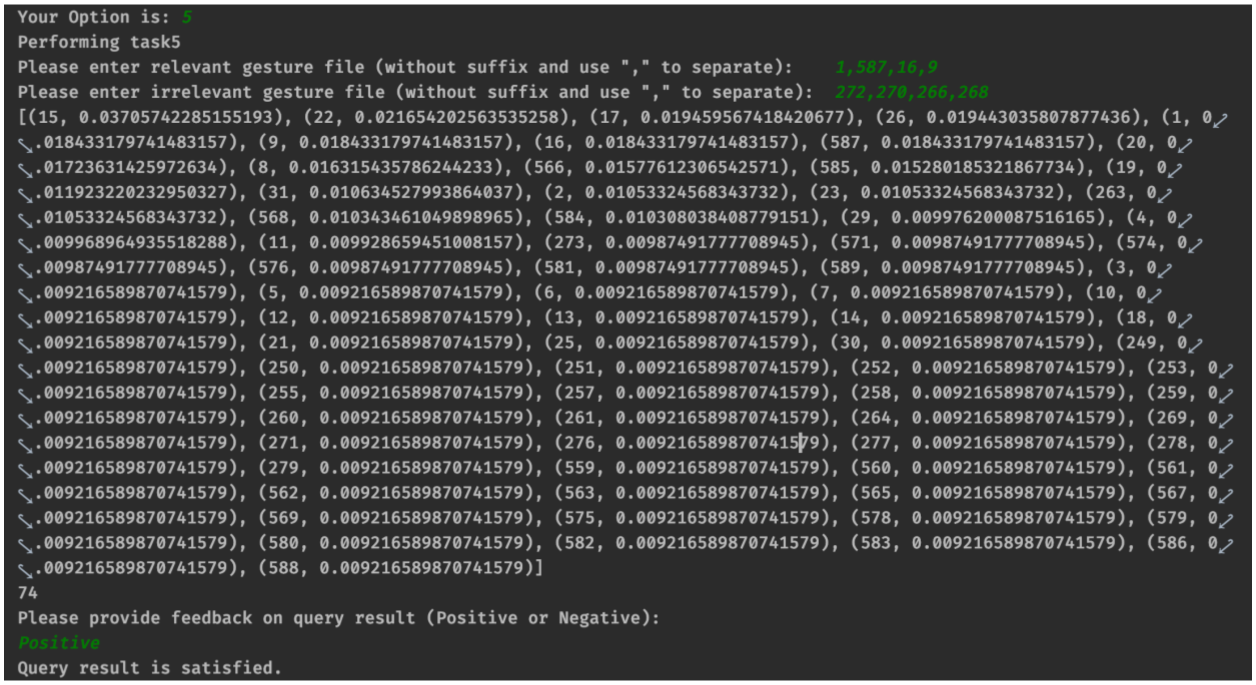
```
Your Option is: 3
Please enter number of layers (value of l): 5
Please enter number of hash function in each layer (value of k):    5
Please enter number of similar gesture file (value of t):   10
Please enter gesture filename (without suffix): 1
unique and overall number of gestures:  14
['1', '272', '587', '16', '270', '9', '266', '15', '25', '268']
```

Task4 Output:



```
        task  2:  2
        task  3:  3
        task  4:  4
        task  6:  6
           quit:  q
Your Option is: 4
Please enter a value for relavent boundary:     10
Query gesture:  165
Query mode (1) for sensor, and (2) for WXYZ:    1
Query sensor (1 to 20): 1
[['159', 0.12440962929922772], ['66', 0.1353004546102913], ['63', 0.14131671592979878], ['621', 0.14685119566635835], ['622', 0.14
685119566635835], ['623', 0.14685119566635835], ['624', 0.14685119566635835], ['625', 0.14685119566635835], ['626', 0.146851195666
35835], ['627', 0.14685119566635835], ['628', 0.14685119566635835]]
Performing feedback action:
Is file 159 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        1
Is file 66 relevant to file 165? [(1) for yes, (0) for no, (q) to quit] 1
Is file 63 relevant to file 165? [(1) for yes, (0) for no, (q) to quit] 0
Is file 621 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        0
Is file 622 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        q
Please choose following command:
        task  1:  1
        task  2:  2
        task  3:  3
        task  4:  4
        task  6:  6
           quit:  q
Your Option is: 4
Please enter a value for relavent boundary:     10
Query gesture:  165
Query mode (1) for sensor, and (2) for WXYZ:    1
Query sensor (1 to 20): 1
[['159', 0.12440962929922772], ['66', 0.1353004546102913], ['622', 0.14685119566635835], ['623', 0.14685119566635835], ['624', 0.1
4685119566635835], ['625', 0.14685119566635835], ['626', 0.14685119566635835], ['627', 0.14685119566635835], ['628', 0.14685119566
635835], ['629', 0.14685119566635835], ['630', 0.14685119566635835]]
Performing feedback action:
Is file 159 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        1
Is file 66 relevant to file 165? [(1) for yes, (0) for no, (q) to quit] 1
Is file 622 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        1
Is file 623 relevant to file 165? [(1) for yes, (0) for no, (q) to quit]        █
```

Task5 Output:



```
Your Option is: 5
Performing task5
Please enter relevant gesture file (without suffix and use "," to separate):    1,587,16,9
Please enter irrelevant gesture file (without suffix and use "," to separate):  272,270,266,268
[(15, 0.03705742285155193), (22, 0.021654202563535258), (17, 0.019459567418420677), (26, 0.019443035807877436), (1, 0↙
↘.018433179741483157), (9, 0.018433179741483157), (16, 0.018433179741483157), (587, 0.018433179741483157), (20, 0↙
↘.01723631425972634), (8, 0.016315435786244233), (566, 0.01577612306542571), (585, 0.015280185321867734), (19, 0↙
↘.011923220232950327), (31, 0.010634527993864037), (2, 0.01053324568343732), (23, 0.01053324568343732), (263, 0↙
↘.01053324568343732), (568, 0.010343461049898965), (584, 0.010308038408779151), (29, 0.009976200087516165), (4, 0↙
↘.009968964935518288), (11, 0.009928659451008157), (273, 0.00987491777708945), (571, 0.00987491777708945), (574, 0↙
↘.00987491777708945), (576, 0.00987491777708945), (581, 0.00987491777708945), (589, 0.00987491777708945), (3, 0↙
↘.009216589870741579), (5, 0.009216589870741579), (6, 0.009216589870741579), (7, 0.009216589870741579), (10, 0↙
↘.009216589870741579), (12, 0.009216589870741579), (13, 0.009216589870741579), (14, 0.009216589870741579), (18, 0↙
↘.009216589870741579), (21, 0.009216589870741579), (25, 0.009216589870741579), (30, 0.009216589870741579), (249, 0↙
↘.009216589870741579), (250, 0.009216589870741579), (251, 0.009216589870741579), (252, 0.009216589870741579), (253, 0↙
↘.009216589870741579), (255, 0.009216589870741579), (257, 0.009216589870741579), (258, 0.009216589870741579), (259, 0↙
↘.009216589870741579), (260, 0.009216589870741579), (261, 0.009216589870741579), (264, 0.009216589870741579), (269, 0↙
↘.009216589870741579), (271, 0.009216589870741579), (276, 0.009216589870741579), (277, 0.009216589870741579), (278, 0↙
↘.009216589870741579), (279, 0.009216589870741579), (559, 0.009216589870741579), (560, 0.009216589870741579), (561, 0↙
↘.009216589870741579), (562, 0.009216589870741579), (563, 0.009216589870741579), (565, 0.009216589870741579), (567, 0↙
↘.009216589870741579), (569, 0.009216589870741579), (575, 0.009216589870741579), (578, 0.009216589870741579), (579, 0↙
↘.009216589870741579), (580, 0.009216589870741579), (582, 0.009216589870741579), (583, 0.009216589870741579), (586, 0↙
↘.009216589870741579), (588, 0.009216589870741579)]
74
Please provide feedback on query result (Positive or Negative):
Positive
Query result is satisfied.
```

System requirements/installation and execution instructions

There are two source scripts written in python. Please use python3.x to compile source scripts. Put two source scripts under same execution folder, and execute the following command:

```
python3 phase3.py
```

After entering command, the system will provide a suitable execution interface which allows the user to give a proper input for each task.

Each task is separated into different functions under same scripts, however, some task has dependency (eg. PPR classifier and task 5 are depended on task1, task 5 and task4 are depended on task 3). Please make sure to execute previous tasks before performing other tasks, which can prevent program crashes.

Note:

All input should be exactly matched with specification in order to prevent preventable crashes including type conversion or file not found.

This phase of the project utilizes dynamic time wrapping(DTW) and uses original data (No dimensional reduction) for all tasks in order to preserve data best. Therefore, the execution time will be increased based on the number of data.

System requirements:

- Python 3.7
- collections
- re
- secrets
- shutil
- pandas 1.1.3

- numpy 1.19.2
- os
- csv
- scipy 1.5.2
- quad
- xlrd 1.2.0
- Matplotlib 3.3.2

Execution instructions:

Run the program by the command line *python3 phase3.py*. Or create a new project in Pycharm or other Python IDEs to run the program file. Follow the interface to enter inputs and execute the program. Be sure to import *phase3_decisiontree.py* in *phase3.py* to run the decision tree algorithm.

Conclusion

All 6 tasks have been implemented successfully and have followed and met all requirements specified in the instruction. All tasks can be run without errors or crashes. This project has helped us to take deep insight into Personalized Page Rank, KNN classification, Decision Tree, Locality Sensitive Hashing and probabilistics relevance feedback. We have analyzed huge data and extracted features to form a proper matrix for tasks. We have experimented with indexing and classification problems while facing challenges.

Bibliography

- *Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback Journal of the American Society for Information Science. 41, pp. 288-297, 1990.*
- *Candan, K. Selçuk, and Maria Luisa. Sapino. Data Management for Multimedia Retrieval. Cambridge University Press, 2010.*
- *Candan, K. Selçuk. CSE515 Multimedia and Web Databases phase 3. Sep 2020.*

Appendix

Hong Ai	Task 4
Xinyi Liu	Task 2 Decision Tree, Task 6
Dunchuan Wu	Task 4
Wei Xin	Task 1, Task 3
Wenzhe Zheng	Task3, Task 5

Zian Zhang	KNN, PPR classifier, Task 3, Task5