DeepFake Detection Program v2

Hypothesis: People have unique combinations of facial features that will typically move together. For example, Person A's eyes lift up when they speak, and Person B wrinkles their nose when they blink. These combinations of two can be used to create probability maps for each person, and can be used to determine if two profiles share similar pairing probabilities.

Loading Data into Program

1. Have "raw" CSV files saved into local directory (by "raw" I mean straight from OpenFace)
2. In program, declare a List of input files called "rawFiles" and a List of output files called "processedFiles"
3. Load the raw CSV files into "rawFiles"

Processing Raw CSV Files

1. Have a function capable of stripping the CSV file into the important data we need
   - delete unconfident/unsuccessful rows
     - iterate through rows and look at success column (column index 4, contains boolean values)
     - keep rows with true, delete rows with false
   - delete all columns, except save frame and AU_c values
     - in the far right columns of the CSV file, there are AU values, which are the broad action unit values determined from the hundreds of raw point values in earlier columns. We will only use AU_c values, which range from AU01_c to AU_45_c. These are boolean values and will be helpful for comparisons. Be aware that there are less than 45 values, some numbers are skipped.
2. Send raw CSV files into the function and store processed files in "processedFiles"

Objects/Classes Needed (so far)

<u>public class ActionUnit:</u>
int id; // number given to specific action unit (ex. AU01_c's id is 0, and AU45_c's id is 17, because there are skipped numbers and only 18 AUs total)
bool* activeVals; // ptr to array of booleans containing this Action Unit's values. Length of array is the number of frames.
int numFrames; // total number of frames used from the video. Used to decide the length of activeVals

ActionUnit(); // default constructor: assigns id = -1, numFrames = 0, and initializes activeVals to an array of length 0 (unused)
ActionUnit(int id, int numFrames); // overloaded constructor: assigns id and numFrames, and initializes activeVals to an empty array of length numFrames
~ActionUnit(); // destructor: delete activeVals

void setActiveVals(bool* vals); // assigns activeVals

<u>public class Profile:</u>
string name; // name of the person to whom the profile belongs
float** probMatrix; // 2D ptr array where the the pair-probability values are stored (used for current file upload)
float** avgMatrix; // 2D ptr array where the average of all pair-probability values are stored (used for displaying and can be added to by the probMatrix to make new average)
static const int numActionUnits = 18; // number of action units used, which is 18 currently (if a static const int is not possible, delete static and keep const)
int currNumFrames; // how many frames are in the current video used by probMatrix (used for calculating average)
int totNumFrames; // stat of how many total frames used in the entire Profile (used for calculating average and displaying to user)
int totNumVideos; // stat of how many total videos have been used in the entire Profile (only for simply displaying to the user)

Profile(); // default constructor: name = "", probMatrix is empty 2D ptr array, totNumFrames & totNumVideos = 0 (unused)
Profile(string name); // overloaded constructor: same as default but assigns the name (used when creating a new profile)
Profile(string saveData); // overloaded constructor: string of Profile info loaded from save file
~Profile(); // destructor: delete probMatrix and avgMatrix

void calcProbMatrix(ActionUnit** au); // takes in ptr to array of ptrs to action units, fills probMatrix with float values determined from action units (####more detailed description below#####)
void updateAvgMatrix(); // updates avgMatrix with new probMatrix values (calculation: avgMatrix[i][j] =(totNumFrames * avgMatrix[i][j] + probMatrix[i][j]) / (totNumFrames + currNumFrames);

Main Computations

1. Read/Write Files (explained above)
2. Declare Profile Object
    - declared by either creating new one or loading existing data from

    database
3. Declare ActionUnit array
    - Have length be equal to numFrames
    - Initialize using a for loop, iterating numFrames times and creating new ActionUnits, constructing as such: ActionUnit(i, numFrames)
    - load data from the processed CSV file into all the ActionUnits
4. Calculate Probability Matrix
    - call method in the Profile
5. Display new matrix, ask if it should be added to the Main Matrix (avgMatrix)
    - If yes:
        - call updateAvgMatrix()
        - do totNumFrames += currNumFrames
        - do totNumVideos++
    - If no:
        - quit? discard current values? not sure, we will figure this out later

####
CalcProbMatrix() Further Description
1. Iterate over the matrix, but only on the upper triangular to avoid duplication (all values below and including the diagonal are unused)
2. Iterate for amount of frames and store the probability of how likely both ActionUnits will be on and both be off at the same time (XNOR comparison)
3. store each calculated probability value in its corresponding place in the matrix