Documentation for Alpha Version of Face Scan

Introduction

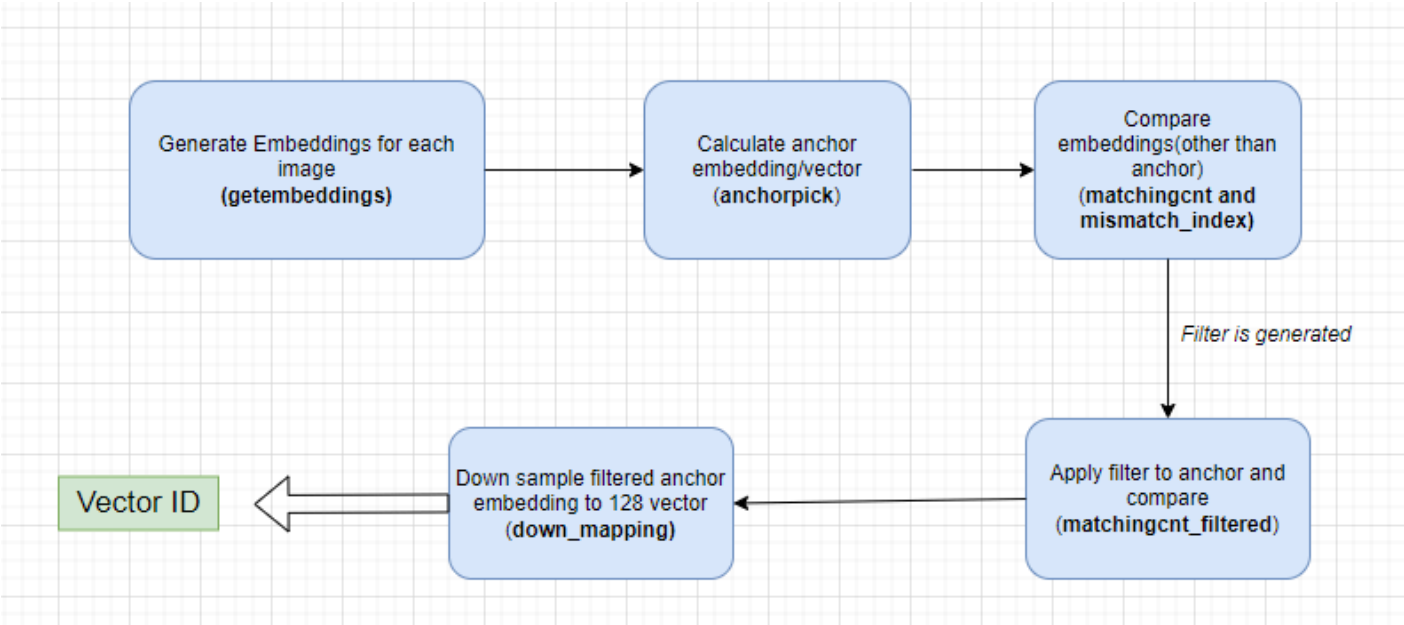
There are two files comprising the Alpha Version, ***driver.py*** and ***helper.py***. Both files are required to generate vector ID for a person where *five images* are taken as input. The file ***driver.py*** has to be run after installing the necessary Python libraries which will be provided in the README. All the functions are packaged in ***helper.py*** meanwhile ***driver.py*** only calls the required functions.

Helper.py

This module contains various helper functions used to process facial image embeddings. These functions include calculating the number of matching values between vector embeddings, picking an anchor vector, generating embeddings using the DeepFace library, and more.
The list of functions in helper.py are:

1. ***matchingcnt***: Counts the number of matching values between two vector embeddings and prints the percentage of matching values.
2. ***matchingcnt_filtered***: Same as ***matchingcnt*** but for filtered_vectors.
3. ***mismatch_index***: Returns a list of indices with mismatched values.
4. ***anchorpick***: Selects the anchor vector by choosing the closest vector to all other vectors.
5. ***down_mapping***: Down-maps the values in an array to a vector of length 128.
6. ***getembeddings***: Generates embeddings for a list of image paths using the Facenet512 model from DeepFace.
7. ***mismatch***: Primary function that performs the embedding mismatch process, calculates matching values and returns the filtered results.

The diagram below shows which function is being used at which stage.



Code Walk-through

The first function call starts with ***getembeddings*** which generates embeddings of size 512 using Facenet. The output is an embedding matrix of size (512, 5) where each column corresponds to each image.

The next function call starts with ***mismatch.*** This is the primary function that calls other functions as and when needed. The input of this function is the embedding matrix. The output consists of accuracy, vector_length, and anchor vector.

(Side Note: Accuracy is synonymous with the percentage of matching between two vectors.)

Initialization:

- An empty list `mmi` is initialized to keep track of mismatched indices.
- The `anchorpick` function is called with the `matrix` of embedding vectors to determine the anchor vector. The anchor vector is the one that is closest to all other vectors based on Euclidean distance.
- The function returns the anchor vector and its index, stored in `anchor` and `ainx`, respectively.

Comparison of Vectors:

- The function enters a nested loop to compare pairs of embedding vectors.
- The outer loop iterates from 0 to 3 (four vectors), and the inner loop iterates from `i+1` to 4 (remaining vectors).

- Apart from the anchor vector all other vectors are compared among themselves.

Counting Matches and Mismatches:

- The `matchingcnt` function is called to compare the selected vectors `x` and `y`. This function counts the number of matching values between the two vectors and appends mismatched indices to `mmi`.

Mismatched Indices Processing:

- After all comparisons, the `mismatch_index` function is called with `mmi` to get a list of unique mismatched indices (synonymous with filter).
- A mask array of length 512 (the length of the embedding vectors) is created and initialized to `True`.
- Indices in the mask that correspond to mismatched indices are set to `False`.

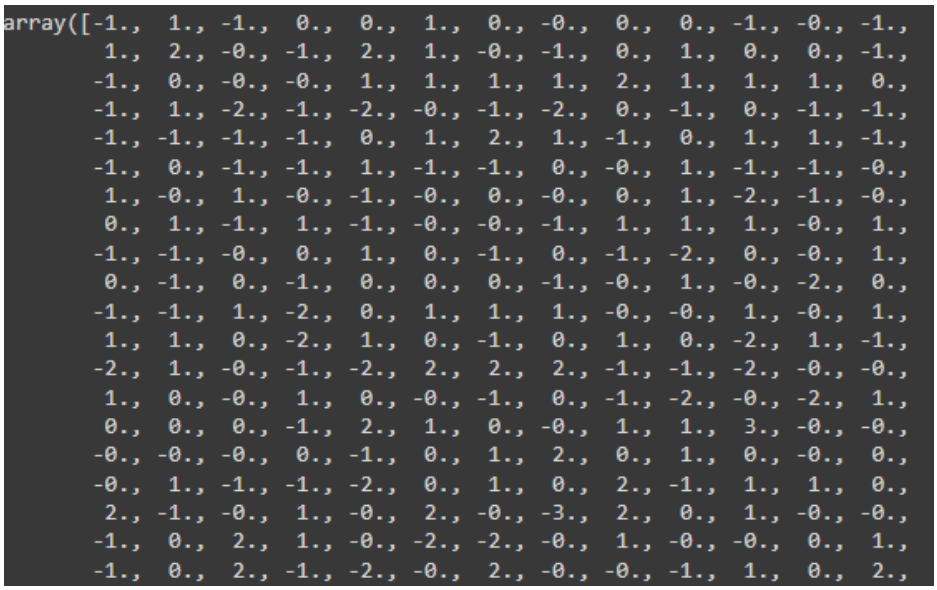
Filtering Embedding Vectors:

- The mask is applied to the anchor vector to get the `filtered_anchor`.
- The filtered lengths of the embedding vectors are calculated by subtracting the number of mismatched indices from 512.
- The `matchingcnt_filtered` function is called to count the number of matching values between the filtered anchor and another filtered vector from the matrix.

Return Values:

- The function returns three entities:
    - The percentage of matching values after filtering.
    - The length of the filtered embedding vectors (`mcnt`).
    - The filtered anchor vector (`filtered_anchor`).

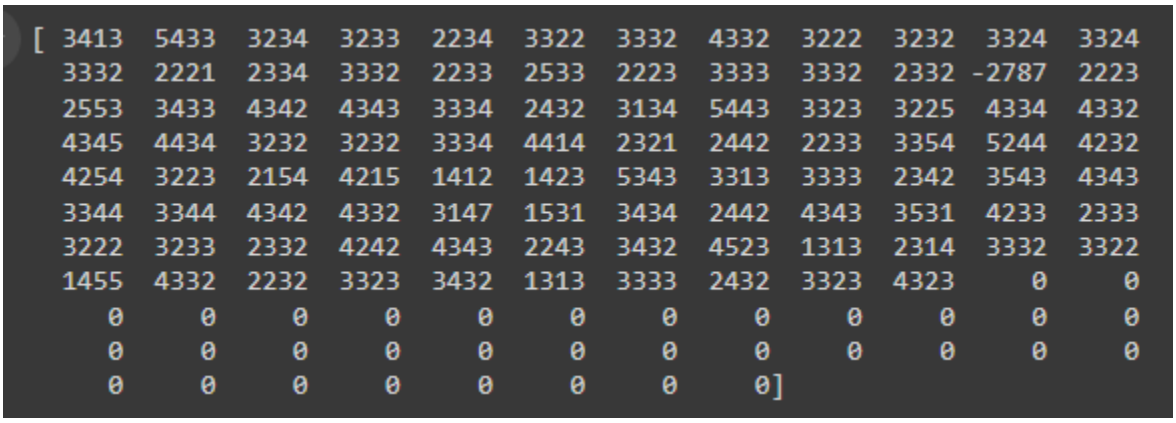
Once the filtered anchor vector is generated it looks like this:



```
array([-1.,  1., -1.,  0.,  0.,  1.,  0., -0.,  0.,  0., -1., -0., -1.,
        1.,  2., -0., -1.,  2.,  1., -0., -1.,  0.,  1.,  0.,  0., -1.,
       -1.,  0., -0., -0.,  1.,  1.,  1.,  1.,  2.,  1.,  1.,  1.,  0.,
       -1.,  1., -2., -1., -2., -0., -1., -2.,  0., -1.,  0., -1., -1.,
       -1., -1., -1., -1.,  0.,  1.,  2.,  1., -1.,  0.,  1.,  1., -1.,
       -1.,  0., -1., -1.,  1., -1., -1.,  0., -0.,  1., -1., -1., -0.,
        1., -0.,  1., -0., -1., -0.,  0., -0.,  0.,  1., -2., -1., -0.,
        0.,  1., -1.,  1., -1., -0., -0., -1.,  1.,  1.,  1., -0.,  1.,
       -1., -1., -0.,  0.,  1.,  0., -1.,  0., -1., -2.,  0., -0.,  1.,
        0., -1.,  0., -1.,  0.,  0.,  0., -1., -0.,  1., -0., -2.,  0.,
       -1., -1.,  1., -2.,  0.,  1.,  1.,  1., -0., -0.,  1., -0.,  1.,
        1.,  1.,  0., -2.,  1.,  0., -1.,  0.,  1.,  0., -2.,  1., -1.,
       -2.,  1., -0., -1., -2.,  2.,  2.,  2., -1., -1., -2., -0., -0.,
        1.,  0., -0.,  1.,  0., -0., -1.,  0., -1., -2., -0., -2.,  1.,
        0.,  0.,  0., -1.,  2.,  1.,  0., -0.,  1.,  1.,  3., -0., -0.,
       -0., -0., -0.,  0., -1.,  0.,  1.,  2.,  0.,  1.,  0., -0.,  0.,
       -0.,  1., -1., -1., -2.,  0.,  1.,  0.,  2., -1.,  1.,  1.,  0.,
        2., -1., -0.,  1., -0.,  2., -0., -3.,  2.,  0.,  1., -0., -0.,
       -1.,  0.,  2.,  1., -0., -2., -2., -0.,  1., -0., -0.,  0.,  1.,
       -1.,  0.,  2., -1., -2., -0.,  2., -0., -0., -1.,  1.,  0.,  2.,
```

The length of this anchor vector depends on the filter that was generated using the other 4 embeddings. Due to this, the length of the anchor vector varies from person to person. To resolve this down sampling is done to get a final vector ID of length 128.

The function ***down_mapping*** handles this. It does value mapping by iterating over each element and mapping it to a specific positive integer. Then to downsize, for each chunk of 4 values a new integer is created by concatenating the four values (v128).



```
[ 3413  5433  3234  3233  2234  3322  3332  4332  3222  3232  3324  3324
  3332  2221  2334  3332  2233  2533  2223  3333  3332  2332 -2787  2223
  2553  3433  4342  4343  3334  2432  3134  5443  3323  3225  4334  4332
  4345  4434  3232  3232  3334  4414  2321  2442  2233  3354  5244  4232
  4254  3223  2154  4215  1412  1423  5343  3313  3333  2342  3543  4343
  3344  3344  4342  4332  3147  1531  3434  2442  4343  3531  4233  2333
  3222  3233  2332  4242  4343  2243  3432  4523  1313  2314  3332  3322
  1455  4332  2232  3323  3432  1313  3333  2432  3323  4323     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0]
```

This ensures that the final length of vector IDs will be 128.

Driver.py

This script uses helper functions to process face images, generate embeddings, and produce a vector ID for a person based on facial images.

Modules Imported:

1. ***mismatch***,
2. ***down_mapping***,
3. ***getembeddings*** from helper.py.

Code Walk-through

As the driver code is only a few lines we can summarize it in 5 steps as follows:

1. Set the image directory path.
2. Read image paths from the directory.
3. Generate embeddings using the getembeddings function.
4. Perform mismatch analysis to filter the embeddings and get an anchor vector.
5. Down-map the anchor vector to produce a vector ID.