Semi-automatic dataset labelling

Yuanzheng Fang

11.04.2019

Superviser: Hörmann Stefan

1. Background knowledge

In my work, the face matching consists of 3 steps of matching, each step with the following cleaning, and at last, all the pictures would be matched in different sets. The face matching with Matlab is based on face embeddings generated by the neural network. In the beginning, we need to load all face embeddings for the face matching. In the first step of matching, by calculating the distance between casts and aligned faces, several representatives of each cast are filtered, and the rest faces are with very high confidence.

After comparing candidates in each representative, the pictures with relatively low confidence would be cleaned. In order to avoid the empty set of representatives, representatives would be filled after cleaning according to the confidence of matching. Then, the high confidence set would be generated by matching representatives with other unmatched faces. After cleaning the high confidence set, other unmatched faces are matched with high conf set to generate low confidence set with the following cleaning. After 3 steps of matching, remaining faces would be matched with high confidence set manually.

2. Specification for face matching GUI

2.1 Generate face embeddings

At the beginning of the matching, face embeddings are essential. In order to get embeddings, we use the neural network with python. The python part could generate face embeddings and corresponding json files, which are

available to Matlab, as figure 1 shows. And then the user should add the path of all folders and subfolders in GUI_MATLAB to Matlab.

Moreover, in order to load the json files with Matlab, the jsonlab toolbox is required. After downloading the toolbox, the user should add the path of this toolbox to Matlab.

```
# Save the face embeddings as json (only once and then set to false)
with open(os.path.join(os.path.join(output_dir, movie), 'face_embeddings.json'), 'w') as file:
    json.dump(embeddings_aligned, file, sort_keys=True, indent=4)
with open(os.path.join(os.path.join(output_dir, movie), 'labels_aligned.json'), 'w') as file:
    json.dump(labels_aligned, file, sort_keys=True, indent=4)
with open(os.path.join(os.path.join(output_dir, movie), 'face_embeddings_unaligned.json'), 'w') as file:
    json.dump(embeddings_unaligned, file, sort_keys=True, indent=4)
with open(os.path.join(os.path.join(output_dir, movie), 'labels_unaligned.json'), 'w') as file:
    json.dump(labels_unaligned, file, sort_keys=True, indent=4)
with open(os.path.join(os.path.join(output_dir, movie), 'cast_embeddings.json'), 'w') as file:
    json.dump(embeddings_cast, file, sort_keys=True, indent=4)
with open(os.path.join(os.path.join(output_dir, movie), 'labels_cast.json'), 'w') as file:
    json.dump(labels_cast, file, sort_keys=True, indent=4)
```

Figure 1 Generate json files with python.

2.2 Face matching part

The face matching part consists of three matchings, namely the first, second and third matching. After each matching, user could get the corresponding matching result, which is necessary for the next matching step. Here are some source codes for the matching.

Compute_dist_cert: This function is used to calculate the distance and the certainty between face embeddings, which is also available to predict the matched cast for each face.

Compute_infra_cast_distance: The function is designed to calculate the mean value of the match result, which could be used for cleaning.

2.3 Introduction for GUI

The graphical user interface for face matching consists of 2 button groups, namely **'Main matching'** and **'Matching remaining faces'**.

‘Main matching’ part is used for the three-step main matching. In figure 2, we could see that **‘Main matching’** button group is composed of **‘movie menu’**, **‘cast’** panel, **‘number of casts’**, **‘Matching status’**, and **‘next’** button**.**

‘movie_menu’: In this popup-menu, the user could choose one movie to start the matching, and there would be a dialog that reminds you of the start. And then a folder, which shows the match result for the current cast showed in **‘cast’** panel, would be opened, as figure 3 shows.

‘number of casts’: This edit txt shows the number of casts of the current movie. And the txt above it indicates which cast the user is matching now.

‘cast’ panel: The panel shows the cast are matching now.

‘Matching status’: This part is designed to tell user the current matching step.

‘next’ button changes both the cast showed in ‘cast’ panel and corresponding matching result in the show folder each time user presses it. When the main matching completes, a new dialog box appears to remind user of the end of main matching.

‘Matching remaining faces’ button group is used to match remaining faces. In the figure 2, we could see that **‘movie cast’** shows the representatives of casts in movie. For each remaining face, we could get the information of the current face and recommended cast from the button group. If the recommended cast is not correct, the user could also choose the correct cast by himself. And then the face would be assigned to the matched cast by pressing **‘verify’** button.

2.4 Procedure of face matching

After generating the json file and adding the path of jsonlab toolbox to Matlab, the user could run **'/GUI_MATLAB/gui_src/gui_0403.m'** to start matching. After that, the user could change the prefix in gui_0403_OpeningFcn, in order to have access to the database in his own PC.

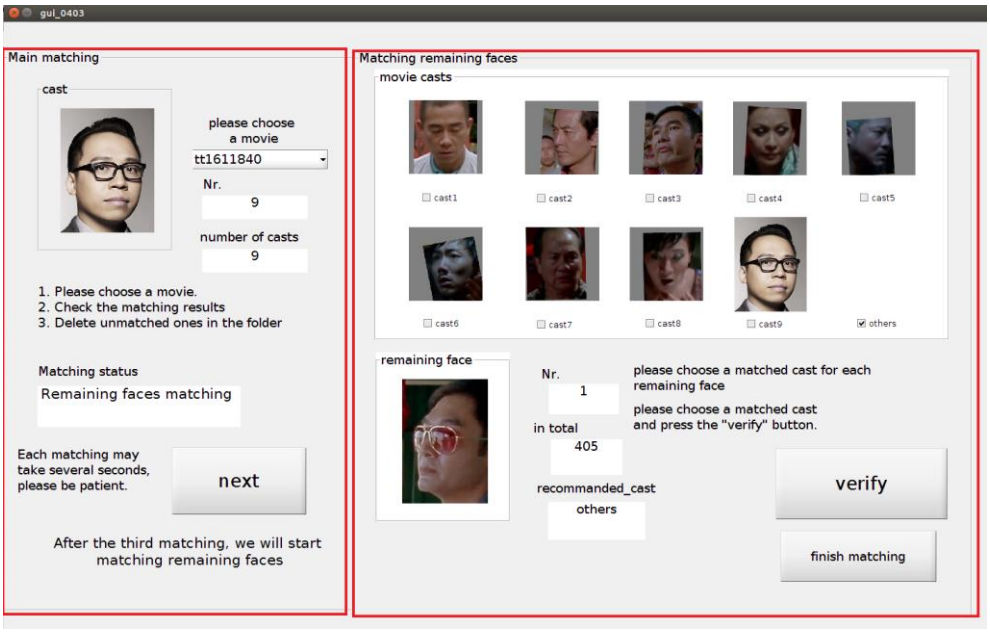

Figure 2 Graphical user interface for face matching.

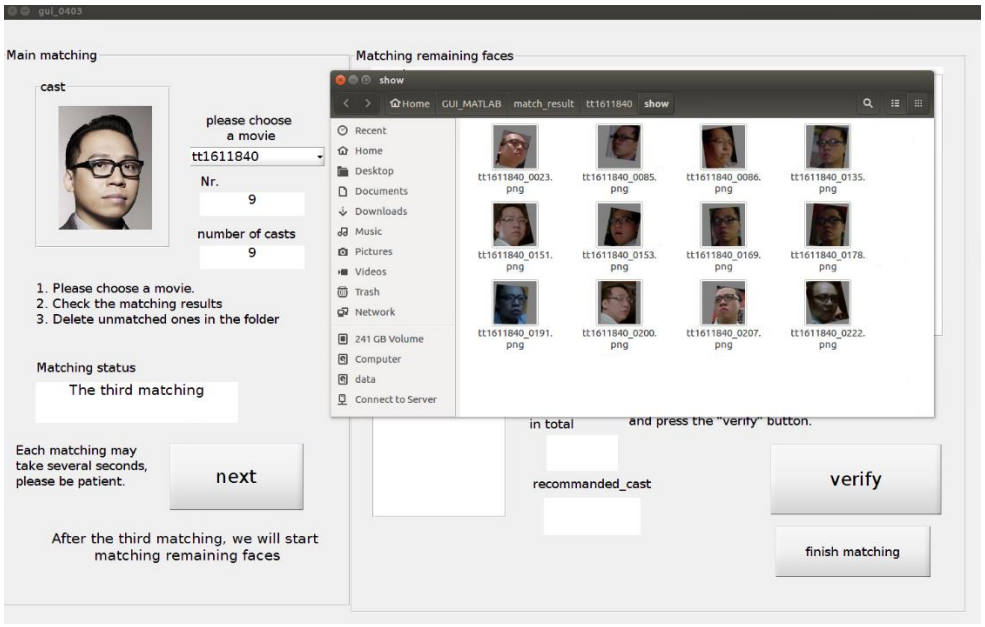

Figure 3 The show folder for matching result.

Firstly, the user chooses one movie for matching, and then the main matching starts. After that the user deletes unmatched pictures in the current folder, he could press **'next'** button to check the match result of next cast and remove pictures from the match result. Moreover, after that all casts have been matched in the current matching step, the button **'next'** leads to the next matching step, until the main matching part completes. It takes several seconds when a new matching step starts. Moreover, when the main matching part completes, the matching of remaining faces begins at the same time.

For each remaining face, GUI supplies recommended cast automatically. Furthermore, the user could choose the matched cast by himself as well. Next, the user could press the **'verify'** button to verify the matching and check the next face. When the whole matching for remaining faces finishes, it pops up a dialog box to remind user of the end of the matching. And then, the user could choose another movie for matching. Moreover, match result of movie would be saved in the corresponding movie folder in GUI_MATLAB. At last, when all movies have been matched, the user could press **'finish matching'** button to close GUI.