Evidenzen

10th April 2019

Gruppe: Anne Born, Andreas Fuchs

ABBREVIATIONS

Features:

- Fob = furrowing of brow
- Lea = left eye aperture
- Rea = right eye aperture
- Lbd = left brow distance
- Rbd = right brow distance
- Hnc = horizontal nose crinkles
- Vnc = vertical nose crinkles
- Rcw = right cheek wrinkles
- Lcw = left cheek wrinkles
- Ma = mouth aperture

Emotions:

- N = neutral
- S = sadness
- F = fear
- H = happiness
- D = disgust

Values:

- S = small/none
- M = medium
- L = large/high

General:

- M = masses (basismass)
- DS = Dempster-Shafer

GLOSSARY

Features:

Features are the facial features relevant for the determination of the emotion that is most likely shown in a picture. The specific features that are analysed are based on the example data.

Emotions:

The emotions are the possible outcomes for the dempster-shafer theory. They are based on the description given with the task.

Mass:

The mass is an essential part of the Dempster-Shafer Theory. For each ‘hint’ (in our case a single feature and its value) a mass is assigned to all ‘affected’ sets of outcomes. A special kind of mass is the remaining mass (spanning all possible outcomes, omega), representing the lack of knowledge (Unwissenheit). In the code this mass is called ‘O’ (for omega).

Dempster-Shafer:

The Dempster-Shafer Theory, also called theory of belief functions, is used to model reasoning with uncertainty. In this project the usage of dempster-shafer (ds) always refers to the dempster-shafer rule for combination. It describes the accumulation of two masses to a new third one. In an abstract sense this means combining two ‘hints’ you have together and thus narrowing down the knowledge you are still lacking and the possible outcomes.

Conflict:

A Conflict happens when two power-sets are accumulated by using the dempster-shafer rule but are disjoint, meaning they have no common outcomes. In the code the conflict is referred to as ‘k’.

OVERVIEW

Emotionen	FOB	LEA	REA	LBD	RBD	HNC	VNC	LCW	RCW	MA
Neutral	s	m	m	m	m	none**	none**	none**	none**	N/A*
Sadness	l	s-m	s-m	m	m	m	N/A*	s-m	s-m	N/A*
Fear	l	l	l	l	l	l	N/A*	N/A*	N/A*	l
Happiness	m	l	l	l	l	none**	N/A*	m-l	m-l	N/A*
Disgust	s	s	s	s	s	s, m, l	N/A*	m	m	N/A*

* N/A means that the task description does not make a statement for this feature. It was represented in the code by the python type 'None', which represents the empty state

**none means that the task description explicitly says that the value is none. For reasons of simplicity and because it was not further specified, it is treated as 'small' in the code

GOALS

1. The program should be able to read input of a specific format (representing a list of features as specified in the table above) and determine the emotion that is most probably depicted by the values.
2. The program should determine the categories (small, medium, large) for the given feature-values dynamically, based on the range of values in the given dataset
3. The determination of emotions should be done by the so called "dempster-shafer theory"

SPECIFICATION:

Programming language:

Python 3.8

Git-Repository:

https://github.com/dhbw-stginf16a/wbs-evidence

FUNCTIONS:

In this section all functions used by the programm are explained:

1. create_emotion_object(fob, lea, rea, lbd, rbd, hnc, vnc, lcw, rcw, ma)
    - Input: 10 lists
        i. One for each feature
    - Output: Emotion Object, containing the facial expression
    - Used to fill the global emotions dict with the values given in the task description
2. import_csv(file)
    - Input: path to the dataset file (.csv)
    - Output: array of frames
    - Used to import the dataset (.csv file) into an array of frames.
3. evaluate_features(frames)
    - Input: dict frames
    - Output: dict mapped_frames
    - This function maps the numerical values for features of each frame to the categories small (s), medium (m) and large (l) and returns a dict of a similar structure but with the respective categories instead of numerical values
4. calc_m(frame):
    - Input: dict frame
    - Output: dict m_vals
    - This function takes a frame with categorical values for each feature and mapps the corresponding power sets (Potenzmengen) with a mass. Since not further specified we set this mass to 0.8 and thus the remaining mass (Unwissenheit), called ‘O’ for omega to 0.2. However those numbers can easily be changed.
    - For each information about a feature a mass is calculated for all afflicted emotions and for the remaining mass so that the sum of all values for one mass is1.
5. ds_accum(m1, m2):
    - Input: two dicts m1 and m2
        i. Both represent the power sets for a mass
    - Output: dict result_m
    - This function implements the dempster shafer rule, accumulating to masses to a new, third mass.
    - It considers all possible cases (intersection, omega, and conflict
    - If the conflict sums up to 1, the user is informed about the error and the program terminates since this would lead to a division by zero and ist not defined by the dempster-shafer rule.

6. iterate_ds_accum(dict_masses):
    - Input: dict dict_masses
        i. A dict with all masses
    - Output: dict_masses['masse'][X]
        i. The last calculated mass, since this is the final result
    - This function applies the ds_accum(m1, m2) function to a set of masses and returns the last calculated mass which automatically has the smallest remaining mass 'O' and is based on the most 'knowledge'
    - Since the Dempster Shafer rule can only accumulate two masses to a third one, it needs to be applied iteratively if more than two masses exist.
7. calc_plaus(mass):
    - Input: list mass
    - Output: dict plausibility
    - This function calculates the plausibility of an outcome (in this use case an emotion) based on a given mass (with all power sets of this mass)
8. check_max_plaus(plausibility):
    - Input: dict plausibility
    - Output: emotion with the highest plausibility
9. map_plausibility(frames):
    - Input: dict frames
    - Output: dict result
    - This function takes the frames as a parameter and calculates the masses and plausibilities for each frame. Then it determines the emotion with the highest plausibility and returns a frame containing all the plausibilities and the classified emotion.
10. print_result(result):
    - Input: array of dictionaries containing the evaluated frames
    - Output: None
    - This function formats the given input to print a human readable result

DATA STRUCTURES:

In this part of the documentation, the most important data structures of the program are explained shortly to facilitate a deeper understanding

Global:

Dict emotions:

A dictionary that contains one entry for each emotion. The emotions are abbreviated with the beginning letter to enable the usage of sets and the functions that come with them for the dempster shafer rule. Each emotion entry is another dictionary containing key-value pairs with the features as keys and the category as values.

Example:

'd': {'fob': ['s'], 'lea': ['s'], 'lbd': ['s'], 'rea': ['s'], 'rbd': ['s'], 'hnc': ['s', 'm', 'l'], 'vnc': [None], 'lcw': ['m'], 'rcw': ['m'], 'ma': [None]}

The given example is the 'subdictionary' of the disgust (d) emotion. We can see that the furrowing of brow (fob), when showing disgust is typically small (s), whereas the horizontal nose crackings (hnc) can vary from small to large.

Local:

Dict frames:

A dictionary that contains the numerical values for each feature in each frame and the second as an identifier. A frame is represented as a sub-dictionary:

Example:

{'sec': 60, 'fob': 23963, 'lea': 21, 'lbd': 18, 'rea': 19, 'rbd': 18, 'hnc': 5, 'vnc': 3, 'lcw': 10, 'rcw': 10, 'ma': 78}]

The given example shows the values that were given in the csv file for the 60th second. It can be seen, that the pixel-based value for rea is 19.
The frames dict is the input for the evaluate_frames function.

Dict mapped_frames:

A dictionary similar to the frames dict but with values from the categories (s, m, l) instead of numerical values.

Example:

{'sec': 60, 'fob': 's', 'lea': 'm', 'lbd': 's', 'rea': 'm', 'rbd': 's', 'hnc': 'm', 'vnc': 's', 'lcw': 's', 'rcw': 's', 'ma': 'l'}

This example shows the same frame from the same dataset as the example for the frames dict. Now we can see that the value for rea was evaluated to be medium (m).
This dict is the output of the evaluate_frames function.

Dict m_vals:

This dict contains all the masses, for each frame and each feature. It is structured in sub-dictionaries for single frames that are divided in two key-value pairs - ‘sec’ and ‘masse’. The ‘sec’ keeps track of the frame number (or the second of the video clip) that the masses are for, since each frame has to be viewed separately. The value of the ‘masses’ is a list of dicts each with two key-value pairs. One key-value pair represents the power set with the emotions that are supported by the frame value concatenated to a single string and the other always has ‘O’ as a key (for omega) and represents the remaining mass (1 - the other value).

Example:

{'sec': 60, 'masse': [{'O': 0.2, 'nd': 0.8}, {'O': 0.2, 'ns': 0.8}, {'O': 0.2, 'd': 0.8}, {'O': 0.2, 'n

s': 0.8}, {'O': 0.2, 'd': 0.8}, {'O': 0.2, 'sd': 0.8}, {'O': 0.2, 'n': 0.8}, {'O': 0.2, 'ns': 0.8}, {'O

': 0.2, 'ns': 0.8}, {'O': 0.2, 'f': 0.8}]}

This example shows the sub-dictionary that contains all masses related to the 60s second (or the 60s frame). We can see the two key-value pairs ‘sec’ and ‘masse’.

Dict m:

This dict contains a single mass with all related power sets as keys and the according ‘probabilities’ as values.

Example:

{'O': 0.04000000000000001, 'ns': 0.16000000000000003, 'nd': 0.16000000000000003, 'n': 0.6400000000000001}

PROGRAM-FLOW

Now that all functions and data structures have been explained, the overall workflow is described. The first thing that happens is the initialisation of the emotions data-structure with the ‘create_emotion_objects’ function. The values that are passed into the function for each emotion are hard-coded since they are given with the task and do not change dynamically. Except for this, all other function calls either happen within another function or in the ‘main entry’ block. After the initialisation of the emotions the condition in the main entry block is evaluated. If the condition evaluates to true, meaning that some argument was given along with the execution of the file (see Usage), the file is read by the import_csv function and the return value is assigned to the emotion_frames list. Now a list with a dictionary entry for each frame (or each second of video), containing key value pairs with the feature and the numerical values for each given feature exists. This data structure is then passed into the evaluate_features function. This function then calls the value_range function which calculates the range of all values of a feature (except for ‘sec’) and then returns the minimum value, the maximum value and the step_size, which is the range divided by 3. The returned values are then used to map each feature value of each frame to the categories (s, m and l). This mapped_frames data-structure is then returned and assigned to emotion_frames. Next the map_plausibility function is called with the mapped_frames structure as an argument. In this function all the calculation for the dempster-shafer theory is conducted with the help of multiple other functions;
First for each frame, the initial masses are calculated with the calc_m function. Next the result of this (initial_m) is passed to the iterate_ds_accum function to accumulate all masses step by step. Since the iterate_ds_accum function returns the last calculated mass, this value is assigned to final_m. Now that the final masses are calculated, the next step ist to calculate the plausibility of all outcomes based on this. This is done by calling the calc_plaus function. The code only calculates the plausibility for the final masses because this is what determines the decision made by the programm, since it is based on the most accumulated ‘knowledge’.
Finally the maximum plausibility is determined by the check_max_plaus function and returned as ‘emotion’. Now the result_frame is filled with the all plausibilities, the number and the abbreviation of the emotion with the highest plausibility.

This result is then printed, showing the user the plausibility of each emotion (rounded to two decimals) and the classification based on the highest plausibility. This printing is done by the print_result function

USAGE

To run the python script you need to execute the following command:

python main.py data.csv

EVALUATION

While testing the final version of the application strange behavior was noticeable. The program didn't seem to be returning the same result, even though the same input was given. Concretely the 3rd frame of *emo_muster_1_1.csv* was alternating between "Sadness" and "Neutral" on different executions. After a short investigation it was clear, that this variation is occuring due to rounding errors in Python. Since the "Neutral" and "Sadness" Emotion are really similar (e.g. 's'=0.5098645782346409, 'n'=0.5098645782346408), the rounding error would vary on every execution leading to a different result.

Nevertheless the application returns good results - even though not perfect - for all of the provided datasets.