Structure of DashBot Implementation

SDC

22 octobre 2021

Root

DashBot is implemented to run via a user interface as well as perform experiments. The folder implem contains the main part of the code, shared by both usages. experiment/ contains the code specific to experiments and interface/ the code specific to the interface.

In the root folder, you will also find the python script to execute :

— start-api.py for the interface

— start-experiment.py to run experiments.

1 Main (implem folder)

Interface (resp. experiment) runs with an instance of class Interface (resp. Experiment) that inherites from class DashBot.

Class DashBot is defined in implem/dashbot.py. It uses :

— modules, found in folder implem/modules/ ;

— binary files, found in folder implem/preprocessing_cache/, storing the result of the preprocessing step for each available dataset, if it has already been done once ;

— csv files, found in implem/datasets/, if the binary file corresponding to the chosen dataset is not found ;

— implem/datasets/datasets.py links the name of the dataset (as asked by user) and the binary or csv file.

When initializing an instance of DashBot, the parameters are loaded into an instance of class Parameters (found in modules/parameters.py) from a yaml file found in experiments/ or interface/. The class Parameters allows to check if entered parameters are consistent and process them to be used into DashBot.

The main methods of DashBot are :

— preprocess_data(dataset_name) ;

— initialize_dashboard_generation() ;

— update_system() ;

— find_next_suggestion() ;

The first 2 methods are called once at the beginning, the last 2 after each user feedback.

1.1 Preprocess Data

When this method is called, an instance of class DataPreprocessor (found in modules/data_preprocessor.py) is first created. Then an instance of class AttributesRanking (found in modules/rankings.py) is created.

1.1.1 DataPreprocessor

During its initialization, 2 cases can be found :

1. the corresponding pickle file already exists :
   Then the result of preprocessing is loaded
2. the corresponding pickle file is not found :
   Then, the preprocessing is done and the pickle file is created.

The preprocessing result is :

— a list of instances of class Attribute (found in modules/attribute.py), for each attribute in the dataset : These instances have attributes describing the attributes (if it is numeric or not, etc.) and resulting from preprocessing computations (entropy, variance, class to rank them, if it has been discretized or not, etc.).

— the relation R^*, containing all data + new columns for discretized attributes, stored in a pd.DataFrame instance.

1.1.2 AttributesRanking

This instance will store rankings of attributes in terms of relevance.

— self.preprocessed :
rankings for groupy and aggregation, as the direct result of preprocessing (no update depending on what happens during generation)

— self.general :
rankings for groupy and aggregation, taking into account what happens during generation if it is a persistent information :

1. bad groupby attributes reported by user are put in the 'bad' list, others in a 'good' list.
2. attributes already on dashboard (if diversity is asked and not achieved) are put in 'less good' if previously in 'good', 'very bad' if previously in 'bad'.

— self.local :
modification of self.general depending on the current state. These non-persitent modifications can be applied on e.g. :

— bad aggregation attributes linked to the chosen groupby attribute(s).
This depends on the current panel under construction. They are put in the 'bad' or 'very bad' list.

— attributes that just have been reported by user explanation.
This depends on the explanation given by the user during the last feedback, if applicable. They are completely removed from rankings and not just set as 'bad' or 'very bad' as they will be afterwards.

1.2 Initialize Dashbord Generation

This method initializes usefull instances for generation :

1.2.1 Diversity

— self.diversity['asked'] : True/False, depending on if it is asked by the user (this is an option that can be changed by the user on the interface, this is a parameter that have to be put in the parameters file for experiments).

— self.diversity['achieved'] : True/False, depending on the attributes that are already on the dashboard.

1.2.2 Dashboard

An instance of class Dash (found in modules/panel.py), which is an object representing the dashboard, i.e. containing several panels.

1.2.3 DashBot History

A list of the panels that have already been suggested. Each panel is stored as a list of integers corresponding to the vector representation of the panel.

1.2.4 Panel

An instance of class Panel (found in modules/panel.py), which is an object representing the current panel that is about to be / just have been shown to the user.

A panel can be either represented by :

1. its vector representation panel.vector
   a instance of pd.Series with a multi-index ('attribute', 'function').
2. objects containing attributes on groupby and aggregation dimensions :

   — panel.groupBy : list of instances of class Attribute

   — panel.aggregates : dict with instances of class Attribute as keys and list of function-strings as values.

For example, if dataset contains only attributes gender, age and BMI, the panel corresponding to the SQL query :

```
SELECT      gender, min(age), max(age)
FROM        dataset
GROUP BY    gender,
```

is either represented by :

1. the pd.Series panel.vector

*	gender					age					BMI				
count	gb	min	avg	max	sum	gb	min	avg	max	sum	gb	min	avg	max	sum
0	2	0	0	0	0	0	1	0	1	0	0	0	0	0	0

2. a combination of :

   — panel.groupBy = [Attribute(gender)]

   — panel.aggregates = { Attribute(age) : ['min', 'max'] }

Depending on the algorithm used, modifications are made on one representation or the other. The untouched representation is then updated with methods panel.vector_to_attributes() if changes have been made on vector representation, panel.attributes_to_vector() otherwise.

1.3 Update System

1.4 Find Next Suggestion

2 Interface (interface folder)

3 Experiment (experiment folder)