FCApy

Release 0.0.5

Egor Dudyrev

Dec 04, 2020

Table of Contents

class `fcapy.context.formal_context.`**`FormalContext`** (*data=None*, *object_names=None*, *attribute_names=None*, ***kwargs*)

A class used to represent Formal Context object from FCA theory.

`intention` (*objects*)

Return maximal set of attributes which are shared by given `objects`

`extension` (*attributes*)

Return maximal set of objects which share given `attributes`

`intention_i` (*object_indexes*)

Offer the same logic as intention(…) but objects and attributes are defined by their indexes

`extension_i` (*attribute_indexes*)

Offer the same logic as extension(…) but objects and attributes are defined by their indexes

`to_cxt` (*path=None*)

Convert the FormalContext into cxt file format (save if `path` is given)

`to_json` (*path=None*)

Convert the FormalContext into json file format (save if `path` is given)

`to_csv` (*path=None*, ***kwargs*)

Convert the FormalContext into csv file format (save if `path` is given)

`to_pandas` ( )

Convert the FormalContext into pandas.DataFrame object

Notes

Formal Context K = (G, M, I) - is a triplet of: 1. set of objects G (the property `object_names` in this class) 2. set of attributes M (the property `attribute_names` in this class) 3. binary relation I between G and M (i.e. "gIm holds True" means "object g has attribute m")

(the property `data` in this class)

property `attribute_names`

Get of set the names of the attributes in the context

Parameters **`value`** (`list of `str) – The list of names for the attributes (default are "0","1",…,"*n_attributes*-1")

Raises **`AssertionError`** – If the number of names in the `value` does not equal to the

number of attributes in the context If the the elements of `value` are not of type str

property data

Get or set the data with relations between objects and attributes (`list of `list)

Parameters **value** (`list of `list) – value[i][j] represents whether i-th object shares j-th attribute

Raises **AssertionError** – If `value` is not a *list* *If ``value`` of type list is given (should be `list of `list)* *If some lists ``value[i]`` and `value[j]` have different length (should be the same for any `value[i]`)* If any `value[i][j]` is not of type `bool`

property description

Get or set the human readable description of the context
JSON is the only file format to store this information. The description will be lost when saving context to .cxt or .csv

Parameters **value** (`str, None) – The human readable description of the context

Raises **AssertionError** – If the given `value` is not None and not of type `str`

extension (*attributes*)

Return maximal set of objects which share given `attributes`

Parameters **attributes** (`list of `str) – Names of the attributes (subset of `attribute_names`)

Returns **extension** – Names of the maximal set of objects which share given `attributes`

Return type `list of `str`

extension_i (*attribute_indexes*)

Return indexes of maximal set of objects which share given `attribute_indexes`

Parameters **attribute_indexes** (`list of `int) – Indexes of the attributes (from [0, `n_attributes`-1])

Returns **extension_indexes** – Indexes of maximal set of objects which share `attributes`

Return type `list of `int`

intention (*objects*)

Return maximal set of attributes which are shared by given `objects`

Parameters **objects** (`list of `str) – Names of the objects (subset of `object_names`)

Returns **intention** – Names of maximal set of attributes which are shared by given `objects`

Return type `list of `str`

intention_i (*object_indexes*)

Return indexes of maximal set of attributes which are shared by given ``object_indexes`

Parameters **object_indexes** (`list of `int) – Indexes of the objects (from [0, `n_objects`-1])

Returns **intention_i** – Indexes of maximal set of attributes which are shared by `objects`

Return type `list of `int`

property n_attributes

Get the number of attributes in the context (i.e. len(*data[0]*)

property n_objects

Get the number of objects in the context (i.e. len(*data*))

property `object_names`
Get of set the names of the objects in the context

Parameters **`value`** (`list of `str) – The list of names for the objects (default are '0','1',...,'*n_objects*-1')

Raises **`AssertionError`** – If the number of names in the `value` does not equal to the number of objects in the context If the the elements of `value` are not of type str

`print_data` (*max_n_objects=20*, *max_n_attributes=10*)
Get the FormalContext date in the string formatted as the table

Parameters
- **`max_n_objects`** (`int) – Maximal number of objects to print. If it is less then `n_objects` then print `max_n_objects/2` objects from the "top" and the "bottom" of the context
- **`max_n_attributes`** (`int) – Maximal number of attributes to print. If it is less then `n_attributes` then print `max_n_attributes/2` attributes from the "left" and the "right" part of the context

Returns **data_to_print** – A string with the context data formatted as the table

Return type `str

`to_csv` (*path=None*, ***kwargs*)
Convert the FormalContext into csv file format (save if `path` is given)
Parameters

Returns **context** – If `path` is None, the string with .csv file data is returned. If `path` is given - return None

Return type `str

`to_cxt` (*path=None*)
Convert the FormalContext into cxt file format (save if `path` is given)

Parameters **`path`** (`str or None) – Path to save a context

Returns **context** – If `path` is None, the string with .cxt file data is returned. If `path` is given - return None

Return type `str

`to_json` (*path=None*)
Convert the FormalContext into json file format (save if `path` is given)

Parameters **`path`** (`str or None) – Path to save a context

Returns **context** – If `path` is None, the string with .json file data is returned. If `path` is given - return None

Return type `str

`to_pandas` ( )
Convert the FormalContext into pandas.DataFrame object

Returns **df** – The dataframe with boolean variables, `object_names` turned into `df.index`, `attribute_names` turned into `df.columns`

Return type pandas.DataFrame

fcapy.context.converters.**from_pandas** (*dataframe*)

Create FormalContext object based on pandas.DataFrame
context.object_names are parsed from dataframe.index. context.column_names are parsed from dataframe.columns

Parameters **dataframe** (`pandas.DataFrame) –

Returns **ctx**

Return type `FormalContext

fcapy.context.converters.**read_csv** (*path*, *sep=','*, *word_true='True'*, *word_false='False'*)

Read FormalContext from .csv file

Parameters **path** (`str) – A path to requested .csv file

Returns **ctx** – The loaded FormalContext object

Return type `FormalContext

fcapy.context.converters.**read_cxt** (*path*)

Read FormalContext from .cxt file

Parameters **path** (`str) – A path to requested .cxt file

Returns **ctx** – The loaded FormalContext object

Return type `FormalContext

fcapy.context.converters.**read_json** (*path*)

Read FormalContext from .json file

Parameters **path** (`str) – A path to requested .json file

Returns **ctx** – The loaded FormalContext object

Return type `FormalContext

fcapy.context.converters.**to_pandas** (*context*)

Convert FormalContext object to pandas.DataFrame object
context.object_names are turned into df.index. context.column_names are turned into df.columns

Parameters **context** (`FormalContext) – A context to convert

Returns **df** – The binary dataframe based on FormalContext object

Return type `pandas.DataFrame

fcapy.context.converters.**write_csv** (*context*, *path=None*, *sep=','*, *word_true='True'*, *word_false='False'*)

Write FormalContext object to the .csv file

Parameters

- **context** (`FormalContext) – A context to write to a file
- **path** (`str) – A path to the file to write a FormalContext object

Returns **file_data** – The date from the .csv file. Returned if path is None

Return type `str

fcapy.context.converters.**write_cxt** (*context*, *path=None*)

Write FormalContext object to the .cxt file

Parameters
- **context** (`FormalContext) – A context to write to a file
- **path** (`str) – A path to the file to write a FormalContext object

Returns **file_data** – The date from the .cxt file. Returned if `path` is None

Return type `str

`fcapy.context.converters.`**write_json** (*context*, *path=None*)

Write FormalContext object to the .json file

Parameters
- **context** (`FormalContext) – A context to write to a file
- **path** (`str) – A path to the file to write a FormalContext object

Returns **file_data** – The date from the .json file. Returned if `path` is None

Return type `str

- *Index*
- *Module Index*
- *Search Page*

f