openclean-metanome

New York University

Jun 16, 2021

CONTENTS:

This package is an extension for the openclean-core package. It provides access to data profiling algorithms from the Metanome project in **openclean**. The algorithms themselves are executable via the Metanome Wrapper that enables to run Metanome algorithms via the command line.

CHAPTER

ONE

INSTALLATION & CONFIGURATION

The package can be installed using `pip`.

```
pip install openclean-metanome
```

The **openclean-metanome** package uses flowServ to run Metanome algorithms as serial workflows in **openclean**. **flowServ** supports two modes of execution: (1) using the Python sub-process package, and (2) using Docker.

1.1 Python Sub-Process

When running Metanome algorithms as Python sub-processes you need to have an installation of the *Jave Runtime Environment (Version 8 or higher)* on your local machine. You also need a local copy of the `Metanome.jar` wrapper. The file can be *downloaded from Zenodo <https://zenodo.org/record/4604964#.YE9tif4pBH4> `_*. The package also provides the option to download the file from within your Python scripts.

```
from openclean_metanome.download import download_jar

download_jar(verbose=True)
```

The example will download the jar file into the default directory (defined via the *METANOME_JARPATH* environment variable). If the variable is not set, the users default cache folder is used. Note that the `Metanome.jar` is currently about 75 MB in size. Make sure that the environment variable *METANOME_JARPATH* contains a reference to the downloaded jar-file if you did not download the file into the default location.

1.2 Docker

If you have Docker installed on your machine you can run Metanome using the provided Docker container image. To do so, make sure that the environment variable *METANOME_WORKER* references the configuration file `docker_worker.yaml` that is included in the config folder of this repository.

CHAPTER

TWO

ALGORITHMS

The package currently supports two data profiling algorithms.

2.1 HyFD

The HyFD algorithm (A Hybrid Approach to Functional Dependency Discovery) is a functional dependency discovery algorithm. Details about the algorithm can be found in:

:: Thorsten Papenbrock, Felix Naumann A Hybrid Approach to Functional Dependency Discovery ACM International Conference on Management of Data (SIGMOD '16)

For an example of how to use the algorithm in **openclean** have a look at the example notebook Run HyFD Algorithm - Example.

2.2 HyUCC

The HyUCC algorithm (A Hybrid Approach for Efficient Unique Column Combination Discovery) is a unique column combination discovery. Details about the algorithm can be found here.

For an example of how to use the algorithm in **openclean** have a look at the example notebook Run HyUCC Algorithm - Example.

CHAPTER

THREE

DOCKER CONTAINER FOR METANOME ALGORITHMS

Build the container containing the Metanome.jar file:

```
docker image build -t openclean-metanome:0.1.0 .
```

Push container image to DockerHub.

```
docker image tag openclean-metanome:0.1.0 heikomueller/openclean-metanome:0.1.0
docker image push heikomueller/openclean-metanome:0.1.0
```

CHAPTER

FOUR

OPENCLEAN_METANOME

4.1 openclean_metanome package

4.1.1 Subpackages

openclean_metanome.algorithm package

Submodules

openclean_metanome.algorithm.base module

openclean_metanome.algorithm.base.**run_workflow**(*workflow: flowserv.controller.serial.workflow.base.SerialWorkflow*, *arguments: Dict*, *df: pandas.core.frame.DataFrame*, *worker: Optional[Dict] = None*, *volume: Optional[Dict] = None*, *managers: Optional[Dict] = None*, *verbose: Optional[bool] = True*) → flowserv.controller.serial.workflow.result.RunResult

Run a given workflow representing a Metanome profiling algorithm on the given data frame.

Returns the run result. If execution of the Metanome algorithm fails a RuntimeError will be raised.

This implementation assumes that all algorithms operate on a single input file that contains a serialization of the data frame and that they all produce a single output file in Json format.

Parameters

- **workflow** (*flowserv.controller.serial.workflow.base.SerialWorkflow*) – Serial workflow to run a Metanome profiling algorithm on a given data frame.
- **arguments** (*dict*) – Dictionary of algorithm-specific input arguments.
- **df** (*pd.DataFrame*) – Input data frame.
- **worker** (*dict, default=None*) – Optional configuration for the main worker.
- **volume** (*dict, default=None*) – Optional configuration for the volume that is associated with the main worker.
- **managers** (*dict, default=None*) – Mapping of workflow step identifier to the worker that is used to execute them.
- **verbose** (*bool, default=True*) – Output run logs if True.

Returns

Return type flowserv.controller.serial.workflow.result.RunResult

openclean_metanome.algorithm.hyfd module

Wrapper to run the HyFD algorithm (A Hybrid Approach to Functional Dependency Discovery) from the Metanome data profiling library. HyFD is a functional dependency discovery algorithm.

Thorsten Papenbrock, Felix Naumann A Hybrid Approach to Functional Dependency Discovery ACM International Conference on Management of Data (SIGMOD '16)

From the abstract: [...] HyFD combines fast approximation techniques with efficient validation techniques in order to findall minimal functional dependencies in a given dataset. While operating on compact data structures, HyFD not only outperforms all existing approaches, it also scales to much larger datasets.

`class` openclean_metanome.algorithm.hyfd.**HyFD**(*max_lhs_size: int = - 1*, *input_row_limit: int = - 1*, *validate_parallel: bool = False*, *memory_guardian: bool = True*, *null_equals_null: bool = True*, *env: Optional[Dict] = None*, *verbose: Optional[bool] = True*)

Bases: `openclean.profiling.constraints.fd.FunctionalDependencyFinder`

HyFD is a hybrid discovery algorithm for functional dependencies. HyFD combines fast approximation techniques with efficient validation techniques in order to findall minimal functional dependencies in a given dataset:

Thorsten Papenbrock, Felix Naumann A Hybrid Approach to Functional Dependency Discovery ACM International Conference on Management of Data (SIGMOD '16)

run(*df: pandas.core.frame.DataFrame*) → List[openclean.profiling.constraints.fd.FunctionalDependency]

Run the HyFD algorithm on the given data frame.

Returns a list of all discovered functional dependencies. If execution of the Metanome algorithm fails a RuntimeError will be raised.

Parameters **df** (*pd.DataFrame*) – Input data frame.

Returns

Return type list of FunctionalDependency

openclean_metanome.algorithm.hyfd.**hyfd**(*df: pandas.core.frame.DataFrame*, *max_lhs_size: int = - 1*, *input_row_limit: int = - 1*, *validate_parallel: bool = False*, *memory_guardian: bool = True*, *null_equals_null: bool = True*, *env: Optional[Dict] = None*, *verbose: Optional[bool] = True*) → List[openclean.profiling.constraints.fd.FunctionalDependency]

Run the HyFD algorithm on a given data frame. HyFD is a hybrid discovery algorithm for functional dependencies.

Parameters

- **df** (*pd.DataFrame*) – Input data frame.
- **max_lhs_size** (*int, default=-1*) – Defines the maximum size of the left-hand-side for discovered FDs. Use -1 to ignore size limits on FDs.
- **input_row_limit** (*int, default=-1*) – Limit the number of rows from the input file that are being used for functional dependency discovery. Use -1 for all columns.
- **validate_parallel** (*bool, default=False*) – If true the algorithm will use multiple threads (one thread per available CPU core).
- **memory_guardian** (*bool, default=True*) – Activate the memory guarding to prevent out of memory errors,

- **null_equals_null** (*bool, default=True*) – Result value when comparing two NULL values.
- **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None
- **verbose** (*bool, default=True*) – Output run logs if True.

Returns

Return type list of FunctionalDependency

openclean_metanome.algorithm.hyfd.**parse_result**(*outputfile: str*, *colmap: Dict*) → List[openclean.profiling.constraints.fd.FunctionalDependency]

Parse the result file of the FD discovery run to generate a list of discovered functional dependencies.

Parameters

- **outputfile** (*string*) – Path to the output file containing the discovered FDs.
- **colmap** (*dict*) – Mapping of column names from surrogate names to column names in the input data frame schema.

Returns

Return type list of FunctionalDependency

openclean_metanome.algorithm.hyucc module

Wrapper to run the HyUCC algorithm (A Hybrid Approach for Efficient Unique Column Combination Discovery) from the Metanome data profiling library. HyUCC is a unique column combination doscovery algorithm.

class openclean_metanome.algorithm.hyucc.**HyUCC**(*max_ucc_size: int = - 1*, *input_row_limit: int = - 1*, *validate_parallel: bool = False*, *memory_guardian: bool = True*, *null_equals_null: bool = True*, *env: Optional[Dict] = None*, *verbose: Optional[bool] = True*)

Bases: openclean.profiling.constraints.ucc.UniqueColumnCombinationFinder

HyUCC is a hybrid discovery algorithm for unique column combinations. The HyUCC algorithm uses the same discovery techniques as the hybrid functional dependency discovery algorithm HyFD. HyUCC discovers all minimal unique column combinationsin a given dataset:

Thorsten Papenbrock and Felix Naumann, A Hybrid Approach for Efficient Unique Column Combination Discovery, Datenbanksysteme fuer Business, Technologie und Web (BTW 2017),

run(*df: pandas.core.frame.DataFrame*) → List[Union[int, str, List[Union[str, int]]]]

Run the HyUCC algorithm on the given data frame. Returns a list of all discovered unique column sets.

If execution of the Metanome algorithm fails a RuntimeError will be raised.

Parameters **df** (*pd.DataFrame*) – Input data frame.

Returns

Return type list of columns

openclean_metanome.algorithm.hyucc.**hyucc**(*df: pandas.core.frame.DataFrame*, *max_ucc_size: int = - 1*, *input_row_limit: int = - 1*, *validate_parallel: bool = False*, *memory_guardian: bool = True*, *null_equals_null: bool = True*, *env: Optional[Dict] = None*, *verbose: Optional[bool] = True*) → List[Union[int, str, List[Union[str, int]]]]

Run the HyUCC algorithm on a given data frame. HyUCC is a hybrid discovery algorithm for unique column

combinations. The algorithm returns a list of discovered column combinations.

Parameters

- **df** (*pd.DataFrame*) – Input data frame.
- **max_ucc_size** (*int, default=-1*) – Defines the maximum size of discovered column sets. Use -1 to return all discovered unique column combinations.
- **input_row_limit** (*int, default=-1*) – Limit the number of rows from the input file that are being used for column combination discovery. Use -1 for all columns.
- **validate_parallel** (*bool, default=False*) – If true the algorithm will use multiple threads (one thread per available CPU core).
- **memory_guardian** (*bool, default=True*) – Activate the memory guarding to prevent out of memory errors,
- **null_equals_null** (*bool, default=True*) – Result value when comparing two NULL values.
- **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None
- **verbose** (*bool, default=True*) – Output run logs if True.

Returns

Return type list of columns

`openclean_metanome.algorithm.hyucc.`**`parse_result`**`(`*outputfile: str*, *colmap: Dict*`)` → List[Union[int, str, List[Union[str, int]]]]

Parse the result file of the UCC discovery run to generate a list of discovered unique column sets.

Parameters

- **outputfile** (*string*) – Path to the output file containing the discovered UCCs.
- **colmap** (*dict*) – Mapping of column names from surrogate names to column names in the input data frame schema.

Returns

Return type list of columns

4.1.2 Submodules

openclean_metanome.config module

Configuration variables and helper methods for running Metanome algorihms in openclean.

`openclean_metanome.config.`**`CONTAINER`**`(`*env: Optional[Dict] = None*`)` → str

Get the identifier of the Metanome container image from the environment variable

Parameters **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None

Returns

Return type string

`openclean_metanome.config.`**`JARFILE`**`(`*env: Optional[Dict] = None*`)` → str

Get path to the Metanome.jar file from the environment.

By default, the jar file is expected to be in the OS-specific user cache directory.

Parameters **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None

Returns

Return type string

openclean_metanome.config.**VOLUME**(*env: Optional[Dict] = None*) → Dict
Get specification for the volume that is associated with the worker that is used to execute the main algorithm step.

Parameters **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None

Returns

Return type dict

openclean_metanome.config.**WORKER**(*env: Optional[Dict] = None*) → Dict
Get specification for the worker that is used to execute the main algorithm step using the metanome wrapper Jar-file.

Parameters **env** (*dict, default=None*) – Optional environment variables that override the system-wide settings, default=None

Returns

Return type dict

openclean_metanome.config.**read_config_obj**(*var: str*, *env: Dict*) → Dict
Read configuration object from a given environment variables.

If the variable is set and contains a dictionary as value that value is returned. Otherwise, it is assumed that the variable references a Json or Yaml file that contains the configuration object.

Parameters

- **var** (*string*) – Name of the environment variable.
- **env** (*dict*) – Dictionary representing the current environment settings.

Returns

Return type dict

openclean_metanome.converter module

Helper functions to prepare inputs and read outputs when running Metanome algorithms on the contents of pandas data frames.

openclean_metanome.converter.**read_json**(*filename: str*) → Union[Dict, List]
Read a JSON object or list from the given output file. By convention, the Java wrapper for Metanome algorithms stores all algorithm as JSON seriaizations.

filename: string Path to the input file on disk.

Returns

Return type dict or list

openclean_metanome.converter.**write_dataframe**(*df: pandas.core.frame.DataFrame*, *filename: str*) → Dict
Write the given data frame to a CSV file. The column names in the resulting CSV file are replaced by unique names (to account for possible duplicate columns in the input data frame).

The created file is a standard CSV file with the default settings for delimiter, quote char and escape char.

Returns the pmapping of unique column names to the original columns in the given data frame.

Parameters **df** (*pd.DataFrame*) – Data frame that is written to disk.

Returns

Return type dict

openclean_metanome.download module

Helper function to download the Metanome.jar file tha is hosted on Zenodo.

`openclean_metanome.download.`**`download_jar`**`(`*dst: Optional[str] = None*, *verbose: Optional[bool] = True*`)`
Download the Metanome.jar file.

The file will be stored at the given destination. If no destination is specified, the file will be stored in the default location as defined by the `config.JARFILE()` method.

The file will only be downloaded if the destination file does not exist.

Parameters

- **dst** (*str, default=None*) – Target pathname for the downloaded file.
- **verbose** (*bool, default=True*) – Print downloaded file target path if True.

openclean_metanome.tests module

Helper funcitons for unit testing.

`openclean_metanome.tests.`**`input_output`**`(`*rundir: str*, *cmd: str*`)` → Tuple[str, str]
Extract name of input file and output file from commands that are used to run the Metanome algorithms from the command line.

Returns the path to the input and output files.

Parameters

- **rundir** (*string*) – Path to the run directory. The input and output file references in the command string are relative to this directory.
- **cmd** (*string*) – Command to run the Metanome algorithms from the command line.

Returns

Return type tuple of (string, string)

openclean_metanome.version module

Version information for the openclean metanome wrapper package.

PYTHON MODULE INDEX

O

INDEX