labutils Documentation

Release 0.1.0

University of Waterloo's NetLab

Jul 14, 2017

Contents

CHAPTER 1

Overview

`labutils` is a collection of utilities intended to be used by NetLab developers and researchers. `labutils` is a place to share useful Python code amongst NetLab members. Contributions may include miscellaneous helper functions, developer feature prototypes, or project-specific workhorse functions.

CHAPTER 2

Installing

First, clone the `labutils` GitHub repository. `labutils` can then be installed by running `python setup.py develop` from inside the main directory.

CHAPTER 3

labutils API Reference

Miscellaneous

labutils.**hello_world**()
Instantly gratifying reward for installing labutils. :return: None

labutils.**new_identifier_name**(*base*, *names*, *sep='_'*)
For finding an unused identifier name.

Examples:

- 'sum' is taken, so 'sum2' is used.
- 'sum', 'sum2', and 'sum3' are taken, so 'sum4' is used.

Parameters **base** – A base string e.g.

Returns

class labutils.**bcolors**
Use this class to conveniently add colours and formatting to printed output. Taken from https://stackoverflow.com/questions/287871/print-in-terminal-with-colors-using-python. Example:

```
print(bcolors.WARNING + "Warning: No active frommets remain. Continue?" +␣
↪bcolors.ENDC)
```

Data Fusion

This set of functions is a prototype data fusion workflow for use with recordlinkage.

labutils.**rank_pairs**(*comp*, *by*, *method='cols'*, *ascending=False*)
rank_pairs sorts pairs from a recordlinkage.Compare object, based on computed comparison values.

Available methods:

- 'cols': sort by first, column, ties broken by subsequent columns. See pandas.DataFrame.sort_values.
- 'sum': sort by the sum of a list of columns.
- 'avg': sort by the mean of a list of columns.

Parameters

- **comp** (*recordlinkage.Compare*) – A populated Comparison object.
- **by** (*list*) – A list of column name strings to sort on by "method".
- **method** (*str*) – A the method to sort by (see above).
- **ascending** (*bool*) – Specifies ordering of sorted rows.

Returns recordlinkage.Compare

labutils.**refine_mapping**(*comp*, *left_unique=True*, *right_unique=True*)

Removes pairs that violate uniqueness rules. Matches may be one-to-one (default), one-to-many (right_unique=False), or many-to-one (left_unique=False). refine_mapping always keeps the first instance of an index. To keep the best matches, sort (or filter) pairs before passing to refine_mapping, e.g. with rank_pairs (or a classification algorithm).

Parameters

- **comp** (*recordlinkage.Compare*) – A populated Comparison object.
- **left_unique** (*bool*) – Specifies uniqueness of left (top-level) indices.
- **right_unique** (*bool*) – Specifies uniqueness of right (top-level) indices.

Returns recordlinkage.Compare

labutils.**fast_fuse**(*comp*, *left_suffix='_l'*, *right_suffix='_r'*)

Performs data fusion using a recordlinkage.Compare object. All data is kept from both original data frames, renaming columns to avoid conflits. The result is comp.vectors but with each rows populated with data from the original two data frames corresponding to the compared pair.

Parameters

- **comp** (*recordlinkage.Compare*) – Compared pairs to be fused.
- **left_suffix** (*str*) – The suffix stem to be used to resolve naming conflits for columns in df_a.
- **right_suffix** (*str*) – The suffix stem to be used to resolve naming conflits for columns in df_b.

Returns pandas.DataFrame

Feature Comparison

This set of functions implement prototype comparison methods for use with recordlinkage.

labutils.**lcss**(*s1*, *s2*)

A custom comparison function to be used with the Compare.compare() method within recordlinkage. This is used to compare two strings, computing a match score based on the length of the longest common substring between the two strings.

Parameters

- **pandas.Series) s1** ((*label,*) – Series or DataFrame to compare all fields.

- **pandas.Series) s2** (*(label,)* – Series or DataFrame to compare all fields.

Returns pandas.Series of integers (the length of the substring).

labutils.**normed_lcss**(*s1*, *s2*)

A custom comparison function to be used with the Compare.compare() method within recordlinkage. This is used to compare two strings, computing a match score based on the length of the longest common substring between the two strings.

The score resulting from the comparison can be expressed as the length of the longest common substring, divided by the length of the shorter string. The resulting score is equal or between 0 and 1.

Parameters

- **pandas.Series) s1** (*(label,)* – Series or DataFrame to compare all fields.
- **pandas.Series) s2** (*(label,)* – Series or DataFrame to compare all fields.

Returns pandas.Series with similarity values equal or between 0 and 1.

labutils.**fuzzy_lcss**(*s1*, *s2*, *match=1*, *mismatch=-0.5*, *gap=-1*)

A custom comparison function to be used with the Compare.compare() method within recordlinkage. This is used to compare two strings, computing a match score based on the length of the longest similar substring between the two strings.

The score resulting from the comparison is computed using a dynamic programming algorithm that creates a score based on the existence of character matches, mismatches, and gaps. This algorithm is equivalent to the Smith-Waterman algorithm used in the field of bioinformatics. To learn more about this algorithm see: https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm.

In this method, the raw numeric score is produced.

Parameters

- **pandas.Series) s1** (*(label,)* – Series or DataFrame to compare all fields.
- **pandas.Series) s2** (*(label,)* – Series or DataFrame to compare all fields.
- **match** (*float*) – Value added to score for matching characters.
- **mismatch** (*float*) – Value added to score for mismatching characters.
- **gap** (*float*) – Value added to score for gaps between similar characters.

Returns pandas.Series with numeric similarity values.

labutils.**normed_fuzzy_lcss**(*s1*, *s2*, *match=1*, *mismatch=-0.5*, *gap=-1*)

A custom comparison function to be used with the Compare.compare() method within recordlinkage. This is used to compare two strings, computing a match score based on the length of the longest similar substring between the two strings.

The score resulting from the comparison is computed using a dynamic programming algorithm that creates a score based on the existence of character matches, mismatches, and gaps. This algorithm is equivalent to the Smith-Waterman algorithm used in the field of bioinformatics. To learn more about this algorithm see: https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm.

In this method, the final match score is normalized by dividing the observed score by the maximum possible score (i.e. the length of the shorter string multiplied by the "match" parameter).

Parameters

- **pandas.Series) s1** (*(label,)* – Series or DataFrame to compare all fields.
- **pandas.Series) s2** (*(label,)* – Series or DataFrame to compare all fields.
- **match** (*float*) – Value added to score for matching characters.

- **mismatch** (*float*) – Value added to score for mismatching characters.
- **gap** (*float*) – Value added to score for gaps between similar characters.

Returns pandas.Series with similarity values equal or between 0 and 1.

labutils.**compare_in**(*s1*, *s2*)

Parameters

- **s1** (*(pandas.Series)*) –
- **s2** (*(pandas.Series)*) –

Returns

labutils.**compare_lists**(*s1*, *s2*)

A custom comparison function to be used with the Compare.compare() method within recordlinkage. This is used to compare two lists, computing a match score based on the number of items shared between two lists.

The score resulting from the comparison can be expressed as the number of shared items between two sets, divided by the total number of unique items in the smaller set.

Parameters

- **pandas.Series) s1** (*(label,)* – Series or DataFrame to compare all fields.
- **pandas.Series) s2** (*(label,)* – Series or DataFrame to compare all fields.

Returns pandas.Series with similarity values equal or between 0 and 1.

Pandas Utilities

labutils.**clip_df**(*df*, *tablefmt='html'*)

Copy a dataframe as plain text to your clipboard. Probably only works on Mac. For format types see tabulate package documentation.

Parameters

- **df** (*pandas.DataFrame*) – Input DataFrame.
- **tablefmt** (*str*) – What type of table?

Returns None.

labutils.**expand_on**(*df*, *col1*, *col2*, *rename1=None*, *rename2=None*, *drop=[]*, *drop_collections=False*)

Returns a reshaped version of extractor's data, where unique combinations of values from col1 and col2 are given individual rows. This method was pasted form tidyextractors on 2017-07-10.

Example function call from tidymbox:

```
self.expand_on(my_df, 'From', 'To', ['MessageID', 'Recipient'], rename1='From',␣
↪rename2='Recipient')
```

Columns to be expanded upon should be either atomic values or dictionaries of dictionaries. For example:

Input Data:

col1 (Atomic)	col2 (Dict of Dict)
value1	{valueA : {attr1: X1, attr2: Y1}, valueB: {attr1: X2, attr2: Y2}
value2	{valueC : {attr1: X3, attr2: Y3}, valueD: {attr1: X4, attr2: Y4}

Output Data:

col1_extended	col2_extended	attr1	attr2
value1	valueA	X1	Y1
value1	valueB	X2	Y2
value2	valueA	X3	Y3
value2	valueB	X4	Y4

Parameters

- **df** (*pandas.DataFrame*) – Input DataFrame.
- **col1** (*str*) – The first column to expand on. May be an atomic value, or a dict of dict.
- **col2** (*str*) – The second column to expand on. May be an atomic value, or a dict of dict.
- **rename1** (*str*) – The name for col1 after expansion. Defaults to col1_extended.
- **rename2** (*str*) – The name for col2 after expansion. Defaults to col2_extended.
- **drop** (*list*) – Column names to be dropped from output.
- **drop_collections** (*bool*) – Should columns with compound values be dropped?

Returns pandas.DataFrame

labutils.**drop_collection_columns**(*df*)

Drops columns containing collections (i.e. sets, dicts, lists) from a DataFrame. This method was pasted from tidyextractors on 2017-07-10.

Parameters **df** (*pandas.DataFrame*) – Input data.

Returns pandas.DataFrame

labutils.**col_type_set**(*col*, *df*)

Determines the set of types present in a DataFrame column. This function was pasted from tidyextractors on 2017-07-10.

Parameters

- **col** (*str*) – A column name.
- **df** (*pandas.DataFrame*) – Input data.

Returns A set of Types.

CHAPTER 4

Contributing

This guide is meant for members of NetLab who want to add their code to `labutils`. If to contribute to `labutils`, do the following:

- Make sure you're a "collaborator" on the GitHub repo.
- Pull the latest version of `labutils`.
- Write and document your code. (See existing functions for examples of how to document functions. Documentation is important so that other people know how to use your function. Any functions with documentation string will automatically be included in the ReadTheDocs page.)
- Add your code to the appropriate `.py` file. If your code doesn't make sense in any of the existing files, add it to `misc.py` or start a new `.py` file.
- If you create a new file, add it to `__init__.py`.
- Add any functions, classes, submodules, etc. to `./docs/source/api_ref.rst`.
- Push your changes! Anyone who pulls the latest version will be able to import your code. Documented functions will be automatically added to the API documentation.

Python Module Index

l

Index