callgraph Documentation

Release 0.1.0

Oliver Steele

Jan 08, 2018

Contents:

Compute the minimal unique keys for a collection of strings or sequences.

This is intended for use in presenting data in a user interface. For example, the `nbcollate` command-line interface from the nbcollate package uses it to guess student names from notebook filenames, for use in the creation of the collated Jupyter notebook.

The minimal keys from `["assignments/alice/hw1.txt", "assignments/bob/hw1.txt"]` are `["alice", "bob"]`, because the input strings share the common prefix `"assignments/"` and the common suffix `".txt"`.

The minimal keys from `["alice/assignments/hw1.txt", "bob/assignments/hw1.txt"]` are also `["alice", "bob"]`, because the input strings share the common suffix `"/assignments/hw1.txt"`.

Finally, the minimal keys from `["assignments/alice.txt", "assignments/bob.txt"]` are—wait for it—`["alice", "bob"]`.

There are options to ignore case, and to split the strings at different boundaries. (The default is split to on words, so that `["alice", "adam"]` is not abbreviated to `["lice", "dam"]`.)

This is the same basic idea as a database superkey, except that the actual minimal unique keys are returned, instead of the attributes that select these keys.

The current implementation trims only the beginnings and ends of sequences, because this is all that I've needed so far. I have in my head a more sophisticated implementation that uses *difflib*, but it is too long to fit in the header of this README.

CHAPTER 1

Install

```
$ pip install minimalkeys
```

CHAPTER 2

Usage

```
>>> from minimalkeys import minimal_keys
>>> minimal_keys(["assignments/alice/hw1.txt", "assignments/bob/hw1.txt"])
['alice', 'bob']
```

For more examples, see the API documentation.

CHAPTER 3

License

MIT

CHAPTER 4

API

Compute the minimal unique keys for a collection of strings or sequences.

minimalkeys.**minimal_keys**(*keys*, *, *key=None*, *split='(\\w+)'*)

Compute the minimal unique keys for a collection of string.

Parameters

- **items** (*[str]*) – A sequence of keys.
- **key** (*function*) – A function of one argument that normalizes substrings before they are tested for equality.
- **split** (*str*) – A regular expression string that is used to split the string into the substrings that are candidates for removal. It should capture characters that should be included in the resulting minimal keys.

Returns A sequence of keys.

Return type str

Examples

```
>>> minimal_keys(['assignments/alice/hw1.txt', 'assignments/bob/hw1.txt'])
['alice', 'bob']
>>> minimal_keys(['alice/assignments/hw1.txt', 'bob/assignments/hw1.txt'])
['alice', 'bob']
```

Use `key` to normalize components before comparing them:

```
>>> minimal_keys(['assignments/alice.txt', 'Assignments/bob.txt'])
['assignments/alice', 'Assignments/bob']
>>> minimal_keys(['assignments/alice.txt', 'Assignments/bob.txt'], key=str.
↪upper)
['alice', 'bob']
```

Use `split` to specify the boundaries for common substring removal:

```
>>> minimal_keys(['assignments/alice_smith.txt', 'assignments/alice_jones.txt
↪'])
['alice_smith', 'alice_jones']
>>> minimal_keys(['assignments/alice_smith.txt', 'assignments/alice_jones.txt
↪'],                          split=r'([a-z]+)')
['smith', 'jones']
```

minimalkeys.**minimal_seq_keys**(*keys*, *key=None*)

Compute the minimal unique keys for a collection of sequences.

Parameters

- **items** (*[seq]*) – A sequence of keys. Each key is itself a sequence.
- **key** (*function*) – A function of one argument that normalizes substrings before they are tested for equality.

Returns A sequence of keys.

Return type seq

Python Module Index

m

Index

M