Clashgap

NioGreek

Jul 29, 2021

GETTING STARTED

Clashgap is a diff/compression module in Python

CHAPTER

ONE

HOW IT WORKS

In case if you have two strings:

> “This is a sentence…” and “This is a word…”

you could “clash” both of them together and find their gap, to get an array loking something like:

> [“This is a”, [“sentence”, “word”], “…”]

As you can the clashgap algorithm looks for collisions in the two strings to find the gap. The clashgaped string maybe used for compression or as the diff of the input strings

CHAPTER

TWO

GETTING STARTED

To start using the package you would have to install it. Have a look at the Installation guide. After that out of the way, give it a try by following the instructions at the Give it a try guide

- *Installing Clashgap*
- *Give it a try*

2.1 Installation

The installation is simple and obvious. Assuming you have Python and pip installed, enter the following into the command line

```
$pip install clashgap
```

This will install clashgap and all it's dependencies. After installation, you may want to *give it a try*

2.2 Give it a try

This page assumes that you have already *installed clashgap*

Open a compatible Python shell and follow along

```
>>> import clashgap as cg

>>> cg.gap(["spam", "ham"])
[['sp', 'h'], 'am']

>>> cg.fill([['sp', 'h'], 'am'])
["spam", "ham"]
```

You have successfully demonstrated the usage of the *gap* and *fill* functions, which are the most fundamental function in the clashgap module. Also chack out other features in the module like the *Clash class*.

CHAPTER

THREE

FUNDAMENTAL FUNCTIONS

The fundamental functions of the package is `gap()` and `fill()`

3.1 The fill() function

The fill function is used to combine a gap object, to get the clash. It accepts a list representing the gap as parameter, and returns another list representing the clash

```
fill(clash: list) -> list
```

Here is a demonstration of it's usage

```
import clasgap
gap = clashgap.fill([["sp", "h"], "am"])
print(gap)
```

stdout:

```
["spam", "ham"]
```

The gap() and fill() functions are inverse of each other. So `gap(fill(x)) == x` and `fill(gap(y)) == y` are both True, for any valid `x` and `y`

Note: The fill function currently only supports input list of length two

3.2 The gap() functions

The gap function is used to combine a list with two elements into a gap object. It accepts a list with two elements as parameter, and returns another list representing the gap of the input list

```
gap(clash: list) -> list
```

Here is a demonstration of it's usage

```
import clasgap
gap = clashgap.gap(["spam", "ham"])
print(gap)
```

stdout:

```
[["sp", "h"], "am"]]
```

The gap() and fill() functions are inverse of each other. So `fill(gap(x)) == x` and `gap(fill(y)) == y` are both True, for any valid `x` and `y`

Note: The gap function currently only supports input list of length two

CHAPTER

FOUR

ADVANCED FEATURES

You could also check out thre Clash class, which enables you instantiate a Clash object, with the input array.

- *The Clash class*

4.1 Using the Clash class

Clash is a name given to the combination of multiple strings which is used to find the gap of the strings. Think it of a object between gap and fill

You could make use of the Clash class implemented in the clashgap module to Clash two strings and find it's gap and fill. The Clash can be initialized by passing the input strings as a list. Here is a demonstration

```
clash = Clash(["spam", "ham"])
```

Note: The Clash class currently only supports input list of length two

4.1.1 Constituent Methods

Name	Description
gap()	use this method to find the gap of the clash
fill()	this methods returns the filled gap

gap

Clash.gap() method returns the gap of the clash as a list

```
def gap(self: Clash) -> list
```

Here is a demonstration on how you can call the function

```
>>> clash = Clash(["spam", "ham"])
>>> clash.gap()
[["sp", "h"], "am"]
```

fill

Clash.fill() method returns the clash itself as a list

```
def fill(self: Clash) -> list
```

Here is a demonstration on how you can call the function

```
>>> clash = Clash(["spam", "ham"])
>>> clash.fill()
["spam", "ham"]
```

As you see, the fill() just returns the clash as a list. `Clash(x).fill() == x` is True for any valid `x`

4.1.2 Using in-built function on Clash objects

In-built functions	Description
str()	str(clash) returns the gap of Clash as a string
repr()	repr(clash) returns the gap of Clash as a string

__str__

```
def __str__(self: Clash) -> str
```

The __str__ is a magic method used to define the behaviour on using the in-built str() function on a Clash object. Passing a Clash object on the str function will return the gap of the Clash object as a string

Here is a demonstration

```
>>> clash = Clash(["spam", "ham"])
>>> str(clash)
"[['sp', 'h'], 'am']"
>>> type(str(clash))
<class 'str'>
```

As per the demonstration, `str(x) == str(x.gap())` is True, for any valid Clash `x`

__repr__

```
def __repr__(self: Clash) -> str
```

The __repr__ is a magic method used to define the behaviour on using the in-built repr() function on a Clash object. Passing a Clash object on the repr function will return the gap of the Clash object as a string

Here is a demonstration

```
>>> clash = Clash(["spam", "ham"])
>>> repr(clash)
"[['sp', 'h'], 'am']"
>>> type(repr(clash))
<class 'str'>
```

As per the demonstration, `repr(x) == str(x.gap())` is True, for any valid Clash `x`

CHAPTER

FIVE

ABOUT CLASHGAP

To find more about the project, you may head on to any of these links

- PyPI project page
- GitHub repo
- *Changelog*
- *All Links*

5.1 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

5.1.1 Unreleased

5.1.2 v1.0.0: The fill() function - 2021-07-29

Added

- Introducing CHANGELOG
- You can now find the Changelog link on the PyPI project page
- Implemented fill() function
- Defined __all__; Now you can use `from clashgap import *`

Changed

- Internal code quality improvements and standardization

5.1.3 v0.2.0: Quality Rollup 0x00 - 2021-07-23

Added

- Now you can use the str() function for clash objects
- You can now find Issue Tracker link on the PyPI project page

Changed

- Several Internal code quality improvements and standardization

5.1.4 v0.1.0: The Birth! - 2021-07-22

- Initial release

5.2 Project Links

Links	Description
Repository	The Source Code of the module lies here
Documentation	You are here
Changelog	All notable changes to this project are documented here
Bug Tracker	Bug-reports and feature-requests goes here
PyPI project page	This is where the project is hosted
LGTM analysis Page	lgtm analysis our source and makes sure the quality of the code is at it's best