web-monitoring-diff

Release 0.1.4.post3+g665eeae

Environmental Data & Governance Initiative

May 01, 2024

CONTENTS

Web-monitoring-diff is a suite of functions that *diff* (find the differences between) types of content commonly found on the web, such as HTML, text files, etc. in a variety of ways. It also includes an optional web server that generates diffs as an HTTP service.

This package was originally built as a component of EDGI's Web Monitoring Project, but is also used by other organizations and tools.

CHAPTER

ONE

INSTALLATION

web-monitoring-diff requires **Python 3.7 or newer**. Before anything else, make sure you're using a supported version of Python. If you need to support different local versions of Python on your computer, we recommend using pyenv or Conda.

1. **System-level dependencies:** web-monitoring-diff depends on several system-level, non-Python libraries that you may need to install first. Specifically, you'll need: `libxml2`, `libxslt`, `openssl`, and `libcurl`.

   On MacOS, we recommend installing these with Homebrew:

   ```
   brew install libxml2
   brew install libxslt
   brew install openssl
   # libcurl is built-in, so you generally don't need to install it
   ```

   On Debian Linux, use `apt`:

   ```
   apt-get install libxml2-dev libxslt-dev libssl-dev openssl libcurl4-openssl-dev
   ```

   Other systems may have different package managers or names for the packages, so you may need to look them up.

2. **Install this package** with *pip*. Be sure to include the `--no-binary lxml` option:

   ```
   pip install web-monitoring-diff --no-binary lxml
   ```

   Or, to also install the web server for generating diffs on demand, install the `server` extras:

   ```
   pip install web-monitoring-diff[server] --no-binary lxml
   ```

   The `--no-binary` flag ensures that pip downloads and builds a fresh copy of `lxml` (one of web-monitoring-diff's dependencies) rather than using a pre-built version. It's slower to install, but is required for all the dependencies to work correctly together. **If you publish a package that depends on web-monitoring-diff, your package will need to be installed with this flag, too.**

   On MacOS, you may need additional configuration to get `pycurl` to use the Homebrew `openssl`. Try the following:

   ```
   PYCURL_SSL_LIBRARY=openssl \
     LDFLAGS="-L/usr/local/opt/openssl/lib" \
     CPPFLAGS="-I/usr/local/opt/openssl/include" \
     pip install web-monitoring-diff --no-binary lxml --no-cache-dir
   ```

The `--no-cache-dir` flag tells *pip* to re-build the dependencies instead of using versions it's built already. If you tried to install once before but had problems with `pycurl`, this will make sure pip actually builds it again instead of re-using the version it built last time around.

For local development, clone the git repository and then make sure to do an editable installation instead.

```
pip install .[server,dev] --no-binary lxml
```

3. **(Optional) Install experimental diffs.** Some additional types of diffs are considered "experimental" — they may be new and still have lots of edge cases, may not be publicly available via PyPI or another package server, or may have any number of other issues. To install them, run:

```
pip install -r requirements-experimental.txt
```

CHAPTER

TWO

BASIC USAGE

web-monitoring-diff can imported as a library that provides diffing functions for use in your own Python code, or it can be run as a standalone web server.

2.1 Library Usage

Import *web_monitoring_diff*, then call a diff function:

```
import web_monitoring_diff

page1 = "<!doctype html>\n<html><body>This is page 1.</body></html>"
page2 = "<!doctype html>\n<html><body>This is page 2.</body></html>"
comparison = web_monitoring_diff.html_diff_render(page1, page2)
```

2.2 Web Server

Start the web server:

```
$ web-monitoring-diff-server
```

This starts the web server on port *8888*.

Then use cURL, a web browser, or any other HTTP tools to get a list of supported diff types:

```
$ curl "http://localhost:8888/"
```

That should output some JSON like:

```
{"diff_types": ["length", "identical_bytes", "side_by_side_text", "links", "links_json",
↪"html_text_dmp", "html_source_dmp", "html_token", "html_tree", "html_perma_cc", "links_
↪diff", "html_text_diff", "html_source_diff", "html_visual_diff", "html_tree_diff",
↪"html_differ"], "version": "0.1.0"}
```

You can use each of these diff types by requesting the URL:

```
http://localhost:8888/<diff_type>?a=<url_to_left_side_of_comparison>&b=<url_to_right_
↪side_of_comparison>
```

For example, to compare how the links on the National Renewable Energy Laboratory's "About" page changed between 2018 and 2020 using data from the Internet Archive:

```
# URL of a version of the page archived in 2018:
$ VERSION_2018='http://web.archive.org/web/20180918073921id_/https://www.nrel.gov/about/'
# URL of a version of the page archived in 2020:
$ VERSION_2020='http://web.archive.org/web/20201006001420id_/https://www.nrel.gov/about/'
# Use the `links_json` diff to compare the page's links and output as JSON:
$ curl "http://localhost:8888/links_json?a=${VERSION_2018}&b=${VERSION_2020}"
```

If you have `jq` installed, you might want to use it to format the result in a nicer way:

```
$ curl "http://localhost:8888/links_json?a=${VERSION_2018}&b=${VERSION_2020}" | jq
```

You can pass additional arguments to the various diffs in the query string. See the full documentation of the server and of the various diffs for more details.

CHAPTER

THREE

API REFERENCE

3.1 Diff Types

web-monitoring-diff provides a variety of diff algorithms for use in comparing web content. They all follow a similar standardized signature and return format.

Diff Signatures

All diffs should have parameters named `a_<body|text>` and `b_<body|text>` as their first two arguments. These represent the two pieces of content to compare, where `a` represents the "from" or left-hand side and `b` represents the "to" or right-hand side of the comparison. The name indicates whether the function takes bytes (`a_body`/`b_body`) or a decoded string (`a_text`/`b_text`). The web server inspects argument names to determine what to pass to a given diff type.

Additionally, diffs may take several other standardized parameters:

- `a_body`, `b_body`: Raw HTTP reponse body (bytes), described above.
- `a_text`, `b_text`: Decoded text of HTTP response body (str), described above.
- `a_url`, `b_url`: URL at which the content being diffed is found. (This is useful when content contains location-relative information, like links.)
- `a_headers`, `b_headers`: Dict of HTTP headers.

Finally, some diffs take additional, diff-specific parameters.

Return Values

All diffs return a `dict` with a key named `"diff"`. The value of this dict entry varies by diff type, but is usually:

- An array of changes. Each entry will be a 2-tuple, where the first item is an `int` reprenting the type of change (`-1` for removal, `0` for unchanged, `1` for addition, or other numbers for diff-specific meanings) and the second item is the data or string that was added/removed/unchanged.
- A string representing a custom view of the diff, e.g. an HTML document.
- A bytestring representing a custom binary view of the diff, e.g. an image.

Each diff may add additional, diff-specifc keys to the dict. For example, *web_monitoring_diff.html_diff_render()* includes a `"change_count"` key indicating how many changes there were, since it's tough to inspect the HTML of the resulting diff and count yourself.

web_monitoring_diff.**compare_length**(*a_body*, *b_body*)

Compute difference in response body lengths. (Does not compare contents.)

web_monitoring_diff.**identical_bytes**(*a_body*, *b_body*)

Compute whether response bodies are exactly identical.

web_monitoring_diff.**side_by_side_text**(*a_text*, *b_text*)

Extract the visible text from both response bodies.

web_monitoring_diff.**html_text_diff**(*a_text*, *b_text*)

Diff the visible textual content of an HTML document.

Examples

```
>>> html_text_diff('<p>Deleted</p><p>Unchanged</p>',
...                '<p>Added</p><p>Unchanged</p>')
[[-1, 'Delet'], [1, 'Add'], [0, 'ed Unchanged']]
```

web_monitoring_diff.**html_source_diff**(*a_text*, *b_text*)

Diff the full source code of an HTML document.

Examples

```
>>> html_source_diff('<p>Deleted</p><p>Unchanged</p>',
...                  '<p>Added</p><p>Unchanged</p>')
[[0, '<p>'], [-1, 'Delet'], [1, 'Add'], [0, 'ed</p><p>Unchanged</p>']]
```

web_monitoring_diff.**links_diff**(*a_text*, *b_text*, *a_headers=None*, *b_headers=None*, *content_type_options='normal'*)

Extracts all the outgoing links from a page and produces a diff of an HTML document that is simply a list of the text and URL of those links.

It ignores links that merely navigate within the page.

NOTE: this diff currently suffers from the fact that our diff server does not know the original URL of the content, so it can identify:

```
>>> <a href="#anchor-in-this-page">Text</a>
```

as an internal link, but not:

```
>>> <a href="http://this.domain.com/this/page#anchor-in-this-page">Text</a>
```

web_monitoring_diff.**links_diff_json**(*a_text*, *b_text*, *a_headers=None*, *b_headers=None*, *content_type_options='normal'*)

Generate a diff of all outgoing links (see *links_diff()*) where the *diff* property is formatted as a list of change codes and values.

web_monitoring_diff.**links_diff_html**(*a_text*, *b_text*, *a_headers=None*, *b_headers=None*, *content_type_options='normal'*)

Generate a diff of all outgoing links (see *links_diff()*) where the *diff* property is an HTML string. Note the actual return type is still JSON.

web_monitoring_diff.**html_diff_render**(*a_text*, *b_text*, *a_headers=None*, *b_headers=None*, *include='combined'*, *content_type_options='normal'*, *url_rules='jsessionid'*)

HTML Diff for rendering. This is focused on visually highlighting portions of a page's text that have been changed. It does not do much to show how node types or attributes have been modified (save for link or image URLs).

The overall page returned primarily represents the structure of the "new" or "B" version. However, it contains some useful metadata in the *<head>*:

1. A *<template id="wm-diff-old-head">* contains the contents of the "old" or "A" version's *<head>*.
2. A *<style id="wm-diff-style">* contains styling diff-specific styling.
3. A *<meta name="wm-diff-title" content="[diff]">* contains a renderable HTML diff of the page's *<title>*. For example:

   The ~~old~~<ins>new</ins> title

NOTE: you may want to be careful with rendering this response as-is; inline *<script>* and *<style>* elements may be included twice if they had changes, which could have undesirable runtime effects.

Parameters

a_text
[str] Source HTML of one document to compare

b_text
[str] Source HTML of the other document to compare

a_headers
[dict] Any HTTP headers associated with the *a* document

b_headers
[dict] Any HTTP headers associated with the *b* document

include
[str] Which comparisons to include in output. Options are:

- *combined* returns an HTML document with insertions and deletions together.
- *insertions* returns an HTML document with only the unchanged text and text inserted in the *b* document.
- *deletions* returns an HTML document with only the unchanged text and text that was deleted from the *a* document.
- *all* returns all of the above documents. You might use this for efficiency – the most expensive part of the diff is only performed once and reused for all three return types.

content_type_options
[str] Change how content type detection is handled. It doesn't make a lot of sense to apply an HTML-focused diffing algorithm to, say, a JPEG image, so this function uses a combination of headers and content sniffing to determine whether a document is not HTML (it's lenient; if it's not pretty clear that it's not HTML, it'll try and diff). Options are:

- *normal* uses the *Content-Type* header and then falls back to sniffing to determine content type.
- *nocheck* ignores the *Content-Type* header but still sniffs.
- *nosniff* uses the *Content-Type* header but does not sniff.
- *ignore* doesn't do any checking at all.

url_rules
[str] Use specialized rules for comparing URLs in links, images, etc. Possible values are:

- *jsessionid* ignores Java Servlet session IDs in URLs.
- *wayback* considers two Wayback Machine links as equivalent if they have the same original URL, regardless of each of their timestamps.

- *wayback_uk* like *wayback*, but for the UK Web Archive (webarchive.org.uk)

You can also combine multiple comparison rules with a comma, e.g. *jsessionid,wayback*. Use None or an empty string for exact comparisons. (Default: *jsessionid*)

Examples

```
>>> text1 = '<!DOCTYPE html><html><head></head><body><p>Paragraph</p></body></html>'
... text2 = '<!DOCTYPE html><html><head></head><body><h1>Header</h1></body></html>'
... test_diff_render = html_diff_render(text1,text2)
```

3.1.1 Experimental External Diffs

The functions in `web_monitoring_diff.experimental` wrap diff algorithms available from other repositories that we consider relatively experimental or unproven. They may be new and still have lots of edge cases, may not be publicly available via PyPI or another package server, or may have any number of other issues.

They are not installed by default, so calling them may raise an exception. To install them, use pip:

```
$ pip install -r requirements-experimental.txt
```

Experimental modules are typically named by the package they wrap, and can be called with a function named `diff`. For example:

```
>>> from web_monitoring_diff.experimental import htmldiffer
>>> htmldiffer.diff("<some>html</some>", "<some>other html</some>")
```

web_monitoring_diff.experimental.htmldiffer.**diff**(*a_text*, *b_text*)

Wraps the `htmldiffer` package with the standard arguments and output format used by all diffs in `web-monitoring-diff`.

`htmldiffer` is mainly developed as part of Perma CC (https://perma.cc/), a web archival service, and the Harvard Library Innovation Lab. At a high level, it parses the text and tags of a page into one list and uses Python's built-in `difflib.SequenceMatcher` to compare them. This contrasts with `web_monitoring_diff.html_render_diff`, where it is primarily the *text* of the page being diffed, with additional content from from the surrounding tags added in as appropriate (tags there are still kept in order to rebuild the page structure after diffing the text).

While `htmldiffer` is available on PyPI, the public release hasn't been updated in quite some time. Its authors recommend installing via git instead of PyPI:

```
$ pip install git+https://github.com/anastasia/htmldiffer@develop
```

You can also install all experimental differs with:

```
$ pip install -r requirements-experimental.txt
```

NOTE: this differ parses HTML in pure Python and can be very slow when using the standard, CPython interpreter. If you plan to use it in a production or performance-sensitive environment, consider using PyPy or another, more optimized interpreter.

Parameters

a_text
[`str`] Source HTML of one document to compare

b_text
[str] Source HTML of the other document to compare

Returns

dict

web_monitoring_diff.experimental.htmltreediff.**diff**(*a_text*, *b_text*)

Wraps the `htmltreediff` package with the standard arguments and output format used by all diffs in `web-monitoring-diff`.

`htmltreediff` parses HTML documents into an XML DOM and attempts to diff the document *structures*, rather than look at streams of tags & text (like `htmldiffer`) or the readable text content of the HTML (like `web_monitoring_diff.html_render_diff`). Because of this, it can give extremely accurate and detailed information for documents that are very similar, but its output gets complicated or opaque as the two documents diverge in structure. It can also be very slow.

In practice, we've found that many real-world web pages vary their structure enough (over periods as short as a few months) to reduce the value of this diff. It's best used for narrowly-defined scenarios like:

- Comparing versions of a page that are very similar, often at very close points in time.
- Comparing XML structures you can expect to be very similar, like XML API responses, RSS documents, etc.
- Comparing two documents that were generated from the same template with differing underlying data. (Assuming the template is fairly rigid, and does not leave too much document structure up to the underlying data.)

`htmltreediff` is no longer under active development; we maintain a fork with minimal fixes and Python 3 support. It is not available on PyPI, so you must install via git:

```
$ pip install git+https://github.com/danielballan/htmltreediff@customize
```

You can also install all experimental differs with:

```
$ pip install -r requirements-experimental.txt
```

Parameters

a_text
[str] Source HTML of one document to compare

b_text
[str] Source HTML of the other document to compare

Returns

dict

3.2 Web Server

web_monitoring_diff.server.**make_app**()

Create and return a Tornado application object that serves diffs.

web_monitoring_diff.server.**cli**()

Start the diff server from the CLI. This will parse the current process's arguments, start an event loop, and begin serving.

3.3 Exception Classes

class web_monitoring_diff.exceptions.**UndecodableContentError**

Raised when the content downloaded for diffing could not be decoded.

class web_monitoring_diff.exceptions.**UndiffableContentError**

Raised when the content provided to a differ is incompatible with the diff algorithm. For example, if a PDF was provided to an HTML differ.

CHAPTER

FOUR

RELEASE HISTORY

4.1 In Development

n/a

4.2 Version 0.1.4 (2024-01-01)

This is a minor release that updates some of the lower-level parsing and diffing tools this package relies on:

- Updates the diff-match-patch implementation we rely on for simple text diffs to fast_diff_match_patch v2.x. (Issue #126)
- Fix misconfigured dependency requirements for html5-parser. This should have no user impact, since there are no releases (yet) in the version range we were accidentally allowing for. (Issue #126)
- Support lxml v5.x. (Issue #163)

4.3 Version 0.1.3 (2022-04-18)

This releases fixes some minor issues around content-type checking for HTML-related diffs (`html_diff_render` and `links_diff`). Both lean towards making content-type checking more lenient; our goal is to stop wasted diffing effort early *when we know it's not HTML,* not to only diff things are definitely HTML:

- Ignore invalid `Content-Type` headers. These happen fairly frequently in the wild — especially on HTML pages — and we now ignore them instead of treating them as implying the content is not HTML. (Issue #76)
- Ignore the `application/x-download` content type. This content-type isn't really about the content, but is frequently used to make a browser download a file rather than display it inline. It no longer affects parsing or diffing. (Issue #105)

This release also adds some nice sidebar links for documentation, the changelog, issues, and source code to PyPI. (Issue #107)

4.4 Version 0.1.2 (2021-04-01)

- The server uses a pool of child processes to run diffs. If the pool breaks while running a diff, it will be re-created once, and, if it fails again, the server will now crash with an exit code of `10`. (An external process manager like Supervisor, Kubernetes, etc. can then decide how to handle the situation.) Previously, the diff would fail at this point, but server would try to re-create the process pool again the next time a diff was requested. You can opt-in to the old behavior by setting the `RESTART_BROKEN_DIFFER` environment variable to `true`. (Issue #49)
- The diff server now requires Sentry 1.x for error tracking.

4.5 Version 0.1.2rc1 (2021-01-01)

- The server uses a pool of child processes to run diffs. If the pool breaks while running a diff, it will be re-created once, and, if it fails again, the server will now crash with an exit code of `10`. (An external process manager like Supervisor, Kubernetes, etc. can then decide how to handle the situation.) Previously, the diff would fail at this point, but server would try to re-create the process pool again the next time a diff was requested. You can opt-in to the old behavior by setting the `RESTART_BROKEN_DIFFER` environment variable to `true`. (Issue #49)

4.6 Version 0.1.1 (2020-11-24)

This is a bugfix release that focuses on *web_monitoring_diff.html_diff_render()* and the server.

- Fix an issue where the diffing server could reset the process pool that manages the actual diffs multiple times unnecessarily, leading to wasted memory and CPU. If you are tracking logs and errors, this will also make error messages about the diffing server clearer — you'll see "BrokenProcessPool" instead of "'NoneType' object does not support item assignment." (Issue #38)
- Ensure the server shuts down gracefully when pressing ctrl+c or sending a SIGINT signal. (Issue #44)
- Fix *web_monitoring_diff.html_diff_render()* to make sure the spacing of text and tags in the HTML source code of the diff matches the original. This resolves display issues on pages where CSS is used to treat spacing as significant. (Issue #40)
- Improve handling of lazy-loaded images in *web_monitoring_diff.html_diff_render()*. When images are lazy-loaded via JS, they usually use the `data-src` or `data-srcset` attributes, and we now check those, too. Additionally, if two images have no detectable URLs, we now treat them as the same, rather than different. (Issue #39)
- Stop showing inline scripts and styles in *web_monitoring_diff.html_diff_render()*. These still get wrapped with `<del>` or `<ins>` elements, but they don't show up visually since they aren't elements that should be visually rendered. (Issue #43)

4.7 Version 0.1.0

This project used to be a part of web-monitoring-processing, which contains a wide variety of libraries, scripts, and other tools for working with data across all the various parts of EDGI's Web Monitoring project. The goal of this initial release is to create a new, more focused package containing the diff-releated tools so they can be more easily used by others.

This release is more-or-less the same code that was a part of `web-monitoring-processing`, although the public API has been rearranged very slightly to make sense in this new, stand-alone context.

PYTHON MODULE INDEX

W

INDEX