### Contents

1 **How to use it** 3
   1.1 Getting started .................................................. 3
   1.2 Archiving URLs .................................................. 3
   1.3 Analyzing archived URLs ...................................... 5
   1.4 Visualizing archived URLs ...................................... 6
   1.5 Ingesting archived URLs from the Wayback Machine ............ 10

2 **Python interfaces** 11
   2.1 Archiving .................................................. 11
   2.2 Analysis .................................................. 12
   2.3 File handling .................................................. 16

3 **Command-line interfaces** 19
   3.1 storytracker-archive ........................................... 19
   3.2 storytracker-get ........................................... 19
   3.3 storytracker-links2csv ....................................... 20

4 **Changelog** 21
   4.1 0.0.9 .................................................. 21
   4.2 0.0.8 .................................................. 21
   4.3 0.0.7 .................................................. 21
   4.4 0.0.6 .................................................. 21
   4.5 0.0.5 .................................................. 21
   4.6 0.0.4 .................................................. 21
   4.7 0.0.3 .................................................. 22
   4.8 0.0.2 .................................................. 22
   4.9 0.0.1 .................................................. 22

5 **Credits** 23

6 **Contributing** 25
Tools for tracking stories on news homepages

Contents:
How to use it

1.1 Getting started

You can install storytracker from the Python Package Index using the command-line tool pip. If you don’t have pip installed follow these instructions. Here is all it takes.

$ pip install storytracker

You won’t need it for archival, but the analytical tools explained later one require that you have a supported web browser installed. Firefox will work, but it’s recommended that you install PhantomJS, a “headless” browser that runs behind the scenes.

On Ubuntu Linux that’s as easy as:

$ sudo apt-get install phantomjs

In Apple’s OSX you can use Homebrew to install it like so:

$ brew update && brew install phantomjs

1.2 Archiving URLs

1.2.1 From the command line

Once installed, you can start using storytracker’s command-line tools immediately, like storytracker.archive().

$ storytracker-archive http://www.latimes.com

That should pour out a scary looking stream of data to your console. That is the content of the page you requested compressed using gzip. If you’d prefer to see the raw HTML, add the --do-not-compress option.

$ storytracker-archive http://www.latimes.com --do-not-compress

You could save that yourself using a standard UNIX pipeline.


But why do that when storytracker.create_archive_filename() will work behind the scenes to automatically come up with a tidy name that includes both the URL and a timestamp?
$ storytracker-archive http://www.latimes.com --do-not-compress --output-dir="./"

Run that and you’ll see the file right away in your current directory.

```bash
# Try opening the file you spot here with your browser
$ ls | grep .html
```

### 1.2.2 With Python

UNIX-like systems typically come equipped with a built-in method for scheduling tasks known as cron. To utilize it with storytracker, one approach is to write a Python script that retrieves a series of sites each time it is run.

```python
import storytracker

SITE_LIST = [
    # A list of the sites to archive
    'http://www.latimes.com',
    'http://www.nytimes.com',
    'http://www.kansascity.com',
    'http://www.knoxnews.com',
    'http://www.indiatimes.com',
]

# The place on the filesystem where you want to save the files
OUTPUT_DIR = "/path/to/my/directory/"

# Runs when the script is called with the python interpreter
# ala "$ python cron.py"
if __name__ == "__main__":
    # Loop through the site list
    for s in SITE_LIST:
        # Spit out what you're doing
        print "Archiving %s" % s
        try:
            # Attempt to archive each site at the output directory
            # defined above
            storytracker.archive(s, output_dir=OUTPUT_DIR)
        except Exception as e:
            # And just move along and keep rolling if it fails.
            print e
```

### 1.2.3 Scheduling with cron

Then edit the cron file from the command line.

```bash
$ crontab -e
```

And use cron’s custom expressions to schedule the job however you’d like. This example would schedule the script to run a file like the one above at the top of every hour. Though it assumes that storytracker is available to your global Python installation at /usr/bin/python. If you are using a virtualenv or different Python configuration, you should begin the line with a path leading to that particular python executable.

```
0 * * * * /usr/bin/python /path/to/my/script/cron.py
```
1.3 Analyzing archived URLs

1.3.1 Extracting hyperlinks

The cron task above is regularly saving archived files to the OUTPUT_DIR. Those files can be accessed for analysis using tools like `storytracker.open_archive_filepath()` and `storytracker.open_archive_directory()`.

```python
>>> import storytracker

# This would import a single file and return a object we can play with

# This returns a list of all the objects found in the directory
>>> url_list = storytracker.open_archive_directory("/path/to/my/directory/")

# And remember you can still always do it on the fly
>>> url = storytracker.archive("http://www.cnn.com")
```

Once you have an url archive imported you can loop through all the hyperlinks found in its body tag which are returned as `ArchivedURL` objects.

```python
>>> url.hyperlinks
```

You could filter that list to just those estimated to be news stories like so.

```python
>>> [h for h in url.hyperlinks if h.is_story]
```

A complete list of hyperlinks and all their attributes can be quickly printed out in comma-delimited format.

```python
>>> f = open("./hyperlinks.csv", "wb")

>>> f = url.write_hyperlinks_csv_to_file(f)
```

The same thing can be done with our command line tool `storytracker-links2csv`.

```bash
```

Which also accepts a directory.

```bash
$ storytracker-links2csv /path/to/my/directory/
```

1.3.2 Tracking hyperlinks across a set of URLs

You can analyze how a particular hyperlink moved across a set of archived URLs like so:

```python
>>> urlset = storytracker.ArchivedURLSet(["http!www.nytimes.com!!!!@2014-08-25T01:00:02.455702+00:00.html", "http!www.nytimes.com!!!!@2014-08-25T01:15:02.464296+00:00.html"])

>>> urlset.sort()

>>> urlset.print_href_analysis("http://www.nytimes.com/2014/08/24/world/europe/russian-convoy-ukraine.html")
```

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
</tbody>
</table>
1.4 Visualizing archived URLs

1.4.1 Highlighted overlay

You can output a static image that pops out headlines, stories and images on the page using the `ArchivedURL.write_overlay_to_directory` method available on all `ArchivedURL` objects.

```python
obj = storytracker.archive("http://www.cnn.com")
obj.write_overlay_to_directory("/home/ben/Desktop")
```

The resulting image is sized at the same width and height of the real page. Images have a red stroke around them. Hyperlinks the system thinks link to stories have a purple border. The rest of the links go blue.
1.4. Visualizing archived URLs

Are missing students in mass graves?

Unmarked graves found in Mexico’s Guerrero state

Confessions of a Mormon housewife

A near-fatal illness that left 11 Straubing housebound opened her eyes to the Mormon ‘culture of perfection’.

OPINION

Why you want to pay this tax

MORE TOP STORIES

CNN’s human tumors

The week in TV
doc
tv

Reader: The CDC in lies about Ebola
deb
tv

Confessing a Mormon housewife

Once vast sea dries up to almost nothing

REPORTER

Reporters unflamed by haunted house
1.4.2 Abstract illustration

You can output an abstract image visualizing where headlines, stories and images are on the page using the `ArchivedURL.write_illustration_to_directory` method available on all `ArchivedURL()` objects. The following code will write a new image of the CNN homepage to my desktop.

```python
obj = storytracker.archive("http://www.cnn.com")
obj.write_illustration_to_directory("/home/ben/Desktop")
```

The resulting image is sized at the same width and height of the real page, with images colored red. Hyperlinks are colored in too. If our system thinks the link leads to a news story, it’s filled in purple. Otherwise it’s colored blue.
1.4.3 Animation that tracks hyperlink’s movement

You can create an animated GIF that shows how a particular hyperlink’s position shifted across a series of pages with the following code.

```python
>>> urlset.write_href_gif_to_directory(
>>>     # First give it your hyperlink
>>>     "http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html",
>>>     # Then give it the directory where you’d like the file to be saved
>>>     "./"
>>> )
```

1.5 Ingesting archived URLs from the Wayback Machine

A page saved by the Internet Archive’s excellent Wayback Machine can be integrated by passing its URL to `storytracker.open_wayback_machine_url()`.

This pulls down the CNN homepage captured on Sept. 11, 2001.

```python
>>> import storytracker
```

Now you have a `ArchivedURL` object like any other in the storytracker system.

```python
>>> obj
```

So, if for instance you wanted to see all the images on the page you could do this.

```python
>>> for i in obj.images:
>>>     print i.src
http://a388.g.akamai.net/f/388/21/1d/www.cnn.com//images/hub2000/1.gif
```

---

Chapter 1. How to use it
2.1 Archiving

Tools to download and save URLs.

2.1.1 archive

Archive the HTML from the provided URLs

\[\text{storytracker}.\text{archive}(\text{url, verify=True, minify=True, extend_urls=True, compress=True, output_dir=None})\]

Parameters

- \textbf{url} (str) – The URL of the page to archive
- \textbf{verify} (bool) – Verify that HTML is in the response’s content-type header
- \textbf{minify} (bool) – Minify the HTML response to reduce its size
- \textbf{extend_urls} (bool) – Extend relative URLs discovered in the HTML response to be absolute
- \textbf{compress} (bool) – Compress the HTML response using gzip if an output_dir is provided
- \textbf{output_dir} (str or None) – Provide a directory for the archived data to be stored

Returns An ArchivedURL object

Return type ArchivedURL

Raises ValueError If the response is not verified as HTML

Example usage:

```python
>>> import storytracker

>>> # This will return gzipped content of the page to the variable
>>> obj = storytracker.archive("http://www.latimes.com")
< ArchivedURL: http://www.latimes.com@2014-07-17 04:08:32.169810+00:00>

>>> # You can save it to an automatically named file a directory you provide
>>> obj = storytracker.archive("http://www.latimes.com", output_dir="./")
>>> obj.archive_path
'./http!www.latimes.com!!!!@2014-07-17T04:09:21.835271+00:00.gz'
```
2.1.2 get

Retrieves HTML from the provided URLs

```python
storytracker.get(url, verify=True)
```

**Parameters**

- `url (str)` – The URL of the page to archive
- `verify (bool)` – Verify that HTML is in the response’s content-type header

**Returns**

The content of the HTML response

**Return type**

`str`

**Raises** `ValueError` If the response is not verified as HTML

Example usage:

```python
>>> import storytracker

>>> html = storytracker.get("http://www.latimes.com")
```

2.2 Analysis

2.2.1 ArchivedURL

An URL’s archived HTML with tools for analysis.

```python
class ArchivedURL(url, timestamp, html, gzip_archive_path=None, html_archive_path=None, browser_width=1024, browser_height=768, browser_driver="PhantomJS")
```

**Initialization arguments**

- `url` – The url archived
- `timestamp` – The date and time when the url was archived
- `html` – The HTML archived

**Optional initialization options**

- `gzip_archive_path` – A file path leading to an archive of the URL stored in a gzipped file.
- `html_archive_path` – A file path leading to an archive of the URL storied in a raw HTML file.
- `browser_width` – The width of the browser that will be opened to inspect the URL’s HTML By default it is 1024.
- `browser_height` – The height of the browser that will be opened to inspect the URL’s HTML By default is 768.
- `browser_driver` – The name of the browser that Selenium will use to open up HTML files. By default it is `PhantomJS`.

**Other attributes**
**height**
The height of the page in pixels after the URL is opened in a web browser

**width**
The width of the page in pixels after the URL is opened in a web browser

**gzip**
Returns the archived HTML as a stream of gzipped data

**archive_filename**
Returns a file name for this archive using the conventions of `storytracker.create_archive_filename()`.

**hyperlinks**
A list of all the hyperlinks extracted from the HTML

**images**
A list of all the images extracts from the HTML

**largest_headline**
Returns the story hyperlink with the largest area on the page. If there is a tie, returns the one that appears first on the page.

**largest_image**
The largest image extracted from the HTML

**story_links**
A list of all the hyperlinks extracted from the HTML that are estimated to lead to news stories.

**summary_statistics**
Returns a dictionary with basic summary statistics about hyperlinks and images on the page

**Analysis methods**

**analyze()**
Opens the URL’s HTML in a web browser and runs all of the analysis methods that use it.

**get_cell(x, y, cell_size=256)**
Returns the grid cell where the provided x and y coordinates appear on the page. Cells are sized as squares, with 256 pixels as the default.

The value is returned in the style of algebraic notation used in a game of chess.

```python
>>> obj.get_cell(1, 1)
'a1'
>>> obj.get_cell(257, 1)
'b1'
>>> obj.get_cell(1, 513)
'a3'
```

**get_hyperlink_by_href(href, fails_silently=True)**
Returns the Hyperlink object that matches the submitted href, if it exists.

**open_browser()**
Opens the URL’s HTML in an web browser so it can be analyzed.

**close_browser()**
Closes the web browser opened to analyze the URL’s HTML

**Output methods**

**write_hyperlinks_csv_to_file(file, encoding="utf-8")**
Returns the provided file object with a ready-to-serve CSV list of all hyperlinks extracted from the HTML.
**write_gzip_to_directory** *(path)*
Writes gzipped HTML data to a file in the provided directory path

**write_html_to_directory** *(path)*
Writes HTML data to a file in the provided directory path

**write_illustration_to_directory** *(path)*
Writes out a visualization of the hyperlinks and images on the page as a JPG to the provided directory path.

Example usage:

```python
>>> import storytracker


>>> obj.url
'http://www.latimes.com'

>>> obj.timestamp
datetime.datetime(2014, 7, 6, 16, 31, 57, 697250)
```

### 2.2.2 ArchivedURLSet

A list of `ArchivedURL` objects.

**class ArchivedURLSet** *(list)*
List items added to the set must be unique `ArchivedURL` objects.

**hyperlinks**
Parses all of the hyperlinks from the HTML of all the archived URLs and returns a list of the distinct href hyperlinks with a series of statistics attached that describe how they are positioned.

**summary_statistics**
Returns a dictionary of summary statistics about the whole set of archived URLs.

**print_href_analysis** *(href)*
Outputs a human-readable analysis of the submitted href’s position across the set of archived URLs.

**write_href_gif_to_directory** *(href, path, duration=0.5)*
Writes out animation of a hyperlinks on the page as a GIF to the provided directory path

**write_hyperlinks_csv_to_file** *(file, encoding="utf-8")*
Returns the provided file object with a ready-to-serve CSV list of all hyperlinks extracted from the HTML.

Example usage:

```python
>>> import storytracker

>>> obj_list = storytracker.open_archive_directory('/home/ben/archive/)

>>> obj_list[0].url
'http://www.latimes.com'

>>> obj_list[1].timestamp
datetime.datetime(2014, 7, 6, 16, 31, 57, 697250)
```

### 2.2.3 Hyperlink

A hyperlink extracted from an `ArchivedURL` object.
class Hyperlink (href, string, index, images=[], x=None, y=None, width=None, height=None, cell=None, font_size=None)

Initialization arguments

href
  The URL the hyperlink references

string
  The strings contents of the anchor tag

index
  The index value of the links order within its source HTML. Starts counting at zero.

images
  A list of the Image objects extracted from the HTML.

x
  The x coordinate of the object’s location on the page.

y
  The y coordinate of the object’s location on the page.

width
  The width of the object’s size on the page.

height
  The height of the object’s size on the page.

cell
  The grid cell where the provided x and y coordinates appear on the page. Cells are sized as squares, with
  256 pixels as the default.

  The value is returned in the style of algebraic notation used in a game of chess.

font_size
  The size of the font of the text inside the hyperlink.

Other attributes

__csv__
  Returns a list of values ready to be written to a CSV file object

domain
  The domain of the href

is_story
  Returns a boolean estimate of whether the object’s href attribute links to a news story. Guess provided
  by storysniffer, a library developed as a companion to this project.

2.2.4 Image

class Image (src)

An image extracted from an archived URL.

Initialization arguments

src
  The src attribute of the image tag

x
  The x coordinate of the object’s location on the page.
The y coordinate of the object’s location on the page.

width
The width of the object’s size on the page.

height
The height of the object’s size on the page.

cell
The grid cell where the provided x and y coordinates appear on the page. Cells are sized as squares, with 256 pixels as the default.

The value is returned in the style of algebraic notation used in a game of chess.

Analysis methods

area
Returns the square area of the image

orientation
Returns a string describing the shape of the image.
‘square’ means the width and height are equal
‘landscape’ is a horizontal image with width greater than height
‘portrait’ is a vertical image with height greater than width None means there are no size attributes to test

2.3 File handling

Functions for naming, saving and retrieving archived URLs.

2.3.1 create_archive_filename

Returns a string that combines a URL and a timestamp of for naming archives saved to the filesystem.

storytracker.create_archive_filename(url, timestamp)

Parameters

• url (str) – The URL of the page that is being archived
• timestamp (datetime) – A timestamp recording approximately when the URL was archived

Returns A string that combines the two arguments into a structure can be reversed back into Python

Return type str

Example usage:

>>> import storytracker
>>> from datetime import datetime
>>> storytracker.create_archive_filename("http://www.latimes.com", datetime.now())
2.3.2 open_archive_directory

Accepts a directory path and returns an `ArchivedURLSet` list filled with an `ArchivedURL` object that corresponds to every archived file it finds.

```python
storytracker.open_archive_directory(path)
```

Parameters  
**path** (`str`) – The path to directory containing archived files.

Returns  
An `ArchivedURLSet` list

Return type  
`ArchivedURLSet`

Example usage:

```python
>>> import storytracker

>>> obj_list = storytracker.open_archive_directory('/home/ben/archive/)
```

2.3.3 open_archive_filepath

Accepts a file path and returns an `ArchivedURL` object

```python
storytracker.open_archive_filepath(path)
```

Parameters  
**path** (`str`) – The path to the archived file. Its file name must conform to the conventions of `storytracker.create_archive_filename()`.

Returns  
An `ArchivedURL` object

Return type  
`ArchivedURL`

Raises ArchiveFileNameError If the file’s name cannot be parsed using the conventions of `storytracker.create_archive_filename()`.

Example usage:

```python
>>> import storytracker

```

2.3.4 open_wayback_machine_url

Accepts a URL from the Internet Archive’s Wayback Machine and returns an `ArchivedURL` object

```python
storytracker.open_wayback_machine_url(url)
```

Parameters  

Returns  
An `ArchivedURL` object

Return type  
`ArchivedURL`

Raises ArchiveFileNameError If the file’s name cannot be parsed.

Example usage:

```python
>>> import storytracker

```
2.3.5 reverse_archive_filename

Accepts a filename created using the rules of `storytracker.create_archive_filename()` and converts it back to Python. Returns a tuple: The URL string and a timestamp. Do not include the file extension when providing a string.

```
storytracker.reverse_archive_filename(filename)
```

**Parameters**
- `filename` *(str)* – A filename structured using the style of the `storytracker.create_archive_filename()` function

**Returns**
A tuple containing the URL of the archived page as a string and a datetime object of the archive’s timestamp

**Return type**
tuple

Example usage:

```python
>>> import storytracker
```

2.3.6 reverse_wayback_machine_url

Accepts an url from the Internet Archive’s Wayback Machine and returns a tuple with the archived URL string and a timestamp.

```
storytracker.reverse_wayback_machine_url(url)
```

**Parameters**

**Returns**
A tuple containing the URL of the archived page as a string and a datetime object of the archive’s timestamp

**Return type**
tuple

Example usage:

```python
>>> import storytracker
```
3.1 storytracker-archive

Usage: storytracker-archive [URL]... [OPTIONS]

Archive the HTML from the provided URLs

Options:
- `-h, --help` show this help message and exit
- `-v, --do-not-verify` Skip verification that HTML is in the response’s content-type header
- `-m, --do-not-minify` Skip minification of HTML response
- `-e, --do-not-extend-urls` Do not extend relative urls discovered in the HTML response
- `-c, --do-not-compress` Skip compression of the HTML response
- `-d OUTPUT_DIR, --output-dir=OUTPUT_DIR` Provide a directory for the archived data to be stored

Example usage:

```
# This will pipe out gzipped content of the page to stdout
$ storytracker-archive http://www.latimes.com

# You can save it to an automatically named file a directory you provide
$ storytracker-archive http://www.latimes.com -d ./

# If you’d prefer to have the HTML without compression
$ storytracker-archive http://www.latimes.com -c

# Which of course can be piped into other commands like anything else
$ storytracker-archive http://www.latimes.com -cm | grep lakers
```

3.2 storytracker-get

Usage: storytracker-get [URL]... [OPTIONS]

Retrieves HTML from the provided URLs
Options:
- \( -h, \, --help \) show this help message and exit
- \( -v, \, --no-verify \) Skip verification that HTML is in the response’s content-type header

Example usage:

\# Download an url like this
\$ storytracker-get http://www.latimes.com

\# Or two like this
\$ storytracker-get http://www.latimes.com http://www.columbiamissourian.com

### 3.3 storytracker-links2csv

Usage: storytracker-links2csv [ARCHIVE PATHS OR DIRECTORIES]...

Extracts hyperlinks from archived files or streams and outputs them as comma-delimited values.

Options:
- \( -h, \, --help \) show this help message and exit

Example usage:

\# Extract from an archived file

\# Extract from a directory filled with archived file
\$ storytracker-links2csv /path/to/my/directory/
4.1 0.0.9

- Created a new method to write out an visualization of the page as an image file.

4.2 0.0.8

- Refactored analysis tools to use Selenium and PhantomJS rather than BeautifulSoup, which allowed for a whole of size and location attributes to be parsed from the fully rendered HTML document.

4.3 0.0.7

- Added `open_wayback_machine_url` and `reverse_wayback_machine_url` functions to introduce support for files saved by the Internet Archive’s Wayback Machine.

4.4 0.0.6

- `is_story` estimate added to each `Hyperlink` object as an attribute

4.5 0.0.5

- `Hyperlink` and `Image` classes
- `hyperlink` and `images` methods that extract them from `ArchivedURL`
- `write_hyperlinks_csv_to_file` method on `ArchivedURL` for outputs
- `storytracker-links2csv` command-line interface

4.6 0.0.4

- Timestamping of `archive` method now includes timezone, set to UTC by default
4.7 0.0.3

- More forgiving `urlparse` imports that work in both Python 2 and Python 3

4.8 0.0.2

- Changed automatic file naming process to work better with long file names
- Added basic logging to the archival functions

4.9 0.0.1

- Python functions for retrieving and saving URLs
- Command line tools for interactive with those function
Credits

This is a joint project of PastPages.org, The Reynolds Journalism Institute and the University of Missouri.
The lead developer is Ben Welsh.
CHAPTER 6

Contributing

• Code repository: https://github.com/pastpages/storytracker
• Issues: https://github.com/pastpages/storytracker/issues
• Packaging: https://pypi.python.org/pypi/storytracker
• Testing: https://travis-ci.org/pastpages/storytracker
Symbols

__csv__ (Hyperlink attribute), 15

A
analyze() (ArchivedURL method), 13
archive_filename (ArchivedURL attribute), 13
ArchivedURL (built-in class), 12
ArchivedURLSet (built-in class), 14
area (Image attribute), 16

B
browser_driver (ArchivedURL attribute), 12
browser_height (ArchivedURL attribute), 12
browser_width (ArchivedURL attribute), 12

cell (Hyperlink attribute), 15
cell (Image attribute), 16
close_browser() (ArchivedURL method), 13

domain (Hyperlink attribute), 15

F
font_size (Hyperlink attribute), 15

G
get_cell() (ArchivedURL method), 13
get_hyperlink_by_href() (ArchivedURL method), 13
gzip (ArchivedURL attribute), 13
gzip_archive_path (ArchivedURL attribute), 12

H
height (ArchivedURL attribute), 12
height (Hyperlink attribute), 15
height (Image attribute), 16
href (Hyperlink attribute), 15
html (ArchivedURL attribute), 12
html_archive_path (ArchivedURL attribute), 12
Hyperlink (built-in class), 14
hyperlinks (ArchivedURL attribute), 13
hyperlinks (ArchivedURLSet attribute), 14

I
Image (built-in class), 15
images (ArchivedURL attribute), 13
images (Hyperlink attribute), 15
index (Hyperlink attribute), 15

is_story (Hyperlink attribute), 15

L
largest_headline (ArchivedURL attribute), 13
largest_image (ArchivedURL attribute), 13

O
open_browser() (ArchivedURL method), 13

orientation (Image attribute), 16

P
print_href_analysis() (ArchivedURLSet method), 14

S
src (Image attribute), 15
story_links (ArchivedURL attribute), 13
storytracker.archive() (built-in function), 11
storytracker.create_archive_filename() (built-in function), 16
storytracker.get() (built-in function), 12
storytracker.open_archive_directory() (built-in function), 17
storytracker.open_archive_filepath() (built-in function), 17
storytracker.open_wayback_machine_url() (built-in function), 17
storytracker.reverse_archive_filename() (built-in function), 18
storytracker.reverse_wayback_machine_url() (built-in function), 18
string (Hyperlink attribute), 15
summary_statistics (ArchivedURL attribute), 13
summary_statistics (ArchivedURLSet attribute), 14

T
timestamp (ArchivedURL attribute), 12

U
url (ArchivedURL attribute), 12

W
width (ArchivedURL attribute), 13
width (Hyperlink attribute), 15
width (Image attribute), 16
write_gzip_to_directory() (ArchivedURL method), 13
write_href_gif_to_directory() (ArchivedURLSet method), 14
write_html_to_directory() (ArchivedURL method), 14
write_hyperlinks_csv_to_file() (ArchivedURL method), 13
write_hyperlinks_csv_to_file() (ArchivedURLSet method), 14
write_illustration_to_directory() (ArchivedURL method), 14

X
x (Hyperlink attribute), 15
x (Image attribute), 15

Y
y (Hyperlink attribute), 15
y (Image attribute), 15