WARC Documentation

Release 0.1

Internet Archive

February 07, 2017

ARC is a file format for storing web crawls as sequences of content blocks. It was developed in 1996 by Internet Archive.

WARC (Web ARChive) is an extension of the ARC file format, which adds more freedom by adding more metadata to each record and allowing named headers.

This python library works with files stored in both ARC and WARC formats.

CHAPTER 1

Installation

Installing warc is simple with pip:

```
$ pip install warc
```

or, with easy_install:

```
$ easy_install warc
```

Or you can get the sources by cloning the public git repository:

```
git clone git://github.com/anandology/warc.git
```

and install from sources:

```
$ python setup.py install
```

Reading a WARC File

Reading a warc file is as simple as reading a simple file. Instead of returning lines, it returns WARC records.

```
import warc
f = warc.open("test.warc.gz")
for record in f:
    print record['WARC-Target-URI'], record['Content-Length']
```

The `open` function is a shorthand for `warc.WARCFile`.:

```
f = warc.WARCFile("test.warc.gz", "rb")
f = warc.WARCFile(fileobj=StringIO(text))
```

Writing WARC File

Writing to a warc file is similar to writing to a regular file.:

```
f = warc.open("test.warc.gz", "w")
f.write_record(warc_record1)
f.write_record(warc_record2)
f.close()
```

CHAPTER 4

Working with WARC Header

The `warc.WARCHeader` object contains the list of WARC headers specified before the payload. It is just a dictionary.

```
>>> h = warc.WARCHeader({
...    "WARC-Type": "response",
...    "WARC-Date": "2012-02-03T04:05:06Z",
...    "WARC-Record-ID": "<urn:uuid:80fb9262-5402-11e1-8206-545200690126>",
...    "Content-Length": "42"
... })
>>>
>>> h['WARC-Type']
'response'
>>> h['WARC-Record-ID']
'<urn:uuid:80fb9262-5402-11e1-8206-545200690126>'
>>> h['Content-Length']
'42'
```

The headers are case-insensitive.

```
>>> h['warc-type']
'response'
>>> h['WARC-RECORD-ID']
'<urn:uuid:80fb9262-5402-11e1-8206-545200690126>'
```

The `WARCHeader` object is a real dictionary.

```
>>> h.keys()
['warc-type', 'content-length', 'warc-date', 'warc-record-id']
>>> h.values()
['response', '42', '2012-02-03T04:05:06Z', '<urn:uuid:80fb9262-5402-11e1-8206-545200690126>']
>>> h.get("Content-Type", "application/octet-stream")
'application/octet-stream'
```

The commonly used headers are accessible as attributes as well.

```
>>> h.type
'response'
>>> h.record_id
'<urn:uuid:80fb9262-5402-11e1-8206-545200690126>'
>>> h.content_length
42
>>> h.date
"2012-02-03T04:05:06Z"
```

Note that, `h.content_length` is an integer where as `h['Content-Length']` is a string.

When a new `WARCHeader` object is created, the `WARC-Record-ID`, `WARC-Date` and `Content-Type` headers can be initialized automatically.

```
>>> h = warc.WARCHeader({"WARC-Type": "response"}, defaults=True)
>>> h['WARC-Record-ID']
'<urn:uuid:3457ee2c-5e2c-11e1-a8ff-c42c0325ac11>'
>>> h['WARC-Date']
'2012-02-23T14:39:34Z'
>>> h['Content-Type']
'application/http; msgtype=response'
```

The `WARC-Record-ID` is set to a UUID, `WARC-Date` is set to current datetime and `Content-Type` is initialized based on the `WARC-Type`.

CHAPTER 5

Working with WARCRecord

A `WARCRecord` can be created by passing a `WARCHeader` object and payload, which defaults to None when unspecified.

```
>>> header = warc.WARCHeader({"WARC-Type": "response"}, defaults=True)
>>> record = warc.WARCRecord(header, "helloworld")
```

Or by passing a dictionary of headers.

```
>>> record = warc.WARCRecord(payload="helloworld", headers={"WARC-Type": "response"})
```

CHAPTER 6

License

The warc library is licensed under GPL v2 license. See LICENSE file for details.