serpextract

Release 0.2.5

Sep 27, 2017

Contents

CHAPTER 1

Overview

serpextract provides easy extraction of keywords from search engine results pages (SERPs).

Contents:

serpextract Package

serpextract.serpextract Package

serpextract.serpextract Module

Utilities for extracting keyword information from search engine referrers.

serpextract.serpextract.**get_parser**(*referring_url*)
Utility function to find a parser for a referring URL if it is a SERP.

Parameters **referring_url** (str or urlparse.ParseResult) – Suspected SERP URL.

Returns *SearchEngineParser* object if one exists for URL, None otherwise.

serpextract.serpextract.**is_serp**(*referring_url*, *parser=None*, *use_naive_method=False*)
Utility function to determine if a referring URL is a SERP.

Parameters

- **referring_url** (*str or urlparse.ParseResult*) – Suspected SERP URL.
- **parser** (*SearchEngineParser* instance or None.) – A search engine parser.
- **use_naive_method** (True or False) – Whether or not to use a naive method of search engine detection in the event that a parser does not exist for the given referring_url. See *extract()* for more information.

Returns True if SERP, False otherwise.

serpextract.serpextract.**extract**(*serp_url*, *parser=None*, *lower_case=True*, *trimmed=True*, *collapse_whitespace=True*, *use_naive_method=False*)

Parse a SERP URL and return information regarding the engine name, keyword and *SearchEngineParser*.

Parameters

- **serp_url** (str or urlparse.ParseResult) – Suspected SERP URL to extract a keyword from.
- **parser** (*SearchEngineParser*) – Optionally pass in a parser if already determined via call to get_parser.
- **lower_case** (True or False) – Lower case the keyword.
- **trimmed** (True or False) – Trim keyword leading and trailing whitespace.
- **collapse_whitespace** (True or False) – Collapse 2 or more \s characters into one space ' '.
- **use_naive_method** (True or False) – In the event that a parser doesn't exist for the given serp_url, attempt to find an instance of _naive_re_pattern in the netloc of the serp_url. If found, try to extract a keyword using _naive_params.

Returns an *ExtractResult* instance if serp_url is valid, None otherwise

serpextract.serpextract.**get_all_query_params**()

Return all the possible query string params for all search engines.

Returns a list of all the unique query string parameters that are used across the search engine definitions.

serpextract.serpextract.**get_all_query_params_by_domain**()

Return all the possible query string params for all search engines.

Returns a list of all the unique query string parameters that are used across the search engine definitions.

serpextract.serpextract.**add_custom_parser**(*match_rule*, *parser*)

Add a custom search engine parser to the cached _engines list.

Parameters

- **match_rule** (unicode) – A match rule which is used by *get_parser()* to look up a parser for a given domain/path.
- **parser** (*SearchEngineParser*) – A custom parser.

class serpextract.serpextract.**SearchEngineParser**(*engine_name*, *keyword_extractor*, *link_macro*, *charsets*, *hidden_keyword_paths=None*)

Bases: object

Handles persing logic for a single line in Piwik's list of search engines.

Piwik's list for reference:

https://raw.github.com/piwik/piwik/master/core/DataFiles/SearchEngines.php

This class is not used directly since it already assumes you know the exact search engine you want to use to parse a URL. The main interface for users of this module is the *extract()* method.

charsets

engine_name

get_serp_url(*base_url*, *keyword*)
Get a URL to a SERP for a given keyword.

Parameters

- **base_url** (str) – String of format '<scheme>://<netloc>'.
- **keyword** (str) – Search engine keyword.

Returns a URL that links directly to a SERP for the given keyword.

hidden_keyword_paths

keyword_extractor

link_macro

parse(*url_parts*)
Parse a SERP URL to extract the search keyword.

Parameters **serp_url** (A urlparse.ParseResult with all elements as unicode) – The SERP URL

Returns An *ExtractResult* instance.

class serpextract.serpextract.**ExtractResult**(*engine_name*, *keyword*, *parser*)

class serpextract.serpextract.**SearchEngineParser**(*engine_name*, *keyword_extractor*, *link_macro*, *charsets*, *hidden_keyword_paths=None*)
Handles persing logic for a single line in Piwik's list of search engines.

Piwik's list for reference:

https://raw.github.com/piwik/piwik/master/core/DataFiles/SearchEngines.php

This class is not used directly since it already assumes you know the exact search engine you want to use to parse a URL. The main interface for users of this module is the *extract()* method.

get_serp_url(*base_url*, *keyword*)
Get a URL to a SERP for a given keyword.

Parameters

- **base_url** (str) – String of format '<scheme>://<netloc>'.
- **keyword** (str) – Search engine keyword.

Returns a URL that links directly to a SERP for the given keyword.

parse(*url_parts*)
Parse a SERP URL to extract the search keyword.

Parameters **serp_url** (A urlparse.ParseResult with all elements as unicode) – The SERP URL

Returns An *ExtractResult* instance.

CHAPTER 2

Examples

Python

```
from serpextract import get_parser, extract, is_serp, get_all_query_params

non_serp_url = 'http://arstechnica.com/'
serp_url = ('http://www.google.ca/url?sa=t&rct=j&q=ars%20technica&source=web&cd=1&
↪ved=0CCsQFjAA'
            '&url=http%3A%2F%2Farstechnica.com%2F&ei=pf7RUYvhO4LdyAHf9oGAAw&
↪usg=AFQjCNHA7qjcMXh'
            'j-UX9EqSy26wZNlL9LQ&bvm=bv.48572450,d.aWc')

get_all_query_params()
# ['key', 'text', 'search_for', 'searchTerm', 'qrs', 'keyword', ...]

is_serp(serp_url)
# True
is_serp(non_serp_url)
# False

get_parser(serp_url)
# SearchEngineParser(engine_name='Google', keyword_extractor=['q'], link_macro=
↪'search?q={k}', charsets=['utf-8'])
get_parser(non_serp_url)
# None

extract(serp_url)
# ExtractResult(engine_name='Google', keyword=u'ars technica',␣
↪parser=SearchEngineParser(...))
extract(non_serp_url)
# None
```

Command Line

Command-line usage, returns the engine name and keyword components separated by a comma and enclosed in quotes:

```
$ serpextract "http://www.google.ca/url?sa=t&rct=j&q=ars%20technica"
"Google","ars technica"
```

You can also print out a list of all the SearchEngineParsers currently available in your local cache via:

```
$ serpextract -l
```

CHAPTER 3

Naive Detection

The list of search engine parsers that Piwik and therefore *serpextract.serpextract* uses is far from exhaustive. If you want *serpextract.serpextract* to attempt to guess if a given referring URL is a SERP, you can specify `use_naive_method=True` to *serpextract.serpextract.is_serp()* or *serpextract.serpextract.extract()*. By default, the naive method is disabled.

Naive search engine detection tries to find an instance of `r'\.?search\.'` in the `netloc` of a URL. If found, *serpextract.serpextract* will then try to find a keyword in the `query` portion of the URL by looking for the following params in order:

```
_naive_params = ('q', 'query', 'k', 'keyword', 'term',)
```

If one of these are found, a keyword is extracted and an `ExtractResult` is constructed as:

```
ExtractResult(domain, keyword, None)  # No parser, but engine name and keyword
```

```
# Not a recognized search engine by serpextract
serp_url = 'http://search.piccshare.com/search.php?cat=web&channel=main&hl=en&q=test'

is_serp(serp_url)
# False

extract(serp_url)
# None

is_serp(serp_url, use_naive_method=True)
# True

extract(serp_url, use_naive_method=True)
# ExtractResult(engine_name=u'piccshare', keyword=u'test', parser=None)
```

CHAPTER 4

Custom Parsers

In the event that you have a custom search engine that you'd like to track which is not currently supported by Piwik/*serpextract.serpextract*, you can create your own instance of *serpextract.serpextract.SearchEngineParser* and either pass it explicitly to either *serpextract.serpextract.is_serp()* or *serpextract.serpextract.extract()* or add it to the internal list of parsers.

```
# Create a parser for PiccShare
from serpextract import SearchEngineParser, is_serp, extract

my_parser = SearchEngineParser(u'PiccShare',          # Engine name
                               u'q',                  # Keyword extractor
                               u'/search.php?q={k}',  # Link macro
                               u'utf-8')              # Charset
serp_url = 'http://search.piccshare.com/search.php?cat=web&channel=main&hl=en&q=test'

is_serp(serp_url)
# False

extract(serp_url)
# None

is_serp(serp_url, parser=my_parser)
# True

extract(serp_url, parser=my_parser)
# ExtractResult(engine_name=u'PiccShare', keyword=u'test',␣
↪parser=SearchEngineParser(engine_name=u'PiccShare', keyword_extractor=[u'q'], link_
↪macro=u'/search.php?q={k}', charsets=[u'utf-8']))
```

You can also permanently add a custom parser to the internal list of parsers that *serpextract.serpextract* maintains so that you no longer have to explicitly pass a parser object to *serpextract.serpextract.is_serp()* or *serpextract.serpextract.extract()*.

```
from serpextract import SearchEngineParser, add_custom_parser, is_serp, extract
```

```
my_parser = SearchEngineParser(u'PiccShare',          # Engine name
                               u'q',                  # Keyword extractor
                               u'/search.php?q={k}',  # Link macro
                               u'utf-8')              # Charset
add_custom_parser(u'search.piccshare.com', my_parser)

serp_url = 'http://search.piccshare.com/search.php?cat=web&channel=main&hl=en&q=test'
is_serp(serp_url)
# True

extract(serp_url)
# ExtractResult(engine_name=u'PiccShare', keyword=u'test',␣
↪parser=SearchEngineParser(engine_name=u'PiccShare', keyword_extractor=[u'q'], link_
↪macro=u'/search.php?q={k}', charsets=[u'utf-8']))
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index