SerpScrap Documentation

Release 0.13.0

ecoron

Aug 26, 2019

Contents

SEO python scraper to extract data from major searchengine result pages. Extract data like url, title, snippet, rich-snippet and the type from searchresults for given keywords. Detect Ads or make automated screenshots. You can also fetch text content of urls provided in searchresults or by your own. It's usefull for SEO and business related research tasks.

CHAPTER 1

Extract these result types

- ads_main - advertisements within regular search results
- image - result from image search
- news - news teaser within regular search results
- results - standard search result
- shopping - shopping teaser within regular search results
- videos - video teaser within regular search results

1.1 For each result of a resultspage get

- domain
- rank
- rich snippet
- site links
- snippet
- title
- type
- url
- visible url

Also get a screenshot of each result page. You can also scrape the text content of each result url. It is also possible to save the results as CSV for future analytics. If required you can also use your own proxylist.

1.2 Ressources

See http://serpscrap.readthedocs.io/en/latest/ for documentation.

Source is available at https://github.com/ecoron/SerpScrap

CHAPTER 2

Contents

2.1 Install

```
pip uninstall SerpScrap -y
pip install SerpScrap --upgrade
```

On the first run SerpScrap will try to install the required Chromedriver or PhantomJS binary on Windows and Linux instances. If self install doesnt work you can configure your custom path to the chromedriver or phantomjs binary. For Linux SerpScrap provides https://github.com/ecoron/SerpScrap/blob/master/install_chrome.sh, this should be executed automaticly on the first run.

2.1.1 Chrome headless is recommended

By default SerpScrap is using the headless Chrome. You can also use phantomJS, but it is deprecated and it is also blocked very fast by the searchengine. We recommend to use headless Chrome.

2.1.2 lxml

lxml is required.

Windows

for windows you may need the lxml binary form here: http://www.lfd.uci.edu/~gohlke/pythonlibs/ For your convenience here are the direct links: * lxml[1]

In some cases you may need also Microsoft Visual C++ Build Tools[2] installed.

[1] http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml
[2] http://landinghub.visualstudio.com/visual-cpp-build-tools

iOS

is not supported yet

2.1.3 cli encoding issues

To avoid encode/decode issues use this command before you start using SerpScrap in your cli.

```
chcp 65001
set PYTHONIOENCODING=utf-8
```

References

2.2 Result Data

The result is returned as list of dictionaries like the example below. If url_scrape is enabled it may contain an additional property. If you prefer to save the results use the as_csv() method.

```
{
 'query': 'example',
 'query_num_results_total': 'Ungefähr 1.740.000.000 Ergebnisse (0,50 '
                            'Sekunden)\xa0',
 'query_num_results_page': 10,
 'query_page_number': 1,
 'serp_domain': 'dictionary.cambridge.org',
 'serp_rank': 4,
 'serp_rating': None,
 'serp_sitelinks': None,
 'serp_snippet': 'example Bedeutung, Definition example: something that is '
                 'typical of the group of things that it is a member of: .',
 'serp_title': 'example Bedeutung im Cambridge Englisch Wörterbuch',
 'serp_type': 'results',
 'serp_url': 'http://dictionary.cambridge.org/de/worterbuch/englisch/example',
 'serp_visible_link': 'dictionary.cambridge.org/de/worterbuch/englisch/example',
 'screenshot': '/tmp/screenshots/2017-05-21/google_example-p1.png'
}
```

If scrape_urls is True additional fields are appended to the resultset

```
{
 'meta_robots': 'index, follow', # value of meta tag robots
 'meta_title': 'Title of the page', # title of the url
 'status': '200', # response code
 'url': 'https://de.wikipedia.org', # scraped url
 'encoding': 'utf-8', # encoding of the url
 'last_modified': '26.08.2018  11:35:40', # datetime url lastmodified
 'text_raw': 'The raw text content scraped from url'
}
```

2.2.1 serp_type

The following serp_types are supported

- ads_main - advertisments within regular search results
- image - result from image search
- news - news teaser within regular search results
- results - standard search result
- shopping - shopping teaser within regular search results

2.2.2 Related keywords

To fetch related keywords for your given keyword you can use the method get_related() which returns a list of dicts

```
[{'keyword': 'example deutsch', 'rank': 1},
{'keyword': 'example email', 'rank': 2},
{'keyword': 'example definition', 'rank': 3},
{'keyword': 'example rapper', 'rank': 4}]
```

2.3 Configuration

Here we describe how you can configure SerpScrap to fit your needs. But it is also possible to run SerpScrap with the default settings.

2.3.1 Permissions

By default all needed or generated files are written into the local /tmp/ folder. The location can changed by configuration. Ensure the executing user has read/write permissions for this folder.

2.3.2 Default configuration

- cachedir: '/tmp/.serpscrap/' - path cachefiles
- chrome_headless: True - run chrome in headless mode, default is True
- clean_cache_after: 24 - clean cached files older then x hours
- database_name: '/tmp/serpscrap' - path and name sqlite db (stores scrape results)
- dir_screenshot: '/tmp/screenshots' - basedir for saved screenshots
- do_caching: True - enable / disable caching
- executable_path: '/usr/local/bin/chromedriver' - path to chromedriver, should detected automaticly
- google_search_url: 'https://www.google.com/search?' - base search url, modify for other countries
- headers: - dict to customize request header, see below
- num_pages_for_keyword: 2 - number of result pages to scrape
- num_results_per_page: 10 - number results per searchengine page
- results_age: 'Any' - specify age of results default Any, y - last year, m - last month, w - last week, d - last 24h, h - last hour
- proxy_file: '' - path to proxy file, see below

- sel_browser: 'chrome' - browser (chrome, phantomjs)
- scrape_urls: False - scrape urls of search results
- screenshot: True - enable screenshots for each query
- search_engines: ['google'] - search engines (google)
- sleeping_max: 15 - max seconds to sleep between scrapes
- sleeping_min: 5 - min seconds to sleep between scrapes
- url_threads: 3 - number of threads if scrape_urls is true
- use_own_ip: True - if using proxies set to False

2.3.3 Custom configuration

Change some config params.

```
import serpscrap

config = serpscrap.Config()
config.set('scrape_urls', False)

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
```

You can apply your own config dictionary. It is not required to provide any possible config key. by applying the default config values will be overwritten by the new values. for not provided config keys the deault values still exists.

```
import serpscrap

config = serpscrap.Config()
config_new = {
   'cachedir': '/tmp/.serpscrap/',
   'clean_cache_after': 24,
   'database_name': '/tmp/serpscrap',
   'do_caching': True,
   'num_pages_for_keyword': 2,
   'scrape_urls': True,
   'search_engines': ['google'],
   'google_search_url': 'https://www.google.com/search?',
   'executable_path', '/usr/local/bin/chromedriver',
}

config.apply(config_new)
scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
# scrap.init(config=config_new, keywords=keywords)
```

2.3.4 Headers

You can customize your searchengine request headers if you are using phantomJS (deprecated) by providing a dict in your configuration. If you don't customize this setting, the default is used.

```
config = {
  ...
  'headers': {
      'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
      'Accept-Language': 'de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4',
      'Accept-Encoding': 'gzip, deflate, sdch',
      'Connection': 'keep-alive',
  },
  ...
```

2.3.5 Proxy file

This feature works not stable in versions <= 0.9.1, if you use more then one worker and have more then one proxy in your file.

You can provide a list of proxies which should used for scraping the search engines. For this you have to create a proxy_file and to set the path to the file in the configuration.

The proxy_file should look like this

```
http XX.XXX.XX.XX:80
socks4 XX.XXX.XX.XX:80 username:password
socks5 XX.XXX.XX.XX:1080 username:password
```

In the configuration you need the following settings:

```
config.set('use_own_ip', False)
config.set('proxy_file', 'path_to_your_file')
```

2.4 Docker

You can run SerpScrap with Docker. The Image has any requirements and dependencies installed. The image is based on Debian Jessie (slim) and Python 3.6.

```
docker pull ecoron/serpscrap
```

Docker Hub: https://hub.docker.com/r/ecoron/serpscrap/

2.5 Example Usage

Here we show how you can use SerpScrap for your SEO and research tasks. You can use it from command line or as module in your application. Take also a look into the examples[1] on github.

2.5.1 Simple Example

```
python examples\example_simple.py
```

[1] https://github.com/ecoron/SerpScrap/tree/master/examples

In this example (example_simple.py[2]) we scrape results for the keyword "computer since". Also the serp result pages are crawled to scrape the raw text content of it. You can disable url scraping by setting the config value scrape_urls to False.

```
import serpscrap

keywords = ['computer since']

config = serpscrap.Config()
config.set('scrape_urls', True)

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
results = scrap.run()

for result in results:
    print(result)
```

2.5.2 Simple example using phantomjs (deprecated)

```
python examples\example_phantomjs.py
```

It is possible to use phantomJS, but we recomment Chrome. Depending on your choice both will be tried to install automaticly. For using Chrome you need the latest chromedriver[4] and to set the executable_path.

```
import pprint
import serpscrap

keywords = ['berlin']

config = serpscrap.Config()
config.set('sel_browser', 'phantomjs')

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
results = scrap.run()

for result in results:
    pprint.pprint(result)
    print()
```

2.5.3 Simple Example - custom phantomjs path (deprecated)

If phantomjs could not installed, configure your custom path to the binary.

```
import serpscrap

keywords = ['seo trends', 'seo news', 'seo tools']

config = serpscrap.Config()
config.set('sel_browser', 'phantomjs')
```

(continues on next page)

[2] https://github.com/ecoron/SerpScrap/blob/master/examples/example_simple.py
[4] https://sites.google.com/a/chromium.org/chromedriver/downloads

(continued from previous page)

```
# only required if phantomjs binary could not detected
config.set('executable_path', '../phantomjs/phantomjs.exe')
config.set('num_workers', 1)
config.set('scrape_urls', False)

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
results = scrap.run()
for result in results:
    if 'serp_title' in result and len(result['serp_title']) > 1:
        print(result['serp_title'])
```

2.5.4 Image search

```
python examples\example_image.py
```

To scrape the image search instead the standard serps, it's just enough to change the change the search_type in the config.

```
import pprint
import serpscrap

keywords = ['lost places']

config = serpscrap.Config()
config.set('search_type', 'image')

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
results = scrap.run()

for result in results[:10]:
    pprint.pprint(result)
    print()
```

2.5.5 Url Scrape Example

```
python examples\example_url.py
```

In this example we scrape only an url, without crawling any searchengine.

```
import serpscrap

url = 'https://en.wikipedia.org/wiki/Special:Random'

config = serpscrap.Config()

urlscrape = serpscrap.UrlScrape(config.get())
result = urlscrape.scrap_url(url)

print(result)
print()
```

2.5.6 Command Line

```
python serpscrap\serpscrap.py -k your keywords
```

As arguments provide one or more space separeted keywords. the result is printed into your console.

2.5.7 Example as_csv()

save the results for later seo analytics by using the as_csv() method. this method needs as argument the path to the file. The saved file is tab separated and values are quoted.

```
import serpscrap

keywords = ['seo tools', 'seo news']

config = serpscrap.Config()
config.set('scrape_urls', False)

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
scrap.as_csv('/tmp/seo-research')
```

2.5.8 Example serp results and raw text of result urls

You can scrape serp results and fetching the raw text contents of result urls at once

```
python examples\example_serp_urls.py
```

The resulting data will have additional fields containing data from the scraped urls.

```
import serpscrap

keywords = ['blockchain']

config = serpscrap.Config()
config.set('scrape_urls', True)

scrap = serpscrap.SerpScrap()
scrap.init(config=config.get(), keywords=keywords)
scrap.as_csv('/tmp/output')
```

2.5.9 Example related

If you are interested in related keywords for additional research tasks take a look into example_related.py[3] on github.

References

2.6 Additional Ressources

Here we list additional project related knowledge ressources.

[3] https://github.com/ecoron/SerpScrap/blob/master/examples/example_related.py

2.6.1 Documentation

For documentation we follow the sphinxs documentation Style Guide[1].

2.6.2 Tests

We use pytest for this project.

2.6.3 TFIDF

http://www.bogotobogo.com/python/NLTK/tf_idf_with_scikit-learn_NLTK.php

References

2.7 Usage

SerpScrap in your applications

```
import serpscrap

keywords = ['one', 'two']
scrap = serpscrap.SerpScrap()
scrap.init(keywords=keywords)
result = scrap.scrap_serps()
```

More detailes in the examples[1] section of the documentation.

[1] http://documentation-style-guide-sphinx.readthedocs.io/en/latest/style-guide.html
[1] http://serpscrap.readthedocs.io/en/latest/examples.html

CHAPTER 3

Supported OS

- SerpScrap should work on Linux, Windows and Mac OS with installed Python >= 3.4
- SerpScrap requieres lxml
- Doesn't work on iOS

3.1 Changes

Notes about major changes between releases

CHAPTER 4

0.13.0

- updated dependencies: chromedriver >= 76.0.3809.68 to use actual driver, sqlalchemy>=1.3.7 to solve security issues and other minor update changes
- minor changes install_chrome.sh

CHAPTER 5

0.12.0

I recommend an update to the latest version of SerpScrap, because the searchengine has updated the markup of search result pages(serp)

- Update and cleanup of selectors to fetch results
- new resulttype videos

CHAPTER 6

0.11.0

- Chrome headless is now the default browser, usage of phantomJS is deprecated
- chromedriver is installed on the first run (tested on Linux and Windows. Mac OS should also work)
- behavior of scraping raw text contents from serp urls, and of course given urls, has changed
- run scraping of serp results and contents at once
- csv output format changed, now it's tab separated and quoted

CHAPTER 7

0.10.0

- support for headless chrome, adjusted default time between scrapes

CHAPTER 8

0.9.0

- result types added (news, shopping, image)
- Image search is supported

CHAPTER 9

0.8.0

- text processing tools removed.
- less requirements

9.1 References

SerpScrap is using Chrome headless[2] and lxml[3] to scrape serp results. For raw text contents of fetched URL's, it is using beautifulsoup4[4] . SerpScrap also supports PhantomJs[5] ,which is deprecated, a scriptable headless WebKit, which is installed automaticly on the first run (Linux, Windows). The scrapcore was based on GoogleScraper[6] , an outdated project, and has many changes and improvements.

[2] http://chromedriver.chromium.org/
[3] https://lxml.de/
[4] https://www.crummy.com/software/BeautifulSoup/
[5] https://github.com/ariya/phantomjs
[6] https://github.com/NikolaiT/GoogleScraper