Experiment – 3: Search engine optimization- implement Spamdexing

Using Python scripts to analyse SEO (Search Engine Optimization) and broken links on your site

Python is all about automating repetitive tasks, leaving more time for your other Search Engine Optimization (SEO) efforts. Not many SEOs use Python for their problem-solving, even though it could save you a lot of time and effort. Python, for example, can be used for the following tasks:

- Data extraction
- Preparation
- Analysis & visualization
- Machine learning
- Deep learning

We'll be focussing mostly on data extraction and analysis in this article. The required modules will be indicated for each script.

Python SEO analyzer

A really useful script for analyzing your website is called **'SEO analyzer'**. It's an all round website crawler that analyses the following information:

- Word count
- Page Title
- Meta Description
- Keywords on-page
- Warnings
- Missing title
- Missing description
- Missing image alt-text

This is great for a quick analysis of your basic SEO problems. As page title, meta descriptions and on-page keywords are important ranking factors, this script is perfect for gaining a clear picture of any problems that might be in play.

Using the SEO analyzer

After having installed the necessary modules (BeautifulSoup 4 + urllib2) for this script and having updated your Python to version 3.4+, you are technically ready to use this script. Json or working

variants, however, can be useful for exporting the data you gain from the SEO analyser. After having installed the script, these are the commands you can use:

seoanalyze http://internetvergelijnk.nl/

seoanalyze https://telefoonvergelijk.nl --sitemap

https://telefoonvergelijk.nl/sitemap_index.xml

As seen in the examples above, for both internetvergelijk and telefoonvergelijk , it's possible to either crawl the website, or the XML sitemap of a website in order to do an SEO analysis. Another option is to generate HTML output from the analysis instead of using json. This can be done through the following command:

seoanalyze http://internetvergelijk.nl/ --output-format-html

If you have installed json and want to export the data, use the following command:

from seoanalyzer import analyse output = analyse(site, sitemap) print(output)

You can also choose for an alternative path, running the analysis as a script, as seen in the example below:

This will export the file into a html after having run the –output-format html script. This seoanalyze script is great for optimizing your page titles, meta descriptions, images and on-page keywords. It's also a lot faster than Screaming Frog, so if you're only looking for this information, running the seoanalyze script is more efficient.

Link status analyser

Another way to use Python for Search Engine Optimization is by using a script that crawls your website and analyses your URL status codes. This script is called Pylinkvalidator and can be found here). All it requires is BeautifulSoup if you're running it with Python 3.x. If you're running a 2.x version like 2.6 or 2.7, you should not need BeajutifulSoup.

In order to speed up the crawling, however, it might be useful to install the following libraries:

1) lxml – Speeds up the crawling of HTML pages (requires C libraries) 1) gevent – enables pylinkvalidator to use green threads 1) cchardet – Speeds up document encoding detection

Do keep this in mind, they could be very useful for crawling larger websites, and just to enhance the link status analyser.

What this script essentially does, is crawl the entire URL structure of a website in order to analyse the status codes of each and every URL. This makes it a very long process for bigger websites, hence the recommendation of using the optional libraries to speed this up.

Using the link status analyser

Pylinkvalidator has a ton of different usage options for usage. For example, you can:

- Show progress
- Crawl the website and pages belonging to another host
- Only crawl a single page and the pages it links to
- Only crawl links, ignore others (images, stylesheets, etc.)
- Crawl a website with more threads or processes than default
- Change your user agent
- Crawl multiple websites
- Check robots.txt
- Crawling body tags and paragraph tags

Showing progress through **-P** or **--progress** is recommended, as without it, you will find yourself wondering when your crawl will be done without any visual signs. The command for crawling more threads **(-- workers='number of workers')** and processes **(-- mode=process -- workers='number of workers')** can be very useful as well.

Of course, the script has many more options to explore. The examples below show some of the possible uses:

pylinkvalidate.py -p http://www.example.com/

The function above crawls the website and shows progress.

pylinkvalidate.py -p workers=4 http://www.example.com/

This function crawls a website with multiple threads and shows progress.

pylinkvalidate.py -p --parser=lxml http://www.example.com/

This function uses the lxml library in order to speed up the crawl while showing progress.

pylinkvalidate.py -P --types=a http://www.example.com/

The function above only crawls links **(<a href>)** on your website, ignoring images, scripts, stylesheets and any other non-link attribute on your website. This is also a useful function when crawling the URLs of large websites. After the script has run its course, you'll get a list of URLs with status codes 4xx and 5xx that have been found by crawling your website. Along with that, you'll gain a list of URLs that link to that page, so you'll have an easier time fixing the broken links. The regular crawl does not show any 3xx status codes. For more detailed information about what URLs your pages can be reached from, try the following function:

pylinkvalidate.py --report-type=all http://www.example.com/

This give information about the status code of a page, and all the other pages that link to a page.

An incredibly useful **SEO** tool you can use to crawl your website for broken links (404) and server errors. Both of these errors can be bad for your SEO efforts, so be sure to regularly crawl your own website in order to fix these errors ASAP.

Conclusion

While these scripts are incredibly useful, there are a lot of various uses for Python in the world of SEO. Challenge yourself to create scripts that make your SEO efforts more efficient.

Spamdexing

Spamdexing (also known as **search engine spam**, **search engine poisoning**, **black-hat search engine optimization**, **search spam** or **web spam**) is the deliberate manipulation of **search engine indexes**. It involves a number of methods, such as **link building** and repeating unrelated phrases, to manipulate the relevance or prominence of resources indexed in a manner inconsistent with the purpose of the indexing system.

Spamdexing could be considered to be a part of **search engine optimization**, although there are many SEO methods that improve the quality and appearance of the content of web sites and serve content useful to many users.

Common Spamdexing techniques can be classified into two broad classes: **content spam** (or term spam) and **link spam**.

Content spam

These techniques involve altering the logical view that a search engine has over the page's contents. They all aim at variants of the vector space model for information retrieval on text collections.

- **Keyword stuffing**
- **Hidden or invisible text**
- **Meta-tag stuffing**
- **Doorway pages**
- **Scraper sites**
- **Article spinning**
- **Machine translation**

Link spam

Link spam is defined as links between pages that are present for reasons other than merit. Link spam takes advantage of link-based ranking algorithms, which gives websites higher rankings the more other highly ranked websites link to it. These techniques also aim at influencing other link-based ranking techniques such as the HITS algorithm.

Link farms

- **Private blog networks**
- **Hidden links**
- **Sybil attack**
- **Spam blogs**
- **Guest blog spam**
- **Buying expired domains**

Using world-writable pages

Web sites that can be edited by users can be used by spamdexers to insert links to spam sites if the appropriate anti-spam measures are not taken.

Automated spambots can rapidly make the user-editable portion of a site unusable. Programmers have developed a variety of automated spam prevention techniques to block or at least slow down spambots.

- **Spam in blogs**
- **Comment spam**
- **Wiki spam**
- **Referrer log spamming**
- **Countermeasures**