Sign-of-life crawler

This paper describes the approach used by the crawler. Updated on 28/06/2019.

Agenda

[I -MAIN APPLICATION 2](#_Toc12622635)

[1) Collecting domain data 2](#_Toc12622636)

[2) Parking classification 2](#_Toc12622637)

[II - LANGUAGE EXTENSION FOR PARKING DETECTION 5](#_Toc12622638)

[1) The parking pattern 5](#_Toc12622639)

[2) Getting the pattern key words 6](#_Toc12622640)

[III- LIMITS AND IMPROVEMENT DIRECTIONS 7](#_Toc12622641)

[1) False positive for parking patterns 7](#_Toc12622642)

[2) Languages 7](#_Toc12622643)

[3) External JavaScript redirection 7](#_Toc12622644)

[4) Texts in images 7](#_Toc12622645)

## I -MAIN APPLICATION

### 1) Collecting domain data

* Collect zonefiles of a given TLD and convert it into a csv file of all URLs to visit.
* Visiting each URL of the above TLD using the following format of full URL: http:// + URL (without www.)
  + Python, like many other general purpose codes and like all browsers (Chrome, Firefox…), has HTTP client libraries. So far, “aiohttp” is used. This library allows communicating on the internet by sending messages following the HTTP or HTTPS protocols. In particular, the standard GET method will yield the content of the main page of an URL website, which is an HTML page. This method is also used by browsers when we type the URL. With this method, the code will send GET messages to all websites of the above TLD.
* In DEBUG mode, it stores the content of the website as well as some server interactions (e.g. redirections) for later use

For URLs that provided a correct response, the next two steps are performed:

* Extracting the displayed text from the HTML and storing it
  + An HTML page contains HTML, CSS, JavaScript and PHP codes. It defines precisely the content and format of a website. In particular, some of the displayed text is directly written on the page, like “this domain is parked”. To collect this text, some scraping libraries help to partially interpret the HTML code. It will identify tags, for example the tag “h1” is often used for titles of a webpage and “p” for paragraphs. So far, the library “beautifulSoup” is used. At the date of writing, JavaScript is not interpreted, but might be implemented in a later development (parked domains are often very simple websites).
* Identifying the language of the website
  + A simple natural language processing (NLP) library, called “langdetect”, allows identifying the language of the displayed text. The recent progress in NLP provides access to more advanced approaches for this task.

With all this collected information, the classification models can begin: So far, a parking classification is implemented.

### 2) Parking classification

The classification follows the decision tree below

#### Errors:

* If the HTTP Client fail to identify the IP address of a URL, the URL is categorized as **Connection Error**.
* If the IP address is found but the corresponding server fails to give a valid answer, the URL is categorized as **Invalid Response**.
* If the server provides a response with a status code 301, 302, 307, 308, the domain is redirecting to another page. This page is visited and the classification parking is reinitialized. A redirection flag keeps track of this operation.
* If the server provides a response with status code 401-2-3, we are not authorized to communicate to this server. A second run will be planned with a VPN to avoid region restrictions. If the code 401-2-3 persists, the URL is categorized as **HTTP error**.
* If the server answers with other 4XX (client errors) or 5XX (server errors) code, the URL is categorized as **HTTP Error**.

#### Valid responses

The server provides a 2XX code with an HTML page

* The HTML page is parsed to extract specific information related to links to other websites, as well as type of content in the page:
  + “a” and “link”: this tag indicates a link to another website/page/object
  + “frameset”,”frame” and ”iframe” : HTML tag to display a page from a different location
  + “window.location” : similar to iframe but with JavaScript code
  + The META tag with the key-value “http-equiv=refresh”: another way of redirecting
  + Presence of JavaScript code: “script” tag
  + Count number of non-trivial HTTP lines, words and letters

The second part of the tree can now be implemented:

* if the domain of the URL of the response is different than the initial domain 🡪 the target page is used for classification.The redirection flag is raised.
* Following the redirection tree in the picture below, if there is only one significant meta-refresh AND/OR frameset and frame OR iframe OR window.location(.href), the HTML page contains less than 150 lines and the link leads to another domain. Here “significant” means the link is not empty/blank.html/notfound and the element is not included in basic content tags like P, TR, TL, SPAN… 🡪 the redirection target page is visited and used for classification.The redirection flag is raised.
* If there are links of one and only one registrar website 🡪 **Parking Response Registrar**

If Redirected is true and the Parking Response Registrar is false, the target link is saved then visited for information purpose. The domain remains Redirected. The following steps are also performed on the target link (parking notice and blank page detection).

* The displayed text is then analyzed in four steps, using a regular expression library:
  + Get the patterns for the identified language. These patterns are defined in the language extension below. In addition, English patterns are always searched by default, no matter the detected language
  + Find a core key word of the pattern
  + Extract the surrounding sentence. The definition of a sentence depends on the language.
  + Look up for any attribute key word in the sentence, before or after the core key word.

🡪If these four steps yield at least one full pattern, the site is categorized as **Parking Individual Content**. For now, English, French, Spanish, Russian, Chinese and Swedish are supported.

* If the displayed text has less than 5 words and has no JavaScript and no frames, the URL is considered as **Blank Page**
* If the page does not fit any of the above criteria, it is considered as a **Normal**.

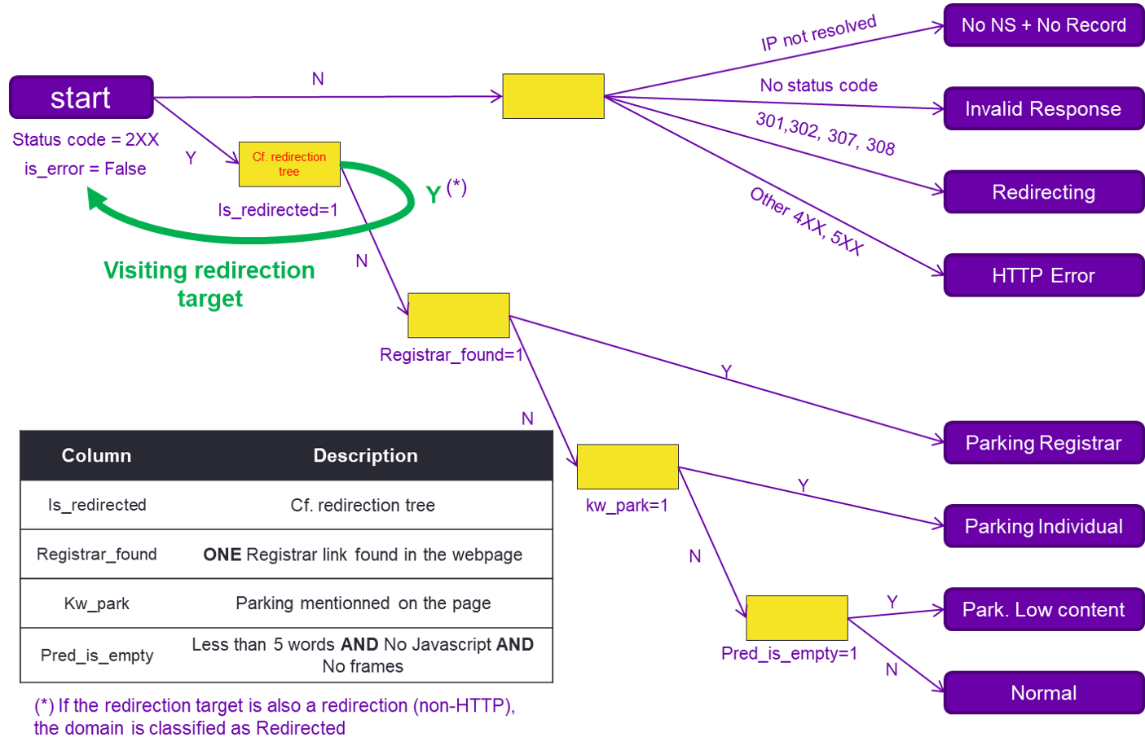


Figure 1 - parking decision tree

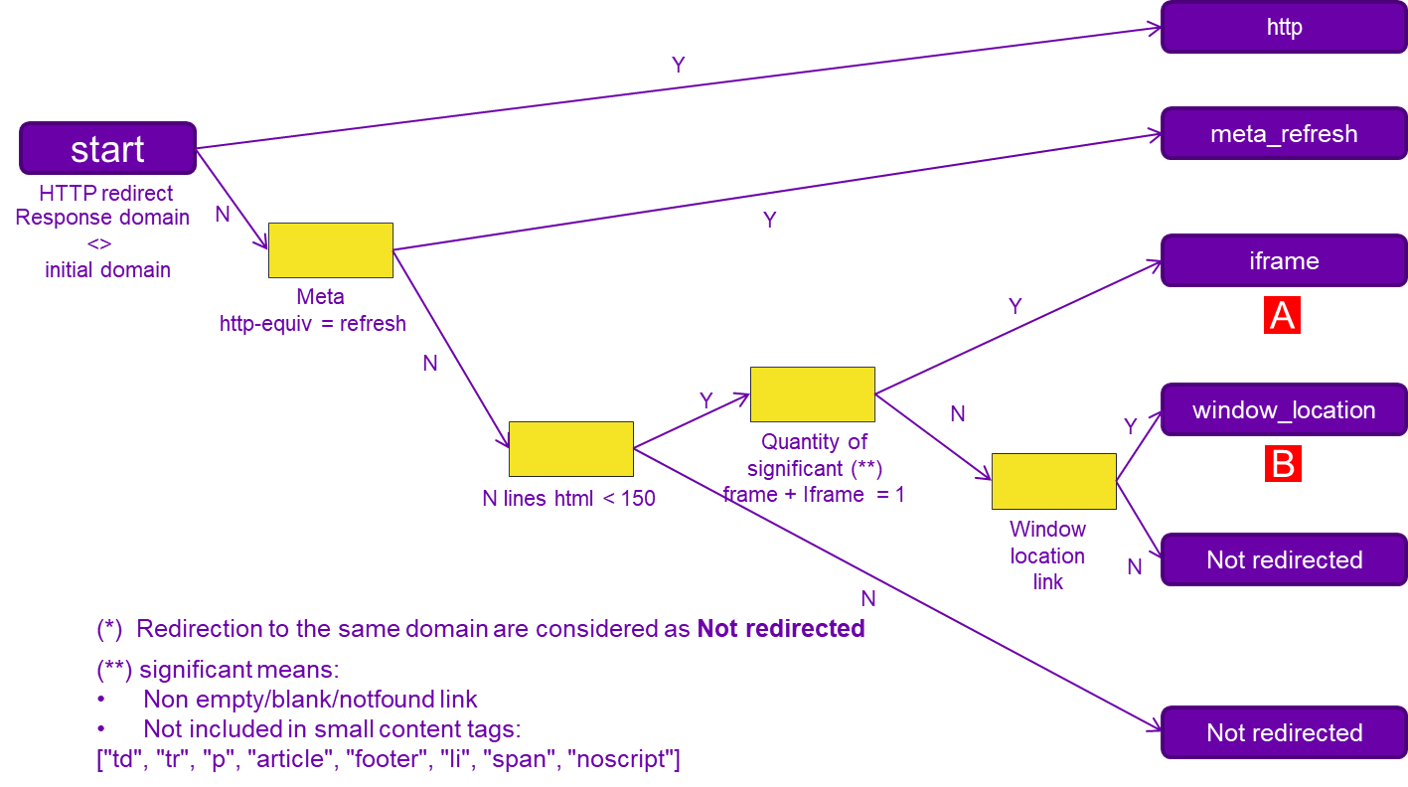


Figure 2 - Redirection tree

## II - LANGUAGE EXTENSION FOR PARKING DETECTION

To add a new language, an extension has been developed. It uses a bottom up approach to identify the various sentences that indicates a parked page. It outputs a selection of keywords that are good candidates for a parking pattern.

### The parking pattern

In all visited websites, a parking notice (like “this website is for sale” or “This is a parked domain”) always follow a simple pattern described in the picture below. The pattern is a core key word and an attribute key word in the same sentence.

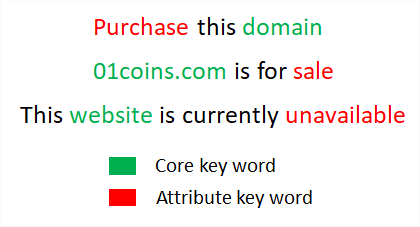


Figure 3 - parking pattern

To detect this pattern, Regular Expressions (=RegEx) are used. This library allows to find a sequence of characters in a text. For example, the regex “sales\*” will allow to detect both the keywords “sale” and “sales”. A more advanced use is implemented for the sentences “ www.abc.com is reserved” and www.abc.com All rights reserved”. The first sentence is clearly a parked notice but not the second one, which is only the copyright. To handle this case, the regex “ (?<!rights) reserved” will detect all the words “ reserved” that are not preceded by “rights”.

Finally, some attribute key words are specific to one or more core key words. For example, the attribute “register” indicates a parked domain when it is associated with the core “domain”, but not with “website” (= login).

### Getting the pattern key words

To get the relevant keywords for a given language, a Semi-Automatized research is implemented. The process to add a language includes the following steps:

* In function get\_unicode\_set (cf. next figure), add the regular expression pattern that matches all characters for that language. This is available online. Indeed, each language has a set of specific characters (e.g. ñ in Spanish). This character set defines the sentence. As a consequence, sentence separators must be excluded (points, question mark…).
* Translate the usual core keywords equivalent to “domain”, “website” and “page”.
* Extract the surrounding sentence for a given subset of URLs. Ideally the URLs should contain a lot of websites with the target language (AUTO)
* Count attribute key words across all these URLs and sort them by recurrence (AUTO)
* Using Google Translate (with Google Sheets) as well as visits on the website, validate the attribute key words that matches a parked website. Usually, translations of “parked” or “for sale” are the most popular attributes. For the detailed list of patterns, please refer to the script language\_patterns.py. The exact sentences they are built to capture are listed in test\_parking\_pattern.py .
* Update the referential patterns in language\_patterns.py with the key words found.
* For maintainability, add the unique patterns to the test cases in test\_parking\_pattern.py

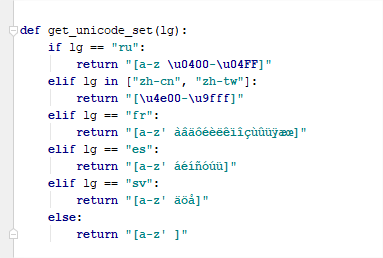


Figure 4 - characters used in Russian, Chinese, French, Spanish, Swedish and English

## III- LIMITS AND IMPROVEMENT DIRECTIONS

### False positive for parking patterns

Getting a comprehensive list of parking notice patterns is most likely impossible. Since millions of websites are to be visited, there will always be a few outliers with a specific phrasing. As long as the dominant practices are captured, we can detect 95% + of the parked domain with a notice.

To address this issue, the patterns have been built by giving priority to the most common sentences around the core keywords (domain, page, website, url.com). New patterns can be added to the referential patterns (language\_patterns.py) as soon as they are discovered. The test use cases will help prevent regression.

### Languages

As mentioned, for now, English, French, Spanish, Russian, Chinese and Swedish are supported. To check relevance of results, a table sums up how many unsupported languages have been found at the end of a run. According to logs of crawling, German and Japanese are prevailing as well.

### External JavaScript redirection

When a redirection is performed inside a JavaScript only code, the only way to detect it is to interpret all the webpage and its outbound links (all .JS files). To do that, a JavaScript interpreter, like a browser, is necessary. However, this is much slower. For parking classification, we can assume that most of the redirection occurs in the main HTML page.

### Texts in images

Texts that are saved as pixels in an image are currently not considered. For this case, we rely on the text surrounding the image to classify the page.