Sign-of-life crawler

This paper describes the approach used by the crawler. Last updated on 25/06/2020.

Agenda

Agenda 1

I - TECHNICAL CLASSIFICATION 2

A) TECHNICAL CATEGORIES DEFINITION 2

B) TECHNICAL CLASSIFICATION 5

C) TEXT CLASSIFICATION 10

D) PIPELINE PERFORMANCE 13

II- SOCIAL MEDIA ACTIVITY 14

1) Objective 14

2) Detailed approach 18

III- MAIL SERVER IDENTIFICATION 19

1) Objective 19

2) Approach 20

IV- LIMITS AND IMPROVEMENT DIRECTIONS 20

1) Complex dynamic websites 20

2) Dataset size 20

3) Texts in images 20

I - TECHNICAL CLASSIFICATION

A) TECHNICAL CATEGORIES DEFINITION

Given a top-level domain, our objective is to identify how much of it is used in a meaningful way, at domain name granularity. In particular, we want to classify a domain name in one of the following categories, from lower to higher granularity (level 1 to 4).

category_lv1	category_lv2	category_lv3	category_lv4	description
content	high content	high content	high content	Website with significant content
	low content	Parked Notice Registrar	Parked Notice Registrar	Website or default page of a registrar
		Blocked	Blocked	Website with an explicit "blocked" note
		Upcoming	Under construction	Website with an explicit "under construction" note
			Starter	Initial page of a website builder
		Abandoned	Expired	Website with an explicit "expired" note
		Not used	Index of	Website with the "index of" structure
			Blank Page	empty page (no link, images, redirections)
		Parked Notice Individual Content	For sale	Website with a selling note
			Reserved	Website with a reserved note
			Parked Notice Individual Content	Other website with no significant content
no content	errors	DNS Error	No address found	No server/IP address assiociated to the domain
		Connection Error	Refused Connection	Connection refused by the server
			Timeout	Domain took too long to answer
		Invalid Response	No Status Code	Domain returns an uninterpretable response (= error occured while trying to decode and separate all the elements of the response)
		HTTP Error	HTTP_401	HTTP Status code 401
			HTTP_403	HTTP Status code 403

			HTTP_404	HTTP Status code 404
			HTTP_408	HTTP Status code 408
			HTTP_500	HTTP Status code 500
			HTTP_502	HTTP Status code 502
			HTTP_504	HTTP Status code 504
			HTTP_other	Other HTTP status code

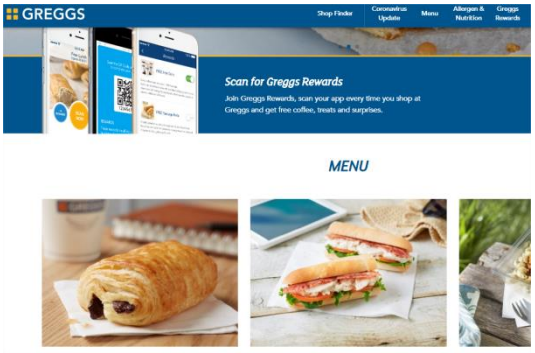


Figure 1 - example of “High content” page

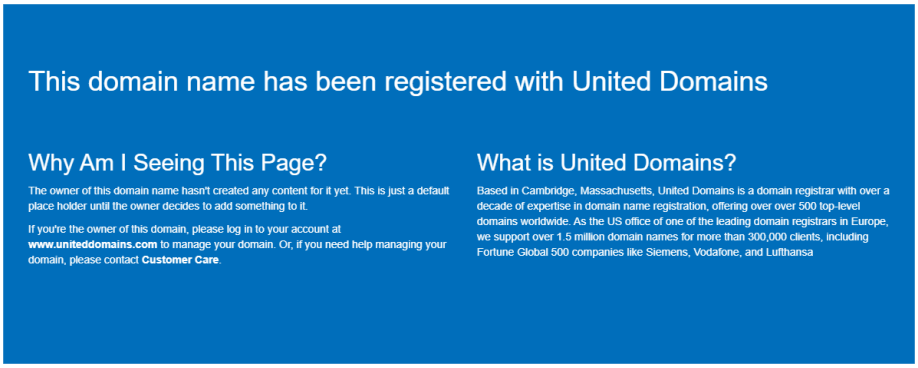


Figure 2 - example of "Parked Notice Registrar" page

Figure 3 - example of " Under construction" page

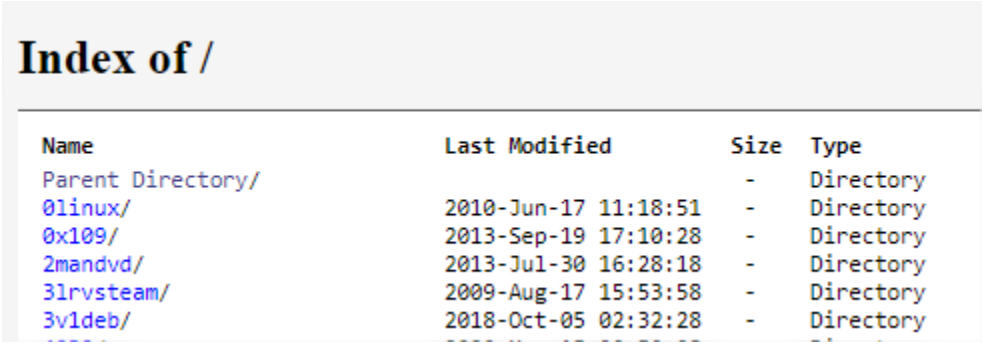


Figure 4 - example of "Index Of" page

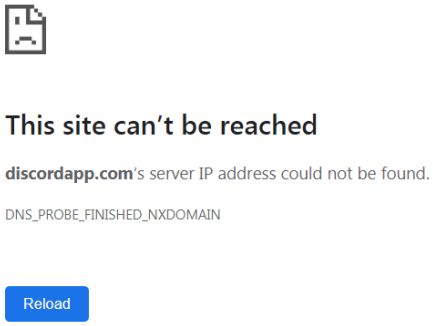


Figure 5 - example of "No address found" page

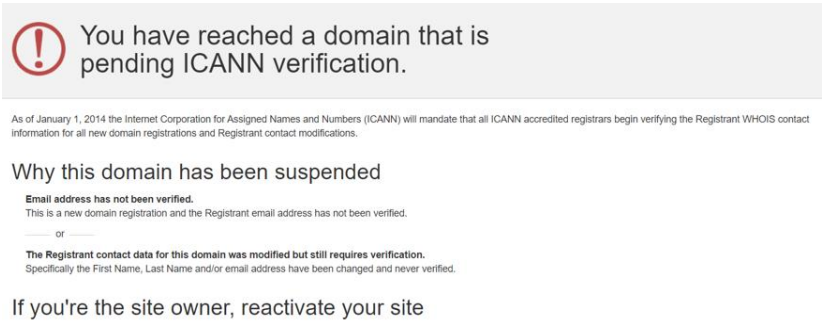


Figure 6 - Example of expired website

B) TECHNICAL CLASSIFICATION

To reach this goal, a fully automatic pipeline has been set up with Python. Its general architecture is described in the following chart.

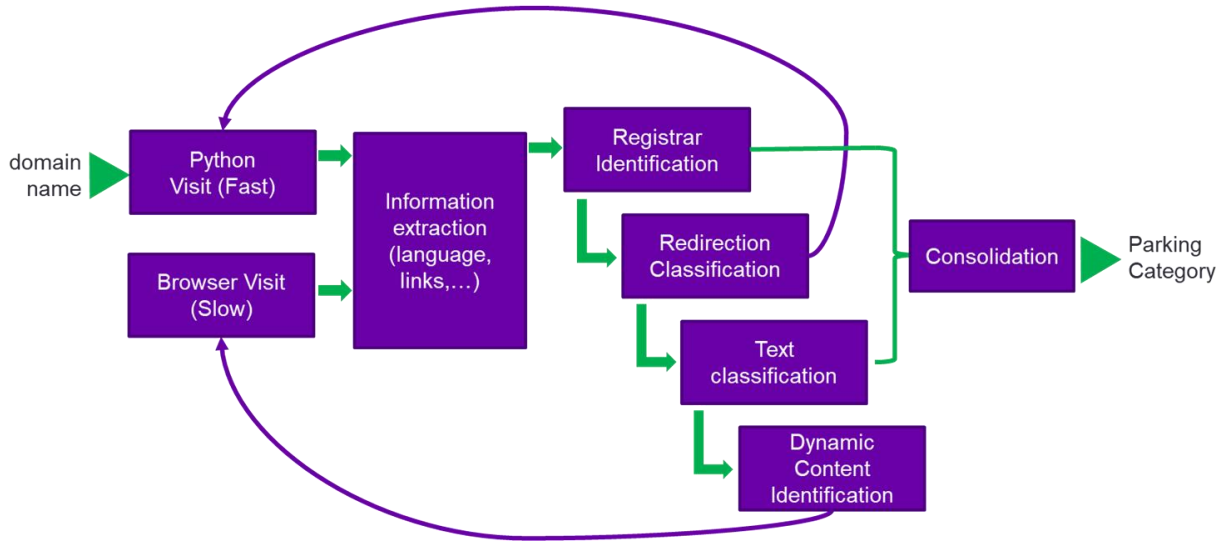


Figure 7 - General architecture of the crawler

In general:

- The domain is visited with Python to collect the HTML page
- Information is extracted from that first visit
- Registrar links are searched, if one is found, the page is classified as “Parked Notice Registrar”
- If a redirection is detected, we repeat from the first step with the redirection target link (the results of the target link classification are used for the initial domain)
- The displayed text is analyzed to identify parking notes (ex: “website under construction”)
- If the page hasn’t been classified as parked so far, the page is analyzed to identify clues of dynamic content generation.
- If the page needs to be interpreted, the domain is visited with Chrome browser, controlled by Python. When the full page is rendered, we repeat the steps from information extraction to text classification

The following sections describe each step in more details.

1 ） Python Visit

For a given TLD:

- 50,000 domain names are collected from its zonefile into a csv file. Based on samples analysis, 50,000 URLs will yield a confidence interval below 1 % range.
- Each URL is visited 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 a website, which is an HTML page. However, the html code is not interpreted to maintain a high speed of visits.

2) Browser Visit

In some domains, the only way to have access to the content of the page is to interpret the code. When interpreting a code, each instruction is implemented, whether it is a resource collection (image, video, library, 3rd party services), a content generation (text, charts…) or page formatting. To do that, we use Chrome through its webdriver. To convert instructions from Python to the webdriver, the library “selenium” is used. This approach is equivalent to opening Chrome with your mouse and type the url in the address bar (except with regard to scarcely implemented bot detectors). As output we get the final HTML page with its rendered displayed content.

Even though multiple pages are visited at the same time through multiprocessing, this approach is very slow and must be used only when necessary.

3) Information Extraction

At this stage, we possess the main HTML page as well as the HTTP communication messages in a raw format. Various information is extracted with the library BeautifulSoup for the following stages:

A) Redirection elements: Iframes, framesets, meta tag with attribute http-equiv=refresh and window.location object.
B) Links: Extracted from “a” and “area” tags as well as from redirection objects. Cleaned and converted to absolute urls.
C) Page Complexity indicators: Size of formatted HTML, quantity of non-trivial HTML tags, quantity of displayed words/letters.

D) Displayed text: All the visible text is extracted from titles, divs, spans, paragraphs, titles of links, as well as the non-visibile ALT attribute of images and “meta name=description” tag … Formatting objects are removed (ex: bold tag “<b>”).

E) Language: From the displayed text, we identify the main language of the website preceded by a filtering of acronyms, codes and numbers

- A simple natural language processing (NLP) library, called “langdetect” developed and open sourced by Google (Apache License 2.0), allows identifying the language of the displayed text. The recent progress in NLP provides access to more advanced approaches for this task, leaving room for future developments.

  It can identify 55 languages with the ISO 639-1 codes :af, ar, bg, bn, ca, cs, cy, da, de, el, en, es, et, fa, fi, fr, gu, he, hi, hr, hu, id, it, ja, kn, ko, lt, lv, mk, ml, mr, ne, nl, no, pa, pl,pt, ro, ru, sk, sl, so, sq, sv, sw, ta, te, th, tl, tr, uk, ur, vi, zh-cn, zh-tw. For more details on the languages, please refer to http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes .

F) Non-text: collection of all attributes and tags as well as inline Javascript

All this information is reused for the downstream stages.

4) Registrar Identification

If a link to one and only one registrar website is found, the website might be a registrar. However some registrars also provide website building services (ex: Wordpress) and other registrars have a diversified business portfolio (like Microsoft). To filter out the first category, a restriction on the quantity of non-trivial non-repetitive tags is applied. Generated websites tend to have a complex HTML structure. For the second category, a website that displays a lot of other contents must also have a parking note (cf text classification) to be considered in this category. A domain that satisfies these conditions is classified as **Parking Response Registrar.** The registrar that are considered are listed in the Table-2.

Table 1 - List of detected registrars/hosting companies

Registrars
ICANN members
CENTR members

In addition, some registrars have specific parking libraries/links (=Non-text). They usually use the keyword “park” in the naming of their variable. If such keyword is detected in a small page, the domain is considered **Parking Response Registrar**.

5) Redirection Classification

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, etc… The redirection target page is visited and used for classification. The redirection flag is raised.

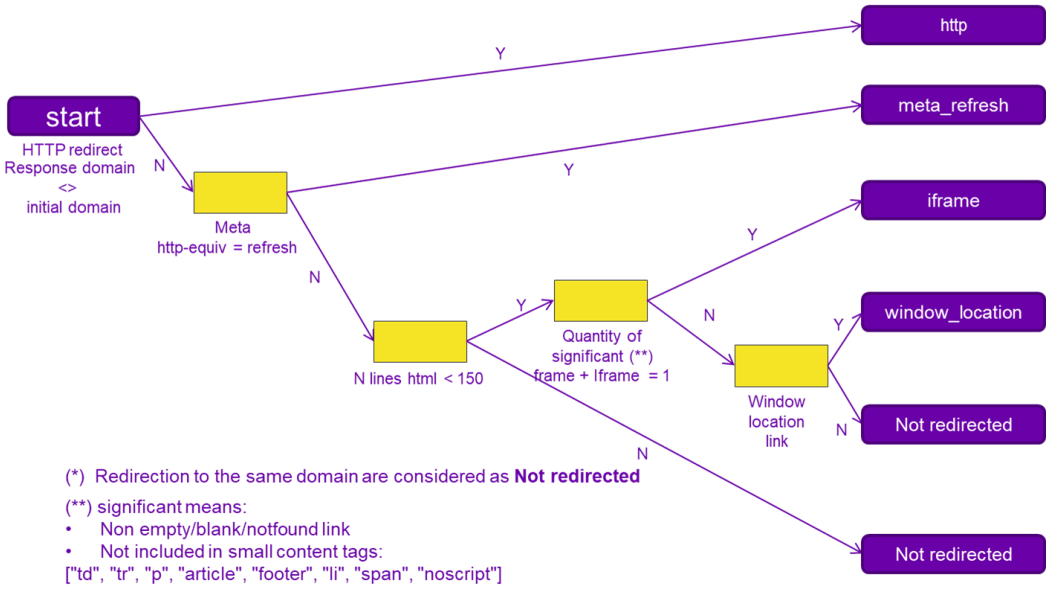


Figure 8 - Redirection tree

6) Text Classification

In short, a machine learning algorithm classify if the displayed text is a parking note or not. It takes as input the displayed text and a reference vocabulary with parking connotations. This step is described in more detail in the section III.

7) Dynamic Content Identification

When there is little displayed text, not enough for a relevant text classification, it might mean that some text is generated dynamically. A visit by browser is done in three cases:

- Displayed text size below a threshold AND high ratio of inline Javascript compared to HTML size (= length of Javascript code / length of all HTML)

- Displayed text size below a threshold AND detection of dynamic content creation functions like “document.write”, “.createElement”, “.appendChild”
- Displayed text size below a threshold AND detection of Javascript required note (usually in “noscript” tag)

If one of the conditions is met, the results compiled so far are all ignored and recomputed with the browser-rendered page.

8) Consolidation

Finally, a consolidation step is gathering all the outputs to infer the final categories, starting with errors then content page.

a) Errors

- If the HTTP Client fail to identify the IP address of a URL, the URL is categorized as **No address found**.
- If the IP address is found but the corresponding server AND/OR the client fail to connect, the URL is categorized as **Refused Connection**.
- If the IP address is found but the corresponding server is taking too long to answer, the URL is categorized as **Timeout**.
- If the server respond, but the response cannot be interpreted due to various errors like encoding, the URL is categorized as **No Status Code**
- If the server provides a response with no-content AND a non-200 status code, the URL is classified based on the status code among:

Status Code	Description
401	Unauthorized
403	Forbidden
404	Not Found
408	Request Timeout
500	Internal Server Error
502	Bad Gateway
504	Gateway Timeout

Other status codes are gathered in the category **HTTP_other**.

b) Page with content

In the following order:

- If a registrar page is identified, the domain is classified as **Parked Notice Registrar**
- If the text classification step predicts a parking note, the domain is classified according to the specific vocabulary identified in the page:

Cateogory LV4	Example of specific vocabulary
Blocked	Censored, Blocked, suspended…
Under construction	Construction, Maintenance…
Starter	Wordpress, Wix…
Expired	Expired, Deleted
Index of	Index of
For sale	Purchase, for sale, acquire…
Reserved	reserved, owned…
Parked Notice Individual Content	None of the above

- If the displayed text has less than 5 words and has no JavaScript, no frames and no links, the URL is considered as **Blank Page**.

C) TEXT CLASSIFICATION

In order to train a parking note classifier, the following steps are implemented:

- Identify relevant indicators of parking note. Parking is associated to a list vocabulary that needs to be collected.
- A classifier only takes numeric values as input. To get the most relevant numeric values called “features”, a feature engineering step is designed.
- A ground truth dataset is manually compiled.
- With the above data, the model can be trained to learn the relationship between the features and the parking note ground truth.

1) Parking vocabulary collection

There is a limited vocabulary that is associated with parking pages. With the gathered experience, 66 “root” english words have been identified like “park”, “sale”, “register”… . The detailed list is in the input file taxonomy.csv, at column root.

A same “root” word can appear in different forms due to :

- Plurality: website --> websites
- Tense: park --> parked
- POS tag: reserve (verb) --> reservation (noun)

To expend the list to all forms, we use the library word_forms.

Finally, a same word can be found in different languages, for example “domain” is “域名“ in Chinese and “домен“ in Russian. To account for all possible languages:

- Each form of each root is translated to 103 languages using Google Translate model. The detailed list is available at https://developers.google.com/admin-sdk/directory/v1/languages.
- These translations are sometimes joined with prepositions. Prepositions are examples of “stopwords” (= recurrent word of a language with basic meaning). To remove them, we trim leading and trailing stopwords from the tokenized translation (=translation text converted into a list of words).

The final list of words is provided in taxonomy.csv. Whenever a word from that list is found on a website, it is a strong indicator of parking but not always. The goal of the machine learning is to figure out when it is the case.

2) Feature Engineering

We provide different numeric values to the model:

- The count of each root words: if a page only contains “park”, “parking” and “website”, the model will be provided the following table. The relevance of these features is straightforward.

park	website	other roots		
2	1	0	0	…

- Count of root words “pair” in the same and successive sentences:

The root words are split into three categories:

1. Core words: They are words that implicitly refers to the website/page/domain like “website”, “content”, “space”, “page”, “account”, …
   The domain of a given website “abc.com” is one of those keyword: “abc.com” is replaced by “x_url”
2. Attribute words: They are words that implicitly indicates a lack of content like “parked”, “blocked”, “coming”, “unavailable”, “construction”…
3. The other words that doesn’t fit in the first two categories like “hello_world”, “index_of”, “lorem ipsum” …

The vast majority of parking notes are a combination of one core word and one attribute word in the same sentence or successive sentences like “parked page” or “Welcome to abc.com. Purchase now”. To

provide this information to the model, the text is parsed into sentences, then into words. Sentences are looped over. In each sentence we identify the core and attribute words and count the pairs. When a pair is identified, the size of the smallest sentence is also provided to the model as a way to filter pairs that occurs by chance.

For example, the text “abc.com. Purchase now. We accept cookies. this page is reserved.” will become:

n_pair_same	n_pair_successive	min_sent_size_same	min_sent_size_successive
1	1	3	1

For information, splitting a text into sentences then words is depending on the language. English have sentence separators (=points) as well as word separators (= spaces). But other languages like Chinese, Korean, Japanese or Indian languages have a more complex rationale. To achieve our goal, various NLP libraries are used to adapt to each language. For this step, so far, 8 sentence tokenizers and 16 word tokenizers are implemented. The detailed list and the corresponding languages are provided in the input file lgg_tokenizers.csv.

- Quantity of sentences and words: Often parking pages are short. In addition, the bigger the website, the more likely parking words and pairs are to be found. The size of the page will give the model a chance to filter out false positives.

In total, 72 features are provided to the model (=66+ 4 + 2)

3) Dataset Labelling

For now, 2000 websites with WHOIS location from all around the world have been visited manually. Among them, 1420 have yielded a page with non-registrar content. Among these, 472 are identified as parked with a note. Additional data would help, especially with more rare features (“censored page”, “domain not linked to a website” …).

To get them, 650 additional websites have been manually labelled. They have been selected so that each root word appears in a minimum of 10 websites. To find them, an initial large scale domain scan, with international s, has been performed.

4) Model Training

After a quick iteration on a few models (Logistic Regression, SVC and decision tree classifiers), a tree based XGBoost model has been used. It is a model ensemble of decision trees. One decision tree example is represented below. This tree split the dataset into subsets with higher purity with respect to parking class. It tries to form groups of either only non-parked page or only parked page. One split is always done with one feature and one threshold for that feature (for example, domains with more than 100 words vs domains with less than that). The choice of feature and threshold is mathematically

optimized to minimize the prediction error. XGBoost resort to multiple decision trees, trained sequentially, where one tree focus on the error of all the former trees together.

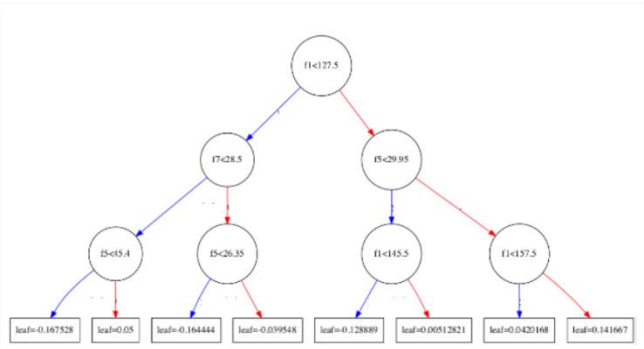


Figure 9 - Example of decision tree where "f_i" is the feature *i*

On the dataset, in cross validation, we get the following performance:

Accuracy	90.2 %		Confusion Matrix		PREDICTED	
Precision	85.3 %				not parked	parked
Recall	88.7 %		ACTUAL	not parked	1188	116
f1 score	87.0 %			parked	86	675

This trained model is integrated to the crawler pipeline.

D) PIPELINE PERFORMANCE

1016 domains have been randomly sampled from the .COM TLD and manually labelled. The labelling matches the level 2 category with three classes (High Content, Errors, Parking).

The whole pipeline predicted the classes with an accuracy of 85.7%, which correspond to 159 prediction mistakes out of 1016. They are spread as follows:

ACTUAL	PREDICT	COUNT
parked	error	50
parked	normal	44
normal	error	31
error	parked	16
normal	parked	12
error	normal	6

TOTAL	159

II- SOCIAL MEDIA ACTIVITY

1) Objective

The objective of this section is to identify how a given domain is leveraging social media like (ex: Facebook, Twitter). It can be used in three ways:

- 1) The domain has its own official social media pages. This is identified through the presence of backlinks inside the HTML page, for example in the form of an icon band like the figure below.

Figure 10 - Example of social media icon band

So far, we have been focusing on western media:

- Twitter
- Facebook
- Instagram
- Linkedin
- Reddit
- Github

Plenty of others may be implemented in the future, among them other western medias (Youtube, Whatsapp, Messenger, Tumblr, Viber, Snapchat, Pinterest, Medium ...) as well as eastern media (Wechat, QQ, Qzone, Tik Tok, Sina Weibo, Baidu Tieba, Line, Telegram …) with hundreds of millions of registered users.

- 2) The websites has adapted its content for social media display. Websites can resort to specific HTML tags to optimize display/sharing on Media. Among them:

- Twitter Cards:

```
<meta data-rh="true" name="twitter:title" content="Medium – Get smarter about what matters to you."/>
<meta data-rh="true" name="twitter:app:name:iphone" content="Medium"/>
<meta data-rh="true" name="twitter:app:id:iphone" content="828256236"/>
<meta data-rh="true" name="twitter:card" content="summary_large_image"/>
<meta data-rh="true" name="twitter:creator" content="@Medium"/>
<meta data-rh="true" name="twitter:description"
      content="Medium is not like any other platform on the internet. Our sole purpose is to
      help you find compelling ideas, knowledge, and perspectives. We don't serve ads—we serve
      you, the curious reader who loves to learn new things. Medium is home to thousands of
      independent voices, and we combine humans and technology to find the best reading for
      you—and filter out the rest."/>
<meta data-rh="true" name="twitter:image:src"
      content="https://cdn-images-1.medium.com/max/1200/1*29XAq2WrtejUCxRzSgDLXA.png"/>
<meta data-rh="true" name="twitter:site" content="@Medium"/>
```

Image 1 - Code snippet of twitter cards: indicating the key information to display in social media

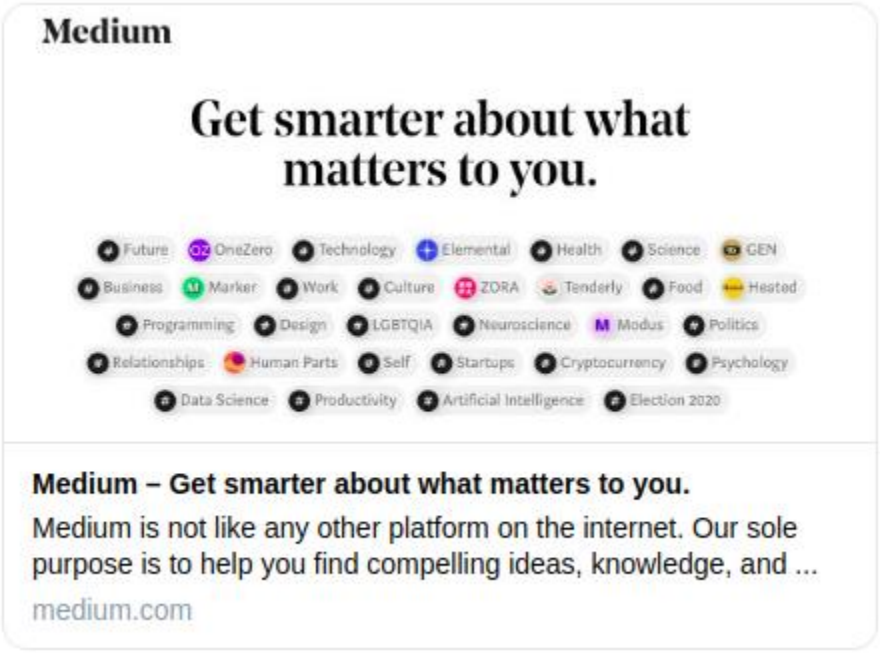


Figure 11 - The same website as diplayed on twitter whenever the link medium.com is used

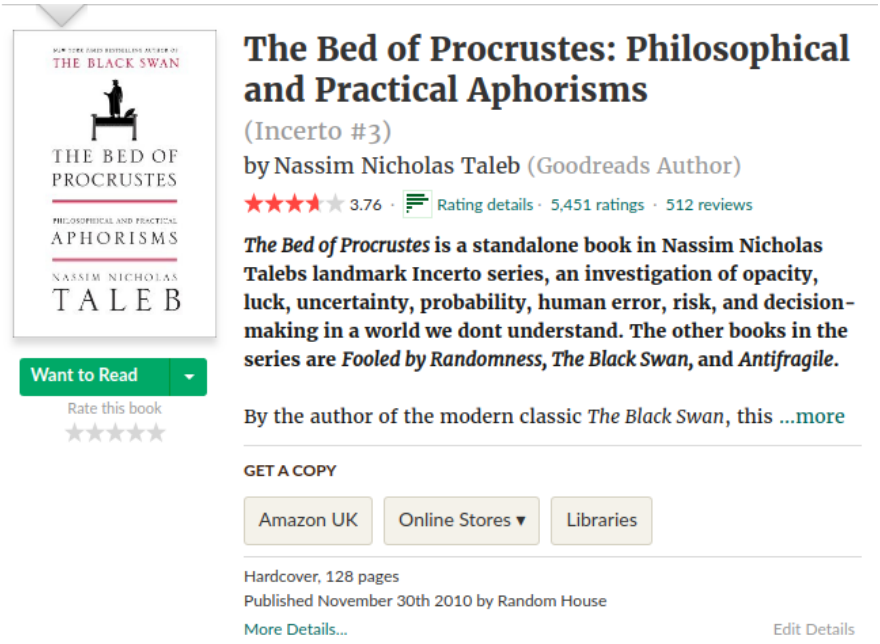
- Open Graph: Similar to Twitter cards, but developed and open-sourced by Facebook, it helps social media users understand the content of the page using only its link.

```
<meta data-rh="true" property="og:url" content="https://www.nytimes.com"/>
<meta data-rh="true" property="og:type" content="website"/>
<meta data-rh="true" property="og:title" content="Breaking News, World News &amp;amp; Multimedia"/>
<meta data-rh="true" property="og:description"
      content="The New York Times: Find breaking news, multimedia, reviews &amp;amp; opinion on Washington,
      business, sports, movies, travel, books, jobs, education, real estate, cars &amp;amp;
      more at nytimes.com."/>
<meta data-rh="true" property="og:image"
      content="https://static01.nyt.com/newsgraphics/images/icons/defaultPromoCrop.png"/>
```

Code snippet with key information: url, title, description...

Link, as displayed on social media (Facebook)

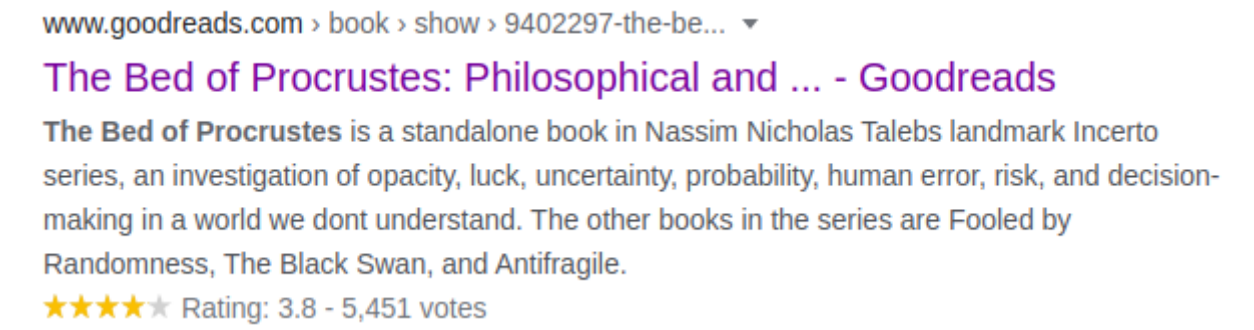
- Schema Tags: As a result of a collaboration between major search engine (Google, Bing, Yandex, and Yahoo!), schema tags helps identify key attributes of a product/service, for an enhanced display in search engine results.



The page as displayed by the website: formatting and content are, a priori, complex to identify...

```
<h1 id="book" class="gr-h1 gr-h1--serif" itemprop="name">
    The Bed of Procrustes: Philosophical and Practical Aphorisms
</h1>
<span itemprop='author' itemscope='' itemtype='http://schema.org/Person'>
<div class='authorName__container'>
<a class="authorName" itemprop="url" href="https://www.goodreads.com/author/show/21559.Nassim_Nicholas_Taleb"><span
        itemprop="name">Nassim Nicholas Taleb</span></a> <span class="greyText">(Goodreads Author)</span>
</div>
</span>
<a rel="nofollow" itemprop="image" href="/book/photo/9402297-the-bed-of-procrustes"><img
        id="coverImage"
        alt="The Bed of Procrustes: Philosophical and Practical Aphorisms"
        src="https://i.gr-assets.com/images/S/compressed.photo.goodreads.com/books/1320538208l/9402297.jpg"/>
</a>
```

... but schema tags, clearly identifies the characteristics of the book with the parameters “itemprop”...

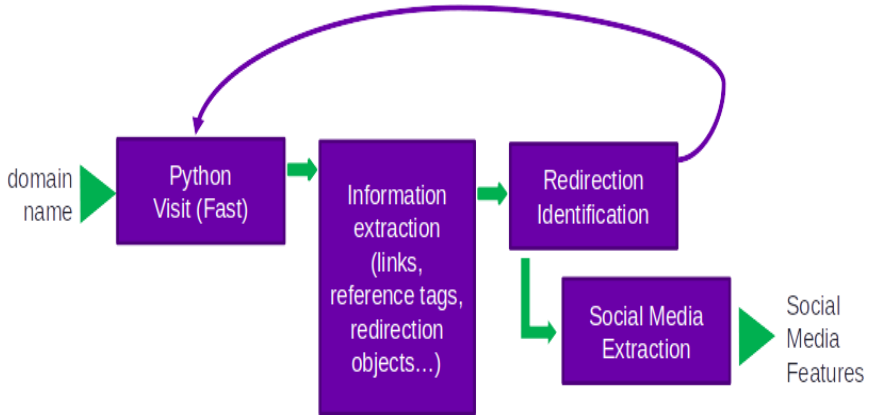


...the result on Google select and displays the most relevant information for the user (title, rating, description)

- 3) Finally, websites can resort to a sign-on service with Google, Facebook and Linkedin. With that service, the website can use the platform security of a big tech company as well as requesting for additional information from the social media accounts (eg: List of friends on Facebook…). In return, the social media company gets to know when a given user is connected to the website and adapt its services accordingly (eg: advertisement). **NOT IMPLEMENTED YET.**

2) Detailed approach

To detect those features, the following architecture is implemented.



General architecture of the social media pipeline

The fast visit and redirection identification are reused directly from the parking crawler. The information extraction is adapted to feed the new social media extraction bloc.

Additional information is extracted:

- All links to social media
- Tag information of links: type of tag (img, a …) and ancestors line in the HTML tree
- Twitter Cards, Open Graph and Schema tags

The downstream identification of social media is rather straightforward except for one challenge. There might be noisy links to any social media, which doesn't refer to the official account of the domain owner. To make sure we get them, the following logic is applied:

- We filter out libraries (eg. facebook.com/share.php), trivial page identifiers (eg. twitter.com/tr?q=abc") and normalize the remaining links (lowercase, dash and underscore removed..)
- If multiple links are found for a single social media (occurs rarely), a custom logic is implemented:
    - We select the name that matches the domain name
    - If it doesn't match, we select the name that is included in the domain name or that includes the latter.
    - If it doesn't, the name with the smallest Levenshtein distance to the domain name is selected (= number of letters to add/change/remove to go from the domain to the link name)

III- MAIL SERVER IDENTIFICATION

1) Objective

If a domain doesn't return a website, it doesn't necessarily mean it is not used. It can also be associated to a mail server. Our objective is to know which one does.

A mail is sent through a protocol called SMTP (Simple Mail Transfer Protocol), different than HTTP(S) used in the previous section. When sending a mail at name@domain.com, this protocol starts with a query to the DNS server of .COM. The DNS returns the MX record of "domain.com" (MX= Mail eXchange). This MX records contains the list of domains of mail exchange servers. These domains can have the same domain as the initial mail (eg, mail0.domain.com) or an external one (eg. mail1.yahoo.com for servers hosted at Yahoo, mail2.live.com for Microsoft). From there, the mail content can be sent to the mail server that will attribute it to the correct final address according to the left side of the e-mail.

Two key information can be extracted for a given input domain:

- Whether a domain has a mail service attached ie. if it has an MX record.
- Whether it manages its own mail server ie. if the mail servers have the same domain as the input domain

2) Approach

To get this information, a different but simpler pipeline is implemented. Each domain is queried for MX records using dnspython library. The first mail address in the priority list is selected to compare with the input domain.

IV- LIMITS AND IMPROVEMENT DIRECTIONS

1) Complex dynamic websites

Some parked pages generate parking notes using non-explicit libraries (ex: 1234.js) hidden in one or more level of libraries (i.e. the main page calling a library, that calls another library, which writes the parking note). The generated text is not hard-coded even in the browser-rendered page.

Specific technologies like .aspx can also trick the crawler into thinking that there is no displayed text. The use of such libraries could be integrated into the dynamic page identification step.

2) Dataset size

The current machine learning for text classification could benefit from a more diversified dataset, especially for rare parking notes. Further tuning will come as more rare notes are collected.

3) 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 as well as ALT parameter to classify the page. An OCR extension could be developed to extract the text of websites with only one image.