Wykop.pl – Webscrapping using BS4, Selenium, Scrapy

1. Introduction

The site **wykop.pl** is regarded (and rightly so) as a forum where an atmosphere of heckling and name-calling prevails. Nevertheless, the main page of wykop contains articles on current news from Poland and the whole World. The site is updated practically every minute, so by spending 15-20 minutes at evening, you can get reliable news of the day. In our work, we decided to use popular webscrapping libraries, so that we could check what topics currently dominate on the site, what hashtags are the most used and what people think about each article based on their reactions.

Unfortunately, due to the restrictive policy of the wykop.pl website, scraping a very large number of articles is impossible. **Therefore, the scope of our analysis will be limited to scraping 120 pages of the site, where from each subpage we will only select the first article that is on the page**. Once we are inside the article, we try to scrape the following things from it:

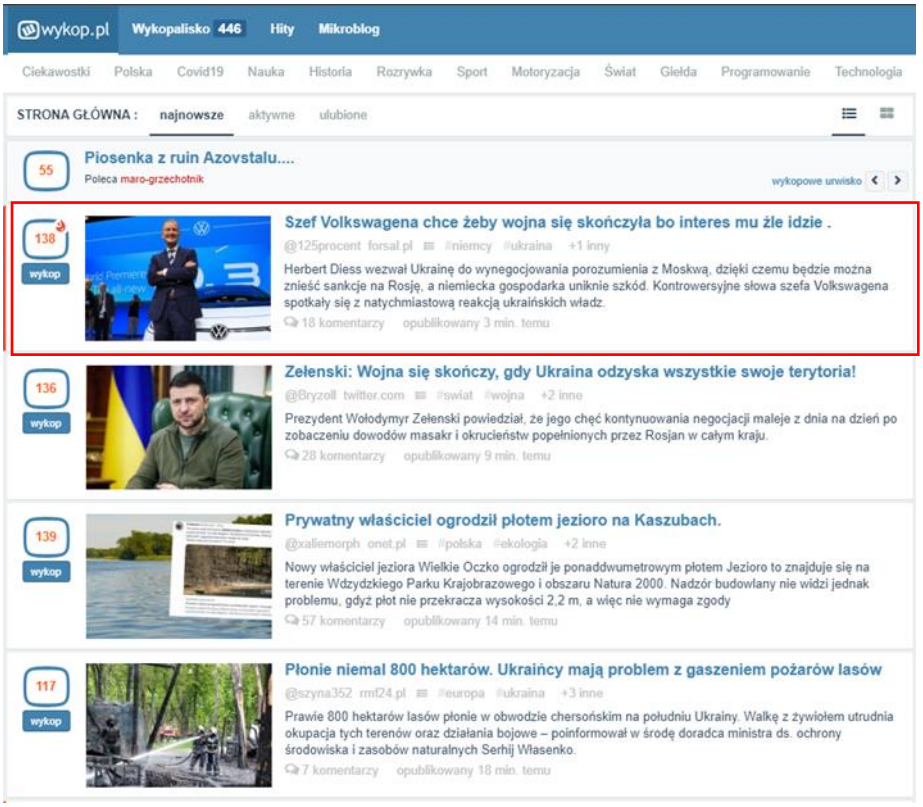
- title
- the nickname of the user who posted the article
- the number of likes under the article
- the number of dislikes
- number of views
- all the hashtags that have been placed under the article

We would like to **warn you in advance** that the final number of records you receive may vary slightly from 120, but will certainly be higher than the lower limit required to pass the course (i.e. 100). We have discovered two reasons for this:

- the first article may be an advertisement, which when clicked on, automatically leaves wykop.pl and goes to the advertiser's website

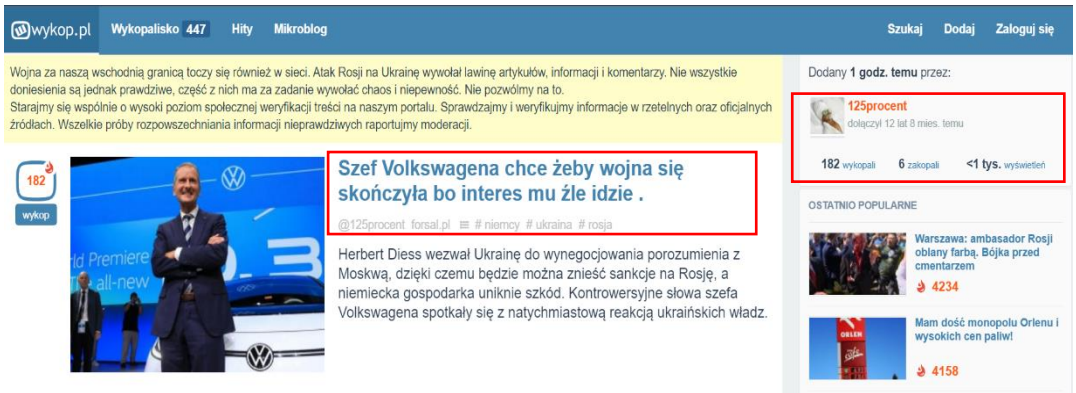
-the first article, may also contain content allowed only for persons over 18 years of age, which requires a prior account creation and proof of age.

Furthermore, the site is **updated on a regular basis**, so running through the codes at different time intervals will **undoubtedly give us different results**. Nevertheless, below is a brief roadmap of how our tool should work in general.



Select only the first article from each subpage.

From both red boxes, scrape those things that will be useful for further analysis.



2. Description of coding parts

2.1. BeautifulSoup

[Script name BS_final.py.]

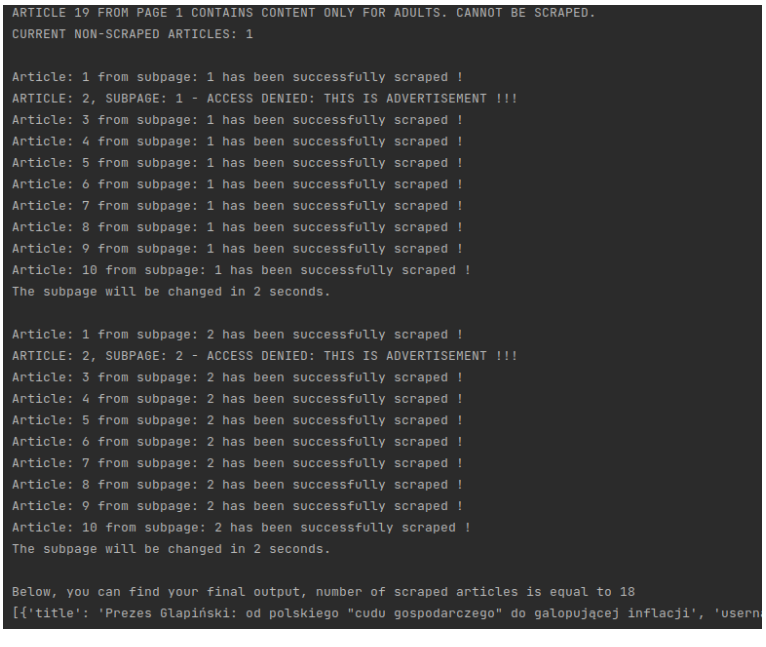
At the very beginning we put two parameters to change. The first one *START_100_SCRAPE* is boolean and setting it to True, will make our scraper start retrieving information from the next 120 pages. In case of parameter *SET_PAGES_TO_SCRAPE_PER_ONE_LOOP*, we can control how many links to articles our scraper should scrape. Default parameter is equal to 1, but it can be set to a maximum of 35. However, setting this parameter so high would result in 4200 requests, which would cause our connection to be broken by the anti-scraping program.

Our scraper starts by creating an empty list where we will store our results. Next, we create a loop which goes through the following subpages of the wykop.pl/strona/{number}. Inside the function we save to the links variable all articles found in a given tab. After that, we are looping over the values of the links variable which opens a link to each article from the subpage. Once we are inside the article, we try to scrape our 6 variables of interest.

Finally, the results are stored in the *post_summary* dictionary, which is then added to the *list_of_result*, containing all the results scraped so far. We also include a 10 second pause between subpage changes.

Below, we've included an example of the result for our scraper. In order to demonstrate how our artificial log works, we have changed the range of scraped pages to 2 on one occasion and the range of scraped articles per page to 10.

Figure 1. Example output from tool based on beautifulsoup library.



2.2. Selenium

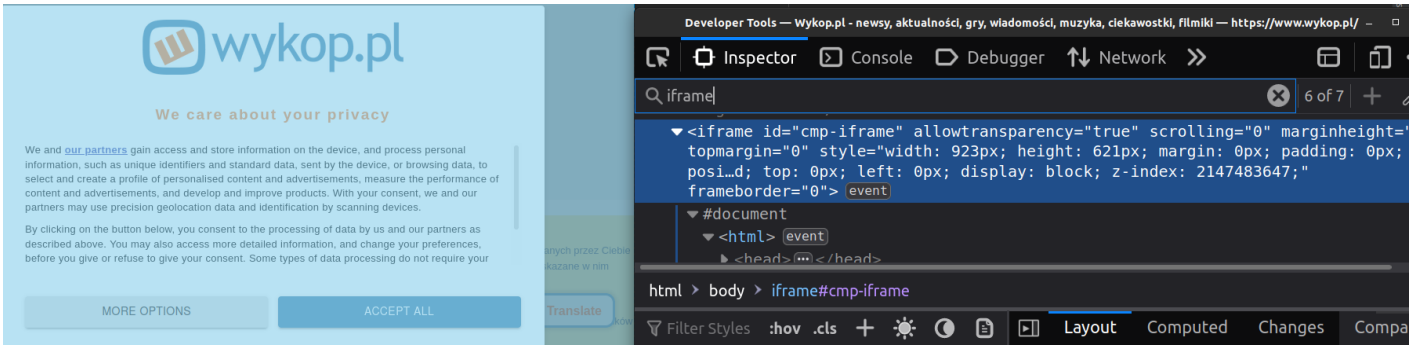
[Script name selenium_wykop.py.]

As with scrapping through the beautiful soup, we have entered the same two launch parameters (boolean: *START_100_SCRAPE*, integer: *SET_PAGES_TO_SCRAPE_PER_ONE_LOOP)* that define the number of scrapped pages and articles.

Initially, when we start firefox geckodriver, we get a pop-up about accepting cookies. Acceptance of this, is important because the window is in a special "iframe". In a nutshell, the Iframe HTML element (<iframe>) represents a nested browsing context (en-US) by embedding another HTML page in the current one.

In our case, if you don't click on anything, the page doesn't allow you to continue browsing the site. This required us to change the format of the page to the "frame" structure, accept cookies and then return to the default character of the page (driver.switch_to.default_content()). After this step, we additionally accept privacy politics, located at the bottom of the page.

Figure 2. Wykop's iframe element.



Moving on to the scraping process itself, we have created a loop within a loop that goes through each article from a given tab and initially checks if the article is an advertisement (variable *check_adv_txt*). If not, the scraping process begins and the same things already mentioned are retrieved, otherwise the article will be skipped. The results obtained, are written to the *post_summary* dictionary and then added to the *list_of_result*. After a 2-second pause, our tool returns to the previous page. If the limit of scraped articles for a tab is reached, after 5 seconds, the program automatically clicks the "next" button and the user is redirected to the next page.

2.3. Scrapy

[Script name scrapy_wykop.py.]

Unlike the previous libraries, with scrapy we leave only one boolean control parameter which controls the number of scraped pages.

Firstly, we instantiate the *Wykop* class, which defines which parameters from the page we expect to be scraped. Next, we create another class - *WykopSpider*, which will send the requests to subsequent pages of the wykop.pl/strona/ website.

Again, when we open each subpage, the program will select only the FIRST ARTICLE that appears on the page and then extract the information that we are interested in. Obviously, we are aware that this first article may be an advertisement or an 18+ link (which will not be scraped). That's why, the whole loop has been encapsulated in the try function that will return an empty field if the data cannot be scraped. After calling the appropriate command (scrapy crawl wykop -o wykop.csv), we should get a csv file in our working directory, which in a later stage of our work was used for a short data analysis.

Below, we show the first 5 records of our examplary desired output.

Figure 3. Scrapy csv output.

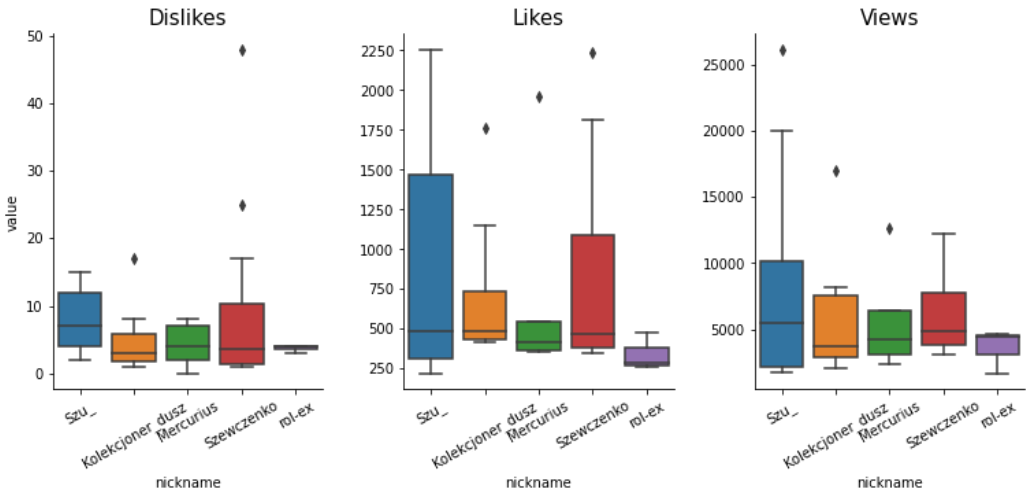
	dislikes	hashtags	likes	nickname	title	views
0	0	technologia,nauka,zainteresowania,kosmos,mars,...	186	elektryk91	NASA ma problem z Ingenuity, marsjańskim wirol...	2.4 tys.
1	0	rosja,historia,polityka,zbrodniekomunistyczne	86	Qtasus2Wielgus	Badacz zbrodni NKWD został wysłany do łagru	<1 tys.
2	14	polska,chlewobsranygownem	269	BennyHarvey	Łódzkie. Kradł znicze wrzucając do automatu st...	12.5 tys.
3	23	polska,kultura,rozrywka	359	Wirtualnemedia_pl	Polsat zrealizuje polską wersję „Sługi Narodu"	7.9 tys.
4	39	polska,technologia,amazon,afera	476	Mirxar	Strach coś zamawiać na Amazonie. Polacy płacą ...	13.4 tys.

3. Exploratory Data Analysis

[Script name wykop_analysis.ipynb]

For our analysis we used 120 articles which are the first articles from 120 pages (as of 11.05.2022, 21.30). First step, aims to show the distribution of dislikes, likes and views per top 5 most frequent users.

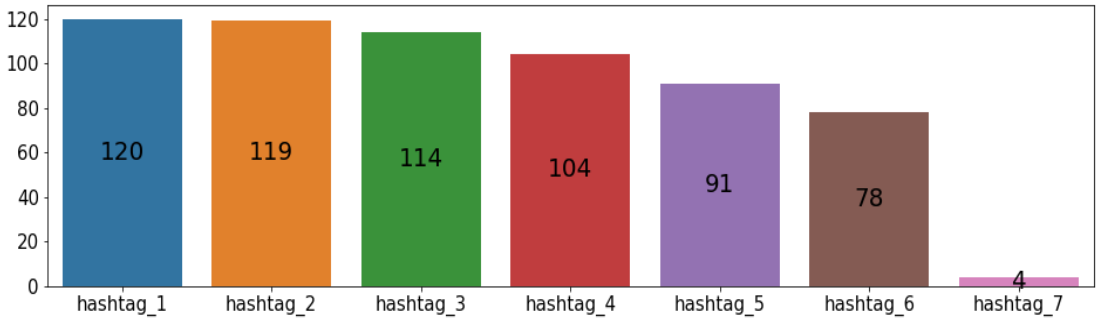
Figure 4. Boxplots for top 5 users.



From the charts above, we can see that the posts of two users, stand out from the others in terms of the number of responses they received. However, for all users, the median scores across all variables are similar.

The second step was to check the frequency of hashtag usage per article.

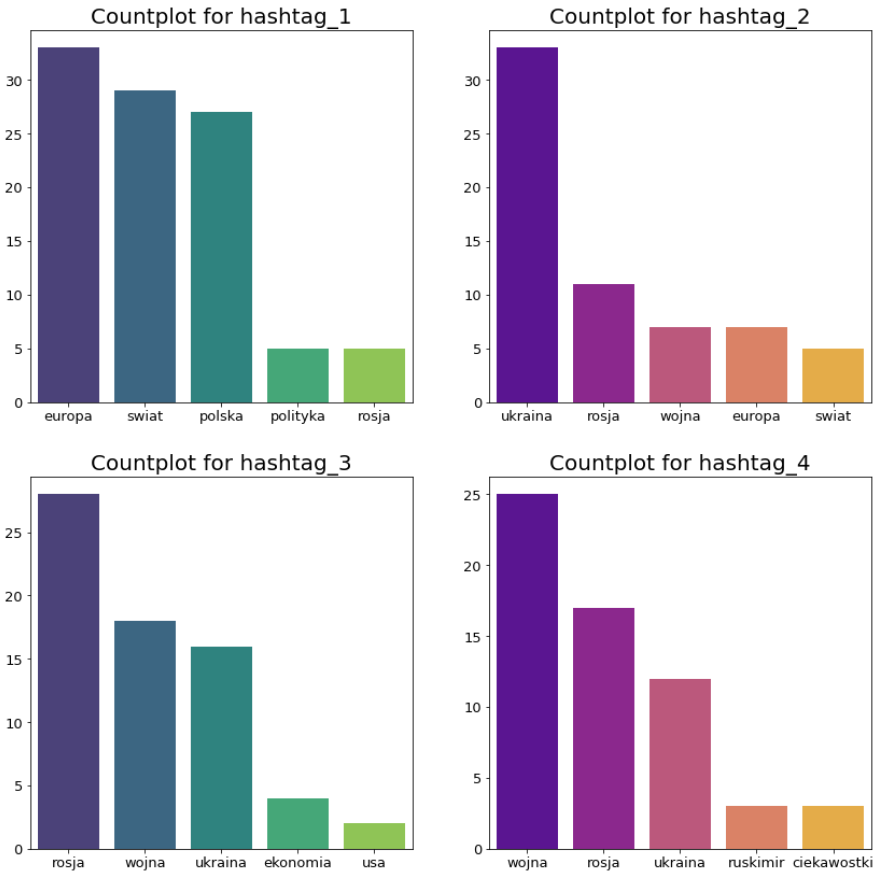
Figure 5. Distribution of hashtags used.



From the graph above, we can see that at least one hashtag was used for all articles. We can therefore assume that this is a restriction of the portal. Interestingly, for the scraped articles, the average hashtag used is around 5.

In order to find out what topics are currently on trend, we checked the relevance of the top 5 words for 4 hashtags.

Figure 6. Top five hashtags.



From the graph above we can see that the articles were mostly about Europe and in particular the war between Russia and Ukraine.

4. Conclusions

In our work, we collected data on articles that are found on wykop.pl. For this purpose, we used three different libraries. We believe that the aim of the work, i.e. to retrieve information from at least 100 consecutive pages, has been achieved. The quality of the data is good enough that we were able to carry out a short analysis of the data, which allowed us to find out what news from the world are currently on top.

Should you wish to continue the work, we suggest that you also download descriptions for each article, which could be used to build an NLP model. Another step worth considering is to try to circumvent the anti-scrap system of the wykop.pl website (in the good spirit of the law, of course). This would make it possible, without any restrictions, to scrape every article from the site.