Project Urbandictionary.com - Description

1. Team members

Name	Student ID
Michał Stryjek	348899
Szymon Zientalak	447382
Dustin Pacholleck	437968

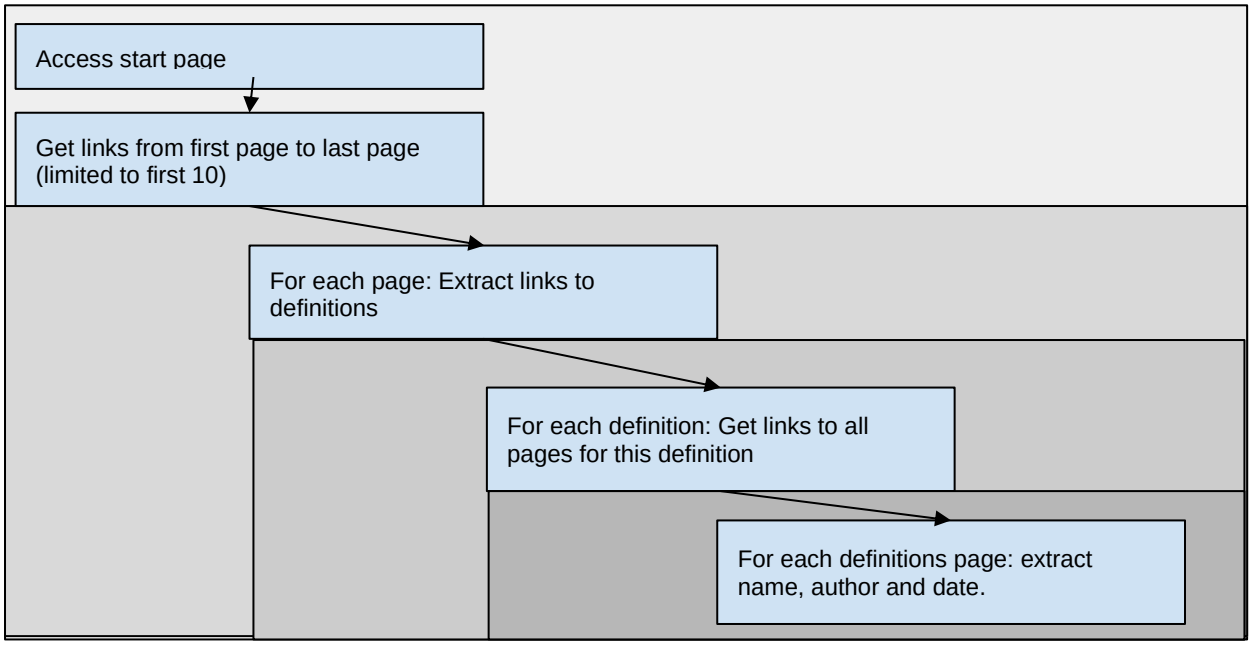
2. Topic and page description

The page scraped is urbandictionary.com. It is a page containing a description of slang terms used in English. The descriptions are provided mostly in a joking manner, sometimes being a joke itself. Each word has its dedicated page containing sometimes multiple definitions. When the number of definitions is too large it will be split across multiple pages. The main page changes daily and depicts random definitions. Those definitions span across multiple pages, going up to around 850 at the time of this document creation.

The topic of the analysis is checking the popularity of the words from the main page and activity authors of the definitions across time. It could show what topics were relevant at the time in the Internet community.

3. Scraper mechanics

The general approach of the scrapers looks like in the chart below.



Length of the output table is limited by page limiter and main page limiter. Just for data analysis, the main page limiter was set to 50. From those 50 pages word definition links were scraped. Next, total 1000 pages from those links were scraped. By default main page limiter is set to 10, total page limiter is set to 90 for total 100 pages.

Nevertheless, each of the scrapers has to go a different way to achieve this:

Beautiful Soup

To start, Beautiful Soup creates an empty data frame and checks the limit_pages variable to determine whether it should scrape 100 pages or a different, potentially much higher number of them. After that it accesses the main page of Urban Dictionary from where it extracts a link to every main word using the html parser in combination with regex, and appends it to a list by combining the main part of the URL and the parsed href, then proceeds to the next page by iterating over the number of pages stored in a separate variable to repeat the process until a limit of pages is reached. From there, it starts a loop which opens a link stored in the previously created list of links and loops through every page of definitions to scrape the author and date of creation, along with the word in question, and appends it to the data frame created at the very beginning, then proceeds to the next link. Lastly, the script exports the data frame in the form of a csv file, containing all scraped data.

Scrapy

First, scrapy identifies at the start page the maximum number of existing pages to iterate through. Having this number allows for iterating over the pages from 1 to the extracted maximum page since the link has the following format: *https://www.urbandictionary.com/?page=**1***. Here we store the links in an intermediate csv file for further processing.
From each of those pages, the scraper extracts the links to the words, also stored in a csv.
As a next step, the scraper has to work in the same principle as in the first step to iterate over all the pages per definition. This is since a word can have multiple pages of definitions and the link looks as follows: *https://www.urbandictionary.com/define.php?term=rediculous&page=**1***.
The resulting links are again stored in csv.
Finally, the scraper can extract from each of the definition pages per definition the word, author, and date. The final output gets exported as csv.

Selenium

Selenium goes through the first pages from the domain, iterating pages using *?page=#* parameter in the url. The value was hard coded as 10 in order to provide a way to implement a 100 page limiter. From those pages links to every word are taken from the headers and saved to the list. Next, by iterating through this list, the program goes to every of those links opening definition pages. On those pages, similarly to the main page, the function iterates through *?page#* parameter, but doesn't stop at 10. Instead it goes to the last page, which number is read from the last page link If the link is not present it means there is only one page of definitions of this particular word and the program continues to the next word. For each of those pages the program calls *getcontent()* function that retrieves the necessary div objects. Then it saves appropriate data into the dictionary, and appends it to the pandas dataframe. If the 100 pages limiter is turned on and limit is reached the program breaks the loop and finishes immediately. Otherwise the page limit is set in the "Settings" section of the code.At the end the Pandas dataframe exports the data into the "Output.csv" file.

4. Output description

The output is a CSV table with a separate entry for each definition. As displayed below, the data in the table consists of the defined word, the author of the entry and the date of the submission of the definition.

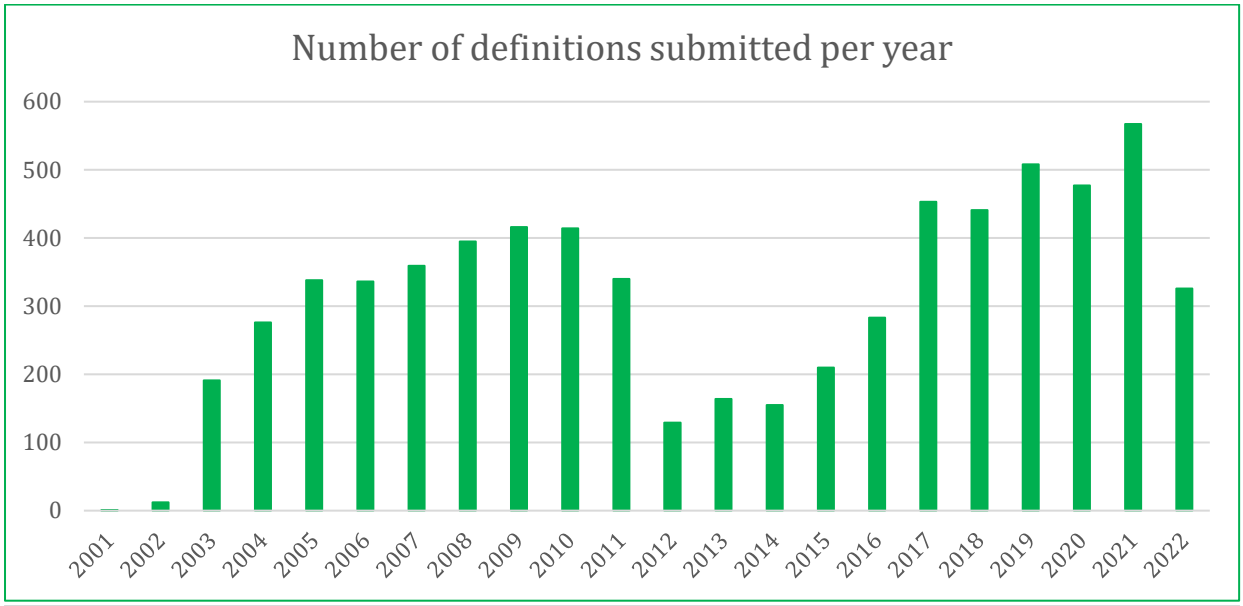
Word	Author	Date
…	…	…

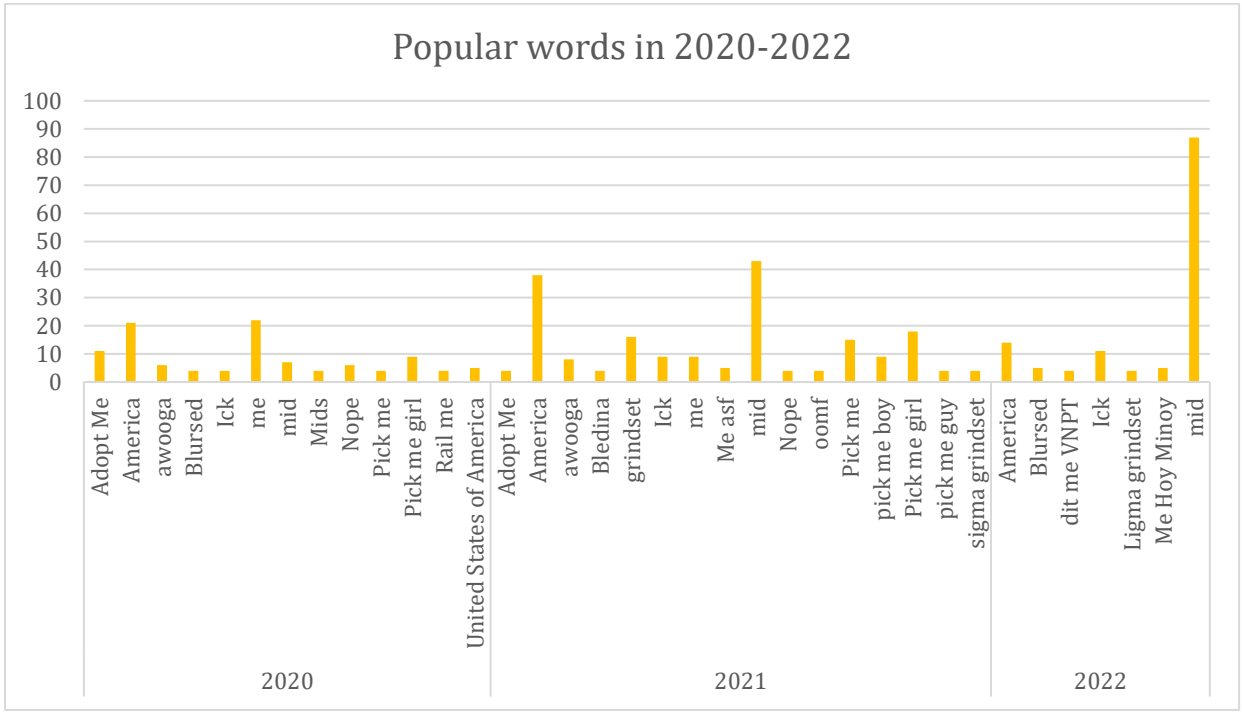
5. Scrapers comparison

Using Beautiful Soup was a task by itself so it was decided not to use it while using Selenium. Instead, objects were received using XPaths. Unfortunately, due to the rough way the page is written, using the XPath with Selenium as well as with Scrapy proved to be problematic. Using Scrapy turned out to be the fastest, with Beautiful Soup in second place. However, what Beautiful Soup lacked in speed it gained in simplicity. Parsing HTML using BS, gave less trouble and errors during the writing of the code. Selenium lacked both speed and simplicity. For a static page like this, with no mocking of user interaction necessary, it proved to be the most unsuitable tool for the task.

6. Data analysis

Output table was converted to charts to illustrate trends on page activity thorough time.





First chart shows number of definitions submitted per year, which can show popularity of the page across the years. The second chart shows words that were submitted by more than 3 authors in years 2020-2022. This shows trend in slang language in those years, as words with more definitions are more likely more popular.

7. Distribution of Work

The work was distributed by tool:

- Beautiful Soup: Szymon Zientalak
- Scrapy: Dustin Pacholleck
- Selenium: Michał Stryjek
-

All project members worked jointly on the project description and the other formal requirements. Descriptions of each scraper were provided by the person responsible for the code of their respective scraper.