News Scrapper

Run install.sh on first run
To run flask server run news_server.sh
To run scrapper run news_feed.sh

The module works for finance.yahoo and timesofindia. Support for more sites can be added by passing a configuration file to Scrapper class.

To make a configuration file for other site, follow given instructions:-
A configuration file has two things that are passed to scrapper class as arguments.

1) A list which contains the name of information that is to be extracted for a single article during scrapping.NOTE:- Do not include ‘current_date’ or ‘security’ field.

2) A dictionary which contains details of where to look for the data in html that is extracted from the webpage. The parameters are described below:

    a) upload_to_mongo: Data will only be uploaded to Mongo if this is set to ‘True’. mongoURI will be fetched from constants.py file from the current directory
    b) base_url: base url for the site which will be used to make search queries later on.
    c) make_search_url: This is a function that takes two parameters as input. First base url of site and search query and return the final url that can be used to fetch search results. This function has to be implemented by the user as different sites have different search query url strings so it’s not possible to generalize it.
    d) search: This contains two elements of type dictionary which will be used to extract needed information from the search page.
        i) container: It contains two fields in a dictionary format as is used to fetch individual containers incorporating articles. NOTE:- This fetches all results from the search

page so make sure the parameters are consistent among them. Regex string can also be used for matching in options

(a) tag: The html tag that is used to wrap around the single article in the search page.

(b) options: This is also a dictionary that specifies certain unique property like class name or other html property of the html tag that has to be extracted.

ii) links: Contains three fields in dictionary format. This is used to fetch links of articles and this is applied to each container extracted from the previous filter.
   (a) Tag: The html tag that is used to wrap around the single article link. Generally it is ‘a’ tag but one can change it according to needs.
   (b) options: This is also a dictionary that specifies certain unique property like class name or other html property of the html tag that has to be extracted.
   (c) property: Name of the html property to extract from the specified tag. Generally it would be ‘href’ for ‘a’ tags.

iii) pagination: Boolean value to specify whether the search page has pagination or not.

iv) pages: If the search page has pagination, this section will be used to extract total pages. This is also a dictionary type.
   (a) tag
   (b) options
   (c) html_property: html property to be extracted
   (d) custom: This is optional. If page data is not available in the html property of some tag

then one can use a function to extract the info. It takes a single argument which contains the html of the search page. The input is already passed to BeautifulSoup parser so all its functions are available without parsing it again.

e) article_container: This is used to extract the containers that include article details. It has a tag and options field as described by previously. It includes one extra field:-
    (a) single: Boolean value to indicate whether to extract only the first article in page or all articles that match the tag and options filter specified.

f) js : Boolean value to specify whether the webpage returns html or javascript. Since some sites return javascript instead of html to reduce bots from scraping the site so it is needed to process the javascript returned to extract the page html. Chrome headless is used for the same.

g) Also include the fields specified in Point 1. And include the following details. NOTE:- Do not include ‘current_date’ or ‘security’ field.
    (a) tag:- The tag which wraps around the specified info.
    (b) options: Something unique html property to search in tag to find accurate result. Dictionary format
    (c) single: Boolean value to specify whether to extract first matching in case of True or all in case of False.
    (d) filter: A custom function that takes a result text extracted. It is optional. Has to be implemented by user.

(e) html_property: If the required info is a html property then specify the name of it. So instead of searching the text it will extract the html property. It is optional and only applied if ‘single’ is True
(f) is_domain: If the extracted text is a website then setting this option to true will extract the domain name. It is optional and only works with ‘single’ set to True

The module takes 3 arguments:-

1) symbols: dictionary which maps search query parameter to symbols
2) list from Point 1
3) options of type dictionary from Point 2

After making the object of the Scrapper class with the above 3 arguments. Start the process with object_name.start()

To process things faster make a init function at the end which will contain the object and call object_name.start() from here

Now import this file in init.py file in scrapper directory and call module_name.init(document., ‘column_name’, destination) with 3 arguments.

1) document : pickle file loaded in pandas dataframe
2) column name within previous pandas dataframe which will be used for making the search query
3) destination:- will be extracted from cmd line arguments(--root_dir=destination) from run_feed.sh file. So pass the argument as it is.

Enter the MongoURI of form given below in a constants.py file in the project that will be used to upload data.
Ex:- mongodb://<username>:<password>@<connection_url>
Ex:-mongodb+srv://<username>:<password>@<connection_url>