International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT2174120

Web Crawling on News Web Page using Different Frameworks

Harshala Bhoir*[1], K. Jayamalini[2]

[1]Department of Computer Engineering, Shree. L. R. Tiwari College of Engineering, Mira Road(E), Thane, Maharashtra, India

[2]Assistant Professor, Department of Computer Engineering, Shree. L. R. Tiwari College of Engineering, Mira Road(E), Thane, Maharashtra, India

Article Info
Volume 7, Issue 4
Page Number: 513-519

Publication Issue :
July-August-2021

Article History
Accepted : 07 Aug 2021
Published : 14 Aug 2021

ABSTRACT

Now a days Internet is widely used by users to find required information. Searching on web for useful information has become more difficult. Web crawler helps to extract the relevant and irrelevant links from the web. Web crawler downloads web pages through the program. This paper implements web crawler with Scrapy and Beautiful Soup python web crawler framework to crawls news on news web sites.Scrapy is a web crawling framework that allow programmer to create spider that define how a certain site or a group of sites will be scraped. It has built-in support for extracting data from HTML sources using XPath expression and CSS expression. BeautifulSoup is a framework that extract data from web pages. Beautiful Soup provides a few simple methods for navigating, searching and modifying a parse tree. BeautifulSoup automatically convert incoming document to Unicode and outgoing document to UTF-8.Proposed system use BeautifulSoup and scrapy framework to crawls news web sites. This paper also compares scrapy and beautiful Soup4 web crawler frameworks.

Keywords: Scrapy , BeautifulSoup ,Web Crawler , XPath

I. INTRODUCTION

A crawler is a program that visits Web sites and reads their web pages and other information to create entries for a search engine index. To find the information search engine uses the web crawler. Crawler starts collecting the set of URLs, called seed URLs. It retrieves the pages identified by the seed URLs, extracts all URLs in the pages, and adds the new URLs to a queue called the crawl frontier for scanning. Then the crawler gets URLs from the queue, and repeats the process. Mainly crawlers are used to create a copy of all the visited pages for processing by a search engine and also index the downloaded pages to provide fast searches. Many legitimate sites, in particular search engines, use crawling as a means of providing up-to-date data.

A Search Engine Spider is a program that most search engines use to find what new and useful on the Internet. Google's web crawler is known as Google Bot. The Bot is actually "crawls" the web and collects documents to build index for the different search engines. There are basically three steps that are involved in working of web crawler. First, the search bot starts by crawling pages of web site. Then it indexing the words and content of the site, and finally it visit URLs which is on site. When the spider doesn't find a page, it will be deleted from the index.

The working step of a web crawler is as follows [5]:

1 Initialize the seed URL.
2 Adding URLs to the frontier
3 Selecting these URL from the frontier
4 Fetching the web-page related to that URLs
5 Parsing retrieved web page to extract the URLs
6 Adding all unvisited links to the list of URL
7 Repeat step 2 till the frontier is empty.

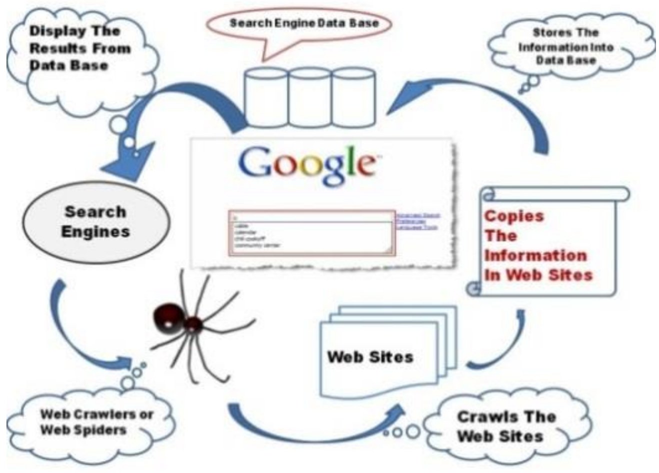

Display The Results From Data Base
Search Engine Data Base
Stores The Information Into Data Base
Google
Search Engines
Copies The Information In Web Sites
Web Sites
Web Crawlers or Web Spiders
Crawls The Web Sites

Figure 1 Working Of Web Crawler [5]

1.1 Scrapy framework

Scrapy is a framework use to crawl web sites and extracting data that can be used for various applications.

Scrapy is implemented in Python, which allowed for easy extension of the framework to suit our specific web-crawling applications, and allows for multi-platform support. To illustrate Scrapy's execution, refer to Figure 2.Scrapy uses object classes called spiders, as shown in Figure 2 item 7, to allow for customized crawling per website. An automatic chocking of the spiders is required for them to be friendly to the sources being crawled. If the spiders are not friendly, then they would quickly be blocked from the source and the crawl would fail. The scheduler shown in Figure 2 [3] handled the automated chocking.

Though Scrapy was originally designed for web scraping, it can also be used to extract data using APIs. We can get HTML source code easily through Scrapy Downloader. The Downloader of Scrapy can access to web data according to the specified URL. It communicates with the Internet through the HTTP protocol. The Scrapy Engine controls the data flow in Scrapy. The Engine gets URLs to scrape from the Spider and schedules them in the Scheduler, as Requests. Then, the Engine asks the Scheduler for URLs to crawl, as Requests, and send them to the Downloader, passing through the Downloader Middleware. Once the page finishes downloading, the Downloader generates a Response and sends it to the Engine [5]. The Response exactly contains the HTML source code. And the Spider can deal with the Response. Information is saved in the Items. You can transform the Items into text files or CSV files and store them in a database through the Item Pipeline. The process of data flowing is shown in Fig. 2.

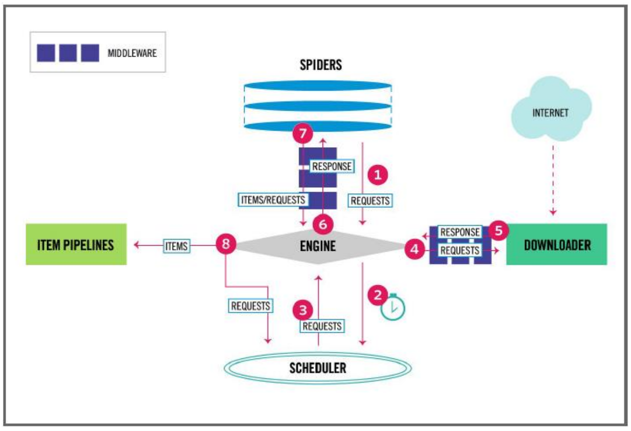


Figure 2 Scrapy Architecture Diagram[3]

Features :

- Fast and powerful
- Easily extensible

- Built-in support for selecting and extracting data from HTML/XML sources using extended CSS selectors and XPath expressions, with helper methods to extract using regular expressions.
- Portable – It is written in Python and it runs on Linux, Windows, Mac and BSD.
- It has Interactive shell console
- Robust encoding support and auto-detection, for dealing with foreign, non-standard and broken encoding declarations.
- Strong extensibility support
- Wide range of built-in extensions and middlewares
- DNS resolver

1.2 BeautifulSoup Framework

Beautiful Soup is a Python framework and it is used for extracting, analyzing, and editing data in the DOM (document object model) tree of webpages. It supplies a series of concise DOM visitor interfaces, helping developers quickly build a system prototype and obtain experiment data. Additionally, it has high cross-platform flexibility [2].

It will analyze any document passed to it, including all kinds of HTML or XML documents and normal documents. But only with HTML and XML documents that meet specifications will Beautiful Soup generate a corresponding DOM tree and carry out a series of API calls to obtain the designated data. Using HTML as an example, Beautiful Soup first uses a HTML analytics engine to convert the HTML document into a DOM tree, and then the user can use Beautiful Soup's supplied inquiry and editing functions to operate a DOM tree.

Features:

- Beautiful Soup provides methods for navigating, searching, and modifying a parse tree. It is a toolkit for dissecting a document and extracting useful information. It requires less code to write an application
- Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8.
- Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

II. LITERATURE REVIEW

Web crawlers also are known as spiders. Chakrabarti et al. [8] coined the term Crawler and the crawler mechanism proposed by them used a classifier within the crawl frontier. The term centered Crawler has truly became a word of Topical Crawler and is additional wide used. To fetch topic related information search engine use the web crawler. A Search Engine Spider (also known as a crawler, Robot, Search Bot or simply a Bot) is a program that most search engines use to find what new and useful on the Internet. Google's web crawler is known as Google Bot[5]. AnujaLawankar and Nikhil Mangrulkar [5] gives review on optimizing web crawling techniques.They also explain working of web crawler. Ernesto Cortes et al.[3] proposed architecture of Scrapy . Chunmei et al. [2] describe process of extracting, analyzing, and editing information in the DOM tree of webpages using BeautifulSoup. Pratiksha Ashiwal et al.[7] developed method for retrieving web information using BeautifulSoup and python script . Zejian Shi et al. [10] implemented an intelligent dynamic crawler, which stored the data extraction rule of XPath in database, loaded the rules dynamically according to the target, and used TF-IDF method to calculate the relevance.

III. Proposed system

In this project I uses scrapy and beautifulsoup web crawler framework for crawling news pages. Beautiful Soup provides a few simple methods for navigating, searching and modifying a parse tree.

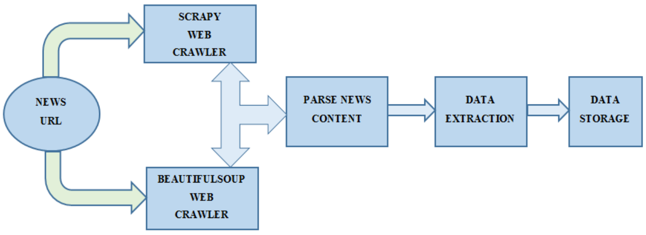


Figure 3 Process of news crawler

BeautifulSoup automatically convert incoming document to Unicode and outgoing document to UTF-8.Proposed system use BeautifulSoup 4 and Requests on system with Python 2.7 installed.

This system hits url and read data from news html pages and collects the data. Crawler then parses the news content and extracts the data. Extracted data then stored in to mongo db. You can see process of news crawling in Fig. 3.

3.1 comparative study of beautifulsoup and scrapy framework

Scrapy is a web crawling framework for developer to write code to create spider, which define how a certain site or a group of sites will be scraped. The biggest feature is that it is built on Twisted, an asynchronous networking library, so Scrapy is implemented using a non-blocking (aka asynchronous) code for concurrency, which makes the spider performance is very great. Scrapy also works fine on Python 2 and Python 3, so compatibility will not be a problem. It has built-in support for extracting data from HTML sources using XPath expression and CSS expression.

BeautifulSoup is a tool which help programmer quickly extract valid data from web pages, its API is very friendly to newbie developer, and it can also handle malformed markup very well. However, in most cases, BeautifulSoup alone cannot get the job done, you need use another package such as urlib2 or requests to help you download the web page and then you can use BeautifulSoup to parse the HTML source code. The doc of BeautifulSoup is very comprehensive you can get a lot of examples there and quickly learn how to use it.BeautifulSoup works fine on Python 2 and Python 3, so compatibility will not be a problem

The two Python web scraping tools are created to do different jobs. BeautifulSoup is only used to parse HTML and extract data, Scrapy is used to download HTML, process data and save it.

TABLE 1 COMPARISION OF BEAUTIFULSOUP AND SCRAPY

Beautiful Soup	Scrapy
Beautiful Soup is a Python library for pulling data out of HTML and XML files.	Scrapy is a fast high-level web crawling and web scraping framework, used to crawl websites and extract structured data from their pages.
It is simple and easy to learn	It is as compared hard to learn
Project extendibility is low with Beautiful soup	Scrapy allows to develop custom middleware or pipeline to add custom function and it is easy to maintain.
It requires less time as compare to Scrapy Framework for crawling.	It requires more time as compare to Beautifulsoup Framework for crawling

3.2 Implementation and Results

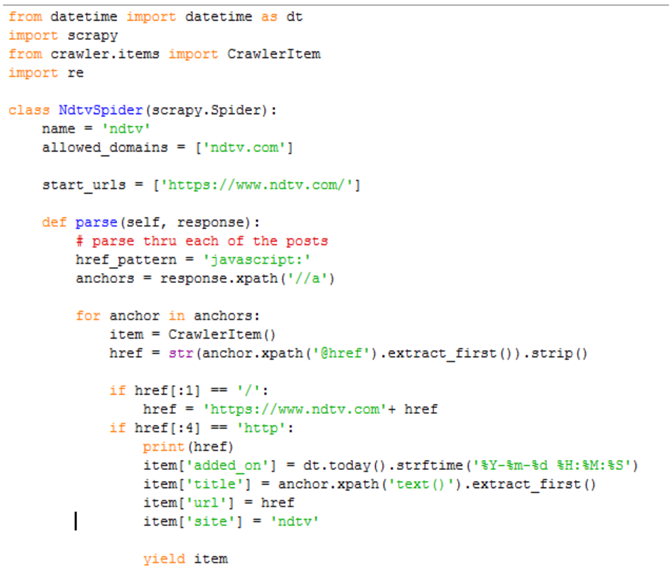
3.2.1 Crawling News web pages using Scrapy with Xpath

Scrapy is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing.

The files included by a Scrapy project and their functions are shown in Table 2. Item objects are simple containers used to collect the scraped data. They are defined in the file of items.py. After items have been crawled by a crawler, they are sent to the Item Pipeline, which processes it through several components that are executed sequentially. Through the Item Pipeline, you can clean HTML data, validate crawled data, check for duplicates and store the item in a database.

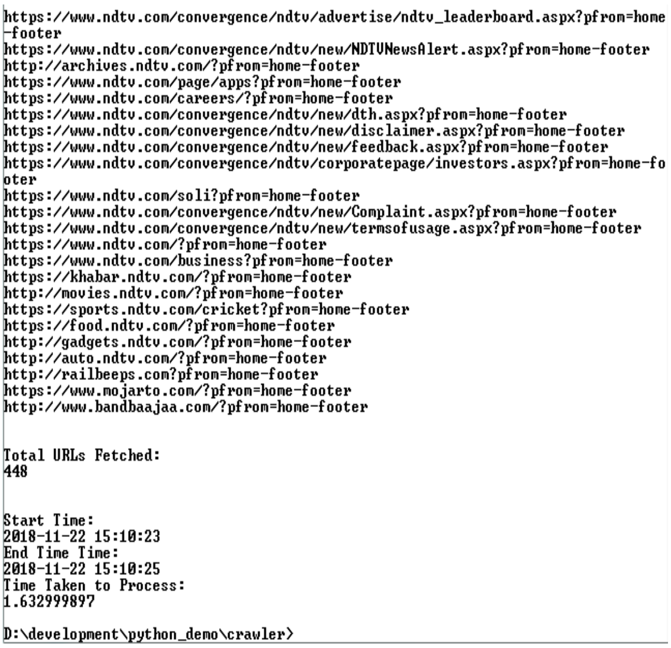
TABLE 2 FILE STRUCTURE OF SCRAPY

scrapy.cfg	Deploy configuration file
items.py	Items definition file of project
pipelines.py	Data transmission file of project
settings.py	Settings file of project
spiders/	The directory where you put your spiders



```python
from datetime import datetime as dt
import scrapy
from crawler.items import CrawlerItem
import re

class NdtvSpider(scrapy.Spider):
    name = 'ndtv'
    allowed_domains = ['ndtv.com']

    start_urls = ['https://www.ndtv.com/']

    def parse(self, response):
        # parse thru each of the posts
        href_pattern = 'javascript:'
        anchors = response.xpath('//a')

        for anchor in anchors:
            item = CrawlerItem()
            href = str(anchor.xpath('@href').extract_first()).strip()

            if href[:1] == '/':
                href = 'https://www.ndtv.com'+ href
            if href[:4] == 'http':
                print(href)
                item['added_on'] = dt.today().strftime('%Y-%m-%d %H:%M:%S')
                item['title'] = anchor.xpath('text()').extract_first()
                item['url'] = href
                item['site'] = 'ndtv'

                yield item
```

Figure 1 Snapshot of Scrapy Spider Implementation



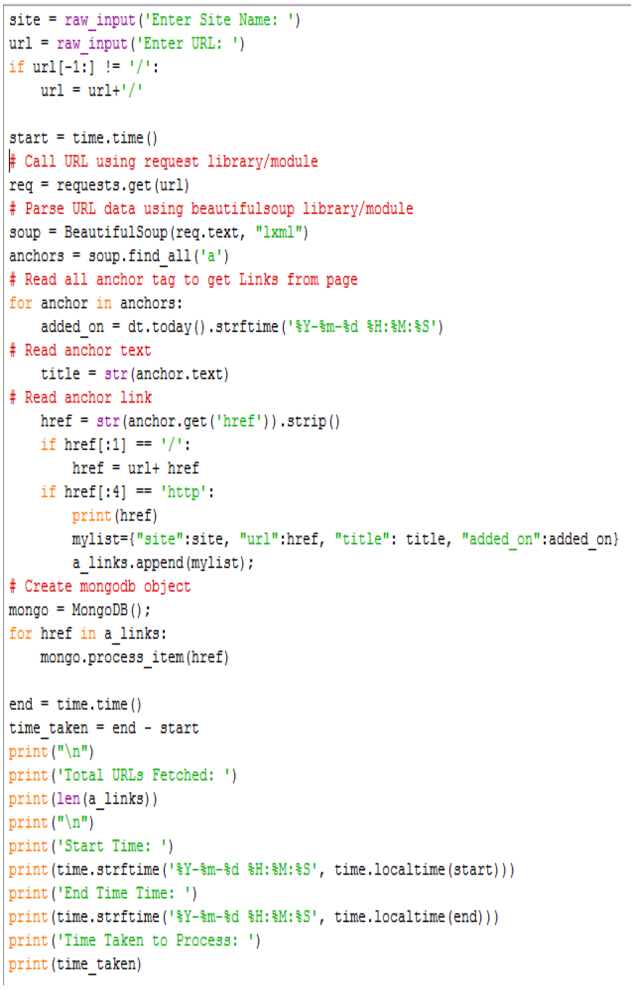
```
https://www.ndtv.com/convergence/ndtv/advertise/ndtv_leaderboard.aspx?pfrom=home
-footer
https://www.ndtv.com/convergence/ndtv/new/NDTVNewsAlert.aspx?pfrom=home-footer
http://archives.ndtv.com/?pfrom=home-footer
https://www.ndtv.com/page/apps?pfrom=home-footer
https://www.ndtv.com/careers/?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/dth.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/disclaimer.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/feedback.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/corporatepage/investors.aspx?pfrom=home-fo
oter
https://www.ndtv.com/soli?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/Complaint.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/termsofusage.aspx?pfrom=home-footer
https://www.ndtv.com/?pfrom=home-footer
https://www.ndtv.com/business?pfrom=home-footer
https://khabar.ndtv.com/?pfrom=home-footer
http://movies.ndtv.com/?pfrom=home-footer
https://sports.ndtv.com/cricket?pfrom=home-footer
https://food.ndtv.com/?pfrom=home-footer
http://gadgets.ndtv.com/?pfrom=home-footer
http://auto.ndtv.com/?pfrom=home-footer
http://railbeeps.com?pfrom=home-footer
https://www.mojarto.com/?pfrom=home-footer
http://www.bandbaajaa.com/?pfrom=home-footer


Total URLs Fetched:
448


Start Time:
2018-11-22 15:10:23
End Time Time:
2018-11-22 15:10:25
Time Taken to Process:
1.632999897

D:\development\python_demo\crawler>
```

Figure 2 Result of Web Scraping Using Scrapy Framework

Figure 4 shows snapshot of scrapy spider implementation of NDTV url and Figure 5 shows the result of scrapy framework It requires 1.63 seconds to crawl NDTV news website.

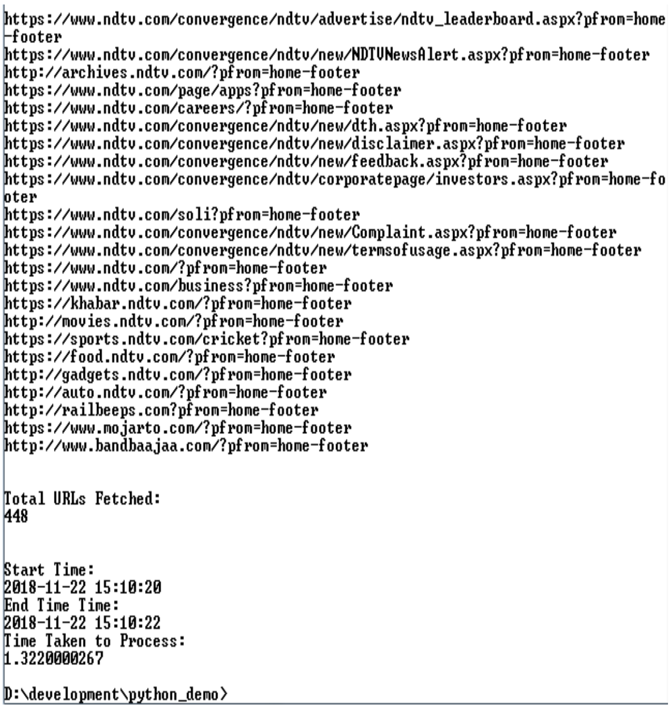
3.1.2 Crawling News Web pages using BeautifulSoup and Request

For crawling news websites I am using Beautiful Soup 4 and Requests. Figure 6 shows snapshot of beautifulsoup implementation and Figure 7 shows the result of web scarping using BeautifulSoup framework.It requires 1.32 seconds to crawl NDTV news website.



```
site = raw_input('Enter Site Name: ')
url = raw_input('Enter URL: ')
if url[-1:] != '/':
    url = url+'/'

start = time.time()
# Call URL using request library/module
req = requests.get(url)
# Parse URL data using beautifulsoup library/module
soup = BeautifulSoup(req.text, "lxml")
anchors = soup.find_all('a')
# Read all anchor tag to get Links from page
for anchor in anchors:
    added_on = dt.today().strftime('%Y-%m-%d %H:%M:%S')
# Read anchor text
    title = str(anchor.text)
# Read anchor link
    href = str(anchor.get('href')).strip()
    if href[:1] == '/':
        href = url+ href
    if href[:4] == 'http':
        print(href)
        mylist={"site":site, "url":href, "title": title, "added_on":added_on}
        a_links.append(mylist);
# Create mongodb object
mongo = MongoDB();
for href in a_links:
    mongo.process_item(href)

end = time.time()
time_taken = end - start
print("\n")
print('Total URLs Fetched: ')
print(len(a_links))
print("\n")
print('Start Time: ')
print(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(start)))
print('End Time Time: ')
print(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(end)))
print('Time Taken to Process: ')
print(time_taken)
```

Figure 3 Snapshot of Beautifulsoup Implementation



```
https://www.ndtv.com/convergence/ndtv/advertise/ndtv_leaderboard.aspx?pfrom=home
-footer
https://www.ndtv.com/convergence/ndtv/new/NDTVNewsAlert.aspx?pfrom=home-footer
http://archives.ndtv.com/?pfrom=home-footer
https://www.ndtv.com/page/apps?pfrom=home-footer
https://www.ndtv.com/careers/?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/dth.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/disclaimer.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/feedback.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/corporatepage/investors.aspx?pfrom=home-fo
oter
https://www.ndtv.com/soli?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/Complaint.aspx?pfrom=home-footer
https://www.ndtv.com/convergence/ndtv/new/termsofusage.aspx?pfrom=home-footer
https://www.ndtv.com/?pfrom=home-footer
https://www.ndtv.com/business?pfrom=home-footer
https://khabar.ndtv.com/?pfrom=home-footer
http://movies.ndtv.com/?pfrom=home-footer
https://sports.ndtv.com/cricket?pfrom=home-footer
https://food.ndtv.com/?pfrom=home-footer
http://gadgets.ndtv.com/?pfrom=home-footer
http://auto.ndtv.com/?pfrom=home-footer
http://railbeeps.com?pfrom=home-footer
https://www.mojarto.com/?pfrom=home-footer
http://www.bandbaajaa.com/?pfrom=home-footer


Total URLs Fetched:
448


Start Time:
2018-11-22 15:10:20
End Time Time:
2018-11-22 15:10:22
Time Taken to Process:
1.3220000267

D:\development\python_demo>
```

Figure 4 Result of Web Scraping Using BeautifulSoup Framework

IV. CONCLUSION

This paper explains how data can be extract from news web pages using scrapy with Xpath and BeautifulSoup with Requests web crawling framework. It also calculate time of Scrapy and Beautifulsoup frameworks required to crawl news web sites and number of urls of news web site. To crawl NDTV news web site with scrapy framework I require 1.63 seconds and with beautifulsoup I require 1.32 seconds that shows scrapy requires more time as compare to beautifulsoup framework.

V. REFERENCES

[1] Chatur Unnati N. Bhakare And Dr.Prashant N. Content Extraction from Deep Web Interfaces [Journal] // International Conference on Electronics, Communication and Aerospace Technology ICECA. - 2017.

[2] Chunmei Zheng Guomei He, Zuojie Peng A Study of Web Information Extraction Technology Based on Beautiful Soup [Journal] // Journal of Computers. - 2015.

[3] Ernesto Cortes Groman Kipp Dunn, Sar Gregorczyk, and Alex Schmidt Opinion Mining & Summarization Final Report [Report]. - Virginia Tech, Blacksburg VA 24061 : Multimedia, Hypertext, and Information Access, May 2 ,2018.

[4] Gupta Shruti Sharma and Parul The Anatomy of Web Crawlers [Journal] // International Conference on Computing, Communication and Automation (ICCCA2015). - 2015.

[5] Mangrulkar AnujaLawankar and Nikhil A Review on Techniques for Optimizing Web Crawler Results [Journal] // World Conference on Futuristic Trends in Research and Innovation for Social Welfare (WCFTR'16). - 2016.

[6] Mengmeng Lu Shuhong Wen, Yan Xiao, Pei Tian ,Fang Wang The Design and Implementation of Configurable News Collection System Based On

Web Crawler [Journal]. - [s.l.] : IEEE, 2017. - Vol. Implementation of Configurable News Collection System Based On Web Crawler.

[7] Pratiksha Ashiwal S.R.Tandan , Priyanka Tripathi , Rohit Miri Web Information Retrieval Using Python and BeautifulSoup [Journal] // Volume 4 Issue VI, June 2016 IJRASET. - 2016.

[8] V.Mahajan Gunjan H. Agre and Nikita Keyword Focused Web Crawler [Journal] // IEEE SPONSORED 2'ND INTERNATIONAL CONFERENCE ON ELECTRONICS AND COMMUNICATION SYSTEMS (ICECS '2015). - 2015.

[9] VinayArora Chandni Saini And Information Retrieval in Web Crawling: A Survey [Journal] // Conference on Advances in Computing, Communications and Informatics (ICACCI). - Sept-2016. - pp. 21-24.

[10] Zejian Shi Minyong Shi and Weiguo Lin The Implementation of Crawling News Page Based On Incremental Web Crawler [Journal]. - [s.l.] : IEEE, 2016.

[11] ZHENG Guojun JIA Wenchao, SHI Jihui, SHI Fan, ZHU Hao, LIU Jiang Design and Application of Intelligent Dynamic Crawler for Web Data Mining [Journal]. - [s.l.] : IEEE, 2017.

Cite this article as :

Harshala Bhoir, K. Jayamalini, "Web Crawling on News Web Page using Different Frameworks", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 4, pp. 513-519, July-August 2021. Available at doi : https://doi.org/10.32628/CSEIT2174120
Journal URL : https://ijsrcseit.com/CSEIT2174120