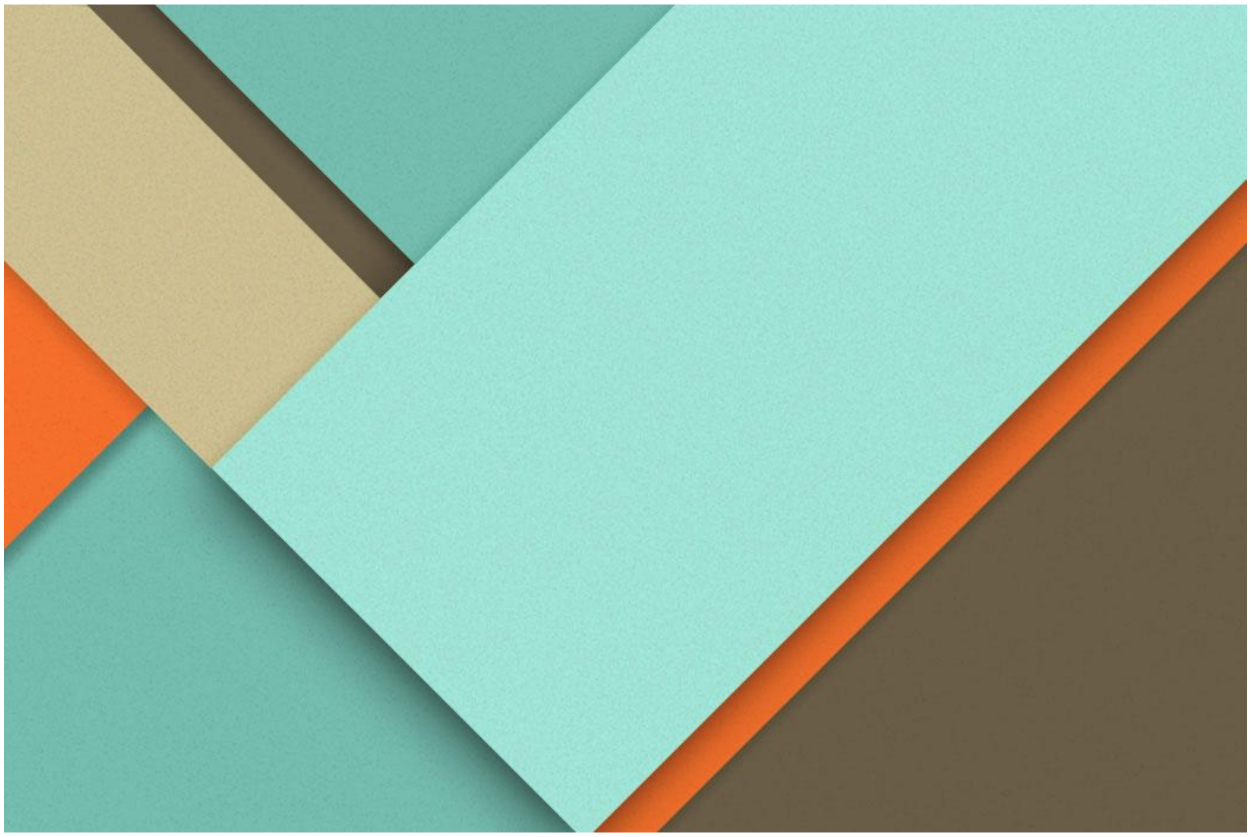


Github Topic Scraping using Beautiful soup, Scrapy & Selenium

Principal Investigators: Anna Lewczuk & Przemyslaw Kurek

—

Karan Vijay Kashyap
428882

Amrute Shrivallabh
431021

Project Overview

We have scraped https://github.com/topics/ Where we got a list of topics and for every topic, we will scrape Topic Title, Topic Description, and Topic Url using Beautifulsoup, Scrapy, and Selenium. Finally, we will create CSV file in below format:

title,description,url

3D,3D modeling is the process of virtually developing the surface and structure of a 3D object.,https://github.com/topics/3d

Ajax,Ajax is a technique for creating interactive web applications.,https://github.com/topics/ajax

Algorithm,Algorithms are self-contained sequences that carry out a variety of tasks.,https://github.com/topics/algorithm

Amp,Amp is a non-blocking concurrency framework for PHP.,https://github.com/topics/amphp

Website Description: GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration, and wikis for every project

Team Member

1. Karan Vijay Kashyap - 428882
2. Amrute Vallabh - 431021

Scraper Mechanics

In scraping Mechanics, we have used the request library to download the URL and parse the website. As mentioned above output, we have to extract Topic_title, Topic_Description, and Topic_url for each Topic present in Github. We have done it using the below three scraper by extracting the required information step by step as explained below using:

Beautiful soup

Libraries used:

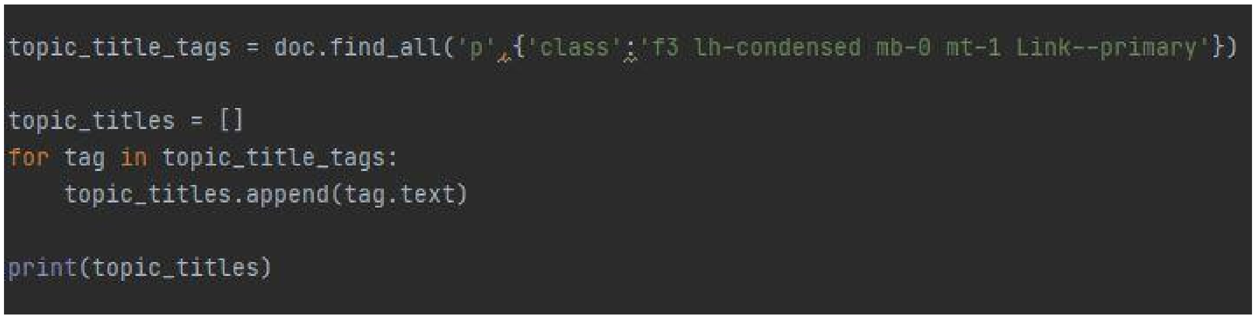
$pip install requests

$pip install beautifulsoup4 or File >> settings >> python interpreter >> + >> beautifulsoup4

$pip install pandas or File >> settings >> python interpreter >> + >> pandas

In Beautiful soup, we have first find the topic_title tags using find.all function and website url was parsed in doc file - doc.find.all, then we have created a list with name topic_titles where we have appended the text from find.all function to the topic_title list using for loop function for iteration. The result was printed to the console to check the output and later to the CSV using pandas data frame.

Using the above same approach we found title_description and title_url.



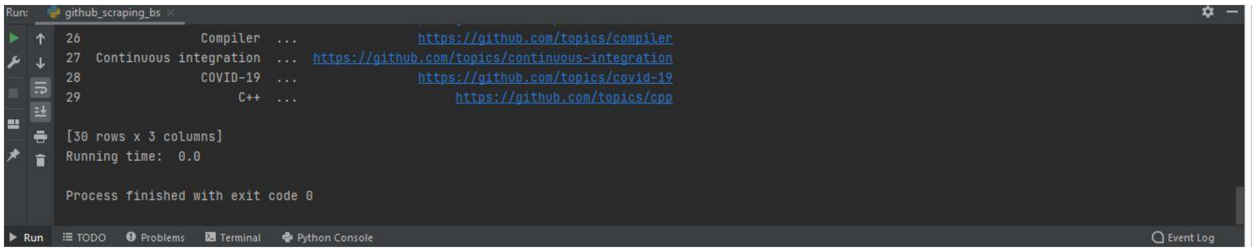
```
topic_title_tags = doc.find_all('p',{'class':'f3 lh-condensed mb-0 mt-1 Link--primary'})

topic_titles = []
for tag in topic_title_tags:
    topic_titles.append(tag.text)

print(topic_titles)
```

The beautiful soup was very easy to understand, quick in response and It is better for scraping when the content to scrape is small.

Result Terminal: Running Time - 0.0

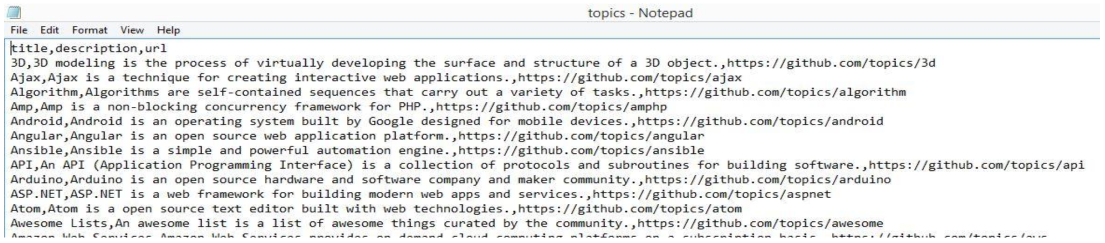


```
26                Compiler  ...                 https://github.com/topics/compiler
27  Continuous integration  ...  https://github.com/topics/continuous-integration
28                COVID-19  ...                 https://github.com/topics/covid-19
29                     C++  ...                      https://github.com/topics/cpp

[30 rows x 3 columns]
Running time:  0.0

Process finished with exit code 0
```

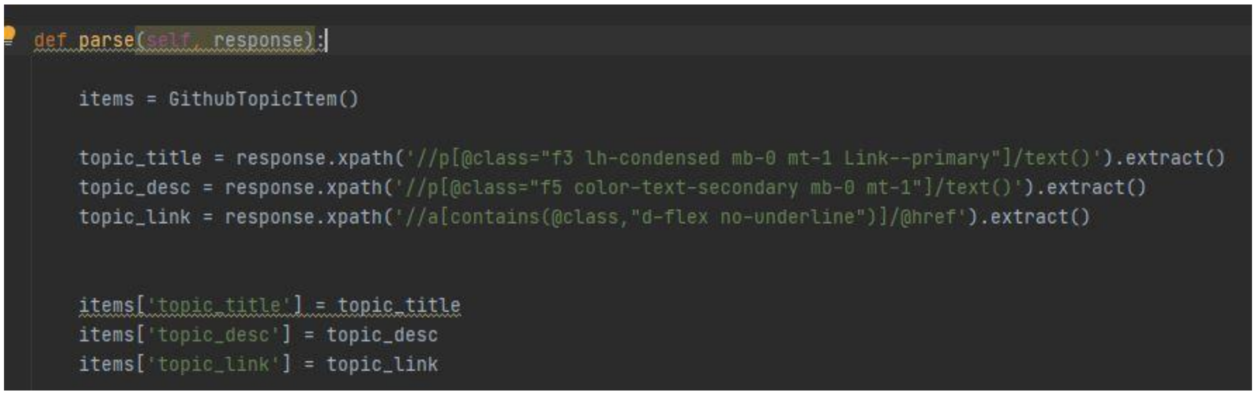
Result: The file was saved in CSV file as output and It contains title, description, and URL.



```
title,description,url
3D,3D modeling is the process of virtually developing the surface and structure of a 3D object.,https://github.com/topics/3d
Ajax,Ajax is a technique for creating interactive web applications.,https://github.com/topics/ajax
Algorithm,Algorithms are self-contained sequences that carry out a variety of tasks.,https://github.com/topics/algorithm
Amp,Amp is a non-blocking concurrency framework for PHP.,https://github.com/topics/amphp
Android,Android is an operating system built by Google designed for mobile devices.,https://github.com/topics/android
Angular,Angular is an open source web application platform.,https://github.com/topics/angular
Ansible,Ansible is a simple and powerful automation engine.,https://github.com/topics/ansible
API,An API (Application Programming Interface) is a collection of protocols and subroutines for building software.,https://github.com/topics/api
Arduino,Arduino is an open source hardware and software company and maker community.,https://github.com/topics/arduino
ASP.NET,ASP.NET is a web framework for building modern web apps and services.,https://github.com/topics/aspnet
Atom,Atom is a open source text editor built with web technologies.,https://github.com/topics/atom
Awesome Lists,An awesome list is a list of awesome things curated by the community.,https://github.com/topics/awesome
```

Scrapy

In Scrapy we have used a parse function which contains items and stores the data collected from Xpath in topic_title, topic_desc and topic_link. Scrapy has built-in support for extracting data from HTML sources, and it uses XPATH expressions, plus it's a portable library and can run it anywhere (in any OS) and actually Scrapy is faster than other tools. We can also import external libraries.



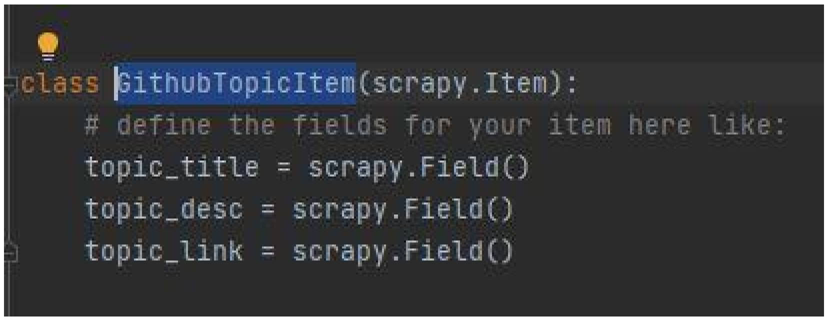
```
def parse(self, response):

    items = GithubTopicItem()

    topic_title = response.xpath('//p[@class="f3 lh-condensed mb-0 mt-1 Link--primary"]/text()').extract()
    topic_desc = response.xpath('//p[@class="f5 color-text-secondary mb-0 mt-1"]/text()').extract()
    topic_link = response.xpath('//a[contains(@class,"d-flex no-underline")]/@href').extract()


    items['topic_title'] = topic_title
    items['topic_desc'] = topic_desc
    items['topic_link'] = topic_link
```

The below code is saved in items.py, where each topic_title is equal to scrapy.field()

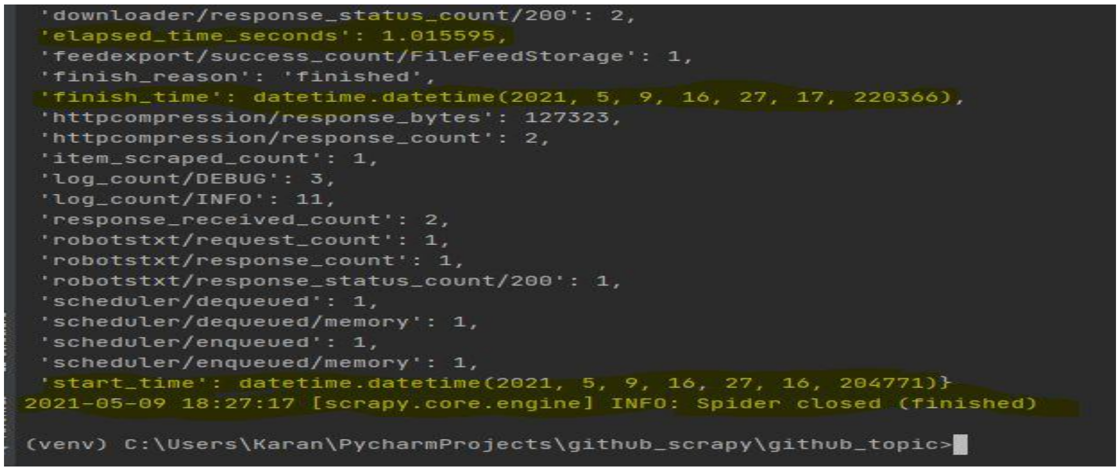


```
class GithubTopicItem(scrapy.Item):
    # define the fields for your item here like:
    topic_title = scrapy.Field()
    topic_desc = scrapy.Field()
    topic_link = scrapy.Field()
```

Result: The code spider was running successfully using code

>> scrapy crawl topic

To save in CSV file >> `scrapy crawl topic -o file.csv -t csv`

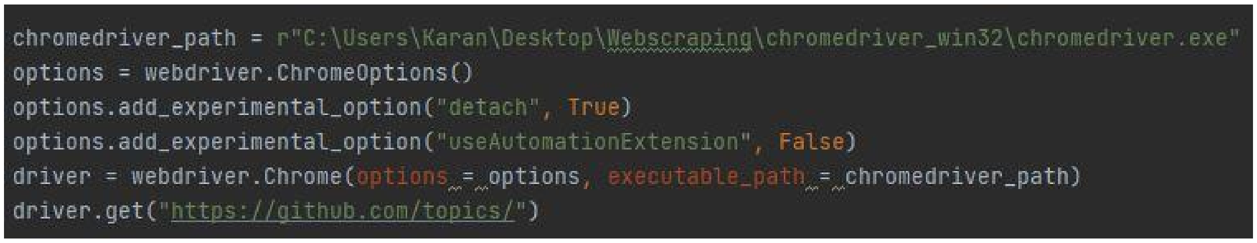


```
 'downloader/response_status_count/200': 2,
 'elapsed_time_seconds': 1.015595,
 'feedexport/success_count/FileFeedStorage': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2021, 5, 9, 16, 27, 17, 220366),
 'httpcompression/response_bytes': 127323,
 'httpcompression/response_count': 2,
 'item_scraped_count': 1,
 'log_count/DEBUG': 3,
 'log_count/INFO': 11,
 'response_received_count': 2,
 'robotstxt/request_count': 1,
 'robotstxt/response_count': 1,
 'robotstxt/response_status_count/200': 1,
 'scheduler/dequeued': 1,
 'scheduler/dequeued/memory': 1,
 'scheduler/enqueued': 1,
 'scheduler/enqueued/memory': 1,
 'start_time': datetime.datetime(2021, 5, 9, 16, 27, 16, 204771)}
2021-05-09 18:27:17 [scrapy.core.engine] INFO: Spider closed (finished)

(venv) C:\Users\Karan\PycharmProjects\github_scrapy\github_topic>
```

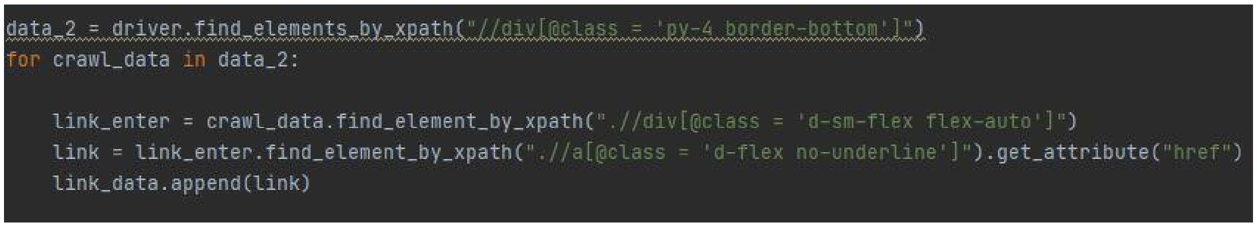
Selenium

It is also called browser automation, The selenium is best tool, we have combined it with beautifulsoup and applied in the project. we have used chrome as selenium web driver and the path to the web driver is of the chrome_driver.exe stored in the folder. Which acts in the same way as user do it manually by using scrapy or beautiful soup of extracting topic_title,url and description.



```
chromedriver_path = r"C:\Users\Karan\Desktop\Webscraping\chromedriver_win32\chromedriver.exe"
options = webdriver.ChromeOptions()
options.add_experimental_option("detach", True)
options.add_experimental_option("useAutomationExtension", False)
driver = webdriver.Chrome(options = options, executable_path = chromedriver_path)
driver.get("https://github.com/topics/")
```

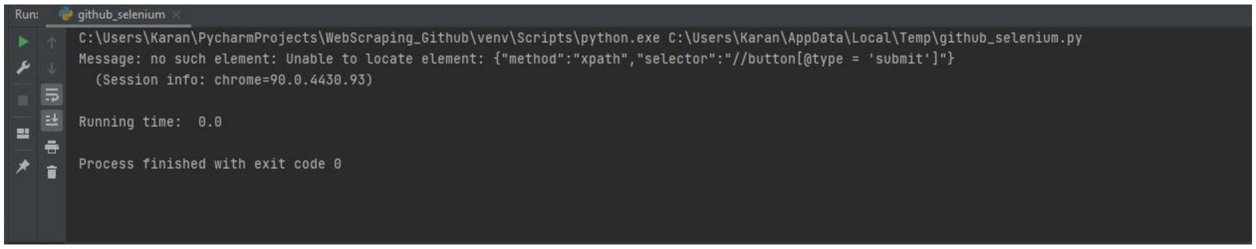
This is used to extract data and append the data in the CSV file



```
data_2 = driver.find_elements_by_xpath("//div[@class = 'py-4 border-bottom']")
for crawl_data in data_2:

    link_enter = crawl_data.find_element_by_xpath(".//div[@class = 'd-sm-flex flex-auto']")
    link = link_enter.find_element_by_xpath(".//a[@class = 'd-flex no-underline']").get_attribute("href")
    link_data.append(link)
```

Result: After successful completion, Running Time - 0.0



```
C:\Users\Karan\PycharmProjects\WebScraping_Github\venv\Scripts\python.exe C:\Users\Karan\AppData\Local\Temp\github_selenium.py
Message: no such element: Unable to locate element: {"method":"xpath","selector":"//button[@type = 'submit']"}
  (Session info: chrome=90.0.4430.93)

Running time:  0.0

Process finished with exit code 0
```

Performance Comparision

As you noticed in the above result screenshot, running time is 0.0 which is the same for scrapy , selenium and beautifulsoup scraper.

```
start = time.time()
```

```
print("Running time: ", time.time() - start)
```

We have imported the time library and used the above python code to check the running time.

Technical Description

The Output for all three scrapers was successfully saved in the CSV file.

Beautiful soup and Selenium - By clicking ‘Run’

Scrapy -using command terminal >> `scrapy crawl topic -o file.csv -t csv`

Exploratory Data Analysis

We have used Topic description text for Clustering and Dimension reduction using Topic Modeling, Where we extract topics from Unstructured data!

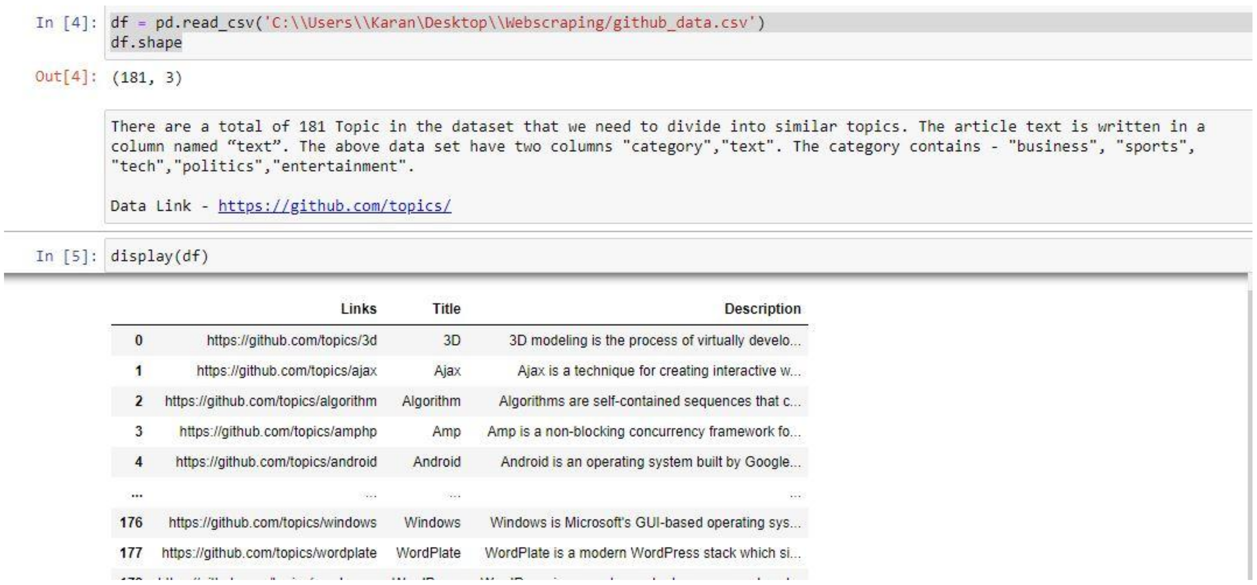
What is Topic Modeling, and how does it work?

Automatically extracting the topics people are discussing from vast volumes of text is one of the key applications of natural language processing. Large text examples include social media feeds, consumer reviews of hotels, movies, and other companies, user input, news reports, and e-mails from customers with complaints.

Manually reading through such vast volumes and assembling the topics is extremely difficult.

As a result, we needed an automatic algorithm like LDA to read the text and output the discussed topics.

Topic modeling Use Cases: Document categorization Document summarization Dimension reduction- Doc words matrix is reduced to doc topic matrix



```
In [4]:  df = pd.read_csv('C:\\Users\\Karan\Desktop\\Webscraping/github_data.csv')
         df.shape

Out[4]:  (181, 3)
```

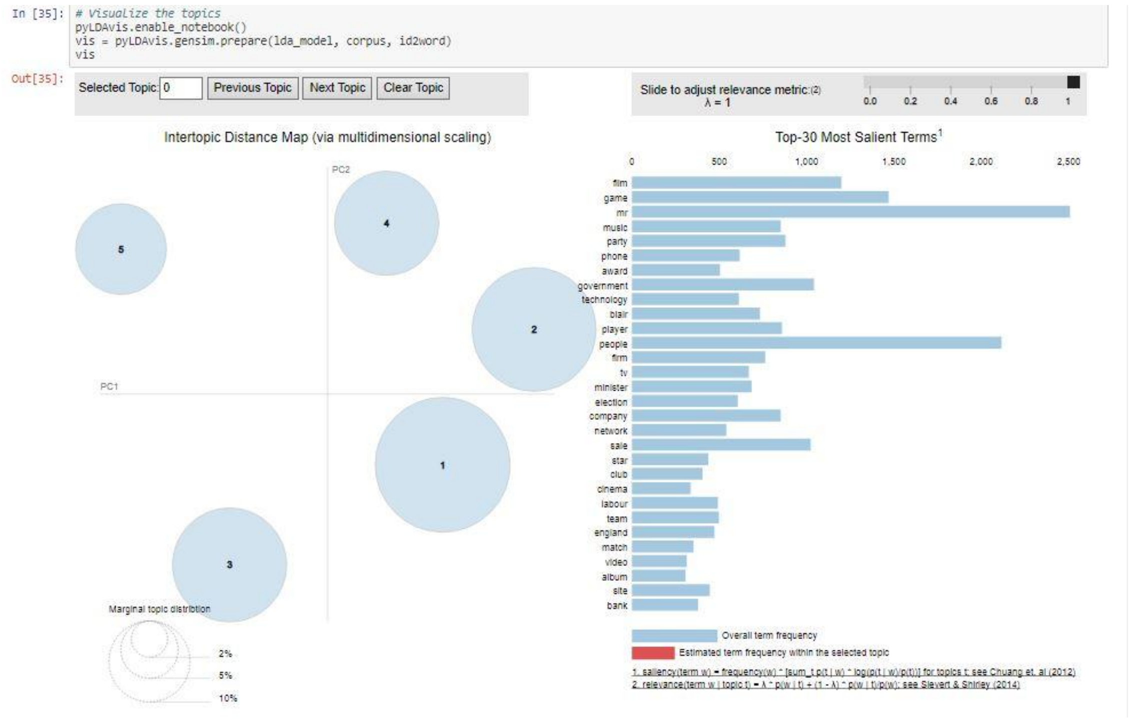
There are a total of 181 Topic in the dataset that we need to divide into similar topics. The article text is written in a column named “text”. The above data set have two columns "category","text". The category contains - "business", "sports", "tech","politics","entertainment".

Data Link - https://github.com/topics/

```
In [5]:  display(df)
```

	Links	Title	Description
0	https://github.com/topics/3d	3D	3D modeling is the process of virtually develo...
1	https://github.com/topics/ajax	Ajax	Ajax is a technique for creating interactive w...
2	https://github.com/topics/algorithm	Algorithm	Algorithms are self-contained sequences that c...
3	https://github.com/topics/amphp	Amp	Amp is a non-blocking concurrency framework fo...
4	https://github.com/topics/android	Android	Android is an operating system built by Google...
...	...	...	...
176	https://github.com/topics/windows	Windows	Windows is Microsoft's GUI-based operating sys...
177	https://github.com/topics/wordplate	WordPlate	WordPlate is a modern WordPress stack which si...

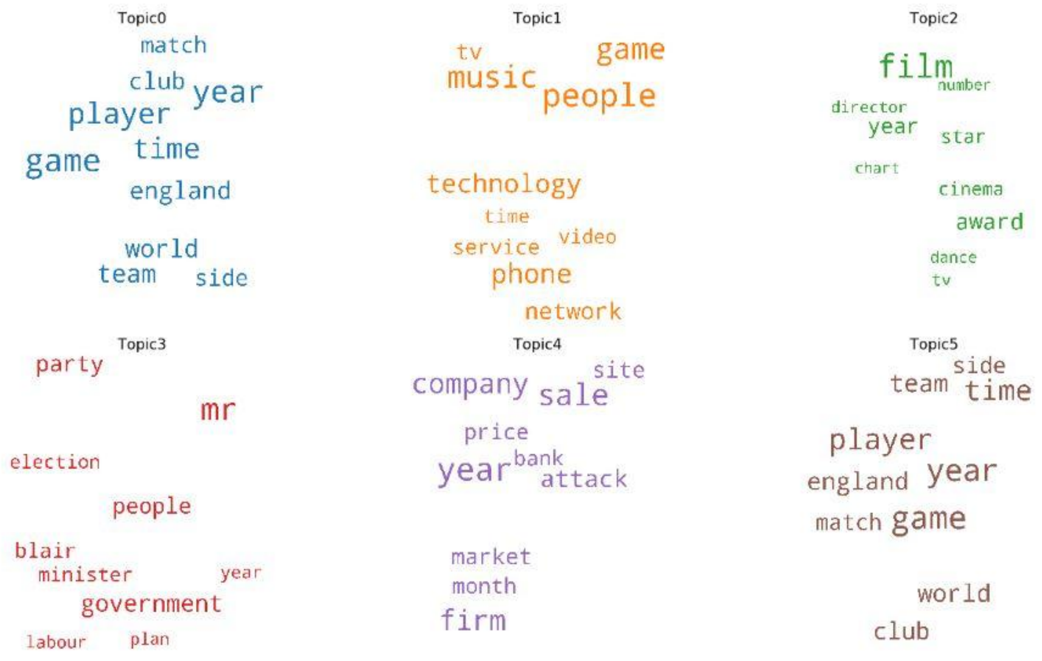
Output Result:



```
In [35]:  # Visualize the topics
          pyLDAvis.enable_notebook()
          vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
          vis

Out[35]:
```

1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



We started with understanding what topic modeling can do. We built a basic topic model using Gensim's LDA and visualize the topics using pyLDAvis, As you can see clearly that we have extracted topics from a large volume of data.

We have done by using Dimension reduction technique using Topic Modeling-Latent Dirichlet Allocation (LDA) using Gensim have extracted topic from a large volume of data.

This is a highly valuable algorithm and technique as it is really hard to manually read through such large volumes and compile the topics for which the above algorithm is very efficient

Knowing what people are talking about and understanding their problems and opinions is highly valuable to businesses, administrators, political campaigns. And it's really hard to manually read through such large volumes and compile the topics.

Thus is required an automated algorithm that can read through the text documents and automatically output the topics discussed.

Work Distribution:

Karan Vijay Kashyap	Amrute Shrivallabh
Project Idea & Execution	Project Proposal & Support in execution
Scrapy & Selenium	Beautiful Soup & Selenium
Exploratory Data Analysis	Error Debugging in code
Project Description file	Project Description file