<u>News Articles Sentiment Analysis</u>

Author(s):
Darshan Solanki das968@nyu.edu
Hsia-ming Hsu hmh371@nyu.edu
Olivia zhao - olivia.zhao@nyu.edu
Renyue Zhang - rz1535@nyu.edu
Weihao Bi - wb832@nyu.edu
Raman Kannan - rk1750@nyu.edu

This project was completed as a partial requirement for CS6513 Big Data Tools and Techniques at Tandon School of Engineering, NYU
Generous support from IBM Power Systems Academic Initiative is acknowledged

1. Overview

This project helps us find the trend in the US over the years with the type of news published by legit sources like CNN and The Guardian to make inferences about the things happening, are they more pessimists or positive progress in the country. We could also make inferences about the number of negative and positive news over the years has not been a consistent ratio and it's variably changing. We have used several algorithms to not influence the bias in the articles and give sentiments to the articles in the most accurate way. Graphical Representations has been incorporated for end results in a visual way.

Keywords : beautiful soup, nltk, newspaper, articles, naive bayes, multinomialNB, bernoulliNB, logisticregression,linearSVC.

2. Workflow and Working Environment

2.1 Workflow

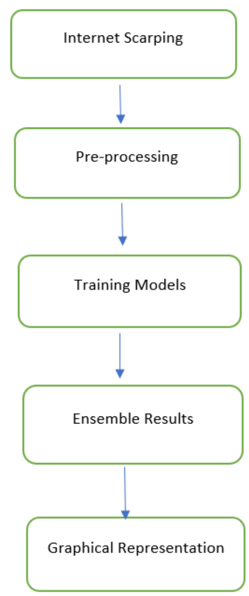


Fig 2.1 WorkFlow

Internet Scraping - In this stage data is scraped from CNN website and stored in different formats like json, csv etc.

Pre-processing - Scarped Data is then pre-processed with more features added for graphical purpose.

Training Models - Used Different Algorithms to run sentiment analysis on to get the better estimation of the sentiment rather than relying on one or two algorithm.

Used Supervised Algorithms :

Ensemble Result : Training algorithms results are merged using majority voting technique to get the most probable sentiment outcome for an article(s)

Graphical Representation : Visualizing the trend other the years about negative and positive published articles and making inferences out of it.

2.2 Environment Requirement

2.2.1 Operation System

We mainly use Windows and Mac OS for this project.

2.2.1 Programming System

Python: Spyder, Jupyter Notebook and PyCharm
JavaScript: WebStorm

2.2.3 Framework and Libraries

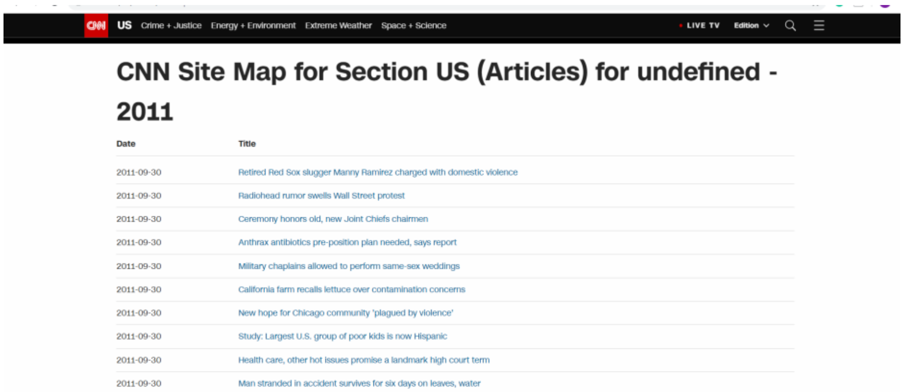
Web Crawler: BeautifulSoup, Requests,
Content Parsing: Newspaper3k,newspaper,articles,fulltext
Python libraries : nltk, scikit-learn
Proxies IPs: daxiangdaili
Natural Language Processing: Naive Bayes, MultinomialNB, BernoulliNB, LogisticRegression and LinearSVC
Front-End: ECharts, D3.js and Bootstrap

3. Data Gathering

3.1 Learning Data Structure

This is one of the initial and most crucial parts of our project. We have worked with real data, scraped from CNN websites pertaining to only US regions news. Scraping data needs to have a solid understanding of how data has embedded inside the page source.

After analyzing the structure of the data displayed, here is the screenprint from one of the website link of CNN to show the format pattern which remain consistent for crawling all of CNN data.



CNN Site Map for Section US (Articles) for undefined - 2011

Date	Title
2011-09-30	Retired Red Sox slugger Manny Ramirez charged with domestic violence
2011-09-30	Radiohead rumor swells Wall Street protest
2011-09-30	Ceremony honors old, new Joint Chiefs chairmen
2011-09-30	Anthrax antibiotics pre-position plan needed, says report
2011-09-30	Military chaplains allowed to perform same-sex weddings
2011-09-30	California farm recalls lettuce over contamination concerns
2011-09-30	New hope for Chicago community 'plagued by violence'
2011-09-30	Study: Largest U.S. group of poor kids is now Hispanic
2011-09-30	Health care, other hot issues promise a landmark high court term
2011-09-30	Man stranded in accident survives for six days on leaves, water

Fig 3.1

source : *https://www.cnn.com/us/article/sitemap-2011-09.html*

```
class="sitemap-entry"><ul><li><span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/30/us/sport-florida-ramirez-charged/index.html"
Manny Ramirez charged with domestic violence</a></span></li><li><span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/30/us/radiohe
street/index.html">Radiohead rumor swells Wall Street protest</a></span></li><li><span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011
ceremony/index.html">Ceremony honors old, new Joint Chiefs chairmen</a></span></li><li><span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.co
antibiotics/index.html">Anthrax antibiotics pre-position plan needed, says report</a></span></li><li><span class="date">2011-09-30</span><span class="sitemap-link"><a
href="https://www.cnn.com/2011/09/30/us/same-sex-marriage-military/index.html">Military chaplains allowed to perform same-sex weddings</a></span></li><li><span class="date">2011
class="sitemap-link"><a href="https://www.cnn.com/2011/09/30/us/lettuce-recall/index.html">California farm recalls lettuce over contamination concerns</a></span></li><li><span c
<span class="sitemap-link"><a href="https://www.cnn.com/2011/09/29/us/cnnheroes-latiker-top10/index.html">New hope for Chicago community &apos;plagued by violence&apos;</a></spa
class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/29/us/latino-kids-poverty/index.html">Study: Largest U.S. group of poor kids is now
<span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/30/us/scotus-preview-health-care/index.html">Health care, other hot issues pr
term</a></span></li><li><span class="date">2011-09-30</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/30/us/california-mountain-crash/index.html">Man stran
six days on leaves, water</a></span></li><li><span class="date">2011-09-29</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/29/us/census/index.html">White U
drops in overall percentage</a></span></li><li><span class="date">2011-09-29</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/28/us/immigration-sting/index.
convicted criminal immigrants arrested, ICE says</a></span></li><li><span class="date">2011-09-29</span><span class="sitemap-link"><a href="https://www.cnn.com/2011/09/28/us/new
fdny/index.html">Medal of Honor recipient declines city&apos;s offer to file late application for FDNY</a></span></li><li><span class="date">2011-09-29</span><span class="sitema
href="https://www.cnn.com/2011/09/28/us/gitmo-trial/index.html">Guantanamo prepares for next military trial of terrorism suspect</a></span></li><li><span class="date">2011-09-29
```

Fig 3.2 Page Source of the article

source *view-source:https://www.cnn.com/us/article/sitemap-2011-09.html*

From fig 3.2, we can see, all those articles links have been defined inside the class tag of sitemap-link.

So, we made use of python Beautiful soup library that works well with html parser where we can search,filter, extract data using tag names. Once we get the article links, we write it in each different sitemaps file having the conventional as 'SiteMap-{Year}-{Month}.json'. This will be convenient to read and understand to outside users - this file indicates all the links scraped from particular year and particular month of that year.

Still we have got the links of the articles and not the actual article data. This task is done using newspaper, articles, fulltext libraries of python that has leveraged in our code.
We pass the URL to the article library and it makes get request to the website and bring us back the data.

3.1.1 Data

These are the attributes after we scrap the articles.

Outlet : From which source we have gathered the data in our case CNN

Date : The published date of the article

Title : Title of that article:

Url: Actual URLof that article if someone wants to read it from the website

File_name : The article data is not directly written in this field, rather to make things organized, we have mapped each article to a file and writing it's content in that file and giving file name as a field in the data table.

Scarped : 70,000 articles over the past 10 years only specific to US news.

Another Data Source we worked on is **The Guardian** – UK based news media.

- The data displayed on the front-end has uniformity at the back-end. This was found after studying page source content and embedded url. All comprises under the class tag **u-faux-block-link__overlay js-headline-text.**
- US based news can easily be found with the same convention used even 12 years before. So, it made it easy to scrap historical data. Since it's UK origin news, not many articles were seen on those web pages.

Overall we managed to scrap ~47k articles in less than 90 mins using the third approach as explained in next section.

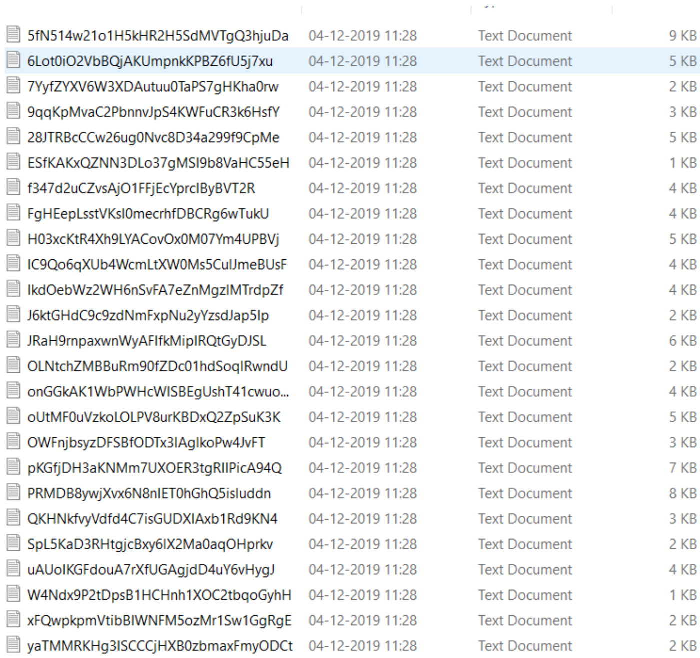
3.1.2 Approaches Implemented:

Sequential Hit : In this case, we hit the url one after the other and not concurrent which seems to be inefficient as many articles needs to be scrapped in optimal amount of time. The time taken will surpass the time threshold we want to achieve the results in. If one of the article faced issue and was not able to processed it will hung up the lined url's behind him, so the failure will stop the code working.

Parallel Concurrent Method: Multiple threads are created to hit CNN server with processing pool of threads ranging from 10-50 which is explained in the next stage as difficulty faced of an IP block and to achieve faster extraction of the data.

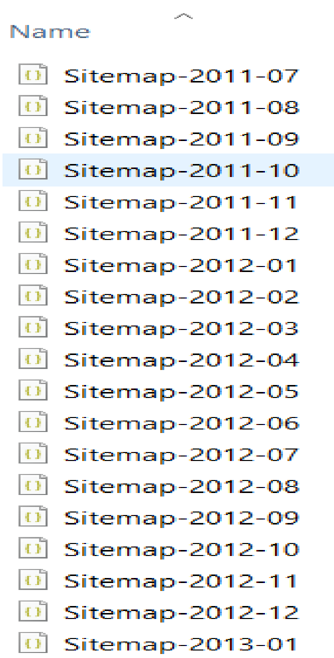
Distributed Computing: Multiple systems (3) were used to scarp assigned year-wise data. All data uploaded to Google drive was then collected to merged together locally on one machine. This approach did not used any multiple IP's that we faced earlier and took approx. 1hr 20 mins to scrap ~47k articles. Cost could go high little bit as many systems are involved but time has reduced significantly.

Please find below the screen print for sitemaps scraped and Data Format table after scraping the articles.



5fN514w21o1H5kHR2H5SdMVTgQ3hjuDa	04-12-2019 11:28	Text Document	9 KB
6Lot0iO2VbBQjAKUmpnkKPBZ6fU5j7xu	04-12-2019 11:28	Text Document	5 KB
7YyfZYXV6W3XDAutuu0TaPS7gHKha0rw	04-12-2019 11:28	Text Document	2 KB
9qqKpMvaC2PbnnvJpS4KWFuCR3k6HsfY	04-12-2019 11:28	Text Document	3 KB
28JTRBcCCw26ug0Nvc8D34a299f9CpMe	04-12-2019 11:28	Text Document	5 KB
ESfKAKxQZNN3DLo37gMSI9b8VaHC55eH	04-12-2019 11:28	Text Document	1 KB
f347d2uCZvsAjO1FFjEcYprcIByBVT2R	04-12-2019 11:28	Text Document	4 KB
FgHEepLsstVKsI0mecrhfDBCRg6wTukU	04-12-2019 11:28	Text Document	4 KB
H03xcKtR4Xh9LYACovOx0M07Ym4UPBVj	04-12-2019 11:28	Text Document	5 KB
IC9Qo6qXUb4WcmLtXW0Ms5CuIJmeBUsF	04-12-2019 11:28	Text Document	4 KB
IkdOebWz2WH6nSvFA7eZnMgzIMTrdpZf	04-12-2019 11:28	Text Document	4 KB
J6ktGHdC9c9zdNmFxpNu2yYzsdJap5Ip	04-12-2019 11:28	Text Document	2 KB
JRaH9rnpaxwnWyAFIfkMipIRQtGyDJSL	04-12-2019 11:28	Text Document	6 KB
OLNtchZMBBuRm90fZDc01hdSoqIRwndU	04-12-2019 11:28	Text Document	2 KB
onGGkAK1WbPWHcWISBEgUshT41cwuo...	04-12-2019 11:28	Text Document	4 KB
oUtMF0uVzkoLOLPV8urKBDxQ2ZpSuK3K	04-12-2019 11:28	Text Document	5 KB
OWFnjbsyzDFSBfODTx3IAgIkoPw4JvFT	04-12-2019 11:28	Text Document	3 KB
pKGfjDH3aKNMm7UXOER3tgRIIPicA94Q	04-12-2019 11:28	Text Document	7 KB
PRMDB8ywjXvx6N8nIET0hGhQ5isluddn	04-12-2019 11:28	Text Document	8 KB
QKHNkfvyVdfd4C7isGUDXIAxb1Rd9KN4	04-12-2019 11:28	Text Document	3 KB
SpL5KaD3RHtgjcBxy6IX2Ma0aqOHprkv	04-12-2019 11:28	Text Document	2 KB
uAUoIKGFdouA7rXfUGAgjdD4uY6vHygJ	04-12-2019 11:28	Text Document	4 KB
W4Ndx9P2tDpsB1HCHnh1XOC2tbqoGyhH	04-12-2019 11:28	Text Document	1 KB
xFQwpkpmVtibBIWNFM5ozMr1Sw1GgRgE	04-12-2019 11:28	Text Document	2 KB
yaTMMRKHg3ISCCCjHXB0zbmaxFmyODCt	04-12-2019 11:28	Text Document	2 KB

Fig 3.3 Actual data content files CNN



Name
Sitemap-2011-07
Sitemap-2011-08
Sitemap-2011-09
Sitemap-2011-10
Sitemap-2011-11
Sitemap-2011-12
Sitemap-2012-01
Sitemap-2012-02
Sitemap-2012-03
Sitemap-2012-04
Sitemap-2012-05
Sitemap-2012-06
Sitemap-2012-07
Sitemap-2012-08
Sitemap-2012-09
Sitemap-2012-10
Sitemap-2012-11
Sitemap-2012-12
Sitemap-2013-01

Fig 3.4 SiteMaps File CNN

```
df = pd.read_csv('Complete_Articles_Data.csv',names = ['outlet','date','title','url','text_file'],sep='|')
df.tail(10)
```

	outlet	date	title	url	text_file
69965	CNN	2019-4-30	USA Gymnastics director of sports medicine is ...	https://www.cnn.com/2019/04/30/us/usa-gymnasti...	AspBQGmdgSd7rWBwE5mGA1QjE8qg6bqN.txt
69966	CNN	2019-5-28	Ohio tornado survivor: It's heartbreaking	https://www.cnn.com/videos/us/2019/05/28/ohio-...	pccnTjjQw4isvX1eQgnLn8GVJPRPBYUN.txt
69967	CNN	2019-4-29	The Illinois plant shooter threatened to kill ...	https://www.cnn.com/2019/04/29/us/aurora-illin...	RrUrBJ7VgJCXs6kbb1OkJsHTSOVqAKSe.txt
69968	CNN	2019-5-1	New York is the first major city to allow free...	https://www.cnn.com/2019/05/01/us/free-calls-f...	pJYZgTfHITU148t36CqiecYnVn6UkRDZ.txt
69969	CNN	2019-5-1	A police officer responded to a noise complain...	https://www.cnn.com/2019/05/01/us/police-offic...	rvoZ6QVtXtcVpZuqWKdkh5xPgIYx0dXg.txt
69970	CNN	2019-5-1	Maine becomes the first state to ban Styrofoam	https://www.cnn.com/2019/05/01/us/maine-ban-st...	9q06ZWJXGZhZvWMDdiC6QwTat6y2lMrO.txt
69971	CNN	2015-8-25	John Kasich Fast Facts	https://www.cnn.com/2015/08/25/us/john-kasich-...	OLYILIXggHawfNoZU7wDHb7e8Dni7KpT.txt
69972	CNN	2019-5-1	Chicago sees slight drop in violent crime in A...	https://www.cnn.com/2019/05/01/us/chicago-crim...	ZI62b6lQyKrl2UsGTenOgWd7k2M8Z30W.txt
69973	CNN	2019-5-1	Students stage walkout at Illinois high school...	https://www.cnn.com/2019/05/01/us/blackface-il...	wLWhjf4tAliyQLJlbC3F5opWW7750oCP.txt
69974	CNN	2019-5-1	2 Swarthmore fraternities will disband after d...	https://www.cnn.com/2019/05/01/us/swarthmore-f...	crbFZ8w60LmDKAhnaTydaJTubHpRWCCK.txt

Fig 3.5 show concurrent scarping data table

```
#Reading the modified csv

df = pd.read_csv('Complete_Articles_Data_Embedded.csv',sep='|',index_col = 0,encoding='utf-8')
df.head(10)
```

	outlet	title	url	text_file
date				
2011-01-20	CNN	2011: Who really killed Daniel Pearl?	https://www.cnn.com/videos/us/2011/01/20/todd....	X1P2NkD3ATSkaOM2I5MnAitcreebubYo.txt
2011-01-24	CNN	1999: First twins in space	https://www.cnn.com/videos/us/2011/01/24/1999....	ac0ZqvYcppz29oS6G6cFubp4Rh53bM4a.txt
2011-01-27	CNN	Challenger disaster remembered	https://www.cnn.com/videos/us/2011/01/27/natpk...	0ajiuk9RaGDosWWrSn48ElHNqlfUFfQZ.txt
2011-01-31	CNN	2003: Space shuttle Columbia disaster	https://www.cnn.com/videos/us/2011/01/31/natpk...	x0skEvZt5dg92jLAx4QiQx7b6KY9K8TL.txt
2011-02-20	CNN	Tips for dating online	https://www.cnn.com/videos/us/2011/02/20/nr-ka...	CeJMtXcCmMouDwGocJruuUCSoxluP0B0.txt
2011-02-23	CNN	30 years, 135 launches in 135 seconds	https://www.cnn.com/videos/us/2011/02/23/nat.1...	a29OpPZrvAPGAzHNZe4dD1dkLBuPnlhT.txt
2011-02-24	CNN	2011: Shuttle Discovery's final launch	https://www.cnn.com/videos/bestoftv/2011/02/24...	JamLURFru7yY4x7MEjakAuy2j4ZXeB6V.txt
2011-02-28	CNN	1986: Reagan's 'Just say no' campaign	https://www.cnn.com/videos/us/2011/02/28/vault...	w0B3CNm0Qh85xE9d56UUaWXapbdEg7yi.txt
2011-03-01	CNN	2011: Rodney King's nightmare	https://www.cnn.com/videos/us/2011/03/01/lemon...	LUqBcztJGTLPqCi0pfHH77PAy7thhgZO.txt
2011-03-01	CNN	Is that really deductible?	https://www.cnn.com/videos/us/2011/03/01/chern...	L4jBakg8p2HXXzVY9hKwoLE0QQdpg8B3.txt

Fig 3.6 shows modified formatting version of the Data Table indexed by date

Similar way data has been collected in The Guardian

Problems Faced:

- Exceptions are incorporated inside the code to avoid internal server error, forbidden request or url expired, url has suspicious cookies
- If the url is not responding within the time limit we assume there is something wrong with it and move to the next url with the set configuration timeout limit.
- Per day CNN has a daily request limit of 3500 calls, so this problem has been catered in next stage.

- Proxies used for Guardian were resulting slowness as those IPs were publically distributed over the internet and hence others might be leveraging their usage.

3.2 Improvement

3.2.1 Proxies

Since many websites have IP block mechanism, it is important to find a way to prevent IP block when running web crawler, or it will cost too much time. The main source data is from CNN. After testing, we can find that the IP will be blocked after visiting CNN server for nearly 3 thousand times. Then it costs a long period of time(usually 24 hours) to be able to visit and gather data again. Thus, we might need proxies server to change our IP address when visiting the CNN server. The diagram of the proxy server is shown below:

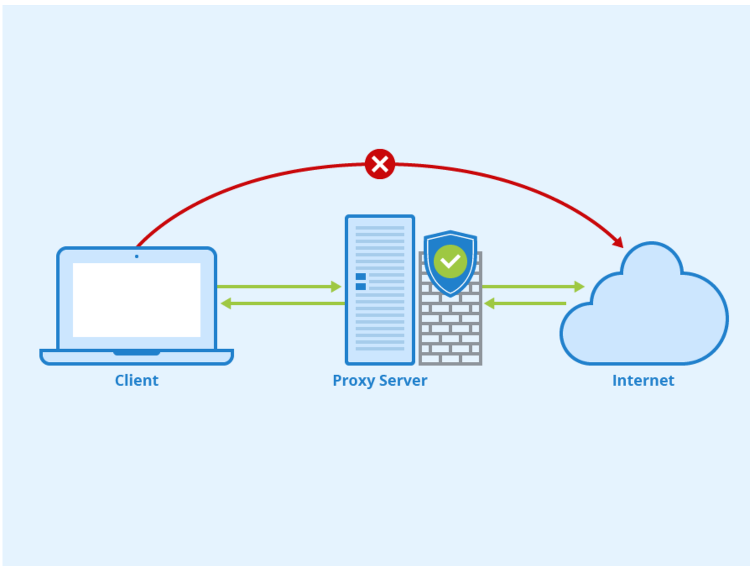


Figure 3.7

and a sample proxy server is like below:

figure 3.8

However, some problems might occur when requesting to the CNN server via proxy server. First, since we need to request nearly 80 thousand articles, and each IP can only request about 3 thousand articles then being prohibited, we need about 25 to 30 IPs to finish the data gathering. It is very difficult to find so many available proxy servers. Second, as the image shown above, the price of proxies is much high. Here is a price plan for a certain proxy server:

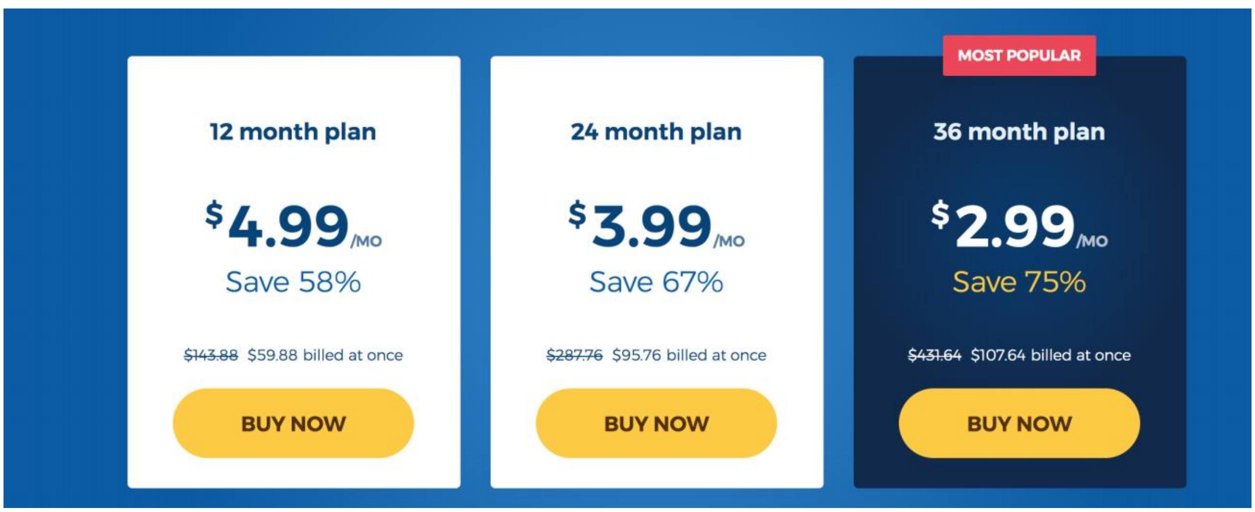
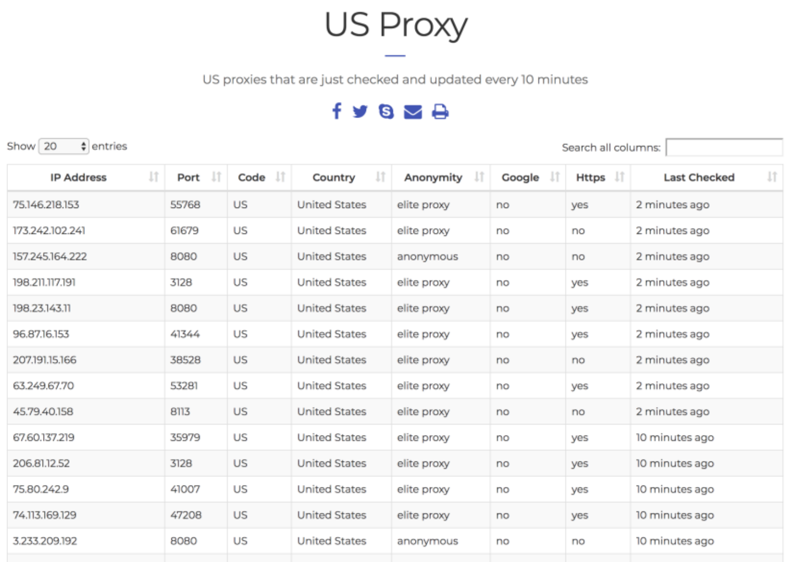


figure 3.9

Therefore, it is unrealistic to directly use the proxies, we have to find some more suitable solutions.

3.2.2 Rotating IPs

In addition to the direct proxy server, it is possible to just use rotating IPs. Rotating IP is a method that a project change the IP frequently to avoid IP block. It usually needs a large amount of IP, and these IP can be available for just a short time, maybe some minutes to an hour. There are many services that can provide the proxy IPs. A sample website is shown below:


US Proxy

US proxies that are just checked and updated every 10 minutes

Show 20 entries

Search all columns:

IP Address	Port	Code	Country	Anonymity	Google	Https	Last Checked
75.146.218.153	55768	US	United States	elite proxy	no	yes	2 minutes ago
173.242.102.241	61679	US	United States	elite proxy	no	no	2 minutes ago
157.245.164.222	8080	US	United States	anonymous	no	no	2 minutes ago
198.211.117.191	3128	US	United States	elite proxy	no	yes	2 minutes ago
198.23.143.11	8080	US	United States	elite proxy	no	yes	2 minutes ago
96.87.16.153	41344	US	United States	elite proxy	no	yes	2 minutes ago
207.191.15.166	38528	US	United States	elite proxy	no	no	2 minutes ago
63.249.67.70	53281	US	United States	elite proxy	no	yes	2 minutes ago
45.79.40.158	8113	US	United States	elite proxy	no	no	2 minutes ago
67.60.137.219	35979	US	United States	elite proxy	no	yes	10 minutes ago
206.81.12.52	3128	US	United States	elite proxy	no	yes	10 minutes ago
75.80.242.9	41007	US	United States	elite proxy	no	yes	10 minutes ago
74.113.169.129	47208	US	United States	elite proxy	no	yes	10 minutes ago
3.233.209.192	8080	US	United States	anonymous	no	no	10 minutes ago
167.172.21.60	3128	US	United States	elite proxy	no	yes	11 minutes ago

figure 3.10

Like the figure showing above, these IPs have a short life time, and can be used by the web crawlers to be proxies. However, it is easy to find these IPs are unavailable to gather data from CNN or other popular websites. The reason is that these IP are public, so many automatic web crawlers may gather them once they are released. So It is better to get proxy IPs from service with charge.

We got some charged IPs and downloaded them as text files like this:

```
24.113.141.227:48678
104.236.156.59:80
24.172.82.94:53281
50.210.111.187:8080
209.250.3.38:80
209.250.3.7:80
209.250.3.35:80
205.185.115.100:8080
76.250.137.241:8080
208.67.183.240:80
173.249.35.163:655
24.113.38.149:48678
198.98.55.168:8080
65.182.5.212:8080
165.22.45.183:80
209.250.3.42:80
50.197.38.230:60724
67.218.155.47:3128
209.250.3.72:80
209.250.3.94:80
74.83.246.125:8081
208.114.192.126:8080
98.190.250.150:48678
209.250.3.97:80
199.195.248.24:8080
```

figure 3.11

Then we need to change some configuration of the code and API to make sure that we can gather data by valid proxies.

3.2.3 Configuration Modification

Since we use the newspaper API to download and parse the articles posted by CNN, we need to handle with configuration of newspaper API to ensure that it can use the proxies properly. We read the change the related code of the API, added all the IPs, and modified the code to let the API use a random valid proxy IP in every single time. The brief workflow is shown as below:

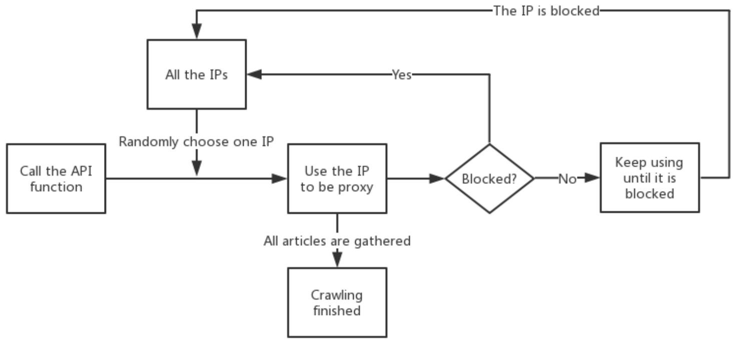


Figure 3.12

As the figure shown above, every time we call the Article API(The API that catch and download the content of a web page), we have to use a proxy randomly from all the valid IPs. We analyze the original code in the API, and modify the related code as below:

```
proxies_candidates = [
    '24.113.141.227:48678', '104.236.156.59:80', '24.172.82.94:53281', '50.210.111.187:8080', '209.250.3.38:80'
    '209.250.3.7:80', '209.250.3.35:80', '205.185.115.100:8080', '76.250.137.241:8080', '208.67.183.240:80',
    '173.249.35.163:655', '24.113.38.149:48678', '198.98.55.168:8080', '65.182.5.212:8080', '165.22.45.183:80',
    '209.250.3.42:80', '50.197.38.230:60724', '67.218.155.47:3128', '209.250.3.72:80', '209.250.3.94:80',
    '74.83.246.125:8081', '208.114.192.126:8080', '98.190.250.150:48678', '209.250.3.97:80',
    '199.195.248.24:808'
]
self.proxies = {'http': random.choice(proxies_candidates)}
```

Figure 3.13

Thus, before the API is called, it will choose an IP to be the proxy and then start to catch. If it is not working for requesting, we will change randomly choose another IP to restart it. The situation of not working consists two main possible scenarios, one is connecting too slowly, another is already exceeded the limitation of the server requests.

After multiple times of testing, we use about 50 IPs, and around 30 IPs are valid to be used.

3.2.4 Result

After applying this method, the time cost of article gathering is rapidly reduced. We used about 30 valid IPs, and catch all the data(probably about 70k) in 5 hours. Compared with the time cost we previously assume, that is use one IP until it is blocked and try it again after that, the total time is saved about 70%.

4. Preprocessing

- SiteMap contains duplicate urls and we need to get rid of duplicate entries to avoid training on redundant articles.
- Once we get distinct urls, we fetch the actual data content and save it in a file. This contained unicode data, bitmap translated data, so retained those data we have used encoding in the file.

As we have used NLP to get the text sentiment where positive or negative. So cleaning the data is important feature to let NLP models trained with efficacy.

Preprocessing script does the following thing:

1. Clean extra spaces, special characters, single characters, new lines, make it to lowercase.
2. Removing the stop words using stop words library from nltk
3. Lemmatizing the filtered words to make get the parent word and get rid of participle, tense form of the words used.
4. The script managed to discard 50% of the verbose stuff in a file which is great improvement as a preprocessing steps.

5. NLP

5.1 Purpose

Binary classify each article with positive and negative label for two purposes:
*For statistics analysis about the newspaper perspective in past years.
*For users' query about the sentiment binary results for each article.

5.2 I/O requirement

After getting the data preprocessing .txt file from each month in the past years.
Generating the corresponding results:
1. with five classifiers results and final voting result + name of file + year and month info. in .csv files for users' query
2. with five classifiers statistic results and final statistic voting result for information visualization

5.3 Algorithm Introduction

There are mainly three steps for this NLP algorithm:

1. Used a large amount of positive / negative adjective as the input to training the models (instead of the articles for general and objective purpose)
2. Training the data based on five different classifiers, the five NLP algorithms are used are Naive Bayes, MultinomialNB, BernoulliNB, LogisticRegression, and LinearSVC.
3. Voting for the most confident result as the final result (each classifier might have their own corner cases, by doing so can eliminate the outliers for each classifier)

Then we test the input .txt file based on this algorithm and get the final binary classified result.

5.4 Implementation

For the step one:

We use the known positive words and negative words to train our classifier models.

short_pos = open("trainning_files/positive.txt","r", encoding='iso-8859-1').read()
short_neg = open("trainning_files/negative.txt","r", encoding='iso-8859-1').read()

positive.txt

```
wonder
wonderful
wonderfully
wonderous
wonderously
wonders
wondrous
```

negative.txt

```
whore
whores
wicked
wickedly
wickedness
wild
wildly
```

One thing need to mention is that we do not need to token the article (unlike the testing articles) because we have already separate each adjective word into separate line.

For the step two:

We use the nltk API to help us build the five classifier models. We use 80% input data for training, and 20% input data for validating. After the training, we just used the python API to pickle the models information and parameters into some .pickle files. By doing so, it can save our time to training the models each time when we want to use these models. We just simply load the previous saving pickled models for our testing.

```
total length (surpervised learning):  6791
trainning set number (80%):  5433
testing set number (20%):  1358
```

Fig 5.1 (training set 80% testing set 20%)

```
save_classifier = open("pickled_models/MNB_classifier.pickle","wb")
pickle.dump(MNB_classifier, save_classifier)
save_classifier.close()
```

Fig 5.2

For the step three:

After getting each classifier result, simply voting for the most confident result

```
for c in self._classifiers:
    v = c.classify(features)
    votes.append(v)
choice_votes = votes.count(mode(votes))
```

Fig 5.3

5.5 Verification and Testing

First simply testing the three sentences:

"This article was rich, clear, willing, ingenuous, attractive, sensational, and hot"

"This is the best marvellous, imaginative, and realistic one I have seen"

"This article was utter junk. There were absolutely 0 points. I don't see what the point was at all. Horrible essay, suck"

with the corresponding result shown below:

```
['pos', 'pos', 'pos', 'pos', 'pos']
('pos', 1.0)
['pos', 'pos', 'pos', 'pos', 'pos']
('pos', 1.0)
['neg', 'neg', 'neg', 'neg', 'neg']
('neg', 1.0)
```

Fig 5.4

The results shows the individual results and the final voting result.

And then we test the whole years data and got the results like:

The static results for pos/neg number of articles in Feb. 2015 with five classifiers result and the final voting result (partly):

166	158	107	168	116	149	2015	2	pos
561	569	620	559	611	578	2015	2	neg

Fig 5.5

The results for pos/neg number of each article in Feb. 2015 with five classifiers result and the final voting result and the file name (partly):

neg	neg	neg	neg	neg	neg	887rQqB6Hur8gS9Su33jwziLHcspuq28_pp.txt	2015	2
neg	neg	neg	neg	neg	neg	8bTzZIkbBP4gDj2TA24hzS0hJdqlhgca_pp.txt	2015	2

Fig 5.6

5.6 Result Analysis

The accuracy we got after using 80% input data for training, and 20% input data for validating for five models. We can find that they are normally 72% accuracy for eac model (shown in below figure and table). The first figure below shown the top 15 words contributed the most info. among the features set for the NB models. Thus if we want to decrease the redundant feature dimensions based on this information, like do the PCA with only several important features.

```
Original Naive Bayes model accuracy percent: 72.16494845360825
Most Informative Features
                    free = True              pos : neg    =     10.9 : 1.0
                   clear = True              pos : neg    =      8.6 : 1.0
                  famous = True              pos : neg    =      5.5 : 1.0
                    best = True              pos : neg    =      5.5 : 1.0
                    safe = True              pos : neg    =      5.5 : 1.0
                   sharp = True              pos : neg    =      5.5 : 1.0
               effective = True              pos : neg    =      4.2 : 1.0
              attractive = True              pos : neg    =      3.9 : 1.0
               equivocal = True              pos : neg    =      3.9 : 1.0
                  static = True              pos : neg    =      3.9 : 1.0
                   noble = True              pos : neg    =      3.9 : 1.0
             sensational = True              pos : neg    =      3.9 : 1.0
                 envious = True              pos : neg    =      3.3 : 1.0
                 willing = True              pos : neg    =      3.3 : 1.0
                creative = True              pos : neg    =      2.4 : 1.0
MNB_classifier accuracy percent: 72.60677466863034
BernoulliNB_classifier accuracy percent: 72.23858615611192
LogisticRegression_classifier accuracy percent: 72.38586156111928
LinearSVC_classifier accuracy percent: 72.82768777614137
```

Fig 5.7

Also, the average executing time (around 16800 .txt files) for each classifier are:

classifier	NB	MultiNB	BinaryNB	Logistic	SVC
accuracy percentage	72.16%	72.61%	72.24%	72.38%	72.83%
executing time	0.00639	0.00191	0.00184	0.00219	0.00177

```
each classofoers average calculating time in sec:  [0.006393251199988299, 0.0019089575999787485,
0.0018411747999889485, 0.0021917007999800262, 0.0017657084000275063]
```

Fig 5.8

5.7 Future Improvement

We can also replace the different classified models for this voting algorithm, like the Random Forest, Adaboost, XGboost, and etc. And we can also do the PCA for reducing the time complexity. We can also try to add more training data set for positive and negative words (it can also be chosen according to different fields)

Topic Modeling can be applied to learn the cateogories of the article which can be pivotal to do comparative study of each section like crimes, business, politics etc.

5.8 Conclusion

We use the voting way to average the hidden corner cases for each kind of classifiers and. We can also find that the simple algorithm like Naive Bayes seems having the similar result as the others in this binary classification because the binary classification might do not need too complicated structure to train the data.

6. Data Visualization

We get the sentimental result utilizing 5 different algorithms. The data we want to display is not about a single article but the statistical analysis. So we divide the visualization to 3 parts.

1. Full picture of ten years
2. 12 months changes in a year
3. Any 2 years sentimental trend comparison

6.1 Full picture of ten years

In each year, we show the number of positive articles above the x axis and negative ones below x axis.

6.2 12 months changes in a year

We can query a single year. For a single year, we show the statistical results in both bar graph and pie graph.

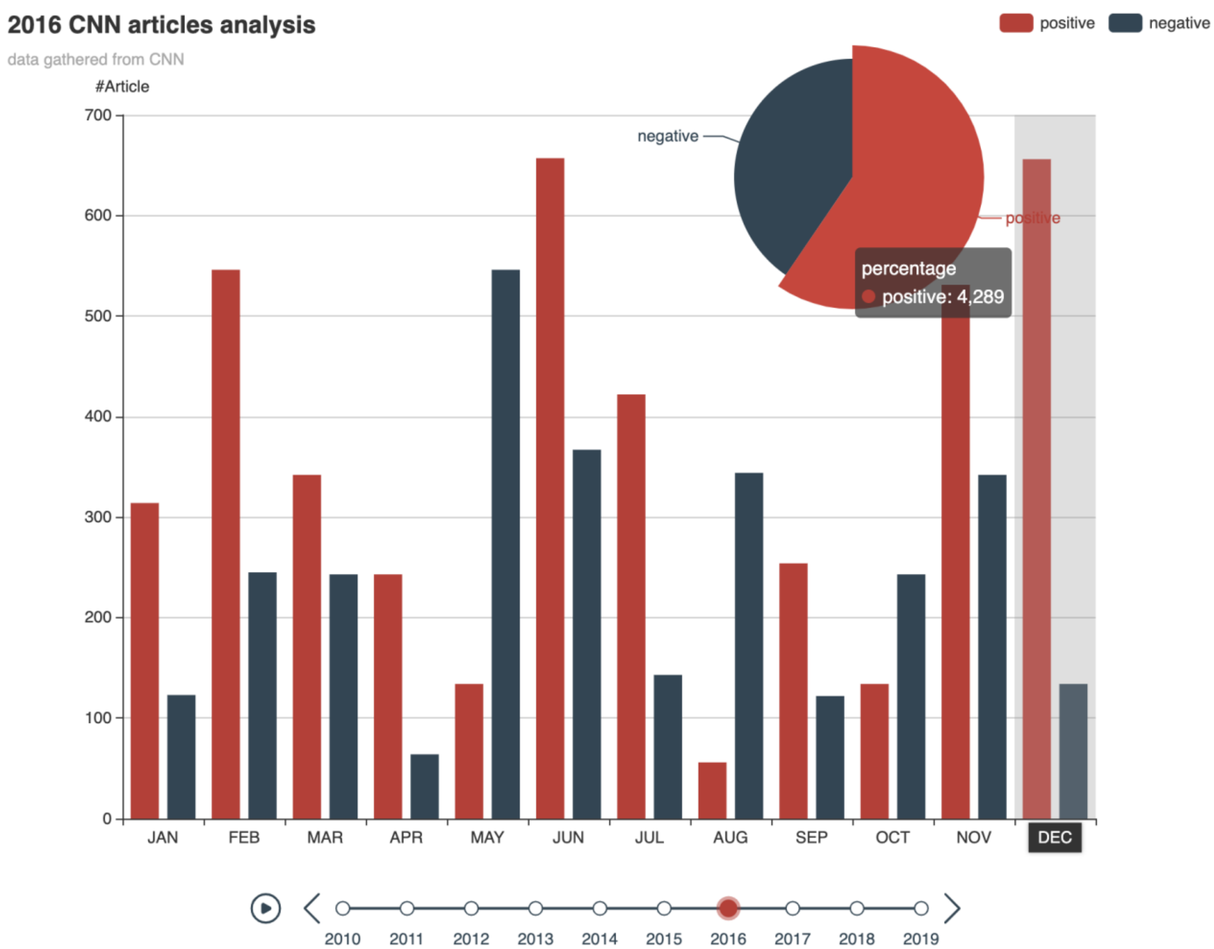


Fig 6.1

6.3 Any 2 years sentimental trend comparison

In addition, we provide a function to compare two different years to visualize the trend of pos and neg and tell the difference from the line graph.

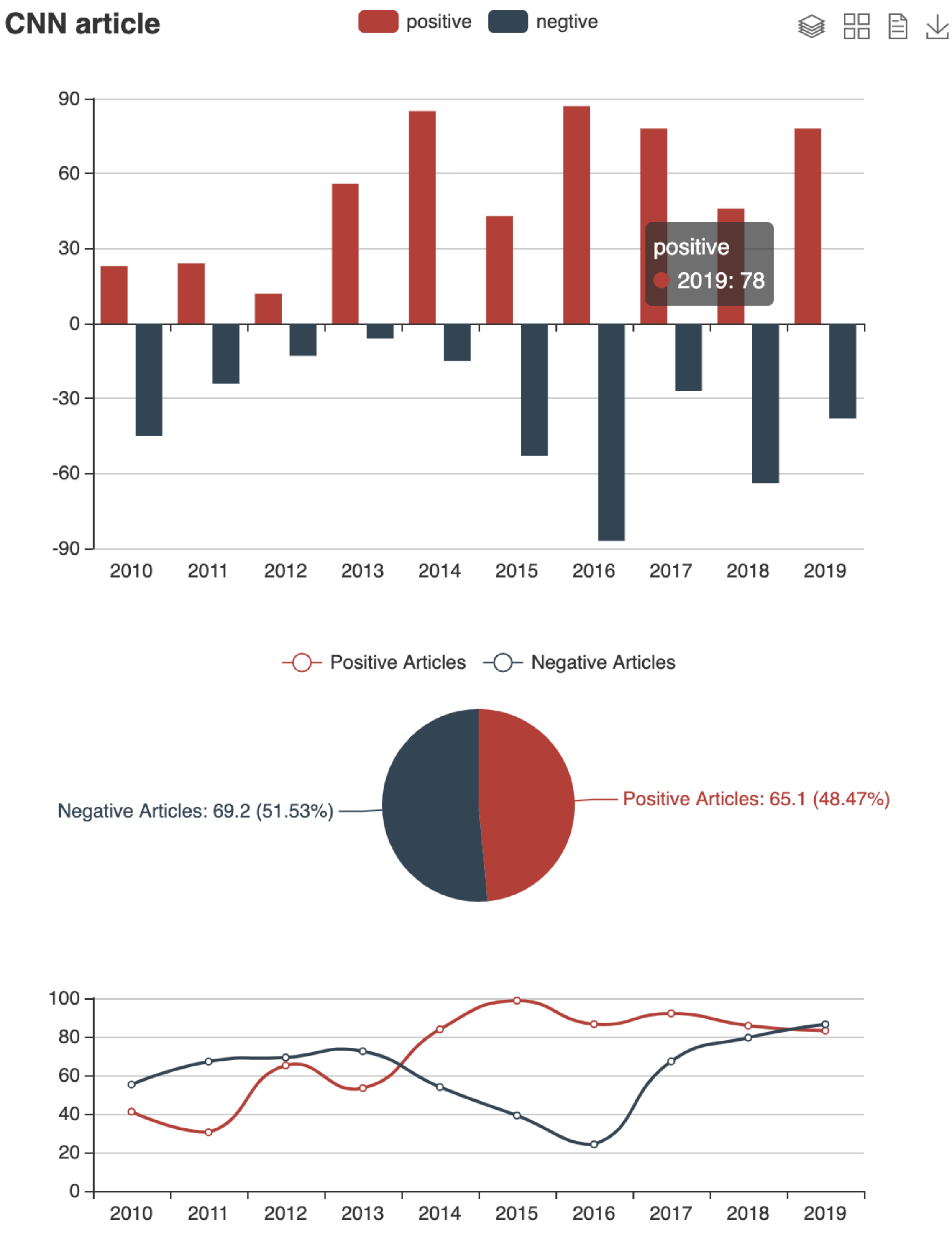

CNN article
positive
negtive
90
60
30
0
-30
-60
-90
positive
2019: 78
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Positive Articles
Negative Articles
Negative Articles: 69.2 (51.53%)
Positive Articles: 65.1 (48.47%)
100
80
60
40
20
0
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019

Fig 6.2

6.4 Production Deployment Urls:

All work has been deployed using Microsoft Azure AWS.
Below are the production urls: To view the website

1. https://newsarticlessentimentanalysis.azurewebsites.net/api/sentiment_engine?code=WXK9ko1U88HTrfB3oiOyFDsJDGpAa6JAuzLRjCNNkRoPInQNUqASKw==&name=web.htm
2. http://newsarticlessentimentanalysis.azurewebsites.net/api/sentiment_engine?code=WXK9ko1U88HTrfB3oiOyFDsJDGpAa6JAuzLRjCNNkRoPInQNUqASKw==&name=comparison.html
3. http://newsarticlessentimentanalysis.azurewebsites.net/api/sentiment_engine?code=WXK9ko1U88HTrfB3oiOyFDsJDGpAa6JAuzLRjCNNkRoPInQNUqASKw==&name=fancy.html
4. http://newsarticlessentimentanalysis.azurewebsites.net/api/sentiment_engine?code=WXK9ko1U88HTrfB3oiOyFDsJDGpAa6JAuzLRjCNNkRoPInQNUqASKw==&name=posvsneg.html

7. Resources:

1. Python newspaper documentation https://buildmedia.readthedocs.org/media/pdf/newspaper/latest/newspaper.pdf
2. Text Summarization of an Article: https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1
3. Insights about Nltk library: https://medium.com/datadriveninvestor/python-data-science-getting-started-tutorial-nltk-2d8842fedfdd
4. Usage of MultiCore Processing: https://medium.com/python-pandemonium/how-to-speed-up-your-python-web-scraper-by-using-multiprocessing-f2f4ef838686
5. How Lemmatization works: https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/
6. Data Source : https://www.cnn.com/us/article/sitemap-{yyyy}-{mm}.html, starting from 2011-07 till 2019-12-07
7. Data Source : https://www.theguardian.com/us-news/{yyyy}/{mm}/{dd}/all starting from 2008/jan/01 day till 2019/dec/31
8. Data is uploaded on GitHub :https://github.com/Darshansol9/News_Articles_Sentiment_Analysis