# Data Collection:

The aim of the project was to extract data from Pakistani online news articles, and then use information gained from the data for valuable insights. The first step to this, of course, was collecting this data.

For this purpose, our programming language of choice was Python. We used the **scrapy** library in Python to create 2 spiders which would crawl through the **Dawn Newspaper** website’s archives and access the links to the articles.

The need to use 2 spiders requires some explanation. Let’s first start with explaining the first spider: ‘news.py’.

**news.py**

Let’s first start by explaining the code. This spider accesses the Dawn News website’s archive pages for each day in a month, as specified (look at important variables). It accesses the links for all articles available on the page, each of which specifies an article published on that date, and then accesses each article’s webpage. From that webpage, it extracts the heading and text of the article, as well as the link of the article’s webpage, and stores all this information in a single record using scrapy’s ‘yield’ function. It first accesses the first url (start\_url: look below in key variables), changes the date of the archive page in the link and then accesses that, continuing for 30 days.

Following are the key variables that need to be altered to set the correct timeframe of the articles to be collected from Dawn’s website:

* **year\_this (String):** Year of publication of the article(s) which need to be extracted
* **month\_this (String):** 3 letter form of month of publication of the article(s) which need to be extracted
* **name (String):** Name of the spider to be shown on the cloud
* **allowed\_domains (List of Strings):** Allowed domains for all urls to be accessed by the spider
* **start\_urls (List of Strings):** The first url to be accessed by the spider
* **DOWNLOAD\_DELAY (Double):** Delay in seconds between each new GET Request for a url, which prevented the spider from being blocked, which could have resulted in the loss of news articles (recommended value: 3.0)

For further details, refer to the code itself which has been adequately commented and can be found using the following path: Data Collection/scrapy project/news\_crawler/news\_crawler/spiders/news.py

Details about the various functions of the scrapy library can be found here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initially, news.py was also used to extract the date of publication from the article’s webpage, but this caused issues as the date on the articles webpage was not necessarily the actual publication date, but often was the date when the article was updated, which would cause problems in a time-series data analysis.

This had an obvious solution: save the date according to the archive page’s date. There were 2 constraints to this:

1. We were not able to find a way to pass in parameters to the ‘parse\_details’ function (look at code for further details) which prevented us from passing in the date of the archive page when extracting and storing information on the article from the webpage’s link.
2. Using a global variable to store the date for each archive page was not viable because the code was running asynchronously.

Hence, there was a need to create another spider to access the dates and links of each article, along with other useful information including short excerpts and article categories from just the archive webpage itself, and then combine the data collected from both spiders to get a complete and useful dataset.

**cats\_spider.py**

The second spider is called cats\_spider.py. This spider accesses the Dawn News website’s archive pages for each day in a month, as specified (look at important variables). It accesses all available links on the page, each of which specifies an article published on that date, along with it’s excerpt and category. It then stores the link, excerpt and category, along with the date of the archive page (as all this is now done in a single function: ‘parse’) using scrapy’s ‘yield’ function.

Following are the key variables that need to be altered to set the correct timeframe of the articles to be collected from Dawn’s website:

* **year\_this (String):** Year of publication of the article(s) which need to be extracted
* **month\_this (String):** 3 letter form of month of publication of the article(s) which need to be extracted
* **name (String):** Name of the spider to be shown on the cloud
* **allowed\_domains (List of Strings):** Allowed domains for all urls to be accessed by the spider
* **start\_urls (List of Strings):** The first url to be accessed by the spider
* **DOWNLOAD\_DELAY (Double):** Delay in seconds between each new GET Request for a url, which prevented the spider from being blocked, which could have resulted in the loss of news articles (recommended value: 2.0)

**merge.py**

Results from both spiders then needed to be merged. To do this, we created **merge.py**. The resulting file with the complete results is **ALL\_complete.csv**.

Initially, we simply used pandas and merged on the basis of the link field in both results. This though, created issues as the links for articles on the archive pages (that got stored in the results for cats\_spider.py) were redirected to a new link for the actual webpage of the article (that got stored in the results for news.py). This caused a huge number of articles to be missed out in the final result. Hence, for all those links from the cats\_spider.py results for which we couldn’t find a matching link in the news.py results, we used the ‘get’ function in Python’s ‘request’ library to get the final redirected link and then merge the results on the basis of that.

We could just have simply used the ‘Heading’ field to merge the 2 results, but that too caused articles to be missed out as when links were redirected, it was found that sometimes the heading of the article too was changed from the one on the archive page. Please note that for this reason, the heading, link and of course text of the article is extracted from the results of news.py, whereas the excerpt, category and date (read above for reason for date) is extracted from the results of cats\_spider.py.

We could also have simply used the ‘get’ function on all links and then merged on the basis of those new redirected links, but the ‘get’ function was found to be quite time-consuming, and the current solution was found to be much more time-efficient.

**ALL\_complete.csv**

Finally, the file: **ALL\_complete.csv** was created which included all articles on the Dawn Newspaper’s website from 1st July, 2017 to 31st May, 2018. It contains the following fields:

* **category**: The category of each article as done by Dawn’s website itself. The possible categories are as follows: ‘Blogs’, ‘Business’, ‘Entertainment’, ‘Home’, ‘Newspaper’, ‘Pakistan’, ‘Sport’, ‘Tech’, ‘World’
* **excerpt**: A few lines from the article that Dawn’s website’s authors felt gave a basic description of what the article talks about
* **final\_date**: The date of publication of the article.
* **heading:** The heading of the article as mentioned on the article’s actual webpage
* **link:** The link of the article’s actual webpage
* **text:** The actual text of the article

This was the dataset used to run and explore LDA and it’s possible uses for online Pakistani newspaper analysis.

How to run the code for Data Collection:

Required python libraries:

* scrapy (to install, run command: ‘pip install scrapy’)
* shub (to install, run command: ‘pip install shub’)

Step 1:

Considering the amount of time this requires, and the need to keep the code running consistently, we needed to upload the spider on a cloud. For that we used ScrapingHub, which is an online cloud service provided that allows us to run scrapy spiders. For this, follow the following steps:

* Create an account:
  + Go to <https://scrapinghub.com/scrapy-cloud> and click the ‘SIGN IN’ Button. From there, click ‘REGISTER NOW!’ and follow the registration process.
  + Once registered, sign in.
* Pay Bill to allow for spider runs of over one hour:
  + To allow for your spider to run for over an hour (it requires 3-4 hours) before being terminated, you will need to pay a fees of $9/month.
  + Sign in, and then on the left bar, under the ‘Overview’ section, click on ‘Billing’.
  + Press the ‘Select’ button under the ‘Scrapy Cloud’ section.
  + Set the number of units to ‘1’.
  + Then click ‘Subscribe’ button
  + Now, enter your credit card details and pay the required amount.
  + Please do note that the payment for the monthly subscription will continue until you unsubscribe.
* Create a Project:
  + Now, go to the top bar and open the drop-down menu, ‘Scrapy Cloud’ and click ‘Create Project’, enter an appropriate name, and click ‘CREATE’
* Upload the spider:
  + Go to the Dashboard which can be found under the ‘JOBS’ section in the bar on the left
  + Now, your url should look something like this: <https://app.scrapinghub.com/p/317929/jobs>
  + Copy and save the number in the url, which in our example would be ‘317929’
  + Now, install Python’s ‘shub’ library by entering the command ‘pip install shub’ in the command prompt
  + Once installation is complete, enter the command: ‘shub login’
  + It will then prompt you to enter the ‘API key’. To get your API key, enter the following url: <https://app.scrapinghub.com/account/apikey>
  + From there, copy the API key, paste it in the command prompt and press the ‘Enter’ key
  + Go to Data Collection folder and then go through to the file path: Data Collection/scrapy project/news\_crawler
  + Next, enter the command: ‘shub deploy’
  + It will prompt you for the ‘Target project ID’. For this, go to the number you copied and saved, and enter it there.
* Set the Download Delay:
  + Under the ‘SPIDERS’ section on the left-hand side bar, click on ‘Dashboard’.
  + Click on the appropriate Spider according to the name set in the spider’s code (look above).
  + Click on ‘Settings’ and then click on the small ‘+’ icon found on the page.
  + This will add a single setting. To set it, click the dropdown menu and choose ‘DOWNLOAD\_DELAY’ and set it’s value to ‘3’. Now save the settings by clicking the ‘SAVE’ button.
* Run the Spider:
  + Next, click the ‘RUN’ button right next to the ‘WATCH’ button on the right side.
  + Set ‘Job Units’ as 1
  + Set the priority depending on whether you want this running in order of uploads (Normal), before (‘Lowest’/’Low’) or after(‘Highest’/’High’) other jobs waiting to be run. For our purposes though, we usually just set the priority as ‘Normal’.
* View Progress and Download the Results:
  + Go back to the Jobs Dashboard by clicking ‘Dashboard’ under the ‘JOBS’ section on the left-hand side tab.
  + There, under ‘Running Jobs’ you can view the progress of your spider. Give it around 5 minutes. If the Job has stopped running due to an error (if so, it will be now found in the ‘Completed Jobs’ section with very few items scraped), upload and run the spider again. The error is probably because of a possible 403 error on accessing the start\_url which prevented the spider to move to the next urls. Please look in the ‘news\_spider.py’ section for more details.
  + Once the job is completed, it can be found under the ‘Completed Jobs’. This should take around 3-4 hours.
  + Next, click on the ‘Items’ tab in your relevant completed job. Now, click on ‘EXPORT’, choose the appropriate file type to store the scraped data, and the download should begin automatically. Save the file with the following name format: ‘\_<3-letter form of month>\_news.<file\_extenstion>’

Step 2:

* In the Data Collection folder, open command prompt and enter the following code:

scrapy runspider <name of spider’s code file> --logfile <filename of log of spider’s working> -o <name of results-containing file, along with extension, which should have format: cats\_<3-letter form of month>.<file\_extenstion> >

e.g. scrapy runspider cats\_spider.py –logfile log.txt –o cats\_jan.csv

(possible extensions for the results-containing and log files include ‘.json’, ‘.csv’ and ‘.txt’ to the best of my knowledge)

* The results will be stored in the file name as inputed in the command(cats\_jan.csv) in our example.

Step 3:

* Simply run merge.py in same folder as that containing the results from both spiders to get ALL\_complete.csv

Topic Modelling:

\*The following codes can be found in Yearly LDA/code

The following codes are run in the given order:

**create\_docs.py**: sorts out the data from ALL\_complete.csv according to the date, as well as remove any irrelevant categories, given the kind of analysis you wish to do, hence creating data.csv which well then be used by the following code.

**LDA\_1.py**: uses the nltk and genism libraries to process the data by removing stopwords and punctuation, as well as stemming the words.

**LDA\_2.py**: runs the actual model on the corpus with varying number of topics (all even numbers between 2 and 40)

**lda\_choose.py**: finds the coherence values of the each LDA model and thus allows us to choose the better models to use (the higher the coherence value, the better)

**doc\_to\_topic\_dist.py**: creates document to topic distribution, which has the perecentage representations of all topics for each document

**lda\_extract\_topic\_rep\_csvs.py**: creates list of document headings along with the representation of a topic for that topic, for all topics, with a separate csv file for each topic

# **Results**

### (LDA was run multiple times on the dataset, but only the more prominent results are shown)

#### **Prominent Results No. 1 (PR-1)**

* The dataset includes all categories and the entire range of available articles.
* The number of topics was set to 20.
* Top words from each topic the more interesting, relevant and interpretable topics:
* Headings of the articles in descending order of the topic representation were viewed, and a consistent manual threshold was set for all topic models.

*Note that while some topics remain particularly consistent, those with spikes do not necessarily have a very concrete reason behind it. This could be due to chance but also, when looking at a difference of 10-15 articles, there isn’t necessarily a particular reason behind that.*

##### Topic 0 (Cricket articles)

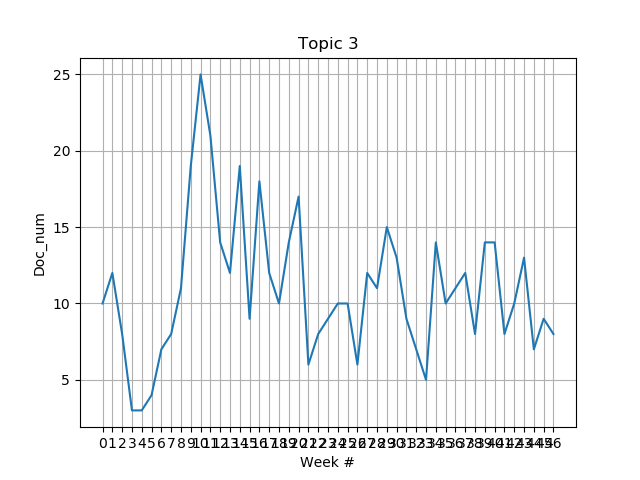
*Keywords:*

test, England, match, cricket, first, run, play, ball, say, team

##### Topic 3 (Rohingya Crisis related articles)

##### *Keywords:*

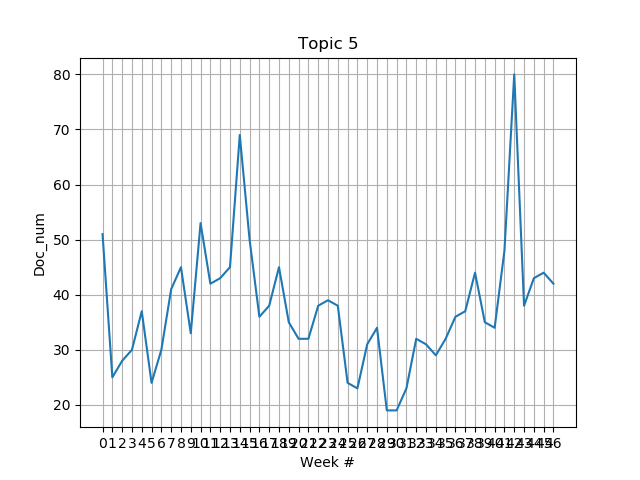
say, myanmar, rohingya, Bangladesh, international, britain, year, country, government, refugee



##### Topic 5 (US foreign relations)

##### *Keywords:*

say, trump, iran, president, nuclear, russia, united\_state, china, state, russian

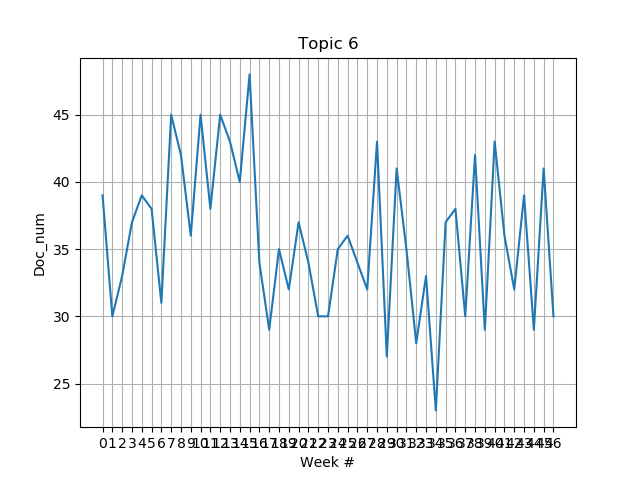


* *Spike # 1: Trump announces sanctions on North Korea, Florida shooting, US-Russia tussle over Ghouta*

##### Topic 6 (Panama Papers Case related articles)

##### *Keywords:*

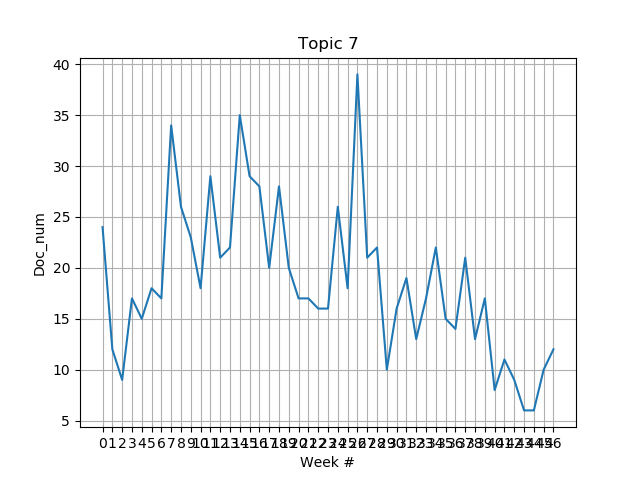
court, justice, case, say, nab, chief, judge, order, also, former



##### Topic 7 (Pakistan Foreign Relations related articles)

##### *Keywords:*

pakistan, say, country, political, party, would, also, government, state, military

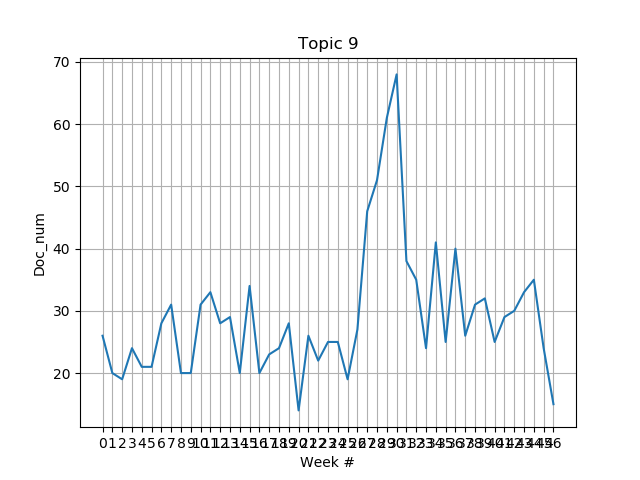


* *Spike # 1: US suspends aid, puts Pakistan on special-watchlist and Trump tweets about Pakistan*
* *Spike # 2: Pakistan placed on FATF grey-list, steps up foreign relations efforts*
* *Spike # 3: Concerns over concerns over construction of kashianga dam in india, articles and public statements on cpec due to upcoming cpec summit 2018*

##### Topic 9 (killings, kidnappings, assaults)

##### *Keywords:*

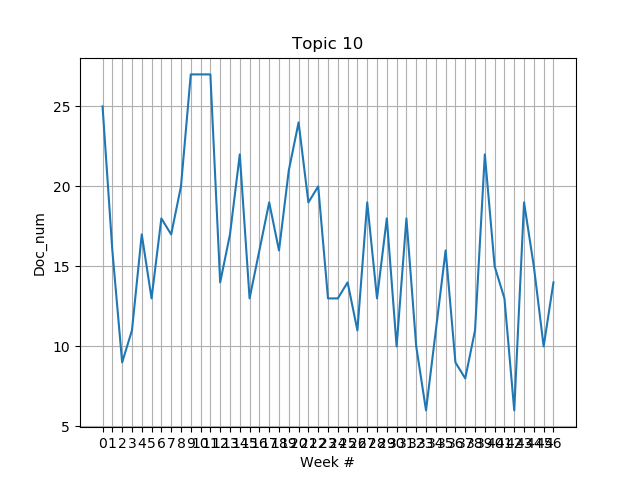
police, say, case, report, arrest, court, suspect, also, officer, investigation



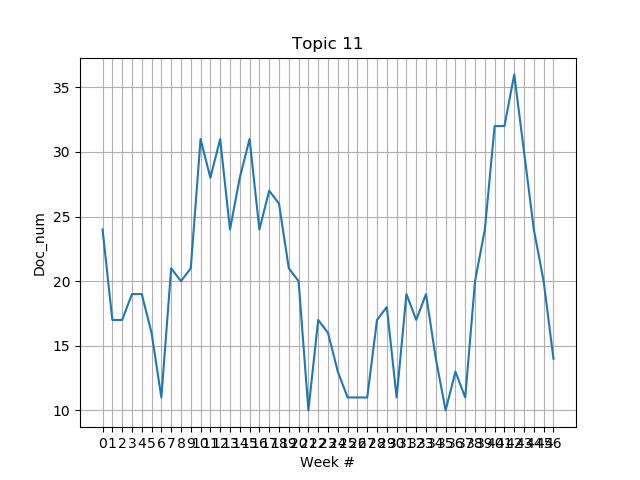
* *Spike # 1: Here, it can be seen that there is a particular rise around January. This could imply that it was due to the Zainab murder case, but this required further research.*

##### Topic 10 (Parliament related news)

##### *Keywords:*

say, party, election, government, minister, would, assembly, chief, pml, pti

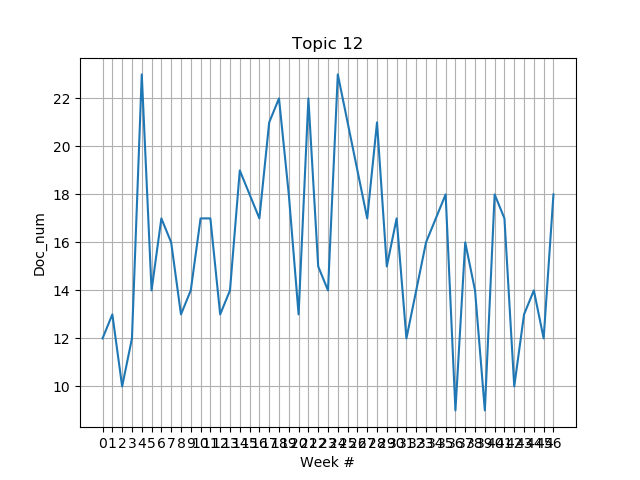
##### Topic 11 (Kashmir-related news)

*Keywords:* india, indian, kashmir, state, journalist, say, medium, people, protest, delhi

##### Topic 12 (Syria, Afghanistan, Iran, Saudi Arabia)

##### *Keywords:*

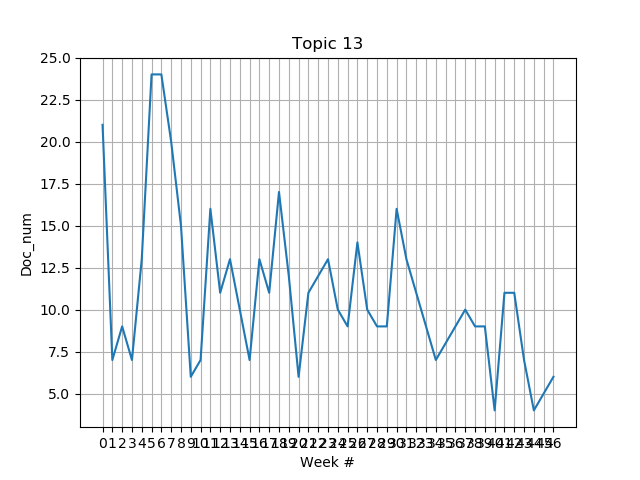
Say, muslim, kill, year, government, people, attack, force, group, state



##### Topic 13 (NAB, corruption charges)

##### *Keywords:*

say, official, land, project, also, airport, railway, security, investigation, authority



#### **Prominent Results No. 2 (PR-2)**

#### LDA was also run on separate months

**Note that here, many of the major events in the separate months were revealed, though if we are to properly automate this process, we need to find a way to differentiate between a relevant topic vs an improper classification of certain articles which do not necessarily have an underlying event, let alone a theme.**

#### October 2017:

* The dataset includes all categories and all articles in the month October 2017.
* The number of topics was set to 18.
* Topic 3: NAB, ECP and PTI related news
* Topic 4: Elections Act 2017, Khatm-e-Nabwooat issue and events relevant to Nawaz Sharif’s disqualification
* Topic 5: Middle-eastern crisis, including Gaza, Libya and Hammas (increasing tenstions between Hammas and Israel over the Gaza strip as well as increasing tensions between Libya and Saudi Arabia)
* Topic 6: News related to Tillerson’s foreign trips to Central Asia, CPEC and US-Pak-India relations
* Topic 8: Cricket-related news
* Topic 9: Economy related news
* Topic 12: Mass-death related incidents .e.g. wildfires and terrorism
* Topic 15: Qandeel murder case as well, including Mufti Qavi’s trials, as well as other reports on sexual harassment or child abuse
* Topic 17: Catalan and Kurd independence

#### November 2017:

* The dataset includes all categories and all articles in the month October 2017.
* The number of topics was set to 20.
* Topic 3: Includes case against Dar, but no consistency
* Topic 2: cricket-related news
* Topic 5: Faizabad sit-in
* Topic 6: Middle eastern crisis, as well as some news on Iran and Afghanistan
* Topic 8: Killings and murders, including honor killings and child abuse
* Topic 9: Investment projects
* Topic 11: Pak-Afghan-US relations (increased pressure on Pakistan to deal with terrorist factions, as well as tensions between Afghanistan and Pakistan over border patrol)

**Comments:**

##### *Note that for October 2017, LDA was able to classify articles on the following major events of the month:*

* Elections Act 2017 and events relevant to Nawaz Sharif’s disqualification
* Increased tensions between Hammas and Israel over the Gaza Strip
* Tillerson’s visit to Central Asia
* Mufti Qavi’s trial for his alleged involvement in Qandeel Baloch’s murder
* Catalan independence movement

##### *And for November 2017:*

* The Faizabad sit-in
* Increased pressure from US on Pakistan to deal with terrorist factions

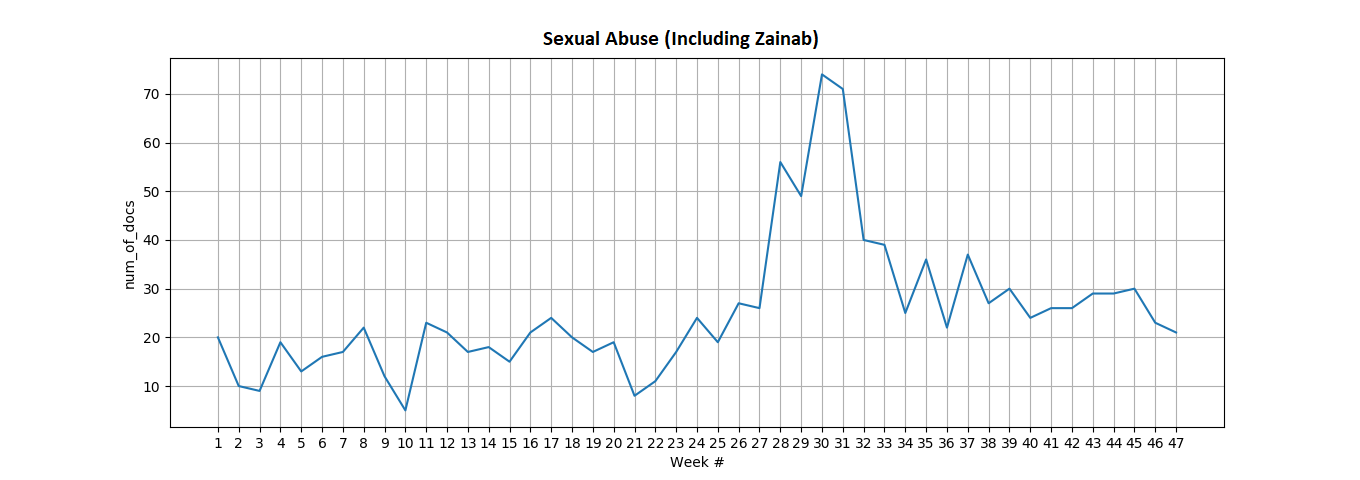
#### **Prominent Results No. 2 (PR-2)**

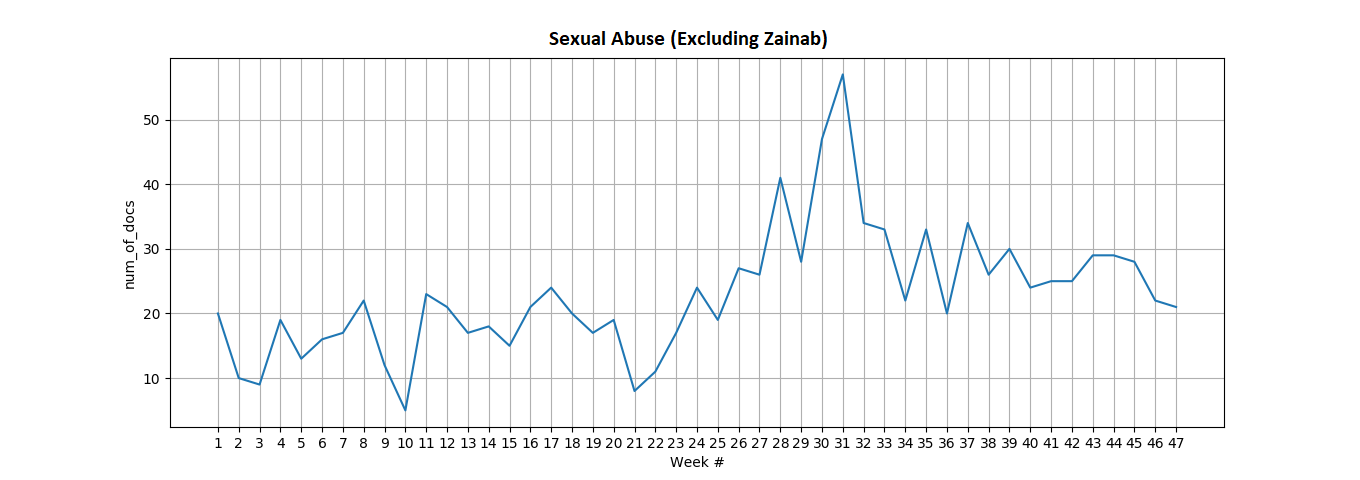
#### Guided LDA

\*The entire code for this can be found in Guided LDA/guidedlda.py

**As previously mentioned, Topic 9 provided some interesting results which required further investigation. The guidedlda library was used to allow for running LDA while also specifying particular keywords for certain topics.**

* The dataset includes only the categories Pakistan, Newspaper and Blogs on all available dates.
* The number of topics used was 20.
* Keywords were only specified for Topic 0, and they were as follows:
* ['rape', 'raped', 'rapes', 'gangrape', 'gangraped', 'harassment', 'harasser', 'harasses', 'harassing', 'harassed', 'harass', 'harassers', 'harrassed', 'harrassment', 'sexual']
* The textmining library was used to create the appropriate document-term matrix for the guidedlda library, which did not have a very efficient stemming code, hence the redundancy in the keywords. Though, it should be noted that the vocabulary was searched for all words containing the series of letters: ‘rape’, ‘harras’, ‘haras’ and ‘sexual’. Then words to be found as relevant keywords were added to the list. Hence, essentially there were only 4 keywords: ‘rape’, ‘gangrape’, ‘harrass’, and ‘sexual’.
* Guided LDA was run twice, once with all articles, and the other without any articles containing the word ‘Zainab’ in the heading, so as to eliminate any possible bias for a certain spike in articles due to the Zainab case. In both cases though, the spike remained, showing that there was a clear rise in sexual abuse cases during that time.



**Comments:**

This was expected. The Zainab case was a huge story and really affected the overall views and reactions of the Pakistanis toward sexual harassment.

One thing though, was quite interesting. While doing a brief overview of the headings of these articles around the time of the spike, we found that famous murder cases like the Mashal Khan, Shahzeb Khan, Perween Rahman and NaqeebUllah Mehsud murder cases were also being mentioned along side within the same topic.

So we mapped out the mentions of these cases within articles, and the results are as follows:

|  |  |
| --- | --- |
| Month # | Month name |
| 1 | July 2017 |
| 2 | August 2017 |
| 3 | September 2017 |
| 4 | October 2017 |
| 5 | November 2017 |
| 6 | December 2017 |
| 7 | January 2018 |
| 8 | February 2018 |
| 9 | March 2018 |
| 10 | April 2018 |
| 11 | May 2018 |

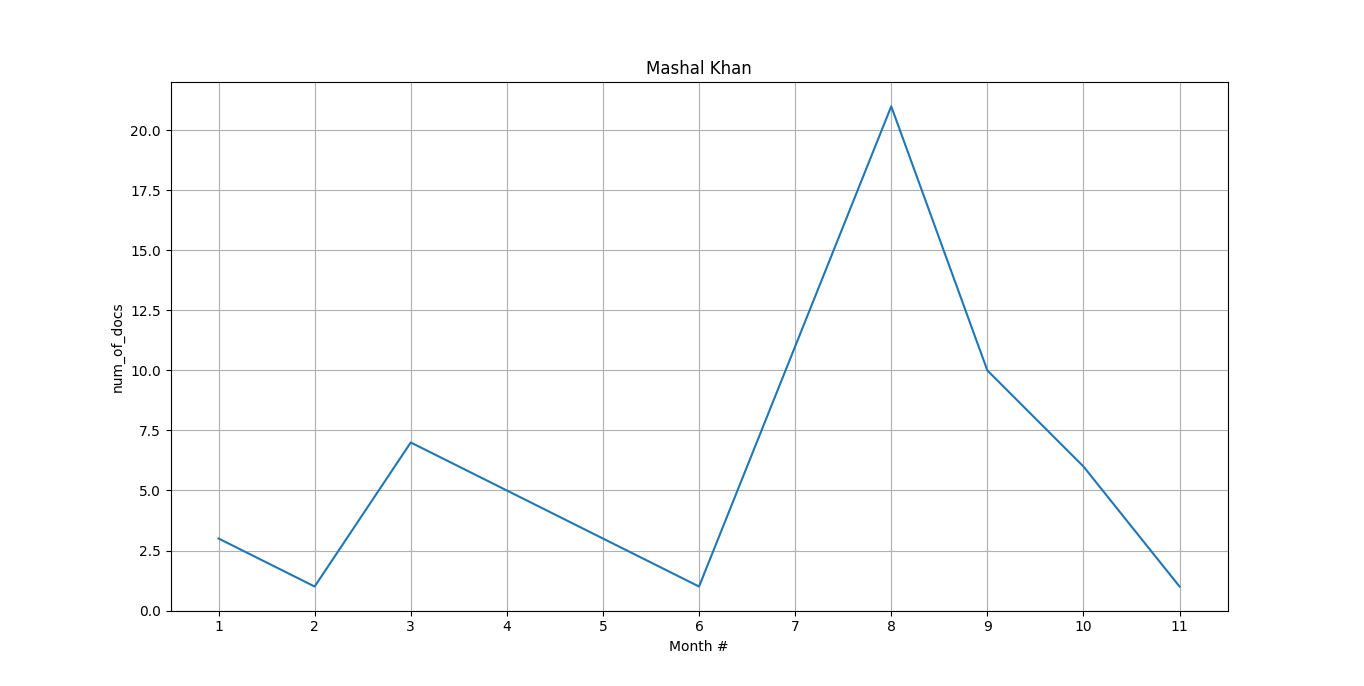


Figure 1: Mashal Khan Murder case articles (Keywords: ‘Mashal Khan’ or ‘Mashal’ and ‘blasphemy’)

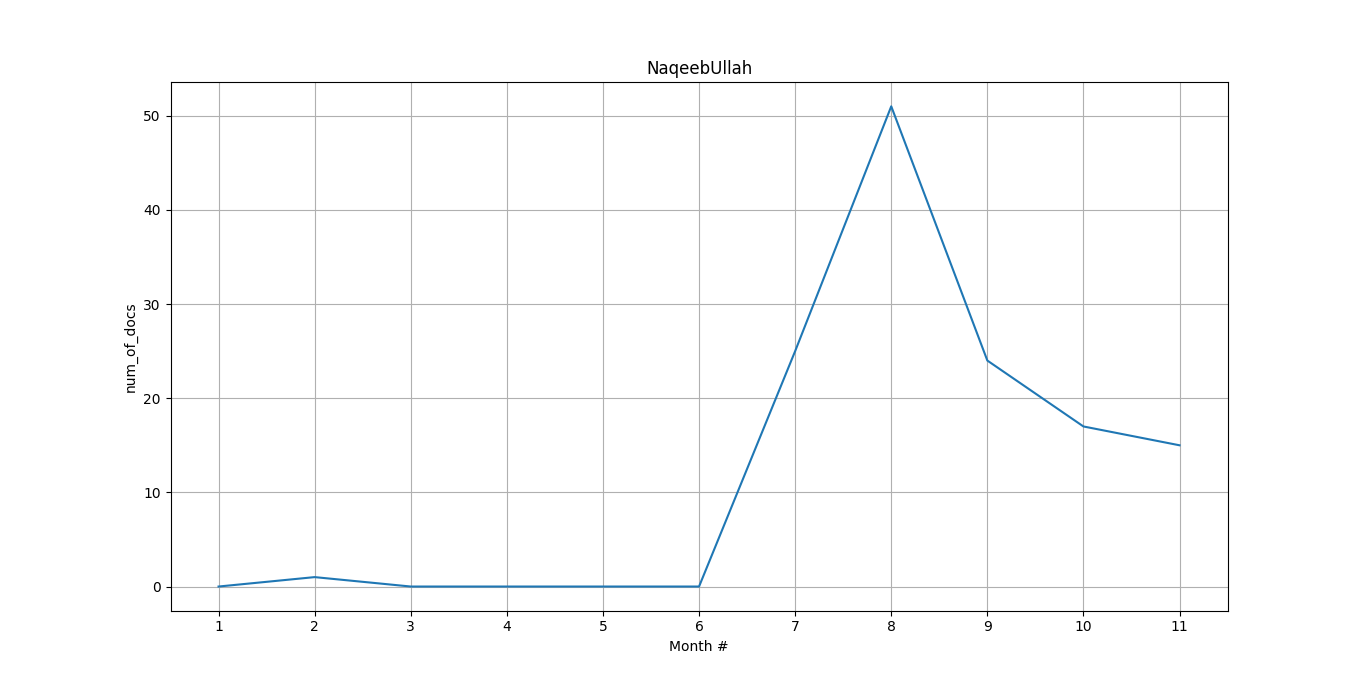


Figure 2: NaqeebUllah Masood murder case (Keywords: 'NaqeebUllah' or 'Naqeebullah' or 'Naqeeb Ullah')

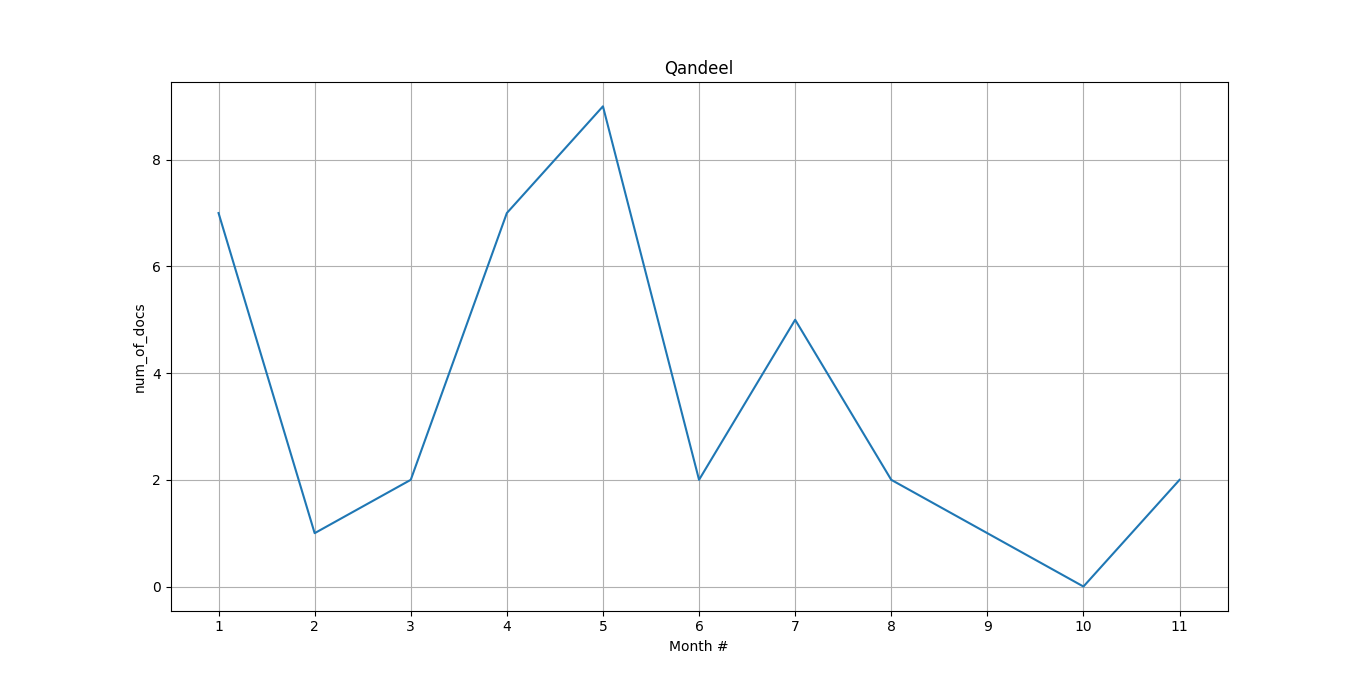
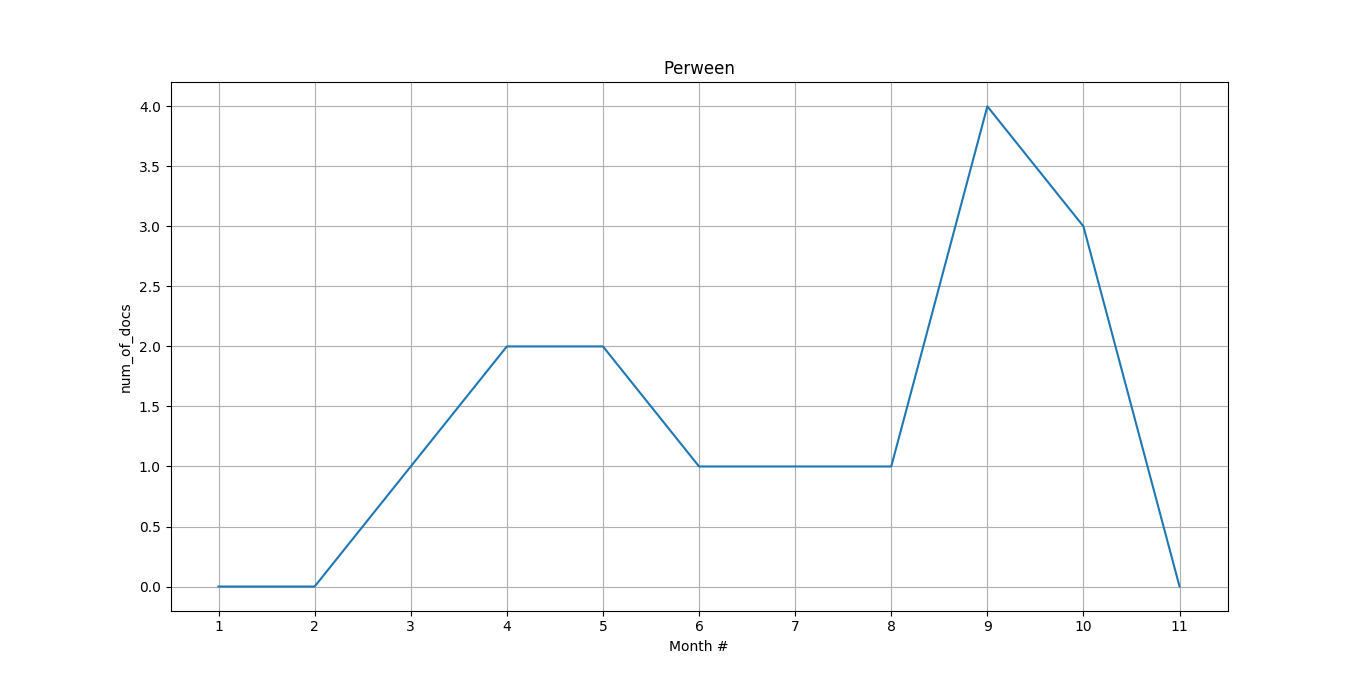


Figure 3: Perween Rahman murder case (Keywords: 'Perween Rahman' or 'Perween Rehman')

Figure 4: Qandeel Baloch murder case (Keywords: ‘Qandeel Baloch’)

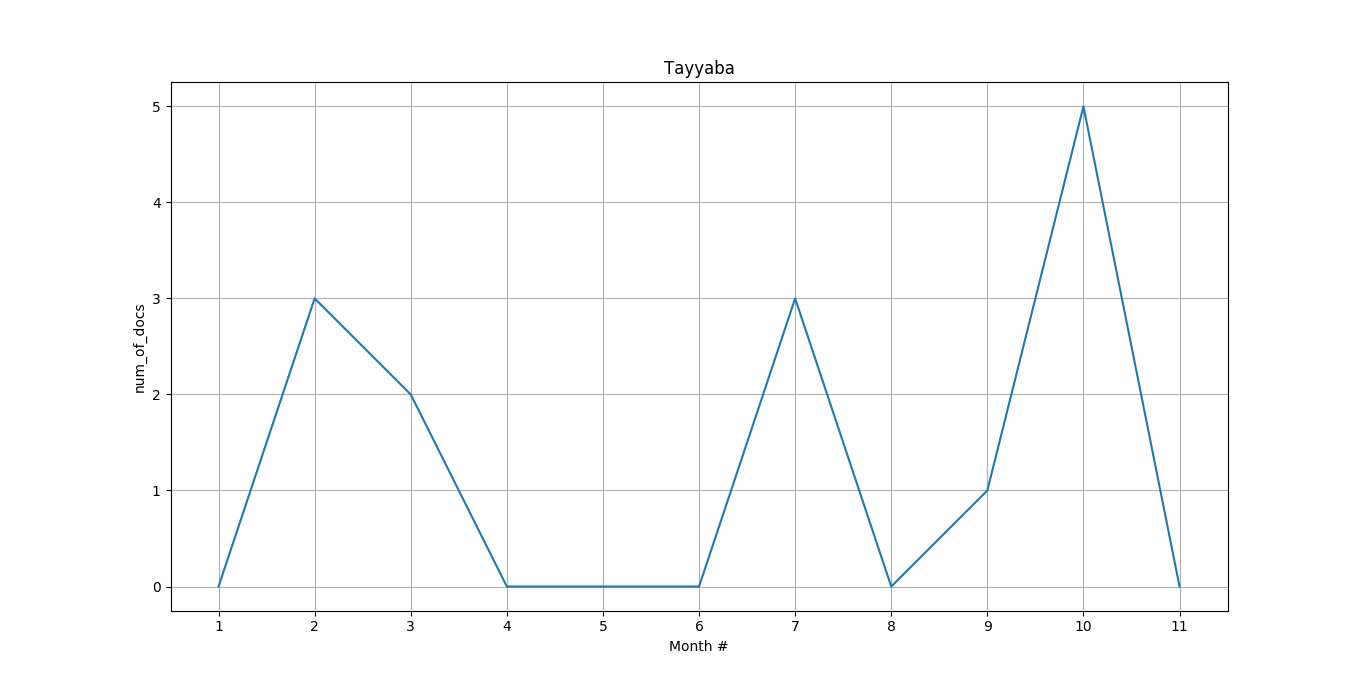


Figure 5: Tayyaba torture case (Keywords: 'Tayyaba')

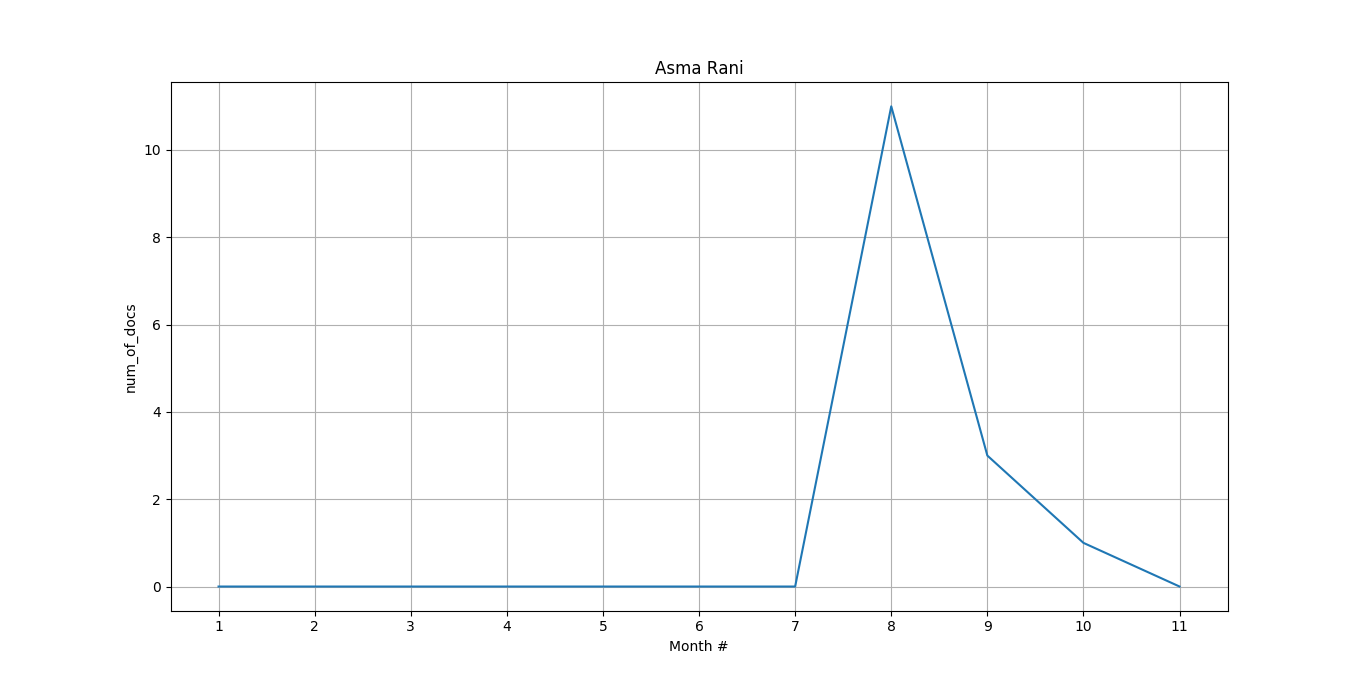


Figure 6: Asma Rani murder case (Keywords: 'Asma Rani')

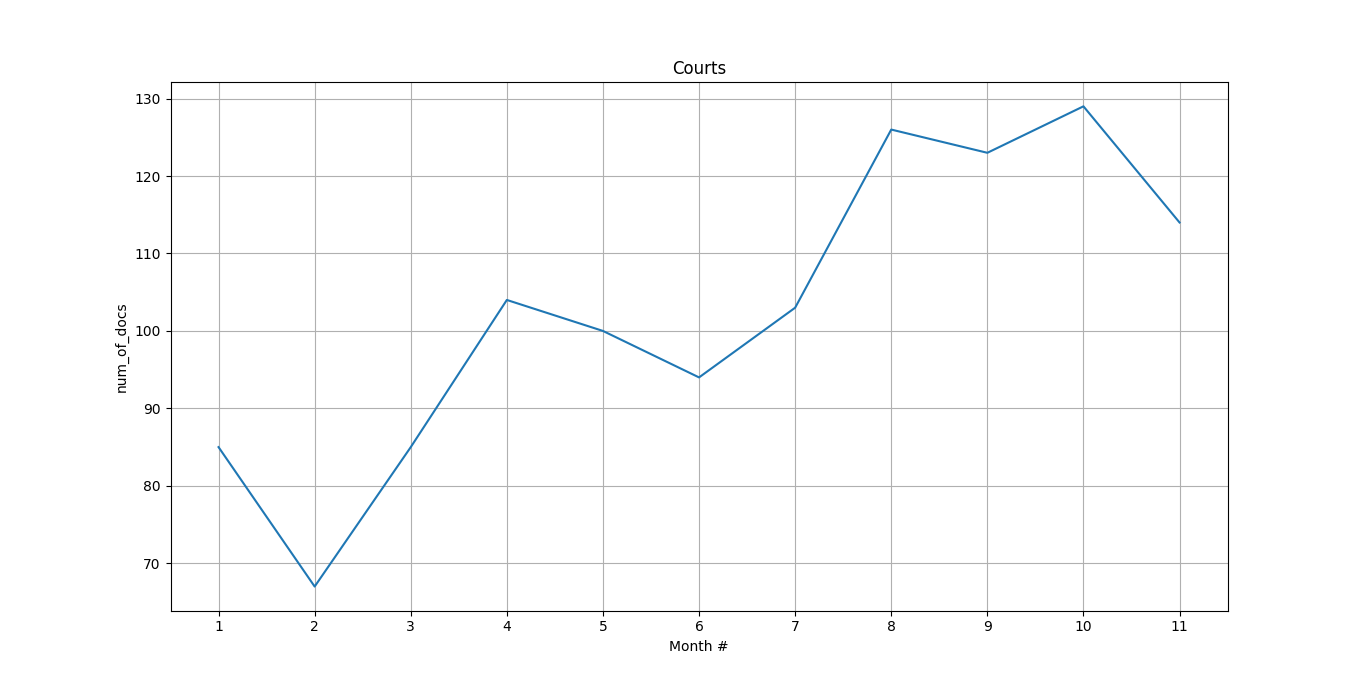
**Comments:**

There are a few things to note here. Firstly, the a number of cases, particularly the Mashal Khan case, Perween Rahman case, and also the Tayyaba and Qandeel Baloch cases began to gain significantly extra attention around the time of the Zainab case despite the fact that they had been going on for quite some time before as well. As a hypothesis, it can be said that there could be a certain bias towards these cases, either from the media or the justice system.

For example, it is quite clear that corruption cases against politicians which were being delayed for years, suddenly started to be looked into after the Panama Papers case. The data’s proof of this can be seen in the following two graphs:

Figure 7: Mentions of the courts including articles related to the Zainab case (Keywords: ‘SHC’ or ‘LHC’ or ‘SC’ or ‘court’ or ‘Court’ in text or heading)

Figure 7: Mentions of the courts including articles related to the Zainab case (Keywords: ‘SHC’ or ‘LHC’ or ‘SC’ or ‘court’ or ‘Court’ in text or heading)



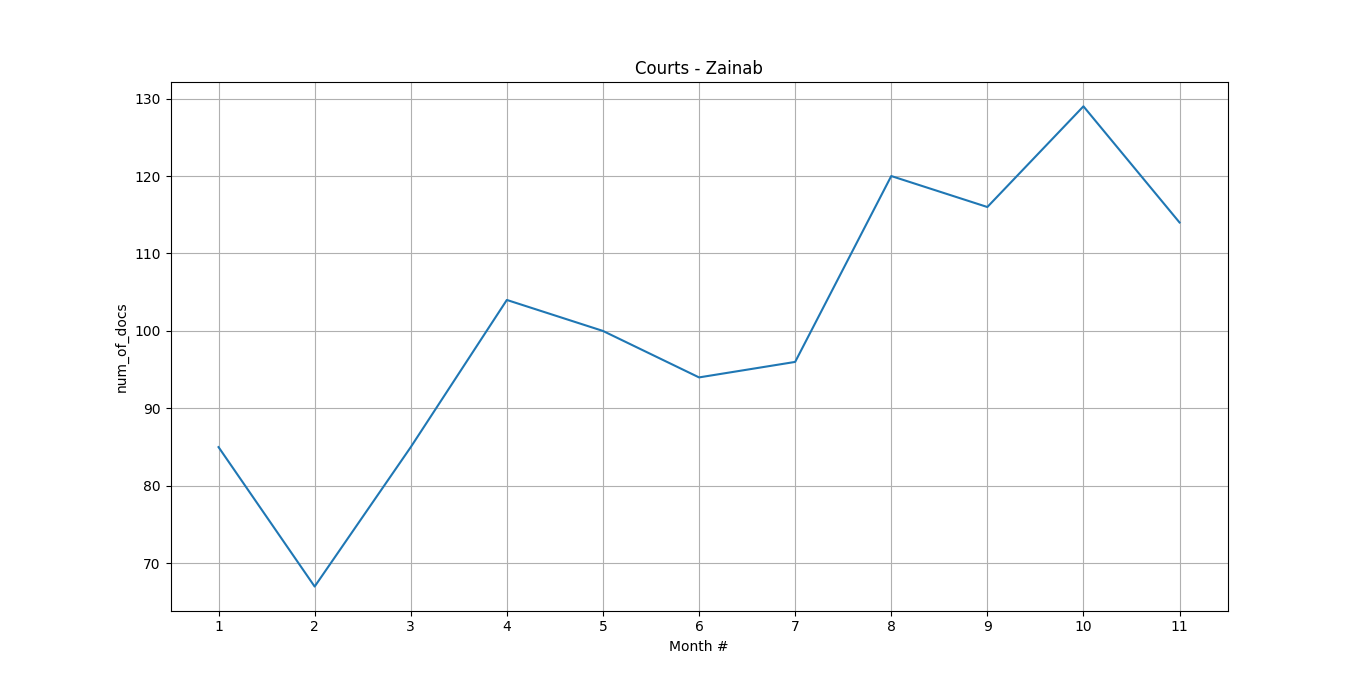


Figure 8: Mentions of the courts including articles related to the Zainab case (Keywords: ‘SHC’ or ‘LHC’ or ‘SC’ or ‘court’ or ‘Court’ in text or heading and NO Zainab in heading)

**This clearly shows a rise in work by the courts. But then we look at the same mentions for police as well:**

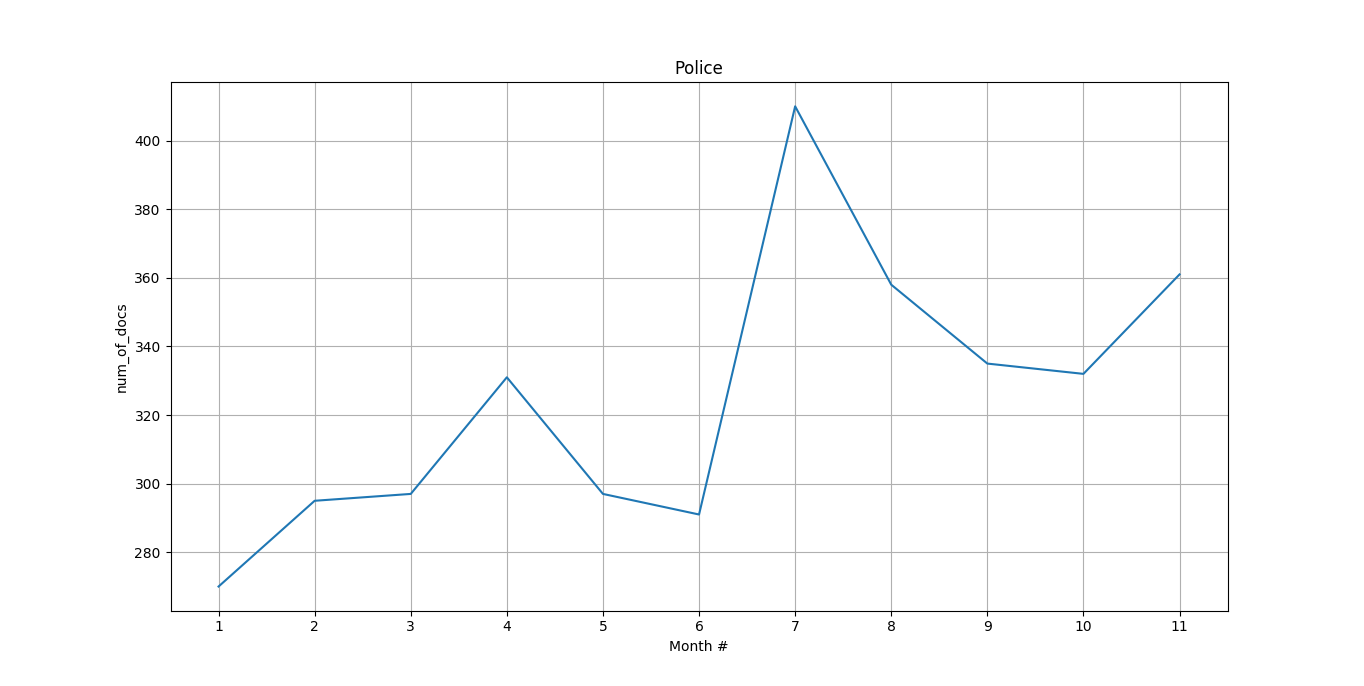


Figure 9: Mentions of the police including articles related to the Zainab case (Keywords: 'Police’ or ‘police’ in text or heading)

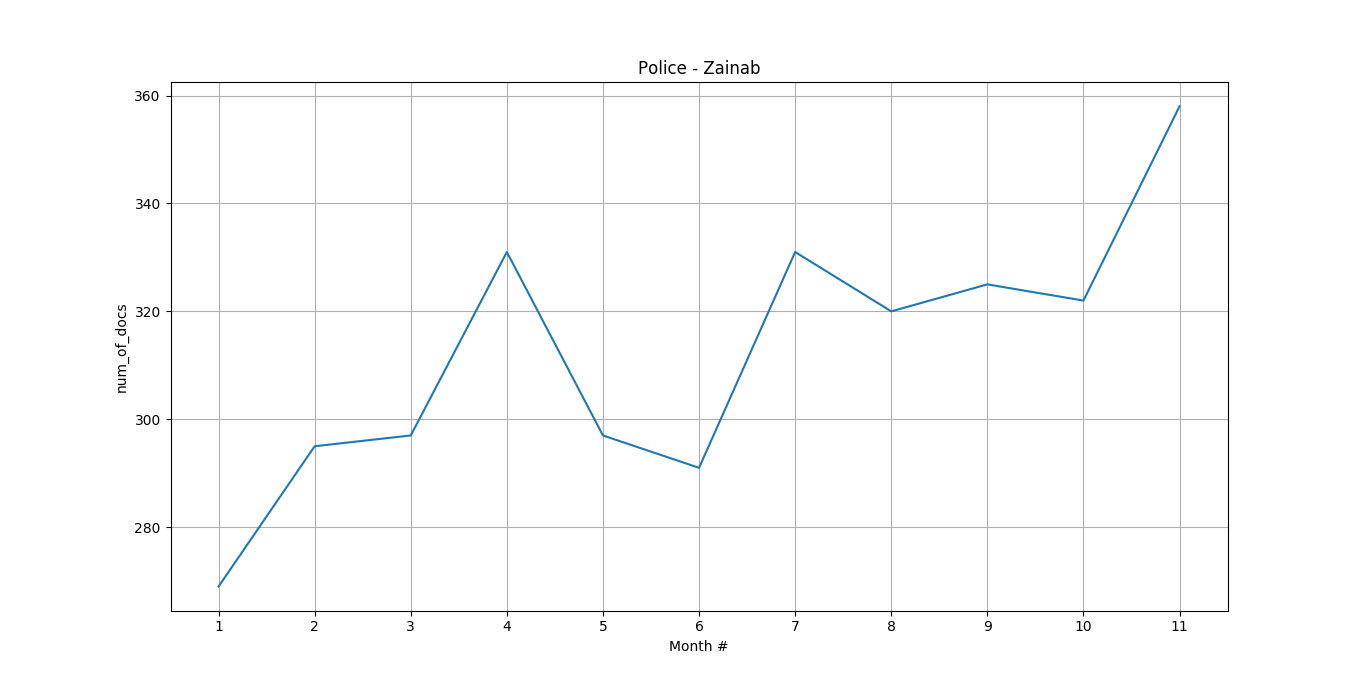


Figure 10: Mentions of the police including articles related to the Zainab case (Keywords: 'Police’ or ‘police’ in text or heading and NO 'Zainab' in heading)

**Comments:**

From the data, we can hypothesize that the police are somewhat influenced by media attention. This in turn, could lead to the courts being more active, because it is only when people are more vocal about reporting crimes to the police, followed by investigations by the police that are hyped up by the media, that cases can be properly looked into by the court.

Also, note how round about the same time as the Zainab case, two other major cases, including the Naqeeb Ullah case and Asma Rani cases regained momentum as well. The cases possibly gained more attention than their actual time of occurrence, due to the reasons mentioned above, and can also help explain the sudden hype among the Pakistani population