1. **Database Collection**
   1. **Article collection from GoogleNews API**
      * **Requirements -** pygooglenews (GoogleNews), pandas
      * Created an object of **GoogleNews** class. It was used to fetch news articles wrt given search titles.
      * Created two lists containing search titles and indian sources.
      * Then the articles were fetched by a search function, which scraped data of all the articles available on google news, for each given search title. The attributes stored in the dataframe were : Title, Published date, Summary, Source.
      * The data was reduced by only keeping the details of Indian articles whose publishers were listed previously. Also, sources with less than 30 articles had their articles removed as well.
      * The remaining articles (dataframe) were saved in articles (3).csv .
   2. **Collecting article content, and changing headlines**
      * **Requirements -** re, requests, bs4 (BeautifulSoup), Selenium, webdriver (headless browsing), Proxy service (Proxy Rotation), json, time, pandas, random, concurrent (for Threading or, **Parallel Processing**), os
      * Initialize Max\_Threads (**concurrent connections**), Batch size (processing requests in batches of 100), user agents list (for changing headers), proxy list
      * Load articles (3).csv into dataframe.
      * Extract google news rss links from the summary section, put them into news\_url column and drop the summary section.
      * Then the urls were then decoded with the help of parallel processing, to return the final article link after redirection. These were stored in final\_url column.
      * Articles with no final\_url were dropped.
      * Title was refined.
      * A Content column was added, with all empty cells.
      * Stored into df.csv.
      * A **pipeline** was created which took 100 article details at a time (batches), then passed them in packets of 10 for content, title extraction. After a batch has finished processing the last processed article’s index is stored, and the new batch is used. This is done, to save the fetched content after fixed intervals, to reduce data loss due to external factors.
      * **Proxy rotation**, and headers were used to avoid **403 Error**. Along with this **headless browsing** was used, so that the Chrome tabs are opened only in the background, saving system resources.
      * **Beautiful Soup** was used to parse the html code, helping in finding the title tag, script tag containing the content in most of the pages.
      * **Selenium web driver** was used to scrape content stored in JS code (within script tag).
      * A delay of 3s was added after each get operation, so that the **JS** part loads properly.
      * Titles were modified to remove publisher names using regular expressions.
      * It was then saved into test\_df.csv
      * Now, both df.csv and test\_df.csv were loaded, performing multiple preprocessing operations in the title section of test\_df, and in the end comparing both the titles, to save the best one along with the rest of the data into df\_1.csv
      * Lastly, articles with no content extracted were dropped thus saving the remaining to final\_df.csv.

**Dataset Size :**

* **Initial article collection using google news =** 12613
* **Articles after keeping only selected Indian sources =** 3886
* **After adding final\_url in df.csv =** 2702
* **After saving content into final\_df.csv =** 1576
* **Challenges Faced :** 
  + Significant bottleneck of proxy availability. With only 100 premium, reliable proxies having a limit of 10 concurrent connections, the content extraction time was much higher.
  + Some websites took too long to respond, thus returning empty title and content cells. Also, some websites have content within other <p> tags, that are located in the body of the web pages, nested within multiple <div> tags.Thus, their content extraction was not possible as it requires specific code for each publisher.