IRE Major Project Final Report

Team 19:

1. Pratyanshu Pandey (2019101025)
2. Pavan Baswani (2021701035)
3. Prince Singh Tomar (2019101021)

Video Folder:

The folder contains a subfolder with a video for each major module of the project called “Module Wise Videos” and a finalvideo.mp4 - which is just a concatenation of all the videos.
Link:
https://drive.google.com/drive/folders/1UvdJwSpws59-7u0dzs7D4J8NB-0fUgpP?usp=sharing

INDEX

Problem Statement:

Collecting domain specific data given a seed set of structured and unstructured sources

Given a domain, the task is to build a structured, clean and high quality dataset for the domain. This will involve searching for structured and unstructured sources for the domain, scraping the web pages to get the raw data, extracting tabular information from the raw data, processing and formatting this tabular information to give a structured, clean and high quality json output.

In specific, the domain we will be working on is “Companies Registered in Telangana”.

Related Work:

Although scraping data from online websites to create your own dataset is not something new, the specific problem of collecting data about “Companies Registered in Telangana” has only been tackled once before and the resultant dataset is one of the seed datasets we will be using.
We will also compare our final results with this old seed dataset.

General Approach:

1. For every source that we decided to work with, the general approach is to look at the source code of the web page and what data can be accessed in what manner.
2. Some data is freely available while others are behind a cache or a login mechanism. Some data is in tabular form while others are in text inside html tags.
3. We used selenium and BeautifulSoup to overcome these walls to get the data.
4. We write the code making sure that it can handle exceptions well and generates an output log for manual analysis of errors. Since the number of companies is high the code is run in batches.
5. The log file is then checked manually after execution to find any errors. These errors have been few in number and have been handled manually.
6. Once there are no more errors, the data is fully scraped. Now the data is combined, cleaned up and pre-processed - duplicates and extra whitespaces are removed, dates are formatted etc.
7. Then for each source the data is reformatted to suit the global paradigm.
8. Data from all the sources are then combined in a fixed priority to resolve conflicts and then analysed.

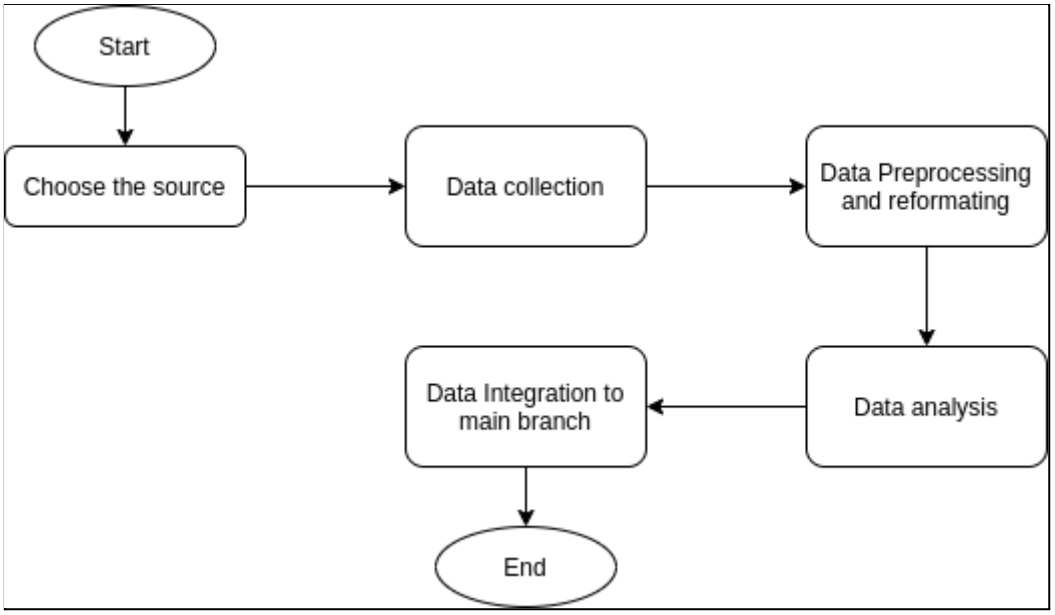

Figure-1: Model Architecture

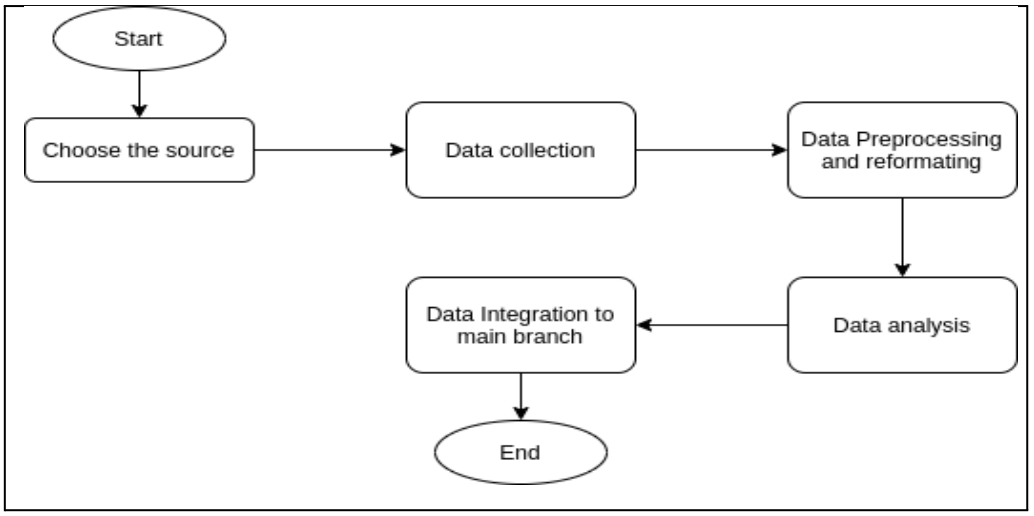

Figure-2: Flowchart of general approach

Data Structure:

The data we aim to collect is tabular in nature with **140,280** rows, each row containing data for a company.
Each company has **33 major attributes** that are represented by columns. Some of these attributes are subdivided into sub-attributes each with its own column. In this manner the total number of **columns exceeds 60**.
A **cell** in this table generally holds a single value but it can also be **multivalued, textual or empty.**

The following table lists all the column names, sub columns if they exist, sources from which data for these columns are extracted and the current status of our work for that column.

The column **Additional** holds data from wikipedia and holds possible websites in key value format like a dictionary. This is described in detail in further sections.

S.No.	Column Name	Sub Columns	Sources
1	CIN/LLPIN	-	zaubacorp.com mca.gov.in seed data
2	Company Name		zaubacorp.com mca.gov.in seed data
3	Date of Incorporation		zaubacorp.com mca.gov.in seed data
4	Registration Number		zaubacorp.com mca.gov.in
5	Age of Company		zaubacorp.com
6	Company Status		zaubacorp.com mca.gov.in seed data
7	Class of Company		zaubacorp.com mca.gov.in seed data
8	Company Category		zaubacorp.com mca.gov.in seed data
9	Company Sub Category		zaubacorp.com mca.gov.in seed data
10	Listing Status		zaubacorp.com mca.gov.in
11	Number of Members		zaubacorp.com mca.gov.in
12	Previous Names		zaubacorp.com
13	Previous CINs		zaubacorp.com
14	Number of Employees		zaubacorp.com
15	Authorised Capital		zaubacorp.com mca.gov.in seed data

16	Paid Up Capital		zaubacorp.com mca.gov.in seed data
17	RoC		zaubacorp.com mca.gov.in seed data
18	Activity		zaubacorp.com mca.gov.in
19	Address		zaubacorp.com mca.gov.in seed data
20	Email		zaubacorp.com mca.gov.in
21	Website		zaubacorp.in
22	Date of Last Annual General Meeting		zaubacorp.in
23	Date of Latest Balance Sheet		zaubacorp.in
24	Important People	DIN	zaubacorp.com mca.gov.in
		Name	
		Appointment Date	
		Designation	
25	Prosecutions	Defaulting Entities	zaubacorp.com
		Court Name	
		Prosecution Section	
		Date of Order	
		Status	
26	Charges	Charge ID	zaubacorp.com mca.gov.in
		Creation Date	
		Modification Date	
		Closure Date	
		Assets under Charge	
		Amount	

		Charge Holder	
27	Trademarks	Name	zaubacorp.com
		Class	
		Application Date	
		Status	
		Goods and Services Description	
		Applicant Address	
28	NSE	capital	nseindia.com
		symbol	
29	BSE	Security Code	bseindia.com
		Security Id	
		Security Name	
		Status	
		Group	
		Face Value	
		ISIN No	
		Industry	
		Instrument	
30	Establishment Details	Name	zaubacorp.com
		City	
		PinCode	
		Address	
31	Total Obligation of Contribution		zaubacorp.com mca.gov.in
32	Number Of Partners		zaubacorp.com mca.gov.in
33	Additional		wikipedia.org Google Search

Data Collection:

zaubacorp.com :

Libraries Used : Selenium, bs4, pandas, requests, json, time, re, os, numpy, collections, openpyxl

Creating exhaustive list of Companies

1. From the list of all companies registered in Telangana we extracted the company name, CIN, status and link to the full page for the company.
2. We performed uniqueness and integrity checks for the list of companies and removed the duplicates.
3. This list now had **140,280** which is close to double of the seed data available to us.
4. We made sure that all the companies present in the seed data were present in the new list as well. This list is now the exhaustive list of all companies.

Scrape Company data from zaubacorp.com

1. For accessing the full company page on zaubacorp we used "*selenium*". It helped in the automation of the process and made it easier. We first logged into the site by sending username and password to the required fields of the website.
2. The problem arose when we had to solve the captcha for that we accessed the region where the mathematical equations were presented and solved it using the eval function in python.
3. After logging in we used the driver.get() function to access the required webpage. Since the data on zauba corp is organised in tabular fashion, which helped in automating the process.
4. We extracted the required tables and did some processing on it. We cleaned the data to remove unnecessary noise, For example, to remove advertisements.
5. After all cleaning the data was stored in a dictionary and later the dictionary was exported in json format.

Scrap Trademarks Data

1. From the list of companies we generated the url to get the trademarks registered by the companies.
2. Since login was not necessary to access this part of the website, we simply used requests.get() to get the complete web page.
3. The data was not in a table but rather within tags so we used the BeautifulSoup library to extract it.
4. Integrity and uniqueness checks were performed on the data.
5. This data was stored in a dictionary and exported in json format.

mca.gov.in :

Libraries Used : Selenium, bs4, pandas, json, time, os, numpy, PIL, pytesseract

1. The website mca.gov.in requires the login with cin and captcha present in the image to access the company data in tabular form.
2. Initially, the screenshot of the whole page is saved and cropped the image where the captcha is located using the python package **PIL**.
3. With the help of the "*pytesseract*" package, the text present in the cropped image is extracted.
4. Using the selenium web driver, the fields are located with their ids and sent the keys (cin and captcha text) to submit the form.
5. Once the form is submitted, the form will be evaluated with the login details and redirected to the company details page. The exceptions (invalid cin or captcha) are handled to run the code without any interrupt/crash.
6. Finally, all the company details are downloaded in excel format with the selenium driver with button click.

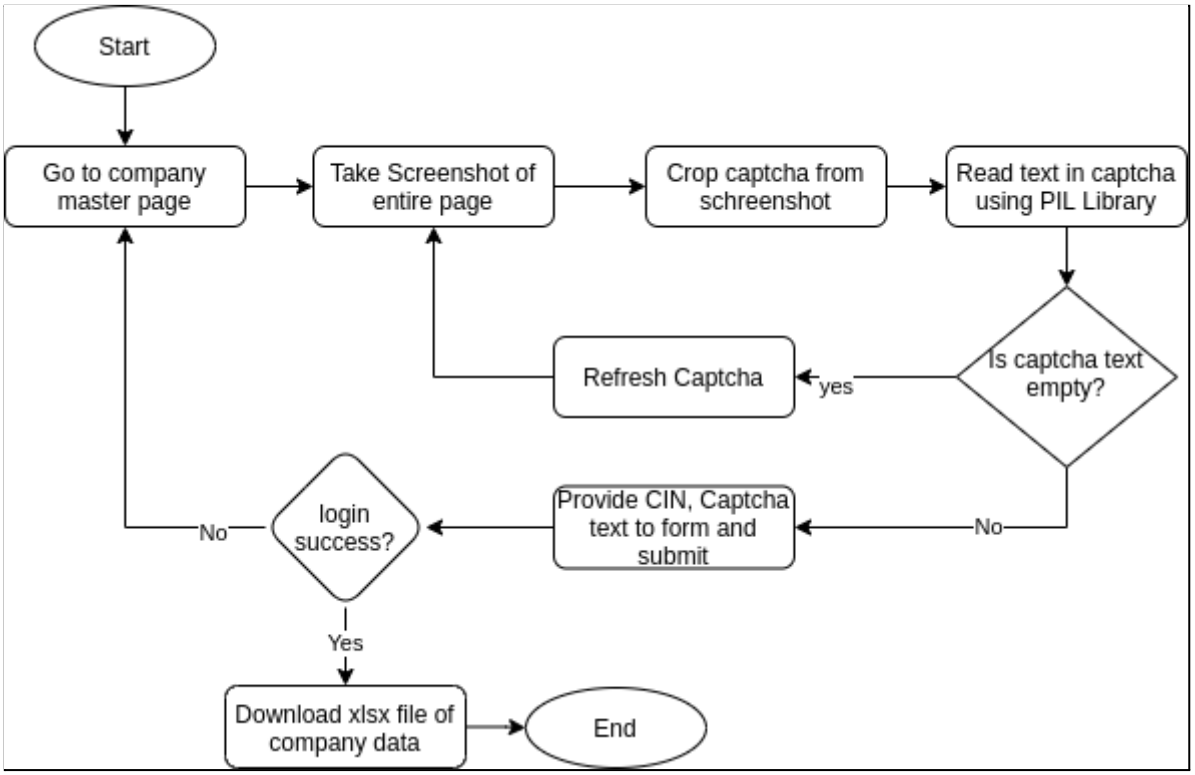

Figure-3: Flowchart of mca.gov.in crawling

wikipedia.org :

Libraries Used : fuzzywuzzy, wikipedia, bs4, pandas, json, time, os, numpy

1. For extracting data from wikipedia we used wikipedia api to search all wikipedia pages for their titles, the api returned a list of titles most related to

the searched query (company name). We then used a name matching algorithm to detect the returned title most similar to the query.
2. After Getting the title of the wikipedia page, we used requests in python to get the wikipedia page.Infobox was extracted from the requested wikipedia page.
3. Since the name matching algorithm was not fully accurate we ran a double check to get all the wikipedia docs which are linked to the same wikipedia page. These pages were attached to the company whom they were most similar to.
4. The data extracted contained info like Area served, products, type, subsidiaries etc.

Google search :

Libraries Used : bs4, json, time, os, requests, urllib3

1. A lot of smaller companies have their own websites but details about the website are only easily available for bigger companies in Wikipedia or zaubacorm.com.
2. A simple google search with the company name as a query however reveals the website and other basic information compiled from various sources.
3. Initially, the google search link will be created with query terms, number of links required, language etc. (refer here to check the format and creation of google search url).
4. Then using the requests.get() method, the page source of google search url will be given as input and extract the top links and google table (right most side) if present.
5. Then using the requests.get() method, we will perform a google search for each company and extract data from the Google's Knowledge panel created by google as well as look at the top 10 links and their descriptions' to determine if any of them is a link to the official company website (by extracting the keywords/metadata of the websites present in top 10).
6. Most of the company names contain the special characters and symbols, which itself creates an issue if the terms in the company name are separated by space and concatenated by '+' symbol.
7. The approach is to encode the query terms using urllib.parse.quote() method, which will automatically encode all the special characters and symbols to google searchable query format.
8. Also, continuously sending requests to google.com, the HTTPError will occur to prevent the bulk requests. To avoid this issue, we need to wait till the 'retry-after' time period (we can find this in headers of the exception occured).
9. Once we have the page source of the search query, using the beautifulsoup the top results with their description and Google's knowledge panel (if present) extracted.

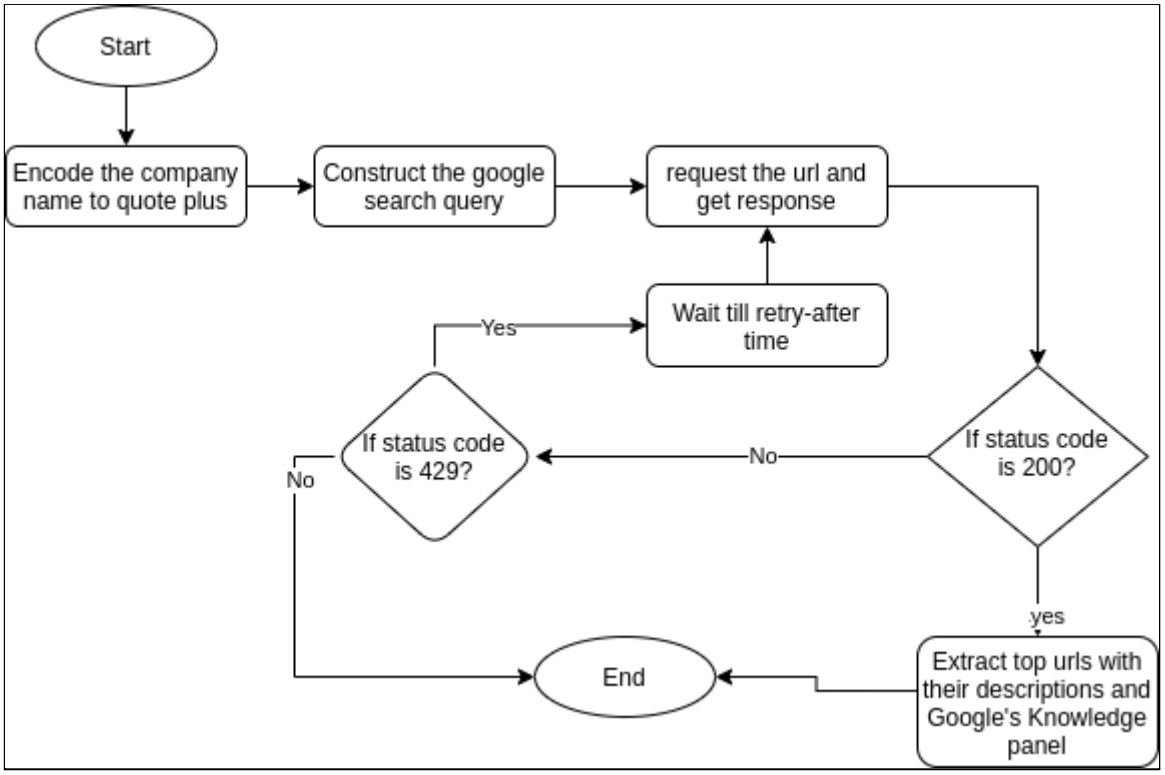

Figure-4: Flowchart of google crawler

nseindia.com/bseindia.com:

1. The list of all companies on NSE or BSE is available in excel format on their website.
2. The excel sheet contains the company name as the primary key.
3. We have downloaded excel and during the data processing phase, we will extract relevant information for the relevant company from the excel sheets using the match of company names.

Data Processing and Cleanup:

First data from each source was processed and cleaned up at the individual source level and then data from all the sources were combined.

First we describe processing and cleanup for each source:

zaubacorp.com :

Libraries Used : json, nltk, os, urllib

Creating exhaustive list of Companies

1. The collected list of companies had 301 duplicates that were identified using the CIN/LLPIN of the companies which is unique to each company. The duplicates were then removed.
2. This list was then matched with the seed dataset and the differing CIN/LLPIN were noted down. This is analysed in the next section.
3. The final list of companies of length **140,280** acts as the base for collecting all data.

Company data from zaubacorp.com

1. The collected dataset contains a list of previous CINs for every company. This list was matched with the previously stored mismatch from seed dataset(mentioned in point 2 of previous section) to ensure that the company list was exhaustive.
2. Some columns/attributes of companies (“Establishment Details” and “Prosecutions”) could have multiple values and had duplicates. Exact duplicates in “Prosecutions” were removed.
3. In “Establishment Details”, the **Levenshtein distance** or edit distance between 2 addresses and establishment names was used as a measure of duplicity. Strings with distance <= 2 were considered duplicates and were removed.
4. A list of all columns present in the dataset was prepared and for every company, if the company did not have the column, the column was added with a null value. This eases further processing.
5. The dates present in the dataset were present in multiple formats. Thus all the formats were normalized to follow the DD-MM-YYYY pattern.
6. The whitespaces were trimmed and special unicode characters for ₹ were replaced by “” string.
7. Columns were then renamed to the global column names. Some columns that had less useful or redundant data were removed.

Trademarks Data

1. The collected data had a lot of duplicates. To remove duplicates a group attribute for each trademark was identified as the key and then duplicates of the key were removed.
2. The urls collected were verified to be valid.
3. The dates were also present in YYYY-MM-DD format which was reformatted to the global DD-MM-YYYY format.
4. The list-like data structure of the data was reformatted to a dictionary to better suit global standards.

mca.gov.in :

Libraries Used : pandas, json, os, urllib, collections, sys, math

1. The data collected was in excel format for each company. This data was extracted using the pandas library to the dictionary format.
2. A list of all columns present in the dataset was prepared and for every company, if the company did not have the column, the column was added with a null value. All NaN values were also replaced with “”. This eases further processing.
3. The dates were also present in DD/MM/YYYY format which was reformatted to the global DD-MM-YYYY format.
4. The columns were renamed according to the global names and the list-like data for “charges” and “directors” were converted to dictionary structure for further processing.

wikipedia.org :

Libraries Used : json

1. Some fields were not required as the data was already obtained from other trusted sources. Hence those fields were dropped.
2. Converted products, subsidiaries, services etc. in the list format from string format. Some Companies' website fields had the name of the company, hence those fields were dropped.
3. Some of the wikipedia pages got attached to more than one company, these instances were given to the company with the highest matching name percentage and other related companies were dropped.
4. Then the data was cleaned up removing white spaces and unicode characters.
5. A link check on websites was performed and websites that were not in proper format were removed.

Google search :

Libraries Used : json, nltk, os, urllib

1. Once the data is extracted, it contains a query and the resultant list of url and description. We use rule based methods to find top candidates for website domains.
2. First the domains are extracted using the urllib.parse library. Now if the description says official website, or a big enough part of the company name is present in the domain or if the domain is an abbreviation of the company name, the domain is considered to be a candidate website.
3. Since fully ensuring that the website is correct is difficult, what we do is instead of using this as the main data, we store this in “Additional” as

"Possible Website" in absence of the main website. The data is formatted to suit this.

nseindia.com:

Libraries Used : json, nltk, openpyxl, collections

1. The downloaded data is an excel sheet which is parsed using openpyxl into a dictionary-like structure.
2. The data has company names and no CIN/LLPIN. Since company names can be written in several ways using company name as a key to link the datasets will not yield good results.
3. Thus the **Levenshtein distance** or edit distance between 2 company names(lowercased with some suitable substitutions) were used as a proxy for similarity. Thus for every company in our global exhaustive list candidates were selected based on cutoff of edit distance.
4. This code was written in C++ as python was very slow in processing this.
5. The C++ output was then parsed and further filtering was done to get the top candidate data match for each company. There were also no candidates for a lot of companies.
6. The final dataset was then created with these top candidate matches after trimming the white spaces from strings.

bseindia.com:

Libraries Used : json, collections

1. The csv file was first parsed into a list-like structure.
2. The observations from nseindia.com data showed that the difference in company names occur in fixed patterns like (&/and, ltd./limited etc.).
3. Thus instead of running the edit distance code, the company names were normalized to a pattern (removed all spaces and with appropriate substitutions) and now this name was used as key to match the 2 datasets(global list and bse data).
4. The strings in the data were trimmed and the list-like structure was reformatted to a list of dictionary-like structures to better suit the global standard.

Combination:

Libraries Used : json

For combining the data across all the sources:

1. We created a new dictionary and filled it in a fixed priority order of:
    a. Nse
    b. Bse
    c. Trademarks
    d. Zaubacorp
    e. Mca
    f. Wikipedia
    g. Websites/Google search
2. Conflicts were resolved as follows:
    a. Charges and Directors: These columns were picked from zaubacorp and the corresponding values from mcs.gov.in were ignored. This is because zaubacorp has more detailed data.
    b. For all the other common attributes between zaubacorp and mca.gov.in priority was given to value from mca.gov.in if the value was not null, else the value from zaubacorp was maintained.
    c. Website: If website data is present in Wikipedia then it is given priority otherwise, the data from zaubacorp is used. If no data is available on websites, then the possible candidates of websites are included in the Additional column.
    d. Other non-common attributes are simply added.

Evaluation and Analysis:

Sparsity Analysis:

The following is the density of the data collected for each major column. multi valued columns are counted only once.
We are outputting density i.e. the higher the value the higher is the number of entries and lower the sparsity.

Previous CINs : 50.92600513259196
Charges : 13.689763330481894
Company Name : 100.0
Number of Employees : 100.0
Authorised Capital : 91.69375534644996
Number Of Partners : 8.306244653550042
Establishments : 6.492728828058169
Registration Number : 91.69375534644996
Date of Last Annual General Meeting : 45.397775876817796
Email : 84.16595380667236
Age of Company : 100.0
Listing Status : 81.41716566866268
Trademarks : 3.2456515540347874

Class of Company : 91.69161676646706
Date of Incorporation : 99.9992871400057
Important People : 82.84858853721128
Prosecutions : 0.01211861990305104
Website : 2.2476475620188197
CIN/LLPIN : 100.0
Date of Latest Balance Sheet : 45.91958939264328
Company Status : 100.0
Address : 99.9992871400057
Company Sub Category : 91.69161676646706
Number of Members : 81.50484744796123
BSE : 0.32435129740518964
NSE : 0.07556315939549473
Company Category : 91.69161676646706
Previous Names : 8.41103507271172
Paid up capital : 91.69375534644996
Activity : 99.99287140005703
Total Obligation of Contribution : 8.306244653550042
RoC : 100.0
Additional : 0.02209865982321072
Average: 58.9761233835

The total number of entries is **2780023** overall **density** of the dataset is **58.98%**.

Reasons:

While some of the columns present are sparse or not 100% because of the lack of data from online sources(most of the columns), there are also some columns like Trademarks, NSE, BSE that are inherently sparse as not many companies have them and here sparsity reflects the reality of this data.

Most and Least Occurring Columns:

The most occurring attributes are :

1. Company Name
2. Number of Employees
3. Age of Company
4. CIN/LLPIN
5. Company Status
6. RoC
7. Date of Incorporation
8. Address
9. Activity

This also stands to reason as we expect this data to be present for all companies.

The least occurring attributes are :

1. Prosecutions
2. Additional
3. NSE
4. BSE
5. Website
6. Trademarks
7. Establishments
8. Number Of Partners
9. Total Obligation of Contribution
10. Previous Names
11. Charges

This also stands to reason as we expect some of this data to be inherently for all companies, while the data available online for others was very sparse.

Duplicate Removals:

The number of duplicates removed varied from 301 in main base links, 150 in trademarks and 120 in Establishments to 2 or so in prosecutions.
But wherever possible we checked for duplicates and removed them.

Column Removals:

Some not so useful columns were removed from the mca.gov.in, wikipedia as well as zaubacorp data. These columns were either too small sparsity or too less important to be included in the dataset.
There were also columns with different names representing the same global column. Such columns were merged to the global column.
On a total basis we removed **12 columns.**

Evaluation:

We have improved upon the previous work by close to doubling the number of companies and close to tripling the number of columns with a very decent . This shows a massive improvement over the previous seed dataset. In terms of sheer size our dataset is more than 330 MBs of data.
In terms of quality we have made sure that the table on an overall basis is dense and we have maintained the quality of data by removing duplicates and irrelevant columns and ensuring consistency in data.

Code and Data Link:

The code is present in the src folder and the extracted data is present in the data folder. Overall structure of the code is also described in the README file on the repository.
Since the data folder has also become too big, and the repository cannot hold it, here is the Google Drive link for the data folder. The **integrated.json** file in the data folder is the final dataset. A link to it is also provided.

Data Folder:
https://drive.google.com/drive/folders/122s1rhnApXVexOag-IvDUW1pEmWbukFE?usp=sharing
Github Link: https://github.com/IRE-Project/Data-Collector
Link to Final Dataset Folder:
https://drive.google.com/drive/folders/14RuhmEesHnLbOMxDmH0KzSa0YDJuVK95?usp=sharing

Difference as compared to Scope Document:

1. Instead of collecting all sorts of raw data and then analyzing the columns we switched these steps and analyzed the sources to get a list of columns. We then started with collection of data.
2. Instead of keeping data collection and processing as 2 completely independent tasks we have introduced some amount of pre-processing during the data collection phase itself. These include uniqueness check, integrity check etc.
3. Instead of just settling with tabular data we are also extracting some data that is present within html tags and not tables.
4. For the domain that we are working with we no longer feel the necessity of using named entity recognition to extract data from unstructured sources, so we will not be using this methodology.
5. The Google search results were not as good as we expected and properly determining the domain name was a difficult task.
6. The amount of data on wikipedia was lower than we imagined.

References:

(not all of the below proved useful)

- Data collection & harvesting :
    1. Building domain specific dataset or corpora from Wikipedia for ML

2. Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation
3. Domain-Specific Corpus Expansion with Focused Webcrawling
4. (PDF) Harvesting Domain Specific Ontologies from Text
5. https://arxiv.org/pdf/1906.11405.pdf
6. Extracting tabular data from PDFs made easy with Camelot.
7. Extraction of Biographical Information from Wikipedia Texts
8. Using Wikipedia for Hierarchical Finer Categorization of Named Entities
9. (PDF) A Journey of Tabular Information from Unstructured to Structured Data World Using a Rule Engine
10. https://www.marcomrobot.com/blog/how-to-find-website-url-from-company-name-in-bulk

- Data integration:
    1. Data Preprocessing in Data Mining -A Hands On Guide

- Dbpedia(not utilized):
    1. (PDF) DBpedia and the live extraction of structured data from Wikipedia
    2. Building domain specific dataset or corpora from Wikipedia for ML
    3. Web Scraping: Introduction, Best Practices & Caveats | by Velotio Technologies | Velotio Perspectives
    4. An introduction to web scraping with Python | by Jonathan Oheix
    5. Domain-Specific Entity Extraction from Noisy, Unstructured Data Using Ontology-Guided Search
    6. Beautiful Soup: Build a Web Scraper With Python – Real Python
    7. How to build a simple web crawler | by Low Wei Hong

END