Natural Language Processing for Extracting Rich Disease Data Aligned To Satellite Meteorological Data

1st Mahi Pasarkar
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
pasarkarmahi@gmail.com

2nd Junseob Kim
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
kimj43@rpi.edu

3rd Eoin O'Gara
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
ogarae3@rpi.edu

4th Alan Zhang
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
zhanga10@rpi.edu

5th Malik Magdon-Ismail
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
magdon@cs.rpi.edu

6th Thilanka Munasinghe
Rensselaer Polytechnic Institute
Troy, NY, USA
& NASA Goddard Space Flight Center
Greenbelt, MD, USA
munast@rpi.edu\thilanka.munasinghe@nasa.gov

7th Jiaqi Weng
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
wengj3@rpi.edu

8th David Qiu
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
qiud@rpi.edu

9th Ethan Cruz
School of Science,
Rensselaer Polytechnic Institute
Troy, NY, USA
cruze6@rpi.edu

10th Jennifer C. Wei
Goddard Space Flight Center
National Aeronautics and Space Administration
Greenbelt, MD, USA
jennifer.c.wei@nasa.gov

11th Ashan Pathirana
Health Promotion Bureau,
Ministry of Health
Colombo, Sri Lanka
ashanpathirana@gmail.com

***Abstract*—Global climate change is redefining our understanding of how diseases spread. In Sri Lanka, vector-borne diseases such as dengue fever, encephalitis, and leptospirosis historically surged during the monsoon seasons when temperatures were high enough for mosquito eggs to hatch. Unfortunately, due to rising temperatures and more erratic rainfall patterns, mosquito eggs can now hatch year-round and are increasingly unpredictable, leading to an alarmingly increasing number of hospitalizations and deaths. More data is needed to adapt our response to these diseases in an increasingly warmer world. In the contemporary landscape, a wealth of disease information is available, yet accessibility remains limited due to unstructured data formats such as PDFs. Therefore, converting unstructured disease reports into structured formats is necessary for effectively leveraging data. This paper introduces a comprehensive framework for collecting unstructured disease reports and transforming them into analyzable formats. By creating separate models tailored to each data format, we can ensure accuracy compared to general models. These straightforward models enhance accessibility and empower other researchers to use our tools. The returned structured data can then be harnessed for analysis, statistical purposes, and informing evidence-based public health interventions, thus facilitating more informed decision-making in healthcare. We deploy this framework to produce geospatial data for Sri Lanka and Brazil for many different conditions and align these data with satellite environmental data, providing for the first time a structured, aligned powerful dataset for disease modeling.**

***Index Terms*—Natural Language Processing, Disease Data, Satellite Data, Parsing PDF, Open Source Open Science**

I. Introduction

Many countries publish weekly or monthly disease reports to aid public health research. However, this data can be stored in formats that are inconvenient for researchers to use. This can create a bottleneck in research, where researchers spend time extracting data when they could be analyzing the data. While general methods to parse unstructured data exist, they are often too broad and inflexible, while other methods to access the raw data are inconvenient and difficult to manipulate. Additionally, studying diseases requires a breadth of earth observational data such as precipitation and surface temperature, which are gathered by satellites. An average person cannot contact these governments to acquire structured disease data, if it exists, or

access satellite data. To remedy this, we created a system to extract such data and publish it in an open-source open-science framework.

We used data from two countries, Sri Lanka and Brazil. Sri Lanka's data was in an unstructured format: PDF. Brazil's data was structured but was separated by each year and accessible through a web interface, requiring data scraping to create a cohesive dataset. Additionally, we added additional data such as locational data and NASA's Earth Observational Satellite Data, specifically Giovanni, NOAH, and MODIS to both datasets. We worked with a disease modeling team to supply them with data for their model, and they supplied us with earth observational data.

To parse the data we used existing techniques such as web scraping using browser tools such as Selenium or browser console, PDF-to-text extraction, and similarity methods to account for multiple spellings or typographical errors. Additionally, we developed our own system using these techniques that is modular and can deal with a variety of different ways of structuring data. Lastly, in the case of inconsistent or erroneous data, we implemented tools to fix any errors manually.

Using our system, we produced a dataset with weekly data for 12 conditions in Sri Lanka from 2013 to the present and weekly or monthly data for 44 conditions in Brazil from 2007 to the present (depending on the condition). The system is modular and easily adapted for parsing data in different countries or formats. We also have a method to update the dataset with new data as it becomes available for future use. All our data and data creation scripts are available online, linked at the end of the paper.

II. Related work

General methods to process PDF table data usually consist of determining where the table is (using machine learning models, OCR lines, etc.), figuring out the structure (OCR lines, spacing rules, etc.), and then converting it to a machine-readable format. Methods include using an OCR such as tesseract to extract lines with which to find tables [1], [2], using neural networks to determine when tables occur [1], and using positional element data to determine tables based on structure [3], [4], [5].

LLMs such as Chat-GPT or LLaMa also can deliver decent results [6], however in our testing it was too slow, and the result often had incorrect data, which made it impossible to process our dataset. There are a few papers that survey most available methods of general table extraction [7], [8], but these methods don't have perfect success, and upon testing with our data fail to extract content properly. Additionally, due to various possible table sizes and combinations, it is difficult to create a universal algorithm [9]. Since general methods are not perfect yet, we opted for a more tailored approach to processing data from the PDFs.

Work for handling abbreviations, such as using similarity functions like longest common subsequence [10], [11], and work for handling typographical errors exists [12], but they are too general and can not directly accomplish the purpose of this paper. Getting accurate location data such as latitude and longitude from location names along with additional context and the structure of the location name (i.e. city, state, country format) and MST calculations [13] could help the data to be more structured, but it was too costly and did not match with the raw dataset we had.

Although processing PDF table data is well-studied, general methods yield varying degrees of success [7], [8]. Additionally, on our data they had many errors. Instead of aiming to create a general method, we opted to use the consistent formatting of disease report documents to our advantage. By making a tailored model for each country, we were able to get more accurate results in a shorter time than if we had pursued a more general approach.

III. Data scraping

A. Sri Lanka

Data was sourced from Sri Lanka's Weekly Epidemiological Report website [14]. We extracted using a two-step method: getting the URLs of each PDF, and then downloading each URL. URLs of each PDF file were extracted using javascript and the browser console. The link to each file was always in the `btn` class. Thus, we were able to get a list of all the URLs using the command `$('.btn').each(function(){console.log($(this).prop('href'))})`. To exclude certain years, we deleted the year's section in the source code under the Elements tab, which would be in divs of class `accordions`. After copying the list of URLs from the console, we used `wget` to download all the files, `wget -i - <<< "files"`, where "files" was the list of URLs separated by newlines. This command downloads all files to the working directory.

B. Brazil

Brazil's data was on a more complex website, requiring more sophisticated scraping tools. We used Python and Selenium. We created `brazil_scraper.py`, with arguments `brazil_scraper.py <data-url> <outfile-name> <output-folder>`. `<data-url>` is the URL for accessing the data. `<outfile-name>` is the basename for the files (filenames are `<outfile-name> <year>.txt`). `<output-folder>` is the folder for storing the data.

The scraper starts by selecting the line and column values(row and column). For the line, it is optional to set it to either "Município de notificação" (Municipality of notification), or "Município de residência" (Municipality of residence). Both descriptors are slightly different. We used Município de residência. Município de notificação is the municipality where the case was reported, while Município de residência is the municipality where the patient resides. For the column, we selected weekly cases, and if not possible, monthly cases. For different datasets, weekly and monthly cases were marked with slightly different spellings, so we had to check through a large list of possible spellings for weekly or monthly cases. Finally, we select the data format to be columns separated by ";", as

it is easier to work with. Then, we iterate through each year, click ”Mostra” to show the data, and collect the data for each year. This data is then saved to a file marked with the year. The data was saved in a text document with the first line being the year, the second line being the link to the data and the third line onwards being the table.

IV. Parsing

A. How to use

We implemented a modular model for parsing. The parser, `dataParser.py`, takes in the arguments: `dataParser.py <folder> <output.csv> <parsing-model>`. `<folder>` is the directory the data is stored at, `<output.csv>` is the name and directory of the output file, and `<parsing-model>` is the name of the model that the user would like to use. Models are made for different document types.

Running command `dataParser.py Data data.csv ParsingModels.sriLankaParser` would take all the files in the `Data` folder, parse them using `ParsingModels.sriLankaParser`, and output the result in `data.csv`

One can continue to update the datasets with new data as it is released. Data to `<output-file.csv>` will be appended, rather than overwritten. By using our data scraping method and running `dataParser.py` on the new data, specifying the output file to be the current dataset, the data will be appended to the dataset. Additionally, if one wants to make the new data in a new file that is also possible by choosing a new filename for `<output-file.csv>`.

B. Program structure

The parser extracts the data from all the files in `<folder>`. If the file is already encoded as a .txt, it simply takes the text data. If it is a .pdf, it converts the data to a text format. For this, we used two different parsers: pyPDF2 and pymuPDF. pyPDF2 is good at identifying and formatting tables with new-lines. However, sometimes it concatenates cells of the table together, making the results sometimes unuseable. pymuPDF simply extracts the text data from a PDF akin to a copy and paste. All the data is preserved, but the structure of the table is obscured. One can combine the structure from pyPDF2 and the data from pymuPDF to assemble an accurate table.

C. Flags

To make development of current and future models more accessible, we have implemented a number of flags in our program. `-q` is quiet mode, and doesn't output a stack trace. `-d` is a debug mode, which will print inputs to each function. `-l [Path/To/File]` Logs all failed files to a file. `-errordir [Path/To/Directory]` copies all failed files to a directory, separated in folders by their error message. This is very helpful for sorting out error types and significantly speeds up development time. Also, we have `-asc` and `-desc` for sorting the data chronically or inverse chronically, while `-s <keyword>` will help to detect specific keyword from the dataset and extract the specified data.

RDHS	Dengue Fever		Dysentery		Encephalitis		E
	A	B	A	B	A	B	A
Colombo	132	4147	0	5	0	3	
Gampaha	66	1768	2	7	0	4	
Kalutara	50	1155	1	11	1	1	
Kandy	47	1564	0	5	0	0	
Matale	3	307	0	1	0	0	

Fig. 1. Semantic meaning of data determined by the lead cell in rows and columns. All cells in the red box relate to Dengue Fever. All cells in the green box relate to Gampaha.

D. Models

Different models encode instructions for each document type (i.e. country). While the reading of files is the same, converting them to structured data is not. In our case, we made a model for Sri Lanka and a model for Brazil. This way, we can keep common elements from both models together. This also sped up the development of models. While Sri Lanka's model took a long time to develop alongside the parser, Brazil's model was completed relatively swiftly as it borrowed a lot of functionality from Sri Lanka.

Models generally use two steps: First, take the raw text data and only return the necessary subsection where the table resides. Second, parse the subsection into table data and return that. In Brazil's data, we were able to skip the first step due to the data already consisting of only the necessary subsection.

1) Parsing tables: The first line of the table is the header, which contains the semantic information for each analogous cell in following rows. For example, for the column in the header with ”Dengue Fever”, all entries in different rows of the same column have data regarding Dengue Fever. (Figure 1) Thus, the first step is to understand the semantic values of the header. Next, you need to take each subsequent row. The first element of each row is usually also semantic data that pertains to the entire row. For example, ”Gampaha” in the first cell means all values in the row have the location Gampaha. Then for each cell, you must assign the number in the cell as the number of cases, and use the semantic data from the header and the first cell to fill in other information.

2) Handling multiple spellings or typos: Ensuring consistent data is ideal, yet it's often challenging to achieve due to factors such as typographical errors, unseen characters, erroneous figures, or misplaced data. For instance, data from Sri Lanka occasionally made changes in writing style, or contained typographical errors. This led to confusion between different names that refer to the same district, such as 'Gampaha' and 'paha', resulting in fluctuations in the list of reported districts.

Initially, our approach was straightforward: we compiled a dictionary of incorrect or mistaken words and replaced them with the correct ones whenever encountered. However,

this method proved insufficient for addressing new errors or emerging types of typographical mistakes, requiring the adoption of a more effective strategy. Next, we decided to input the entire dataset, including text and individual words, into a Large Language Model (LLM) to identify the words within the text. While this method produced results, its execution was slow, and its performance varied, occasionally resulting in inaccuracies.

Therefore, for disease names, we devised a hybrid approach combining a dictionary and a modified version of the similarity method called longest subsequence. When encountering a new error, the system attempts to match the erroneous word with the correct disease name. If the similarity meets a certain threshold, the word is classified as the specific disease, and the erroneous word is stored in the dictionary, indicating its association with that disease. This minimizes the computational resources required for future identification. However, if the similarity does not meet the threshold, the system conducts a secondary search for the closest erroneous word in the dictionary. The threshold for this secondary pass is slightly lower, considering the possibility of dictionary corruption if slightly altered words consistently occur, potentially ends up matching a totally different word into another word.

We adopted a unique approach for handling district and city names due to the frequent occurrence of identically named locations across multiple countries, we cannot use similarity method, since it was too faulty. Instead, to differentiate between these locations, we appended the name of the respective country to each location entry during data retrieval. Furthermore, to ensure consistency and improve processing efficiency, we preprocessed the location names by removing non-alphabetic characters, as geographical names typically do not contain numeric or special characters. This preprocessing step was crucial before feeding the data into subsequent models for further analysis.

3) Locational data: After preprocessing the location names, we utilized an API to retrieve location data. This included gathering information such as the type of location (e.g., district, city, or street level), the location boundary, and precise latitude and longitude coordinates, along with the associated country identification number. To optimize efficiency and reduce redundant API requests, which could potentially decrease computation time, we implemented a simple CSV-based dictionary. This dictionary stored each unique location value with corresponding data, allowing us to reuse the data whenever it was needed.

E. Sri Lanka model

Sri Lanka had the more complex model of the two. The model worked by first extracting the table from the text along with the timestamp of the information. Then, it uses the table and the timestamp to generate individual entries into our dataset.

1) Extracting relevant information from text: To extract the desired data accurately, we implemented a strategy to identify the beginning and end of tables within the dataset. Through observation, we recognized specific keywords such as 'RDHS' or 'Division' that typically marked the start of a table. By setting these keywords as markers for table boundaries, we achieved a high extraction accuracy of 97%. Despite this success, occasional errors occurred due to misplaced keywords within the data. We had difficulties with the mark of table starts being too early. We dealt with this by using a function that determines what disease a word is. We created this function initially to overcome typos and abbreviations in words to map them to a disease consistently, but it also proved to be a good check to ensure the first row of our table were disease names.

Apart from the tabular disease data, the other info we needed to extract from the pdfs were the dates that the data was collected over. For this step of data processing, timestamps played a crucial role. We employed regular expressions for pattern matching; however, we encountered challenges such as typos and formatting inconsistencies. For instance, some timestamps lacked whitespace between numerical values and letters, or between the month and year components. Moreover, incorrect or distorted timestamp data further complicated the extraction process. To mitigate these issues, we meticulously crafted our regular expressions, prioritizing the detection of common timestamp formats like '17th - 23rd May 2020' initially. Subsequently, we adjusted our detection approach to accommodate alternative formats, such as '30th June - 6th July 2018', upon failure to identify the common sequence and so on. Another small hurdle were abbreviated and inconsistent months names (e.g. "dece", "dec", "december") which was solved by mapping possible names for a month to their number(1-12) in a dictionary.

For consistent weekly data, it may be easier to use metadata of the document or define a start date, and assume the successive reports are for exactly the next week instead of relying on NLP techniques for timestamps. This may work with other datasets, but we strayed away from this because of inconsistencies with the Sri Lankan disease reports with dates collected vs the dates reported and because of outliers such as the last week of year sometimes reporting yearly data.

2) Parsing table: The parsing of the table was done using only the pymupdf library. In previous iterations, we used both pyPDF2 and pymupdf in conjunction, but for Sri Lanka we found a way to do it that only required one of the inputs. (If you'll recall, myPDF2 preserves the structure of the table but corrupts data, while pymupdf preserves the data of the table but does not preserve the structure)

We should also mention that Sri Lanka had 'A' and 'B' rows (see Figure 1). 'A' rows represented the count for that week, while 'B' rows represented the running total count for the whole year. For our parser, we skipped each 'B' row and only collected data from 'A' rows.

The initial challenge was how the program would properly parse the header. First, we ran a function to concatenate header elements so they work properly with our system to handle multiple spellings or typos. Since many of the data contained a mix of newline and space characters between words, a simple function was written to organize and unify them according

to the module's expected input. Then, the data was given to another function that returns a list of all the diseases, which will return proper, standard spelling.

Next, the challenge continued with having data that should be separated by spaces instead being concatenated in a single column due to the limitations of pymupdf. Since the expected value for each table was a location name followed by the number of cases, the program detected where the alphabetic characters appear and where the numeric characters appear, then constructed a new row each time an alphabetic character was detected. Additionally, we put a warning when there is an irregular number of diseases found since the table should expect a constant amount of diseases.

3) Manually entered data: With all of the above steps, we tried to account for all possible PDFs and tables, but the data we worked with was not perfect and neither were all the NLP techniques we used. Sometimes numbers were far off, sometimes the PDF and table were inconsistently formatted, sometimes the date would be missing from its usual spot, and sometimes disease names were split into multiple words or abbreviated in ways that are very difficult for code to consistently parse. Because of the specificity and variety of errors we were getting, when we had only 10 errors out of 500+ PDFs, we added a manual flag to our program. This allowed us to look at the text extracted from the PDF and input the date, and the area that the table is. In this step, we could also fix one-off disease name errors, spacing errors, or numbers that were off that week.

F. Brazil model

From the scraper, data was stored in a text file where the first line was the year, the second was the data source, the third was the disease name, and the fourth line onward was the table itself. The year and data source were extracted from each text file, and the table was fed into a parsing algorithm.

1) Parsing table: Since the table is already structured, there is no need for complex methods to parse the data. The first line of the table is the header, which contains the semantic information for each analogous cell in the following rows. For example, for the column in the header with "Semana 03" (Semana means week), all entries in different rows of the same column refer to week 3. We generate data by rows. For every row, we get the location name based on its first column and get the locational data. For every cell in the row, we create an entry with the number of cases in the cell, the locational data, and the timestamps generated from the time column in the header.

G. Make your own model

We provide starter code for new models that can be created. These models will process the data in the form of strings. The first requirement is that the data must be in a form which could be converted into text, or must be in a form of strings, converted prior to input.

Next, there are two models required to process the text. One model is responsible for capturing the important and relevant data from the full strings. The other model converts that raw data into a structured format, for the examples given, into 2D array.

Limitations. Due to the models being based on the structure of the data, documents that don't have consistent structure cannot be parsed by the same model. For example, Sri Lanka's data before 2013 has a different format, requiring a different model. Additionally, typographical errors or formatting errors might result in models returning an error on certain papers. Thankfully, papers that yield an error aren't appended to the dataset, and if such papers are few, they can be manually transcribed.

V. Earth observational satellite data

Collected disease data can be combined with related Earth Observational data to build prediction models on how the disease outbreak could take place with respect to the weather and climate data. Among many EO datasets that can used in analysis work, we showcase how the precipitation (rainfall) data can be obtained from NASA and utilized as a one of the possible usecases.

The Global Precipitation Mission (GPM) data comes in particular formats, such as network Common Data Form (NetCDF) or Hierarchical Data Format 5 (HDF5) [15], as it gets downloaded from the NASA data portal. NetCDF is a self-describing file format designed to store multidimensional scientific data such as temperature, humidity, pressure, wind speed, etc, along with its metadata [16].

NASA's Integrated Multi-satellite Retrievals for GPM (IMERG) [17] using a GPM satellite constellation contains the precipitation data. NASA GPM data downloaded is via an API or a data tool such as NASA Giovanni [18], through the NASA Earthdata portal. Once the precipitation data in the chosen temporal and spatial resolution is downloaded in either netCDF or HDF5, it can be converted to comma-separated value (CSV) format. The converted CSV data can be used to create data frames. The data frames are the most commonly used method for data analytics and machine learning model implementation using Python and R programming languages. Also, the CSV data can be easily shared, and it supports the interoperability aspect and is agnostic from any proprietary data format.

Here, as an example, we showcase how we used NASA Earth Observational data (i.e., Total surface precipitation) downloaded using a data tool called NASA Giovanni. It provided the precipitation data in nc4 (netCDF4) files format, which includes rich metadata such as units, latitude, and longitude resolution, any fill-missing value, etc, shown in the figure 2.

Precipitation data is per region using the chosen temporal resolution. It contains longitude, latitude, and precipitation values, which we converted to machine-readable CSV format to build our data frames, which were used in the analysis shown in the sections below.

```
<class 'netCDF4._netCDF4.Variable'>
float32 M2TMNXFLX_5_12_4_PRECTOT(lat, lon)
    _FillValue: 1000000000000000.0
    fmissing_value: 1000000000000000.0
    fullnamepath: /PRECTOT
    long_name: Total surface precipitation
    missing_value: 1000000000000000.0
    origname: PRECTOT
    vmax: 1000000000000000.0
    vmin: -1000000000000000.0
    standard_name: precipitation
    quantity_type: Precipitation
    product_short_name: M2TMNXFLX
    product_version: 5.12.4
    coordinates: lat lon
    units: kg m-2 s-1
    cell_methods: time: mean
    latitude_resolution: 0.5
    longitude_resolution: 0.625
unlimited dimensions:
current shape = (24, 23)
filling on
```

Fig. 2. Precipitation data in the netCDF4 format before converting to CSV format.

VI. Dataset features

A. Disease data

Data is stored in a comma-separated format (.csv) and has the following features: Disease Name, Cases, Location Name, Country Code, Region Type, Latitude, Longitude, Region Boundary, Timestamp Start, Timestamp End, and Source File (for debugging and data integrity).

For Sri Lanka, we have data for 12 conditions: Dengue Fever, Dysentery, Encephalitis, Enteric Fever, Food Poisoning, Leptospirosis, Typhus Fever, Viral Hepatitis, Human Rabies, Chickenpox, Meningitis, and Leishmaniasis. This data is from 2013 to 2024. It has a district-level spatial resolution and a weekly temporal resolution.

For Brazil, we have data for 44 conditions: Work accident, Accident due to venomous animals, Work Accident with Exposure to Biological Material, Botulism, Work-Related Cancer, Cholera, Whooping cough, Work-Related Dermatosis, Dengue Fever, Diphtheria, Acute Chagas Disease, Exanthematous Diseases, Schistosomiasis, Yellow fever, Chikungunya fever, Rocky Mountain spotted fever, Typhoid fever, Hantavirus, Hepatitis, Pandemic Influenza, Exogenous Poisoning, Visceral Leishmaniasis, American Tegumentary Leishmaniasis, American cutaneous leishmaniasis, Leptospirosis, Read/Dort, Malaria, Meningitis, Acute Flaccid Paralysis, Work-Related NIHL, Plague, Work-Related Pneumoconiosis, Human Rabies, Acquired Syphilis, Congenital Syphilis, Syphilis in Pregnant Women, Congenital Rubella Syndrome, Accidental tetanus, Neonatal tetanus, Congenital Toxoplasmosis, Gestational Toxoplasmosis, Work-Related Mental Disorder, Varicella, Interpersonal/Self-Inflicted Violence, and Zika virus. The earliest of these reports are from 2007, and the latest are to the present, 2024. The data has a municipality-level spatial resolution and depending on the condition, a weekly or monthly temporal resolution.

B. Contextual data And data tools

The disease data was fused with geo-spatial satellite data (see for example, figure 3). Each region has Precipitation rate (Precipitationcal), Temperature (Tair_F_Inst), Surface Air Pressure (Psurf_F_Inst), Plant Canopy Surface Water (Canopint_Inst), Air Quality(Qair_F_Inst), Surface Soil Moisture (Soilmoi0_10cm_Inst), and Normalized Difference Vegetation Index (NDVI). The satellite data was from NASA's Earth Observational Satellite Data, specifically Giovanni, NOAH, and MODIS. The satellite data is daily, so to match our datasets' weekly resolution, we recorded the weekly minimum, maximum, and mean of each week. The spatial resolution of the data is by district for Sri Lanka. For Brazil, we have run the NLP module to obtain the disease/condition data, however, within the scope of this work and due to the size of the Brazil earth-observational satellite data, we have not run the second module to download and align the data. For disease/climate modeling in Brazil, this last step would have to be done following the same methodology we have presented for the Sri Lanka data. (figure 4)

1) Data Visualization: We showed sample visualizations of the data in figure 4, figures 4 and 5 to illustrate the richness of the data by showing some of the insights simple data exploration can give. We do provide such visualizations which supports zooming in on specific regions, diseases, and time scales.

2) Data Management: To enhance the accessibility and usability of data, we developed an efficient model for the extraction of information based on disease name, location, or user-defined date parameters. Employing keywords such as 'Colombo', the module will identify and retrieve relevant data entries. Furthermore, the tool offers ability to sort the data chronically.

VII. Potential use cases

This data provides a fertile ground for developing models for all diseases in the data. New features can also be added. For example, data for diseases such as Dengue, Chikungunya, or Cholera can be fused with data such as healthcare funding, mobility data, and other socioeconomic and contextual factors to determine effective safety measures, predict disease outbreaks, and aid in advocating for meaningful reform. The data also serves as a blueprint for synthetic data. This data can be used in many ways, we give one concrete use case.

A. Spatio-Temporal Dengue Modeling

We used the data to model and predict outbreaks of Dengue in Sri Lanka and Brazil [19]. The authors show that spatio-temporal modeling combining disease dynamics with spatial spread plus economic and demographic data (exactly the data

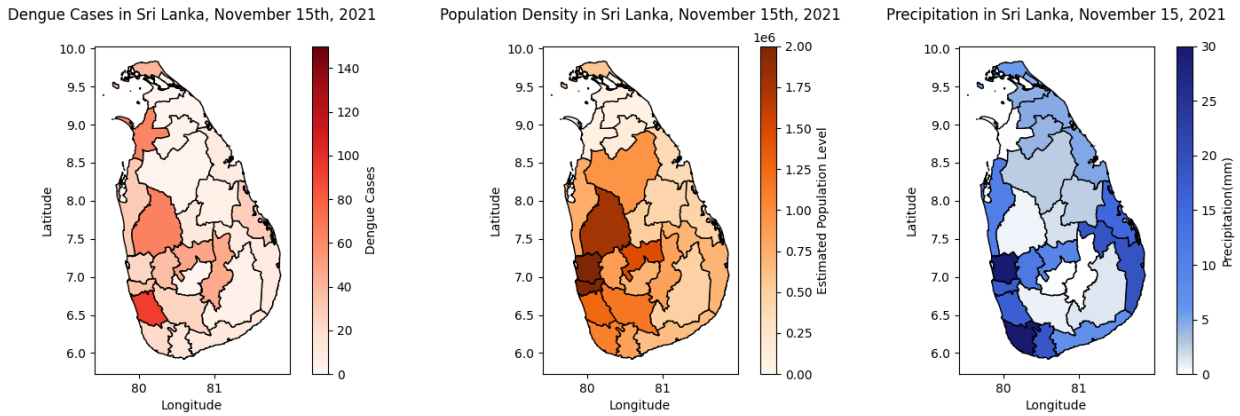

Fig. 3. Number of cases, population density, and precipitation in Sri Lanka, on Nov 15th, 2021

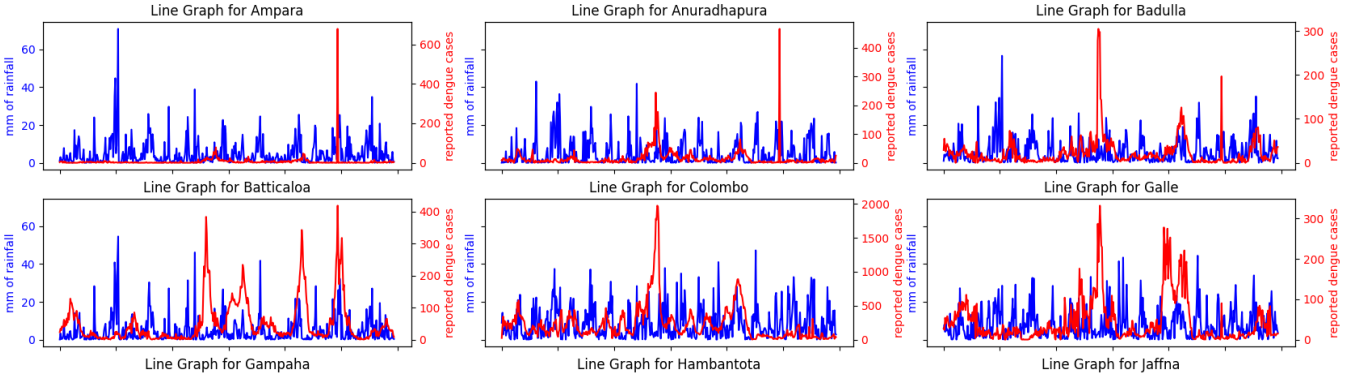

Fig. 4. Dengue cases in Sri Lankan districts with rainfall from satellite data

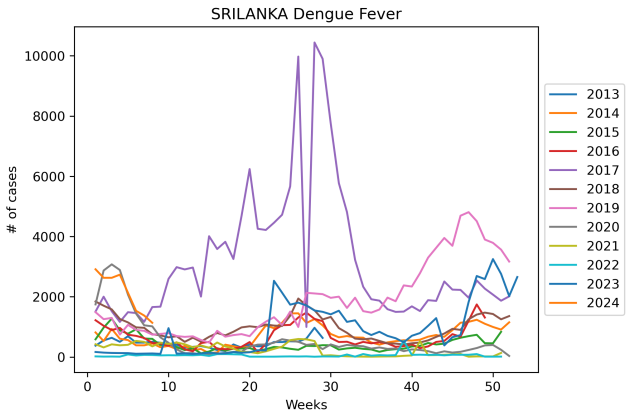

Fig. 5. Dengue cases by year in Sri Lanka

that has been created and aligned in this paper) provides the best predictive performance. We give a brief summary of the model in [19] and demonstrate the predictive performance.

a) Summary of Spatio-Teporal GNN Model: The model was designed to predict disease outbreaks in Sri Lanka and potentially apply it to other countries [19]. The model was based on Graph Neural Network, GNN, using Time Series Cross Validation technique to improve accuracy. Comparative analysis with traditional models, such as Random Forest, demonstrated that the GNN-based model achieved lower error rates in forecasting disease outbreaks [19].

VIII. Conclusion and Future Work

We have created a modular system to convert unstructured data into structured csv data, complemented by sophisticated data scraping mechanisms for efficient data collection. Using our system, we have extracted complete datasets for over 40 diseases and conditions in Brazil and Sri Lanka from 2007 to the 2024. This data has been fused with NASA satellite data and European Union population density data to provide a valuable dataset for researchers. Additionally, the tools are available in an open-source, open-science framework, allowing researchers to use our methods to expand our dataset with new reports or new document types. With this dataset, we hope researchers will make disease modeling, predictions, and policy advancements that can save lives.

So far, we've gathered 12 diseases data in Sri Lanka and 44 diseases data in Brazil. Our code is designed to easily adapt to different PDF formats, making it to be used for any country or data type, such as Bangladesh, which publishes data that would be compatible with our model, but would

require a translation module. With the increasing effects of climate change on public health, the insights from this dataset can be vital in understanding how environmental change might influence disease patterns. Furthermore, additional features can be added such as mobility data, income levels, and other relevant factors in the future.

IX. Acknowledgements

We would like to express our sincere gratitude to the Department of Computer Science at Rensselaer Polytechnic Institute (RPI) and for the computing resources we obtained from the Institute for Data Exploration and Applications (IDEA) and AiMOS supercomputer for the computing requirements at RPI. We thank NASA Goddard Space Flight Center and all other NASA entities who provided the Earth's observational data. We also thank the Ministry of Health in Sri Lanka and Brazil for publicly sharing the data on disease cases on their website.

X. Dataset and GitHub Repository

GitHub: https://bit.ly/3VJTIrA
Dataset: https://bit.ly/4b2yXM8

References

[1] G. Bhatia, A. Tewari, G. Gurbani, S. Gokhale, N. Varyomalani, R. Kirtikar, Y. Bhatia, and S. Athavale, "Extraction of tabular data from pdf to csv files," in *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2020, Volume 1*. Springer, 2021, pp. 183–193.

[2] M. Namysł, A. M. Esser, S. Behnke, and J. Köhler, "Flexible hybrid table recognition and semantic interpretation system," *SN Computer Science*, vol. 4, no. 3, p. 246, 2023.

[3] A. Shigarov, A. Altaev, A. Mikhailov, V. Paramonov, and E. Cherkashin, "Tabbypdf: Web-based system for pdf table extraction," in *Information and Software Technologies: 24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018, Proceedings 24*. Springer, 2018, pp. 257–269.

[4] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A method to extract table information from pdf files," in *IICAI*, vol. 2005. Citeseer, 2005, pp. 1773–1785.

[5] E. Oro and M. Ruffolo, "Pdf-trex: An approach for recognizing and extracting tables from pdf documents," in *2009 10th International Conference on Document Analysis and Recognition*, 2009, pp. 906–910.

[6] M. P. Polak and D. Morgan, "Extracting accurate materials data from research papers with conversational language models and prompt engineering–example of chatgpt," *arXiv preprint arXiv:2303.05352*, 2023.

[7] A. S. Corrêa and P.-O. Zander, "Unleashing tabular content to open data: A survey on pdf table extraction methods and tools," in *Proceedings of the 18th Annual International Conference on Digital Government Research*, ser. dg.o '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 54–63. [Online]. Available: https://doi.org/10.1145/3085228.3085278

[8] S. Sheikhalishahi, R. Miotto, J. T. Dudley, A. Lavelli, F. Rinaldi, V. Osmani *et al.*, "Natural language processing of clinical notes on chronic diseases: systematic review," *JMIR medical informatics*, vol. 7, no. 2, p. e12239, 2019.

[9] D. S. Carrell, S. Halgrim, D.-T. Tran, D. S. Buist, J. Chubak, W. W. Chapman, and G. Savova, "Using natural language processing to improve efficiency of manual chart abstraction in research: the case of breast cancer recurrence," *American journal of epidemiology*, vol. 179, no. 6, pp. 749–758, 2014.

[10] R. I. Dogan and Z. Lu, "An inference method for disease name normalization," in *2012 AAAI Fall Symposium Series*, 2012.

[11] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," *arXiv preprint arXiv:1605.02276*, 2016.

[12] R. Conijn, M. Van Zaanen, M. Leijten, and L. Van Waes, "How to typo? building a process-based model of typographic error revisions," *The Journal of Writing Analytics*, vol. 3, pp. 69–95, 2019.

[13] H. Li, R. K. Srihari, C. Niu, and W. Li, "Location normalization for information extraction," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[14] (2024) Sri lanka weekly epidemiological report. https://www.epid.gov.lk/weekly-epidemiological-report/weekly-epidemiological-report. Sri Lanka Dataset.

[15] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An overview of the hdf5 technology suite and its applications," in *Proceedings of the EDBT/ICDT 2011 workshop on array databases*, 2011, pp. 36–47.

[16] R. Rew and G. Davis, "Netcdf: an interface for scientific data access," *IEEE computer graphics and applications*, vol. 10, no. 4, pp. 76–82, 1990.

[17] G. J. Huffman, D. T. Bolvin, E. J. Nelkin, and J. Tan, "Integrated multi-satellite retrievals for gpm (imerg) technical documentation," *Nasa/Gsfc Code*, vol. 612, no. 47, p. 2019, 2015.

[18] J. G. Acker and G. Leptoukh, "Online analysis enhances use of nasa earth science data," *Eos, Transactions American Geophysical Union*, vol. 88, no. 2, pp. 14–17, 2007.

[19] Anonymous, "Graph representation learning for dengue forecasting (under submission to NeurIPS)," 2024.