Exploration of PDF Scraping solutions

(note: this document will be converted into an rmd)

Background/Rational

In the field of epidemiology, the rapid and accurate extraction of data from various sources, including situation reports on disease outbreaks, is critical for monitoring, understanding, and responding to public health crises. Situation reports are often released by health organizations, governmental bodies, and international agencies to provide up-to-date information on the progression and impact of outbreaks. However, these reports are frequently published in PDF format, which poses significant challenges for data extraction. PDF scraping solutions can empower modelers, epidemiologists and public health professionals to efficiently extract, analyze, and act on crucial epidemiological data, ultimately contributing to better preparedness, response, and control of infectious disease outbreaks.

Objective

As a project in the 2023 Early stage outbreaks analytics hackathon, at the WHO Berlin Hub, we set out to document and test PDF scraping tools. The objective of this work is to explore and highlight the suitability, advantages and limitations of existing PDF scraping solutions.

Approach

Our approach comprised the evaluation of selected R and Python packages, text and image recognition methods, and the utilization of Large Language Model (LLM) capabilities for extraction of data from PDF with varying degrees of complexity.

Target PDFs

We attempted to perform scraping on three target tables, which had varying degrees of complexity. For the purposes of this document, we focus on scraping of a relatively simple, text based table. Further code used to scrape other tables can be seen in our code repository *[link to repo]*.

The following tables show the Epidemiological indicators for the COVID-19 situation by division in the country.

Division	New Tests (last 7 Days)	New Tests (last 8-14 Days)	Test/100K/ Week	% change in new tests	% of new tests	Test Positivity	% change in test positivity	Test/Case
Barishal	123	167	1.3	-26.3%	0.5%	3.3%	-46%	30.8
Chattogram	1 883	2 404	5.6	-21.7%	7.2%	6.7%	-37%	14.9
Dhaka	21 646	21 089	50.2	2.6%	82.9%	3.5%	-37%	28.7
Khulna	277	397	1.5	-30.2%	1.1%	4.0%	-44%	25.2
Mymensingh	521	594	4.0	-12.3%	2.0%	5.2%	-48%	19.3
Rajshahi	584	903	2.7	-35.3%	2.2%	9.9%	-18%	10.1
Rangpur	284	807	1.5	-64.8%	1.1%	3.9%	42%	25.8
Sylhet	800	807	6.8	-0.9%	3.1%	2.4%	-51%	42.1
National	**26 118**	**27 168**	**15.3**	**-3.9%**	**100.0%**	**3.9%**	**-38%**	**25.8**

Example 1. of Target table for scraping

Summary of evaluated tools

Package	Software Type	Category	Advantages	Disadvantages
Tabulizer	R	R package	Simple code Several useful options for different scenarios	Potential installation problems ● Not on CRAN ● Requires Java
pdftools	R	R package	Simple installation, faster than tabulizer	Requires installation of poppler for Linux, requires extensive text parsing work downstream
pdf2image	Python	Python library using OCR-based approach (image recognition)		
img2table	Python	Python library		

		using OCR-based approach (image recognition)		

Approach	Ease of installation	Ease of use	Value accuracy	Table structure complexity	Image parsing
Tabulizer (Tabula API)	Medium	High	High	Medium	No
Text parsing	High	Medium	High	Low	no
OCR	Medium*	Medium	Low	High	Yes

Tabulizer R package

Tabulizer is an R package maintained by ropensci.It acts as an R wrapper for the Tabula java library, which requires a local installation of Java. The package is not currently available on CRAN, but can be installed from github.

Code example:

```
# Install tabulizer package (requires a local installation of Java)
devtools::install_github("ropensci/tabulizer")

#Load packages
library(tabulizer)
library(tidyverse)
library(janitor)

pdf_1 <- "Examples/PDF-1.pdf"
t_1_raw <- extract_tables(pdf_1, output="matrix")
str(t_1_raw)
t_1_raw[[4]]
t_1_raw_headers <- t_1_raw[[4]][1:2,] %>%
  t() %>%
```

```
  as_tibble() %>%
  unite(name, c("V1","V2"), sep=" ") %>%
  mutate(name = trimws(name)) %>%
  pluck("name")
t_1_raw_headers
t_1_clean <-
  t_1_raw[[4]] %>%
  as_tibble() %>%
  tail(-2)
names(t_1_clean) <- t_1_raw_headers
t_1_clean
t_1_clean <- t_1_clean %>%
  mutate(across(c(`New Tests (last 7 Days)`,
                  `New Tests (last 8-14 Days)`,
                  `Test/100K/ Week`,
                  `Test/Case`),
                ~ as.numeric(gsub('\\s','',.x))),
         across(c(`% change in new tests`,
                  `% of new tests`,
                  `Test Positivity`,
                  `% change in test positivity`),
                ~ as.numeric(gsub('%','',.x,fixed=T))))
t_1_clean
```

```
> t_1_clean
# A tibble: 9 × 9
  Division   `New Tests (last 7 Days)` `New Tests (last 8-14 Days)` `Test/100K/ Week` `% change in new tests` `% of new tests` `Test Positivity`
  <chr>                          <dbl>                        <dbl>             <dbl>                  <dbl>            <dbl>              <dbl>
1 Barishal                         123                          167               1.3                  -26.3              0.5                3.3
2 Chattogram                      1883                         2404               5.6                  -21.7              7.2                6.7
3 Dhaka                          21646                        21089              50.2                    2.6             82.9                3.5
4 Khulna                           277                          397               1.5                  -30.2              1.1                4
5 Mymensingh                       521                          594               4                    -12.3              2                  5.2
6 Rajshahi                         584                          903               2.7                  -35.3              2.2                9.9
7 Rangpur                          284                          807               1.5                  -64.8              1.1                3.9
8 Sylhet                           800                          807               6.8                   -0.9              3.1                2.4
9 National                       26118                        27168              15.3                   -3.9            100                  3.9
# i 2 more variables: `% change in test positivity` <dbl>, `Test/Case` <dbl>
```

The tabulizer package correctly identified and parsed the table. Some additional code was required to clean and format the data, including accounting for column headers that spanned multiple rows. This approach was applied to additional examples with similar results. With relatively simple modifications to the cleaning and formatting code, tabulizer was also able to handle tables that spanned multiple pages and columns that included merged cells.

Text Parsing

This approach extracts the texts from the pdf (using the package **pdftools**) and tries to parse the texts into tables. To separate rows, we are using line returns “\n”. To separate columns, we use a changeable regex. We found that in most cases, 2 or more spaces was the best approach ("\\s{2,}"), however we found that this doesn’t work well in some cases where tables had missing values replaced by spaces. In that case an upper bound limit can be used (e.g. "\\s{2,15}"), however we found that this was not always working because different rows could use different amount of spaces to delimit columns.

When it works, the advantages of this approach are speed and simplicity. This is useful for cases where hundreds of pdf’s need to be parsed rapidly.
It can also be used when tabulizer is not available (java issues).

```
pacman::p_load(dplyr, pdftools, stringr)

pdftable_to_dataframe = function(files, keeppages, start, end, setcols=c(),
splitcolsregex="\\s{2,}", debug=FALSE){
  if(debug) print(paste('going to extract' ,length(files), "file(s)"))

  for(file_i in 1:length(files)){
    file = files[file_i]
    # print(paste0('file #', file_i, ': ', file))

    pages = pdf_text(file)                      ## EXTRACT ALL TEXT INTO PAGES
    # print(paste('pages:',length(pages)))
    pages = pages[keeppages]                    ## TRIM SELECTED PAGES
    alltext = paste(pages, collapse = '\n')     ## MERGE ALL PAGES
    # print(alltext)


    lines = alltext %>% strsplit(split="\n")    ## SPLIT INTO LINES
    lines = lines[[1]]                          ## UNLIST
    lines = lines[start:end]                    ## TRIM SELECTED LINES
    if(debug==2) print(lines)

    splittedlines = lines%>%strsplit(split=splitcolsregex)          ## SPLIT VALUES INTO COLS
(default: 2 or more spaces)
    splittedlines = splittedlines[lapply(splittedlines,length)>0]   ## REMOVE E:PTY LINES
    # if(debug) print(splittedlines)

    ### FIND MAX NUMBER OF COLS -----
    ncols = 0
    for (line_j in splittedlines){
      # print(length(line_j))
      if(ncols < length(line_j)) ncols = length(line_j)
    }


    if(file_i==1) df = data.frame()               ## INIT EMPTY DF
    for (line_i in splittedlines){
      if(debug) print(paste(length(line_i), 'cols: ', paste(line_i, collapse = " | ")))
      if(length(line_i) == ncols ) {              ## KEEP ONLY LINES WITH ALL VALUES
(alternative would be to complete vector of values to make sure the length is the same)
```

```
        df_i = data.frame(t(line_i))              ## CONVERT TO SINGLE LINE DF
        df = rbind(df, df_i)                      ## APPEND TO MAIN DF
      }
    }
  }

  if(debug) print(paste('keeping only row with', ncols, 'columns'))

  if(length(setcols) > 0) names(df) = setcols     ## SET COL NAMES

  return(df)
}
```

```
### PDF 1 > PAGE 4 > TABLE 1 -----
setwd(dirname(rstudioapi::getSourceEditorContext()$path))
pdftable_to_dataframe(c('Examples/PDF-1.pdf'), 4, 5, 25, c('Division'
                                                           ,'New Tests (last 7 Days)'
                                                           ,'New Tests (last 8-14 Days)'
                                                           ,'Test/100K/Week'
                                                           ,'% change in new tests'
                                                           ,'% of new tests'
                                                           ,'Test Positivity'
                                                           ,'% change in test positivity'
                                                           ,'Test/Case')) %>%
  mutate(
    `Test/Case` = as.numeric(`Test/Case` ),
## e.g. as numeric
    `New Tests (last 7 Days)` = as.numeric(gsub('\\s', '', `New Tests (last 7 Days)` )),
## e.g. thousands as numeric
    `New Tests (last 8-14 Days)` = as.numeric(gsub('\\s', '', `New Tests (last 8-14 Days)` )),

    `% change in new tests` = as.numeric(gsub('%', '', `% change in new tests` )) / 100,
## e.g. percent as numeric
    `% of new tests` = as.numeric(gsub('%', '', `% of new tests` )) / 100,
    `Test Positivity` = as.numeric(gsub('%', '', `Test Positivity` )) / 100,
    `% change in test positivity` = as.numeric(gsub('%', '', `% change in test positivity` ))
/ 100
    )
```

Result:

	Division	New Tests (last 7 Days)	New Tests (last 8-14 Days)	Test/100K/Week	% change in new tests	% of new tests	Test Positivity	% change in test positivity	Test/Case
1	Barishal	123	167	1.3	-0.263	0.005	0.033	-0.46	30.8
2	Chattogram	1883	2404	5.6	-0.217	0.072	0.067	-0.37	14.9
3	Dhaka	21646	21089	50.2	0.026	0.829	0.035	-0.37	28.7
4	Khulna	277	397	1.5	-0.302	0.011	0.040	-0.44	25.2
5	Mymensingh	521	594	4.0	-0.123	0.020	0.052	-0.48	19.3
6	Rajshahi	584	903	2.7	-0.353	0.022	0.099	-0.18	10.1
7	Rangpur	284	807	1.5	-0.648	0.011	0.039	0.42	25.8
8	Sylhet	800	807	6.8	-0.009	0.031	0.024	-0.51	42.1
9	National	26118	27168	15.3	-0.039	1.000	0.039	-0.38	25.8

OCR based approach with img2table (python)

An approach based on OCR techniques implemented in python was tested. The strategy was to import PDF files as images and then apply common OCR techniques to extract tables. The first can be done by means of the function `convert_from_path` of the python library {pdf2image}, which creates a list-like object containing the corresponding images for each page in the PDF file. By means of the Tesseract OCR implementation provided by the {img2table} library, along with the `extract_tables` method, it was possible to extract tables both in English and Bengali .

{img2table} relies on an optical character recognition (OCR) engine called Tesseract. This can be installed running:

```
sudo apt install tesseract-ocr
```

Tesseract supports a compelling list of languages. To install a particular language, you need to specify its language code. For instance for Bengali:

```
sudo apt-get install tesseract-ocr-ben
```

The python version used by {img2table} is python 3.7. To run the code in a controlled environment, it is recommended to create a virtual environment in your local machine, e.g. with conda:

```
conda create -name py37 python=3.7
conda activate py37
```

The {img2table} and {pdf2image} packages can easily be installed using pip:

```
pip install img2table
```

```
pip install pdf2image
```

In order to tryout the effect of applying filters to the images

Code example:

```
# Install tabulizer package (requires a local installation of Java)
devtools::install_github("ropensci/tabulizer")
```

Example 1:

```
from Scripts.ocr_functions import *
import pdf2image
import cv2
from img2table.document import Image
from img2table.ocr import TesseractOCR

imgs = pdf2image.convert_from_path('./Examples/PDF-1.pdf')
img = imgs[3]
img_array = np.asarray(img)
cv2.imwrite('./Images/PDF-1_orig.jpg', img_array)

# table extraction
image = Image(src = './Images/PDF-1_orig.jpg')
ocr = TesseractOCR(lang = "eng")
tables = image.extract_tables(ocr = ocr)
image.to_xlsx('./Tables/PDF-1_orig.xlsx', ocr = ocr)
```

Division	New Cases (last 7 days)	New Cases (last 8-14 days)	Case/100K/ Week	% change in new cases	% of new cases	Cumulative Cases	% of total cases	Rt
Barishal	4	10	0.0	-60%	0.4%	54 192	2.7%	0.78
Chattogram	126	256	0.4	-51%	12.5%	298 130	14.6%	0.63
Dhaka	755	1 175	1.8	-36%	74.7%	1 252 237	61.5%	0.75
Khulna	11	28	0.1	-61%	1.1%	130 911	6.4%	0.54
Mymensingh	27	59	0.2	-54%	2.7%	45 262	2.2%	0.59
Rajshahi	58	109	0.3	-47%	5.7%	121 759	6.0%	0.68
Rangpur	11	22	0.1	-50%	1.1%	64 982	3.2%	0.64
Sylhet	19	39	0.2	-51%	1.9%	67 679	3.3%	0.67
National	**1 011**	**1 698**	**0.6**	**-40%**	**100.0%**	**2 035 152**	**100%**	**0.72**

New Cases(last 7 days)	New Cases(last 8-14 days)	Case/100K/Week	% change innew cases	% ofnew cases	CumulativeCases	% oftotal cases	Rt
4	10	0.0	-60%	0.4%	54 192	2.7%	
126	256	0.4	-51%	12.5%	298 130	14.6%	
755	1175	18	-36%		1 252 237	61.5%	
11	28	0.1	-61%	1.1%	130911	6.4%	
27	59	0.2	-54%	2.7%	45 262	2.2%	
58	109	0.3		5.7%	121 759	6.0%	
11	22	0.1	-50%	1.1%	64 982	3.2%	
1g	39	0.2	-51%	1.9%	67 679	3.3%	
1011	1698	0.6	-40%	100.0%	2035 152	100%	0.72

Example 2 (Bengali):

```
imgs = pdf2image.convert_from_path('./Examples/PDF-2.pdf')
img = imgs[6]
img_array = np.asarray(img)
cv2.imwrite('./Images/PDF-2_6.jpg', img_array)
image = Image(src = './Images/PDF-2_6.jpg')
ocr = TesseractOCR(lang = "ben")
tables = image.extract_tables(ocr = ocr, borderless_tables =
borderless_tables)
image.to_xlsx('./Tables/PDF-2_6.xlsx', ocr = ocr)
```

ক্রমিক নং	প্রতিষ্ঠানের নাম (সরকারী ও স্বায়ত্তশাসিত)	২৪ ঘন্টায় নতুন ভর্তি				গত ০১-০১-২০২৩ হতে অদ্যাবধি			বর্তমানে ভর্তি রোগী
		ডেঙ্গু ফিবার	ডেঙ্গু হেমরোজিক	ডেঙ্গু শক্ সিনড্রোম	মোট	সর্বমোট ভর্তি	মৃত্যু	ছাড়পত্র প্রাপ্ত রোগী	
১	ঢাকা মেডিকেল কলেজ হাসপাতাল	৫৬	০	০	৫৬	৫৭৭০	১৩৮	৫২৩৫	৩৯৭
২	এস.এস.এম.সি. ও মিটফোর্ড হাসপাতাল	৫৭	০	০	৫৭	৫৯১৮	৪১	৫৬২৫	২৫২
৩	বাংলাদেশ শিশু হাসপাতাল ও ইনস্টিটিউট	১৫	০	০	১৫	১৩৮৬	১৭	১২৬০	১০৯
৪	শহীদ সোহরাওয়ার্দী মেডিকেল কলেজ হাসপাতাল	৪৬	০	০	৪৬	২৫৪০	১০	২২৭৩	২৫৭
৫	বিএসএমএমইউ	৮	০	০	৮	৮৮৪	৫	৮৩১	৪৮
৬	পুলিশ হাসপাতাল, রাজারবাগ	১০	০	০	১০	১৪৯৬	৩	১৪৪৩	৫০
৭	মুগদা মেডিকেল কলেজ হাসপাতাল	৭৬	০	০	৭৬	১০৪০৯	১১৮	১০০১৭	২৭৪
৮	বিজিবি হাসপাতাল, পিলখানা, ঢাকা	৯	০	০	৯	২৯৩	০	২৬৫	২৮
৯	সম্মিলিত সামরিক হাসপাতাল, ঢাকা	২৫	০	০	২৫	২৪৪৩	১২	২৩০৮	১২৩
১০	কুর্মিটোলা জেনারেল হাসপাতাল	৭১	০	০	৭১	৪৪২৯	৩১	৪২২৩	১৭৫
১১	কুয়েত বাংলাদেশ মৈত্রী হাসপাতাল	৩৫	০	০	৩৫	২০৭৭	৪	১৯৫৫	১১৮
১২	৩১ শয্যা বিশিষ্ট হাসপাতাল কামরাঙ্গীর চর	০	০	০	০	৩৫	০	৩৫	০
১৩	২৫০ শয্যা বিশিষ্ট টিবি হাসপাতাল শ্যামলী	০	০	০	০	৮৪	০	৭৬	৮
১৪	সংক্রামক ব্যাধি হাসপাতাল ঢাকা	০	০	০	০	১২২	০	১১৫	৭
১৫	নিটোর	০	০	০	০	০	০	০	০
১৬	সম্মিলিত সামরিক হাসপাতাল (সাভার)	০	০	০	০	১৪	০	১৪	০
১৭	সরকারী কর্মচারী হাসপাতাল	৬	০	০	৬	৪৪৪	১	৪২৮	১৫
১৮	ঢাকা মহানগর জেনারেল হাসপাতাল	১	০	০	১	১৩২	০	১২৮	৪
১৯	ঢাকা মহানগর শিশু হাসপাতাল,ঢাকা দক্ষিণ সিটি কর্পোরেশন	০	০	০	০	৩৬	১	৩৫	০
২০	ডিএনসিসি ডেডিকেটেড কোভিড-১৯ হাসপাতাল, মহাখালী, ঢাকা	৭৪	০	০	৭৪	৫৩৭৮	৩৬	৫০৩০	৩১২
বেসরকারী		৪৮৯	০	০	৪৮৯	৪৩৮৮০	৪১৭	৪১২৮৬	২১৭৭
১	বাংলাদেশ মেডিকেল কলেজ হাসপাতাল	৭	০	০	৭	১১৬৫	২	১১৩১	৩২
২	হলি ফ্যামিলি রেড ক্রিসেন্ট হাসপাতাল	১২	০	০	১২	১৯৭০	৬	১৮৭৬	৮৮
৩	বারডেম হাসপাতাল	৮	০	০	৮	৯০৬	১১	৮৫৯	৩৬
৪	ইবনে সিনা হাসপাতাল, ধানমন্ডি, কল্যাণপুর	১৩	০	০	১৩	১০৭৬	৭	৯৯৭	৭২
৫	স্কয়ার হাসপাতাল, ধানমন্ডি	৫	০	০	৫	৮২২	১৪	৭৮০	২৮
৬	কমফোর্ট নার্সিং, ধানমন্ডি	০	০	০	০	৮৫	০	৮০	৫
৭	শমরিতা হাসপাতাল, পান্থপথ	১	০	০	১	২৭১	১	২৫৫	১৫
৮	ডেলটা মেডিকেল কলেজ, মিরপুর	২৮	০	০	২৮	১০৮৪	২	৯৯৩	৮৯
৯	ল্যাব এইড হাসপাতাল	৬	০	০	৬	১৮৪	০	১৫৮	২৬
১০	সেন্ট্রাল হাসপাতাল, ধানমন্ডি	০	০	০	০	৭২১	০	৬৬৮	৫৩
১১	হাই কেয়ার হাসপাতাল লিমিটেড	৪	০	০	৪	১৯৯	০	১৯০	৯
১২	হেল্থ এন্ড হোপ হাসপাতাল	০	০	০	০	১৪৯	০	১৪৬	৩
১৩	গ্রীন লাইফ মেডিকেল হাসপাতাল	৭	০	০	৭	৯৭৬	১২	৯২৯	৩৫
১৪	ইসলামী ব্যাংক সেন্ট্রাল হাসপাতাল, কাকরাইল, মুগদা	১১	০	০	১১	১৬৬৮	৭	১৬০৩	৫৮
১৫	ইউনাইটেড হাসপাতাল লিমিটেড	১০	০	০	১০	৯৫৯	৮	৯০০	৫১
১৬	খিদমা হাসপাতাল, খিলগাঁও	৪	০	০	৪	৩২৯	০	৩১৮	১১
১৭	শহীদ মনসুর আলী মেডিকেল কলেজ হাসপাতাল	১৯	০	০	১৯	১১১৫	৩	১০২১	৯১
১৮	সিরাজুল ইসলাম মেডিকেল কলেজ হাসপাতাল	৫	০	০	৫	৪৬৯	০	৪৪৯	২০
১৯	এভার কেয়ার হাসপাতাল	৪	০	০	৪	৪৬৮	৩	৪৫০	১৫
২০	আদ-দিন উইমেনস মেডিকেল কলেজ হাসপাতাল	৫	০	০	৫	৫৯৩	১	৫৫৩	৩৯
২১	ইউনিভার্সেল মেডিকেল কলেজ ও হাসপাতাল	৬	০	০	৬	৫০২	৮	৪৫০	৪৪
২২	বিআরবি হসপিটালস লিমিটেড	০	০	০	০	১২৯	৪	১১৮	৭
২৩	আজগর আলী হাসপাতাল	৬	০	০	৬	১৪৮৮	১৪	১৪৪০	৩৪

ক্রমিক নং	প্রতিষ্ঠানের নাম (মেরকারী ও বায়ত্বশাসিত)		২৪ ঘন্টায় নতুন ভর্তি				গত ০১-০১-২০২৩ হতে		অন্যত্রগামী	বর্তমানে ভর্তি রোগী
			কিবার	ডেঙ্গুরোগিক	টাইফয়েড	মোট	সর্বমোট ভর্তি	মৃত্যু	চাহিদাপত্র প্রাপ্ত রোগী	
১	বাংলাদেশ পুলিশ হাসপাতাল						২৯৫		১৬৫	১৯
২	উত্তরা আধুনিক মেডিকেল কলেজ হাসপাতাল		২৫			২৫	১৪৫৪		১৩৮৯	৬৪
৩	সালাউদ্দিন হাসপাতাল		২			২	৮৩৩		৮২৩	১৩
৪			২			২	৮৪২		৮৩৭	২৬
৫	উত্তরা ইন্টেন্ট হাসপাতাল		১৪			১৪	৫৫৯		৫২৩	৩৬
৬	আনোয়ার খান মর্ডান মেডিকেল	কলেজ হাসপাতাল	৪			৪	৭৯২		৬৯৬	১৫
৭	মেডিকেল কলেজ কর্ণার ওয়ানে	হাসপাটাল	১৯			১৯	১৪৪৫		১৪১১	৫৭
৮	ডাঃ এম আর খান শিশু হাসপাতাল		১৫			১৫	৬৩৩		৫৫৩	
৯			১২			১২	৬৩৯		৬১১	২৭
১০	আইচি হাসপিটালস লিমিটেড		৬			৬	৫১১		৪৮২	৩৭
১১	মনোয়ারা হাসপিটাল		২			২	১৮৫		১৮৩	২
১২	জামুনা হক শিকদার মেডিকেল কলেজ হাসপাতাল		১			১	১১১		১০২	৫
১৩	এনাম মেডিকেল কলেজ হাসপাতাল		২১			২১	৫৩৫	১৬	৪৫৪	৬৫
১৪	এ এম জেড হাসপাতাল								৫৩৯	৩৬
১৫	লাইফ এড কেয়ার হাসপাতাল						২৩৩		২৯২	১৮
১৬	ধরাবি জেনারেল হাসপাতাল						১১		৮	৩
১৭	জাপান বাংলাদেশ ফ্রেন্ডশিপ হাসপাতাল						১৪২		১৩৩	ন
১৮	এস্টিম ব্যাংক হাসপাতাল						৮৬		৮৫	১
১৯	আলমাদার হাসপাতাল						৫৩৮		৫২৫	১৩
২০	ঢাকা চেম্বার কেয়ার হাসপাতাল						১		১	০
২১	মেট্রোপলিটন মেডিকেল সেন্টার লিমিটেড						৬৫.		৬৩	
২২	সাজেদা হাসপাতাল, কেরানিগঞ্জ						৮৪		৮৪	০
২৩	বসুন্ধরা আদ-দ্বীন মেডিকেল কলেজ হাসপাতাল						২৯৯		২৫০	২৯
২৪	বি আই এইচ এম জেনারেল হাসপাতাল		১৮			১৮	৬৪৭		৫৯৩	৪৬
২৫	ঢাকা কমিউনিটি মেডিকেল কলেজ হাসপাতাল						৬৬৪		৬৬৬	৪৮
২৬	নদার্ন ইন্টারন্যাশনাল মেডিকেল কলেজ ও হাসপাতাল						২৮		২৭	১
২৭			১৪			১৪	৫৫৭		৪৭৩	৪৪
২৮	আন্দোলক হাসপাতাল মিরপুর						৩৫২		৩৫৩	২২
২৯	টিটি হাসপাতাল ঢাকা						২৯২		২৮৫	
৩০	ইন্টেন্ট গাস্ট্রোলিভার ও জেনারেল হাসপাতাল						১৯		১৯	০
৩১	ইসলামী ব্যাংক হাসপাতাল মিরপুর						২৫৩		২৩২	২০
৩২	মাতৃত্ব মেডিকেল কলেজ হাসপাতাল						২৫৩		২৩৩	২০
৩৩	ইন্টেলিমায় স্পেশালাইজড	লি.					৩৯২		৩৭১	২০
৩৪		ঢাকা					১		০	০
	বেসরকারী হাসপাতালে মোট:		৩৬০			৩৬০	৩১৩৬	১৬১	২৯৯৮	১৮৩৭
	সর্বমোট:		৮৪৯			৮৪৯	৭৪৯৭	৫৭৮	৭৫৫৮	৩৮১৪

Use of LLMs for PDF scraping

LLMs may represent a novel method of PDF scraping. This approach would ideally remove the need for custom code, and allow for non-coding users to perform PDF scraping.

The number of LLMs that are available are increasing rapidly - however, most cannot accept file uploads and execute code. For this reason, we largely focus on OpenAI Advanced Data Analytics.

OpenAI advanced data analytics

OpenAI advanced data analytics is a beta feature for GPT-4. At the moment, to use this requires the use of ChatGPT Plus (20USD/month), and is only possible in-browser. This feature accepts file uploads and can execute python code in-browser. It can also export CSV extracts from PDF files.

We tested several approaches with this feature:

1. A simple text-based table was uploaded, and we asked for an extract of the data to csv. Note this table could be successfully pasted into excel. This was tried twice - in one instance, the table was extracted very poorly, and once with some of the columns retained. Performance may have been improved by specifying what columns were in the table.
2. We supplied an image of the same table, and asked for an OCR-based extraction of the table. Again, this led to inconsistent results, where numbers were occasionally misread, and not all columns were captured. Decimals were particularly hard to read, and in some cases were ignored.

3. We supplied a simple line chart within a PDF, and asked for an extraction of the underlying values in the chart. When doing so, values of axes were misidentified, and not all data values were captured.

Overall, GPT-4 currently does not appear to be sufficiently capable of extracting data from PDFs even in simple cases. Ultimately, the python code executed by GPT-4 does not appear tailored to the specific context of the PDF being extracted, which makes it unreliable. There may be some scope to use GPT-4 or other LLMs as a tool for learning code to extract PDFs, especially if the end user is proficient in python.

Other considerations

Use of OpenAI API

For ideal integration into a pipeline, PDF extraction via LLMs could be called via an API. OpenAI currently offers an API function, although it does not currently allow for advanced data analytics to be run, and therefore cannot execute python code. Use of the API requires use of paid tokens separate from the GPT Plus account, although individual API calls cost 0.01-0.03USD.

Privacy considerations

The majority of LLMs collect and store data to train their models. While this is likely not a problem for publicly available PDFs, in many cases data shared via PDF are not public, and therefore should not be used where privacy concerns are present. Organizational policies may also prevent the use of LLMs in everyday work.

Alternative models

While LLMs tested do not appear to be capable of extracting data from PDFs, close attention should be paid to the developing landscape of available models.

Conclusion and way forwards

Contribution

To continue exploring and testing additional new solutions for PDF scraping, we recommend that users follow the following approach:

· Identify the specific PDF documents or datasets you want to analyze. Consider using publicly available PDFs or documents with varying degrees of complexity to assess the performance of new methods.
· Define clear metrics and criteria for evaluating the effectiveness of the solution (eg: data extraction accuracy, processing speed, and the ability to handle different PDF layouts and formats).
· Test the solution on a sample of target PDFs to have an initial assessment of its performance and any potential challenges or limitations.
· Establish a baseline for comparison, which may include existing PDF scraping tools or methods, and assess how the new method compares in terms of accuracy, efficiency, and reliability.
· Test the solution on a sample of target PDFs to have an initial assessment of its performance and any potential challenges or limitations.
· Document your findings, including a detailed description of the new method, its strengths, weaknesses, and any modifications made during testing. Provide clear instructions for its implementation.
· Share your findings and insights with the broader community, such as through blog posts, or open-source contributions. Collaboration with others in the field can lead to further refinements and improvements.

XXX