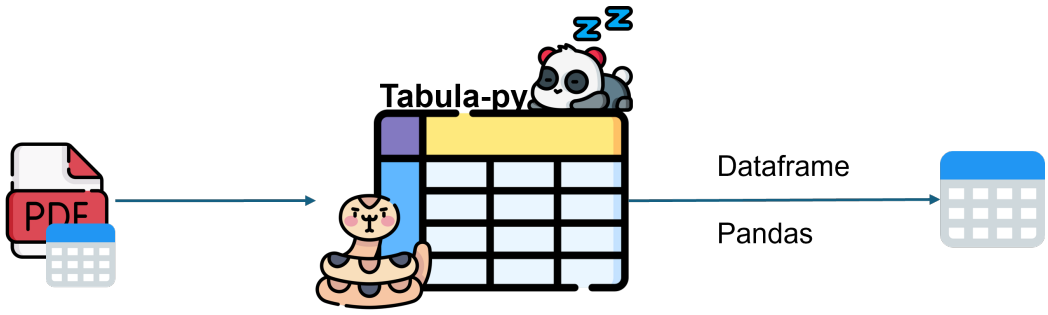
THỰC HÀNH TRÍCH XUẤT BẢNG TRONG PDF VỚI TABULA-PY VÀ PANDAS

Dinh-Tiem Nguyen và Quang-Vinh Dinh

1 Mở đầu

PDF là một trong những định dạng tài liệu phổ biến nhất được sử dụng trên Internet và trong nhiều lĩnh vực khác nhau. Trích xuất dữ liệu từ các tài liệu PDF thường gặp phải nhiều khó khăn, đặc biệt là đối với các bảng dữ liệu. Tiếp nối bài viết "Xử Lý Tệp PDF Đơn Giản Với PYPDF", trong bài viết này chúng ta sẽ tìm hiểu về tabula-py - một thư viện Python hỗ trợ trích xuất bảng từ PDF, đồng thời kết hợp nó với thư viện pandas để xử lý dữ liệu trích xuất.
Yêu cầu:

- Máy tính đã cài đặt Python >=3.8, Java 8+
- Biết lập trình Python, Pandas cơ bản.



2 Cài đặt Tabula-py

Tabula-py là một thư viện Python mã nguồn mở với chức năng chính là trích xuất dữ liệu từ các bảng trong tài liệu PDF. Dữ liệu được trích xuất bằng tabula-py có thể dễ dàng được chuyển đổi thành các cấu trúc dữ liệu của pandas để tiếp tục xử lý và phân tích. Để cài đặt thư viện này, ta sử dụng lệnh sau:

```
pip install tabula-py
```

Vì thư viện này yêu cầu cài đặt java 8+ để xử lý, nên chúng ta cần phải cài đặt nó. Đối với google colab đã được cài đặt sẵn java nên chúng ta bỏ qua bước này. Đối với máy tính window, chúng ta tải java tại đây, sau đó tiến hành mở file và cài đặt theo các tùy chọn mặc định. Sau khi quá trình cài đặt hoàn tất, ta cần thêm đường dẫn của java vào Environment Variables của window theo các bước dưới đây:

- Truy cập vào thư mục java đã cài đặt trên máy và sao chép đường dẫn của thư mục này `C:\Program Files (x86)\Java\jre-1.8\bin`
- Truy cập vào Environment Variables và thêm đường dẫn : Control Panel -> System and Security -> System -> Advanced System Settings -> Environment Variables -> Select PATH –> Edit

Sau khi thiết lập hoàn tất, chúng ta có thể kiểm tra xem thư viện đã cài đặt thành công chưa thông qua lệnh sau:

```
import tabula
!java -version
```

```
====================================== Output ======================================
java version "1.8.0_411"
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)
Java HotSpot(TM) Client VM (build 25.411-b09, mixed mode, sharing)
====================================================================================
```

Nếu bạn nhận được kết quả tương tự như trên tức là quá trình cài đặt đã thành công và có thể bắt đầu sử dụng. Nếu quá trình cài đặt này quá phức tạp với bạn, hãy dùng google colab.

3 Trích xuất bảng với Tabula-py

Trong tài liệu pdf, có rất nhiều định dạng bảng khác nhau. Chúng ta sẽ tìm hiểu cách trích xuất các loại bảng phổ biến nhất như: Bảng với cấu trúc rõ ràng, bảng không có đường viền...

3.1 Bảng với cấu trúc rõ ràng

Bảng có cấu trúc rõ ràng là bảng có đường viền rõ ràng xung quanh cả bảng và các ô bên trong, giúp dễ dàng nhận diện và trích xuất. Chúng thường được tạo ra từ các ứng dụng như Microsoft Word hoặc Excel trước khi chuyển đổi sang định dạng PDF.

Bảng 1: Thông tin Học Sinh

Tên	**Lớp**	**Điểm Trung Bình**
Anh	10A	8.5
Bình	10B	7.2
Châu	10C	9.1
Dũng	10A	6.8

Bảng trên có cấu trúc rõ ràng, ta sẽ thực hiện trích xuất bảng này:

```
import tabula
pdf_path = "https://github.com/NguyenDinhTiem/tabula-py-examples/raw/main/data/tables.pdf"

dfs = tabula.read_pdf(pdf_path, pages="1", lattice=True)
print(len(dfs))
dfs[0]
```

Trong chương trình trên, chúng ta bắt đầu bằng việc khai báo thư viện tabula, sau đó tiến hành trích xuất bảng tại trang 1 trong tệp pdf. Chúng ta sẽ sử dụng hàm read_pdf trong đó:

- pdf_path là đường dẫn đến tệp pdf
- pages là tham số xác định trang nào trong tệp PDF sẽ được trích xuất. Trong ví dụ trên, pages="1"chỉ định rằng chỉ trang đầu tiên của tệp PDF sẽ được xem xét để trích xuất dữ liệu. Ta cũng có thể chỉ định nhiều trang cụ thể bằng cách sử dụng một chuỗi như "1,2,3"hoặc một phạm vi trang như "1-3". Để trích xuất tất cả các trangta có thể sử dụng pages="all".

- lattice là một tham số boolean cho biết liệu hàm có nên sử dụng chế độ "lattice"để trích xuất bảng hay không. Khi lattice=True, tabula-py sẽ cố gắng phát hiện các đường kẻ bảng (cả ngang và dọc) để xác định chính xác ranh giới của các ô trong bảng.

Kết quả trả về từ tabula.read_pdf là một danh sách chứa các dataframe. Trong ví dụ trên kết quả đầu ra là danh sách chứa 1 dataframe, chúng ta có thể truy cập bằng chỉ mục của danh sách, ví dụ dfs[0].

3.2 Bảng không có đường viền

Bảng không có đường viền là loại bảng không có đường viền xác định rõ ràng giữa các hàng và cột, chỉ dựa vào khoảng cách để phân chia các ô. Việc trích xuất từ các bảng này có thể khó khăn hơn do thiếu các dấu hiệu nhận biết.

Bảng 2: Thông tin khách hàng

Tên	Tuổi	Thành phố
An	22	Hà Nội
Cúc	25	TP HCM
Hoa	20	Đà Nẵng
Nụ	23	Cần Thơ

Bảng trên là một ví dụ điển hình cho bảng không có đường viền, chúng ta sẽ thực hiện trích xuất bảng này từ tệp pdf.

```
import tabula
pdf_path = "https://github.com/NguyenDinhTiem/tabula-py-examples/raw/main/data/tables.pdf"

dfs = tabula.read_pdf(pdf_path, pages="2", stream=True)
print(len(dfs))
dfs[0]
```

Trong chương trình trên, ta sử dụng tabula.read_pdf để trích xuất bảng tại trang 2 của tệp pdf. Đối với loại bảng này chúng ta sử dụng tham số stream = True, tabula-py sẽ phân tích trang PDF dựa trên cấu trúc các hàng và khoảng trống giữa chúng để xác định ranh giới.

3.3 Bảng với cấu trúc phức tạp

Bảng có cấu trúc phức tạp là bảng có hàng hoặc cột được gộp lại, chứa các bảng con, hoặc các bảng chứa nhiều cấp độ của thông tin. Các bảng này thường xuất hiện trong các tài liệu kỹ thuật, tài chính hoặc khoa học. Việc trích xuất dữ liệu từ chúng khó hơn các loại bảng thông thường, đòi hỏi chúng ta phải xử dụng các kỹ thuật trong pandas để xử lý đầu ra.

Bảng 3: Lịch Thi Cuối Kỳ

Môn học	Ngày thi	Giờ bắt đầu	Phòng thi
Toán	20/12/2024	08:00	101A
	21/12/2024	09:00	102B
Lý	22/12/2024	08:00	101A
		10:00	102B
Hóa	23/12/2024	08:00	101A
		10:00	102B

Bảng trên có cấu trúc khá phức tạp, có những ô được gộp lại dẫn đến việc trích xuất đòi hỏi chúng ta phải thử các tham số khác nhau và phải xử lý đầu ra với pandas.
Đầu tiên, để trích xuất bảng này từ file pdf, ta sẽ thử các tham số khác nhau để thu được kết quả output phù hợp. Trong ví dụ này, ta sẽ sử dụng stream=True thay vì lattice=True, mặc dù bảng này cũng được bao bởi các đường kẻ, tuy nhiên trong trường hợp này sử dụng lattice sẽ không thu được kết quả mong muốn.

```
import tabula
pdf_path = "https://github.com/NguyenDinhTiem/tabula-py-examples/raw/main/data/tables.pdf"
dfs = tabula.read_pdf(pdf_path, pages="3", stream=True)
```

Bảng 4: Dataframe trích xuất từ bảng trong tệp pdf

#	Môn học	Ngày thi	Giờ bắt đầu	Phòng thi
0	*NaN*	20/12/2024	08:00	101A
1	Toán	*NaN*	*NaN*	*NaN*
2	*NaN*	21/12/2024	09:00	102B
3	*NaN*	*NaN*	08:00	101A
4	Lý	22/12/2024	*NaN*	*NaN*
5	*NaN*	*NaN*	10:00	102B
6	*NaN*	*NaN*	08:00	101A
7	Hóa	23/12/2024	*NaN*	*NaN*
8	*NaN*	*NaN*	10:00	102B

Tiếp theo chúng ta cần xử lí dataframe thu được, nhận thấy dữ liệu của môn Toán nằm rải rác ở 3 dòng đầu, môn Lí nằm ở 3 dòng tiếp theo, môn Hóa nằm ở 3 dòng cuối cùng. Ta sẽ thực hiện lấp đầy các ô có giá trị NaN ở dòng phía sau bởi các giá trị của các ô ở dòng trước đó.

```
# Điền các giá trị NaN bằng các giá trị từ hàng trước đó
df['Môn học'].fillna(method='ffill', inplace=True)
df['Ngày thi'].fillna(method='ffill', inplace=True)
df['Giờ bắt đầu'].fillna(method='ffill', inplace=True)
df['Phòng thi'].fillna(method='ffill', inplace=True)
```

Bảng 5: Dataframe sau khi đã fillna

#	Môn học	Ngày thi	Giờ bắt đầu	Phòng thi
0	*NaN*	20/12/2024	08:00	101A
1	Toán	20/12/2024	08:00	101A
2	Toán	21/12/2024	09:00	102B
3	Toán	21/12/2024	08:00	101A
4	Lý	22/12/2024	08:00	101A
5	Lý	22/12/2024	10:00	102B
6	Lý	22/12/2024	08:00	101A
7	Hóa	23/12/2024	08:00	101A
8	Hóa	23/12/2024	10:00	102B

Sau khi lấp đầy các ô NaN, chúng ta nhận thấy, các giá trị ở hàng 0, 3, 6 không cần thiết, ta sẽ loại bỏ chúng:

```
df_cleaned = df.drop([0, 3, 6])
```

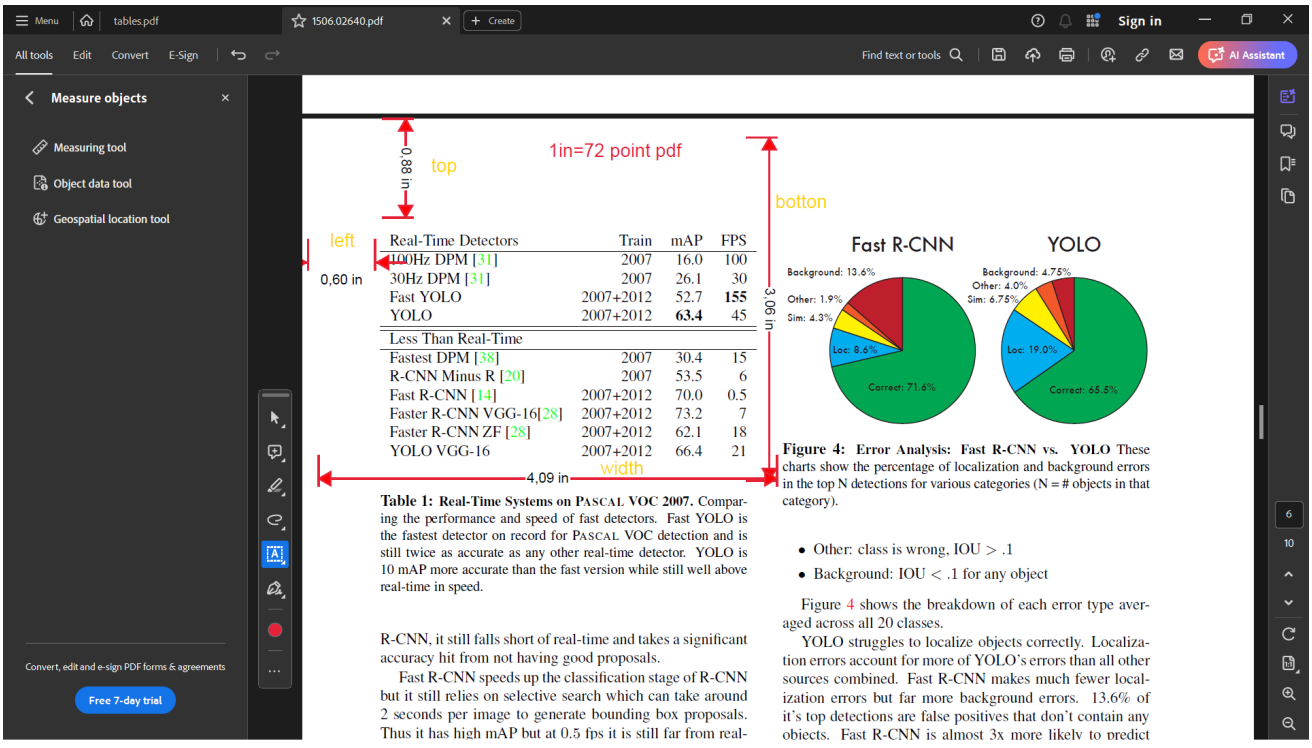
Và cuối cùng, chúng ta sẽ thu được dataframe được làm sạch.

Bảng 6: Dataframe trích xuất từ bảng trong tệp pdf

#	Môn học	Ngày thi	Giờ bắt đầu	Phòng thi
1	Toán	20/12/2024	08:00	101A
2	Toán	21/12/2024	09:00	102B
4	Lý	22/12/2024	08:00	101A
5	Lý	22/12/2024	10:00	102B
7	Hóa	23/12/2024	08:00	101A
8	Hóa	23/12/2024	10:00	102B

3.4 Bảng với vị trí được chỉ định

Trong các tài liệu pdf, đặc biệt là các bài báo khoa học, các bảng thường rất phức tạp và ở nhiều các vị trí khác nhau, thư viện tabula-py thường cho kết quả trích xuất bảng rất tệ trong những tài liệu dạng này. Tuy nhiên có một số trường hợp có thể khắc phục bằng cách yêu cầu tabula-py chỉ thực hiện trích xuất bảng tại vị trí được chỉ định trên trang.
Đây là cách làm thủ công và khá tốn sức, đầu tiên chúng ta cần một công cụ để xác định vị trí của bảng trong tệp tin pdf. Trong bài viết này, chúng ta sẽ sử dụng phần mềm Adobe reader, bạn có thể tải và cài đặt phần mềm tại đây.



Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

R-CNN, it still falls short of real-time and takes a significant accuracy hit from not having good proposals.

Fast R-CNN speeds up the classification stage of R-CNN but it still relies on selective search which can take around 2 seconds per image to generate bounding box proposals. Thus it has high mAP but at 0.5 fps it is still far from real-

Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

- Other: class is wrong, IOU > .1
- Background: IOU < .1 for any object

Figure 4 shows the breakdown of each error type averaged across all 20 classes.

YOLO struggles to localize objects correctly. Localization errors account for more of YOLO's errors than all other sources combined. Fast R-CNN makes much fewer localization errors but far more background errors. 13.6% of it's top detections are false positives that don't contain any objects. Fast R-CNN is almost 3x more likely to predict

Hình 1: Xác định vị trí bảng trong file pdf

Chúng ta sẽ thực hiện ví dụ với bài báo YOLO. Vị trí của bảng chúng ta cần xác định là vị trí của 4 điểm:

- top: Vị trí trên cùng của bảng
- left: Vị trí cạnh trái của bảng
- width: Vị ví cạnh phải của bảng
- botton: Vị trí cạnh dưới cùng của bảng

Chúng ta sẽ sử dụng công cụ mesuaring tool trong phần mềm Adobe reader để đo khoảng cách từ các cạnh của tài liệu đến các vị trí các cạnh của bảng tương ứng. Khoảng cách này được tính theo giá trị inch, để chuyển đổi sang giá trị điểm trong pdf, chúng ta sẽ thực hiện theo phép chuyển đổi: **1 inch = 72 point pdf**

```
top = 0.88
left = 0.6
botton = 3.06
width = 4.09

loc = [i*72 for i in [top, left, botton,
    width]]
print(loc)
```

```
================= Output ================
[63.36, 43.199999999999996, 220.32,
    294.48]




=========================================
```

Sau khi xác định được vị trí của bảng, chúng ta thực hiện trích xuất bảng:

```
import tabula
pdf_path = "https://arxiv.org/pdf/1506.02640.pdf"
dfs = tabula.read_pdf(pdf_path, pages="6", area = loc)
```

Trong chương trình trên, chúng ta sử dụng hàm tabula.read_pdf để trích xuất bảng tại trang 6 và area=loc là vị trí mà chúng ta đã đo phía trên. Dưới đây là kết quả bảng trích xuất được:

Bảng 7: Comparison of Real-Time Detectors

Index	Real-Time Detectors	Train	mAP	FPS
0	100Hz DPM [31]	2007	16.0	100.0
1	30Hz DPM [31]	2007	26.1	30.0
2	Fast YOLO	2007+2012	52.7	155.0
3	YOLO	2007+2012	63.4	45.0
4	Less Than Real-Time	NaN	NaN	NaN
5	Fastest DPM [38]	2007	30.4	15.0
6	R-CNN Minus R [20]	2007	53.5	6.0
7	Fast R-CNN [14]	2007+2012	70.0	0.5
8	Faster R-CNN VGG-16 [28]	2007+2012	73.2	7.0
9	Faster R-CNN ZF [28]	2007+2012	62.1	18.0
10	YOLO VGG-16	2007+2012	66.4	21.0

4 Kết Luận

Trong bài viết này, chúng ta đã tìm hiểu và sử dụng thư viện tabula-py để trích xuất dữ liệu bảng từ các tài liệu PDF và cách làm sạch, xử lý dữ liệu với pandas. Chúng ta đã đi qua từng bước từ việc cài đặt thư viện, sử dụng các phương thức trích xuất bảng, cho đến các kỹ thuật cơ bản để làm sạch dữ liệu. Mặc dù tabula-py còn khá nhiều hạn chế đối với cấu trúc bảng phức tạp tuy nhiên nó cũng có nhiều ưu điểm đáng để ta ưu tiên sử dụng khi trích xuất bảng trong pdf.