**Clause Extraction Synopsis**

I have created a basic AI application for extracting clauses/ classes from the pdf at the page level.

The details of the 5 sub folders are as below:

1. **data:**

Contains raw pdf files downloaded from public sources. This folder also contains the code to divide the pdfs into train & test folders for model training and for testing the built solution

1. **ocr:**

Contains the code to extract the text from the pdf files

1. **ml:**

Contains the code: **a)** to generate the training data (csv format at page level) from the text files created from OCR process and, **b)** to train the ML classification model for detecting one of the 4 types [**contract, legistration, public\_health, mech\_eng**]

1. **new\_file:**

Contains the code to make classification prediction on each page of a pdf present in this folder

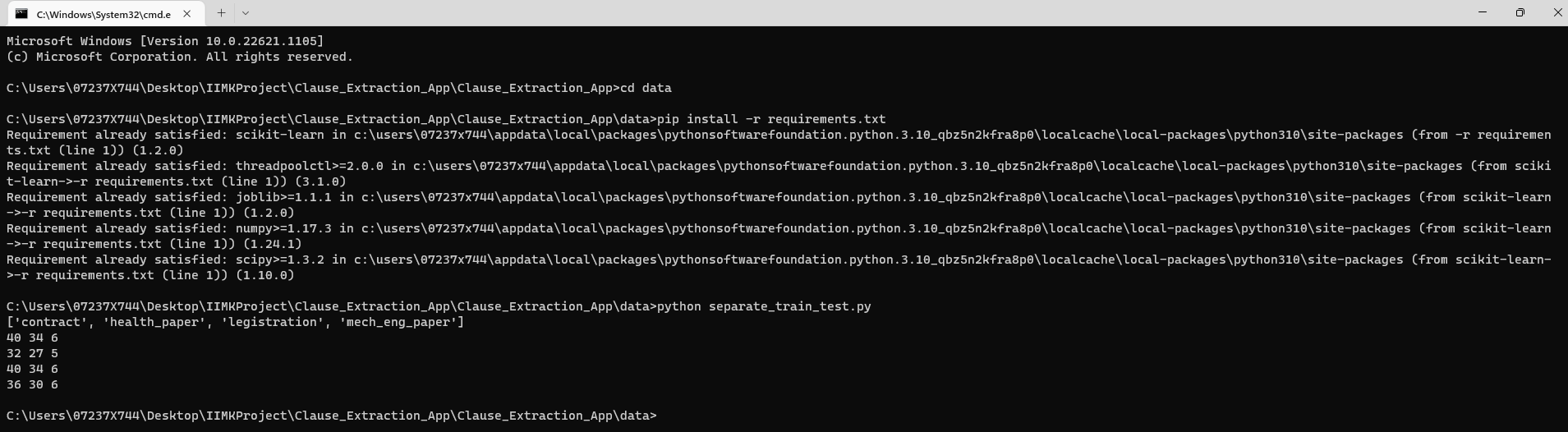
1. **web-app:**

Contains a basic Django web-application of two pages [upload file, view uploaded files]

Each folder has a readme text file further detailing the contents of those folders.

**Steps to run the end-to-end solution from the main “Clause\_Extraction\_App” folder:**

1. **[Folder: data] Create the train & test data split from the raw pdf files**
   * Run the commands given in “Clause\_Extraction\_App/data/readme.txt” to execute the data folder code as shown below



* This would create two folders (train & test) inside data folder as shown below

**Graphical user interface, application

Description automatically generated**

1. **[Folder: ocr] Extract the text out of the pdfs created in step #1**
   * Download poppler library for pdf handling from <https://github.com/oschwartz10612/poppler-windows/releases/>
   * Download tesseract for windows exe from <https://github.com/UB-Mannheim/tesseract/wiki> and install
   * Change the path to downloaded poppler library bin folder in read\_pdf.py file. Also, keep the path of where tesseract has been installed in read\_pdf.py (refer below image)

**Text

Description automatically generated**

* + Run the commands given in “Clause\_Extraction\_App/ocr/readme.txt” to execute the ocr folder code as shown below

**Text

Description automatically generated**

* + The above code will extract the text (ocr) out of all the files present in train folder of step #1

**Graphical user interface, application, table

Description automatically generated**

1. **[Folder: ml] Create the training data and ml model for predicting clause of a pdf page**
   * Run the commands given in “Clause\_Extraction\_App/ml/readme.txt” to execute the ml folder code as shown below

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

The below command creates the train.csv file which contains text of each page and the respective clause class to train the ml model

**Text

Description automatically generated**

Screenshot of the train.csv:

**Graphical user interface, table

Description automatically generated**

The below command runs the ml training on the train.csv and creates the models folder containing 2 models:

* **tfidf\_vectorizer.joblib**: this model converts the text into vector for ml training
* **clause\_clf.joblib**: this is the trained ml model for predicting the clause for the text vectors

**Text

Description automatically generated**

models folder:

**Graphical user interface, application

Description automatically generated**

The above command also displays the results of ML training, for example, in the above cmd screen you can see **the confusion matrix, precision, recall and f1-score** of the training.

Contract, legistration, health and mech are the 4 clause types on which we have trained the ml model.

1. **[Folder: web-app] Web application for user interaction, for uploading contract pdf**
   * Run the commands given in “Clause\_Extraction\_App/web-app/readme.txt” to execute the web-app folder code as shown below

**Text

Description automatically generated**

**Text

Description automatically generated**

The above commands run the web server at localhost. To open web app enter <http://localhost:8080/> in browser

**Graphical user interface, text, application

Description automatically generated**

Click on “Index new Contracts” to upload your pdf

**Graphical user interface, text, application

Description automatically generated**

Now, from the home page (<http://localhost:8080/>) you can click on “See Indexed Contacts” to view the file just uploaded

**Graphical user interface, application

Description automatically generated**

The file just uploaded is saved inside “\web-app\media\crt\_pdfs” folder

**Table

Description automatically generated with medium confidence**

This is only a simple we app. In production, the raw pdf files present in the data folder can directly be uploaded through web app by users, instead of downloading from internet

1. **[Folder: new\_file] To predict the page level clauses of any pdf file**

We must first save the pdf file in this folder and run the commands given in “Clause\_Extraction\_App/new\_file/readme.txt”. The predict\_clause.py uses the pickled files (tfidf\_vectorizer.joblib, clause\_clf.joblib) created during ml training in step 3.

Change the tesseract and poppler paths in the predict\_clause.py as done in step 2

**Text, application

Description automatically generated**

**Text

Description automatically generated**

The predict\_clause.py file predicted the page level clause for the contract\_HNBA-2017-18-Confide.pdf file present inside the folder.

The output is shown on the command prompt as well as saved in out.csv file. As the main file had 4 pages and it was a contract pdf, the output predicts each page of the pdf as contract clause.

**Future Plans:**

**I plan to learn Cloud Application Development soon, so that all the 5 solutions can be deployed on cloud. By deploying this solution on cloud, the application will be available to anyone on internet & these sub-modules can interact with themselves automatically**