Automated Redaction Tool for Protected Health Information

Break Through Tech AI Program RubiconMD Team

Linqing Zhu
(Section VC1C 5545)
linqing201@gmail.com

CISC 4900
Fall 2022

Technical Supervisor
Christian Lewis
Director of Data Science @ RubiconMD
christian@rubiconmd.com

Table of Contents

Abstract

Patients upload medical documents to the RubiconMD platform, then RubiconMD team needs to remove the patients' information on the documents before sending those documents to specialists for review. RubiconMD team currently hires people to do redaction manually on medical documents. In this project, we want to make an automated redaction tool to remove patients' information. It allows RubiconMD to provide more efficient healthcare to the patients of the doctors who use their system. By using an automated redaction tool solution, providers can access patient records faster, ultimately decreasing the time between a patient's initial appointment and their consultation with the specialist. Further, this project will potentially reduce the cost and prevent the unnecessary cognitive load associated with manual redaction allowing efforts to be focused elsewhere.

Project Details

Purpose of Project

The Health Insurance Portability and Accountability Act of 1996 (HIPAA) is a federal law that required the creation of national standards to protect sensitive patient health information from being disclosed without the patient's consent or knowledge. To protect sensitive patient health information, the RubiconMD team needs to do redaction on documents from patients. The RubiconMD team has a specific team to redact patient health information manually by looking at the documents. The process of redacting takes a lot of time where the employee needs to check on each file and identify patient health information, download the files, and use the Adobe file editor draw box to cover the patient health information. In this project, we build an automated redaction tool to accomplish the redaction process in the RubiconMD system so it will save time for patients waiting on the process and the company process documents.

Project Description

Based on the RubiconMD system, we expect to do a redaction on PDF and image files. The input files are medical documents, such as surgical reports, x-rays, and EKGS, from attachments in any given form (pdf, jpeg, png, etc.). This project builds a workflow that accomplishes the reduction process by using machine learning algorithms. The program converts images to text, analysis text, identifies the protected information in text, locates the words in the images, and draws boxes on the located words. It outputs a PDF file like the input file with redacted information. There is no such technology that can do the whole process of redaction and able to attach with the RubiconMD Amazon Web Service system, but each part we want to can accomplish with existent technologies. The optical character recognition model (OCR) is able to convert images to text. There is an existing machine learning algorithm that can be used to detect protected health information where we can get the output from this algorithm to do a redaction. The outcome of our project is compatible with the organization's system like Amazon Web Services. Organizations like RubiconMD will not need to hire people to redact medical documents manually where they can just employ this project into their system to do redaction. The workflow is fully built by our team now, and it is able to do redaction on PDFs and images.

In the future, we need to check on the accuracy of the algorithm and outcome to do more development on the models.

Executive Summary

Problem Statement

The PHI acronym stands for protected health information, also known as HIPAA data. The Health Insurance Portability and Accountability Act (HIPAA) mandates that PHI in healthcare must be safeguarded. PHI is any piece of information in an individual's medical record that was created, used, or disclosed during the course of diagnosis or treatment that can be used to personally identify them. HIPAA protected health information (PHI) are name, address (including subdivisions smaller than states such as a street address, city, county, or zip code), any dates (except years) that are directly related to an individual, including birthday, date of admission or discharge, date of death, or the exact age of individuals older than 89, telephone number, fax number, email address, social security number, medical record number, health plan beneficiary number, account number, certificate/license number, vehicle identifiers, serial numbers, or license plate numbers, device identifiers or serial numbers, web URLs, IP address, biometric identifiers such as fingerprints or voice prints, full-face photos, and any other unique identifying numbers, characteristics, or codes. Healthcare organizations usually hire people to remove protected health information manually which takes lots of time and labor work to accomplish.

Proposed Solution

Our project will build a workflow to help remove the protected health information inside the medical documents. So it will save time and labor work on doing the redaction process. We will accept the medical files, process the files in programming code, identify the protected health information using the existing model in data science, redact protected health information using programming code in the files then output the same type of file with redacted protected health information.

Value

The automated redaction tool for protected health information will be beneficial for all healthcare organizations who want to remove protected health information from medical documents. It will save time and money for healthcare organizations to process documents. It will be also beneficial to patients in the medical system. Patients and organizations will not be worried about the disposal of protected health information with the employment of this automated redaction workflow in the healthcare system. Our project would directly affect the E-consultation market which is concerned with making medical knowledge more accessible to people who need them. One key objective of this market is to deliver healthcare faster to people in need. By automating the redaction process to safeguard privacy, our project fosters this key objective. This machine learning algorithm may be used outside the healthcare field as well.

Technical Overview

Technical Description

In this project, we analyze text to determine if the text is protected health information (PHI), the input files we have are all image and pdf files. So we use optical character recognition (OCR) to convert an image of text into a machine-readable text format. Optical Character Recognition (OCR) models are the process of recognizing characters from images using computer vision and machine learning. Image preprocessing is to do processes on images like grayscale and brightness corrections in order to increase the performance of OCR models. The OCR models we tried are two open-source OCR Pytesseract and EasyOCR.

To pre-process text, we use the python Natural Language Toolkit (NLTK) to remove unnecessary words like stop words. A stop word is a commonly used word (such as “the”, “a”, “an”, or “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We also need tokenization which is breaking the raw text into small chunks. Tokenization breaks the raw text into words, and sentences called tokens. A more step in processing text data is to do stemming and lemmatization where it can do the text normalization technique used in Natural Language Processing (NLP), which switches any kind of word to its base root mode. Stopwords, punkt, and tokenize are the python libraries in the Natural Language Toolkit (NLTK).

To find protected health information (PHI) in the text, we used Amazon Comprehend Medical to detect useful information in unstructured clinical text. Amazon Comprehend Medical is a HIPAA-eligible natural language processing (NLP) service that uses machine learning that has been pre-trained to understand and extract health data from medical text, such as prescriptions, procedures, or diagnoses. Amazon Comprehend Medical uses Natural Language Processing (NLP) models to sort through enormous quantities of data for valuable information gained through advances in machine learning.

We need to use Python DataFrame to locate each text in order to cover the PHI in the image/pdf. The output from the OCR with left, top, width, and height columns where are image located of each word will retain as a DataFrame and then pre-process using NLTK, the clean text will be processed by the Amazon Comprehend Medical with output a list of words is protected health information. We located and draw a rectangle box on those words and output the image as a PDF file.

To test our workflow, RubiconMD provided a small sample of data has images and PDF files. In addition to the provided sample data, we will also be using Kaggle to find additional testing data. We will need to do research on OCR models and find the right one.

Background

Before this project I don’t know anything about optical character recognition (OCR) and natural language processing (NLP). But I know python for data science to build models and Google Colab notebook to run python code. In this project, we need to connect Google Drive to Google Colab notebook in order to access the dataset in Google drive which is something I had done before. Before doing the project, my other teammates and I are all in the Break Through Tech AI program we learned machine learning models, Google Colab, and all other data science

techniques during the summer. But we all don't have experience with OCR, NLP, and Amazon Web Services. But we know how to use python to code and make projects.

We were matching different people into a group along with 18 different projects and I put this project on my top 3 projects list because I like to use programming to reduce machine work in daily life. Before this project, I had done projects on Google App Script to generate mass Google forms and send mass emails containing columns with color in the email body. I feel great about doing those projects so I can see this project will be another good one for me to work on.

Project Objectives and Expected Outcomes

Objectives

The objective of this project is to accept input files as PDFs/images, output the redacted protected health information inside the file then return the output file and store it in a file path. To achieve the text redaction in the image, we need to extract text from different kinds of medical documents, and determine if the medical documents have protected health information. Perform text redaction if there is protected health information in the medical documents. The test of the model will be conducted by the RubiconMD team to identify if the model is correctly reducing protected health information and determining protected health information in the documents. We need to test the text extract of the model and make sure the model highlights all the protected health information inside the documents. The output will be a file with the protected health information covered. The expectation of our challenge advisor is at least create a model that can determine if a file contains protected health information or not. We will know our project is a success when we can identify the text including the files and check the accuracy of the text. We will know if we can identify the meaning of each word that is protected information or not. Here we can check the accuracy of model identification. Lastly, we will measure the accuracy of the protected information to get reduced or not.

Expected Outcomes

Task (What will be done)	**Outcome** (Expected result of task)
Dataset preparation + library and environment setup	1. Generate a dataset of at least 200 examples 2. Each team member should have all necessary libraries and environments set up on their PCs. 3. Dataset should be uploaded to our shared google folder 4. The dataset should be uploaded to our Google Colab notebook
Data Preprocessing	● Enlarge our dataset by 150% using data augmentation techniques ● Data in a ready-to-use state for analysis
Text Exaction	● Completed Feature selection for text extraction ● Defined metrics for text extraction

	● Achieve 90% accuracy rate
Text Identification	● Prepare data for text identification ● Achieve 90% accuracy rate
Text redaction	1. Prepare data for redaction 2. Construct an initial model for redaction 3. Construct final model 4. Achieve 90%accurate
Clean up	● Complete any previous task ● Analyze work ● Look into deployment

Actual Outcomes

Describe what outcomes you have reached with your project. Were these outcomes the same as the expected outcomes? If not, why not?

Task Done	**Actual Outcomes**
Dataset preparation + library and environment setup	● Each team member had all necessary libraries and environments set up on their PCs. ● Dataset uploaded to shared google folder ● The dataset was accessed in Google Colab notebook
Data Preprocessing	● Enlarged our dataset by 150% using data augmentation techniques ● Data in a ready-to-use state for analysis
Text Exaction	● Completed Feature selection for text extraction ● Defined metrics for text extraction
Text Identification	● Prepared data for text identification
Text redaction	● Prepared data for redaction ● Constructed an initial model for redaction
Clean up	● Combined work of redaction progress and also store path ● Processed file input workflow and output file workflow

Obstacles & Risks

The primary ethical concern is disclosing Protected Health Information (PHI) in an attempt to make a model that removes PHI. We don't have access to any PHI from the company in this project, our testing and deployments will run by the RubiconMD. In this project, we are not able to access real medical documents from RubiconMD because those are protected health information. It will be a big challenge for us that we may not able to know if our workflow can process well with the real-life case.

Considering the input files of PDFs and images is a hard problem for us when I was asking the RubicoMD team if we can classify all different kinds of images or files, they told me that we are expecting all different kinds of images and files, so we are not able to do further optimization on the pre-processing. So we tried image processing and decision-making based on the confidence level of the OCR models.

We need to take care of the Amazon Web Service since the RubiconMD uses Amazon Web Service as the backend for storing files and further employment of the code. The part we want to detect protected health information will be done by AWS, and it will be charged by characters. We don’t want to pay lots of money for this project, so we need to reduce as much of characters as possible and then pay to test check if our model works.

Another obstacle is we don’t have time to verify the outcome and the accuracy of our work on this project. In the further deployment of the workflow, we want to see the feedback of the redaction team at RubiconMD to do further optimization.

From the aspect of the project, we were really don’t know how to solve this problem in the beginning and now we had built a workable solution is so impressive. In this experience, I learned all the different technologies to work toward solving one problem and working in a team to resolve conflicts. One key skill I learned is communication I always talked my ideas out then we discussed the problems.

Technology Requirements

Here are details technology we used in the project.

- Google Colab:
    - To write python functions, run python codes, connect Google Drive to access the PDFs/images in the drive, store the PDFs/image into the output folder path, install package of python libraries, and connect to AWS Comprehend Medical Detect PII to get the protected health information list.
    - To download/update features in Google Colab, you need to use !pip command for python packages.
- Python:
    - Python Library:

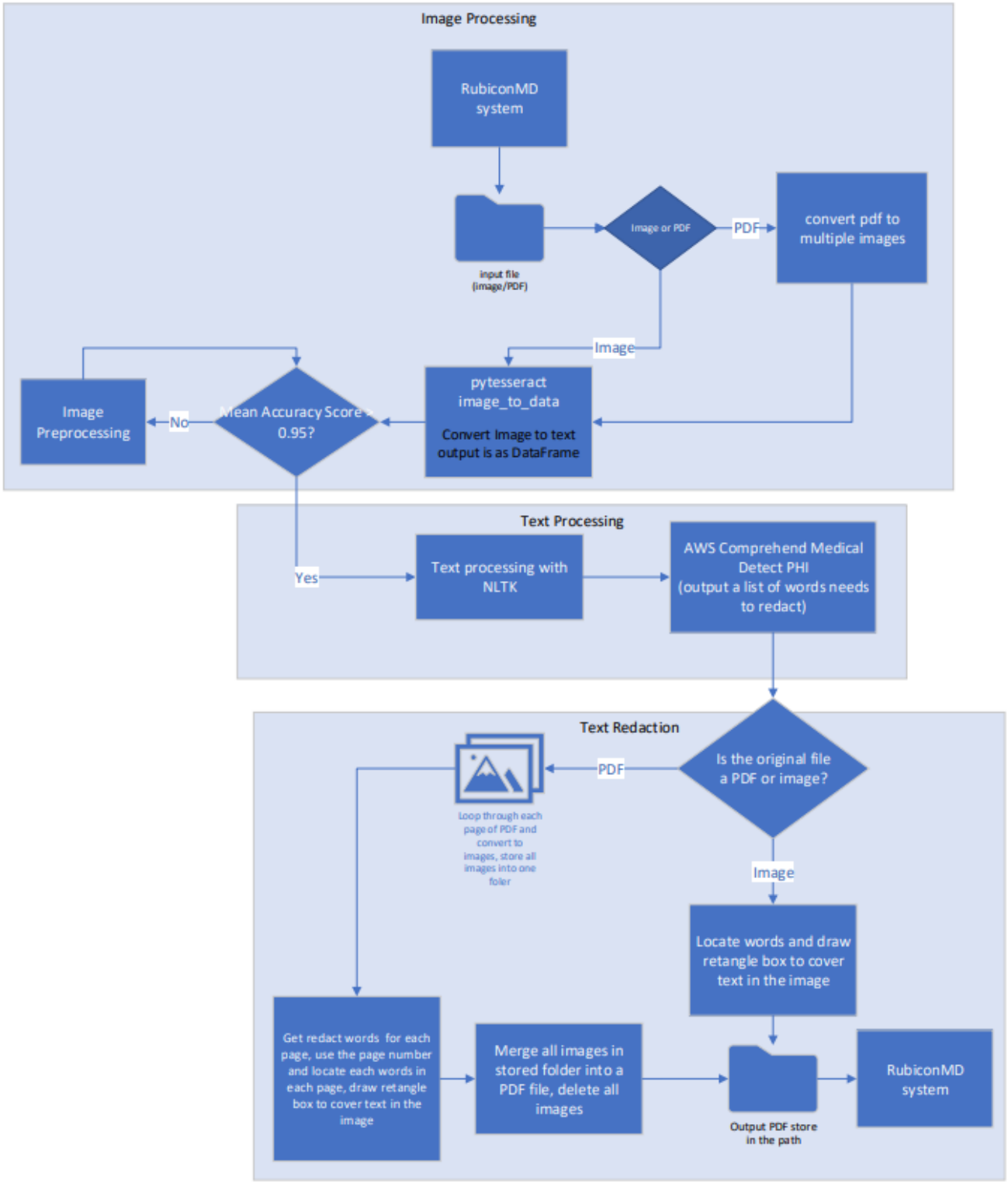
        - import cv2 as cv: an open-source library for computer vision, machine learning, and image processing, and now it plays a major role in real-time operation
        - import numpy as np: open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
        - import PIL, PIL.Image and PIL.Image: Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language.
        - IPython.display: display image or files: interface with a computer and view stored, generated, or transmitted data in the form of text and graphics.
        - import re: A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).
        - import pandas as pd: Pandas Series is a one-dimensional labeled array whose object size cannot be changed. You can also see it as the primary building block for a DataFrame, making up its rows and columns.
        - import os: allows us to run a command in the Python script, just like if I was running it in my shell.
    - Python Package used:
        - !pip install pdf2image: convert_from_path, convert_from_bytes, convert the file from a path into images, wrapper around the pdftoppm and pdftocairo command line tools to convert PDF to a PIL Image list.
- Optical Character Recognition:
    - !pip install pytesseract, import image_to_data, Output: recognize and “read” the text embedded in images. It can read and recognize text in images and is commonly used in python OCR image to text use cases.
- Natural Language Processing using the NLTK python package’s modules:
    - from nltk.tokenize import WordPunctTokenizer: split a sentence into tokens or words, sent_tokenize() method to split a document or paragraph into sentences.
    - from nltk.corpus import stopwords: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. To check the list of stopwords you can type the following commands in the python shell.
    - from nltk.tokenize import word_tokenize: word_ tokenize is extremely important for pattern recognition and is used as a starting point for stemming and lemmatization. Nltk word_tokenize is used to extract tokens from a string of characters using the word tokenize method.
    - from nltk.stem import WordNetLemmatizer: NLTK Lemmatization is the process of grouping the inflected forms of a word in order to analyze them as a single word in linguistics. NLTK has different lemmatization algorithms and functions for using different lemma determinations.

- Amazon Web Services
    - Details about the Comprehend Medical we used https://aws.amazon.com/blogs/machine-learning/detecting-and-redacting-pii-using-amazon-comprehend/
    - To set up the environment on Google Colab and use the Amazon Comprehend Medical
        - !pip install presidio-image-redactor
        - !pip install botocore boto3
            - from botocore.vendored import requests

Software & Hardware Requirements

We used Google Colab and Amazon Web Services on Chrome and it requires at least 4 GB memory, a file size of 2GB, and at least an Intel Core i5 CPU. The programming language we use is Python which is working in Google Colab and we can download all different Python libraries in Google Colab.

Architecture



Milestones and reporting

Milestones	Progress	Status	Challenges	Next Step
Text extractions on images and PDF	Used Tesseract optical character recognition (OCR) and EasyOCR on the image and PDF extracts text from images and documents without a text layer and outputs the document into a new searchable text file, but the output of the text if not accurate in some part	Done	Not be able to get a 100% correctness on Tesseract OCR. EsayOCR can extract text better in some images but takes more time	Do image pre-processing to increase the confidence level of OCR models reading the image
Find out a way to measure the optical character recognition model's correctness on text extraction	Using the Python library cv2 to draw rectangle box on the image to determine which parts of text are not being read, measure each text box by checking the confident level	Done	The Tesseract OCR has -1 confident level text. EasyOCR is different from Tesseract OCR	Convert the output to DataFrame and locate each text
Draw a box to cover protected health information	Use the Python PIL library to draw a rectangle block based on the text extract result from OCR models	Done	The output of OCR models need to transform into DataFrame and then use different columns to locate on text block	Attach with the determine protected health information code to block protected health information
Natural Language Processing on the extracted text	Remove redundant text in the images	Done	Still have lots of words left to the next step	Move to outcome after natural language processing into the Amazon Web Services
Determine the	Using the AWS tools to	Done	Still need to	Continue to

protected health information inside the text	detect protected health information of the image text after Natural Language Processing		make sure if all the protected health infomation is being detected in the future	work on more datasets and may need to see the outcome from real life documents
Connect Google Colab with Amazon Website Service system	Connect our features with AWS so that we can process the text outcome to do protected health information detection after the natural language processing	Done	We need to pay for AWS to do text detection, so we minimize the words we need to test	Still need to follow up with the actual accuracy and outcome

Project Management

The project is managed by us and we report weekly with my supervisor. Each of us has a Google Colab notebook and we try on different parts and catch up with each other on the progress each week. At the final stage of the project, we collaborated and combined work together. Rojanaye is more like a team lead in that she worked on the project management of timeline catchup, and assignments due for the program, and also allocate meetings with RubiconMD, advisor, and team meetings. We also used Trello for the whole project management to keep track of what is need to be done and due.

Decisions Making

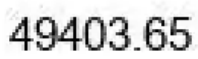
Which Optical Character Recognition (OCR) models should we choose?

In my research, I found lots of Optical Character Recognition (OCR) model exits. I tried those OCRs and found the main three good OCRs: Tesseract, EasyOCR, and Keras-OCR. Here are the pros and cons of each model.

- Tesseract OCR is performing well for high-resolution images. Certain morphological operations such as dilation, erosion, and OTSU binarization can help increase Tesseract OCR performance.
- EasyOCR is a lightweight model which is giving good performance for receipt or PDF conversion. It is giving more accurate results with organized texts like pdf files, receipts, and bills.
- Keras-OCR is image specific OCR tool. If text is inside the image and the fonts and colors are unorganized, Keras-OCR gives good results.

Based on of images and files we input we choose Tesseract OCR and EasyOCR. After lots of trying and reading images, we decided to move on with Tesseract OCR because EasyOCR talks lots of time even on a nice organized PDF image. Tesseract OCR runs better performance in the majority of our input images. The confidence level of the output text increased after we did image pre-processing.

In terms of speed of the two main OCR models Tessaerct OCR and EasyOCR we actually tried.
Input Data for CPU test:



Input Data for GPU test:

Below are the results:

	CPU	GPU
Tesseract	0.3 seconds/image	0.25 seconds/image
EasyOCR	0.82 seconds/image	0.07 seconds/image

In terms of speed, Tesseract outperforms EasyOCR on CPU, while EasyOCR performs amazingly on GPU. As the text amount increases, EasyOCR takes more time to compute. When we are trying to use EasyOCR to read a nice and well PDF file and it tools like 2 minutes to get the output using the CPU mode in Google Codelab. Tesseract is preferable for CPU whereas EasyOCR is for GPU machines. So we can focus more on the image pre-processing images in order to train Tesseract OCR to a better performance.

Why did we take such a long time on Tesseract OCR and reading images?

The challenge of reading images is a big part that we expect to get higher accuracy on reading text out, but there is no such a good Optical Character Recognition model that can extract all the text from any image. I asked the RubiconMD team if we can classify the images and files we will get so we can do different image processing or use different OCR models based on the kinds of images we can get, but RubiconMD told us that we will expect to get any kind of images and pdf files. So we need to have an OCR model that can read text on any kind of image and pdf files which is the hardest part. Along with all those existing technology and company, we didn't find such an OCR model that can give us the closest accuracy text redaction. So we tried to do the image processing part to make the confidence level (indicates the degree to which the model is certain that it has recognized the component correctly) at least 90 (out of 100).

How do we know which image needs to do preprocessing?

The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation) are classified among pre-processing methods here since similar techniques are used. Based on the confidence level we get we can do extra image preprocessing in order to move on next step.

Do I experience any time conflicts with my team?

Yes, on one of the maker days, I was thinking of a solution that when we have the patients' information as a dataset, we can just look for the personal information that is a possible match in the image and then block those text inside of a file. So we will skip the part of natural language processing and do a little bit of software engineering work. After long discussions with my team and we think we should keep moving forward with the natural language solution part because it is better, also in some cases we don't have the patient's information.

After lots of research and testing on other OCR models we still move on with the Tesseract OCR model, if I can do this project again, would I still do those research?

Yes, I will still take my time on research OCR models, because I can find the advantages and disadvantages between OCR models to find the best one. I also learned a lot about how those OCR models work. And if I have time, I would focus more on how to make the Tesseract OCR make 99% accuracy as Google did.

Tasks

Estimated Timeline

Task (What will be done)	**Expected Outcome** (Expected result of task)	**Start Date**	**Deadline**
Dataset preparation + library and environment setup	• Generate a dataset of at least 200 examples • Each team member should have all necessary libraries and environments set up on their PCs. • Dataset should be uploaded to our shared google folder • The dataset should be uploaded to our Google Colab notebook •	9/1/2022	9/5/2022
Data Preprocessing	• Enlarge our dataset by 150% using data augmentation techniques • Data in a ready-to-use state for analysis	10/5/2022	10/25/2022
Text Exaction	• Completed Feature	10/12/2022	11/2/2022

	selection for text extraction ● Defined metrics for text extraction ● Achieve 90% accuracy rate		
Text Identification	● Prepare data for text identification ● Achieve 90% accuracy rate	11/5/2022	11/25/2022
Text redaction	● Prepare data for redaction ● Construct an initial model for redaction ● Construct final model ● Achieve 90%accurate	11/20/2022	12/3/2022
Clean up	● Complete any previous task ● Analyze work ● Look into deployment	11/30/2022	12/8/2022

Actual Timeline

Task Done	**Actual Outcome**	**Start Date**	**Finish Date**
Dataset preparation + library and environment setup	● Generate a dataset of at least 200 examples ● Each team member should have all necessary libraries and environments set up on their PCs. ● Dataset should be uploaded to our shared google folder ● The dataset should be uploaded to our Google Colab notebook	9/1/2022	9/13/2022
Data Preprocessing	● Enlarge our dataset by 150% using data augmentation	10/5/2022	10/25/2022

	techniques ● Check on the OCR models ● Testing the OCR model		
Text Exaction	● Completed Feature selection for text extraction ● Defined metrics for text extraction ● Research and test more OCR model	10/12/2022	11/11/2022
Text Identification	● Prepare data for text identification ● Image preprocessing	11/25/2022	12/5/2022
Text redaction	● Construct an initial model for redaction ● AWS and research	11/10/2022	12/3/2022
Clean up	● Combine work of redaction progress and also store path ● Process file input workflow and output file workflow	11/30/2022	12/8/2022

Project Logs

Name: Linqing Zhu
Areas of responsibilities: In my group, I am in a role as a machine learning engineer. I researched all different optical character recognitions about how can make optical character recognition (OCR) read text better. I took testing on machine learning models to find the best models that fit into our project. I also work on redacting protected health information and store into the corresponding path.

Date	**Duration**	**Description of completed work**	**Challenges or Next steps**
9/4/2022	5 hours	Understood the project based on the introduction video from RubiconMD, prepared to meetings with my teammates, and introductions	Find out the right time to meet with teammates, prepare questions with teammates about the project

9/9/2022	4 hours	Team meeting: filled out the Project Scope and Deliverables assignment, prepared questions to ask RubiconMD Challenge Advisors	Fill out the when2meet timeslots of free time and find out a regular time to meet every week, starting to research the problem, secure a time for meeting with RubiconMD team
9/10/2022	6 hours	Researched the company RubiconMD, looked for the problem of text redaction, and checked if existing technology can accomplish this task	Prepare questions to meet with RubiconMD team
9/12/2022	2 hours	Supervisor discussions: asked about expectations on project and guidelines, asked materials can be provided by RubiconMD, Challenge Advisor suggested use to use Tesseract Optical Rogonzaion Model (OCR) to do text redaction on images	RubiconMD will provide a small sample dataset, we need to find more similar dataset
9/13/2022	5 hours	Data Understanding and Preparation: found and added more similar images for text recognition on images	We don't have actual real-life medical documents, so we look for all kinds of images have text to be our dataset, but we didn't have the output of the accurate text extract
9/16/2022	5 hours	Coding: Tried and downloaded Tesseract OCR	Not able to use Tesseract OCR on Google Colab but able to use it on a local computer. On a good track to learn and use tesseract, find out a way to use it on Google Colab
9/17/2022	5 hours	Designed the test model and data: setup Google Colab to read data files and converted images to PDF files in order to do text redaction	Debug Pytesseract OCR to run on Google Colab to continue share code and data files

9/18/2022	5 hours	Debugged error of using Pytesseract OCR on Google Colab	Try Pytesseract OCR on more images
9/20/2022	2 hours	Coding: tried out Tesseract OCR model	Tesseract OCR didn't read all the text inside the image, some text output is not accurate
9/21/2022	2 hours	Supervisor discussions: talked about the issues with Tesseract OCR and debugged the text output that can get the location of each text being read	Think about another solution or improve the Tesseract OCR model
9/23/2022	3 hours	Coding: Figured out how to run draw marks on the Tesseract OCR to see where the text was being read	The Tesseract OCR is not good enough to extract all the text from an image, need to train the Tesseract OCR to have a better result
9/24/2022	7 hours	Maker Day Team Meeting: Shared and combined work of trying on Tesseract OCR, updated a few dataset	Continue on look a the Tesseract OCR and look for other OCR models
9/25/2022	3 hours	Coding: used Tesseract OCR on more images	Not able to make Tesseract OCR work well, still need to fill out a way of reading images
9/28/2022	3 hours	Filled and submitted assignment presentation for Break Through Tech AI program	
9/29/2022	3 hours	Debugged error of converting pdf files to images	Research on more other OCR models
9/30/2022	2 hours	Meeting with TA discussions: asked about the next direction, asked about the classification of what kind of image or files we can expect to get as input	We are not able to classify what kinds of images we expect to get as input files. So we need to find out a model that can handle any kind of images

10/1/2022	3 hours	Coding: worked on EasyOCR model	The reading of EasyOCR model is much better than Tesseract OCR, check on the accuracy of the text
10/4/2022	4 hours	Supervisor discussions: disused about the EasyOCR work, by not sure about the accuracy of the EasyOCR, which one should we use, we need more comparison in order to move forward	Continue on the discovery of EasyOCR model and Tesseract OCR model
10/5/2022	5 hours	Coding: Changed the parameter of the OCR model to test the data	Tesseract OCR can get better accuracy by changing the parameter
10/7/2022	4 hours	Assignment: Working on the project report and fill out the time logs	Submit the assignment and time logs
10/8/2022	3 hours	Team Meeting: updated work progress and debug	
10/9/2022	3 hours	Coding, keep trying Tesseract OCR	
10/14/2022	2 hours	Meeting with TA: the models are good to work with, advisors suggest we try more on image preprocessing	Research on image preprocessing
10/15/2022	3 hours	Team Meeting: discussed the image pre-processing	Research about how other companies like Google and Amazon do OCR
10/16/2022	5 hours	Learning, researched Azure, Google, and Amazon	Continue to find how Google and Amazon those companies work on OCR
10/21/2022	5 hours	Research: researched how to determine protected health information. Found out what are the way can use Amazon Web Services to detect protected health information in the file, and the way we	The solution I have now is when we had extracted all the text from an image, converted the text extract result into a dataset, then convert it into a paragraph, use Amazon DetectPHI to

		can employ our model in the cloud	check if there is any PHI in the text, if yes then that document needs to do text redaction on the image. The next step is to check where is the location of the text, then put color on it to block.
10/22/2022	7 hours	Maker Day Team Meeting: image pre-processed on Tesseract OCR	
10/23/2022	5 hours	Researched Google OCR and Amazon OCR model	Not able to get how Google works on their OCR, we have to be on Google Cloud to test the OCR, not a workable solution for us. For Amazon, we need to pay for the service in order to use the OCR
10/25/2022	2 hours	Supervisor discussions: Sharpened the images can increase the accuracy score, asked where to store files, and checked the process of the whole project	No need to worry about the place to store files, so continue to check OCR models
10/27/2022	3 hours	Coding: EasyOCR got the locations of each text, converted the output into dataframe	I tried the EasyOCR model on pdf but the not working, still need to debug the error of pdf file to images conversion
10/28/2022	2 hours	Meeting with TA: checked in progress and assignment due	
11/2/2022	5 hours	Researched on diffferent OCR Model	
11/4/2022	5 hours	Debugged EssayOCR model pdf file conversion error to read images	Continue to try more pdf files text redaction by EasyOCR
11/5/2022	5 hours	Tested EasyOCR Model: worked on EasyOCR converts the output to	

		dataframe and draw box on what text is read, also able to draw blocker on the text	
11/6/2022	5 hours	Reseached on more OCR Model: Worked on Keras OCR but error in Google Colab notebook of using Keras OCR model	Continue on Keras OCR model
11/8/2022	2 hours	Supervisor discussions: discussed on how does different OCR functions on image and tried to find out a way to compare OCR models	Contine to research on different OCR models
11/9/2022	3 hours	Prepared for assignment team presentation: presentation on team Maker Day for the Break Through Tech AI program	
11/10/2022	5 hours	Tested EasyOCR Model and compare with Tesseract OCR	EasyOCR took more than 3 minutes even on a good pdf image so we may not move forward with EasyOCR
11/11/2022	2 hours	Meeting with TA: Moved forward with Tesseract OCR because it can be more trainable by doing image preprocessing and it is used by Google where the accuracy can up to 99%	Work on the combine my EasyOCR work with Tesseract OCR work image preprocessing
11/12/2022	3 hours	Project Report: Redone some of the time logs and work due to the lost of the laptop	
11/13/2022	5 hours	Project Report: Took a look at all the templates and submission criteria	Wait and check for the email back from supervisor
11/15/2022	2 hours	Supervisor discussions	
11/18/2022	5 hours	Debugged Tesseract OCR model	

11/19/2022	7 hours	Maker Day Team Meeting:	
11/20/2022	2 hours	Work with team to discuss next step	
11/21/2022	6 hours	Redo the time log and project log	
11/22/2022	2 hours	Supervisor discussions: Showed the work that connected AWS to the Google Colab	Futher step is write the function that locate the text and draw blocker on the text
11/23/2022	2 hours	Meeting with Advisor discussions: Checked in with project progress	
11/24/2022	7 hours	Project Report	
11/25/2022	7 hours	Project Report	
11/26/2022	6 hours	Project Report	
11/27/2022	5 hours	Converted Tesseract OCR output into dataframe and draw box on the output image, also find out function to block text in the image	Find out if color block can be erased by people when they using software, need to take that as a concern
11/28/2022	5 hours	Combined my work with my teammate's work on AWS identify protected health information result	
11/29/2022	2 hours	Supervisor discussions: Combined the work, on reading pdf, the issues after AWS identification	Need to create a path to store of the redacted image, concern of the python library PIL drawing box on the image, the result can still be erased by tool, need to research if it draws on pixels
11/30/2022	7 hours	Worked on the pdf to image and block on the protected health information, working on the path of saving the output of the image, Converted back the image to	Think about all the cases of text extraction file input, and how to handle different cases. There are issues with my teammate's code she only

		PDF, debugged the function to handle multiple pages, and merged all multiple images into one pdf file. Created code to store a multiple-page PDF page as an image after the text redaction and then combine the images back to a pdf	read the first page of the pdf we used, need fix that issues. Think about the input format and output format, and ask the supervisor for clarification as well. Case 1: When the input is an image file, data processing on the text and block protected health information in the image and then store as an image into the path they passed into the parameter Case 2: When the input is a pdf file: 1. Handle more than one page. 2. After the data processing on that pdf, it will output the images, so we need to store images into a pdf folder, and merge all the images in that folder into a pdf page Ask the supervisor: what is output file is like, pdf or images - Merge the code of conversion of PDF and cover text
12/1/2022	3 hours	Meeting with TA	
12/2/2022	5 hours	Work on combining the process with my team on reading PDF file	# Cases need to handle # 1. input an image and output a PDF into a specific path # 2. input a PDF, (convert to the image), draw on each image, and then store each image into a folder and then output a pdf into a specific

			path. An alternative way of case 2: input a pdf, store the image in a temp folder and combine the output of the image a pdf into a path
12/3/2022	8 hours	Debugged the error of reading pdf pages into an image, coded a function to combine all the dataframe output from one PDF file and the connect with teammate's data progress, also able to combine images in a folder into one PDF file	Find out function's connection, we are going to store all the image of a PDF files output into a folder, I still need to get the folder path to combine the images
12/4/2022	3 hours	Working on the final presentation	Work on the architecture diagram of the whole project
12/6/2022	2 hours	Supervisor discussions: Presentation preparation, got feedback, finalized works and asked about the company business need, I confirmed with my supervisor that when input is image and output could just PDF files. Also, the PDF output is secure in the case where some tool can remove the mark on file, we don't need to worry about that part	Practice the presentation, fix the code, and documentation
12/9/2022	3 hours	Final Presentation	
12/11/2022	10 hours	Project Report	Record Demo and submit work, check in with supervisor on email
12/13/2022	2 hours	Supervisor discussions: final works and answer questions	
12/15/2022	2 hours	Meeting with Advisor discussions	

Delivery

Deployment

RubiconMD is using Amazon Web Services on the system and file storing. We connect Google Colab to AWS and showcase what it may look like as an example to the team. The optical character recognition model and the text reduction tool will be deployed by the RubiconMD team in Amazon Web Services.

Testing

In order to test the optical character recognition model and the text reduction tool, we need to deploy it into the real RubcionMD system. RubcionMD employees will review and compare the result after the process of redaction is done. The testing part will be done by the RubiconMD team with a manual check on the errors of our model and comparing it with the expected outcome to check the percentage of accuracy. Currently, we are not able to see the result because it takes time more than our program timelines.

Documentation & Source Code

Each of us in the team was using a Google Colab notebook to keep track of the work and testing on different codes. Here is my Google Colab notebook where I was trying all the different OCR models, and also some other parts I test in other's notebooks. After we decided to move on with pytesseract OCR, I combined my work with my teammates where I am able to do text redaction using the output of OCR, convert images into a PDF, and store the output PDF file into a path. This Google Colab notebook combines our whole workflow of the project.

Bio Sketch

Company, Team Members & Responsibilities

Company/Organization	**Break Through Tech AI Program - RubiconMD**	
Roles	**Name and Background**	**Responsibilities**
Team Members	1. Rojanaye Daley, Hofstra University (major in Computer Science and minor in Engineering) 2. Linqing Zhu, Brooklyn College (major in Computer Science)	1. Rojanaye worked on the project management of timeline catchup, and assignments due for the program, and also allocate meetings with RubiconMD, advisor, and team meetings. She also did the part of Colab

	3. Ashley Diaz, City College (major in Computer Science) 4. Hoai Cao, Hunter College (major in Computer Science and Mathematics) 5. Kosi Ugorji, Columbia University (major in Engineering)	deployment on Amazon Web Services. 2. Linqing worked on the research and testing of optical character recognition (OCR) models, block text, and input-output files stored in path files 3. Ashley worked on Tesseract OCR installation on Colab and Colab deployment with Amazon Web Services 4. Hoai worked on how to do image pre-processing on the image to improve Tesseract OCR confidence level in reading images 5. Kosi dropped the program and team after the first month of the project, she worked on our assignments, prepared datasets, and created shared Google Drive
Challenge Advisor(s)	1. Christian Lewis, Director of Data Science 2. Jen Lau, Senior Director of Platform Operations 3. Cream Pepito, CTO	Challenge Advisors from RubiconMD met with us weekly, kept us on progress, and answered questions about the project and guidelines.
Break Through Tech AI Studio TA	Ananya Ganesh	TA helped us solve issues on our code and checked in with us during the Break Through Tech AI Maker Day with the team