Final project: Burial records classification and text extraction

Northeastern University

EAI6020: AI System Technologies

Daya Rudhramoorthi

July 3, 2021

Group 1

Xinyi Luo

Project Description

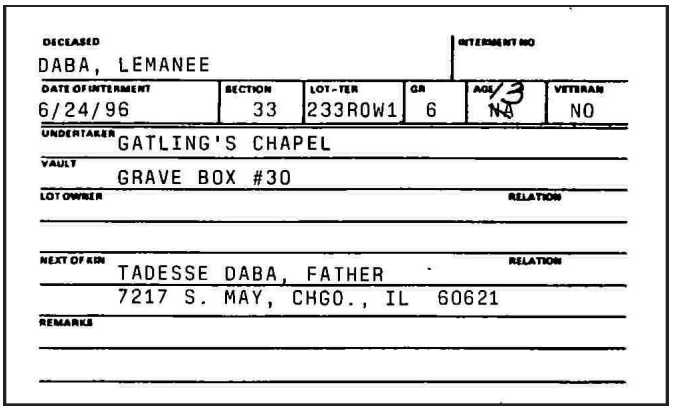
Introduction

Burial records are important evidences for tomb management. They often contain the name of the deceased and the detailed location and date of the burial. Because of the large time span since the burial service started, many dated records have not been electronicized, and there are handwritten records. This project aims to digitize these records. Digitalization of these records helps maintain them as well as analysis on historical populations.

Data description:

The data set consists of 120 scanned images in type '.jpg'. Each image is one burial record. There are two kinds of formats distinct from each other:

1. Organized form with detailed information (*Type 1*)


DECEASED
DABA, LEMANEE
INTERMENT NO
DATE OF INTERMENT | SECTION | LOT-TER | GR | AGE | VETERAN
6/24/96 | 33 | 233ROW1 | 6 | NA 73 | NO
UNDERTAKER GATLING'S CHAPEL
VAULT GRAVE BOX #30
LOT OWNER RELATION
NEXT OF KIN RELATION
TADESSE DABA, FATHER
7217 S. MAY, CHGO., IL 60621
REMARKS

Essential variables:

"Deceased" -Decedent name

"Date of Interment" -Date of burial

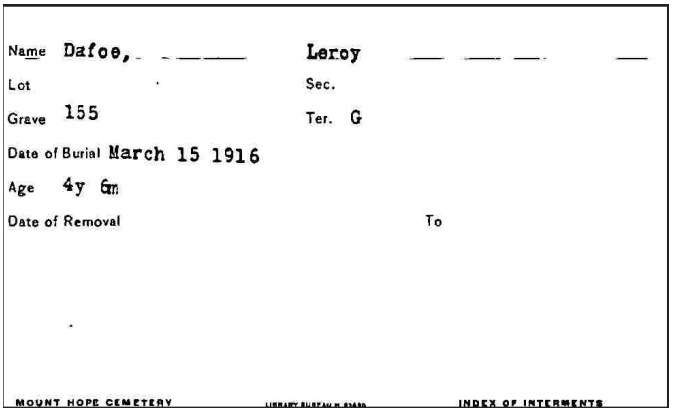
"Section" -Section of cemetery

"Lot" -Lot of cemetery

"GR" -Grave spot

Other variables: "Funeral Home", "Vault", "Next of Kin", "Address", "Age"..

2. Former form with only essential information (*Type 2*)


Name Dafoe, Leroy
Lot Sec.
Grave 155 Ter. G
Date of Burial March 15 1916
Age 4y 6m
Date of Removal To
MOUNT HOPE CEMETERY INDEX OF INTERMENTS

Essential variables:

"Name" -Decedent name

"Lot" -Lot of cemetery

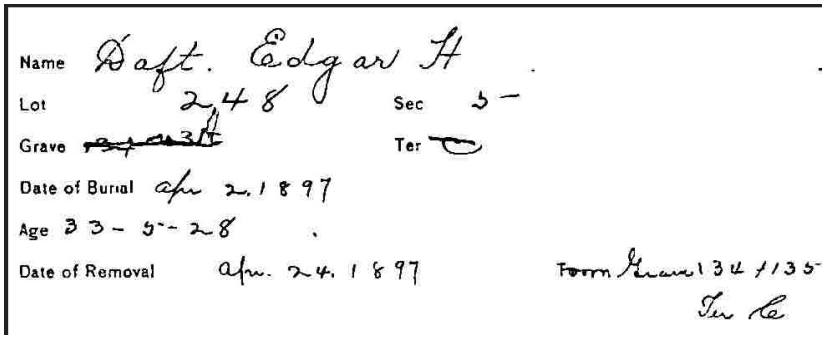
"Date of Burial" -Date of burial,

"GR" -Grave spot of cemetery

"Sec." -Section of cemetery

Other variables: "Ter.", "Date of Removal", "Age"..

One thing to notice for type 2 forms is that there exist several forms with handwritten contents, which could heavily interfere with the machine's cognition ability as shown below.


Name Daft. Edgar H
Lot 248 Sec 5
Grave ~~34~~ 31 Ter C
Date of Burial abt 2.1897
Age 23-5-28
Date of Removal abt. 24.1897 Form Grave 134/135
Ter C

Methodology

This digitization project could be divided to two parts:

1. Image preprocess: train a classifier to recognize different types of forms and process the images separately through image preprocessing tools.

   Techniques applied:

   - CNN model for form classifier.
   - OpenCV for reading and improving image quality.

2. OCR progress: extract text content from treated images and record values into a dataframe file for remaining.

   Techniques applied:

   - Py-tesseract-OCR for text extraction.
   - Azure Form Recognizer API for forms recognition.
   - Power Automate, platform to generate workflow for FormRecognizer.
   - Google Vision API

Form Classifier Building

Convolutional Neural Network(CNN) model:

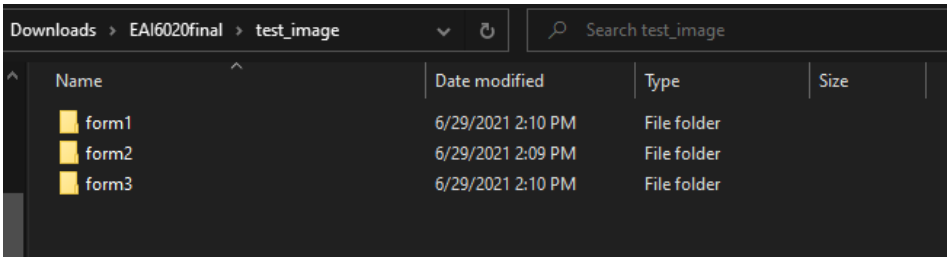
For dealing with images, CNN is known as one of the most powerful models for training with images. Therefore, we are going to build the image classifier based on CNN.

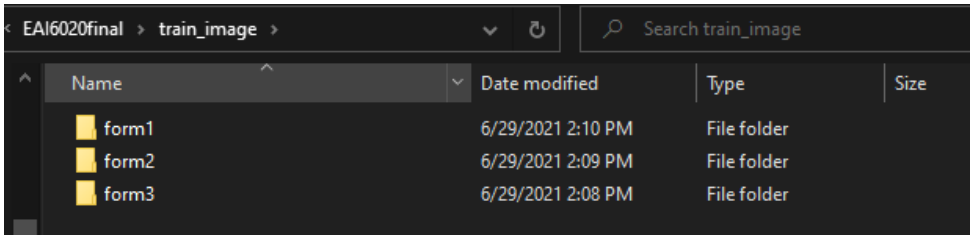
Classifications determination:

Although the dataset has two main types of forms, the handwritten forms seem to cause high error because of their high variance. As a result, our classifier is going to be a multiple classifier for 3 form types:

- Form 1: Organized form
- Form 2: Older form with printed writing and lines
- Form 3: Older form with handwritten content

To prepare for model training, 120 images are divided into ‘train_images’ and ‘test_images’ with a ratio of ‘7/3’, which to say the first 36 images are generated to the test dataset and 84 images are put into the train dataset. Both train and test images are divided into 3 folders with labels ‘form1’, ‘form2’, ‘form3’, as shown below.





Import image data through ImageGenerator and check the condition of both datasets. And the output should be a matrix with one-hotkey indicating which form the image is.

```
train_set = train_gen.flow_from_directory('train_image',
                                          target_size = (64, 64),
                                          batch_size = 4,
                                          class_mode = 'binary')

test_set = test_gen.flow_from_directory('test_image',
                                        target_size = (64, 64),
                                        batch_size = 4,
                                        class_mode = 'binary')
```

```
Found 84 images belonging to 3 classes.
Found 36 images belonging to 3 classes.
```

```
from keras.preprocessing import image
train_set.class_indices
```

```
{'form1': 0, 'form2': 1, 'form3': 2}
```

Model initialization:

For the CNN mode, ‘relu’ is going to be applied as the activation function.

Structure of the model:

- Two convolutional layers with activation function and followed by two pooling layers to feed forward the image data.
- One flatten() layer to connect all nodes.
- One dense layer with ‘relu’ function to generate the result of parameter weights.
- One Softmax() layer with ‘softmax’ function to sparse the result into one of the 3 classifications set before.
- Compile with ‘RmsProp’ optimization function.

```
# Initialising the CNN
cnnmodel = Sequential()

cnnmodel.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
cnnmodel.add(MaxPooling2D(pool_size = (2, 2)))

cnnmodel.add(Conv2D(32, (3, 3), activation = 'relu'))
cnnmodel.add(MaxPooling2D(pool_size = (2, 2)))

cnnmodel.add(Flatten())

cnnmodel.add(Dense(units = 128, activation = 'relu'))
cnnmodel.add(Dense(units = 3, activation = 'softmax'))

cnnmodel.compile(optimizer = 'rmsprop', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

Classifier Performance:

Training parameters: {train_set, steps_per_epoch = 21, epochs = 50, validation_data = test_set, validation_steps = 3}

With a validation dataset with 3*batch_size = 12 images, the model reaches 1.0 accuracy on the test set. The performance seems satisfying. Further image improvement will be based on the classification result.

```
Epoch 48/50
21/21 [==============================] - 0s 23ms/step - loss: 0.1168 - accuracy: 0.9869 - va
y: 1.0000
Epoch 49/50
21/21 [==============================] - 0s 23ms/step - loss: 3.1752e-04 - accuracy: 1.0000
racy: 1.0000
Epoch 50/50
21/21 [==============================] - 0s 23ms/step - loss: 3.8014e-05 - accuracy: 1.0000
racy: 1.0000
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your da
t least `steps_per_epoch * epochs` batches (in this case, 100 batches). You may need to use
ng your dataset.
Accuracy of the model:  [0.00018936669221147895, 1.0]
```

Image Preprocess

Image improvement with OpenCV:

Resize all images to a shape (900, 500). Crop all images with only ‘essential variables’ area.

Process all images through grayscale, denoise and filtering of OpenCV.

Recheck the size of processed images: ‘Form1’: all sized (300, 1800)

‘Form3’: most sized (380,1620). However, ‘Form2’ have sizes distinct in a wide range.

```
[[0. 1. 0.]]
This is a form 2 image
the shape of processed image: (760, 3240)
Decedent D ahleen Myrtle
```

```
[[0. 1. 0.]]
This is a form 2 image
the shape of processed image: (1520, 6480)
Decedent atin Yietor
```

Therefore, to cut small fields of a single variable, we need to cut with ratio. (For example: How much percentage does the 'Decedent' block covering the processed Form 1 image)

```
# Cut needed blocks for different information by ratios\n",
def imacrop(type,attribute,image):
    height,width = image.shape[:2]

    if(type=='f1'):
        if(attribute=='decedent'):
            startrow,startcol = int(height*0.1923),int(width *0.01)
            endrow,endcol = int(height *0.67),int(width *0.6591)
            return image[startrow:endrow,startcol:endcol]
```

Text Extraction with Py-tesseract:

Install the tesseract in system and direct to the commander:

```
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'
```

For a single variable, extract the string from the cropped block and tune slightly with re.sub().

```
image_placeholder = form1process(image_red,a)

f1_decedent = pt.image_to_string(imacrop('f1','decedent',image_placeholder), lang ='eng')
f1_decedent = re.sub('[^A-Za-z0-9]+',' ', f1_decedent)
f1_decedent = f1_decedent.replace('DECEDENT','')
f1 decedent = f1 decedent.replace('DECEASED','')
```

Text Extraction Performance:

Surely, tesseract could extract text from the designed area. However, the results obviously need much more tuning and improvement. I was trying to include as much information as possible to adapt the most images. That causes there is much noises text involved and extracted with needed information, as shown below:

```
[[0. 0. 1.]]
This is a form 3 image
the shape of processed image: (380, 1620)
Decedent  Dahike Reinhold
Date nt “July 13, 1945
Section
Lot NEt 190
Grave
```

```
[[0. 0. 1.]]
This is a form 3 image
the shape of processed image: (380, 1620)
Decedent Dahlman William 3
Date Mareh 10 1919
Section 18
Lot Vis 87
Grave
```

```
[[0. 1. 0.]]
This is a form 2 image
the shape of processed image: (760, 3240)
Decedent Jahlman William HB
Date farch 10 1919
Section
Lot B87
Grave
```

```
[[0. 1. 0.]]
This is a form 2 image
the shape of processed image: (1520, 6480)
Decedent Jahlman William ZH
Date farech 10 1919
Section
Lot 31 Std
Grave
```

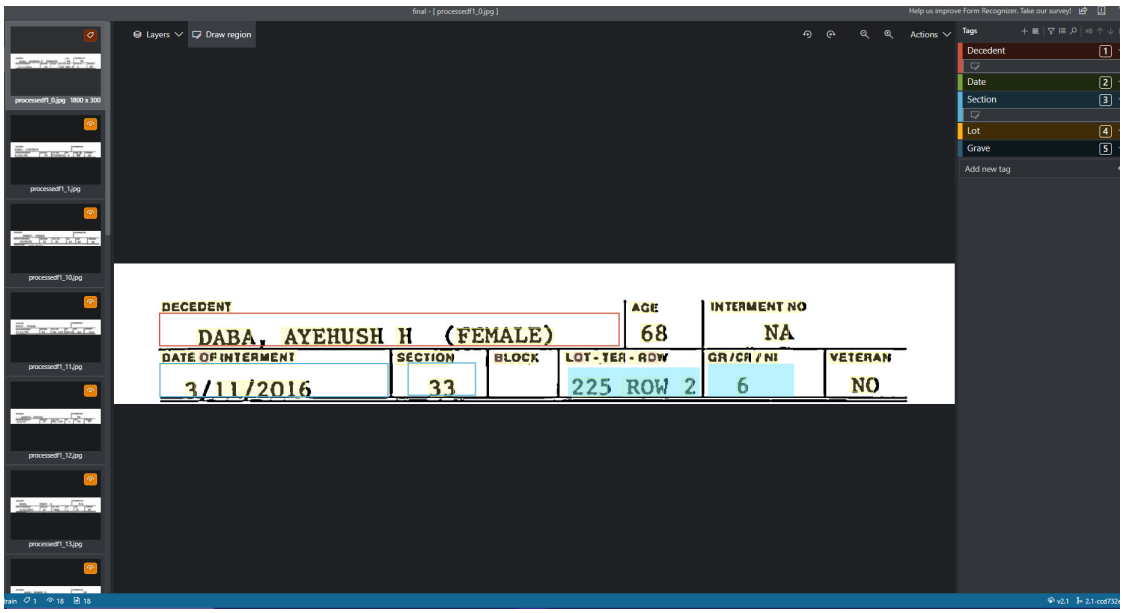
```
[[1. 0. 0.]]
This is a form 1 image
the shape of processed image: (300, 1800)
Decedent DAILEY LEON CATE OF CHTIAMINT gsec
Date 7/21/2003
Section
Lot 125
Grave
```

```
[[1. 0. 0.]]
This is a form 1 image
the shape of processed image: (300
Decedent OEeceasid DAILY DOLORES M
Date Sr aw ee hadi1/2/2001
Section
Lot INW266
Grave
```

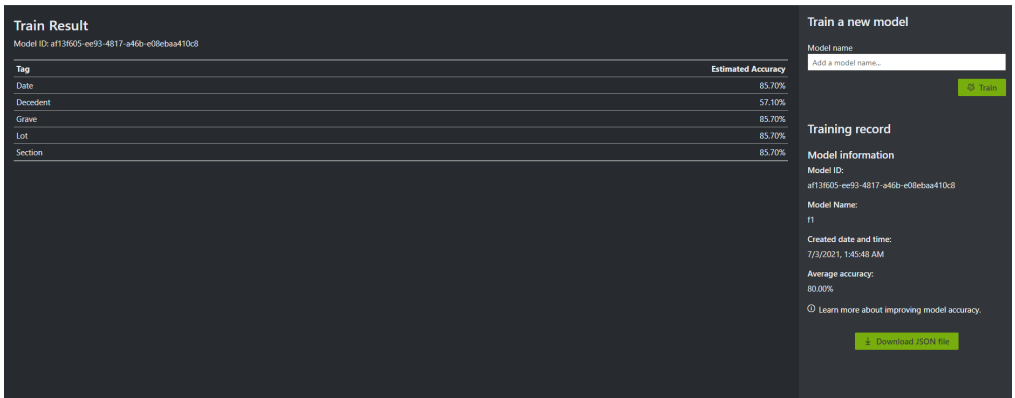
Apply Azure FormRecognizer API

FormRecognizer(FR) is an AI-powered text extractor deployed by Microsoft on their cloud platform, Azure. FormRecognizer could be trained to recognize the layout of certain forms and automatically drag paired information from forms with the same layout.

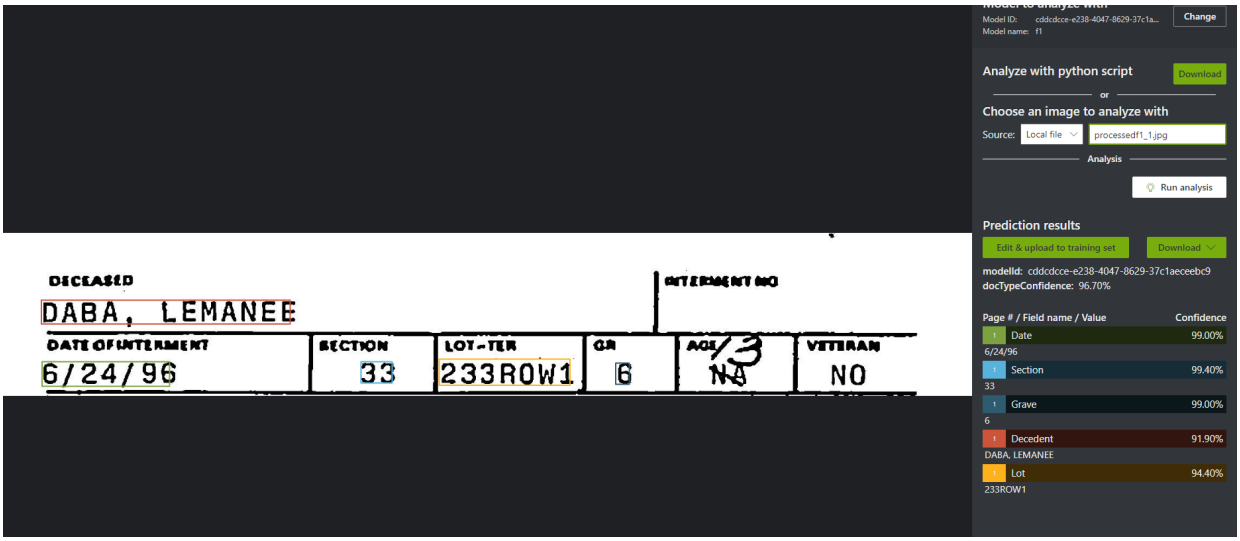
Step 1: Manually draw regions on the train set images and attach labels to them.



Step 2: Train the model

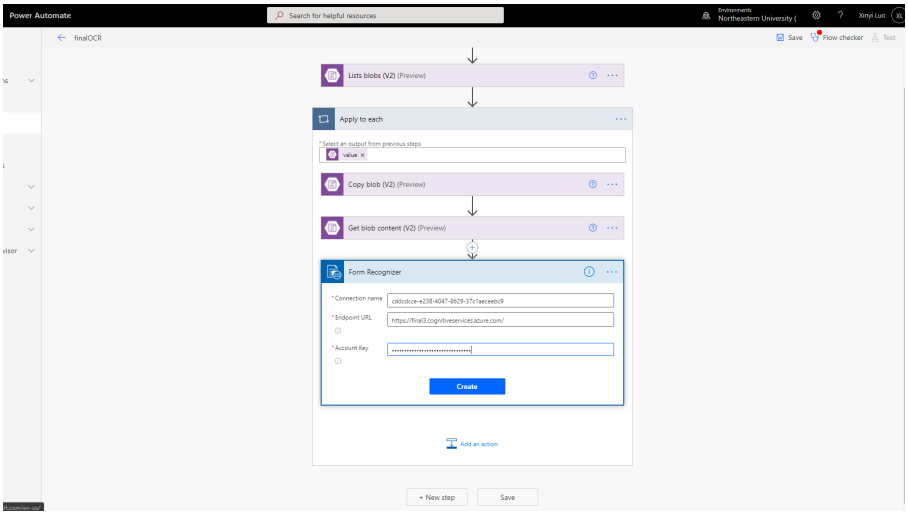


Step 3: Check the performance by predicting a new image.

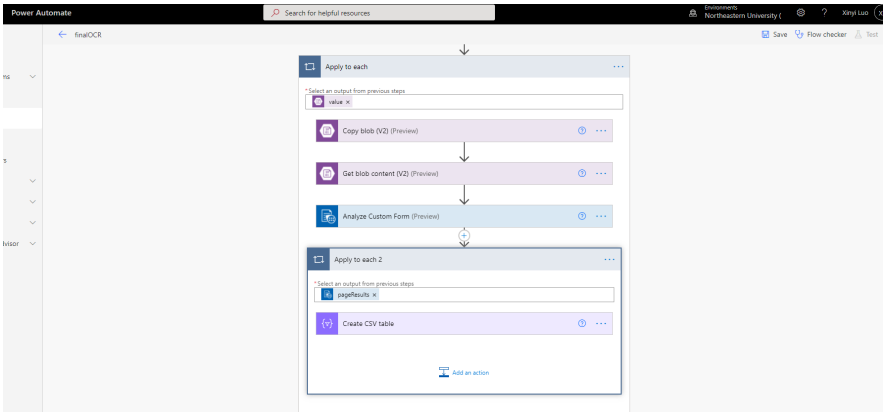


I used the cloud sample trainer to train my FormRecognizer model. And generate a workflow on Power Automate platform to run the pipeline through processed images to excel data.

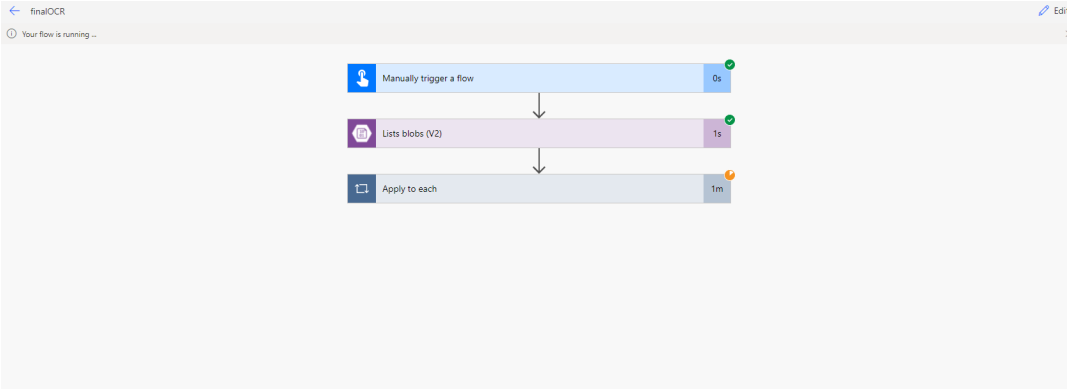
Step 1: Upload data to Azure Blob storage



Step 2: Conduct workflows with function nodes on Power Automate



Step 3: Run the flow and export the result.



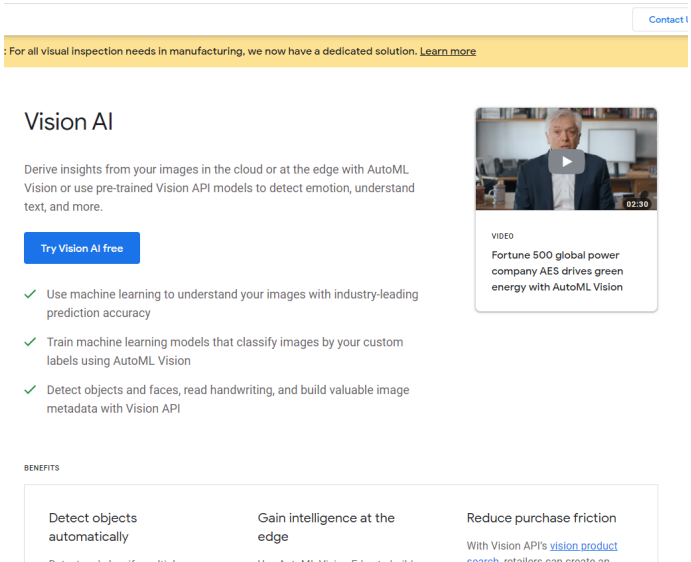
FormRecognizer Performance:

FR works better than the original tesseract. Results come in a well-organized format with clean format. That is because FR firstly recognizes text content clusters in the whole image, then starts to learn the layout though text clustering. That helps it avoid many noise pixels. And I'm using this dataset as my final submission.

A	B	C	D	E
Decedent	Date	Section	Lot	Grave
DABA AYEHUSH	3/11/2016	33	225 ROW 2	6
DABA LEMANEH	6/24/96	33	233ROW1	6
DABA SHORRO	9/19/96	33	233ROW1	2
DABE PATRICIA	5/11/96			510
DABNEY ABRAH	12/8/97	28	16	6
DABNEY ERWIN	11/28/2015	30	30 ROW 2	5

Other text extraction alternatives:

Google Vision API:



It is also a trendy tool for computer vision tasks.

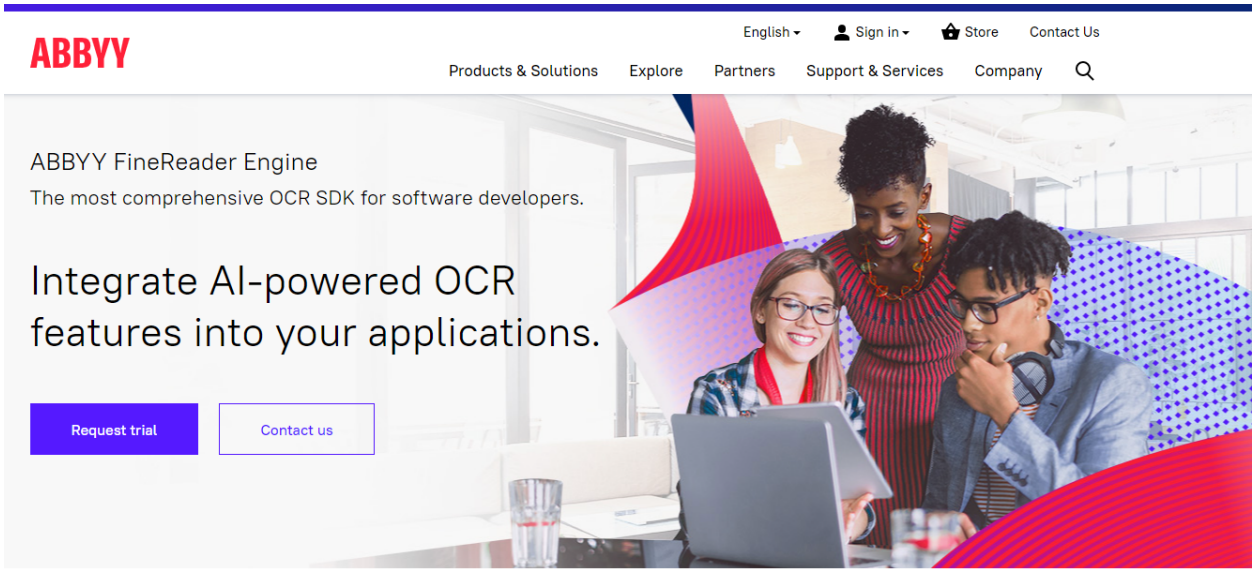
To apply the Google Vision, the most simple way is to deploy a script with access to Google Cloud Platform.

Vision API could be installed and imported as an extension.

With a GCP account, users could apply the API directly to the data and generate text from results.

ABBY Reader Engine:

ABBYY is a company who releases a series of OCR document reader applications. The engine provides an integrated end-to-end text extraction service. But it cost a little much to purchase an engine like that. It is more suitable for business.



References

Microsoft Azure. (). *Form Recognizer – Automated Data Processing Systems* https://azure.microsoft.com/en-us/services/form-recognizer/

Microsoft Docs. (2021). *IDs - Form Recognizer - Azure Applied AI Services* https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/concept-identification-cards

Microsoft Power Platform. (2021). *Microsoft Power Automate Portal* https://us.flow.microsoft.com/en-us/

Issagha BA. (2020). *How to automate form processing with Azure Form Recognizer* - Microsoft Industry Blogs - United Kingdom. https://cloudblogs.microsoft.com/industry-blog/en-gb/technetuk/2020/10/22/how-to-automate-forms-processing-with-azure-form-recognizer/

Lee, M (2021). Pytesseract · PyPI. https://pypi.org/project/pytesseract/

Microsoft Azure. (2021). *Home - Form OCR Testing Tool (fott-2-1.azurewebsites.net)* https://fott-2-1.azurewebsites.net/

Google. (2021). V*ision AI | Derive Image Insights via ML* | Cloud Vision API (google.com) https://cloud.google.com/vision/

ABBYY. (2021). *The Digital Intelligence Company* https://www.abbyy.com/