Receipt Image and PDF Processing

Report Prepared by:
Kyra Melenciano - 100379793

August 8th, 2022

Table of Contents

Abstract 3

Problem Statement 4

Training Data 4

Optical Character Recognition 4

Tesseract OCR 4

Image Preprocessing 5

Keras OCR 6

Paddle OCR 6

Feature Extraction 7

PDF 9

Feature Extraction 9

Spacy Format 10

Named Entity Recognition Model 11

Model Validation 11

Clustering and Similarity Analysis 12

Summary and Recommendations 13

Abstract

The aim of this project is to document the steps behind image and PDF processing to extract information from receipts. The datasets under consideration consist of 118 JPG images and 85 PDF files containing invoices from restaurants, stores, transportation services, among others. The goal is to build a machine learning model that is able to extract business name, invoice date and invoice total from JPG and PDF receipt files.

This report is organized as follows. Section I is a brief description of the problem statement and the goal of this project. Section II explains the process we took to extract training data examples from JPG and PDF files. Section III describes the use of python's Spacy library for Natural Language Processing (NLP) and the creation of a Named Entity Recognition model to extract the desired information from the provided files. Section IV shows the results of clustering and similarity analysis. Finally, section V concludes the report with a summary and recommendations for further stages of the project.

I. Problem Statement

The main purpose of this project is to build a program that, using a machine learning model, is able to automatically extract key information from a JPG or PDF receipt. The program should take as input JPG or PDF receipt files, extract the text from the receipts, parse through a Named Entity Recognition (NER) model to identify the business name, invoice date and invoice total, and output a dataframe which includes the information extracted, along with the text and filename of the receipt.

II. Training Data

To train a model to extract the information we need from the JPG and PDF files, we have to provide the model with training instances. However, if we were to have thousands or millions of receipts, we could not extract this information manually, which is why we have decided to use hard coding to extract the invoice's key information as a first instance. These training examples will be used to train a model, which then should be able to identify the key information on its own.

Since we have two types of input data, JPG and PDF, each of them need to be treated in a different way. To extract the text from the 118 JPG files, we have used Optical Character Recognition from python's tesseractOCR and paddleOCR libraries. To extract the text from the 85 PDF files, we have used python's BORB library.

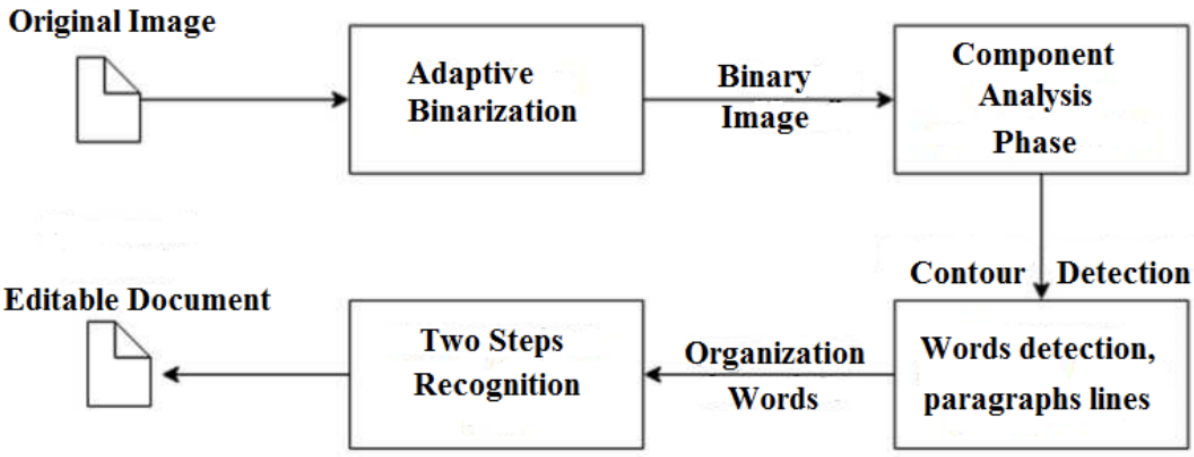
a. Optical Character Recognition

To extract the text from JPG files, we have used Optical Character Recognition (OCR), a technique that locates and recognizes text from images. We have tried two approaches, tesseractOCR and paddle OCR, which are both open source OCR engines released under the Apache License 2.0. The process of extracting the text with each of these engines is explained below.

Tesseract OCR

Tesseract[1] is an open source OCR, which supports a wide variety of languages. Image text extraction with Tesseract consists of taking the original image and "preprocessing by providing a binary threshold, determining the connected components and connections between them (also storing them in objects called blobs), character recognition and character aggregation to form words, lines, paragraphs and finally solving the problem of detecting small capitals". More information about Tesseract OCR's architecture can be found here.

[1] https://tesseract-ocr.github.io/tessdoc/#tesseract-user-manual



1. **Image Preprocessing**

To be able to effectively use tesseract to extract the text in the receipt images, the images have to go through some preprocessing. We defined functions to perform these preprocessing steps, including converting the images to grayscale, dilating and eroding to remove noise. After these, to crop the images and only get the part of the image in which the receipt is present, the images were again converted to a grayscale, blurred and edge detection was applied to reveal the outline of the objects inside the image; with the objects outlined, we grabbed the largest contours and cropped according to these to reveal a cropped image.

The text below on the left is extracted using tesseract on the cropped image; the text on the right is extracted using tesseract on the processed image without cropping.

```
.= extracting Text from cropped image using Tesseract =.
------------------------------------------------------------
'
~
weace ee 4
efits nse ate
Suereys O° yar 2#e |
nant 122 “ onsn0seen |
oa pae23094 |
fy wet
assent
purchas?
sa CREONT
cp PH LEASES
10: gn030031 618
ntrt Hethod? Haved
Batch: gatads
430728 fer59214
pet 0004484084
Inv qoaegT Aer Code? 049060
faount? $ 30
18
Tet : 0.09
Total: yo me
vqva gy wwe

------------------------------------------------------------
```

```
.= extracting Text from original image using Tesseract =.
------------------------------------------------------------
_
a,
*
: B: MAHAVEERS CHEFS CHOICE
- 1 ( KING GEORGE BLVD
4 SURREY, BC V3T 2N6 :
, Nerchant ID: 868000005648419 a
; erm ID: 084623094
Clerk thi 4
25981656018
Purchase
B Visa CREDIT :
Me OXAXXKHAXXKKXL315
AID: AGOABOROO31010
Entry Method: Waved
Batch: (00048
04730720 18:58:14
Refit: 000014484064
Inv : 000697 Appr Code: 049068
Anount: $ 30,18
a Tip: $ 0.00
Total: _
IVR: 00 O86 BO BG BO a
[Sl:66 oo is hes ne
i
oo.
Customer Copy ae

------------------------------------------------------------
```

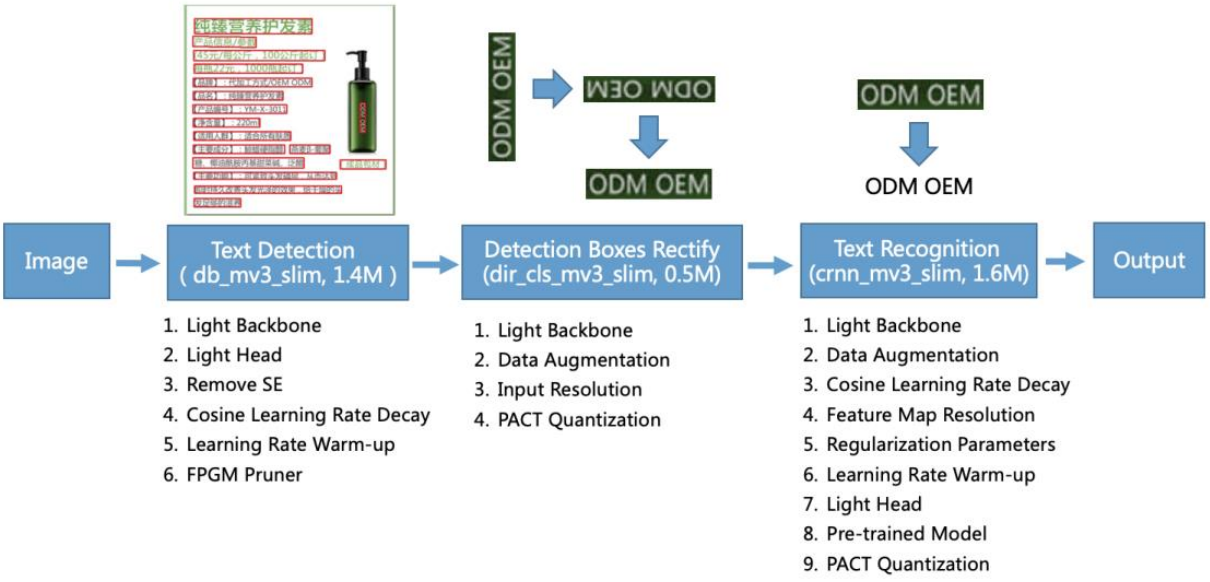
Keras OCR

We also tried text extraction using Keras OCR, which provides an API for training a text detection and OCR pipeline. The text below shows the text extracted using keras on the same image presented above. As can be observed, keras extracts the text in a word for word basis and doesn't keep the document structure or context.

choice mahaveers chefs 10227 blvd king orge ge 216 surrey v31 bc csonossabll merchant ide teton	ida 94823694 clerk ids 1 25a816s001e purchase credit visa xxxxxxxxx1s abggn16 aids entry methoda waved jigga8 batchth 18158611 93120	reftsgo1841694 0967 041060 coder appr inv hi 30 18 amount tip o0 d	lotall 30 18 vra oo 00 ud 00 uu isi 9o 00 s custoner copy

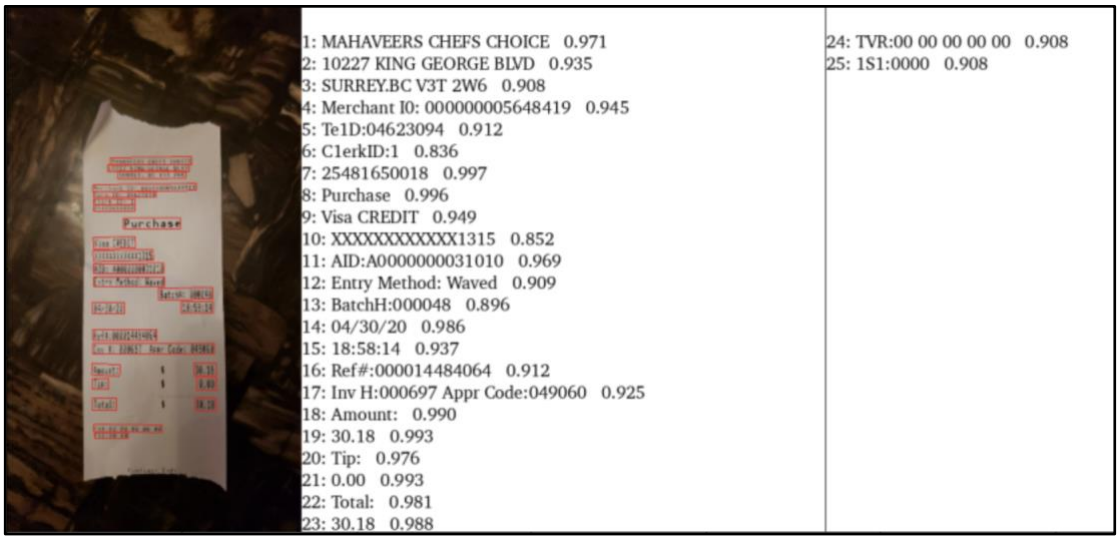
Paddle OCR

PaddleOCR[2] (PP-OCR) is an Optical Character Recognition system based on Convolutional Recurrent Neural Networks (CRNN) to extract text in images. The diagram presented below shows the steps performed by PP-OCR to accomplish text extraction; PP-OCR locates the text area in the image using Differentiable Binarization, performs rectification of the text boxes transforming them into horizontal rectangle boxes, which are then input to the CRNN to recognize the text in the detected boxes and output the extracted text. More information about PP-OCR can be found here. Additionally, PP-OCR is licensed under the Apache License 2.0, which enables permission for commercial use, modification, distribution, patent use and private use; the license can be found here.



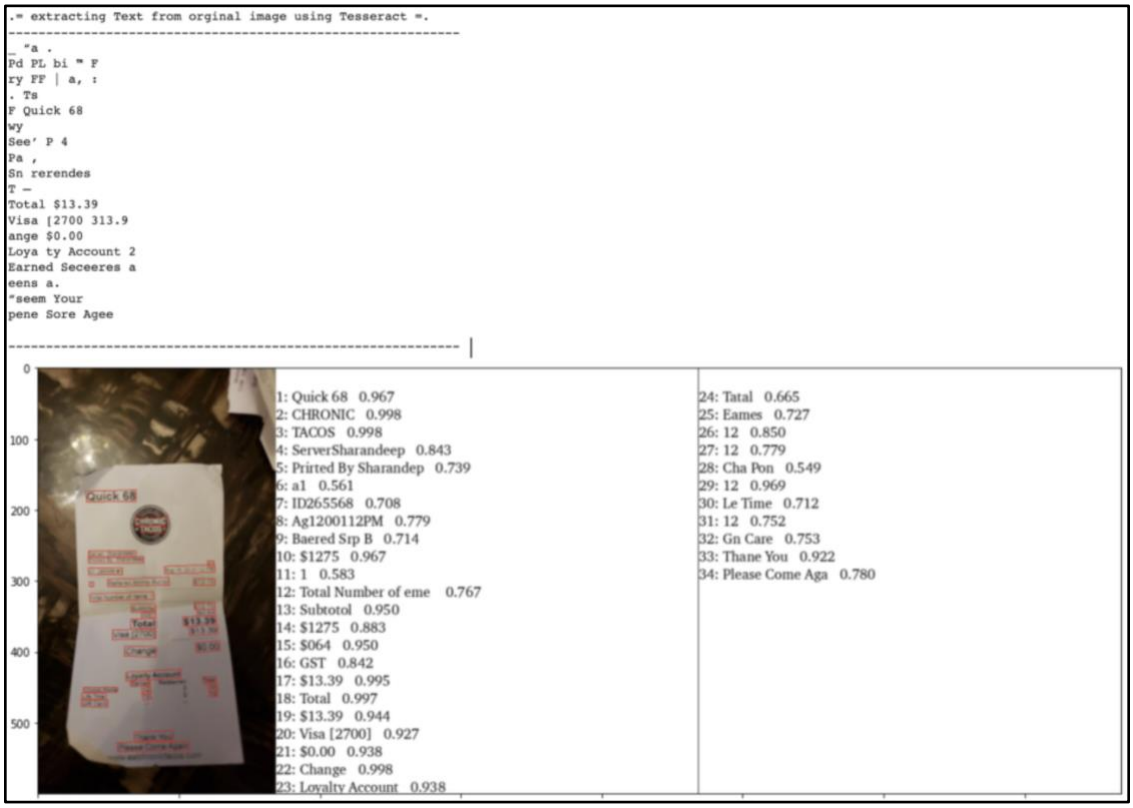
After using PP-OCR on the same receipt processed above, the text below is extracted.

[2] PP-OCR.pdf



Feature Extraction

Since we want to extract the text as it is, not word for word, Keras OCR was disqualified and we decided to compare the results of Tesseract and Paddle. The text below highlights the difference between the two OCR systems.



As can be clearly observed, PP-OCR performs better than tesseract in the text extraction, making this the OCR system we used to extract the text in all the receipt images.

Now, because we want to train a model to be able to extract the business name, invoice total and invoice date from receipts, we need to extract this information ourselves in a first stage to feed in a later stage to the model as training data.

To build the JPG training data, the approach followed was to divide the images into several different folders depending on where the business name was located in the image (i.e. those with the business name in the first line of the extracted text, go into one folder; those with the business name in the second line of the extracted text, go into another folder); this was done because we were not able to find a pattern to initially extract the business name. After this, the program was written to extract the business name based on the condition of location being met. To extract the invoice date and invoice total, regular expressions were written to identify the different patterns of date and total in all the receipts.

Patterns of Business Name

```
if type == '1': # Name in first line
    businessName = text.split('\n')[0]
    businessStartIndex = text.find(businessName)
    businessEndIndex = text.find(businessName) + len(businessName)
elif type == '1-2': # Name in first and second line
    businessName = text.split('\n')[0] + ' ' + text.split('\n')[1]
    businessStartIndex = text.find(businessName.split(' ')[0])
    businessEndIndex = text.find(businessName.split(' ')[0]) + len(businessName)
elif type == '2': # Name in second line
    businessName = text.split('\n')[1]
    businessStartIndex = text.find(businessName)
    businessEndIndex = text.find(businessName) + len(businessName)
elif type == '2-3': # Name in second and third line
    businessName = text.split('\n')[1] + ' ' + text.split('\n')[2]
    businessStartIndex = text.find(businessName.split(' ')[0])
    businessEndIndex = text.find(businessName.split(' ')[0]) + len(businessName)
elif type == '3': # Name in third line
    businessName = text.split('\n')[2]
    businessStartIndex = text.find(businessName)
    businessEndIndex = text.find(businessName) + len(businessName)
elif type == '5-down': # Name 5 lines down
    businessName = text.split('\n')[4]
    businessStartIndex = text.find(businessName)
    businessEndIndex = text.find(businessName) + len(businessName)
elif type == '5-up': # Name 5 lines up
    businessName = text.split('\n')[-6]
    businessStartIndex = text.find(businessName)
    businessEndIndex = text.find(businessName) + len(businessName)
else: # No name
    businessName = ''
    businessStartIndex = ''
    businessEndIndex = ''
```

Patterns of Date

```
patterns = [
  # "Jan 21,20", "Jan 21,2020", "Jan 21 20", "Jan 21, 2020", "Jan 21, 20"
  #  "Jan 21.20", "Jan 21.2020", "Jan 21. 2020", "Jan 21. 20"
  (r"((?:jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec)\s?[0-9]{1,2}(?:,|\.|\s)\s?[0-9]{2,4})\s", 1, re.IGNORECASE),
  # "11/27/19", "12/14/2019", "2020-01-07", "20-01-2017"
  (r'([0-9]{1,4}(?:-|\/)[0-9]{1,2}(?:-|\/)[0-9]{1,4})', 1, 0),
  # "20-Jun-2020"
  (r'([0-9]{1,4}(?:-|\/| )(?:jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec)(?:-|\/| )[0-9]{1,4})', 1, re.IGNORECASE)
]
```

Patterns of Total

```
patterns = [
  (r'\s(?:payable):?\s?((?:CA)??\$?\d+.\d+)',1, re.IGNORECASE),
  (r'\n(?:(?:here )?total|totaldue|paid|tl|cash|Tctal|Totai):?\s?((?:CA)?(?:DS)?\$?\d+.\d+)',1, re.IGNORECASE),
  (r'\n(?:amount):?\s?((?:CA)??\$?\d+.\d+)',1, re.IGNORECASE),
  (r'(\$?\d+.\d+)\s(?:amount|total)', 1, re.IGNORECASE)
]
```

b. PDF

To extract information from the PDF receipts we used borb[3], a "pure python library to read, write and manipulate PDF documents". For our purpose, two cases were considered; first a function was defined to perform simple text extraction for simple PDF files, then a second function was defined to perform OCR text extraction within borb for those files that are PDF but the text can't be extracted (either because it is a scanned image or because there are tables, among other cases).

Feature Extraction

To build the PDF training data, the approach was to divide the receipts into different folders depending on the business name, which allowed us to find different patterns for the date, total and business name in each receipt folder; for each pattern, regular expressions were written to extract the desired information. The code below is an example of the functions defined for each receipt folder.

```
def getUberInvoiceInfo(invoice):
    filename = invoice.split('/')[-1]
    text = getInvoiceText(invoice)
    date = findInText(r'(\S+\s\d+,\s\d+)', text)
    dateStartIndex = text.find(date)
    dateEndIndex = text.find(date) + len(date)
    bussinessName = findInText(
        r"You ordered from (.*?)\n", text, group = 1)
    businessStartIndex = text.find(bussinessName)
    businessEndIndex = text.find(bussinessName) + len(bussinessName)
    total = findInText(r'Total (CA\$\d+\.\d+)', text, group = 1)
    totalStartIndex = text.find(total)
    totalEndIndex = text.find(total) + len(total)
    # city = findInText(
    #    address_re_pattern, text, group = 2)
    # province = findInText(
    #     address_re_pattern, text, group = 3)
    # postalCode = findInText(
    #     address_re_pattern, text, group = 4)

    return {
        'filename': filename,
        'text': text,
        'date': date,
        'dateStartIndex':dateStartIndex,
        'dateEndIndex': dateEndIndex,
        'businessName': bussinessName,
        'businessStartIndex':businessStartIndex,
        'businessEndIndex':businessEndIndex,
        'total': total,
        'totalStartIndex': totalStartIndex,
        'totalEndIndex': totalEndIndex
        # 'city': city,
        # 'province': province,
        # 'postalCode': postalCode
    }
```

```
def getDoordashInvoiceInfo(invoice):
    filename = invoice.split('/')[-1]
    text = getInvoiceText(invoice)
    date = findInText(r'Delivery: \w* (\w* \d*)', text, group = 1)
    dateStartIndex = text.find(date)
    dateEndIndex = text.find(date) + len(date)
    bussinessName = findInText(
       r"Thanks for your order, \w*\s([^\(]*) ", text, group = 1)
    businessStartIndex = text.find(bussinessName)
    businessEndIndex = text.find(bussinessName) + len(bussinessName)
    total = findInText(r'Total (\$\d+.\d+)', text, group = 1)
    totalStartIndex = text.find(total)
    totalEndIndex = text.find(total) + len(total)
    # city = findInText(
    #     text, r"\(([^\)]+)", 1)
    # province = ''
    # postalCode = ''

    return {
        'filename': filename,
        'text': text,
        'date': date,
        'dateStartIndex':dateStartIndex,
        'dateEndIndex': dateEndIndex,
        'businessName': bussinessName,
        'businessStartIndex':businessStartIndex,
        'businessEndIndex':businessEndIndex,
        'total': total,
        'totalStartIndex': totalStartIndex,
        'totalEndIndex': totalEndIndex
        # 'city': city,
        # 'province': province,
        # 'postalCode': postalCode
    }
```

[3] https://github.com/jorisschellekens/borb

c. Spacy Format

After extracting the text and key information from the JPG and PDF receipts, the output looks like presented below; a dictionary with extracted text, business name, invoice date and invoice total, along with the indices for the start and end characters in each of these strings.


```
 {'businessEndIndex': 94,
  'businessName': 'Guacamole Mexican Grill',
  'businessStartIndex': 71,
  'date': 'February 8, 2021',
  'dateEndIndex': 16,
  'dateStartIndex': 0,
  'filename': 'Feb 08 2021.pdf',
  'text': "February 8, 2021\nThanks for ordering, Chinmaya\nHere's your receipt for Guacamole Mexican Grill.\nTotal
CA$14.53\n1 Chimichanga CA$15.60\n Choose your side  Salsa Verde CA$0.00\n Choose your protien  Chicken CA$0.00\n1
Enchiladas Rojas CA$15.60\n Choose your protien  Chicken CA$0.00\nSubtotal CA$31.20\nTax CA$1.77\nService Fee
CA$3.12\nDelivery Fee CA$0.99\nDelivery Discount -CA$0.99\nDiscount -CA$1.56\nPromotion -CA$20.00\nYou ordered from
Guacamole Mexican Grill\nPicked up from\n13648 Grosvenor Rd, Surrey, BC V3R 5C9, Canada\nDelivered to\n9838 Whalley Blvd,
Surrey, BC V3T 5S8, Canada",
  'total': 'CA$14.53',
  'totalEndIndex': 110,
  'totalStartIndex': 102},
```


From these dictionaries, a dataframe was created which contained both PDF and JPG receipts' information. Since we were not able to extract the key information for all the receipts (because we couldn't find patterns for some), the receipts with missing values were removed from the dataframe, which finally contained 180 receipts between PDF and JPG. However, Python's Spacy library, which we'll use to build the Named Entity Recognition (NER) model, accepts a specific format of training data examples, in which the text needs to be provided and the entities' labels (business name, invoice date and invoice total) in that text need to be specified, along with their start and end indices inside of a list, so 180 examples were created from the clean dataframe. These 180 examples were split into training and validation examples in a 80:20 ratio and were then each parsed to a DocBin object, which serializes each training and validation example and saves the indices as annotations indicating which tokens are the entities/labels we want to predict. The output below is a depiction of two training examples for Spacy NER models before converting them to DocBins.


```
[("January 22, 2021\nThanks for ordering, Anindita\nHere's your receipt for Spice of Nepal.\nTotal CA$27.28\n1 Butter
Chicken Momo (Steamed or Deep-Fried) CA$12.99\n Choose your spice level  Medium CA$0.00\n Choose your preparation  Deep-
fried CA$0.00\n1 Chili Chicken Momo CA$12.99\n Choose your spice level  Medium CA$0.00\n Choose your preparation  Deep-
fried CA$0.00\nSubtotal CA$25.98\nService Fee CA$2.60\nDelivery Fee CA$2.99\nDelivery Discount -CA$2.99\nDiscount -
CA$1.30\nYou ordered from Spice of Nepal\nPicked up from\n13490 72 Ave, Surrey, BC V3W 2N8, Canada\nDelivered to\n9838
Whalley Blvd, Surrey, BC V3T 5S8, Canada",
  [(0, 16, 'DATE'), (71, 85, 'ORG'), (93, 101, 'MONEY')]),
 ("January 30, 2021\nThanks for ordering, Chinmaya\nHere's your receipt for A1 Coffee & Donair.\nTotal CA$17.95\n2 Dinair
and Rice CA$14.75\n Choice of Protein  Chicken CA$0.00\n Choice of Ingredients  Lettuce CA$0.00\n Cucumber  CA$0.00
Tomato  CA$0.00\n Onion CA$0.00  Tabbouleh CA$0.00\n Choice of Sauce  Tzatziki Sauce  CA$0.00\n Garlic Sauce  CA$0.00
Extra Sauce\n Hummus Sauce  CA$0.50\nSubtotal CA$29.50\nTax CA$1.73\nService Fee CA$2.95\nDelivery Fee CA$1.99\nDelivery
Discount -CA$1.99\nDiscount -CA$1.48\nPromotion -CA$14.75\nYou ordered from A1 Coffee & Donair\nPicked up from\n14795 108
Ave, Surrey, BC V3R 1V8, Canada\nDelivered to\n9838 Whalley Blvd, Surrey, BC V3T 5S8, Canada",
  [(0, 16, 'DATE'), (71, 89, 'ORG'), (97, 105, 'MONEY')])]
```

III. Named Entity Recognition Model

Named Entity Recognition[4] (NER) is a form of Natural Language Processing (NLP) concerned with identifying and categorizing key information in text. This type of model is useful for our purpose because we want it to be able to identify the tokens "Church's Chicken", "21-jan-2020" and "CA$20.15" and at the same time classify them as "Company/Organization", "Date" and "Money".

Spacy's NER[5] model is trained on examples of text and the labels one wishes to predict, and the predictions are made based on the weight values obtained during training. For the model to be able to generalize to unseen data, the training data has to be representative of the data we want to process, because a model such as ours, trained on receipts, would not be able to generalize to usual text. To train a model from scratch, which is able to generalize, one needs to have hundreds of training and validation examples.

To train the NER model with our annotated training examples, we first obtained a base configuration file from Spacy's documentation, which includes all the settings and hyperparameters depending on which components of Spacy will be used (in our case, only the NER component), the hardware being used (either CPU or GPU) and whether we want to optimize for efficiency or accuracy. After this, we ran the 'spacy train' command indicating the path for the training and validation examples, the configuration file and where to save the trained models. From the saved models, the best performance achieved is presented in the table below.

	Organization	**Date**	**Money**	**Overall**
F1-Score	0.87	0.81	0.80	0.83
Precision	0.96	0.91	0.85	0.90
Recall	0.79	0.72	0.76	0.76

Model Validation

After training the model, we wrote a program to take as input one or more receipts (either JPG or PDF files), extract the text according to the file type, use our model to predict the entity labels and return a dataframe with the information. We tested the model with some new receipts (not seen while training); the result is shown below. Our model is correctly identifying and classifying the invoice total and invoice date (in the cases where it shows in the extracted text); however, it is not generalizing as well for the business name.

[4] https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d
[5] https://spacy.io/usage/training#basics

	filename	businessName	invoiceDate	invoiceTotal	text
0	Invoice-12-19-2020-328814-1.pdf			NaN	Page 1 of 1\nv1.0\n24.99\n0.00\n24.99\n24.99\n...
1	IMG_1305.JPG		Jul 14,2022	25.20	- TRANSACTION RECORD -\n10009 136A ST\nSURREY\...
2	invoice.pdf	KYRA NICOLE MELENCIANO FELIZ	30 July 2022	33.59	Invoice / Facture\nPage 1 of 2 / Page 1 de 2\n...
3	Receipt_22Jun2022_215323.pdf	Hi Five Chicken (Marine Dr.)	June 22, 2022	67.16	Visa ••••3556\n6/22/22 9:53 PM\nJune 22, 2022\...
4	Testimage.jpg	Church's Chicken	07/01/22	50.24	Store # 3154\nChurch's Chicken\n120-9100 Blund...
5	QPP7MK3D1485100_20210212.pdf		2/12/2021	60.00	Receipt\nTransaction Date:\n2/12/2021\nLMI Or...

	filename	businessName	invoiceDate	invoiceTotal	text
0	IMG_1325.jpg			NaN	69\nBreka Bakery&Cafe-Fraser\n07/13/202201:54P...
1	IMG_1326.jpg	QUESADA BURRITOS &TACOS	Jul 12,2022	14.15	TRANSACTION RECORD --\nQUESADA BURRITOS &TACO...
2	IMG_1327.jpg	Gulberg Tandoor &		10.97	Gulberg Tandoor &\nDonair\n119-12578 72 Ave\nS...

	filename	businessName	invoiceDate	invoiceTotal	text
0	CA-FlightstoIndia-Aug18 - Sep04 2019.pdf		2006-2019	NaN	For any changes with your flight, date,\nroute...
1	bbbyreceipt.pdf		08/01/22	85.78	FREE SHIPPING on select orders\nview details›\...
2	IMG_1323.jpg	Ustaad G Indian Cuisine	29-Jul.-2022	38.85	Ustaad G Indian Cuisine\n10009 136A ST\nSURREY...
3	IMG_1324.jpg	Tim Hortons #2946	22/07/06	18.27	Tim Hortons #2946\nLangara Co1iege\nBC\n604)32...

IV. Clustering and Similarity Analysis

Apart from the creation of the model, we also performed similarity and clustering analysis to find similar receipts. The results of this analysis can be found here.

V. Summary and Recommendations

The purpose of this project was to extract key information from receipt images and pdf files. To achieve this goal, we have created a training dataset from the pdf and jpg files provided and trained a Named Entity Recognition model using python's Spacy library.

The model was validated using pdf and jpg receipts that were not seen during training and the results show that the model in most situations, is able to extract the invoice total and invoice date; however, the business name is more difficult to extract.

In further stages of this project, we'll attempt to better our model by including a higher quantity of training receipts and building a neural networks model. Additionally, we plan to group receipts based on similarities and provide this higher scale group (such as restaurant, e-commerce, transportations, etc) as a type of class for each receipt.