CS 410: Text Information Systems

Tech Review

Comparing OCR capabilities of Google Tesseract and AWS Textract

November 10, 2020

Praveen Pathri

ppathri2@illinois.edu

What is Optical Character Recognition (OCR)

Optical character recognition (OCR) is a technology solution for extracting text from typed or hand-written documents, PDF, image files or scanned documents and convert the textual information embedded within the images or documents into a machine-readable text format that can further be leveraged by applications like NLP and other business applications

In general, the OCR process involves the following steps

1. Image Preprocessing (enhancing or correcting the image)
2. Text Localization
3. Character Segmentation
4. Character Recognition
5. Post Processing

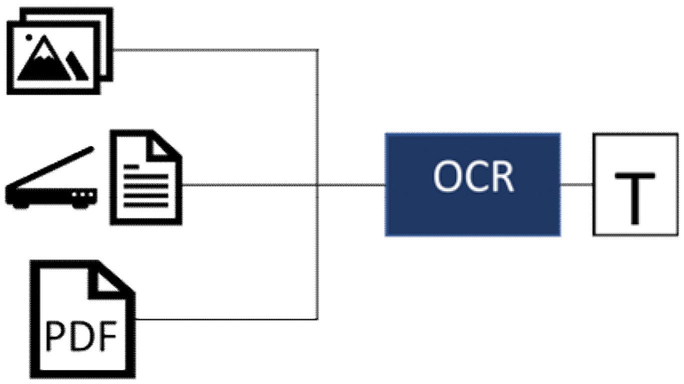


Fig-1.0 - Optical Character Recognition

Early OCR systems faced many challenges in accurately recognizing characters and extracting text out of images or documents, due to the variations in fonts, formats and unevenly spaced characters. Current generation of OCR systems are achieving state-of-the-art accuracies due to advancement in deep learning research, availability of large volumes of public datasets, synthetically generated data for training and availability of high-end compute-resources to train and tune models. Due to the various use-cases and utility for deep learning-based OCR, this technology has immense potential and wide spread usage

There are a lot of open source and commercial OCR systems available in the marketplace. The focus of this paper is to review the features and capabilities of open source Tesseract engine from Google and a commercial OCR solution from AWS – Textract. Each of these systems differ in their applicability and adoptability

Tesseract OCR

Tesseract is an open source optical character recognition engine available under Apache License 2.0. This system evolved over time, the initial version was developed by HP in 1980's and was open sourced in 2005. Google took over the project in 2006 and expanded the knowledge base of OCR to several other languages to recognize new characters.

Tesseract 4.0 version introduced the usage of a deep learning model – Long Short-Term Memory (LSTM) neural network, a configured neural network subsystem and a text line recognizer. It also brought new improvements to the full layout analysis, table detection, equation detection, segmentation search, language model and hand-written models. Tesseract can be directly used from command line or invoked using an API, from a language script like python

The new versions of Tesseract with the neural network-based recognition engines delivers significantly higher accuracy. The base models are trained on large corpus of training data for respective languages and font variations. Tesseract provides the capability to retrain the models with new set of data in the scenario the base models are not adequate

Tesseract provides multiple options for re-training

1. Fine Tune: For addressing recognition needs that are close to the existing trained data but have small differences like font variations. This training augmentation can be done with small set of data which is slightly different from the original trained data
2. Remove the top layer or some arbitrary number of layers from the network and retrain a new top layer using the new data, this is the next best option if fine tuning doesn't work. This approach can be used to train the network for a new language that is close to an existing trained language
3. Retrain from scratch – This can be time consuming and would require a large corpus of training set and compute resources

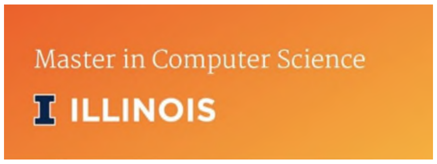
It is relatively easy to install and use Tesseract to extract embedded text into a machine-readable text format. While installing Tesseract the user can specify the languages required, out of the 120+ languages supported, and the respective trained data artifacts will be installed appropriately. The default directory for training data and other orientation and segmentation files used by Tesseract is "/tessdata". Explicit user patterns can be provided in the same directory to improve recognition of non-dictionary words like ID's etc.

Tesseract goes through the process of text localization and detection by creating bounding boxes for every region of text embedded within an image or a PDF file. As an output it recognizes the text in the bounding boxes and extracts them into machine readable format

The simplest command-line usage of tesseract is as below

```
$tesseract home.jpg output --oem 2 -l eng
```

```
Output: Master in Computer Science1! ILLINOIS
```



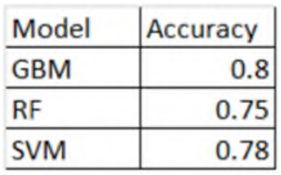
Master in Computer Science

I ILLINOIS

The character “I” represented using a special font is not recognized accurately, this can be addressed by training the network with a small set of training dataset, created using the above font variant

Extracting Table Data

```
$tesseract tableImage.jpg output --oem 2 -l eng
Output:  Model _ Accuracy
          GBM 0.8
          RF 0.75
          SVM 0.78
```



Model	Accuracy
GBM	0.8
RF	0.75
SVM	0.78

Tesseract provides two OCR engines that can be invoked in four different modes and 14 different Page segmentation modes

OCR Engine modes: --oem options

```
 0    Original Tesseract only
 1    Neural nets LSTM only
 2    Tesseract + LSTM
 3    Default, based on what is available
```

Page segmentation modes: --psm options

```
 0    Orientation and script detection (OSD) only
 1    Automatic page segmentation with OSD
 2    Automatic page segmentation, but no OSD, or OCR
 3    Fully automatic page segmentation, but no OSD. (Default)
 4    Assume a single column of text of variable sizes
 5    Assume a single uniform block of vertically aligned text
 6    Assume a single uniform block of text
 7    Treat the image as a single text line
 8    Treat the image as a single word
 9    Treat the image as a single word in a circle
10    Treat the image as a single character
11    Sparse text. Find as much text as possible in no particular order
12    Sparse text with OSD
13    Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific
```

In addition, Tesseract provides close to 700 parameters that can used to configure the extraction process and the output generated

AWS Textract

Textract is a commercial OCR system provided by AWS that can detect and extract text from Images and PDF files. Textract is based on the deep learning technology developed by Amazon's computer vision scientists to analyze large collection images and video files. This solution is provided by AWS as a fully managed service accessible using an API. Deep Learning models used in the detection and extraction of text are pretrained and are constantly retrained and updated for accuracy. New features are continually added to the API service

Textract can extract structured and unstructured data from documents. It can extract text from single and multi-page documents. The API supports synchronous processing for single page documents and asynchronous processing for multi-page documents. Currently Textract supports only English language and PDF, PNG and JPEG file formats. Textract provides a set of operations as part of the API, it bundles OCR, NLP and Text Mining capabilities as part of the overall solution

DetectDocumentText: Detects text in the input document, lines within the document and the words that make up the line of text, this operation returns the output as a block object

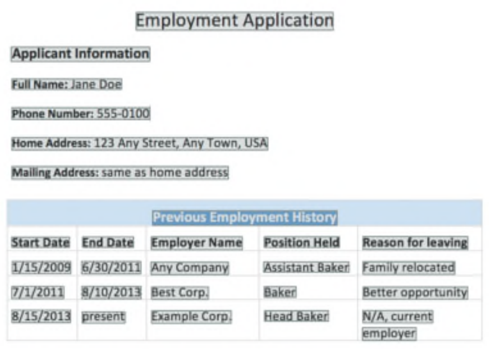
AnalyzeDocument: Analyzes the input document for relationship between detected items, this operation returns the following type of information

- Form data - Key-value pairs of objects
- Table data - Table and cell objects
- Line of text - Line and word objects

In each of the objects the following information is returned

- Lines and words of detected text
- Relationships between the lines and words of detected text
- Page that the detected text appears on
- Location of the lines and words of text on the document page

Below is an example of a sample extraction using AWS Textract



Employment Application

Applicant Information

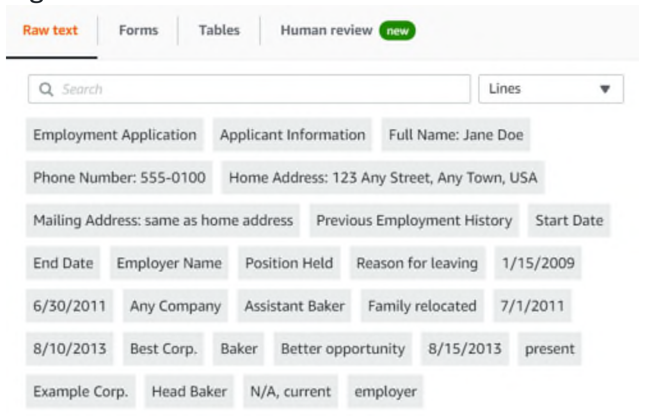
Full Name: Jane Doe

Phone Number: 555-0100

Home Address: 123 Any Street, Any Town, USA

Mailing Address: same as home address

Previous Employment History				
Start Date	**End Date**	**Employer Name**	**Position Held**	**Reason for leaving**
1/15/2009	6/30/2011	Any Company	Assistant Baker	Family relocated
7/1/2011	8/10/2013	Best Corp.	Baker	Better opportunity
8/15/2013	present	Example Corp.	Head Baker	N/A, current employer



Employment Application | Applicant Information | Full Name: Jane Doe | Phone Number: 555-0100 | Home Address: 123 Any Street, Any Town, USA | Mailing Address: same as home address | Previous Employment History | Start Date | End Date | Employer Name | Position Held | Reason for leaving | 1/15/2009 | 6/30/2011 | Any Company | Assistant Baker | Family relocated | 7/1/2011 | 8/10/2013 | Best Corp. | Baker | Better opportunity | 8/15/2013 | present | Example Corp. | Head Baker | N/A, current | employer

There is some limitation on file size, format and character size when using Textract, currently Textract has a minimum height limit of 15 pixels, at 150 DPI for the text to be detected.

Handwritten character recognition is still a challenge for both AWS Textract and Tesseract, while Tesseract can be trained on variation of fonts and handwritten characters the task of improving the accuracy of detection remains a task for future enhancement.

References:

https://github.com/tesseract-ocr/tesseract/wiki

https://www.pyimagesearch.com/2020/05/25/tesseract-ocr-text-localization-and-detection/

https://www.geeksforgeeks.org/text-localization-detection-and-recognition-using-pytesseract/

https://aws.amazon.com/textract/

https://docs.aws.amazon.com/textract/latest/dg/limits.html