

# How to Automate Data Extraction and Digitize Your Document Based Processes?



by Varghese P Kuruvilla 2 months ago

15 MIN READ

## Is manual Data Extraction still a thing in 2021?

The moment I read the title of the blog post, the first question that sprung to my mind was: 'Is Manual data Entry still a thing in 2021?.' A bit of research and I was pleasantly surprised at the scale of the problem. Many organizations still rely on manual data entry. Most of them don't invest in setting up an automated data extraction pipeline because manual data entry is extremely cheap and requires almost zero expertise. However, according to a 2018 Goldman Sachs report, the direct and indirect costs of manual data entry amounts to around \$2.7 trillion for global businesses.

A potential use case for an automated data extraction pipeline was during the COVID-19 pandemic. A lot of data such as the number of people tested, the test reports of each individual etc. had to be manually entered into a database.

Automating the process would have saved a lot of time and manpower.

## DRAWBACKS OF MANUAL DATA EXTRACTION:

1. **Errors:** When performing a tedious and repetitive task like manual data entry, errors are bound to creep in. Identifying and correcting these errors at a later stage might prove to be a costly affair.
2. **Slow Process:** When compared to automated data extraction, manual data entry is an extremely slow process and could stall the entire production pipeline.
3. **Data Security:** When dealing with sensitive data, a manual data entry process can lead to data leakages which could in turn compromise the system.

***Are you facing manual Data Extraction issues? Want to make your organization's data extraction process efficient? Head over to [Nanonets](#) and see for yourself about how Data Extraction from Documents can be automated.***

Get Started

## SECTION 1: THE DATA PIPELINE

To overcome the above mentioned drawbacks, almost all large organisations need to build a data pipeline. The main components of any data pipeline are aptly described by the acronym ETL (Extract, Transform, Load). Data Extraction involves extracting data from various sources, the data transformation stage aims to convert this data into a specific format and data loading refers to the process of storing this data in a data warehouse.

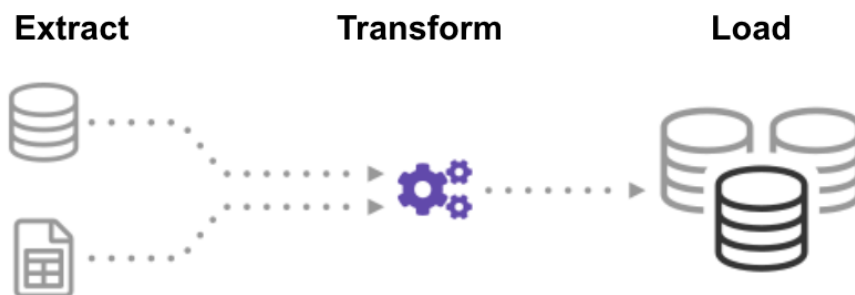


Fig 1. The ETL Process

Being the first stage in the pipeline, data extraction plays a crucial role in any organization. This post explores the various methods and tools that can be used to

perform data extraction and how Optical Character Recognition(OCR) can be employed for this task.

## SECTION 2: AUTOMATIC DATA EXTRACTION:

Almost all modern day data analytics requires large amounts of data to perform well. For example: Any organization would want to keep tabs on their competitors performance, the general market trends, customer reviews and reactions etc. A way to do this is to make use of **data extraction tools** that can scrape the web and retrieve data from various sources. The following section highlights a few popular off the shelf data extraction tools.

### 2.1: DATA EXTRACTION TOOLS

1) **Scrapy**: Scrapy is an open-source web crawler written in python. Let's go through a simple example that illustrates how even a complete novice can scrape the web using Scrapy. In the following example, I have used Scrapy to parse the title of the Nanonets blog page.

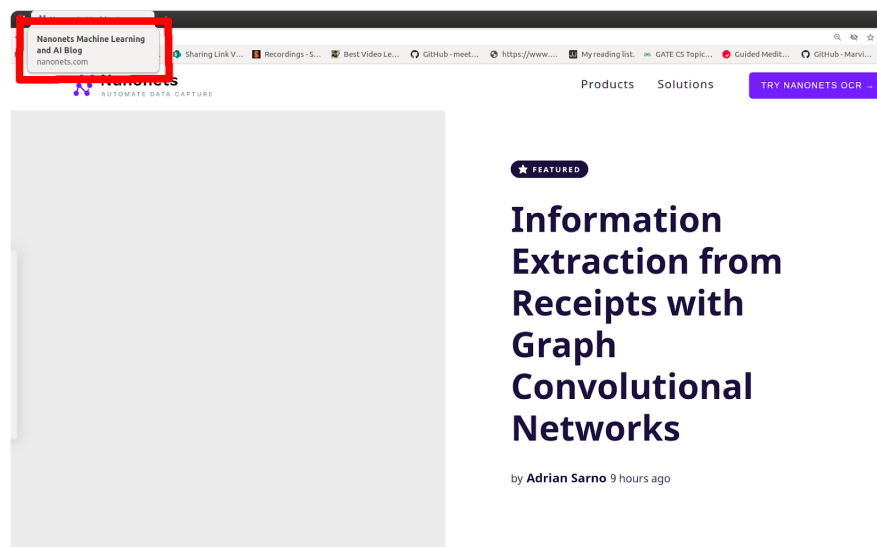


Fig 2. Title of the Nanonets blog page parsed using Scrapy

Although I used the Scrapy shell for the purpose of parsing, the same behaviour could be achieved using a python script.

```

[scrapy.spidermiddlewares.depth.DepthMiddleware']
2021-02-25 15:51:29 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2021-02-25 15:51:29 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2021-02-25 15:51:29 [scrapy.core.engine] INFO: Spider opened
2021-02-25 15:51:30 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://nanonets.com/blog/> (refere
r: None)
2021-02-25 15:51:32 [asyncio] DEBUG: Using selector: EpollSelector
[s] Available Scrapy objects:
[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s] crawler <scrapy.crawler.Crawler object at 0x7f1a9bad74d0>
[s] item {}
[s] request <GET https://nanonets.com/blog/>
[s] response <200 https://nanonets.com/blog/>
[s] settings <scrapy.settings.Settings object at 0x7f1a9b8b8750>
[s] spider <DefaultSpider 'default' at 0x7f1a9b3f7b10>
[s] Useful shortcuts:
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are follow
ed)
[s] fetch(req) Fetch a scrapy.Request and update local objects
[s] shelp() Shell help (print this help)
[s] view(response) View response in a browser
2021-02-25 15:51:32 [asyncio] DEBUG: Using selector: EpollSelector
In [1]: response.css('title::text')[0].get()
Out[1]: 'Nanonets Machine Learning and AI Blog'
In [2]:

```

Fig 3. Title of the Nanonets blog page parsed by Scrapy

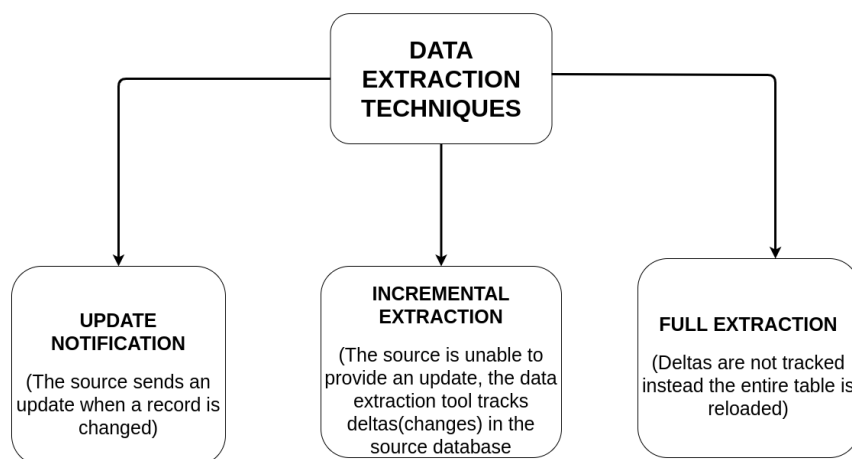
The tool is extremely intuitive and elements from any HTML page can be parsed using CSS. The only downside to the tool from the point of view of a beginner was that parsing dynamic web pages was pretty challenging.

2) Octoparse, Outwit hub, Parsehub etc are other open source tools that provide an intuitive GUI for web scraping.

Apart from these open source tools there are companies that are dedicated to performing data extraction. Small organizations that don't have the resources to build custom data extraction pipelines can outsource the data extraction process by making use of these data extraction services.

## 2.2: DATA EXTRACTION TECHNIQUES

The flowchart given below provides a brief explanation about a few data extraction techniques.



Flowchart1. Data extraction techniques

The following sections explore the use of Optical Character Recognition (OCR) to perform the task of data extraction.

***Are you facing manual Data Extraction issues? Want to make your organization's data extraction process efficient? Head over to [Nanonets](#) and see for yourself how Data Extraction from Documents can be automated.***

Get Started

### **SECTION 3: AUTOMATIC DATA EXTRACTION USING OCR:**

Optical Character Recognition (OCR) is a technology that identifies characters from printed or handwritten material. By setting up a data extraction pipeline using OCR, organizations can automate the process of extracting and storing data.

#### **THE HEART OF ANY OCR SYSTEM:**

Modern OCR tools come with an array of data preprocessing (noise removal, binarization, line segmentation) and postprocessing steps. However, at the core of any OCR system lies two major components:

1. A Feature Extractor and
2. A Classifier

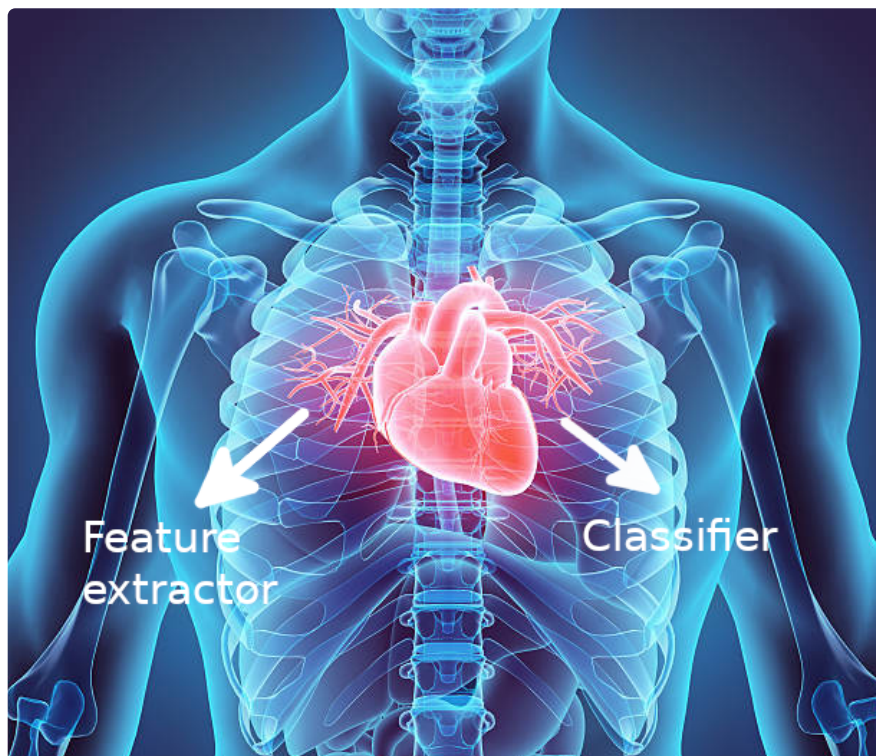


Fig 4

The feature extractor extracts features corresponding to each lexeme (character/word). These extracted features are fed as inputs to the classifier that determines the probability of the lexeme belonging to a specific class.

## TRADITIONAL APPROACHES TO SOLVING THE OCR PROBLEM:

1. **Template Matching:** A set of templates (images of each character of the alphabet) are collected and stored. Each character of the input image is then matched against this collection of templates. Each comparison is associated with a similarity measure using which the best possible matches are identified.

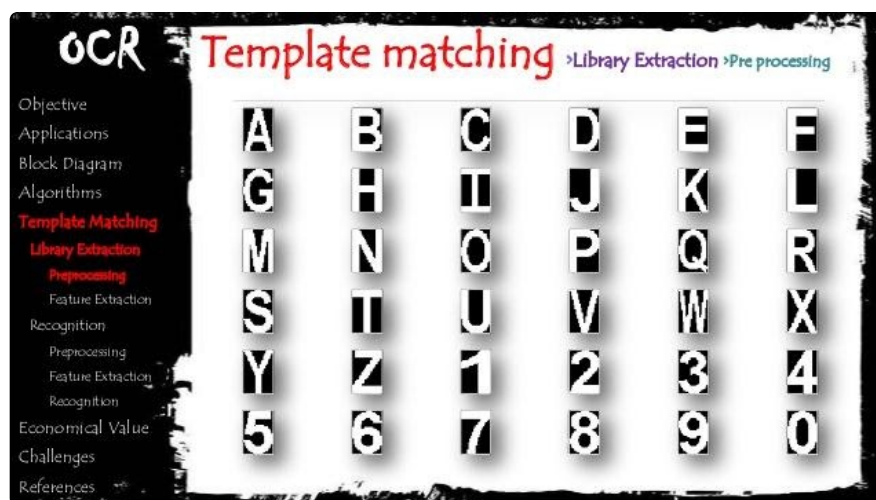


Fig 5. List of templates for the English Alphabet (Source: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.slideshare.net%2Fvj84529%2Focr-color&psig=AOvVaw0u4z1m4DwYNIFQEFKlQLqH&ust=1613545352470000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCKiG8ljr7e4CFQAAAAAdAAAA>)

**Rule based Methods:** As children we were taught to recognise the character 'H' as two vertical lines with a horizontal line connecting them. Intuitively this is what rule based methods try to achieve. Certain structural features are extracted from the input images and a rule based system is used to classify them.

Apart from the above-mentioned approaches, various other methods have been developed for performing OCR based on traditional computer vision - e.g. zonal OCR. However, almost all of them have been replaced by or supplemented by Deep Learning.

Now that we have an idea of what OCR is and some of the traditional approaches used to perform OCR, let's go deeper...

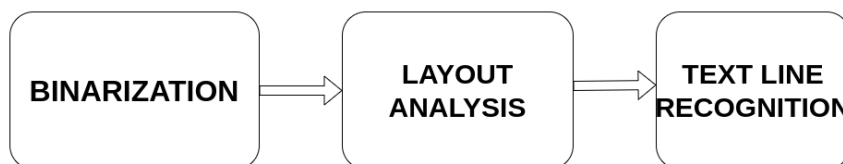


(Source: <https://memegenerator.net/instance/57413687/inception-di-caprio-we-need-to-go-deeper>)

## SECTION 4: OCR TOOLS

Let's look into some of the free open source state of the art OCR tools:

1. **Tesseract:** Tesseract was initially developed by HP and was released as an open source software in 2005. Since then, its development has been taken over by Google. There are numerous tutorials explaining all the details of tesseract OCR and how it can be used. The following blog on Nanonets provides a comprehensive review of the same <https://nanonets.com/blog/ocr-with-tesseract/#introduction>
2. **OCRopus:** OCRopus is a collection of tools used for performing OCR on images. The general pipeline of OCRopus contains three main blocks as shown in the figure below.



Flowchart2. General pipeline of OCRopus

OCRopus is a full GUI engine and can optionally use tesseract in the backend for performing OCR.



**3. Calamari OCR:** Calamari OCR is a relatively new line recognition software that uses deep neural networks implemented in TensorFlow. When compared to Tesseract and OCRopus, Calamari OCR has few explanations detailing its network architecture and its inner workings. This seems like a good point to formalize the OCR problem and peer at it through the eyes of Calamari.

Let's assume that we want to perform Optical Character recognition on the word "Speed" using a Deep Neural Network(DNN) . Let's also assume that we have created a DNN using Convolutional Neural Nets(CNNs) and Long short-term memory(LSTMs) to perform this task. Our network predicts output probabilities associated with each class at every timestep.

For example: In an ideal scenario

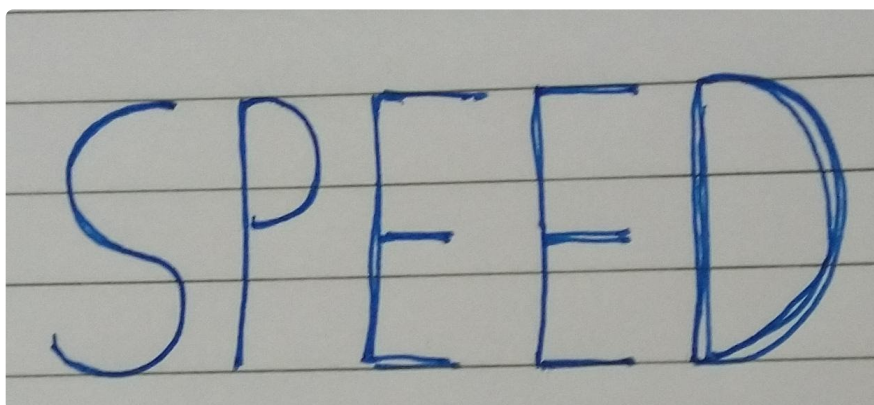


Fig 6. Input Image fed to the Neural Network

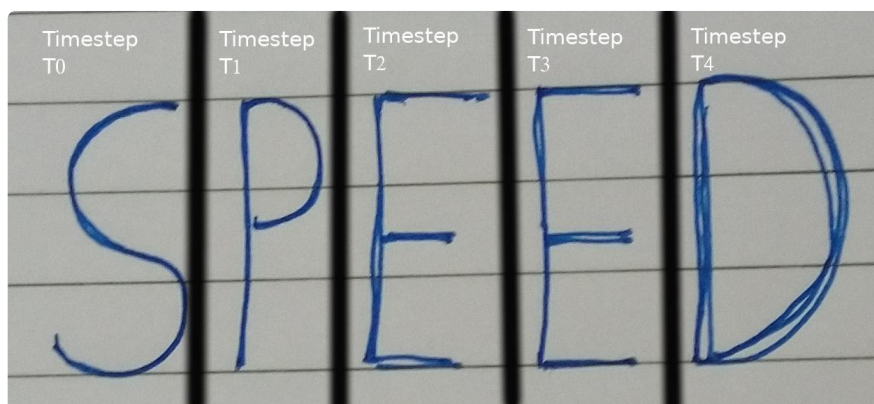


Fig 7. The output(if all goes well)

The table below shows the possible probability values associated with each time step.

	T0	T1	T2	T3	T4
P(a)	0.001	0.002	0.01	0.01	0.001



P(b)	0.001	0.003	0.003	0.002	0.002
P(c)	0.005	0.005	0.002	0.001	0.001
P(d)	0.002	0.001	0.001	0.003	0.7
P(e)	0.001	0.002	0.7	0.8	0.002
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
P(p)	0.003	0.8	0.002	0.004	0.001
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	
P(s)	0.7	0.008	0.002	0.001	0.007
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	

Table 1. Probabilities associated with each class

Taking the maximum probability under each timestep, we get the required output i.e SPEED. What could go wrong with this approach? Let's take a moment to think about an assumption we have made in our reasoning namely the alignment of each timestep.

We assumed that each timestep occurs exactly between successive alphabets. The output would have been very different if the neural network decides to align the timesteps as shown in figure 8.

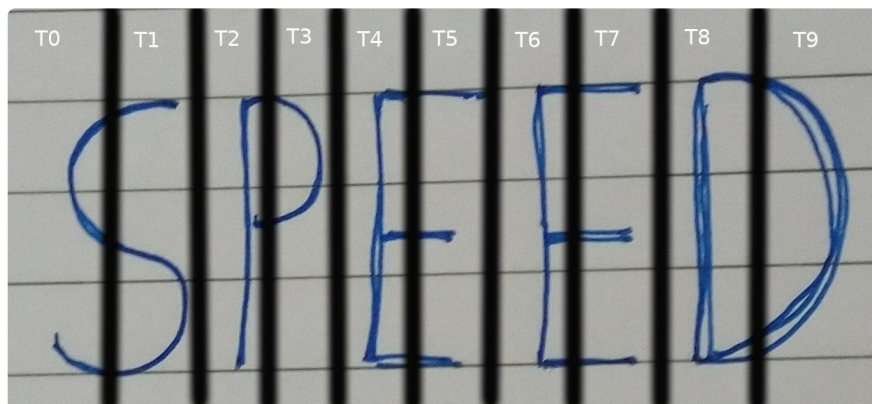


Fig 8. Misaligned timesteps

In this scenario, the neural network might predict SSPPEEEEDD as the output. Secondly, preparing the training data for the neural network might prove to be extremely tedious. We would need to specify the exact pixel location at which each alphabet starts and ends.

What seemed like a straightforward task is proving to be extremely frustrating. The problem of misaligned timesteps and training data annotation can be solved by introducing a new loss function.

### Connectionist Temporal Classification (CTC)



(Source: [https://www.google.com/search?q=memeanimals.com+i+must+go+my+people+need+me&tbm=isch&source=iu&ictx=1&fir=C8adpx9pd63\\_pM%252C6SVZE5KvuruZIM%252C\\_&vet=1&usg=kR44ME7ZPnrJBaiK3LJUtr-hYlyWw&sa=X&ved=2ahUKEwiiiZ2XqonvAhUkmeYKHQpbCgcQ9QF6BAgMEAE#imgsrc=C8adpx9pd63\\_pM](https://www.google.com/search?q=memeanimals.com+i+must+go+my+people+need+me&tbm=isch&source=iu&ictx=1&fir=C8adpx9pd63_pM%252C6SVZE5KvuruZIM%252C_&vet=1&usg=kR44ME7ZPnrJBaiK3LJUtr-hYlyWw&sa=X&ved=2ahUKEwiiiZ2XqonvAhUkmeYKHQpbCgcQ9QF6BAgMEAE#imgsrc=C8adpx9pd63_pM))

### CTC helps us in the following ways:

1. Using the CTC loss, we can train the network without having to specify the pixel wise position of each alphabet. This is achieved by introducing a new character '-'. '-' is used to indicate that no character is seen at a given timestep.

Using this special character '-', the ground truth could be modified to account for all possible positions where the word "speed" occurs in the image. For example, the word "speed" could be written as "---speed", "--speed-", "-speed-", "speed--". Similarly, since we don't know how much space each alphabet might take, we add character repetitions to account for varying character lengths i.e. "speed" can be written as "---sspeed", "---ssspeer", and so on. In the case of actual character repetitions in the ground truth, we need to add a '-' between the characters that are repeated. Thus the word "speed" can be encoded in the following ways: "---spe-ed", "--spe-ed-", "-spe-ed-", "spe-ed--", "--sspe-ed", etc. We calculate the score for each possible encoding and the sum of all the individual scores gives us the loss for each (image, ground truth) pair.

2. Using the CTC decoder is much simpler. Let's say that the decoder outputs "ssppe-eee-dd". We can simply discard duplicates i.e "ssppe-eee-dd" becomes "spe-e-d". Finally, we remove the '-' characters to obtain the word "speed".

I found the following resources extremely helpful when learning about the CTC loss. <https://distill.pub/2017/ctc/>  
<https://dl.acm.org/doi/abs/10.1145/1143844.1143891>

Implementing the network is straightforward. According to the paper(<https://arxiv.org/pdf/1807.02004.pdf>), the default network has the following specifications:

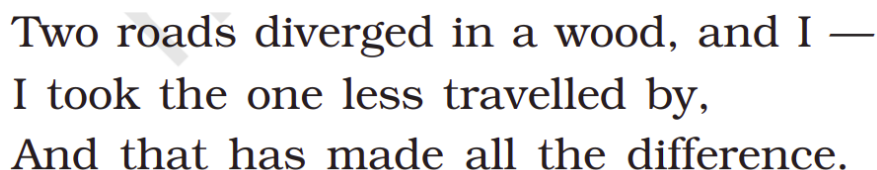
**Architecture:** Conv layer -> Max-Pooling -> Conv layer -> Max Pooling -> LSTM.

**Loss:** CTC loss

**Optimizer:** Adam with a learning rate of 0.001

Phew! That was a lot of theory. Let's get our hands dirty by implementing Optical Character recognition using Calamari.

Getting started from the Calamari github page <https://github.com/Calamari-OCR/calamari> is an easy task and I had no problem during the installation process. I decided to use a model trained on the uw3-modern-english dataset. Figure 9 shows the input fed to the network and Figure 10 shows the corresponding output.



Two roads diverged in a wood, and I —  
 I took the one less travelled by,  
 And that has made all the difference.

Fig 9. Input image to Calamari

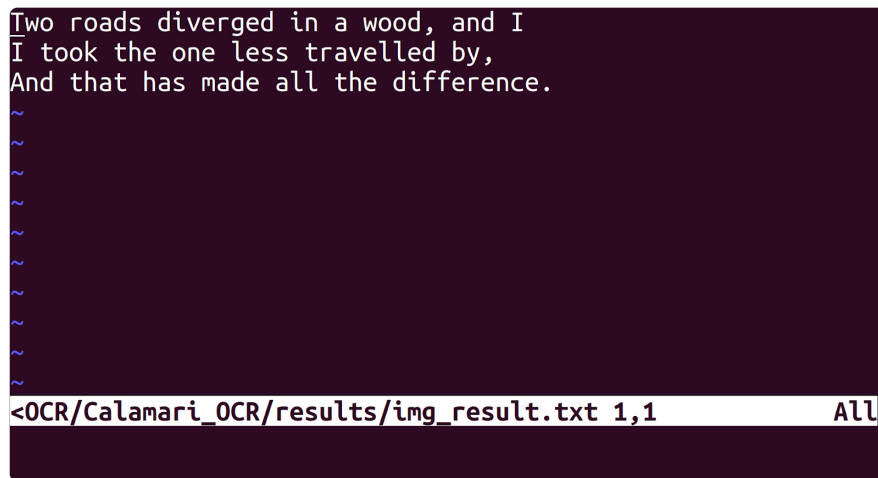


Fig 10. Output from Calamari OCR

Calamari produced the output (Fig 10) with a confidence of 97%. It performs very well in most cases and can easily be fine-tuned to suit your specific use case.

**NOTE:** Calamari performs OCR on a single line of text at a time. If you want to perform OCR on an entire document some preprocessing (layout analysis, line segmentation etc.) is required prior to feeding the image to Calamari.

Apart from the abovementioned free open source OCR tools, there are several paid tools such as Google cloud vision, Microsoft Computer Vision API and Amazon Textract.

The next section talks about how OCR can be used to solve practical problems in various industries and organizations.

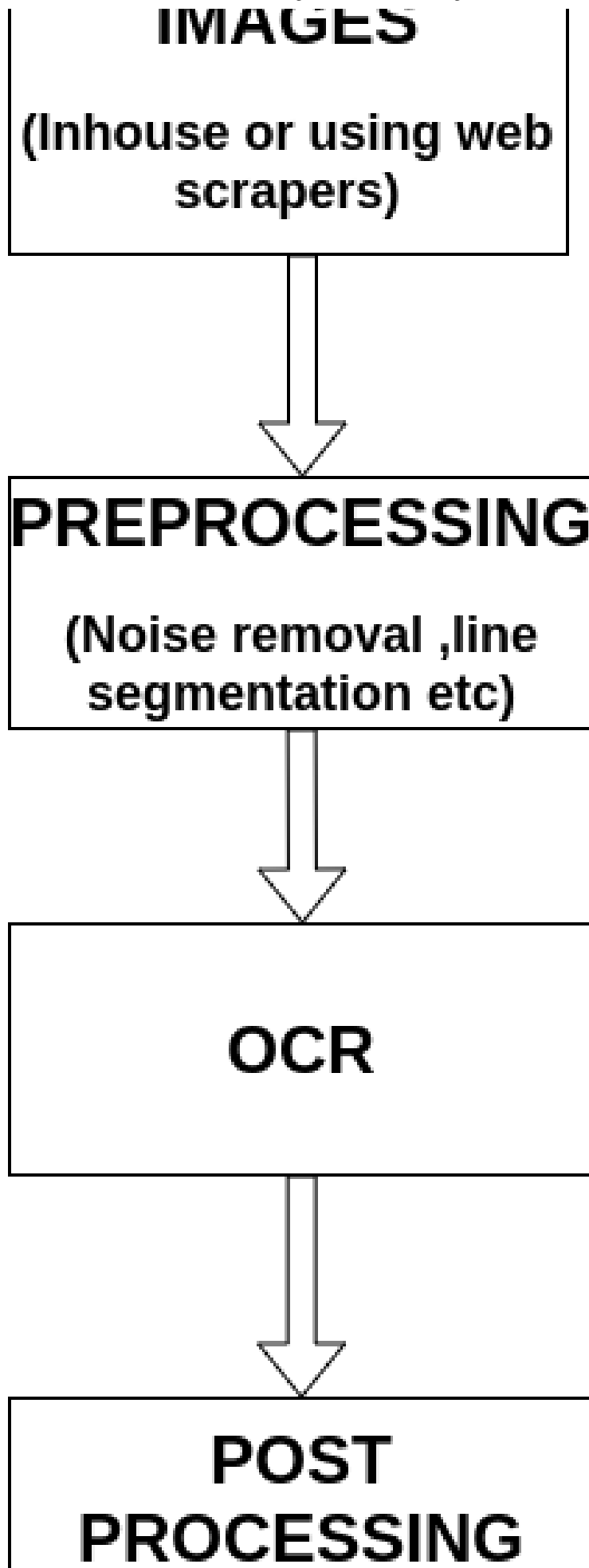
***Do you have a Data Extraction requirement? Head over to [Nanonets](#) and see how you can automate Data Extraction from documents like PDFs, Receipts, Invoices, Forms and More.***

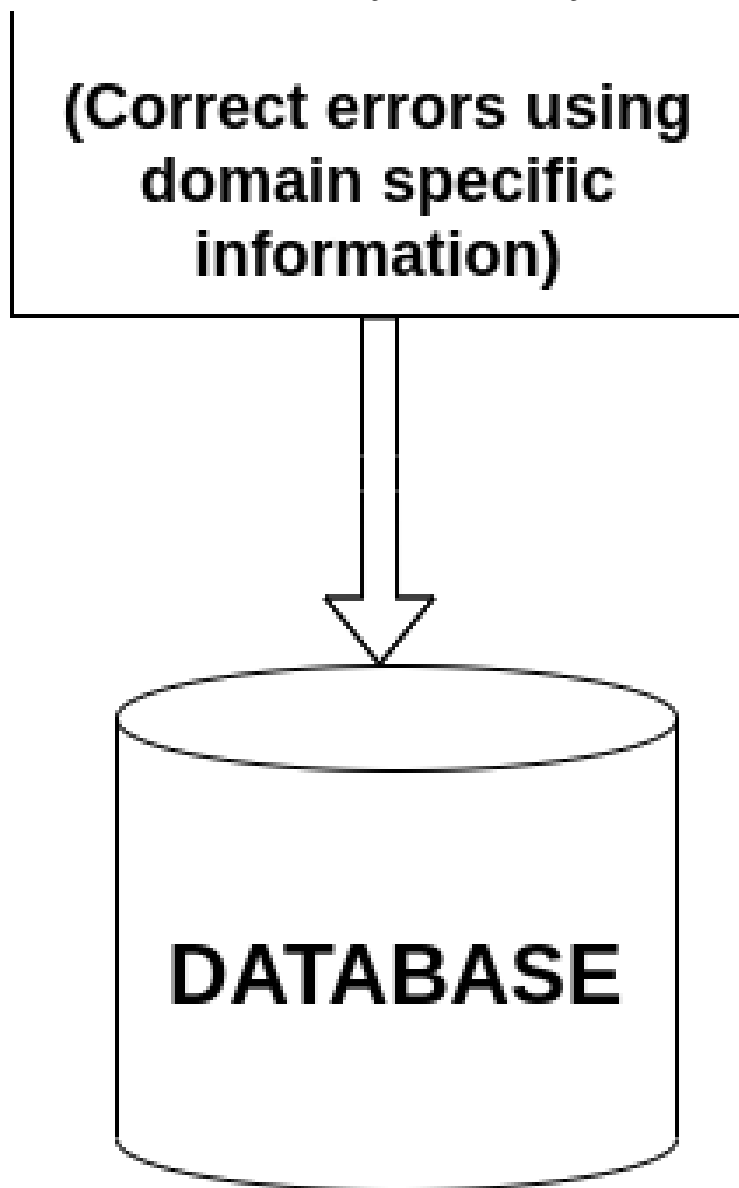
Get Started

## SECTION 5: PRACTICAL USE CASES OF DATA EXTRACTION USING OCR:

Using the generic OCR pipeline shown in FlowChart3, some of the problems that can be solved using OCR are elucidated below.







Flowchart 3. OCR Pipeline

### OCR based Data Extraction Techniques for the Healthcare Sector

**The problem:** Ever since I was a little boy, the following sequence of steps would be performed whenever I visited the hospital. The receptionist would first ask for my ID number. She would then dive into a huge stack of diaries which were sorted in some fashion. Usually, after a prolonged period of searching, I would get my diary and a token number. The doctor would examine the cause of my illness and write down a prescription in my diary. Upon handing over the prescription to the pharmacy, I would receive the required medicines. I assume that this is the routine followed in most local hospitals within the country.

**Solution:** Using our OCR pipeline, all the information could be digitized and stored in a database. A simple way to implement this would be to hand over forms to each



patient which are scanned and fed into the OCR pipeline. The advantages of doing this are manifold:

1. Patients' medical history can be stored in a common database which the doctors can access at their will. This information could help the doctor diagnose the illness.
2. The hospital could analyze the data and allocate its resources accordingly. For example: If the data indicates that the gynaecology section has a maximum number of patients, the hospital can choose to employ more doctors and nurses in this section.

### Potential pitfalls:

1. As you might have guessed, deciphering doctors' prescriptions using OCR is no small challenge. However, by using good quality training data along with some domain-specific information (names of well-known medicines) in the post-processing step, the solution can be made robust to most errors.

### Automated Data Extraction Services that can benefit the Government

**The Problem:** During the past year, the COVID-19 pandemic has brought along with it an array of problems. I was quite surprised to learn that manual data entry was one of them. When the pandemic was at its peak, lakhs of tests were being conducted every day and all the results had to be manually entered into a database.

**Solution:** OCR could have been easily employed in this scenario. A scanned copy of the lab report can be fed into the OCR pipeline. For example, Fig 11 shows the test report which is fed as an input to the pipeline and Fig 12 is the corresponding result.

LPL - PRODUCTION TEST COLLECTION  
CENTRE  
SECTOR - 18, BLOCK-E ROHINI  
DELHI 110085

Name	: DUMMY	Collected	: 24/3/2020 10:06:00AM
Lab No.	: MKPDP1035	Age: 18 Years	Gender: Male
		Received	: 24/3/2020 1:19:22PM
		Reported	: 24/3/2020 1:23:21PM
A/c Status	: P	Ref By	: Dr.Veena Bora
		Report Status	: Final

Test Name	Results	Units	Bio. Ref. Interval
COVID-19 Virus QUALITATIVE PCR (Real Time PCR)	Negative		Negative

#### Interpretation

RESULT	REMARKS
Positive	RNA specific to SARS-CoV-2 Detected
Negative	RNA specific to SARS-CoV-2 NOT detected
Inconclusive	Inconclusive. This could be due to low viral load in the sample. in the sample. A repeat sample is recommended for confirmation.

Fig 11. Scanned copy of a COVID test report(<https://www.lalpathlabs.com/SampleReports/N228.pdf>)

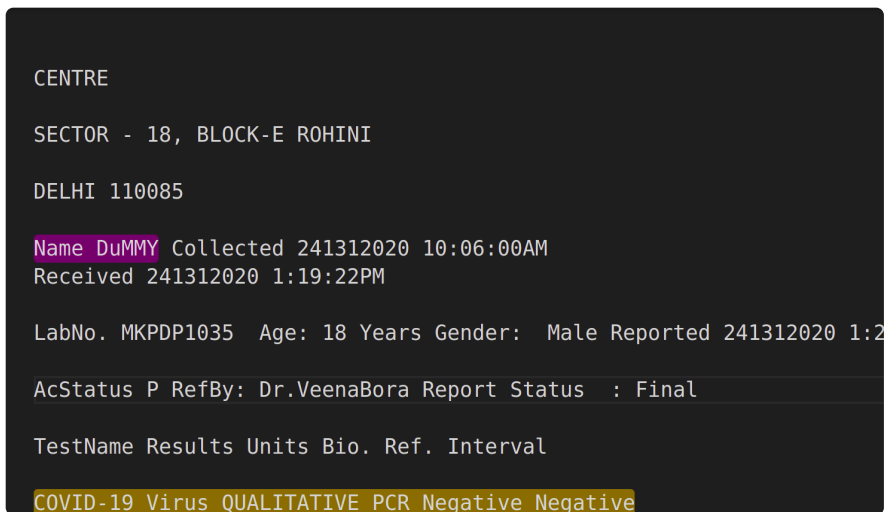


Fig 12. Result of OCR on the COVID test report

The problem could be simplified further by concentrating on the fields that are important and ignoring the rest. In this case, the Name of the individual and the result of the test must be extracted reliably. Since the results of the test are binary i.e. either negative or positive, they could be matched using regular expressions. Similarly, the name field could be replaced by a unique identification number to ensure reliable character recognition.

## OCR Based Data Extraction Software for Invoice Automation

**THE PROBLEM:** Deep within the accounts section of any organization lies a group of people whose job is to manually enter data from invoices into the company's database. This is a highly repetitive and mundane task that can be automated thanks to our OCR pipeline.

**SOLUTION:** Performing OCR on the given invoice can automate the task of manual data entry. A lot of work has already been done in this area and developing a robust solution mainly hinges upon reliably extracting tables and amounts accurately from the invoice.

The following blogposts <https://nanonets.com/blog/table-extraction-deep-learning/> and <https://nanonets.com/blog/extract-structured-data-from-invoice/> provide comprehensive explanations of the same.

## SECTION 6: THE LATEST RESEARCH:

### 1. ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation(<https://arxiv.org/abs/2003.10557>)(CVPR-2020):

This paper addresses the problem of handwritten text recognition (HTR). Although state of the art OCR tools performs well on printed text, handwritten text

recognition is still a developing field. The authors attribute this gap to the lack of training data i.e., the lack of annotated handwritten text. The authors propose a DNN that can generate handwritten images of varying styles.

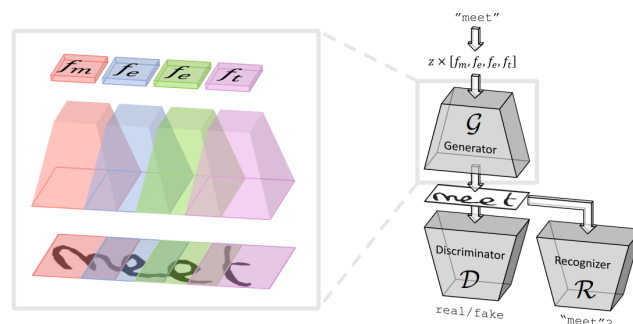


Figure 2: *Architecture overview* for the case of generating the word "meet". **Right:** Illustration of the entire ScrabbleGAN architecture. Four character filters are concatenated ( $f_e$  is used twice), multiplied by the noise vector  $z$  and fed into the generator  $\mathcal{G}$ . The resulting image is fed into both the discriminator  $\mathcal{D}$  and the recognizer  $\mathcal{R}$ , respectively promoting style and data fidelity. **Left:** A detailed illustration of the generator network  $\mathcal{G}$ , showing how the concatenated filters are each fed into a class-conditioned generator, where the resulting receptive fields thereof are overlapping. This overlap allows for adjacent characters to interact, enabling cursive text, for example.

Fig 13. Architecture of ScrabbleGAN

Fig 13. Illustrates the architecture of ScrabbleGAN. The generator generates synthetic images which are fed to a recognizer in addition to the discriminator. The discriminator forces the generator to generate real looking images while the recognizer makes sure that meaningful words are generated by the generator.

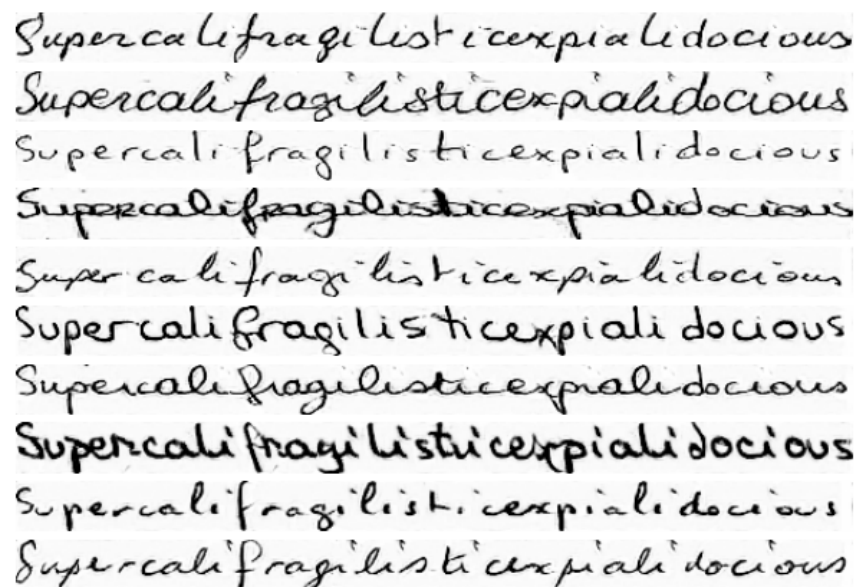


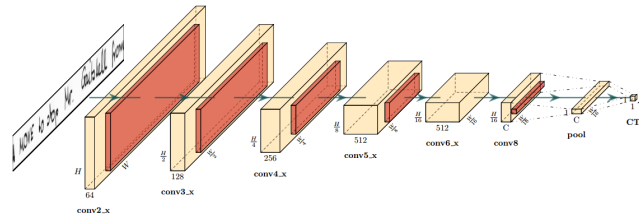
Fig 14. Different styles of the word "supercalifragilisticexpialidocious"

The network is trained in a semi supervised manner and two metrics namely the Word Error Rate (WER) and normalized edit distance(NED) are used for evaluation.

## 2. OrigamiNet: OrigamiNet: Weakly-Supervised, Segmentation-Free, One-Step, Full Page Text Recognition by learning to unfold(<https://arxiv.org/abs/2006.07491>)(CVPR-2020):

The very first OCR architectures tried to segment each character from the input image and classify each segmented character. This progressed to segmentation free approaches where an entire word was segmented and classified. Today, most state-of-the-art approaches operate on an entire line of text.

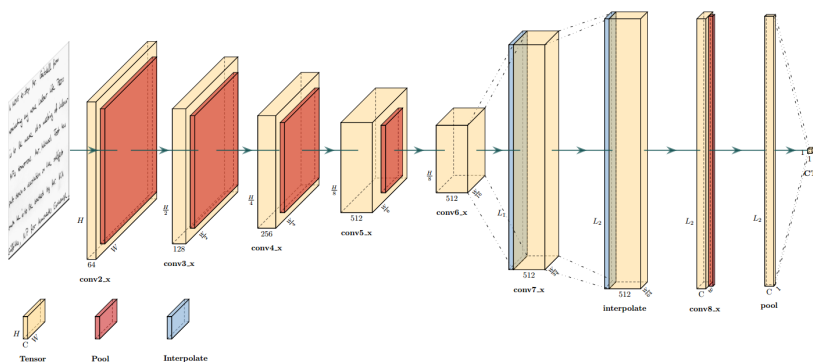
In this paper, the authors propose a simple set of operations that enable OCR to be performed on an entire page in a single forward pass through the network. The major constraint in performing OCR on an entire page is that the CTC loss function requires the input to be 1D. This is clearly illustrated in Fig 15, where the input is down sampled and converted to 1D before the loss calculation stage.



(a) A generic four stage fully convolutional single line recognizer, input is a single line image, training is done using the CTC loss function. Backbone CNN can be any of the ones presented in Table 2. Input gets progressively down-sampled, then converted into 1D by average pooling along the vertical dimension right before the loss calculation. (Figures created via PlotNeuralNet [13])

Fig 15. A fully convolutional single line recognizer

Since CNNs perform well on tasks such as image to image translation, the authors make use of a CNN to learn the 2D to 1D transformation. The feature map from the generic fully convolutional neural network is upsampled vertically and downsampled horizontally in two successive stages before the pooling operation is performed.



(b) Here we convert the fully convolutional single-line recognizer into an OrigamiNet multi-line recognizer; comparing the two figures shows that the main change introduced is up-scaling vertically in two stages, and at the same time, down-scaling horizontally. We obtain a feature-map that is tall and narrow (the shape of one very long vertical line, length  $L_2$ ). After that we proceed exactly as above, average pooling over the short dimension,  $w$  (of the new line not the original image) then using the CTC loss function to drive the training process.

Fig 16. Generic CNN used for performing OCR on a single line of text augmented with additional stages to perform multi-line recognition

The final tall feature map contains all of the lines of text from the input image. The authors argue that providing the model with sufficient spatial capacity allows it to easily learn the required 2D to 1D transformation.

The authors evaluate their work by using standard CNNs such as ResNet, VGG and GTR

## CONCLUSION:

In this post we looked at data extraction in detail and how Optical character recognition can be used to solve this problem. Section1 contains a brief introduction of the data extraction problem. In Section2 we took a look at some data extraction tools and techniques. Section3 gave an overview of the OCR problem and some of the traditional methods used to solve it. In Section4 we explored some popular open-source tools used to perform OCR and understood the CTC loss function. Section5 contains several practical use cases where OCR can be used to solve the data extraction problem. Finally, we looked at the current state of the art research in the field of OCR.

## You might be interested in our latest posts on:

- [AWS Textract](#)
- [Data Extraction](#)
- [Best OCR Software](#)
- [PDF to Excel](#)
- [BPO Automation](#)
- [Invoice Processing](#)
- [Fuzzy Matching](#)
- [Fuzzy Logic](#)
- [Google Cloud Vision](#)
- [Invoice Management](#)
- [Purchase Order Matching or PO Matching](#)
- [Three-way Matching](#)
- [Payment Reconciliation](#)
- [AP Automation](#)