



Search



This member-only story is on us. [Upgrade](#) to access all of Medium.

★ Member-only story

# Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER)

Key concepts, examples, and Python implementation of measuring Optical Character Recognition output quality



Kenneth Leung · [Follow](#)

Published in Towards Data Science

7 min read · Jun 24, 2021



Listen



Share

... More



Since you **cannot improve what you do not measure**, these metrics serve as a vital benchmark for the iterative improvement of your OCR model.

Photo by Brett Jordan on Unsplash

## Contents

In this article, we will look at two metrics used to evaluate OCR output, namely

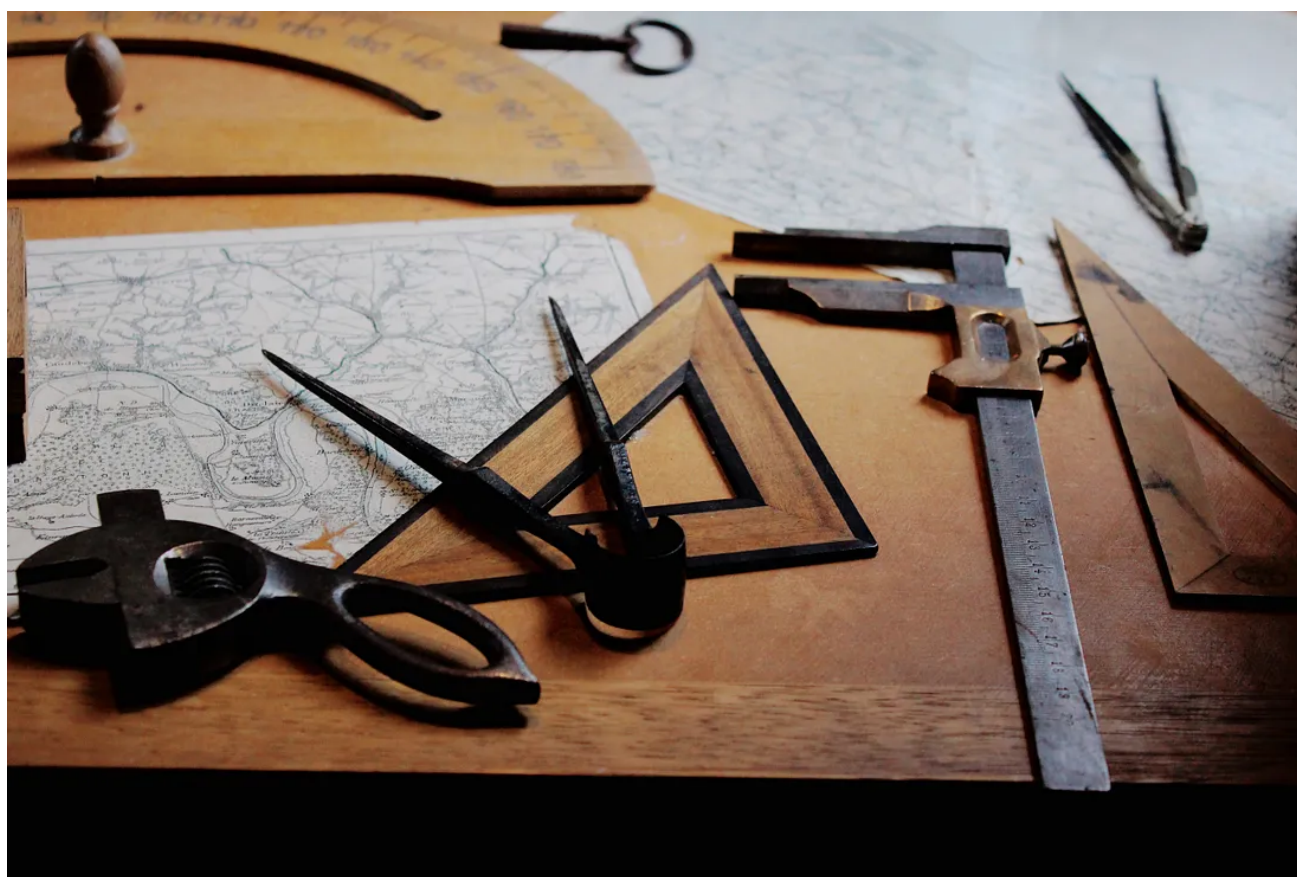
(1) Importance of Evaluation Metrics

(2) Error Rates and Levenshtein Distance

(3) Character Error Rate (CER)

(4) Word Error Rate (WER)

(5) Python Example (with TesseractOCR and fastwer)



A common intuition is to see how many characters were misspelled. While this is correct, the actual error rate calculation is more complex than that. This is because the OCR output can have a different length from the ground truth text.

Photo by Fleur on Unsplash

Furthermore, there are three different types of error to consider:

- **Substitution error:** Misspelled characters/words
- **Deletion error:** Lost or missing characters/words
- **Insertion error:** Incorrect inclusion of character/words

STEAM

STEAL

 Substitution

STEAM

TEAM

 Deletion

STEAM

STREAM

 Insertion

## Examples of the three basic errors | Image by Author

The question now is, how do you **measure the extent of errors** between two text sequences? This is where Levenshtein distance enters the picture.

**Levenshtein distance** is a distance metric measuring the difference between two string sequences. It is the *minimum number of single-character (or word) edits* (i.e., insertions, deletions, or substitutions) required to change one word (or sentence) into another.

For example, the Levenshtein distance between “*mitten*” and “*fitting*” is 3 since a minimum of 3 edits is needed to transform one into the other.

1. *mitten* → *fitten* (substitute **m** with **f**)
2. *fitten* → *fittin* (substitute **e** with **i**)
3. *fittin* → *fitting* (insert **g** at the end)

The more different the two text sequences are, the higher the number of edits needed, and thus the larger the Levenshtein distance.

## Character Error Rate (CER)

### (i) Equation

CER calculation is based on the concept of Levenshtein distance, where we count the **minimum** number of character-level operations required to **transform the ground truth text** (aka *reference text*) into the OCR output.

It is represented with this formula:

$$CER = \frac{S + D + I}{N}$$

Character Error Rate (CER) formula



where:

- **S** = Number of Substitutions
- **D** = Number of Deletions
- **I** = Number of Insertions
- **N** = Number of characters in reference text (aka ground truth)

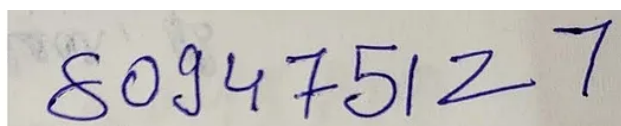
*Bonus Tip: The denominator  $N$  can alternatively be computed with:*

$N = S + D + C$  (where  $C$  = number of **correct** characters)

The output of this equation represents the **percentage** of characters in the reference text that was **incorrectly** predicted in the OCR output. The lower the CER value (with 0 being a perfect score), the better the performance of the OCR model.

## (ii) Illustration with Example

Let's look at an example:



Handwritten ID sample | Source: <https://commons.wikimedia.org/wiki/File:Test-OCR-Handwritten.jpg>

- **Ground Truth Reference Text:** 809475127
- **OCR Transcribed Output Text:** 80g475Z7

Several errors require edits to **transform OCR output into the ground truth**:

1. g instead of 9 (at reference text character 3)
2. Missing 1 (at reference text character 7)
3. Z instead of 2 (at reference text character 8)

With that, here are the values to input into the equation:

- Number of Substitutions (**S**) = 2
- Number of Deletions (**D**) = 1
- Number of Insertions (**I**) = 0
- Number of characters in reference text (**N**) = 9

Based on the above, we get  $(2 + 1 + 0) / 9 = 0.3333$ . When converted to a percentage value, the CER becomes 33.33%. This implies that every 3<sup>rd</sup> character in the sequence was incorrectly transcribed.

We repeat this calculation for all the pairs of transcribed output and corresponding ground truth, and **take the mean** of these values to obtain an overall CER percentage.

### (iii) CER Normalization

One thing to note is that CER values can exceed 100%, especially with many insertions. For example, the CER for ground truth 'ABC' and a longer OCR output 'ABC12345' is 166.67%.

It felt a little strange to me that an error value can go beyond 100%, so I looked around and managed to come across an [article by Rafael C. Carrasco](#) that discussed how **normalization** could be applied:

*Sometimes the number of mistakes is divided by the sum of the number of edit operations ( $i + s + d$ ) and the number  $c$  of correct symbols, which is always larger than the numerator.*

The normalization technique described above makes CER values fall within the range of 0–100% all the time. It can be represented with this formula:

$$CER_{normalized} = \frac{S + D + I}{S + D + I + C}$$

### Normalized CER formula

where  $C$  = Number of **correct** characters

#### (iv) What is a good CER value?

There is no single benchmark for defining a good CER value, as it is highly dependent on the use case. Different scenarios and complexity (e.g., printed vs. handwritten text, type of content, etc.) can result in varying OCR performances. Nonetheless, there are several sources that we can take reference from.

An article published in 2009 on the review of OCR accuracy in large-scale Australian newspaper digitization programs came up with these benchmarks (for **printed text**):

- **Good** OCR accuracy: CER 1-2% (i.e. 98–99% accurate)
- **Average** OCR accuracy: CER 2-10%
- **Poor** OCR accuracy: CER >10% (i.e. below 90% accurate)

For complex cases involving **handwritten** text with highly **heterogeneous** and **out-of-vocabulary** content (e.g., application forms), a CER value as high as around 20% can be considered satisfactory.



$$WER = \frac{S_w + D_w + I_w}{N_w}$$

Word Error Rate (WER) formula

The formula for WER is the same as that of CER, but WER operates at the **word** level instead. It represents the number of **word** substitutions, deletions, or insertions needed to transform one **sentence** into another.

WER is generally well-correlated with CER (provided error rates are not excessively high), although the absolute WER value is expected to be higher than the CER value.

For example:

- Ground Truth: 'my name is kenneth'
- OCR Output: 'my nime iz kenneth'

From the above, the **CER** is **16.67%**, whereas the **WER** is **75%**. The WER value of

75% is clearly understood since 3 out of 4 words in the sentence were wrongly transcribed.

## Python Example (with TesseractOCR and fastwer)

We have covered enough theory, so let's look at an actual Python code implementation.

[Click HERE to see the full demo Jupyter notebook](#)

In the demo notebook, I ran the open-source TesseractOCR model to extract output from several sample images of *handwritten* text. I then utilized the **fastwer** package to calculate CER and WER from the transcribed output and ground truth text (which I labeled manually).

	img_filename	ocr_output	ref_text	cer	wer
0	sample_image_1.png	Nhe 3 Chak of the, Corsenatalth	the 13 States of the Commonwealth	42.42	83.33
1	sample_image_2.png	'b YoS an occasion wort hy of Wig po	it was an occasion worthy of his presence.	42.86	75
2	sample_image_3.png	Let it g6, let it 90; Cand mold it Pack anymore	Let it go, let it go, can't hold it back anymore	20.83	54.55

Output from the sample Python implementation | Image by Author

## Summing it up

In this article, we covered the concepts and examples of CER and WER and details on how to apply them in practice.

While CER and WER are handy, they are **not** bulletproof performance indicators of OCR models. This is because the quality and condition of the original documents (e.g., handwriting legibility, image DPI, etc.) play an equally (if not more) important role than the OCR model itself.

I welcome you to join me on a data science learning journey! Give this [Medium](#)

page a follow to stay in the loop of more data science content, or reach out to me on [LinkedIn](#). Have fun evaluating your OCR model!

### Russian Car Plate Detection with OpenCV and TesseractOCR

Detecting, recognizing, and extracting car license plate numbers with the power of computer vision (A step by step...

[towardsdatascience.com](#)

### The Dying ReLU Problem, Clearly Explained

Keep your neural network alive by understanding the downsides of ReLU

[towardsdatascience.com](#)

Data Science

Computer Vision

Ocr

Python

Machine Learning



Follow

## Written by Kenneth Leung

8.9K Followers · Writer for Towards Data Science

Data Scientist at Boston Consulting Group (BCG) | Tech Writer | 1.5M+ reads on Medium | [linkedin.com/in/kennethleungty](#) | [github.com/kennethleungty](#)

More from Kenneth Leung and Towards Data Science





Kenneth Leung in Towards Data Science

## Running Llama 2 on CPU Inference Locally for Document Q&A

Clearly explained guide for running quantized open-source LLM applications on CPUs using Llama 2, C Transformers, GGML, and LangChain

★ · 11 min read · Jul 19



2.4K



31



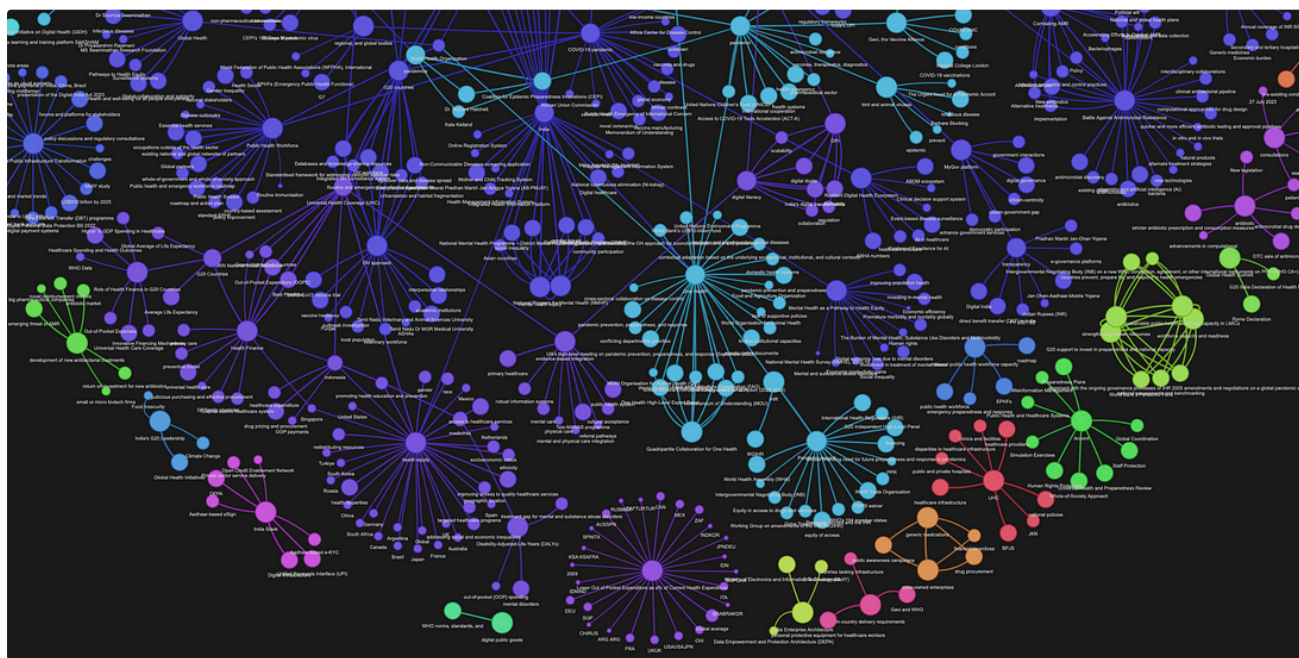
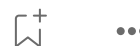
 Marco Peixeiro  in Towards Data Science

## TimeGPT: The First Foundation Model for Time Series Forecasting

Explore the first generative pre-trained forecasting model and apply it in a project with Python

🌟 · 12 min read · Oct 24

 2.5K  22



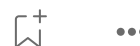
 Rahul Nayak  in Towards Data Science

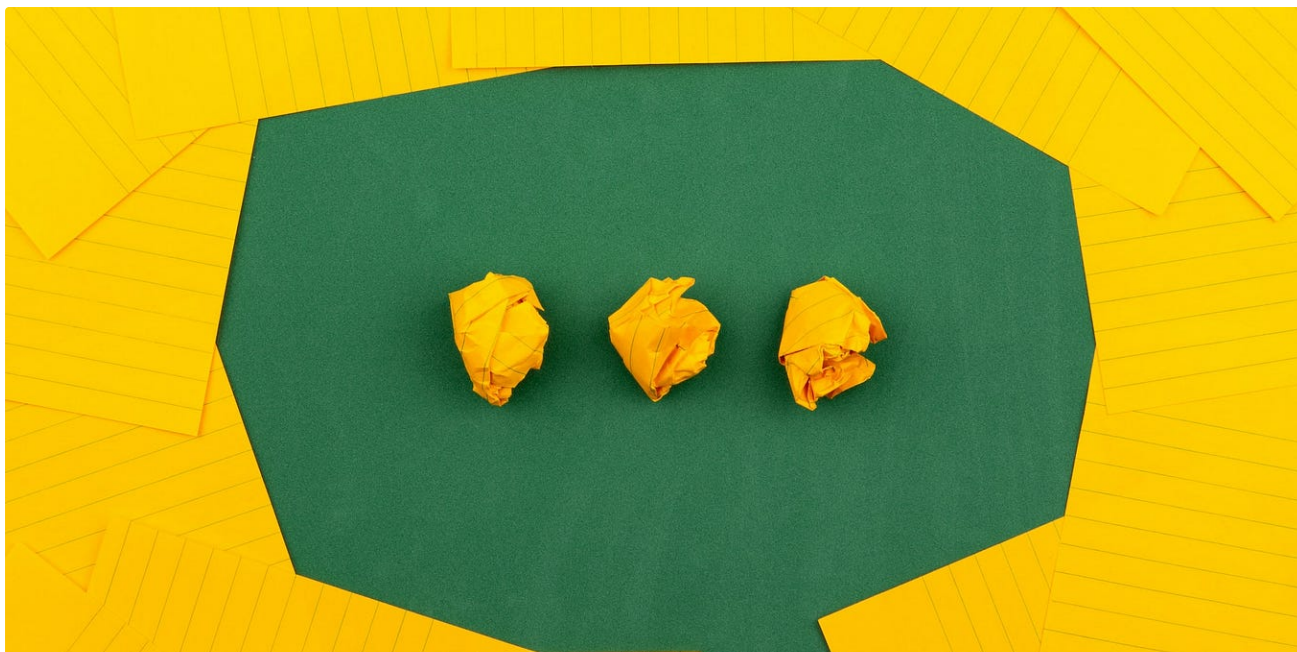
## How to Convert Any Text Into a Graph of Concepts

A method to convert any text corpus into a Knowledge Graph using Mistral 7B.

12 min read · Nov 10

 2.3K  32





Kenneth Leung in Towards Data Science

## Guide to ChatGPT's Advanced Settings — Top P, Frequency Penalties, Temperature, and More

Unlock the full potential of ChatGPT by optimizing extended configurations like Top P, frequency and presence penalties, stop sequences...

★ · 7 min read · Nov 7



195



1

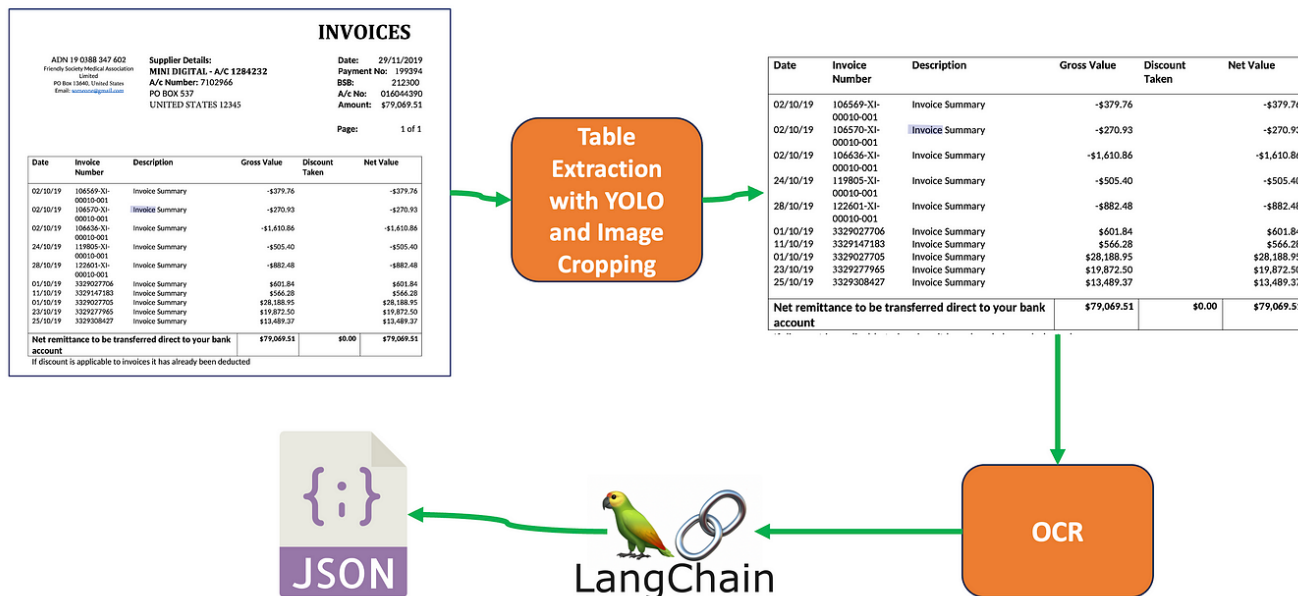


Recommended from Medium

See all from Kenneth Leung

See all from Towards Data Science





Ferry Djaja

## Table Extraction from Images and Information Retrieval using Deep Learning and a Large Language...

In this tutorial, I will guide you through the process of extracting tables and their line items using Deep Learning libraries, OCR, and...

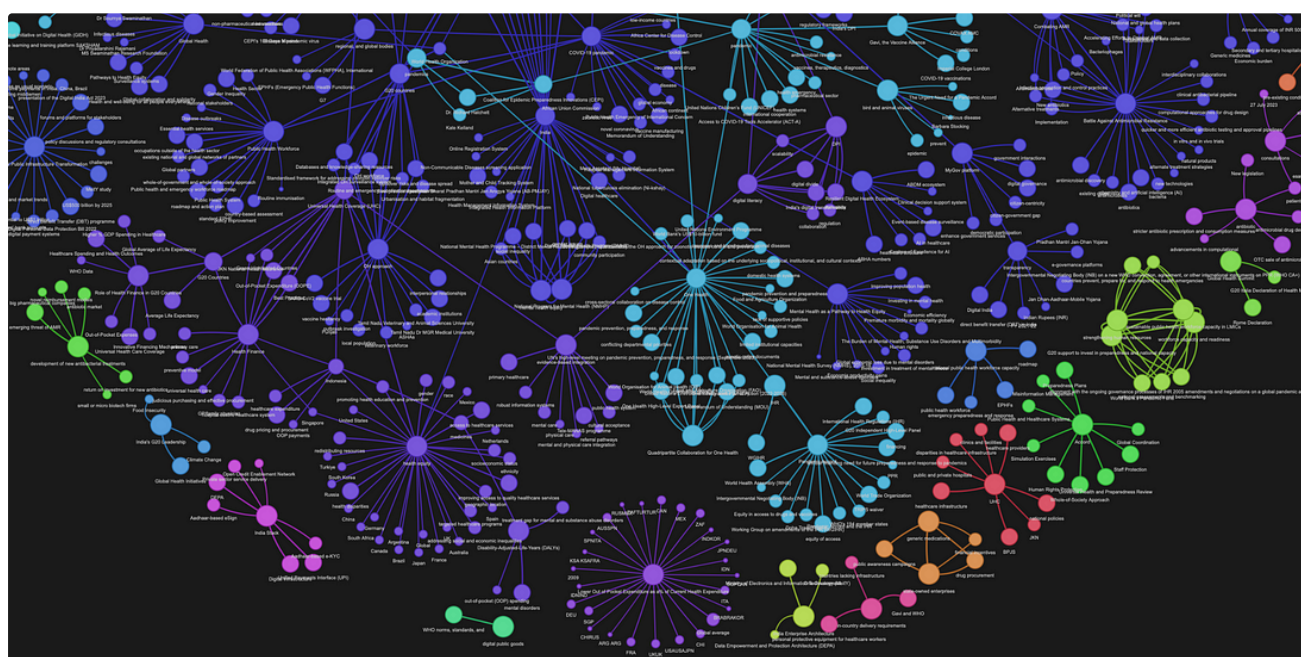
✦ · 6 min read · Oct 10

👍 60

💬 1

🔖

...



 Rahul Nayak in Towards Data Science

## How to Convert Any Text Into a Graph of Concepts

A method to convert any text corpus into a Knowledge Graph using Mistral 7B.

12 min read · Nov 10



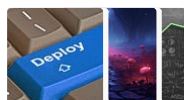
2.3K



32



### Lists



#### Predictive Modeling w/ Python

20 stories · 626 saves



#### Practical Guides to Machine Learning

10 stories · 710 saves



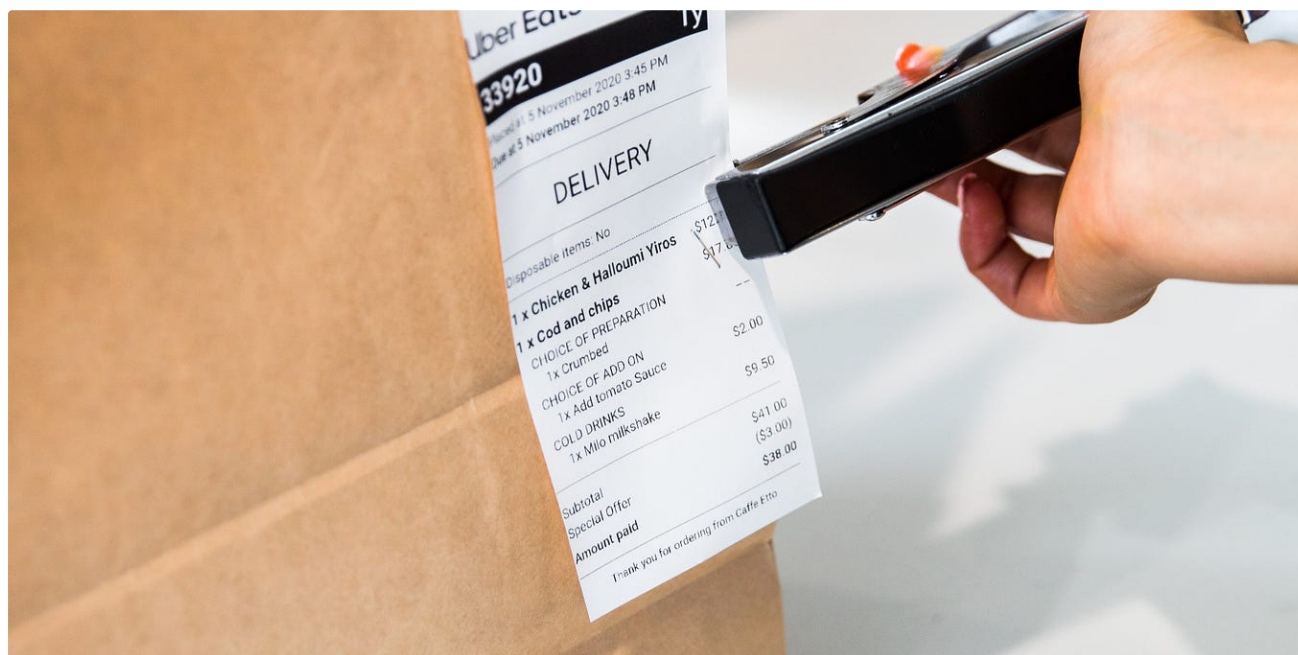
#### Coding & Development

11 stories · 282 saves



#### New\_Reading\_List

174 stories · 198 saves



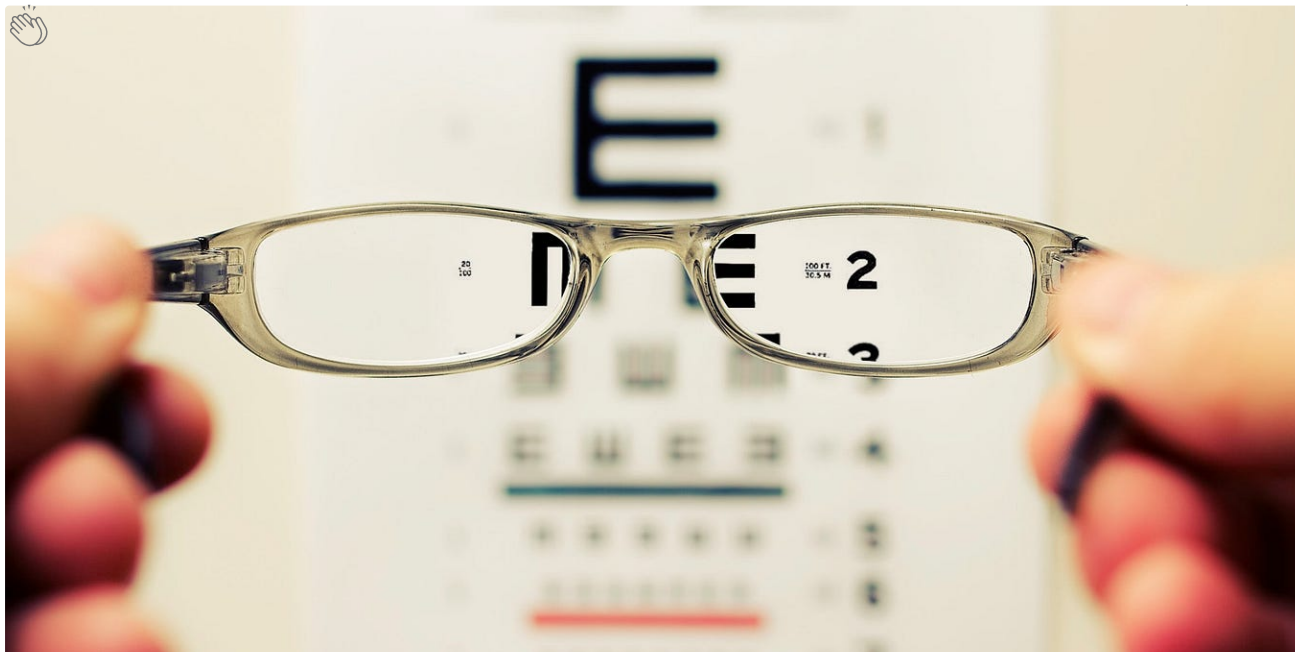


MinYang Chen

## Finetune LLM to convert a receipt image to json or xml

Motivation

10 min read · Oct 7



Vinod Baste

## Unlocking the Power of PaddleOCR

An Introduction to Text Detection and Recognition

8 min read · Sep 21



22







 Jacques-Yves GUILBERT--LY

## Information Extraction from ID Documents with Donut 🍩

We train a Donut model to extract information from the ID documents

6 min read · Sep 22



15



je



Text detection



Text



Drumil Shah in Searce

## Exploring Text and Table Extraction Packages in Python

Introduction

9 min read · Jul 7



17



See more recommendations