Optical Character Recognition

Introduction

Text localization and detection is the process of localizing where an image text is, you can think of it like object detection but for text.
In text detection our goal is to automatically compute the bounding boxes for every region of text in an image. Once we have those regions we can then apply OCR.
OCR is the process of digitizing a document image into its constituent characters. Also it is a complex problem because of the variety of language, fonts, and styles in which text can be written, and the complex rules of languages. OCR generally consists of several sub-processes to perform as accurately as possible. The subprocesses are:

- Preprocessing of the Image
- Text localization
- Character segmentation
- Character recognition
- Post-processing

This list of subprocesses can differ but these are roughly steps needed to approach automatic character recognition. It's main aim is to capture and identify all the unique words using different language from written characters.
OCR belongs to the family of machine recognition techniques performing automatic identification. Automatic identification is the process where the recognition system identifies objects automatically, collects data about them.
In this approach I am using *Tesseract* which is an open source text recognition OCR engine that has gained popularity. it's available under the Apache 2.0 license. It can be used directly or using an API to extract printed text from images, also it supports a wide variety of languages.

Baseline Experiments

In order to perform the task of OCR I followed the following pipeline:

- Prepare the Tesseract environment on pycharm
- Reading Images
- Preprocessing
- Detect the words
- Store results in flat file

In the phase of reading images: I started by reading a single image using cv2 library to load it from the disk using cv2.imread
In the phase of preprocessing: I read the image in grayscale format
In the phase of detecting the words: I started by extracting the bounding boxes coordinates of the text in the image to grab the OCR's text itself and then draw bounding boxes around the detected text. Finally I stored the data in CSV and TXT file formats.
Tesseract performs text detection and OCR is that I can do so in just a single function call, but I should be aware of using robust preprocessing because it performs quite poorly with noise and images not preprocessed well.

Other Experiments

Each experiment must have a goal, a set of steps, results and a conclusion
In order to enhance the preprocessing, I resized the greyscale images by half of its axes and applied adaptive threshold which helped a lot with the detecting accuracy as it segment an image by setting all pixels whose intensity values are above a threshold to a foreground values and all the remaining pixels.
I also tried applying Canny Filter, Opening but it didn't work well for detecting text in my provided dataset, as the preprocessing differs from problem to another, they might be powerful in another dataset.

Overall Conclusion

The benefit of using Tesseract to perform text detection and OCR is that we can do so in just a single function call, making it easier than the multistage OpenCV OCR process.
In this approach I have used Tesseract which uses LSTM to do its great work. Word finding was done by organizing text lines into blobs, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a change to more accurately recognize text lower down the page.
Also the preprocessing techniques applied before the Tesseract helps in enhancing the performance along with the transfer learning. As without the preprocessing the Tesseract will perform quite poorly if there is a significant amount of noise or your image is not properly preprocessed and cleaned before applying Tesseract.

Tools

- Tesseract 0.3.9
- Pycharm Edition 2021.3
- Python 3.10.1
- Anaconda Environnement

External resources

https://pysource.com/2020/04/23/text-recognition-ocr-with-tesseract-and-opencv/
https://towardsdatascience.com/how-to-extract-text-from-images-using-tesseract-ocr-engine-and-python-22934125fdd5
https://en.wikipedia.org/wiki/Optical_character_recognition
https://arxiv.org/abs/1710.05703
https://indatalabs.com/blog/ocr-automate-business-processes
https://github.com/madmaze/pytesseract
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.671.7331&rep=rep1&type=pdf
https://github.com/HusseinYoussef/Arabic-OCR
https://www.v7labs.com/blog/ocr-guide
https://moov.ai/en/blog/optical-character-recognition-ocr/
https://link.springer.com/chapter/10.1007/978-3-319-50252-6_2
https://link.springer.com/content/pdf/10.1007%2F978-3-319-50252-6.pdf

Questions

What was the biggest challenge you faced when carrying out this project?

The biggest challenge for me was how to make Tesseract work on my PC as I didn't install it before. Also there is not much effort put to solve the Tesseract using Arabic language online, which motivates me to do so.

What do you think you have learned from the project?

I learnt how to use Tesseract to detect text, localize it and then OCR it. I didn't know before that it supports multiple languages and can process even right-to-left text such as Arabic.