**OCR Correction for Indic Languages: Experimentation**

Sarbhanu Baidya | LinkedIn: [www.linkedin.com/in/sarbhanubaidya](https://www.linkedin.com/in/sarbhanubaidya)

**Introduction**

I used **Python** with **OpenCV** and **Tesseract** for this presentation.(Tesseract is an open-source OCR Engine completely funded by Google. **[link:** [Documentation](https://tesseract-ocr.github.io/) | [Main Repository](https://github.com/tesseract-ocr/tesseract) | [Installer](https://github.com/UB-Mannheim/tesseract/wiki) (Courtesy: University of Mannheim)**]**

The **advantage** of using **Tesseract** is that it supports more than **100+ languages**, including **major Indic Languages.** I’ve tested it with Sanskrit (for this presentation) and used the data from **Dr. Rohit Saluja’s repository on GitHub** [[link](https://github.com/rohitsaluja22/OpenOCRCorrect)] and got some **interesting results**.

**Previous Work in This Project**

I referred to Dr. Saluja’s work. Normally after getting the initial output from the OCR engine, the process of correction could be partially automated by training Artificial Neural Networks (Using LTSM and FastText) as described in-depth in this paper. [[link](https://www.cse.iitb.ac.in/~rohitsaluja/PID6011473.pdf)]

**Workflow for My Project**

What I did mainly here is to pre-process the data at first. It is extremely hard to get a 100% confident result after doing an OCR. That's why oftentimes it needs to be verified manually after the character recognition engine converts the binary image data into a string.

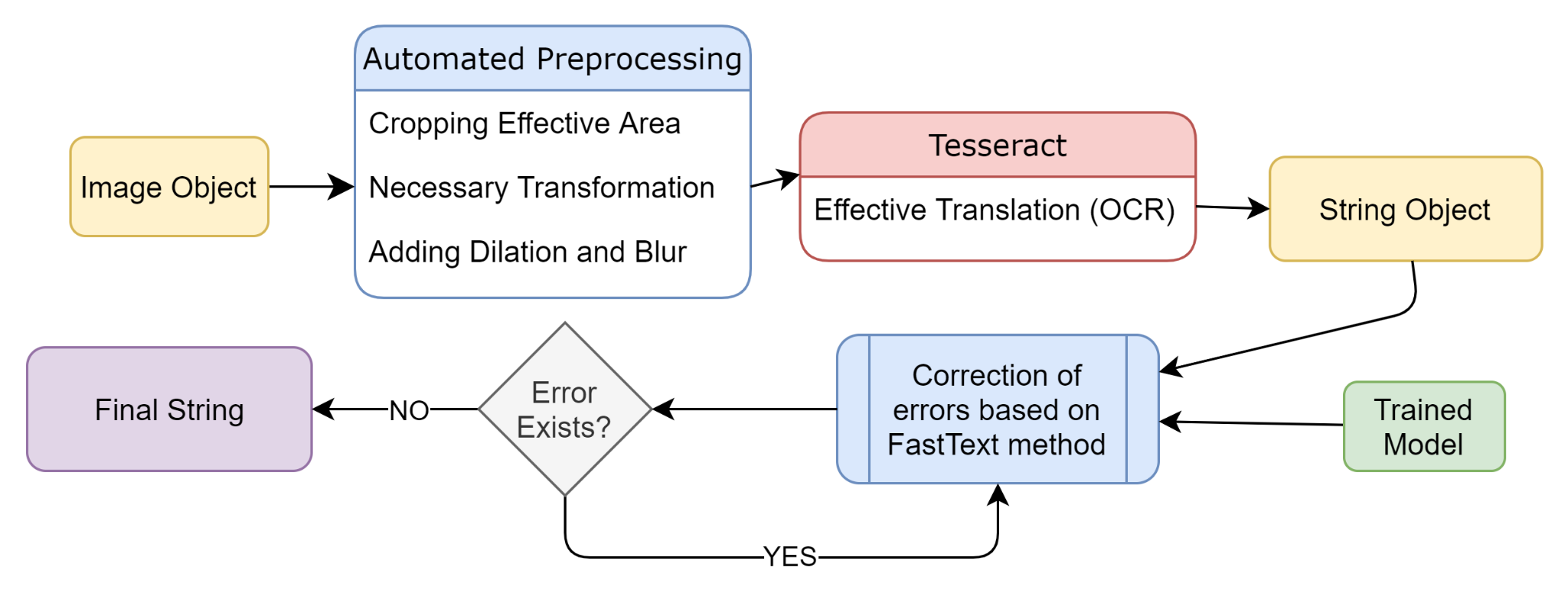
When I tried to OCR the actual image file, I got almost 75% correct output. After that, I did somepre-processing to the data using OpenCV and ran the data via Tesseract and got **almost 95% corrected text** after a pass. The downside is altering an image is custom for different types of pages and different sized fonts, although **the process could be similar for similar types of data**. For e**xample:** A Book has almost identical pages in terms of margin and fonts, so the pre-processing parameters can be settled once.

**Configuration:** Installed Tesseract. Configured my Anaconda Environment and installed libraries (Used: **OpenCV, PyTesseract, NumPy** and **Pillow**). Used PyCharm as my main IDE.

Project Link: <https://github.com/sarbhanub/opencv-ocr-indic>

* **Defining Functions:** Defined Functions to process the image at first for the work. Wrote Functions for **Cropping, Dilating, Eroding** and the last one for **Removing the noise** and added some **dilation**. The functions are defined in this file. [[link for code](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/func.py)] [[image output](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/processed_data/without_noise.jpeg)]
* **Pre-processing the Data:** Adjusted some **parameters** for a particular page using **OpenCV**. (The parameters will be mostly the same for similarly scanned images)[used [sanskrit/files/page-9.jpeg](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/data/sanskrit/files/page-9.jpeg)file for the test.] [[link for code](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/main.py)]
* **Adding Blur:** One thing that is worth mentioning is that adding some **Gaussian blur** (tried median blur too) completely changed the output. Got a **much better output** from a little blurred Image, compared to the sharp one. [[text output](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/output/output-page-9.txt)]
* **Another Example:** I tried doing the same for the **index page** I found in the same directory [[sanskrit/files/page-8.jpeg](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/data/sanskrit/files/page-8.jpeg)] and got mediocre results. I cropped it into columns to obtain a better result and got some satisfactory output. I left the output in two files. [[link for code](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/index_boxes.py)] [[text outputs](https://github.com/sarbhanub/opencv-ocr-indic/tree/master/output/output-page-8)]
* **Bounding Box:** I also tried to define bounding boxes for that page and got this result. [[image output](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/temp/bbox_image.jpeg)]

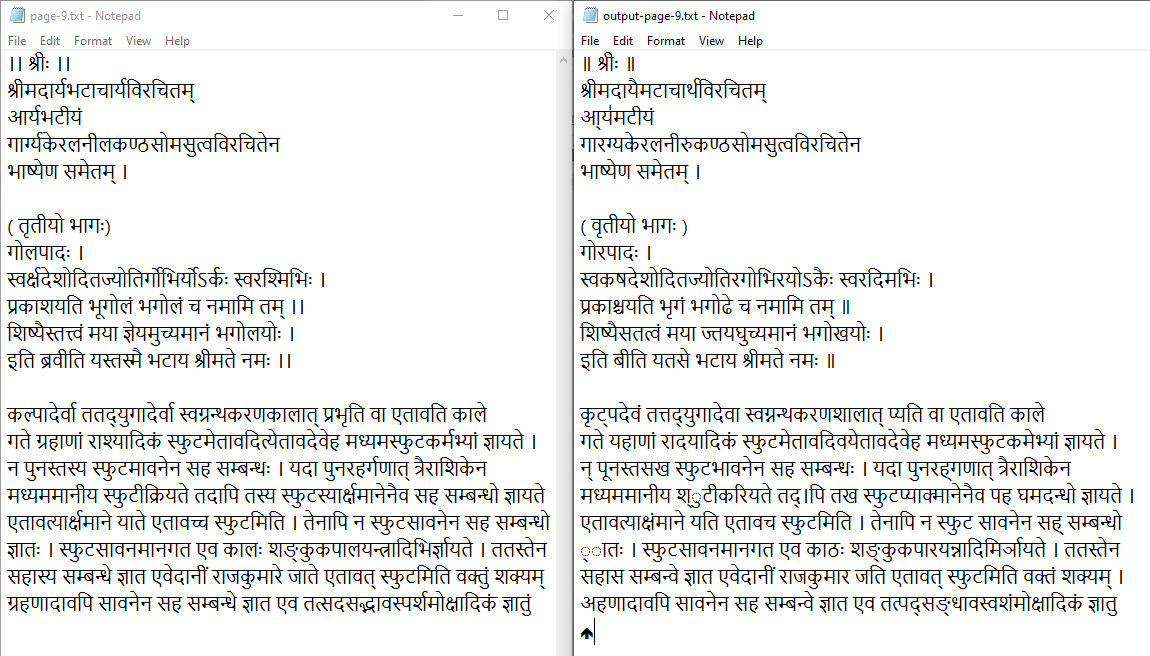
**Diagram**

****

**Results**

Left side. The Correct One.

The right one is my output after pre-processing the image and then doing OCR using that image. (There were some spaces in the output file. I eliminated them manually for comparing them side by side.) You can see the raw text file here. [[text output](https://github.com/sarbhanub/opencv-ocr-indic/blob/master/output/output-page-9.txt)] The output is almost correct with some errors.

****

**Output:** Actual Text [Left] | OCR after pre-processing the Image [Right]

**Conclusion**

The OCR Output depends upon a large scale of factors, one of them is the **Quality of Scan** and unfortunately it is not always in our hands. One thing I also noticed along with the Fusional nature of the words is that the letters themselves are kind of **fragmented** in these texts, sometimes due to the fact that the printing was done in this fashion in the early days. That is largely affecting the results, that's why adding some Gaussian Blur added some precision as those voids are gone after that. I tried Diluting them too, but adding blur yields the best result for my case.

Again, I found another issue with mixed languages. I am working on that issue.

Although OCR can’t be 100% correct, neither any method can provide complete accuracy but if we get better results after doing OCR first, then train the ANN followed by that, it would make a better prediction in the output layer (as the number of **correct words in the input layer** will be **much greater** compared to earlier) and That will ultimately improve the training data by a large factor followed by improving the correction model itself.