https://doi.org/10.69974/glskalp.01.02.54

GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Study of Tesseract OCR

Kartik Joshi
CEO, Techsamvaad Pvt. Ltd.

Abstract

In the current Internet and Digitization era, a huge amount of information is available in different forms like books, newspapers, etc. To preserve the contents of such documents, these documents are converted to a digital format by scanning them as images. Detection of text from the scanned images and correct identification of characters is a challenging problem in such cases. Tesseract is a recognition engine based upon open source license which uses some novel techniques for optical character recognition. Tesseract has been designed to recognize more than 100 languages. Few of these languages are English, Italian, French, German, Spanish, Dutch and many more. It also works for a few Indian languages such as Bengali, Gujarati, Hindi, Kannada, Malayalam, Oriya and others. OCR is the branch of image recognition that is used in applications to recognize text from scanned documents or images. Today combined with the field of Artificial Intelligence this technology is becoming a boon to capture and comprehend the data automatically. In this paper, the researcher has done a detailed study of the working of the Tesseract OCR.

Key Words: *Artificial Intelligence, Optical Character Recognition, Tesseract*

Architecture of Tesseract

As HP never made it page layout analysis technology an open source entity, Tesseract does not have it. Instead it assumes that the Input Image is binary with optional polygonal text regions predefined. The first step of the process is to store the outlines of the component by a connected component analysis. Although computationally expensive, since first introduced it holds an important advantage to detect inverse text & is able to recognize it as black-on-white text by performing inspection of nesting of outlines & child and grandchild outlines. These outlines are then nested into Blobs and then further organized into text lines by analyzing them and regions for fixed pitch or proportional text.

Next, the text lines are broken into words depending upon the character spacing. Fixed pitch text gets chopped into character cells whereas proportional text is broken into words using definite spaces and fuzzy spaces. The recognition process works in two passes. Firstly, every recognized

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

word is passed to an adaptive classifier as a training data and then eventually the classifier gets a chance to more accurately recognize the text in lower half of the page. Due to the training in the first pass, a second pass is made to recognize the words that may not have been recognized earlier. The last step is to remove all fuzzy spaces and check options for x-height to locate small cap text.

Working of Tesseract

The workflow of Tesseract is shown in the figure 1 below:

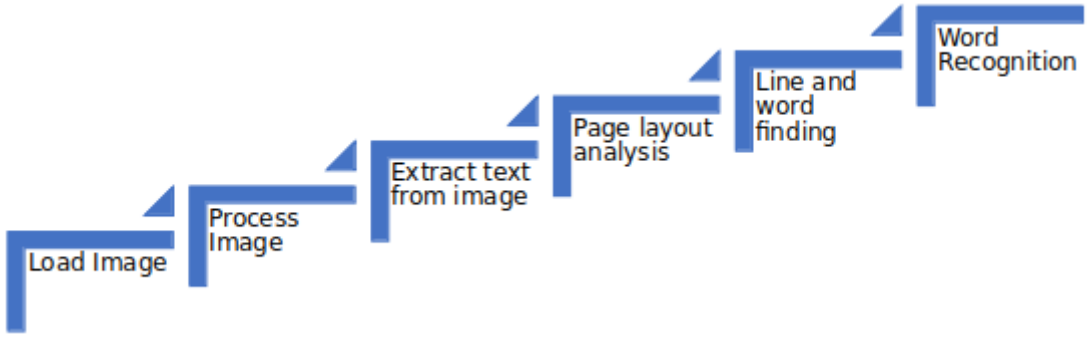
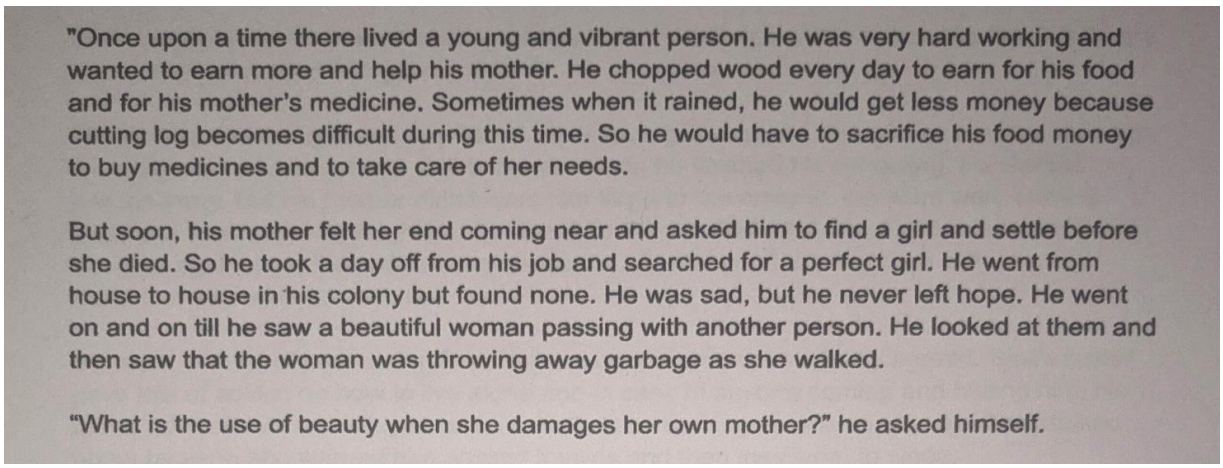


Figure 1: Workflow of Tesseract

The section below gives a detailed description of each of the processes shown in figure 1.

Load Image

To begin with the process the user must provide an input as an image. The quality of the output depends on the quality of the image captured. The image captured has to be of 70 DPI, if not then the engine identifies and converts the input image to a nearby DPI value. For experimentation purposes the image shown in figure 2 has been taken as input.


"Once upon a time there lived a young and vibrant person. He was very hard working and wanted to earn more and help his mother. He chopped wood every day to earn for his food and for his mother's medicine. Sometimes when it rained, he would get less money because cutting log becomes difficult during this time. So he would have to sacrifice his food money to buy medicines and to take care of her needs.

But soon, his mother felt her end coming near and asked him to find a girl and settle before she died. So he took a day off from his job and searched for a perfect girl. He went from house to house in his colony but found none. He was sad, but he never left hope. He went on and on till he saw a beautiful woman passing with another person. He looked at them and then saw that the woman was throwing away garbage as she walked.

"What is the use of beauty when she damages her own mother?" he asked himself.

Figure 2: Input Image

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP
Journal of Multidisciplinary Studies
Volume 1; Issue 2;
April - June 2021

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Process Image

Next, the Tesseract OCR will process the image provided as an input by the user. The Tesseract OCR will check whether the image is clear or not. If the image is blurry or not proper then there might not be a proper output. So the user has to be careful about providing the input image. (R.W. Smith, 1987)

Crop Image

After the processing of the image is done it will move on to the next part i.e. cropping the image. In this part the user will be able to select the part from the according to his need. The user is free to crop the required size of the image. The region selected by the user will be the region from where the text will be extracted.

Extract Text From Image

After cropping the image the text will be extracted from the region selected by the user. Each character in a particular language is stored in the database. The characters from the image will be processed and compared with the characters in the database. If a match is found then the character will be displayed on the output. The same process will be done for all the characters present in that region. In case if the character is not matched then that character will not be considered and the pointer will move to the next character. The Tesseract OCR tries to identify the maximum number of characters from the given input image.

Page Layout Analysis

The first step of Recognition is page layout analysis which divides the input image into two halves of text and non-text. Furthermore, it can even combine a multi-column text into a single column of text.

The page layout analysis in Tesseract is based on the idea of detecting tab-stops in a document image. It has the following steps

Step 1: The Leptonical morphological processing detects the vertical lines and images. Due to this, the input image is now cleansed and passed to the connected component analysis.

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Step 2: The tab-stop connected component candidates which appear to be at the edge of a text region are identified, combined as tab-stop lines.

Step 3: The scanning process goes from left to right and top to bottom. Similarly classified connected components are gathered into Column Partitions (CP) and checked whether no CP may cross a tab-stop line. (P.J. Rousseeuw, A.M. Leroy, 2003.)

Step 4: Chin-like blocks of text CPs are again divided into groups of uniform line-spacing text blocks. Hence now each chain of CPs represents a candidate region. The reading order of the regions are determined by some heuristic rules.

Line and Word Finding

The line and word finding mechanism of Tesseract is a multi step method. The detailed explanation of line finding and word finding is given herewith.

Line Finding

Line finding algorithm is used to recognize a skewed page without de-skewing it and thereby maintaining the image quality. If we assume that the page layout analysis has already been given text regions of uniform text sizes then the percentile height filter removes the drop caps and vertically touching characters. Whereas it is easier to filter out Blobs such as punctuation, diacritical marks and noise due to median height removing the Blobs smaller than the average median height. By sorting the Blobs with the x-coordinate allows to assign Blobs to a unique text line while tracking slope across skew, thus reducing the danger of assigning an incorrect text line in the presence of skew. The next step is to assign filtered Blobs into appropriate lines using median of squares fit to estimate the baselines. Lastly, the horizontal overlapping Blobs are merged by making the diacritical parts of some broken characters. (B.A. Blesser, T.T. Kuklinski, R.J. Shillman, 1976; Bhatt, 2014)

Fixed Pitch Detection and Chopping

The OCR engine tests the text lines to determine whether they are fixed pitch or not. In case if they are then words are chopped into characters and passed to associator for word recognition. (M. Bokser, Jul 1992), (G. Nagy, Jul 1992), (G. Nagy, Y. Xu, Aug 1997), (I. Marosi, Jan 2007) The figure 3 shows a typical example of a fixed-pitch word.

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Figure 3: A fixed-pitch chopped word

Proportional Word Finding

The possibility of fixed pitch chopped proportional word able to find the non-fixed pitch or proportional text spacing is unusual. The figure 4 illustrates some typical problems. The problems identified are as the gap between ‘what’ and ‘is’ is larger than the space between boxes of ‘?’ and ‘he’. These problems are taken care by the Tesseract by making spaces near threshold of baseline and mean-line fuzzy which can be decided after the word recognition.

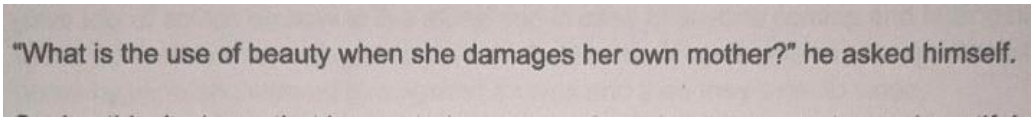


Figure 4: Some Difficult word spacing

Word Recognition

The segmentation of a word into characters is the basic recognition process for any word recognizer. The first step of this process is to classify the segmented output from finding. Then on wards it is applied to a non-fixed pitch text.

Chopping joined characters

In case the result of the word is not good, chopping joined words, Tesseract tries to improve it by chopping the Blob with the worst confidence it has from the character classifier. By conducting the outline polygonal approximation, the chop points candidates are found from its concave vertices or its opposite or a line segment. It can take up to 3 pairs of chop points to separate joined characters from the ASCII set.

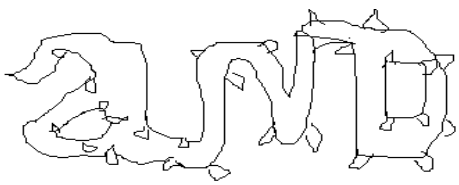


Figure 5: Candidate chop points and chop

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP
Journal of Multidisciplinary Studies
Volume 1; Issue 2 ;
April - June 2021

Figure 5 shows a set of candidate chop points with arrows and the line across the outline. As the chopping is done in priority order, even low confidence chops are kept and not discarded so that the associator can re-use them later.

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Associating Broken Characters

The associator begins searching (A*) the segmentation graph of the possible combinations of the highest chopped Blobs into character candidates without actually building the segmentation graph, but rather by maintaining a hash table of visited states.

The search further proceeds by pulling new state candidates from priority queue and evaluating them by classifying the unclassified fragment combinations. Although the full-chop-then associate is arguable inefficient, it has an advantage of simplifying the data structures used to maintain the segmentation graph.

Figure 6: An easily recognized word

Since the implementation of A* segmentation search in 1989, the accuracy of Tesseract to recognize broken characters is ahead then the other competing commercial engines. Figure 6 is a typical example.

Character Classifier

Character classifier is the process to determine characters from the text extracted from the input image. The following section shows the working of Character classifier.

Features

The features used in classification are the shape outlines component of polygonal approximation. During the training, every polygonal approximation element derives a 4-dimensional feature vector (x, y-positional, direction, height) and forms a prototypical feature vectors, giving the name Tesseract. To eliminate the length dimension, polygon elements are divided into shorter pieces of equal length which allows the recognition of damaged images. However the problem of the high computational cost required to calculate the distance between an unknown and prototype still remains. (R.W. Smith, 2009)

To identify the unknown and reduce the computing time in the first step of classification, a class pruner creates a shortlist of 1-10 characters classes using the method almost similar to Locality

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Sensitive Hashing (LSH). Next, the classifier calculates the weighted distance d^f of each feature from its nearest prototype as follows:

$$d^f = d^2 + w\emptyset^2 \quad \text{---------- (1)}$$

Here d represents the Euclidean distance of the feature coordinates from the line of prototype, whereas θ represents the difference of the angle from the prototype. Using the below equation, the feature distance gets converted in to feature evidence E^f.

$$E^f = \frac{1}{1 + kd^2 f} \quad \text{---------- (2)}$$

The constant k controls the rate of evidence decay with distance and the feature evidence E^f , is copied to the prototypes E^P . The sums are normalized by using the sums of prototype lengths L^P and the number of features, delivering a converted result of distance:

$$d\ final = 1 - \frac{\Sigma fEf + \Sigma pEp}{Nf + \Sigma pLp} \quad \text{---------- (3)}$$

The computed final distance is used by KNN algorithm for classification.

Adaptive Classification

To obtain greater discrimination within each document having limited number of fonts, a more sensitive adaptive classifier trained by the output of the static classifier is used.

As the adaptive classifier is learning in the first run, the contribution done by it on top of the page is very less hence a second run is performed over the page to recognize the words that if failed to recognize in the first run.

Sample Output

The working of tesseract was tested using the image in figure 1. As per the outcome, the frequency is as 153 words out of 173 words in the input image were identified. Table 1 shows the listing of

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

frequency of actual words along with the frequency of words identified by Tesseract omitting the results of the words identified 100%. It also shows the percentage of error encountered.

Words	Word Counted	Words Identified	Error %
and	10	8	20%
because	1	0	100%
before	1	0	100%
earn	2	1	50%
food	2	0	100%
or	3	2	33%
from	2	1	50%
hard	1	0	100%
his	7	6	15%
money	2	0	100%
mother’s	1	0	100%
to	7	5	29%
went	2	1	50%
working	1	0	100%
Average Error count			67.65%

Table 1: Output of Tesseract

As it can be observed from Table 1 the average error is 11.30% on the basis of the frequency count as 153 words out of 173 words were identified. But if we calculate the average error rate for the identified words, it comes to 67.65%. This shows that there is still a lot of scope left to improve the working of the Tesseract OCR Engine for English language.

Conclusion

https://doi.org/10.69974/glskalp.01.02.54
GLS KALP Journal of Multidisciplinary Studies Volume 1; Issue 2; April - June 2021

Tesseract OCR is now included in every leading commercial engines due to its accuracy. Having the advantage of its unusual choice of features, use of polygonal approximation as input to the classifier is still its weakness.

The addition of a Hidden-Markov-model based character n-gram model and maybe an improved character chopper along with internationalization, can improve the accuracy of Tesseract even more.

References

Bhatt, A. (2014). Information needs, perceptions and quests of law faculty in the digital era. The Electronic Library, 32(5), 659–669. https://doi.org/10.1108/el-11-2012-0152

Blesser, B. A., Kuklinski, T. T., & Shillman, R. J. (1976). Empirical tests for feature selection based on a psychological theory of character recognition. Pattern Recognition, 8(2), 77-85.

Bokser, M. (1992). Omnidocument technologies. Proceedings of the IEEE, 80(7), 1066-1078.

Leptonica image processing and analysis library. http://www.leptonica.com.

Macwan, S. J., & Vyas, A. N. (2015, August). Classification of offline Gujarati handwritten characters. In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1535-1541). IEEE.

Marosi, I. (2007, January). Industrial OCR approaches: architecture, algorithms, and adaptation techniques. In Document Recognition and Retrieval XIV (Vol. 6500, p. 650002). International Society for Optics and Photonics.

Nagy, G. (1992). At the frontiers of OCR. Proceedings of the IEEE, 80(7), 1093-1100.

Nagy, G., & Xu, Y. (1997, August). Automatic prototype extraction for adaptive OCR. In Proceedings of the Fourth International Conference on Document Analysis and Recognition (Vol. 1, pp. 278-282). IEEE.

Rousseeuw, P. J., & Leroy, A. M. (2005). Robust regression and outlier detection (Vol. 589). John wiley & sons.

Smith, R. W. (1987). The extraction and recognition of text from multimedia document images (Doctoral dissertation, University of Bristol).

Smith, R. W. (2009, July). Hybrid page layout analysis via tab-stop detection. In 2009 10th International Conference on Document Analysis and Recognition (pp. 241-245). IEEE.