### Tesseract OCR

Documentation on tesseract about

* how it works

1. Outlines are analysed and stored

2. Outlines are gathered together as Blobs

3. Blobs are organized into text lines

4. Text lines are broken into words

5. First pass of recognition process attempts to recognize each word in turn

6. Satisfactory words passed to adaptive trainer

7. Lessons learned by adaptive trainer employed in a second pass, which attempts recognize the words that were not recognized satisfactorily in the first pass

8. Fuzzy spaces resolved and text checked for small caps

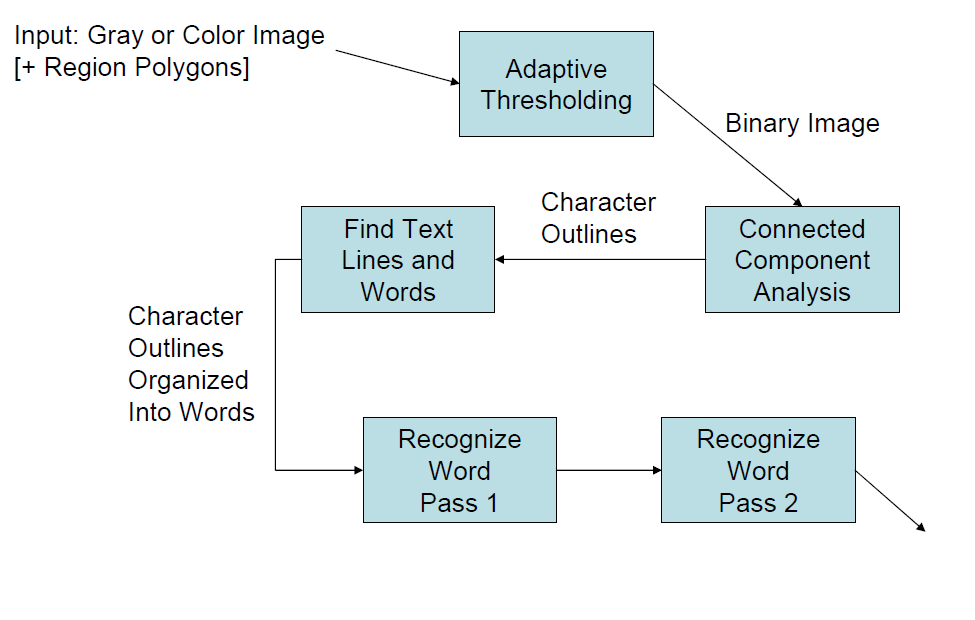
9. Digital texts are outputted

* Tesseract uses:
* Algorithms for detecting text lines from a skewed page
* Algorithms for detecting proportional and non-propositional words (a proportional word is a word where all the letters are the same width)
* Algorithms for chopping joined characters and for associating broken characters
* Linguistic analysis to identify the most likely word formed by a cluster of characters
* Two character classifiers: a static classifier, and an adaptive classifier which employs training data, and which is better at distinguishing between upper and lower case letters
* Accuracy and Limitations refer to

<http://www.saltcymru.org/english/saltcymru_document5.pdf>

## 

## Tesseract Architecture



**ADAPTIVE THRESHOLDING:**

* Thresholding is used to segment an image by setting all pixels whose intensity values are above a threshold to a foreground value and all the remaining pixels to a background value.
* The Thresholding value for each pixel is calculated by finding the mean of the local intensities of neighbouring regions.

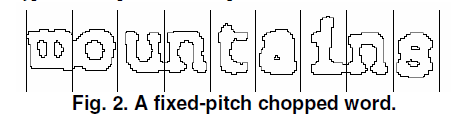
**Processing Steps:**

Give input as binary image(by Adaptive Thresholding)

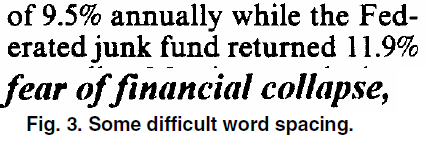
* First step is ----  **Connected component analysis.**
  + In which outlines of the components are stored and gathered together, purely by nesting, into Blobs.
* Second step is ---- **Find Text Lines and Words**
  + Done by  **Line finding algorithm** 
    - The line finding algorithm is designed so that a skewed page can be recognized without having to de-skew.
    - key parts of the process are blob filtering and line construction.
  + Followed by **Baseline Fitting**
    - enables Tesseract to handle pages with curved baselines.
      * Eg:



* + Followed by **Fixed Pitch Detection and Chopping**
    - Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds fixed pitch text, Tesseract chops the words into characters using the pitch



* + Followed by **Proportional Word Finding**
    - When spaces between words are tricky , this method is followed
    - Explanation is given in below example

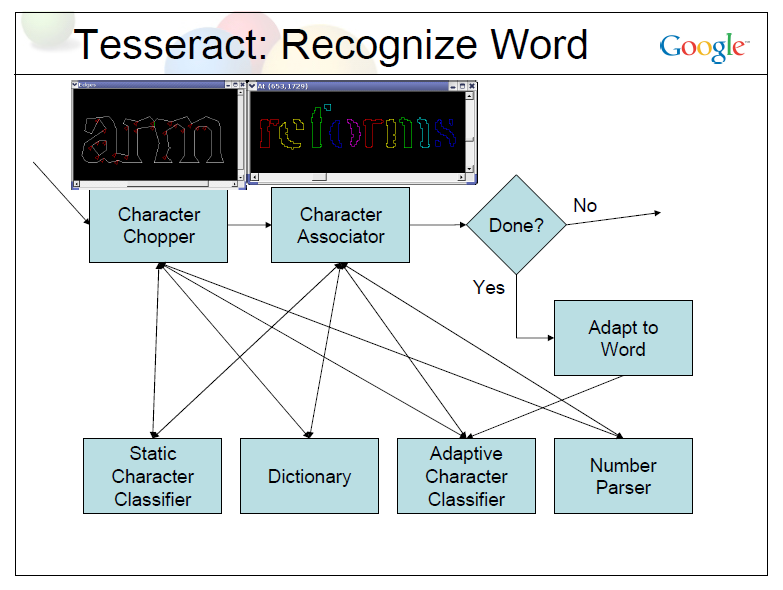


As you can see that gap between the tens and units of ‘11.9%’ is a similar size to the general space, and is certainly larger than the space between ‘erated’ and ‘junk’. There is no horizontal gap at all between the bounding boxes of ‘of’ and ‘financial’. Tesseract solves most of these problems by measuring gaps in a limited vertical range between the baseline and mean

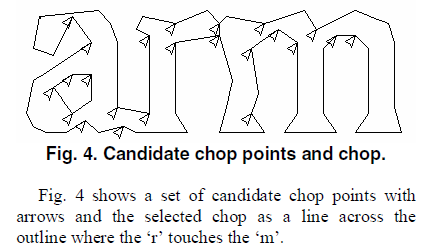
line. Spaces that are close to the threshold at this stage are made fuzzy, so that a final decision can be made after word recognition

* Third step is ----**Word Recognition**
  + Proceeds as 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 chance to more accurately recognize text lower down the page.
    - Second pass is run over the page,in which words that were not recognized well enough are recognized again.

**WORD RECOGNITION**



* **Chopping Joined Characters(Character chopper)**
  + when the results from a word is unsatisfactory ,Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier.
  + Chop points are found from concave vertex of the polygonal approximation of the outlines , and may have either another concave vertex opposite, or a line segment.



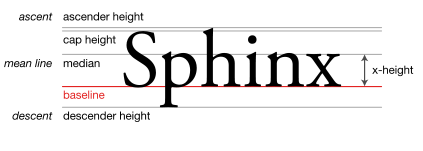
* **Associating Broken Characters(Character associator)**
  + When potential chop is exhausted ,even the word is not good enough it is given to associator.
  + Associator makes an A\* (best first)search of the segmentation graph of possible combinations of the maximally chopped blobs into candidate characters.(i.e,The A\* search proceeds by pulling candidate new states from a priority queue and evaluating them by classifying unclassified combinations of fragments)
* **Static character Classifier**
  + Static classifier uses outline fragments as features. Broken characters are easily recognizable by a small->large matching process in classifier. (This is slow.)
  + The static classifier normalizes characters by the centroid (first moments) for position and second moments for anisotropic size normalization.
  + The main benefit of character moment normalization is removal of font aspect ratio and some degree of font stroke width. It also makes recognition of sub and superscripts simpler, but requires an additional classifier feature to distinguish some upper and lower case characters.
* **Adaptive Classifier**
  + Same technique as Static character classifier but only significant difference between the static classifier and the adaptive classifier, apart from the training data, is that the adaptive classifier uses isotropic baseline/x-height normalization, whereas the static classifier normalizes characters by the centroid (first moments) for position and second moments for anisotropic size normalization.
  + The baseline/x-height normalization makes it easier to distinguish upper and lower case characters as well as improving immunity to noise specks.

# **Improving the quality of the output:**

* Improvement of output depends on various factors .
* DPI
* Image processing
  + Binarisation
  + Noise
  + Orientation /skew
  + Borders
* Segmentation methods
* Dictionary,Word lists and Pattern.
* Refer to the link for more information
* <https://code.google.com/p/tesseract-ocr/wiki/ImproveQuality#Image_processing>
* <http://vbridge.co.uk/2012/11/05/how-we-tuned-tesseract-to-perform-as-well-as-a-commercial-ocr-package/>

# **Is there a Minimum Text Size? (FAQ in tesseract)**

# There is a minimum text size for reasonable accuracy. You have to consider resolution as well as point size. Accuracy drops off below 10pt x 300dpi, rapidly below 8pt x 300dpi. A quick check is to count the pixels of the x-height of your characters. (X-height is the height of the lower case x.) At 10pt x 300dpi x-heights are typically about 20 pixels, although this can vary dramatically from font to font. Below an x-height of 10 pixels, you have very little chance of accurate results, and below about 8 pixels, most of the text will be "noise removed".



# **APIs to get the confidence value of OCR result**

# Refer to the below file to get to know all the APIs supported by the core tesseract library basically in C++.

<https://code.google.com/p/tesseract-ocr/source/browse/api/baseapi.cpp>

For the specific case of confidence the relevant APIs are MeanTextConf and AllWordConfidences.

It also has the APIs for getting the thresholded images, text lines and regions. Any thing of interest for debugging in Tesseract can be obtained from this file.

The same can be obtained for Android at :

<https://github.com/rmtheis/android-ocr>

<https://github.com/rmtheis/tess-two>

Also there is training data that is available for orientation and script detection that can downloaded at: [https://code.google.com/p/tesseract-ocr/downloads/detail?name=tesseract-ocr-3.01.osd.tar.gz&can=2&q=tesseract-ocr-3.01.osd](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Frmtheis%2Ftess-two&sa=D&sntz=1&usg=AFQjCNGE1IzI1cOI0VUEZtAH7hjs3XYG2Q).

**References:**

* For Terminology in above documentation refer to <http://tesseract-ocr.repairfaq.org/tess_glossary.html>
* To Download Tesseract <http://worldofprasanna.in/blog/>
* For accuracy improvement refer to ..<http://stackoverflow.com/questions/9480013/image-processing-to-improve-tesseract-ocr-accuracy>
* For adaptive thresholding refer to <http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm>

***Documentation references*:**

1. <http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf>
2. <http://code.google.com/p/tesseract-ocr/downloads/detail?name=TesseractOSCON.pdf>