*Image Processing Pipline*

**Image processing Pipline:** We have limited our approach to segmenting the resumes into easily documented formats for database management. Typically, we would go through the above mentioned sequence of steps, as a 'pipeline' to finally arrive at a solution to the problem.

**Resume Acquisition:** The resume parser accepts the resumes in various formats such as structured documents (i.e. pdf, docx, …etc) or Images (i.e. png, jpg, slides…etc). Regardless of the entried format, it is obvious that our model performs some *preprocessing* actions as well as *conversion* of the inputed file to a regular image format to be ocred. Also, it accepts the resumes as a document consisting of *multiple pages*. In this situation, please consider the previuously pipline as an iterative process for Resume Information Extraction.

**Frames Segmentation:** At the first level of the pipline, we may consider cropping frames for informmation extraction. The cropped frames makes use of their inter-dependencies in order to identifiying most of the diffrence between segments. This way, we could guarentee that the extracted frames does not share any relative embedding after all. In the case of a multi-pages resume, It migh look more difficulte to achieve this purpose, as multiple frames could normally interconnect their content.

**Frames Segmentation**
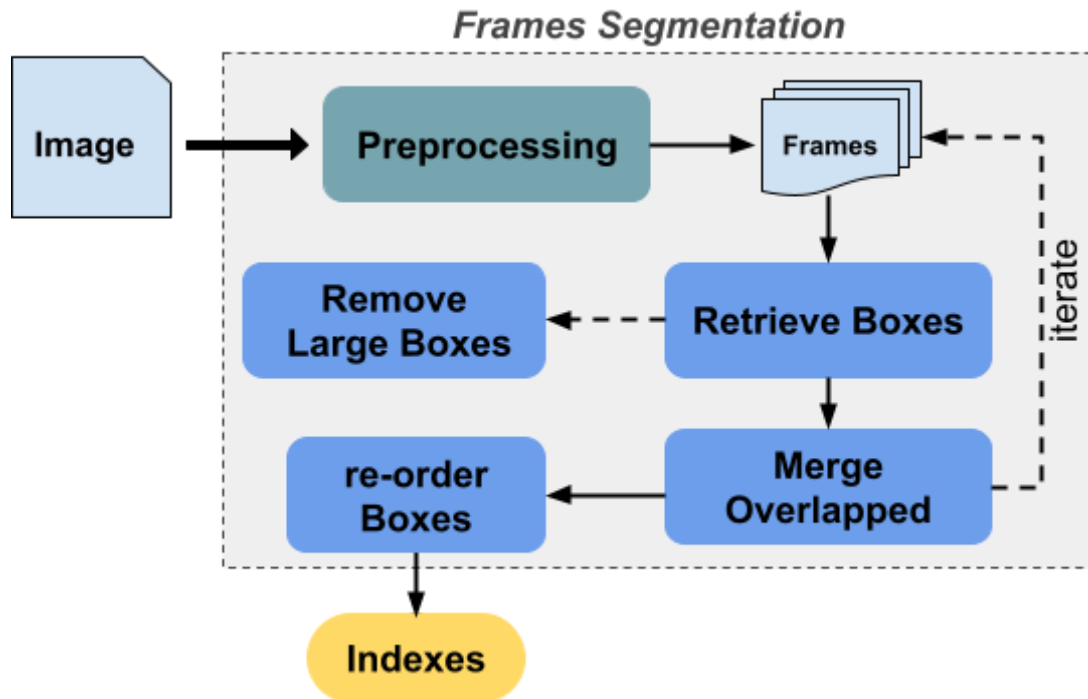
**Lines & Blocs Identification:** Extracting resume information relies at the first place on the identification of its content lines. As the content appears to have logos, tradmarks, …etc, this step would consists only on retrieving both lines and blocs. It should be noted the inheritence derived from the aspect of blocs, as a single bloc may cluster multiple lines, thus the sub-named clustering emerged.

**Meta-data Configuration & Recognition:** At this level, we may refer to an open-source solution (i.e Tesseract) for data recognition. As Tesseract may have various configurations to be used, we typically need to pre-define the cropped lines (resp. blocs) in the config arguments. This helps Tesseract to better parse the segments and retrieve a desired outcomes in an iterative way.

## Lines / Blocs Identification

```
Frame  →  Preprocessing  →  Set Kernel
                                  ↕ config
         Merge              Retrieve Lines
         Overlapped  ←
              ↓                   ↓
         re-order           Cluster Blocs
         Lines
              ↓                   ↓
         Lines               Blocs
         Indexes             Indexes
```