

**EXPERT SYSTEM**

CHAMAKURA SAI KUMAR

### Internship carried out from 27/05/2024 to 19/07/2024

**Internship tutors:** Mr. Muralidharan C and **Mr.** Yashodeepbharat Deshmukh

**Educational institution:** National Institute of Technology, Puducherry

**Internship host company:** Ashok Leyland

**ACKNOWLEDGMENTS**

I would like to take this opportunity to express my deepest gratitude to all the individuals and organizations that have supported and guided me throughout my internship journey. Their unwavering encouragement and assistance have been invaluable, and I am truly thankful for their contributions.

I extend my heartfelt appreciation to Ashok Leyland, for providing me with the opportunity to be part of their esteemed company during my internship. The experience gained and the knowledge acquired from working with this organization have been instrumental in shaping my professional growth.

I am grateful to Mr. Yashodeepbharat Deshmukh, Mr. Muralidharan C and the analytics team for their assistance and contributions that significantly impacted the success of this internship.

**Abstract**

This project aimed to build an Expert System for Ashok Leyland which will help in understanding the standards of Ashok Leyland. Extracting data from documents can be challenging due to the Various Data formats. It is challenging to automatically extract and interpret the data contained in PDFs because they are intended to be an output format rather than an editable one. Among the principal difficulties are:

* Instead of machine-readable text, PDFs may contain scanned images; in order to extract the content, optical character recognition (OCR) is necessary.
* It is challenging to automatically recognize and extract particular data pieces from PDFs because they lack semantic structure and information.
* Because PDFs have such a wide range of layouts and formatting, parsing each sort of document requires a different set of solutions.

**Table of content**

1. Introduction
2. Definition
3. Objective
4. Literature Survey
5. Workflow Overview
6. Detailed Workflow
   1. Converting PDF to Images using pdf2image
   2. Extraction of Text Using SuryaOCR
   3. Table Detection with yolov8
   4. Summarizing Cropped Images and Tables with boto3
   5. Cropping Images using opencv
7. Code Overview
8. Use Cases and Applications
9. Challenges and Solutions
10. Conclusion
11. References

**Introduction**

In realms of Modern industry, the ability to efficiently extract and process data from a variety of document formats is critical. Especially for organizations that are at the forefront of innovation and require sophisticated solutions to manage and utilize their massive data repositories.

As we begin the development of an AI Chabot for Ashok Leyland, we face the challenge of training our model on a wide range of documents provided by the company. These documents contain a variety of content types, such as text, images, tables, and flowcharts, all of which must be carefully extracted and converted into metadata for effective training.

It is not easy to extract data from PDF files, especially when those files contain scanned images and complex structures. When it comes to handling the complexity of these kinds of documents, traditional data extraction techniques frequently fall short. Python's strong libraries and tools become invaluable in this situation. A collection of libraries for handling different parts of PDF data extraction is available in Python. These libraries include tools for handling large amounts of documents, OCR (Optical Character Recognition) for scanned images, and parsing tools for structured data.

One of the most difficult challenges we face is extracting text from scanned images in PDFs. OCR technology is essential in this context because it converts text images into machine-readable characters. To accurately recognize and extract text from scanned documents, we can use libraries like Tesseract, an open-source OCR engine, in our Python-based extraction pipeline. This ensures that no valuable information is lost throughout the extraction process.

In addition to OCR, we need to take care of structured data extraction as well, including tables and flowcharts, in order to extract tabular data from PDFs and transform complicated tables into structured formats that are simple to handle and analyse, libraries like YoloV8 and Boto3 were created expressly for this purpose. Assuring that our metadata appropriately reflects the content of the original documents is possible for flowcharts and other graphical elements by using image processing tools like OpenCV to identify and extract pertinent information.

Ashok Leyland provides an enormous amount of documents, hence an automated and scalable method of data extraction is required. Python is a great option for developing a reliable extraction pipeline because of its adaptability and wide library support. With the use of libraries/models like PDF2Image for manipulating PDF files, Surya for optical character recognition, YoloV8 for extracting tables, and OpenCV for image processing, we can create a comprehensive solution that can manage the various sorts of material within the documents.

The extracted data can be utilized to train our AI chatbot model after it has been transformed into metadata. The metadata will supply the required background and facts, allowing the chatbot to comprehend and efficiently react to user inquiries. In addition to improving the chatbot's functionality, this procedure guarantees that it has a thorough comprehension of Ashok Leyland's knowledge base and documentation.

**Key Python Libraries/Models for PDF Data Extraction:**

**pdf2image:** A wrapper for the command-line programs pdftoppm and pdftocairo that converts PDFs to PIL Image lists.

**SuryaOCR**: A robust set of open-source OCR tools with sophisticated capabilities for text extraction and layout analysis

**YOLO V8**: Is a cutting-edge computer vision model created by Ultralytics that is renowned for its exceptional powers in segmentation, classification, and object detection.

**BOTO3:** A model used Use AWS services to summarize content.

**Opencv:** It is a huge open-source library for [computer vision](https://www.geeksforgeeks.org/a-quick-overview-to-computer-vision/), [machine learning](https://www.geeksforgeeks.org/ml-machine-learning/), and [image processing](https://www.geeksforgeeks.org/components-of-image-processing-system/).

**Definitions**

**Contouring**

The technique of locating and defining the borders or contours of objects inside an image is referred to as contouring in the area of deep learning. It is essential for many computer vision tasks, such as medical imaging, object recognition, and image segmentation.

**Natural language processing**

The artificial intelligence field of natural language processing (NLP) makes it possible for computers to comprehend, interpret, and produce human language. It includes speech recognition, language translation, sentiment analysis, and text analysis. NLP makes it possible for machines to communicate with people in a more intuitive, natural way.

**Transformers**

Transformers are neural networks that learn context and understanding through sequential data analysis. The Transformer models use a modern and evolving mathematical techniques set, generally known as attention or self-attention. This set helps identify how distant data elements influence and depend on one another.  
  
Text is transformed into numerical representations known as tokens in the Transformer paradigm. Next, each token is looked up from a word embedding database to become a vector. Using a parallel multi-head attention technique, each token at each layer is contextualized with other (unmasked) tokens within the context window. This makes it possible to increase the signal for critical tokens and decrease the signal for less significant tokens.

**Large language model**

A large language model (LLM) is a computer model that is renowned for its ability to carry out natural language processing tasks, such as classification and general-purpose language synthesis. These abilities are taught to LLMs through a computationally intensive self-supervised and semi-supervised training process based on language models, which teaches them statistical connections from vast amounts of text. By consistently forecasting the subsequent token or word in an input text, LLMs—a form of generative AI—can be used to generate text.

**Objective**

The main goal of data extraction from PDF documents is to make it possible to create sophisticated chatbots for expert systems that are capable of having insightful, conversation-rich exchanges. Using the vast amount of information found in PDFs, these Chabot’s are able to:

**Provide Comprehensive Answers**  
Through the extraction of pertinent data from PDFs, the chatbot may access a vast knowledge library, allowing it to furnish users with thorough and perceptive answers. This goes beyond simple keyword matching to include comprehending and synthesizing the essential content. The chatbot can provide thorough and insightful explanations by interpreting the context and subtleties of the data. This feature is very helpful for providing complex query answers since it enables the chatbot to give correct, comprehensive information by referencing a variety of sources. As a result, consumers get better educated and pertinent replies to their queries.

**Engage in Contextual Dialogue**  
The chatbot can have dynamic, contextual discussions with users since it can extract and analyze PDF material. It is able to understand the user's intention, pose clarifying queries, and offer answers that are customized to meet certain information demands. This feature makes sure that conversations are extremely tailored and pertinent at the same time. The chatbot may provide accurate and insightful responses by understanding the subtleties and context of the user's questions, which improves the user experience in general. This makes it an effective tool for answering complicated questions and giving thorough, relevant information.

**Offer Personalized Insights**  
Through the analysis of user inquiries and the extraction of data from PDFs, the Chabot is able to provide customized insights, recommendations, and advice that are suited to the individual needs of each user. With this feature, the chatbot becomes a real expert assistant that can comprehend and respond to certain needs. The chatbot guarantees the accuracy and high relevance of its responses by utilizing comprehensive data from PDFs. Through more meaningful and productive interactions and the provision of precise information and support, this tailored approach improves the user experience.

**Handle Complex Queries**

The chatbot can answer complicated questions requiring multi-source research by extracting and correlating data from different PDFs. With the help of this sophisticated capability, the chatbot can now perform tasks beyond basic search and respond with detailed information derived from extensive data research. Because of this, the chatbot becomes an effective instrument for compiling data from many documents and providing accurate responses. With precise and thorough answers to intricate inquiries, this improved feature greatly enhances the user experience.

**Literature Review**

Text extraction from a variety of documents is frequently necessary for the construction of expert systems, but this can be a difficult operation. In order to overcome this, scholars have resorted to studying the foundations of natural language processing (NLP) in order to comprehend the process of extracting text while keeping its relationships and meaning intact.

The stages required to analyse and comprehend human language are described by the Natural Language Processing (NLP) pipeline. The links between the data items are maintained by this procedure through the use of techniques like tokenization and word embedding. These methods are especially helpful when creating expert systems that seek to meaningfully generate or comprehend human language while maintaining all of the underlying relationships.

The GPT (Generative Pre-Trained Transformer) family of big language models is one notable development in NLP. These models, which are based on the transformer architecture, are made to produce language that resembles that of a person by pre-training on enormous volumes of textual data and using that data to identify patterns. Prior to being optimized for particular tasks, the models can acquire generic language patterns through this pre-training.

NLP has undergone a revolution thanks to the transformer architecture, which forms the basis of models like BERT and GPT. Compared to conventional recurrent neural networks, this neural network architecture processes and generates text more efficiently by using self-attention mechanisms to comprehend the relationships between words in a phrase.

Another essential element of contemporary NLP methods is word embedding. Machine learning models are able to better comprehend the subtleties of language because to these numerical representations of words, which capture the semantic meaning and links between them. Word embedding facilitate enhanced performance on tasks like sentiment analysis, text categorization, and language synthesis by mapping words to dense vector representations.

Last but not least, the idea of fine-tuning is a transfer learning strategy used in deep learning to modify previously learned models for particular tasks or datasets. Performance can be greatly enhanced by fine-tuning, particularly if the target dataset is tiny, by utilizing the knowledge gathered from the first large-scale training. Numerous fields, such as computer vision and natural language processing, heavily rely on this methodology.

All things considered, the literature review emphasizes how critical it is to comprehend the foundational ideas of natural language processing (NLP), including tokenization, word embedding, and the transformer architecture. It also discusses the latest developments in language models, such as GPT, and how fine-tuning approaches are applied. The creation of expert systems that need to extract and comprehend text from several sources can benefit from these findings.

Initially, the PyMuPDF library, often known as Fitz, was used to handle PDF. It is an easy and efficient way to read, write, and manipulate PDFs. However, it is primarily concerned with text extraction and does not provide comprehensive support for image-based or scanned documents. Scanned or image-based PDFs.

To address these restrictions, the OpenCV (Open Source Computer Vision Library) was investigated. OpenCV is a highly optimized library for real-time applications. It can process photos and run a wide range of algorithms. It detects patterns, shapes, and characteristics in images, which is useful when working with scanned or image-based PDFs.   
  
Before going deep into OpenCV, it was required to grasp a few key concepts. LLM (Low-Level Motion) is one such idea that is important for understanding motion perception and object tracking in videos. It can be used to static photos by treating successive images like video frames.

Finally, the idea of contours in image processing was learned. Contours are just the outline of an object within a picture. Understanding contours is essential for OpenCV's picture segmentation, object identification, and recognition tasks.

For table extraction, Tabula can be used to detect the tables and reconstruct them. However, while reconstructing the table, we are unable to maintain the meaning and the relationships between the cells. That's the reason why YOLOv8 can be used to crop the tables from the documents. When reconstructing the table, we can extract the cells and store them in a JSON format, which is a Python dictionary. This allows us to maintain the relationships between the cells using key-value pairs.

**Required Libraries and packages**

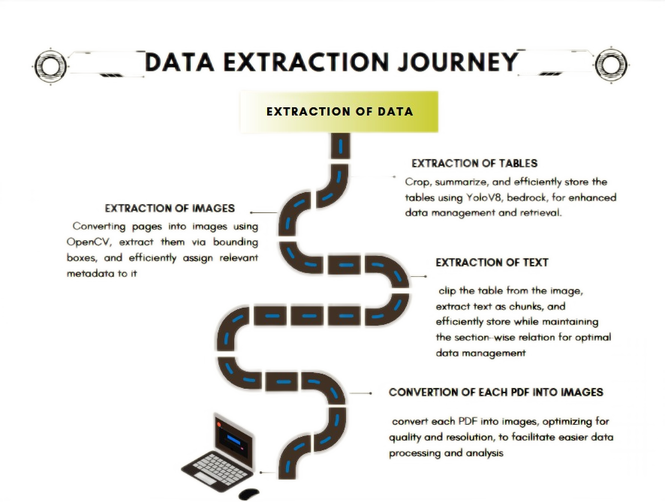
The following Python libraries are needed in order to implement the solution:

* Suryaocr: Text extraction using Optical Character Recognition (OCR).
* pdf2image: Generate images from PDF pages.
* Yolov8: Find tables within pictures.
* boto3: Use AWS services to summarize content.
* OpenCV-python: Used to detect Images in the PDF pages

**You can use pip to install these libraries.**

pip install suryaocr pdf2image yolov8 boto3 opencv-python

**Workflow Overview**

****

**Detailed Workflow**

**A. Using pdf2image to Convert PDF to Images**

**PDF to Image Conversion:** To convert a PDF document into an individual image, utilize the pdf2image library. The content of the PDF can be rendered into a visual representation using this library.  
  
 **Image Generation:** The PDF is converted into individual images on each page. This guarantees that every page's content is distinct and capable of being handled separately.  
  
**OCR preparation:** One of the most important steps before using OCR is to convert PDF pages to pictures. This conversion is required because OCR models need input in the form of images.

**B. Extraction of Text Using SuryaOCR**

**PDF Conversion:** The first step in the process is to turn each PDF page into an image. Usually, libraries that can convert PDF material into picture formats like JPEG or PNG are used for this.  
 **OCR Model Application:** These previously processed pictures are subsequently subjected to the Surya OCR model. With training, this model learned to identify and extract text from pictures.  
**Text Extraction:** The OCR model scans the image's content and transforms it into text that is readable by computers. In order to preserve the original layout and formatting, this entails identifying individual characters and words and maintaining their relative placements.

**C.Yolov8 Table Detection**

**Table Detection:** This process main objective is to locate tables inside the images that were created from the PDF pages. In order to extract data from the structured material in the papers, this is an essential step.  
  
**YoloV8 Model:** For this, the You Only Look Once (YoloV8) model is employed. Popular object detection algorithm YoloV8 is renowned for its accuracy and quickness.  
  
**Model Loading:** A pre-trained YoloV8 model is loaded during this procedure. This model is quite good at this task because it was particularly trained to identify tables in photos.  
Model Application: The model is applied to every image after it has been loaded. This entails producing predictions by passing the visual input through the neural network of the model.  
  
**Table Region Detection:** This technique uses the YOLOv8 model to find areas in the image where tables are present. By giving these regions' bounding box coordinates, it successfully isolates the tables from the remainder of the image's content.  
  
**Data Extraction:** Following the identification of the table regions, additional processing can be carried out to extract and organize the data contained in these tables. This can entail using more OCR to interpret the text contained in the tables as well as algorithms to comprehend the arrangement of the rows and columns.

**D. Image Cropping with OpenCV**

**Table Cropping:** This step's primary goal is to crop the tables that have been identified from the pictures. To do this, the table regions must be isolated for additional processing.

**OpenCV Library:** This is accomplished by using the OpenCV (Open Source Computer Vision) library. For tasks involving image processing and computer vision, OpenCV is an effective library.

**Region Extraction:** The associated regions are taken out of the images once the bounding boxes have been located. The image cropping feature in OpenCV is used for this.

**Image Output:** A collection of cropped images with tables from the original PDF document are what this step produces. Subsequent processing, like data extraction from the tables, can be done with these cropped images.

**Data Extraction:** Extracting and organizing the data from these tables is the last stage. Further OCR to read the text within the tables and algorithms to comprehend the row and column structure of the tables may be required for this.

**E. Using Boto3 to Summarize Cropped Images and Table:**

**Overview of Table Content:** The following stage is to present a summary of the tables contents once they have been identified and cropped from the pictures. This is taking the text contained in the tables and condensing it.

**Boto3 Library:** library is utilized for this objective. Programmers working in Python can create software that utilizes AWS services with Boto3, the Python Software Development Kit (SDK) for Amazon Web Services (AWS).

**Text Extraction with Textract:** The cropped table images can have their text extracted using Textract. An automatic text and data extraction tool for scanned documents is called Textract.

**Text Summarization Using Comprehend:** Following text extraction, the extracted text is summarized using AWS Comprehend. Natural language processing, or NLP, is used by Comprehend to identify patterns and connections in text.

**Data Analysis:** A synopsis of the tables' contents is given in the summarized text. This can be utilized for additional processing or analysis, based on the particular needs of the assignment.

**Output Generation:** A series of summaries that give a quick rundown of each table's contents in the original PDF document are the process's final product. The essential details in each table can be rapidly understood by using these summaries.

**Code Overview**

**TABLE EXTRACTION**

**Using Bedrock**

**Using** **YOLOv8**

**Use Cases and Applications**

There are several situations in which the solution can be used, such as:

* Document management: Using automation to automatically extract text and tables for indexing and searching from massive amounts of PDF documents.
* Data analysis: Taking tables out of research papers and reports and condensing them so that they can be further examined.
* Digital archiving: Transferring paper documents to a digital format and extracting important data for future reference and storage.
* Business intelligence: obtaining information for integration into BI systems by extracting data from receipts, invoices, and other business documents.
* Supply chain management: Extracting data from shipping manifests, purchase orders, and inventory reports to optimize supply chain operations and logistics.
* Marketing analysis: Compiling information to guide marketing campaigns and strategies from surveys, market research reports, and social media analytics.
* Academic research: gathering information for research projects and literature reviews by extracting and organizing data from dissertations, conference papers, and academic journals.
* Human resources: Using data extracted from performance evaluations, resumes, and job applications to expedite hiring and staff management procedures.

**Challenges and Solutions**

There are some number of difficulties while implementing text, table, and image extraction from PDFs. Here's a thorough examination of the problems encountered and the methods we used:

**Text Extraction:**

Initially, text from PDFs was extracted using libraries like py2pdf and pdfplumber. Nevertheless, these tools had trouble handling intricate layouts like nested tables and multi-column text. Variations in font styles and image quality also had an impact on the accuracy of optical character recognition (OCR). SuryaOCR was utilized in order to resolve these problems. By utilizing sophisticated pre-processing methods like binarization and denoising, this tool greatly increased OCR accuracy and handled complex layouts more skilfully.

**Table Extraction:**

There were additional difficulties in extracting tables. At first, tables in a variety of document formats could not be reliably detected by tools such as Tableau, Tesseract and Camelot. The accuracy of the table detection was a significant worry. Table extraction accuracy significantly increased when YOLOv8, a more reliable and well-trained model created especially for object detection, was used instead. With greater accuracy, YOLOv8 along with the Bedrock could consistently recognize and extract tables from a variety of document layouts and also summarize them.

**Image Extraction:**

The first issue we encountered when attempting to extract images from PDFs was that the entire PDF page would extract as a single image without actually extracting the image. Next, we attempted Fitz for image extraction, which initially produced good results but had problems with consistency and image quality for older files. Eventually, we discovered OpenCV, a powerful computer vision library, and this was the solution. OpenCV introduced more advanced techniques for image enhancement and extraction, among other better image processing tools. Following this modification, high-quality and dependable image extraction from PDFs was achieved.

**Conclusion**

The methodology used in this study is not only comprehensive, but also unique, as it makes use of Python, a high-level, interpreted programming language noted for its simplicity and readability. Python's wide library support, which includes OpenCV, YOLOv8, boto3, SuryaOCR, and pdf2image, is critical to the success of this strategy. These libraries and models are not picked at random; rather, they are chosen based on their specialized characteristics, which contribute to the overall efficiency and accuracy of data extraction.

OpenCV, or Open Source Computer Vision, is a collection of programming functions primarily geared toward real-time computer vision. It is used in this manner because of its excellent image processing capabilities. It may alter images in a variety of ways, improving the quality of images taken from PDF files and thereby increasing the accuracy of the OCR process.

YOLOv8, or You Only Look Once version 8, is an advanced object detection model. This approach detects tables within documents. Its ability to reliably recognize and extract tables from a wide range of document types greatly increases the method's adaptability.   
  
Boto3 is a Python Software Development Kit (SDK) from Amazon Web Services (AWS). It enables Python developers to create applications that utilizes services such as Amazon S3, Amazon EC2, and more. Boto3 is utilized in this strategy to combine the data extraction process with cloud services. This improves the method's scalability while also providing robust storage capabilities.

SuryaOCR is another important aspect of this approach. It is an OCR application that improves text extraction accuracy by utilizing advanced pre-processing techniques. Low image quality, complex layouts, and uncommon fonts can all make the OCR process difficult. SuryaOCR addresses these issues by ensuring that the extracted text from PDF files is as precise as possible.

Pdf2image is a simple yet effective program for converting PDF files to images. This is an important stage in the data extraction process since it allows other programs, like as OpenCV and SuryaOCR, to handle the data contained within the PDF documents.

The combination of these tools and models automates the extraction process, minimizing manual labour and increasing production. This is a significant advantage in many industries where vast amounts of data must be extracted from PDF documents, including document management, data analysis, digital archiving, and business intelligence.  
  
The technology enhances not only the efficiency of data extraction, but also the accuracy and reliability of the extracted information. This is especially crucial in industries, where data accuracy can have a substantial impact.

Finally, this paper describes a complete and efficient method for extracting data from PDF files using Python. The method's use of sophisticated models and libraries, together with its interaction with cloud services, make it a useful tool for a variety of applications. Whether extracting text for a digital archive, tables for a business intelligence report, or photos for a document management system, this process produces accurate and consistent outcomes. It is a big leap in data extraction, displaying the strength of Python and its libraries. It demonstrates automation's ability to revolutionize how we work and interact with data.

**References**

1.**SuryaOCR Documentation:**

https://github.com/VikParuchuri/surya  
 2.**Documentation pdf2image:**

https://pypi.org/project/pdf2image

**3.Documentation for Yolov8:**

https://github.com/ultralytics/ultralytics

**4.Documentation for Boto3:**

https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

**5.Documentation for OpenCV:**

https://github.com/opencv/opencv