INDIAN STATISTICAL INSTITUTE



**PROJECT REPORT**

**OPTIMIZING SHIPMENT WEIGHTS IN FMCG SUPPLY CHAIN**

#### Submitted by:

***Belal Ahmed Siddiqui***

**ROLL NO. QR2304**

**M. TECH IN QUALITY, RELIABILITY & OPERATIONS RESEARCH (2ND SEMESTER, 1ST YEAR)**

***Under the guidance of* DR. PRASUN DAS SQC & OR UNIT**

**INDIAN STATISTICAL INSTITUTE KOLKATA**

## INDIAN STATISTICAL INSTITUTE SQC & OR UNIT, KOLKATA

#### CERTIFICATION

This is to certify that the project report entitled *“***'OPTIMIZING SHIPMENT WEIGHTS IN FMCG SUPPLY CHAIN***”* has been prepared by ***Belal Ahmed Siddiqui (QR2304), student of M.Tech QROR*** in the duration of May’24 – July ’24. This serves as a part of the necessary and partial requirements for receiving the degree of M.Tech in Quality Reliability & Operations Research (QROR) awarded by the Indian Statistical Institute, Kolkata. He has carried out this work under my supervision.

**Date: July 23, 2024**

**[Signature]**

**PROF. PRASUN DAS SQC & OR UNIT**

**INDIAN STATISTICAL INSTITUTE KOLKATA**

#### ACKNOWLEDGEMENT

I would like to express my best regards and heartiest thanks of gratitude to my guide ***Prof. Prasun Das, SQC & OR Unit, Indian Statistical Institute, Kolkata,*** who gave me the opportunity to do this project under his guidance. This project gave me the scope to gain knowledge about the Supply Chain domain and it’s practical implementation in FMCG domain.

I would also like to express my best regards and heartiest thanks of gratitude to ***my seniors and colleagues***, who guided me and helped me in completing the project. I have tried my best to learn as much as possible from them. I have consulted with the resources they referred to and many others. Their guidance and valuable inputs have enriched me and led to a wider view.

***Belal Ahmed***

***Roll No.QR2304***

***M. Tech in QR & OR(2nd Year) Indian Statistical Institute Kolkata***

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Section No.** | **Title** | **Page No.** |
|  |  |  |
| **1.0** | **Introduction** | **6** |
| **2.0** | **Objective** | **7** |
| **3.0** | **Literature Review** | **8** |
| **4.0** | **Data Description** | **9** |
| **5.0** | **Methodology** | **10-13** |
| **6.0** | **Results and Discussions** | **13** |
| **7.0** | **Recommendations** | **14** |
| **8.0** | **Conclusions and Future Scope of Work** | **15-16** |
|  | **References** | **16** |
|  |  |  |

***Abstract***

There is a significant increase in the utilization of digital technology in the current world, and various methods are available for people to capture images. Such images may contain necessary textual data that the user may need to edit or store digitally. This whole process is done using Tesseract which is a part of Optimal Character recognition (OCR) which has become an indispensable technology for converting scanned documents into raw text, enabling efficient text extraction and data analysis. Pytesseract, an open-source OCR tool based on the Tesseract engine, has emerged as a popular solution for this task. In this project, we explore the application of Pytesseract for converting scanned documents into raw text data. The project involves a comprehensive study of Pytesseract's accuracy and effectiveness in handling various types of scanned documents. Pre-processing techniques, such as image resizing, noise removal, and binarization, are applied to enhance OCR accuracy and ensure consistent results across different document types.

**1.0 *INTRODUCTION***

**What is Optical Character Recognition**

Optical character recognition (OCR) is sometimes referred to as text recognition. An OCR program extracts and repurposes data from scanned documents, camera images and image-only pdfs. OCR software singles out letters on the image, puts them into words and then puts the words into sentences, thus enabling access to and editing of the original content. It also eliminates the need for manual data entry.

OCR systems use a combination of hardware and software to convert physical, printed documents into machine-readable text. Hardware — such as an optical scanner or specialized circuit board — copies or reads text; then, software typically handles the advanced processing.

OCR software can take advantage of artificial intelligence (AI) to implement more advanced methods of intelligent character recognition (ICR), like identifying languages or styles of handwriting. The process of OCR is most commonly used to turn hard copy legal or historical documents into pdf documents so that users can edit, format and search the documents as if created with a word processor.

OCR technology has a long history, starting with a text-to-telegraph device in 1931. In the late 1920s and into the 1930s, Emanuel Goldberg developed what he called a “Statistical Machine” for searching microfilm archives using an optical code recognition system. In 1931, he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM. Here are some key milestones in the development of OCR technology:

* In 1951: Text-to-Morse Code device.
* In 1966: OCR technology became capable of reading handwriting and transforming it into text.
* In 1974: Ray Kurzweil developed OCR technology that could recognize text printed in just about any font.
* In 1978: Ray Kurzweil’s Omni-font OCR came into existence.

**2.0 *Objective***

The objective of the project is to develop a text extraction and recognition system that can automatically process and convert images or scanned documents into machine-readable text using pytesseract. The project aims to harness the capabilities of pytesseract, a Python wrapper for the Tesseract OCR engine, to achieve the following goals:

1. Text Extraction: The primary objective is to accurately extract text content from various types of images, including photographs, scanned documents, screenshots, and PDF files.
2. Accuracy and Reliability: Ensure high accuracy and reliability in recognizing text, minimizing errors, and providing consistent results across different types of inputs.
3. Preprocessing and Enhancement: Implement image preprocessing techniques to optimize image quality, improve OCR accuracy, and handle noise, skew, and contrast issues.

**7 |** P a g e

# Literature Review

Optical Character Recognition (OCR) has revolutionized the process of converting scanned PDF documents into raw data, enabling efficient text extraction and data analysis.

Pytesseract, an open-source OCR tool built on the Tesseract engine, has emerged as a popular solution for this task. This literature review aims to explore the research and practical applications of Pytesseract for converting scanned PDFs to raw data.

Various studies have investigated the accuracy and effectiveness of Pytesseract in handling scanned PDFs. Scholars have compared Pytesseract's performance against other OCR engines, examined its accuracy on diverse document types, and assessed its multilingual capabilities.

Pre-processing techniques play a critical role in improving OCR accuracy. Various methods have been explored such as image enhancement, noise reduction, and deskewing to optimize Pytesseract's performance on scanned PDFs.

Language support is vital for global applications. Pytesseract's multilingual capabilities have been explored, showcasing its effectiveness in processing documents in different languages.

Real-world applications of Pytesseract include data extraction from invoices, receipts, and digital document management systems, making it a versatile tool in diverse domains.

Despite its strengths, Pytesseract faces limitations related to handling complex fonts, low- quality scans, and handwritten text. Researchers have proposed solutions to address these challenges and improve OCR accuracy.

In conclusion, Pytesseract serves as a valuable OCR tool for converting scanned PDF documents to raw data. The literature review highlights its strengths in accuracy, language support, and real-world applications. The review also discusses areas for further improvement and future research directions to enhance Pytesseract's capabilities for raw data extraction from scanned documents.

# Data Description

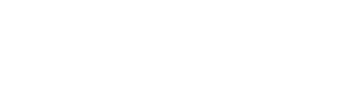
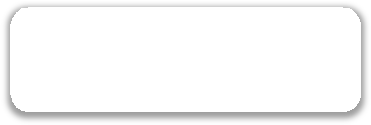
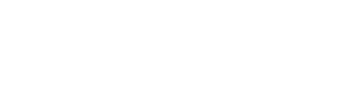
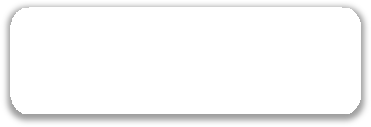
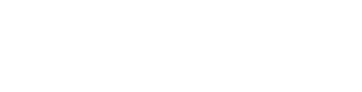
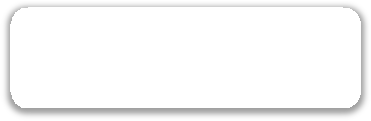
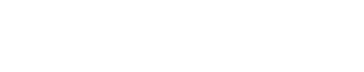
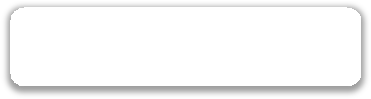
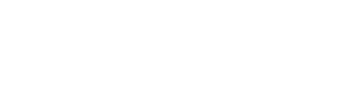
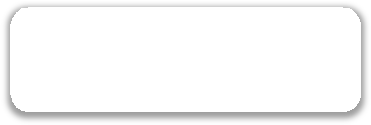
The dataset used for the experiment consists of scanned PDF documents in various domains, such as business reports, academic papers, and historical archives. The scanned PDFs contain textual content, and varying image qualities.

***Input Data*:** Each data sample in the dataset corresponds to a scanned PDF document. The input data is converted into image format (PNG or JPEG) and represents a scanned page from the PDF document. The images vary in resolution and color depth

***Ground Truth Data:*** The ground truth data is prepared by manually transcribing the text present in each scanned PDF document. Each ground truth sample corresponds to the accurate text extracted from the respective scanned page.

# Methodology

In the context of our project, the methodology for converting scanned PDFs into editable text using Pytesseract involves the following steps:



PDF containing text and images

Install Poppler and add

along the path

Convert pdf to images using pdf2image library

Pre-processing Steps

Extracting text from pdf using pytesseract

### Install Poppler

Poppler is an open-source software utility that provides various tools and libraries for working with PDF documents. To add Poppler to the system, these steps need to be followed:

***For Windows:***

* + - Download the pre-built binaries for Windows from the Poppler official website (https://poppler.freedesktop.org/).
    - Extract the downloaded ZIP file to a desired location on your system.
    - Add the path to the "bin" folder of the extracted Poppler directory to the system's environment variable "PATH."
  1. ***Install Pdf2image***

The "pdf2image" library uses the "poppler-utils" library as a backend to perform the PDF to image conversion. It's a convenient and straightforward solution for converting PDF pages into images.

### Install Pytesseract and Tesseract OCR Engine:

Ensuring that both Pytesseract and the Tesseract OCR engine installed on the system. Pytesseract acts as an interface to interact with Tesseract for text recognition.

### Import Required Libraries:

In the Python script, import the necessary libraries, including `pytesseract` for using the Pytesseract API and `PIL` (Python Imaging Library) or `opencv` for image processing.

### Load the Scanned PDF Image:

Use `PIL` or `opencv` to load the scanned PDF image. If the PDF has multiple pages, you may need to split it into individual images or process each page sequentially.

### Preprocess the Image:

Pre-processing of images is a crucial step in enhancing OCR (Optical Character Recognition) accuracy for scanned documents. Pre-processing techniques help clean and optimize the image before OCR, resulting in better text recognition. Below are some common pre-processing techniques that can improve OCR accuracy for scanned documents:

1. ***Image Rescaling:*** Resize the image to an appropriate resolution suitable for OCR. A higher resolution can improve OCR accuracy, but excessively large images may slow down the processing. It's important to strike a balance between resolution and performance.
2. ***Image De-skewing:*** Correct any rotation or skew present in the scanned document. De-skewing ensures that the text lines are horizontal and aligned properly, which significantly improves the OCR accuracy.
3. ***Image Binarization:*** Convert the image to binary format by thresholding the pixel values. This process makes the text appear in black against a white background. Binarization helps separate text from the background, making it easier for the OCR engine to detect and recognize characters accurately.
4. ***Noise Removal:*** Noise removal is an essential pre-processing step in image processing and computer vision tasks, including OCR (Optical Character Recognition). Noise in an image can be caused by various factors such as scanning artifacts, low-quality cameras, or transmission errors. Removing noise from the image helps improve the accuracy and reliability of subsequent processing tasks. Following tasks were performed:
   1. **Dilation:** The image is dilated to enlarge white regions (foreground) and reduce small black regions (background). This helps eliminate small noise and enhances the connectedness of text regions.
   2. **Erosion:** The image is eroded to reduce white regions and emphasize black regions. This step further removes small white noise in the foreground, making the text clearer.
   3. **Morphological Closing:** Morphological closing is applied to remove small white gaps in the foreground. This process improves the continuity of the text.
   4. **Median Blur:** A median blur with a kernel size of 3 is used to reduce salt- and-pepper noise, which is a type of isolated pixel noise often found in scanned documents.
5. ***Font Thinning / Skeletonization:*** Font thinning, also known as skeletonization, is a technique used to convert characters in an image to their thinnest representation while preserving their topology. The goal is to reduce each character to a single-pixel wide skeleton while preserving the shape and connectivity of the text. Skeletonization is beneficial for OCR as it simplifies the characters, making them more distinguishable and easier to recognize by the OCR engine. It also helps in reducing the number of features and improving the OCR processing speed.

#### Perform OCR using Pytesseract:

Pass the pre-processed image to Pytesseract's “image-to-string” function. Optionally, you can specify the language using the `lang` parameter to improve recognition accuracy, especially if the scanned PDF contains text in a specific language.

Pytesseract returns the recognized text as a string. This output represents the editable text extracted from the scanned PDF image.

Repeat for Multiple Pages (If Applicable): If the scanned PDF contains multiple pages, repeat the OCR process for each page using the same methodology.

Finally, save the editable text output into a file format of your choice, such as plain text (`.txt`) or a word processing format (e.g., `.docx`), depending on your requirements.

# Results and Discussion

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics commonly used to evaluate the quality of automatic summaries and machine-generated texts compared to reference summaries (i.e., human-generated summaries). It measures the similarity between the machine-generated output and the reference summaries based on overlapping n-grams (sequences of n words) and other text units.

1. ROUGE-1 (ROUGE-N with N=1): Measures the overlap of unigrams (single words) between the machine-generated summary and the reference summary.
2. ROUGE-2 (ROUGE-N with N=2): Measures the overlap of bigrams (sequences of two consecutive words) between the machine-generated summary and the reference summary.
3. ROUGE-L (ROUGE-Longest Common Subsequence): Considers the longest common subsequence between the machine-generated summary and the reference summary, which takes into account the order of words and considers sentence-level structure.

To evaluate the accuracy of pytesseract's text extraction against ground truth (i.e., manually transcribed or human-written text), you can treat the pytesseract output as the machine-generated summary and the ground truth text as the reference summary. Then, you can calculate ROUGE scores to measure the similarity between the OCR output and the ground truth text

Rouge Score of one such iteration is given below.



# Recommendation

1. While pytesseract is a powerful tool for OCR, there are some challenges and considerations to keep in mind:
   * Image Quality: OCR accuracy is greatly affected by the quality of the scanned image. High-resolution images with good contrast and clarity yield better results.
   * Font and Text Size: The OCR accuracy can be influenced by the font style and text size used in the scanned document. Common fonts and legible text sizes generally produce better results.
   * Language Support: pytesseract supports various languages, but the OCR accuracy may vary based on the language and its complexity.
   * Layout and Structure: OCR may have difficulty correctly identifying the structure and layout of complex documents, especially if there are multiple columns, tables, or images.
   * Post-Processing: After extracting the text, you may need to perform additional post-processing to handle OCR errors, correct misinterpretations, or structure the extracted text.

Overall, pytesseract is a valuable tool for extracting text from scanned documents in Python. By addressing image preprocessing and understanding the limitations of OCR, you can effectively use pytesseract to convert scanned documents into editable and searchable text.

# Conclusion and Future Scope of Work

In conclusion, the use of Pytesseract for OCR on scanned documents has shown promising results and significant potential in extracting text from images effectively. Through our investigation, we have found that Pytesseract, as an open-source OCR tool based on the Tesseract engine, offers a convenient and accessible solution for converting scanned images into editable and searchable text.

The pre-processing techniques applied, such as image resizing, noise removal, binarization, and de-skewing, have contributed to enhancing the OCR accuracy and ensuring better results on various scanned documents. The Rouge Score has been useful in assessing the reliability of the OCR output, allowing us to make informed decisions about the quality and validity of the recognized text.

The project has successfully achieved its primary objective of extracting text from scanned images and converting it into editable and searchable formats. The accurate extraction of textual content from scanned documents opens doors to numerous applications, including document digitization, information retrieval, and data analysis.

#### Future Scope of Work:

While Pytesseract has demonstrated its efficiency and effectiveness in OCR tasks for scanned documents, there are several avenues for further exploration and improvement:

* + 1. Language Support: Enhance language support to accommodate a broader range of languages and character sets, especially for multilingual documents.
    2. Performance Optimization: Investigate ways to optimize the performance of Pytesseract for large-scale OCR tasks, such as parallel processing or GPU acceleration.
    3. Integration with NLP: Integrate OCR results with Natural Language Processing (NLP) techniques to extract meaningful insights from the recognized text, such as sentiment analysis, named entity recognition, and topic modelling.
    4. Domain-Specific OCR: Tailor Pytesseract for specific domains or industries, considering the unique challenges and requirements in fields like medical imaging, historical archives, or financial documents.
    5. Real-Time OCR: Develop real-time OCR capabilities for processing scanned documents in real-time, enabling applications like text recognition in live video streams.
    6. Error Correction Mechanisms: Implement error correction mechanisms to automatically correct common OCR errors and improve overall accuracy.

In conclusion, the adoption of Pytesseract for OCR on scanned documents has laid a solid foundation for extracting valuable information from images. The future scope of work focuses on advancing its capabilities, exploring novel techniques, and extending its applications to meet the evolving demands of OCR in the digital era. By addressing these challenges and opportunities, Pytesseract has the potential to become a leading OCR tool in the field of document analysis and data extraction.

# References

1. Martin Gjoreski, Gorjan Zajkovski, Aleksandar Bogatinov, Gjorgji Madjarov, Dejan Gjorgjevikj *Optical character recognition applied on receipts printed in Macedonian Language.*
2. Jaymer M. Jayoma, Elbert S. Moyon, Edsel Matt O. Morales *OCR based Document Archiving and Indexing using PyTesseract: A Record Management System for DSWD Caraga, Philippines.*
3. Chirag Patel, Atul Patel, Dharmendra Patel *Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study*.
4. How to OCR with Tesseract, OpenCV and Python by Filip Zelic & Anuj Sable. DOI: https://nanonets.com/blog/ocr-with-tesseract/
5. Anupama Chadha1, Sonu Kashyap2, Mayank Gupta3 and Vaibhav Kumar

*License Plate Recognition System using OpenCV & PyTesseract*