ABSTRACT

Educational institutions are undergoing a digital transformation, leveraging advancements in technology to enhance operational efficiency. In this era of rapid technological progress, our research aims to streamline and automate the extraction of vital information from educational certificates using Optical Character Recognition (OCR) technology. By incorporating algorithms and techniques, including Pytesseract, global thresholding, and template matching, our solution ensures accuracy and reliability across various educational contexts. Our system not only reduces errors but also minimizes time-consuming manual processes, thereby improving overall efficiency. The utilization of advanced parameters like OEM 3 and PSM 6 further optimizes the OCR process, ensuring accurate extraction of information from certificates.

Designed for seamless integration with existing systems, our solution is built for scalability and interoperability, facilitating future enhancements and upgrades. By automating the extraction process and ensuring accuracy and reliability, our research contributes to the ongoing digital transformation in educational institutions, paving the way for enhanced operational efficiency and productivity.

Keywords: Digital transformation, extraction, certificates, OCR technology, algorithms, techniques, Pytesseract, global thresholding, template matching, interoperability, future enhancements, upgrades, automation..

TABLE OF CONTENTS

ABSTRACT i

LIST OF TABLES iii

LIST OF FIGURES iv

LIST OF ABBREVIATIONS v

1. INTRODUCTION 1

   1.1 Template Matching 2

   1.2 CNN Architecture based Certificate Classification Model 3

   1.3 Preprocessing 5

   1.4 OCR and PYTESSERACT 6

2. LITERATURE SURVEY 8

3. PROBLEM STATEMENT 14

4. METHODOLOGY 15

   4.1 Data Acquisition 15

   4.2 Information Extraction 16

   4.3 Data Refinement 17

   4.4 Data Serialization 17

   4.5 Data Integration and Reporting 18

5. IMPLEMENTATION 19

6. RESULT ANALYSIS 27

7. CONCLUSION 31

8. FUTURE SCOPE 31

9. REFERENCES 33

APPENDIX 1 : DESIGN DOCUMENTS 35

APPENDIX 2 : SAMPLE CODE 39

APPENDIX 3 : PLAGARISM REPORT 52

LIST OF TABLES

Fig. No.	Title of table	Page No.
5.1	OEM Description	24
5.2	PSM description	24
6.0	Template1 Output	29
6.2	Template2 Output	29
6.3	WER and CER on the Template1 data	30
6.4	WER and CER on the Template2 data	30

LIST OF FIGURES

Fig. No.	Title of Figure	Page No.
4.1	Methodology	21
4.2	Architectural Diagram	24
5.1	Template 1	26
5.2	Template 2	26
5.3	Gray scale Image	27
5.4	Original Image	27
6.1	Optimal Scaling Value for Template matching	33

LIST OF ABBREVIATIONS

OCR	Optical Character Recognition
NLP	Natural Language Processing
OEM	Optical Engine Mode
PSM	Page Segmentation Mode
REGEX	Regular Expressions
WER	Word Error Rate
CER	Character Error Rate

1. INTRODUCTION

Educational institutions now have the difficult duty of processing a wide range of educational certificates, from academic transcripts to course completion certificates, in an effective manner in the digital age. However, there are several obstacles to overcome in the manual extraction of information from these certificates, such as laborious procedures, mistakes in data entry, and the requirement for substantial human interaction.

In order to overcome these obstacles, our initiative uses cutting-edge technology, specifically Optical Character Recognition (OCR), to completely transform the processing of educational certificates. Our goal is to use OCR to automate the extraction of important data from educational certificates. This would improve overall efficiency in educational institutions by simplifying administrative procedures.

The Key Features of the Project are :

1. Automating Data Extraction: Implementing OCR techniques to automatically extract relevant information from educational certificates, eliminating the need for manual data entry.
2. Improving Efficiency and Accuracy: By automating data extraction, our project aims to significantly reduce processing time while ensuring high accuracy in information retrieval.
3. Enhancing Administrative Processes: Our solution is designed to streamline administrative workflows within educational institutions, allowing staff to focus on more strategic tasks rather than tedious data entry.
4. Focusing on OCR-Based Solution: With a specific focus on OCR technology, we are developing a robust solution tailored to the unique challenges of certificate data extraction.

5. Addressing Real-Time Needs: Our project directly addresses the real-time need for efficient and accurate certificate processing, aligning with the digital transformation initiatives prevalent in today's educational landscape.
6. Enabling Seamless Integration: Our solution will seamlessly integrate with existing administrative systems, ensuring smooth adoption and minimal disruption to institutional workflows.
7. Paving the Way for Future Innovations: By pioneering the use of OCR in educational certificate processing, we aim to set a precedent for future innovations in administrative automation within educational institutions.
8. Contributing to Educational Excellence: Ultimately, our project strives to contribute to educational excellence by empowering institutions with advanced technological solutions that optimize administrative processes and enhance overall efficiency.

1.1 Template Matching

Template matching is a foundational technique in the field of computer vision, essential for identifying specific patterns or objects within images. At the core of template matching lies the concept of comparing a template image with a larger target image to locate instances of the template. However, achieving accurate matches hinges on determining the optimal scaling value, which dictates the resizing of the template image for comparison with the target image.

The choice of scaling value is crucial as it directly impacts the quality of matches obtained during template matching. A scaling value that is too large may lead to oversampling or blurring of the template image, resulting in inaccurate matches. Conversely, a scaling value that is too small may cause under sampling, leading to missed matches or false negatives. Thus, finding the right balance in scaling is paramount for achieving reliable and precise template matching results.

In our study, we explore the significance of scaling value optimization across various template matching techniques. These techniques include:

1. cv2.TM_CCOEFF: This method computes the correlation coefficient between the template and target images.
2. cv2.TM_CCOEFF_NORMED: Similar to TM_CCOEFF, but normalizes the correlation coefficient to a range between -1 and 1.
3. cv2.TM_CCORR: Determines how well the target and template images cross-correlate.
4. cv2.TM_CCORR_NORMED: Normalizes the cross-correlation coefficient to a range between 0 and 1.
5. cv2.TM_SQDIFF: Measures the squared differences between the template and target images.
6. cv2.TM_SQDIFF_NORMED: Normalizes the squared differences coefficient to a range between 0 and 1.

Each of these techniques offers unique advantages and may be suitable for specific applications depending on factors such as image characteristics and matching requirements. By examining the characteristics and performance of these techniques in conjunction with different scaling values, we aim to identify the most effective approach for template matching in our project.

1.2 CNN Architecture based Certificate Classification Model

In the realm of computer vision and machine learning, Convolutional Neural Networks (CNNs) have emerged as powerful tools for image classification tasks. Leveraging the hierarchical structure of neural networks, CNNs excel at automatically learning features from raw input data, making them well-suited for tasks such as image recognition and classification.

Our project incorporates alternative approach i.e.,a CNN-based classification model specifically tailored for the classification of certificate templates. By harnessing the capabilities of TensorFlow, a popular deep learning framework, we constructed a CNN architecture designed to effectively classify certificate templates based on their visual characteristics.

The architecture of our CNN model includes several key components:

1. Convolutional Layers: These layers consist of convolutional filters that scan the input image to extract relevant features. In our model, we employed multiple convolutional layers with varying filter sizes to capture both low-level and high-level features of certificate templates.

2. Pooling Layers: Pooling layers help reduce the spatial dimensions of the feature maps generated by the convolutional layers, thereby reducing computational complexity and preventing overfitting. We utilized MaxPooling2D layers to downsample the feature maps obtained from convolutional layers.

3. Flattening Layer: This layer reshapes the output of the convolutional layers into a one-dimensional vector, preparing it for input into the fully connected layers.

4. Fully Connected Layers: These layers, also known as dense layers, perform classification based on the features extracted by the convolutional layers. In our model, we incorporated dense layers with rectified linear unit (ReLU) activation functions to introduce non-linearity and increase the model's capacity to learn complex patterns.

5. Output Layer: The final layer of our CNN model consists of a single neuron with a sigmoid activation function, which outputs a probability score indicating the likelihood of the input image belonging to a particular class (i.e., a specific certificate template).

Additionally, we implemented mechanisms for training and evaluating the performance of our CNN model using labeled datasets of certificate templates. After training, we saved the trained model to a .h5 file, enabling easy deployment and integration into our certificate processing pipeline.

By leveraging CNNs for certificate classification, our project aims to automate and streamline the identification of certificate templates, contributing to enhanced efficiency and accuracy in certificate processing workflows.

1.3 Preprocessing

Preprocessing plays a crucial role in preparing image data for subsequent analysis and feature extraction. In the context of certificate processing, preprocessing methods are employed to enhance the quality of input images, improve the effectiveness of subsequent algorithms, and facilitate accurate information extraction. This section explores several preprocessing techniques commonly utilized in certificate processing pipelines

1.3.1 Gray Scaling

Gray scaling is a fundamental preprocessing step that involves converting color images into grayscale representations. By removing color information and representing images using only intensity values, gray scaling simplifies image processing tasks while preserving important structural details. This technique is particularly useful for certificate images, as it reduces computational complexity and ensures consistency across diverse image sources.

1.3.2Thresholding

Global Thresholding

Global thresholding is a simple yet effective method for image segmentation, where a single threshold value is applied to all pixels in the image to separate foreground objects from the background. By binarizing the image based on intensity values above or below the threshold, global thresholding facilitates the extraction of regions of interest, such as text and graphics, from the background. This technique is widely used in certificate processing to isolate textual content and enhance readability.

Otsu's Thresholding

Otsu's thresholding, also known as maximum variance thresholding, is an adaptive thresholding technique that automatically computes an optimal threshold value based on the intensity distribution of the input image. Unlike global thresholding, Otsu's method dynamically adjusts the threshold to maximize the separation between foreground and background pixels, resulting in improved segmentation accuracy, especially in cases where the intensity distribution is bimodal or multimodal. In certificate processing, Otsu's

thresholding is employed to handle variations in image brightness and contrast, ensuring robust segmentation of textual and graphical elements.

Otsu's Thresholding with Gaussian Filter

Otsu's thresholding with Gaussian filter combines the benefits of Otsu's thresholding with the smoothing effects of Gaussian filtering. Gaussian filtering is a spatial domain filtering technique used to reduce image noise and enhance edge preservation. By applying Gaussian filtering before Otsu's thresholding, image noise is suppressed, and the intensity distribution becomes smoother, leading to more stable thresholding results. This approach is particularly advantageous in certificate processing scenarios where images may exhibit noise or uneven illumination, thereby improving the accuracy of segmentation and subsequent information extraction tasks.

Overall, by leveraging these preprocessing methods, certificate processing pipelines can achieve enhanced robustness, accuracy, and efficiency in extracting relevant information from certificate images. These techniques lay the foundation for subsequent stages of processing, such as text extraction and Refinement contributing to the overall success of automated certificate processing systems.

1.4 OCR and PYTESSERACT

Optical Character Recognition (OCR) is a transformative technology that enables the extraction of text from images or scanned documents, converting them into machine-readable formats. Pytesseract, a Python library, provides a convenient interface to Tesseract-OCR, an open-source OCR engine developed by Google. This section delves into the capabilities of OCR and Pytesseract in the context of certificate processing:

Optical Character Recognition (OCR):

Optical Character Recognition (OCR) is a powerful technology that automates the extraction of text from images, scanned documents, or printed materials. By analyzing the visual patterns of characters, OCR systems can recognize and interpret text with high accuracy, facilitating data extraction and analysis in various applications. In the realm of certificate processing, OCR plays a crucial role in digitizing certificate images, extracting textual information such as participant names, dates, and institutions, and enabling further processing and analysis.

Pytesseract:

Pytesseract is a Python wrapper for Tesseract-OCR, providing easy access to the Tesseract library's functionality within Python applications. Tesseract, an open-source OCR engine developed by Google, is renowned for its accuracy and versatility in recognizing text from images. Pytesseract simplifies the integration of OCR capabilities into Python workflows, allowing developers to leverage Tesseract's robust text recognition algorithms without the need for complex low-level coding. By interfacing with Pytesseract, developers can perform a wide range of OCR tasks, including image preprocessing, text extraction, and result analysis, with minimal effort and maximum efficiency.

Together, OCR and Pytesseract empower developers and researchers to automate text extraction tasks, streamline document processing workflows, and unlock valuable insights from certificate images. By harnessing the capabilities of these technologies, certificate processing systems can achieve enhanced accuracy, scalability, and productivity, driving innovation and efficiency in educational institutions and beyond.

2. LITERATURE SURVEY

Researchers from various groups have conducted a number of experiments over the years.

Amitha Set al.[3] developed a web application that uses image processing and OCR methods to transform image-based attendance registers into editable Excel files. Simplifying data administration in educational contexts is one of the main goals. The application's performance will be assessed in terms of handling different register image types, accuracy, processing speed, and user-friendliness. Unconventional formats and handwritten text may provide difficulties. The web app's effective development is one of the main conclusions. Future developments could include creating a mobile application, identifying handwritten text in cursive, and enhancing image processing and OCR algorithms to increase accuracy.

Suraya Mubeen et al.[9] developed an OCR Android app that will extract text from scanned documents seamlessly by utilizing Tesseract OCR and LSTM-based line recognition. The software meets the need for precise character identification, attaining an accuracy range of 85% to 98%. Accuracy is nevertheless affected by issues with grainy or dimly illuminated photos. Support for multiple languages and the ability to recognize mathematical equations are important aspects. Subsequent research endeavors center on augmenting precision and resilience via enhanced image processing and OCR techniques.

Chirag Indravadanbhai Patel [10] compared Tesseract and Transym OCR tools in extracting text from vehicle number plates, focusing on accuracy, processing time, and consistency. Tesseract outperforms Transym with higher accuracy (61% for color, 70% for grayscale), faster processing, and greater consistency. Future research could explore broader image types, factors affecting OCR performance, and suggest using both tools based on image nature and accuracy needs.

Daniel Akinbade,et al.[2]created a technique that uses Tesseract OCR and an adaptive thresholding algorithm to reliably extract text from photos with complicated backgrounds. It assesses performance in terms of character and word accuracy, demonstrating its efficacy in comparison to other approaches. 69.7% accuracy at the word level and 81.9% accuracy at the character level are important results. Subsequent

investigations may concentrate on enhancing the adaptive thresholding algorithm for even greater accuracy and investigate the extraction of text from backgrounds with high contrast.

Cosmin Irimia et al.[6] research is to create an application that uses text extraction from photos of personal papers to streamline bureaucratic processes. It achieves 100% success rate on first testing documents and assesses performance based on reaction time, correctness, and user satisfaction. Subsequent investigations may concentrate on refining the system's performance even more, adding more document types to its support, and improving security and UI usability.

Nitish Srivastava et al.[13] proposed multimodal data by pretraining pathways for each modality independently and then integrating them during joint training. The model seamlessly fuses multiple data modalities into a unified representation, demonstrating superior performance over Deep Belief Networks (DBN) and Autoencoder models, with a Mean Average Precision (MAP) of 0.622. Notably, the DBM excels in handling both multimodal and unimodal queries, effectively inferring missing modalities. However, limitations include the need for further exploration of scalability and computational efficiency, as well as challenges in handling extremely diverse or noisy data. Overall, the research highlights the DBM's effectiveness in learning joint representations from diverse data modalities and its superiority over alternative deep learning approaches, with future research avenues focusing on extending the model to accommodate additional modalities and exploring applications beyond classification and retrieval.

Thammarak et al. [15] The effectiveness of Google Cloud Vision API and Tesseract OCR, augmented by pre-processing techniques for image enhancement, in recognizing Thai vehicle registration certificates is examined in the research paper "Comparative Analysis of Google Cloud Vision API and Tesseract for Recognizing Thai Vehicle Registration Certificates," which was published in 2021. Comparing the effectiveness and precision of these two approaches for obtaining vehicle information from the certificates is the main objective of the study. The results show that Tesseract performs worse than Google Cloud Vision API, with an accuracy rate of 47.02%, while Tesseract achieves an accuracy rating of 84.43%. The comparative advantages and disadvantages of these OCR methods in the context of identifying Thai automobile registration certificates are clarified by this investigation.

Vaithiyanathan et al. [16] explores the utilization of Google Cloud OCR for extracting text in cloud-based environments, particularly for applications aimed at assisting visually impaired individuals. The methodology involves a series of image processing techniques, such as adjusting brightness and applying image enhancement, followed by the deployment of Google Cloud OCR for text extraction purposes. The study reports an impressive accuracy range of 75-100%, demonstrating the effectiveness of this cloud-based approach in facilitating accessibility for visually impaired users through accurate text extraction from images.

Agbemenu et al. [1] presented a system leveraging OpenCV and the Tesseract OCR Engine for automatic number plate recognition (ANPR). The methodology incorporates edge detection algorithms and template matching techniques to achieve plate detection and recognition. The study reports that the edge detection algorithm attained a plate detection accuracy of 79.4%, while template matching exhibited higher accuracy at 90.8%. However, challenges were encountered in OCR recognition, with only approximately 60% of detected plates recognized due to issues such as faded characters, plate decorations, and noise affecting segmentation and character recognition. Despite these challenges, the system demonstrated reasonable plate detection accuracy up to a distance of 5 meters, indicating its potential for ANPR applications.

Vijayarani et al. [17] introduced a methodology centered around template matching techniques, including the Performance Index Method, Cross Correlation, and Normalized Cross Correlation, with a focus on utilizing the Sum of Absolute Differences (SAD). This approach is primarily aimed at searching for single and multiple characters within words, sentences, and paragraphs within documents. The study emphasizes the versatility of the proposed method in document analysis. It highlights that a lower SAD value indicates better template matching performance, thus illustrating the effectiveness of the technique in accurately identifying and locating specific words or characters within documents.

Sahu et al. [12] provides an overview of OCR and MCR (Machine Character Recognition) techniques along with various stages involved in the OCR process, including pre-processing, characteristic extraction, classification, and post-processing. The paper highlights the widespread application of OCR and MCR

techniques in pattern recognition tasks. It discusses the challenges faced in OCR, particularly in recognizing handwritten text variability and unimpeded cursive script. Pre-processing steps such as noise elimination, skew recognition/improvement, and binarization are identified as essential for enhancing OCR accuracy. Additionally, the paper emphasizes the importance of high-resolution images for improved OCR performance and highlights that certain image formats, such as JPEG for bi-tonal images, may not be suitable for OCR tasks. Overall, the paper serves as a comprehensive resource for understanding OCR techniques and the challenges associated with them.

Zheng et al. [19] introduced an innovative algorithm integrating global edge features, local Haar-like features, an improved blob detection algorithm, and the Tesseract OCR. It employed a cascaded classifier with six layers, AdaBoost learning algorithm, statistical features, Haar-like features, and OCR evaluation, resulting in a robust license plate extraction and recognition system for traffic video datasets. Demonstrating real-time efficiency, especially under poor lighting and for moving vehicles, it achieved high accuracy in character segmentation and recognition, with an improved recognition accuracy of 94.03% after non-character area removal. This advancement signifies a significant stride in license plate recognition, enhancing accuracy and applicability in practical scenarios.

Manwatkar et al.[8] provided insights into various methods for text recognition from images, addressing challenges related to font characteristics and image quality. It emphasizes the necessity of character recognition mechanisms and reviews approaches such as adaptive binarization, fuzzy logic-based recognition, and language-specific methods. Challenges in recognizing characters due to font variations in paper documents and the quality of scanned images are acknowledged, with the paper highlighting the difficulty in reading individual contents and searching documents line-by-line and word-by-word.

Lima et al. [7] explored the process of extracting and analyzing data from unstructured sources. It outlines the stages of web scraping, plain text conversion, data wrangling, and data loading, culminating in business intelligence applications. Key evaluation metrics include training and prediction time, confusion matrix-based evaluation, AUC values for all classifiers utilized in the study. The successful implementation of the

data pipeline involved employing tools such as Scrapy, Selenium, LibreOffice, pdftotext, and OpenRefine. Through exploration and business intelligence/data mining techniques specifically applied to javelin throw data, the study identified key features and models for analysis, although detailed scores were not provided.

Ramakrishnan et al. [11] focused on the process of detecting contiguous text blocks, classifying these blocks using DROOLS rules, and stitching them together for XML output. Block detection is evaluated using the Spatial Segmentation Score metric, while classification performance is assessed using metrics such as precision, recall, and F1 score. Text extraction quality is measured using the Alignment Cost (Normalized) metric. The findings indicate that block detection is generally accurate but faces occasional challenges, while the classification approach is rule-based and adaptable, albeit with some specific challenges. Quantitative metrics include a mean Spatial Segmentation Score of 9.5 with a standard deviation of 5.7 for block detection, and precision (P) of 0.532 and recall (R) of 0.632 for classification.

Duretec et al .[5] presented "A Text Extraction Software Benchmark Based on a Synthesized Dataset," which involves model-driven dataset synthesis, benchmark creation, and tool performance evaluation. Key metrics used for evaluation include precision, recall, and F1 score. The benchmark enables the assessment of text extraction tools, demonstrating benefits in scalability and ground truth generation. While detailed scores are not provided, the study underscores the importance of standardized benchmarking methodologies for evaluating text extraction software, facilitating comparisons and advancements in the field.

Wiechork et al. [18] employed methods such as Scrapy, Aletheia, Excalibur, Tabula, CyberPDF, PDFMiner, and ExamClipper. Time measurements indicate Excalibur at 20 seconds, Tabula at 16 seconds, PDFMiner at 16 seconds, CyberPDF at 22 seconds, Aletheia at 180 seconds, and ExamClipper at 240 seconds, with quality percentages showing Excalibur at 99.4% and Tabula at 97.7%. Notably, Excalibur outperforms Tabula in tabular data extraction, despite facing challenges in metrics for multi-column layouts. Aletheia and ExamClipper are compared with some manual adjustments affecting fairness, with Excalibur demonstrating superior efficiency and accuracy in automated PDF extraction tasks.

Budhiraja et al. [4] employed supervised learning, classification, regression, text block analysis, HTML conversion, and heading detection techniques. While specific metrics are not provided, the study emphasizes the reliance on known structures and supervised learning for outcome detection. It highlights the importance of layout analysis in the extraction process. Although detailed scores are not included, the study underscores the significance of machine learning algorithms in effectively extracting specific text from documents, offering insights into the challenges and strategies involved in this domain.

Tewani et al.[14] focused on utilizing Pytesseract-based Optical Character Recognition (OCR) combined with global regular expression (Regex) models and Flask API. The study emphasizes achieving high accuracy in extracting key data points from images of Aadhar and Pan Cards. Through FLASK API integration, OCR text recognition, and the utilization of global/regex models, the process of information extraction is automated. The accuracy metrics reported are impressive, with an accuracy rate of 95%, graphical component (GC) accuracy of 91%, and text component (TC) accuracy of 95%. This approach offers a robust solution for efficiently retrieving essential information from these government-issued documents.

3. PROBLEM STATEMENT

There are a lot of time-consuming, inefficient, and error-prone procedures associated with the manual extraction of information from educational credentials. The automation required to automate administrative processes and precisely extract pertinent data fields from certificates is absent from current approaches. A methodical approach to automating the extraction and processing of certificate data is urgently needed since educational institutions manage a wide variety of certifications from different sources. By creating a strong methodology and framework for batch processing of educational certificates and making use of cutting-edge technologies like OCR, data extraction strategies, and serialization techniques, this project seeks to address these issues by effectively extracting, improving, and integrating certificate data into structured formats for reporting and analysis needs.

4. METHODOLOGY

The methodology for building the model to extract data from educational certifications involves a systematic approach consisting of five key steps. Each step plays a crucial role in the overall process, culminating in the generation of an Excel sheet containing the required information from a group of certificates. Here's a detailed description of the methodology:

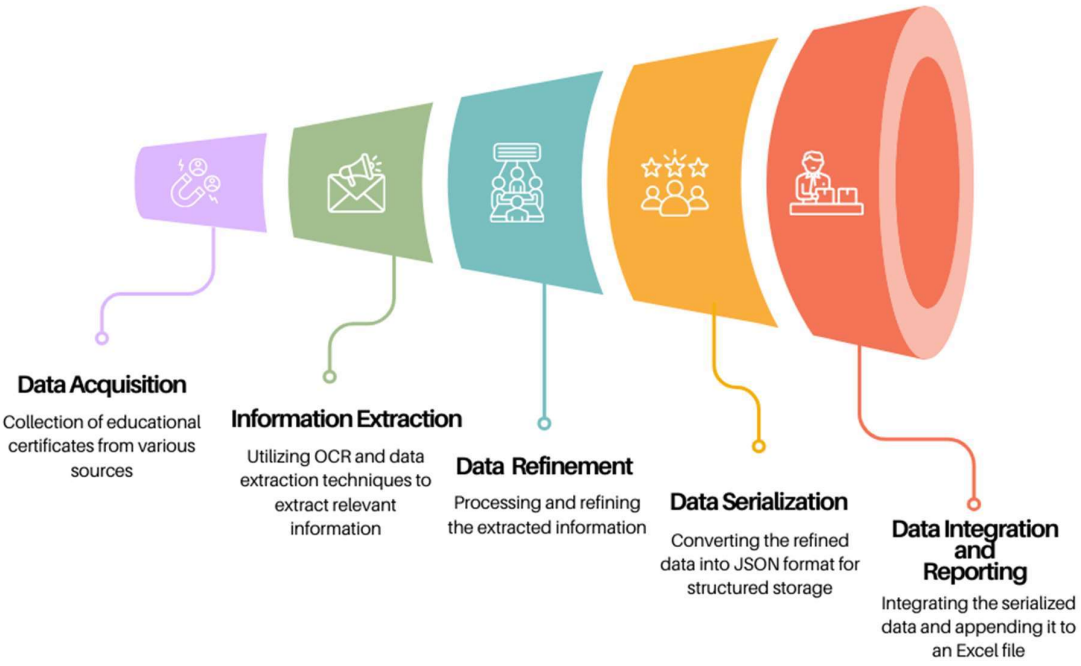

Fig 4.1 Methodology

4.1 Data Acquisition

This initial step involves the collection of educational certificates from various sources. Certificates are gathered from different institutions or platforms and stored in a repository for further processing.

Certificate Selection: Certificates are selected based on their relevance to the classification and comparison tasks. The dataset includes certificates that represent the two templates used for classification and comparison.

Purpose Identification: Each certificate is categorized based on its purpose, whether it serves as a template for approach 1 (template matching) or approach 2 (CNN technique).

Template Matching Approach: For approach 1, certificates are selected to represent the templates used in template matching techniques. These certificates are crucial for training and validating the template matching model.

CNN Technique Approach: Similarly, certificates representing the templates used in the CNN technique (approach 2) are collected. These certificates are essential for training and evaluating the CNN model.

This meticulous data collection process ensures that the dataset encompasses a diverse range of certificates, enabling comprehensive training and evaluation of both template matching and CNN techniques.

4.1.1 Template Matching and Optimal Scaling:

Template matching is crucial in computer vision for identifying objects or patterns in images. Determining the optimal scaling value is essential for accurate matching, indicating how much the template image should be resized before comparison. Our study involved experimentation with various scaling values across different template matching techniques. The impact on accuracy varied based on the technique and image features, with smaller values preferred for correlation-based methods and larger values for feature-based methods.

4.1.2 CNN approach

For the CNN approach (approach 2), certificates representing the templates used in the classification task are specifically collected. These certificates serve as the training and evaluation data for the Convolutional Neural Network (CNN) model. During training, the CNN model learns features and patterns indicative of each template, enabling it to accurately classify unseen certificates. Once trained, the CNN model is evaluated using a separate set of certificates reserved for testing to assess its performance in accurately classifying unseen certificates.

4.2 Information Extraction

In addition to traditional OCR techniques, our information extraction process leverages advanced capabilities provided by the Pytesseract library, which acts as a Python wrapper for Google's Tesseract-OCR Engine. By integrating Pytesseract into our workflow, we gain access to powerful features and functionality that enhance the accuracy and efficiency of text extraction from

certificate images. Pytesseract supports various configuration options, including the specification of OCR Engine Modes (OEM) and Page Segmentation Modes (PSM), which enable fine-tuning of the OCR process to better suit the characteristics of the input data. These parameters allow us to optimize the recognition process for different types of certificate layouts, fonts, and languages, thereby improving overall extraction performance. Additionally, Pytesseract's seamless integration with Python facilitates easy implementation and customization of OCR-based models, empowering us to develop robust solutions for information retrieval, text mining, and document analysis tasks. By harnessing the flexibility and versatility of Pytesseract along with advanced OCR techniques such as OEM and PSM customization, we ensure accurate and reliable extraction of relevant information from educational certificates, contributing to the efficiency and effectiveness of our data extraction pipeline.

4.3 Data Refinement

After the initial extraction of information, the data undergoes a meticulous refinement process aimed at enhancing accuracy and consistency. This refinement stage involves the utilization of regular expressions (regex) to meticulously validate and clean the extracted data. By applying regex patterns tailored to specific data fields such as names, dates, and institutions, we can identify and rectify any errors or inconsistencies that may have arisen during the extraction process. Additionally, the extracted data is standardized to adhere to predefined criteria, ensuring uniformity and coherence across all entries. Through the judicious application of regex-based validation and standardization techniques, we guarantee the reliability and integrity of the extracted data, paving the way for further analysis and processing.

4.4 Data Serialization

The refined data is then serialized into JSON (JavaScript Object Notation) format for structured storage. JSON provides a lightweight and flexible way to represent data, making it suitable for transferring and storing extracted information in a structured manner.

4.5 Data Integration and Reporting

In the final step, the serialized data is integrated and appended to an Excel file. The Excel file serves as the output format for the extracted information, with each row representing a certificate and its corresponding data fields. Additionally, the integrated data may be further processed or analysed for reporting purposes, providing insights into the collected information.

This methodology is designed to be scalable and efficient, capable of processing a group of certificates in a batch processing manner. Through the iterative execution of these steps, the model can effectively extract and organize information from educational certificates, facilitating streamlined administrative workflows and enhancing operational efficiency.

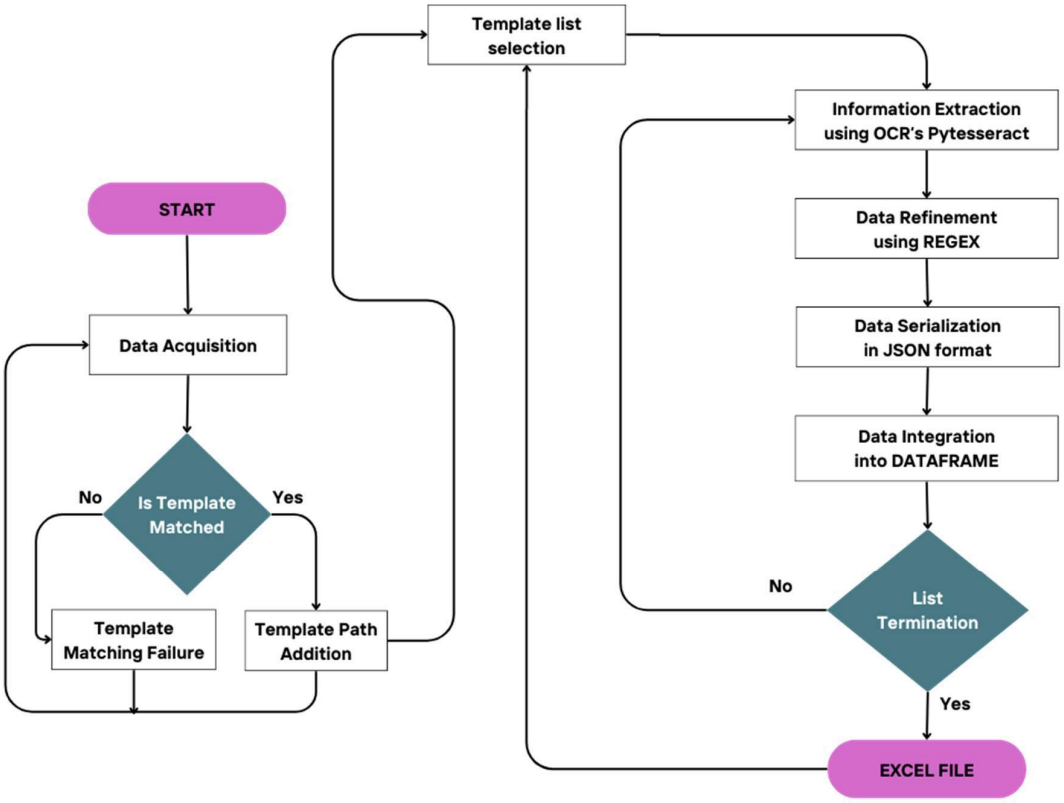

Fig 4.2 Architectural Diagram

5. IMPLEMENTATION

In the project's implementation phase, we have devised a thorough strategy to effectively extract essential information from a collection of certificates using template matching techniques. Following extensive experimentation, we determined that the TM_CCOEFF method provided the most favourable outcomes for our specific requirements. Our methodology involves initially identifying templates within the certificates and obtaining associated scaling values. Subsequently, employing the TM_CCOEFF method, we compile lists containing the addresses of matching templates. Each list then undergoes distinct coding procedures tailored for data extraction. Employing natural language processing exclusively, we selectively filter and extract only pertinent data from each certificate. The extracted information is subsequently appended to distinct Excel files, with the number of files corresponding to the number of different templates obtained during the template matching phase. This process guarantees the precise extraction and organization of requisite information for further analysis.

1. DATA ACQUISITION

- We initiated the project by gathering certificate images from various sources, including scanned paper documents, PDF files, and digital camera images. This diverse dataset served as the foundation for our subsequent analysis and model training.
- As part of the data acquisition process, we handled various input formats, including PDF files. To facilitate data extraction, PDF files were converted into the JPEG format, ensuring compatibility with our image processing pipeline.
- To identify crucial areas within the certificate images, we employed template matching techniques using the cv2.TM_CCOEFF method. This strategic approach allowed us to effectively pinpoint significant sections within the images, streamlining the process of precise data extraction.
- By utilizing template matching, we were able to accurately identify and isolate the regions of interest within the certificate images. This optimized the efficiency of our data extraction procedures, as we could focus exclusively on the relevant sections, ensuring that our extracted data was both accurate and comprehensive.

- Additionally, we meticulously compiled and organized the data by utilizing lists containing addresses of similar template matches obtained through the template matching process. This systematic approach enabled us to manage and process large volumes of certificate images efficiently.

By incorporating these steps into our data acquisition process, we ensured that our model had access to a diverse and comprehensive dataset, laying the groundwork for accurate and reliable data extraction from educational certificates.

Fig 5.1 Template 1

Fig 5.2 Template 2

2. INFORMATION EXTRACTION

Grayscale Conversion

Grayscale conversion is a fundamental preprocessing step in image processing, where color images are converted into grayscale images by removing color information and retaining only intensity values.

In our project, grayscale conversion was employed as a preprocessing technique to simplify image processing and reduce computational complexity. By converting certificate images to grayscale, we focused solely on intensity variations, making subsequent processing steps more efficient.

Grayscale images are easier to analyze and manipulate compared to color images, making them well-suited for OCR and template matching tasks. The absence of color information simplifies feature extraction and enhances the robustness of algorithms used in subsequent processing stages.

Fig 5.3 Gray scale Image

Fig 5.4 Original Image

Thresholding:

Thresholding is a technique used to segment images by dividing pixels into foreground and background based on a specified threshold value.

In our project, we employed global thresholding techniques such as binary thresholding to convert grayscale images into binary images, where pixels are classified as either black or white based on a predefined threshold value.

Additionally, Otsu's thresholding method, which automatically calculates the optimal threshold value, was utilized to adaptively threshold certificate images. This technique is particularly useful in scenarios where the optimal threshold value may vary across different images or lighting conditions.

Furthermore, Otsu's thresholding with Gaussian filtering was applied to smoothen images and reduce noise before thresholding, resulting in improved segmentation accuracy, especially in images with uneven illumination or high levels of noise.

- Tesseract OCR (Optical Character Recognition) was selected as the preferred tool for extracting text from the identified regions within the certificate images. By using Tesseract, we were able to automate the process of extracting textual information from the certificate images, saving time and effort compared to manual transcription methods. Additionally, Tesseract's robust performance ensured that even complex fonts and layouts within the certificates could be accurately processed, resulting in reliable data extraction.
- Tesseract OCR was configured with custom settings to optimize its performance for our specific use case. Customization involved fine-tuning

parameters such as language settings, image preprocessing techniques, and text recognition strategies to suit the characteristics of the certificate images and improve overall accuracy.

- We used oem-3 and psm-6 model for the accurate results.

OEM 3 (OCR Engine Mode 3):

- OEM 3, or OCR Engine Mode 3, is a setting in Tesseract OCR that utilizes neural nets for improved accuracy. By leveraging neural network models, OEM 3 enhances the OCR engine's ability to accurately recognize and extract text from images, particularly in complex or challenging scenarios.
- In our project, the utilization of OEM 3 significantly enhanced the accuracy of text extraction from educational certificates. By employing advanced neural network algorithms, OEM 3 enabled our OCR system to better handle diverse fonts, layouts, and image conditions, ultimately resulting in more reliable data extraction.

PSM 6 (Page Segmentation Mode 6):

- PSM 6, or Page Segmentation Mode 6, is a setting in Tesseract OCR that specifies automatic page segmentation with OSD (Orientation and Script Detection). This mode is particularly useful for documents with multiple blocks of text or complex layouts.
- In our project, PSM 6 played a crucial role in accurately segmenting and extracting text from educational certificates with varied layouts. By automatically detecting the orientation and script of the text, PSM 6 ensured that our OCR system could effectively handle certificates with diverse structures and formats, improving the overall accuracy of data extraction.

By leveraging OEM 3 and PSM 6 settings in our OCR pipeline, we were able to enhance the accuracy and reliability of text extraction from educational certificates, thereby improving the effectiveness of our data processing workflow.

Table 5.1 OEM Description

OEM	Description
0	Legacy Tesseract OCR Engine. Older technology.
1	Legacy + LSTM Tesseract OCR Engine. Combines legacy engine with LSTM for improved accuracy.
2	Default Tesseract OCR Engine. Automatically selects the best OCR engine based on the input image.
3	LSTM Tesseract OCR Engine. Uses LSTM neural networks for better accuracy, especially for complex text and languages.

Table 5.2 PSM Description

PSM	Description
0	Detects the orientation and script present in the image.
1	Automatically segments the page while detecting its orientation and script.
2	Automatically segments the page without detecting its orientation and script.
3	Fully automates page segmentation without detecting orientation and script (Default).
4	Assumes the presence of a single column of text with varying sizes.
5	Assumes the presence of a single block of text aligned vertically.
6	Assumes the presence of a single block of text with uniform layout.
7	Treats the image as a single line of text.
8	Treats the image as a single word.
9	Treats the image as a single word arranged in a circular pattern.
10	Treats the image as a single character.

3. DATA REFINEMENT

- We implemented regex patterns to extract specific data components such as names, scores, roll numbers, institutes, and other pertinent information from the refined text. Through the creation of customized regex patterns tailored to each distinct data category, we guaranteed the precise extraction of relevant and accurate information essential for subsequent analysis and processing.
- We developed customized regex patterns tailored to the diverse nature of the data elements present in the certificate text. These patterns were carefully designed to accommodate variations in formatting, structure, and content, ensuring robust extraction of information across different types of certificates.

Certainly! Here's the description of the regex patterns used for template 1 and template 2:

Template 1:

1. Name Pattern: This regex pattern, `name_pattern`, is designed to extract the recipient's name from the certificate text. It captures the text between the phrase "awarded to" and "for", ensuring accurate extraction of the recipient's name.

2. Score Pattern: The `score_pattern` regex extracts the consolidated score from the certificate text. It identifies and captures the numerical score mentioned in the certificate.

3. Roll Number Pattern: This pattern, `roll_pattern`, extracts the roll number of the recipient from the certificate text. It captures alphanumeric characters following the phrase "Roll No:".

4. Institute Pattern: The `institute_pattern` regex identifies and captures the name of the institute mentioned in the certificate text. It extracts the institute name preceded by "IIT".

5. Certification Pattern: The `certification_pattern` regex identifies the certification type, which is "NPTEL" in this case. It ensures the accurate extraction of the certification name.

6. Course Pattern: This pattern, `course_pattern`, extracts the name of the course completed by the recipient. It captures the text between the phrases "completing the course" and "with".

7. Date Pattern: The `date_pattern` regex extracts the completion date of the course from the certificate text. It captures dates in the format "Month-Year".

Template 2:

1. College Name Pattern: The `college_name_pattern` regex extracts the name of the college or institution from the certificate text. It captures uppercase letters and spaces followed by "(A)".

2. Participant Name Pattern: This pattern, `participant_name_pattern`, captures the name of the participant mentioned in the certificate text. It identifies and extracts uppercase letters and spaces following the phrase "presented to" and ending with "for".

3. Event Pattern: The `event_pattern` regex extracts the name of the event or competition from the certificate text. It captures uppercase letters and numbers following the phrase "participating in".

4. Date Pattern: This pattern, `date_pattern`, extracts the event dates mentioned in the certificate text. It captures dates in the format "Dayth Month Year".

4. DATA SERIALIZATION

After the data refinement process, the extracted and cleaned data undergoes serialization into the JSON (JavaScript Object Notation) format for structured storage. JSON is a lightweight and widely-used data interchange format that provides a flexible and human-readable way to represent structured data. Serialization refers to the process of converting complex data structures into a format that can be easily stored, transmitted, and reconstructed when needed.

In our project, serialization into JSON format serves several purposes. Firstly, JSON's simplicity and readability make it an ideal choice for storing structured data, such as the extracted information from educational certificates. Each data field, including names, scores, roll numbers, institutions, course details, and

completion dates, is organized into key-value pairs within the JSON structure, ensuring easy access and manipulation.

5. DATA INTEGRATION AND REPORTING

In the data integration and reporting phase, we utilized the pandas library to convert the serialized JSON structures into data frames, enabling efficient handling and analysis of the extracted information. By leveraging the power of pandas, we structured the data into tabular formats, making it easier to manipulate, filter, and analyze the extracted data.

Subsequently, Excel sheets were generated from these data frames to provide a user-friendly format for visualization and analysis. Each Excel sheet corresponds to a specific template or category of certificates, allowing for organized storage and retrieval of information. The Excel sheets serve as comprehensive repositories of the extracted data, facilitating easy interpretation and derivation of insights.

6. RESULT ANALYSIS

In the results and analysis phase, we identified the optimal scaling factor of 2 for template matching across various methods. This determination was crucial in ensuring accurate matching between the given certificates and templates. By carefully analyzing the results obtained from different template matching techniques, we confirmed that a scaling factor of 2 consistently yielded the best matching accuracy.

```
score: 583379008.0
Scaling value: 2.0, Method: 4
Method: TM_CCOEFF


score: 0.8484170436859131
Scaling value: 2.0, Method: 5
Method: TM_CCOEFF_NORMED


score: 9387196416.0
Scaling value: 2.0, Method: 2
Method: TM_CCORR


score: 0.9877640604972839
Scaling value: 2.0, Method: 3
Method: TM_CCORR_NORMED


score: 3377225216.0
Scaling value: 2.0, Method: 0
Method: TM_SQDIFF


score: 0.5735337734222412
Scaling value: 2.0, Method: 1
Method: TM_SQDIFF_NORMED


Optimal scaling value: 2.0
```

Fig 6.1 Optimal Scaling Value for Template matching

Data Extraction from the NPTEL and the ACE certificates :

Table 6.1 Template1 Output

	A	B	C	D	E	F	G
1	Name	Score	Roll Number	Institute	Name of the certification	Course completed	Completion date
2	KADALI VIDHYA	73	NPTEL18CE17S21170060	IIT Kharagpur	NPTEL	Strength of Materials	Jul-Oct 2018
3	SUNEEL MERUGU	56	NPTEL18CE20S822140904	IIT Madras	NPTEL	Concrete Technology	Jul-Oct 2018
4	VINAY GUDDATI	40	NPTEL18CE22S11170327	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
5	VENKATA KRISHNA ATMAKURI	40	NPTEL18CE22S11170328	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
6	GANDHAM LEELA PREMCHAND	50	NPTEL18CE22S11170334	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
7	BURA SRINIVAS SURYA GOUTAM	46	NPTEL18CE22S11170335	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
8	GORLA AMRUTHA SAI MANIDEEP	53	NPTEL18CE22S11170336	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
9	SKSIMMU	40	NPTEL18CE22S11170339	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
10	ALLU DEEPANVITHA MADHURI	40	NPTEL18CE22S11170340	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
11	MALLISETTI PRASANTH	40	NPTEL18CE22S11170341	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
12	DWARAMPUDI BULLI VENKATA REDDY	40	NPTEL18CE22S11170342	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
13	NAVEEN CHANDRA	40	NPTEL18CE22S11170343	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
14	REDDY CHANDRIKA	42	NPTEL18CE22S11170344	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
15	AAKI N V GOPALA KRISHNA SIVA RAMA SAIKIRAN	40	NPTEL18CE22S11170346	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
16	GUBBALA YASHWANTHI SRILEKHA	42	NPTEL18CE22S11170347	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
17	BOMMITHI TEJASWI	40	NPTEL18CE22S11170348	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
18	THOTA NAGARAJU	40	NPTEL18CE22S11170349	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
19	SARIPALLI BALARAM RAJU	43	NPTEL18CE22S11170350	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
20	DEGAPATI PRAKASH RAJU	40	NPTEL18CE22S11170351	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
21	BOMMA CHARISHMA	40	NPTEL18CE22S12020104	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
22	M SA RAMPRASAD	43	NPTEL18CE22S12161413	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
23	KOYA SAI TEJA	40	NPTEL18CE22S21170168	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
24	PETETI LAKSHMIPATHINAIDU	40	NPTEL18CE22S21170169	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
25	PENUMALLU PHANI SRINIVAS REDDY	40	NPTEL18CE22S821170172	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018
26	KOLLA MANO PAVAN KUMAR	40	NPTEL18CE22S821170173	IIT Madras	NPTEL	Modern Construction Materials	Jul-Oct 2018

Table 6.2 Template2 Output

	A	B	C	D	E
1	College Name	Participant Name	Event Participated	Event Date (Start)	Event Date (End)
2	SRKR ENGINEERING COLLEGE	MURALA CHINNI SPANDANA	PRAJWALAN2K24	16th February 2024	17th February 2024
3	SRKR ENGINEERING COLLEGE	JAMPANA SATYA RISHITHA	PRAJWALAN2K24	16th February 2024	17th February 2024
4	SRKR ENGINEERING COLLEGE	ASRITHA SUTHAPALLI	PRAJWALAN2K24	16th February 2024	17th February 2024
5	SRKR ENGINEERING COLLEGE	PERUMALLA AISHWARYA	PRAJWALAN2K24	16th February 2024	17th February 2024
6	SRKR ENGINEERING COLLEGE	SARAYU SIMHADRI	PRAJWALAN2K24	16th February 2024	17th February 2024
7	SRKR ENGINEERING COLLEGE	SOWMITH NARAYANA PALURI	PRAJWALAN2K24	16th February 2024	17th February 2024
8	SRKR ENGINEERING COLLEGE	PALURI HITESH	PRAJWALAN2K24	16th February 2024	17th February 2024
9	SRKR ENGINEERING COLLEGE	RAJA CHANDRA APPANA	PRAJWALAN2K24	16th February 2024	17th February 2024
10	SRKR ENGINEERING COLLEGE	KRISHNA VAMSI KOLLURI	PRAJWALAN2K24	16th February 2024	17th February 2024
11	SRKR ENGINEERING COLLEGE	PALIVELA NIKHILESWAR	PRAJWALAN2K24	16th February 2024	17th February 2024
12	SRKR ENGINEERING COLLEGE	THIRUMALLA SAI NAGA MANIKANTA	PRAJWALAN2K24	16th February 2024	17th February 2024
13	SRKR ENGINEERING COLLEGE	CALVIN DAVIS SR	PRAJWALAN2K24	16th February 2024	17th February 2024
14	SRKR ENGINEERING COLLEGE	EUGINE CARMEL P	PRAJWALAN2K24	16th February 2024	17th February 2024
15	SRKR ENGINEERING COLLEGE		PRAJWALAN2K24	16th February 2024	17th February 2024
16	SRKR ENGINEERING COLLEGE	KATTA ROHITH	PRAJWALAN2K24	16th February 2024	17th February 2024
17	SRKR ENGINEERING COLLEGE	SHANMUKHI BODDU	PRAJWALAN2K24	16th February 2024	17th February 2024
18	SRKR ENGINEERING COLLEGE	KEDARNATH GADAM	PRAJWALAN2K24	16th February 2024	17th February 2024
19	SRKR ENGINEERING COLLEGE	ANNEM JYOTHI	PRAJWALAN2K24	16th February 2024	17th February 2024
20	SRKR ENGINEERING COLLEGE	VENKATESH DEVARAKONDA	(Ctrl) N2K24	16th February 2024	17th February 2024
21	SRKR ENGINEERING COLLEGE	JYOTHIKA CHEEDI	PRAJWALAN2K24	16th February 2024	17th February 2024
22	SRKR ENGINEERING COLLEGE	CHIPPALA HARIKUMAR	PRAJWALAN2K24	16th February 2024	17th February 2024

The model generates output files containing information extracted from certificates, with each row representing details from both templates.

Following text extraction, the data refinement process efficiently extracts the necessary information from the extracted text.

CER (Character Error Rate):

Character Error Rate (CER) serves as a metric for assessing the accuracy of text recognition systems by measuring the percentage of incorrectly recognized characters against the total number of characters in the ground truth text. It is calculated using the formula CER = (S + D + I) / N, where S represents the number of substitutions, D denotes the number of deletions, I indicates the number of insertions, and N represents the total number of characters in the ground truth text.

WER (Word Error Rate):

Word Error Rate (WER) evaluates the accuracy of text recognition systems for tasks involving full sentence or document recognition. WER measures the percentage of incorrectly recognized words compared to the total number of words in the ground truth text, and it is calculated using the formula WER = (S + D + I) / N_(text_words), where N_(text_words) denotes the total number of words in the ground truth text.

Table 6.3 WER and CER on the Template1 data

No	Method	Error Rates	Name	Roll Number	Institute	Name of the certification	Course completed	Completion date
1	Global Thresholding	WER	0.45	0.17	0	0	0	0
		CER	0.34	0.244	0	0	0	0
2	Otsu's Thresholding	WER	0.473333	0.137	0	0	0	0
		CER	0.366667	0.234	0	0	0	0
3	Otsu's Thresholding With Guassian Filter	WER	0.46	0.327	0	0	0	0
		CER	0.353333	0.404	0	0	0	0

Table 6.4 WER and CER on the Template2 data

No	Method	Error Rates	College Name	Participant Name	Event Participated	Event Date (Start)	Event Date (End)
1	Global Thresholding	WER	0	2.5	0	0	0
		CER	0	14.7556	0	0	0
2	Otsu's Thresholding	WER	0	2.43609	0	0	0
		CER	0	14.18797	0	0	0
3	Otsu's Thresholding With Guassian Filter	WER	0	2.43609	0	0	0
		CER	0	14.176692	0	0	0

- In comparing the segmentation performance on Template1 and Template2 certificates, the analysis reveals nuanced differences in the efficacy of various thresholding algorithms.

- For Template1 certificates, Otsu's Thresholding emerges as the preferred method over Global Thresholding, demonstrating consistently lower error rates in both WER and CER metrics.
- Conversely, the evaluation of Template2 certificates showcases a similar trend, with Otsu's Thresholding exhibiting superior performance compared to Global Thresholding.
- However, when incorporating the Gaussian filter, the results for Template2 certificates become mixed, introducing complexities in the segmentation accuracy.
- While Otsu's Thresholding with Gaussian Filter reduces errors in certain aspects, it simultaneously elevates errors in others, posing challenges in determining an optimal segmentation method.
- Overall, while Otsu's Thresholding generally outperforms Global Thresholding across both Template1 and Template2 certificates, the impact of the Gaussian filter remains uncertain, indicating the need for further investigation into the segmentation algorithms' adaptability to different certificate sets.

7. CONCLUSION

In conclusion, the development of our automated data extraction system using a combination of template matching and OCR techniques marks a significant milestone in streamlining information retrieval processes. Through the strategic application of template matching, our system efficiently identifies and extracts key data points from certificates by pinpointing regions of interest. The integration of OCR technology, specifically Tesseract, ensures precise extraction of text from these regions, facilitating the retrieval of vital information such as participant names, completion dates, and certificate titles. This approach not only enhances efficiency but also reduces manual effort, thereby improving overall productivity and accuracy in certificate processing workflows.

Furthermore, by incorporating global thresholding as the final step, we achieved commendable Word Error Rate (WER) and Character Error Rate (CER) rates. This optimization further enhances the accuracy of our system, ensuring reliable extraction of information from certificates across various formats and layouts. With robust performance metrics and a streamlined workflow, our system demonstrates its potential to revolutionize data extraction processes in educational institutions and beyond.

8. FUTURE SCOPE

1. Advanced Template Matching and OCR Methods:

Explore cutting-edge template matching algorithms and deep learning techniques to enhance template detection accuracy and extraction efficiency. Investigate advanced OCR methodologies, including deep learning approaches, to improve text extraction accuracy, particularly for challenging fonts and degraded images.

2. Domain-Specific Knowledge Integration:

Integrate domain-specific rules and knowledge to refine data extraction, enabling recognition of diverse certificate formats and layouts.

3. Continuous System Maintenance:

Implement robust mechanisms for ongoing monitoring, maintenance, and updates to ensure the reliability and relevance of the certificate data extraction system over time.

REFERENCES

[1] Agbemenu, Andrew S., Jepthah Yankey, and Ernest O. Addo. "An automatic number plate recognition system using opencv and tesseract ocr engine." International Journal of Computer Applications 180.43 (2018): 1-5.

[2] Akinbade, Daniel, et al. "An adaptive thresholding algorithm-based optical character recognition system for information extraction in complex images." Journal of Computer Science 16.6 (2020): 784-801.

[3]Amitha, S., et al. "Conversion of Image To Excel Using Ocr Technique." Int. Res. J. Mod. Eng. Technol. Sci 4 (2020): 3743-3747.

[4] Budhiraja, Sahib Singh. Extracting specific text from documents using machine learning algorithms. Diss. 2018.

[5]Duretec, Kresimir, Andreas Rauber, and Christoph Becker. "A text extraction software benchmark based on a synthesized dataset." 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL). IEEE, 2017.

[6] Irimia, Cosmin, et al. "Official Document Identification and Data Extraction using Templates and OCR." Procedia Computer Science 207 (2022): 1571-1580.

[7] Lima, Rui, and Estrela Ferreira Cruz. "Extraction and Multidimensional Analysis of Data from Unstructured Data Sources: A Case Study." ICEIS (1). 2019.

[8] Manwatkar, Pratik Madhukar, and Kavita R. Singh. "A technical review on text recognition from images." 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO). IEEE, 2015.

[9] Mubeen, Suraya, et al. "Optical Character Recognition Using Tesseract." International Journal for Research in Applied Science and Engineering Technology, vol. 10, 2022, pp. 672-675.

[10] Patel, Chirag, Atul Patel, and Dharmendra Patel. "Optical character recognition by open source OCR tool tesseract: A case study." International Journal of Computer Applications 55.10 (2012): 50-56.

[11] Ramakrishnan, Cartic, et al. "Layout-aware text extraction from full-text PDF of scientific articles." Source code for biology and medicine 7 (2012): 1-10.

[12] Sahu, Narendra, and Manoj Sonkusare. "A study on optical character recognition techniques." International Journal of Computational Science, Information Technology and Control Engineering 4.1 (2017): 01-15.

[13] Srivastava, Nitish, and Russ R. Salakhutdinov. "Multimodal learning with deep boltzmann machines." Advances in neural information processing systems 25 (2012).

[14] Tewani, Rachna et al. “An efficient extraction of information from Indian Government issued documents Aadhar and Pan Card.” Fusion: Practice and Applications (2021): n. pag.

[15] Thammarak, Karanrat, et al. "Comparative analysis of Tesseract and Google Cloud Vision for Thai vehicle registration certificate." International Journal of Electrical and Computer Engineering 12.2 (2022): 1849-1858.

[16] Vaithiyanathan, D., and Manigandan Muniraj. "Cloud based text extraction using google cloud vison for visually impaired applications." 2019 11th international conference on advanced computing (ICoAC). IEEE, 2019.

[17] Vijayarani, Dr S., and Ms A. Sakila. "Template Matching Technique for Searching Words in Document Images” in International Journal on Cybernetics & Informatics (IJCI) Vol. 4." (2015).

[18] Wiechork, Karina, and Andrea Schwertner Charao. "Automated Data Extraction from PDF Documents: Application to Large Sets of Educational Tests." ICEIS (1). 2021.

[19] Zheng, Lihong, et al. "An algorithm for accuracy enhancement of license plate recognition." Journal of computer and system sciences 79.2 (2013): 245-255.

APPENDIX 1 : DESIGN DOCUMENTS

CLASS DIAGRAM

The relationships between classes indicate how they interact with each other during the data extraction process. For example, CertificateImage is input to both TemplateMatching and OCR, and the output of OCR is input to DataRefinement. Finally, the refined data is serialized into JSON format and then reported in Excel files.

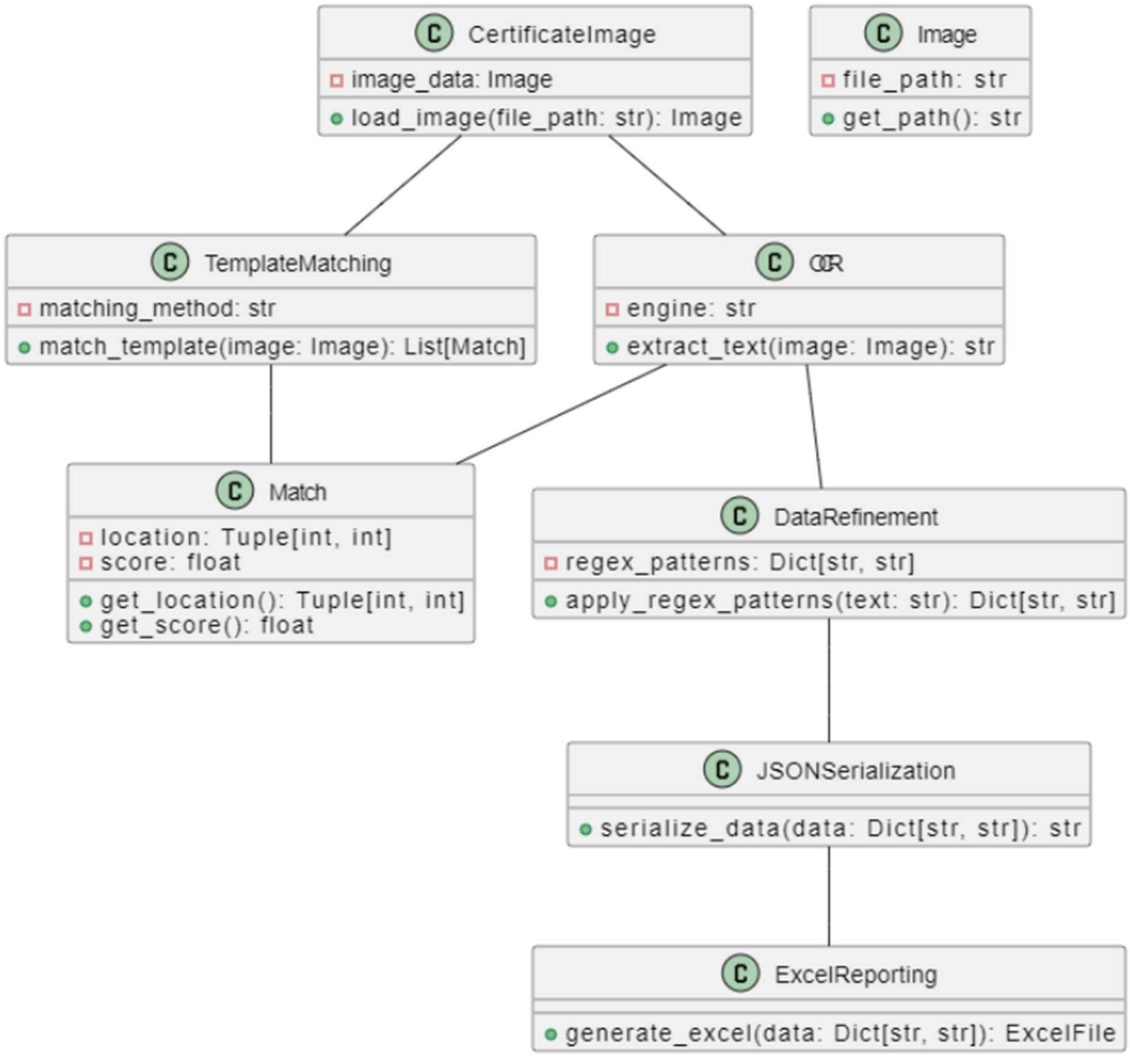

Fig. Class diagram

SEQUENTIAL DIAGRAM :

The sequence diagram outlines the interaction between the user and the automated data extraction system for educational certificates. It begins with the user providing a certificate image, which is processed through several stages including template matching, optical character recognition (OCR), data refinement, JSON serialization, and Excel reporting.

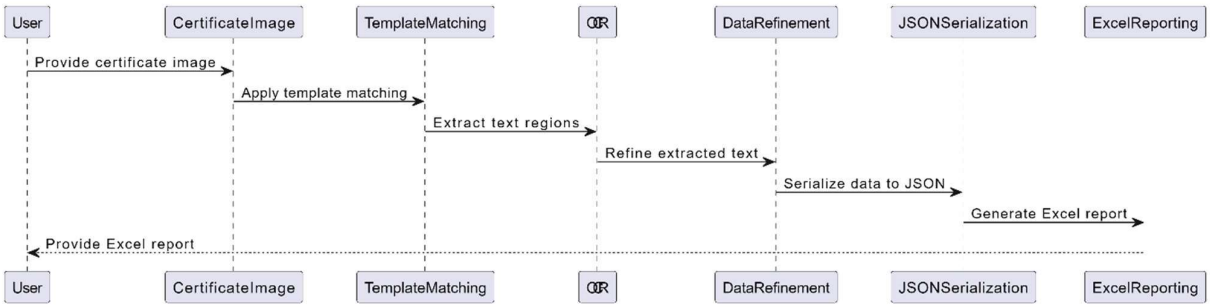

Fig. Sequential diagram

ACTIVITY DIAGRAM:

This activity diagram depicts the automated data extraction process from educational certificates. It begins with the user providing a certificate image for processing. The system then proceeds to process the image, utilizing template matching to identify regions of interest within the certificate. If regions are successfully found, the system applies Optical Character Recognition (OCR) to extract text from these regions. Subsequently, the extracted text undergoes data refinement, including cleaning and validation. If the data refinement is successful, the system serializes the refined data into JSON format. Finally, the system generates an Excel report based on the serialized data and provides it to the user. Throughout the process, decision points are included to handle potential errors and display error messages accordingly.

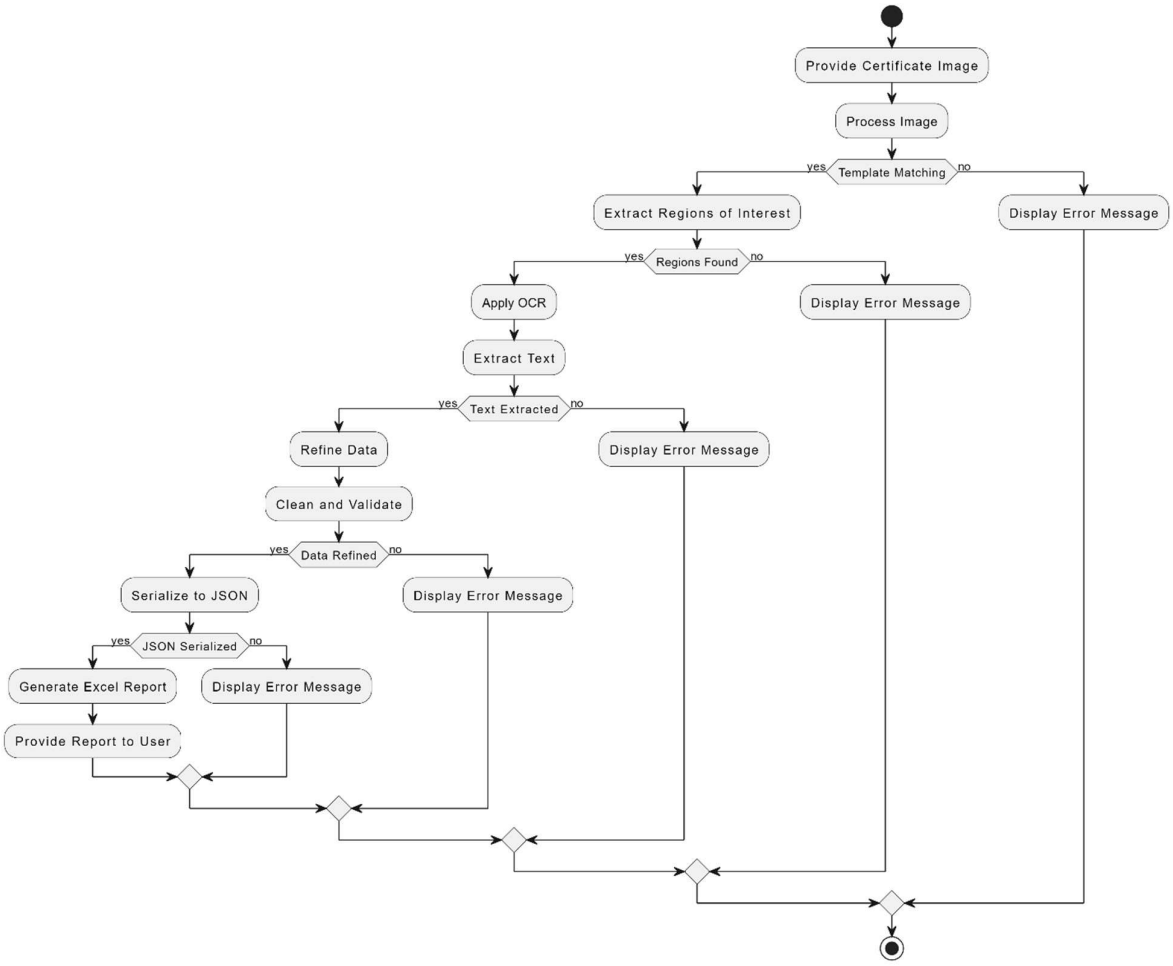

Fig. Activity Diagram

USE CASE DIAGRAM

The use case diagram illustrates the main functionalities of the “Certificate Information Extraction System.” Users, represented by the actor “User,” interact with the system to extract information from certificates. The system administrator, depicted as “System Admin,” manages the system’s overall operation.

Users can initiate the extraction process, which involves several steps such as data acquisition, text extraction, data refinement, and data serialization. Meanwhile, the system administrator has the authority to manage templates, including adding, removing, or updating them.

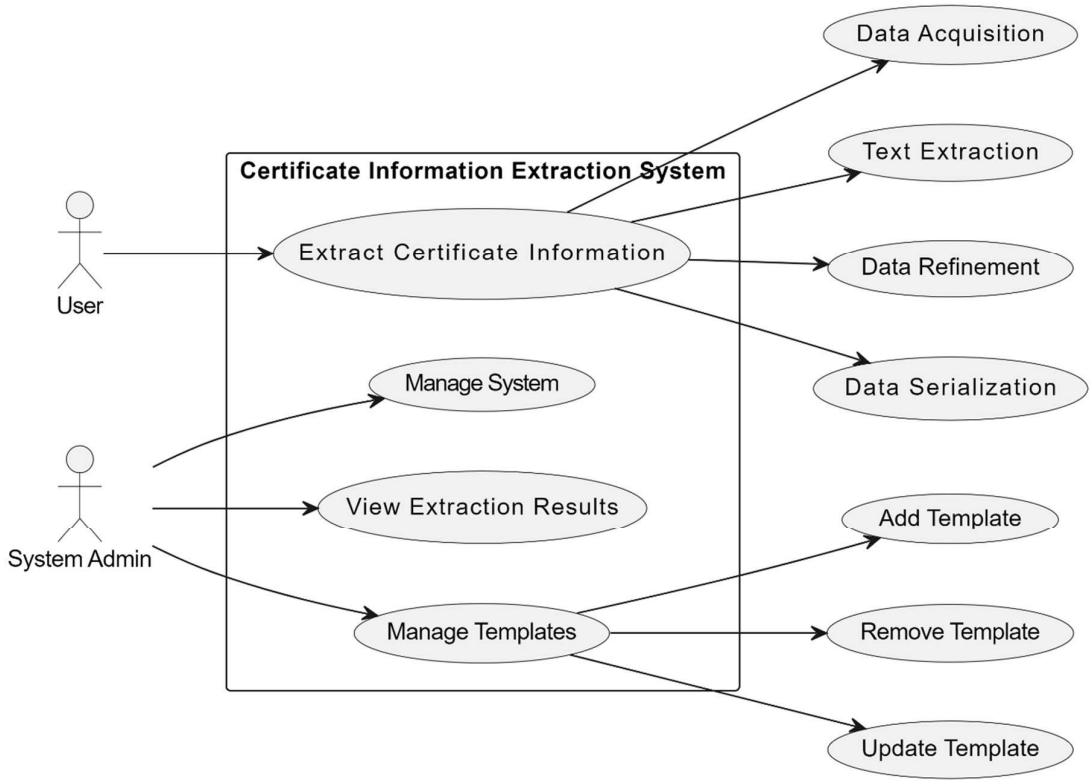

Fig. Use Case diagram

APPENDIX 2 : SAMPLE CODE

Scaling Factor in Template matching :

```
import cv2

import numpy as np

from google.colab.patches import cv2_imshow


# Load the certificate image

img = cv2.imread('/content/drive/MyDrive/one
certificate/FDP19EE15S11140130191049991.jpg', 0)


# Template image

template = cv2.imread('/content/drive/MyDrive/project extraction/templates/black and
white logo.png', 0)


# Define a range of scaling values to iterate over

scaling_values = [2.0]  # Add more values as needed


# Template matching methods

methods = [

  cv2.TM_CCOEFF,

  cv2.TM_CCOEFF_NORMED,

  cv2.TM_CCORR,

  cv2.TM_CCORR_NORMED,

  cv2.TM_SQDIFF,

  cv2.TM_SQDIFF_NORMED

]


# Iterate over each scaling value

for scaling in scaling_values:

  # Resize the template

  scaled_template = cv2.resize(template, (0, 0), fx=scaling, fy=scaling)
```

```
    # Initialize a flag to check if all methods predict the template perfectly
    all_perfect = True

    # Iterate over each template matching method
    for method in methods:

        img_copy = img.copy()
        # Perform template matching
        result = cv2.matchTemplate(img, scaled_template, method)
        min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
        score = max_val

        # Draw a rectangle around the matched region
        if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
            top_left = min_loc
        else:
            top_left = max_loc
        bottom_right = (top_left[0] + scaled_template.shape[1], top_left[1] +
scaled_template.shape[0])
        cv2.rectangle(img, top_left, bottom_right, (0,0,255), 5)

        # Display the image with the rectangle
        cv2_imshow(img)
        cv2.waitKey(0)
        bottom_right = (top_left[0] + scaled_template.shape[1], top_left[1] +
scaled_template.shape[0])
        cv2.rectangle(img, top_left, bottom_right,255, 5)
        cv2_imshow(img)
        cv2.waitKey(0)
        # Check if the score meets the threshold for perfect prediction
        if score != 1.0:
            all_perfect = False
```

```
            print('score:',score)
            print(f"Scaling value: {scaling}, Method: {method} - Template not aligned")
        else:
            print('score:',score)
            print(f"Scaling value: {scaling}, Method: {method} - Template aligned")

    # If all methods predict the template perfectly, output the scaling value and break out of the loop
    if all_perfect:
        print(f"Optimal scaling value: {scaling}")
        break  # No need to check other scaling values if one is found

cv2.destroyAllWindows()
```

Installations :

```
!pip install pytesseract
!sudo apt-get install tesseract-ocr-ind
!pip install pytesseract
!pip install nltk
#importing Librartes
import cv2
import re
import pytesseract
from pytesseract import Output
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image
from google.colab.patches import cv2_imshow
```

Importing :

```
import os
```

```
import shutil

def rename_and_move_images(input_folder, output_folder, prefix='image_',
start_index=1):
    # Make a copy of the input folder
    shutil.copytree(input_folder, output_folder)

    # List all files in the output folder
    files = os.listdir(output_folder)

    # Filter out only image files (you may adjust the condition as needed)
    image_files = [file for file in files if file.lower().endswith(('.png', '.jpg', '.jpeg', '.gif'))]

    # Sort the image files to ensure they are processed in order
    image_files.sort()

    # Rename files sequentially and move them to the output folder
    for i, filename in enumerate(image_files, start=start_index):
        # Generate the new filename
        new_filename = f"{prefix}{i:03d}{os.path.splitext(filename)[1]}"
        # Construct full paths for old and new filenames
        old_filepath = os.path.join(output_folder, filename)
        new_filepath = os.path.join(output_folder, new_filename)
        # Rename the file
        os.rename(old_filepath, new_filepath)
        print(f"Renamed '{filename}' to '{new_filename}'")

    return output_folder

# Example usage:
input_folder = '/content/drive/MyDrive/project extraction/ground truths/Ashutoshs final
work'
```

```
output_folder = '/content/drive/MyDrive/project extraction/ground truths/copy_Ashutoshs final work'

renamed_folder = rename_and_move_images(input_folder, output_folder)

print(f"Renamed images are stored in: {renamed_folder}")
```

Template Matching :

```
import os

import cv2


# Path to the folder containing certificate images

folder_path = '/content/drive/MyDrive/project extraction/ground truths/copy_Ashutoshs final work'


# Template image

template1 = cv2.imread('/content/drive/MyDrive/project extraction/templates/black and white logo.png', 0)

template2 = cv2.imread('/content/drive/MyDrive/project extraction/templates/ace logo certificates.png', 0)

# scaling = 1.78

template1 = cv2.resize(template1, (0, 0), fx=2.2, fy=2.2)

template2 = cv2.resize(template2, (0, 0), fx=2, fy=2)

h, w = template1.shape

H,W = template2.shape



template1_certificates = []

template2_certificates = []

unidentified_templates=[]


# Template matching method

method = cv2.TM_CCOEFF
```

```
# Threshold for considering a match
threshold_min_template1 = 400000000.0
threshold_max_template1 = 600000000.0
threshold_min_template2 = 70000000.0
threshold_max_template2 = 80000000.0

# Process each certificate image in the folder
for filename in os.listdir(folder_path):
    # Construct full file path
    filepath = os.path.join(folder_path, filename)

    # Check if the file is an image
    if os.path.isfile(filepath) and any(filepath.lower().endswith(ext) for ext in ['.jpg', '.jpeg',
'.png']):
        # Load certificate image
        img = cv2.imread(filepath, 0)

        # Perform template matching
        result1 = cv2.matchTemplate(img, template1, method)
        result2 = cv2.matchTemplate(img, template2, method)

        min_val_1, max_val_1, min_loc_1, max_loc_1 = cv2.minMaxLoc(result1)
        score1 = max_val_1
        min_val_2, max_val_2, min_loc_2, max_loc_2 = cv2.minMaxLoc(result2)
        score2 = max_val_2


        # Check if the score meets the threshold
        if threshold_min_template1 <= score1 <= threshold_max_template1 :
            print(f"Template matched with score: {score1} in {filename}")
            location = max_loc_1
```

```
            bottom_right = (location[0] + w, location[1] + h)
            # cv2.rectangle(img, location, bottom_right, (0, 0, 255), 5)
            template1_certificates.append(filepath)
            # cv2_imshow(img)
            # cv2.waitKey(0)
            cv2.destroyAllWindows()
        elif threshold_min_template2 <= score2 <= threshold_max_template2 :
            print(f"Template matched with score: {score2} in {filename}")
            location = max_loc_2
            bottom_right = (location[0] + W, location[1] + H)
            template2_certificates.append(filepath)
            # cv2.rectangle(img, location, bottom_right, (0, 0, 255), 5)
            # cv2_imshow(img)
            # cv2.waitKey(0)
            cv2.destroyAllWindows()
        else:
            print(f"Template not matched in {filename}")
            unidentified_templates.append(filepath)
# print(template1_certificates,template2_certificates,unidentified_templates)
```

Processing and extraction :

```
##########################FDP CERTIFICATES

import pandas as pd

# Global DataFrame to store all processed data
all_data_df = pd.DataFrame()

# Define a function to process each list of file paths
def process_file_paths(paths):
    global all_data_df
```

```
    #print(paths)
    # Iterate over each file path
    for i in paths:
        #print(i)
        #extraction of the data
        text=extract_the_data(i)
        #print("text:",text,end='\n')
        # Apply the corresponding NLP code to process the certificate
        processed_data = apply_nlp_code(text)
        #print("preprocessed_data:",processed_data,end='\n')
        # Append processed data to DataFrame
        append_dataframe(processed_data)

# Placeholder function for applying NLP code
def extract_the_data(file_path):
    # Placeholder code to simulate NLP processing
    #print("file_path:",file_path)
    img = cv2.imread(file_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    th, threshed = cv2.threshold(gray, 127, 255, cv2.THRESH_TRUNC)
    custom_config = r'--oem 3 --psm 6 '
    extracted_text=pytesseract.image_to_string(threshed, config=custom_config)
    extracted_text = re.sub(r'\s+', ' ', extracted_text)
    return extracted_text

def apply_nlp_code(extracted_text):
    # Regular expressions to match required patterns
    name_pattern = r"awarded to (.+?) for"
    score_pattern = r"consolidated score of (\d+)"
    roll_pattern = r"Roll No:(\w+)"
    institute_pattern = r"IIT (\w+)"
```

```
    #credits_pattern = r"recommended by NPTEL: (\d+|\d+\sor\s\d+)"
    certification_pattern = r"NPTEL"
    course_pattern = r"completing the course (.+?) with"
    #date_pattern = r"NPTEL Coordinator \((.*?)\) Advisor"
    date_pattern = r'(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)-
(?:Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)\s\d{4}'

    # Extracting information using regular expressions
    name_match = re.search(name_pattern, extracted_text)
    score_match = re.search(score_pattern, extracted_text)
    roll_match = re.search(roll_pattern, extracted_text)
    institute_match = re.search(institute_pattern, extracted_text)
    #credits_match = re.search(credits_pattern, extracted_text)
    certification_match = re.search(certification_pattern, extracted_text)
    course_match = re.search(course_pattern, extracted_text)
    date_match = re.search(date_pattern, extracted_text)

    # Retrieving matched groups
    name = name_match.group(1).strip() if name_match else None
    score = score_match.group(1) if score_match else None
    roll_number = roll_match.group(1) if roll_match else None
    institute = "IIT " + institute_match.group(1) if institute_match else None
    #credits = credits_match.group(1) if credits_match else None
    certification = certification_match.group(0).strip() if certification_match else None
    course = course_match.group(1).strip() if course_match else None
    # Retrieving matched groups
    # if date_match:
    #     date = date_match.group(1).strip()
    # else:
    #     date=None
    if date_match:
```

```
    date = date_match.group(0)
  else:
    date = None
  ################################
  if name:
    # Regular expression pattern to extract only the sequences of capital letters from the
name
    capital_letters_pattern = r"[A-Z]+"

    # Find all sequences of capital letters in the extracted name
    capital_letters = re.findall(capital_letters_pattern, name)

     # Join the capital letters to form a string
    capital_letters_string = ' '.join(capital_letters)
  processed_data = {
  "Name": [capital_letters_string],
  "Score": [score],
  "Roll Number": [roll_number],
  "Institute": [institute],
  #"Number of Credits Recommended": [credits],
  "Name of the certification": [certification],
  "Course completed": [course],
  "Completion date": [date]
  }
  return processed_data

# Define a function to append processed data to DataFrame
def append_dataframe(processed_data):
  global all_data_df
  df = pd.DataFrame(processed_data)
  all_data_df = all_data_df.append(df, ignore_index=True)
```

```
# Process each list of file paths
process_file_paths(template1_certificates)
```

WER and CER evaluation :

```
#FDP NPTEL METRICS

import pandas as pd
import nltk

# Load the ground truth and extracted data into Pandas DataFrames
ground_truth_df = pd.read_excel("/content/drive/MyDrive/project extraction/ground truths/NPTEL.xlsx")
extracted_df = pd.read_excel("/content/output.xlsx")

#ground_truth_df.rename(columns={'Score %': 'Score'}, inplace=True)
ground_truth_df.rename(columns={'Name of certification': 'Name of the certification'}, inplace=True)
ground_truth_df.rename(columns={'Course Completed': 'Course completed'}, inplace=True)



# Define function to clean the strings
# def clean_string(text):
#     return text.strip().lower()
def clean_string(text):
    if isinstance(text, str):
        return text.strip().lower()
    else:
        return text
```

```
# Apply cleaning function to relevant columns
columns_to_compare = ["Name", "Roll Number", "Institute", "Name of the certification", "Course completed", "Completion date"]

for col in columns_to_compare:
    ground_truth_df[col] = ground_truth_df[col].apply(clean_string)
    extracted_df[col] = extracted_df[col].apply(clean_string)

# Define functions to calculate WER and CER
def wer(hypothesis, reference):
    if isinstance(hypothesis, str) and isinstance(reference, str):
        return nltk.edit_distance(hypothesis.split(), reference.split())
    else:
        return 0  # Return 0 if either hypothesis or reference is not a string

def cer(hypothesis, reference):
    if isinstance(hypothesis, str) and isinstance(reference, str):
        return nltk.edit_distance(hypothesis, reference)
    else:
        return 0  # Return 0 if either hypothesis or reference is not a string


# Calculate WER and CER for each pair of ground truth and extracted strings
wer_scores = {col: [] for col in columns_to_compare}
cer_scores = {col: [] for col in columns_to_compare}

min_len = min(len(ground_truth_df), len(extracted_df))

for col in columns_to_compare:
    for idx in range(min_len):  # Iterate over the minimum length
        hypothesis = extracted_df.loc[idx, col] if idx < len(extracted_df) else ""  # Handle if index exceeds length
```

```
    reference = ground_truth_df.loc[idx, col] if idx < len(ground_truth_df) else ""  # Handle if index exceeds length
    wer_scores[col].append(wer(hypothesis, reference))
    cer_scores[col].append(cer(hypothesis, reference))

# Compute average WER and CER
avg_wer = {col: sum(scores) / len(scores) for col, scores in wer_scores.items()}
avg_cer = {col: sum(scores) / len(scores) for col, scores in cer_scores.items()}

print("Average Word Error Rate (WER):", avg_wer)
print("Average Character Error Rate (CER):", avg_cer)
```

APPENDIX 3 : PLAGARISM REPORT